iWorkshop Node.js e IBM i

Strumenti per sviluppatori

By Davide Raschetti (Faq400.com)

Agenda workshop

- IBM i e Open Source (OPS5733)
  - Cosa è
  - Utilizzo shell e strumenti workshop
- Node js cosa è perchè utilizzarlo
  - Javascript
  - Nodejs
  - ‘Hello word’
  - Nodejs Database
  - System Call
  - Socket
- Dashboard IBMi
  - Esempio utilizzo git

Open Source and IBM i

IBM developerWorks®

IBM i open source PASE (Portable Application Solutions Environment) porting di applicazioni AIX/UNIX su IBM i.

IBM i technology updates http://www.ibm.com/….

Shell

Start openssh su IBMi : è un CL da schedulare all'avvio!

(STRTCPSVR SERVER(*SSHD))

Shell default di IBM molto povera → 5733-OPS Option 7 TOOL contiene **bash** con history, autocomplete, ecc ecc :
Altre info in tools : istruzioni per rendere attiva la shell **bash**

Comandi : **`ls -l, mkdir, cd, uname -a, cat, chroot, git`**

Link : client ssh putty, interessante cosa si può fare con ssh e shell

GIT GCC

GIT : posto dove tenere dettaglio storico codice sorgente e condividerlo è Free OpenSource.
Un repository git può contenere qualsiasi tipo di file. E’ possibile clonare un repository ed utilizzarne il contenuto. Git_guide

GCC : Compilatore C per porting pacchetti linux su IBMi.

CHROOT

Contiene 2 script:

- **chrooot_setup.sh** : crea una ‘root’ sicura dove non possiamo fare danni, non crea solo una cartella ma una nuova ‘/’ (root) senza comandi sistema
- **pkg_setup.sh** : permette di installare RPM (metodo di distribuzione e installazione di pacchetti linux) Esempio:

```
bash-4.3$ cd /QOpenSys/QIBM/ProdData/OPS/GCC
bash-4.3$ ./pkg_setup.sh -help
bash-4.3$ ./pkg_setup.sh pkg_perzl_joe-3.7-1.lst
```

Editor per shell

Altre info : [Chroot Scripts]

Open Source RPM's (Technology Preview)

Da Marzo 2018 esiste possibilità da livello sistema 7.2 di utilizzare RPM con YUM.
Molti pacchetti 5733OPS e tantissimi nuovi.
Sono distribuiti tutti in 'beta' compreso YUM.
Troviamo versione Node.js v8, Python 3.6, GCC 6.3.3 e moltissimo altro.
Esempi: (install, remove search, list installed, list avalable ….)

```
bash-4.3$ yum install <package>
bash-4.3$ yum list avalable
```

Open Source RMP :https://www.ibm.com/……
RPM Getting started https://www.ibm.com/…..

Convenzioni grafiche

Shell command

```
echo 'un comando shell'
comando shell ←  OUTPUT COMANDO
uname
OS400  ← ALMENO QUI!!!!!
```

Javascript

```
//File app/index.js
console.log('questo è javascript')
'number' ←  output js
```

Litmis Spaces

LITMIS SPACES

KRENGELTECH

Aaron Bartell
IBMi hosting

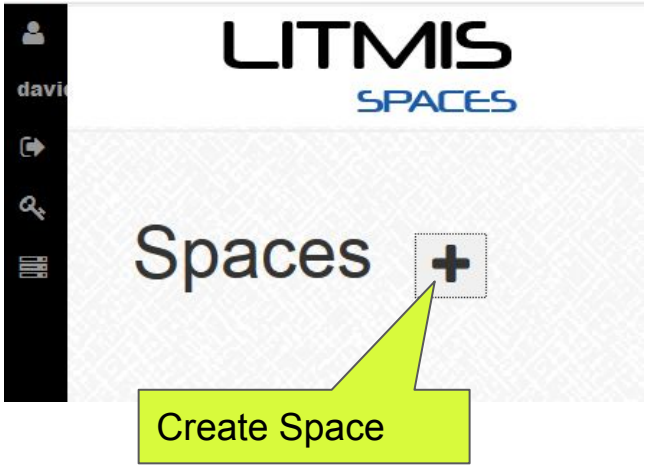
Requisiti : un account google, github o linkedin

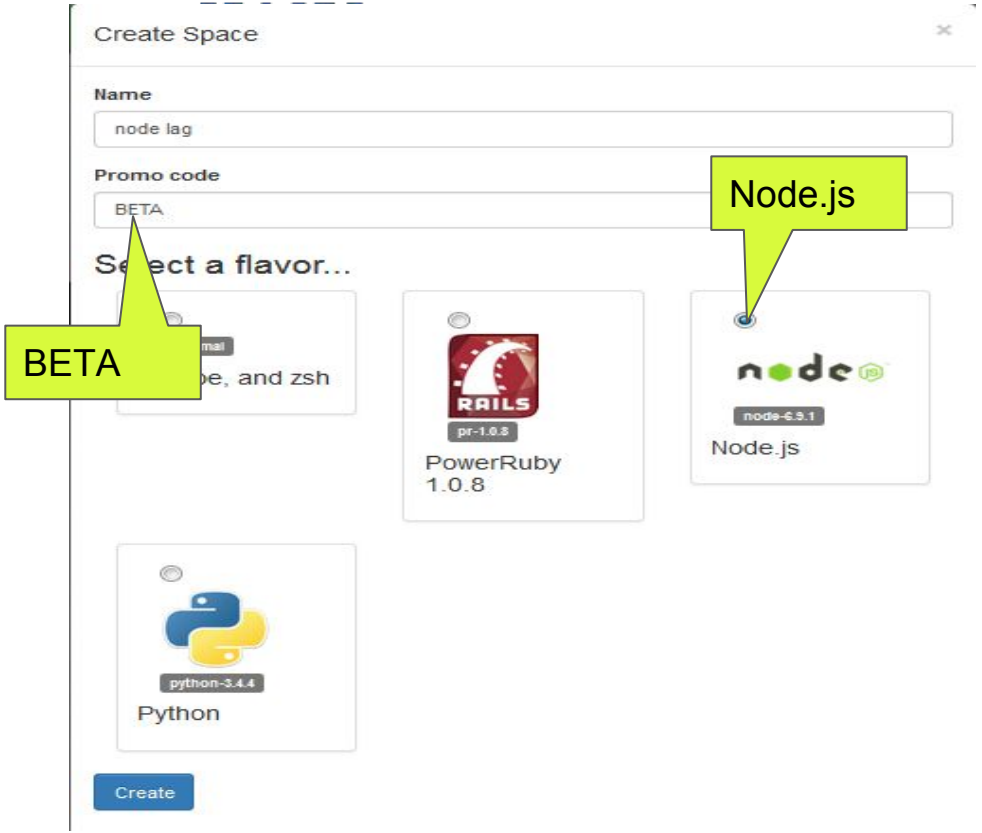
https://spaces.litmis.com/auth/signin

Strumenti Corso

LITMIS

KRENGELTECH





Strumenti Corso

LITMIS SPACES

KRENGELTECH

Git Corso

LITMIS SPACES

KRENGELTECH

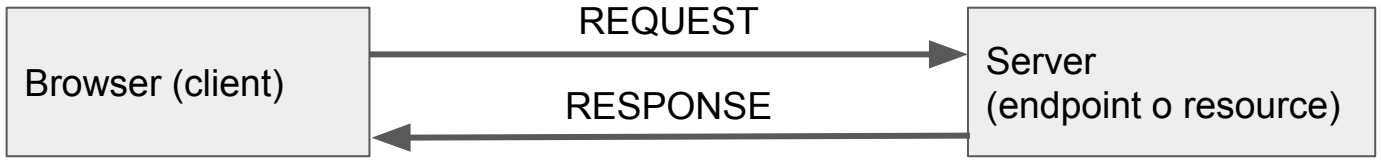
Cloniamo progetto nodejs-Workshop contenente tutti i sorgenti che utilizzeremo nel corso.

```
git config --global http.sslVerify false
git clone https://github.com/DlcF4/nodejs-Workshop.git
ls -l
drwx--S---  …..   8192 Dec  2 17:24 nodejs-Workshop
```

Le ‘app’ contenute non sono immediatamente ‘portabili’ su altro sistema, ma alla fine del corso potrete usarle sui vostri sistemi

INCOLLA :CTRL+SHIFT+V

HTML cenni



Request : url,method… Header : content-type, content-length, ….
GET : tutti i dati (o quasi) nell'url. Limite **256** caratteri!!!!!
POST/PUT/... : il client manda i dati nel request payload no limiti dimensioni

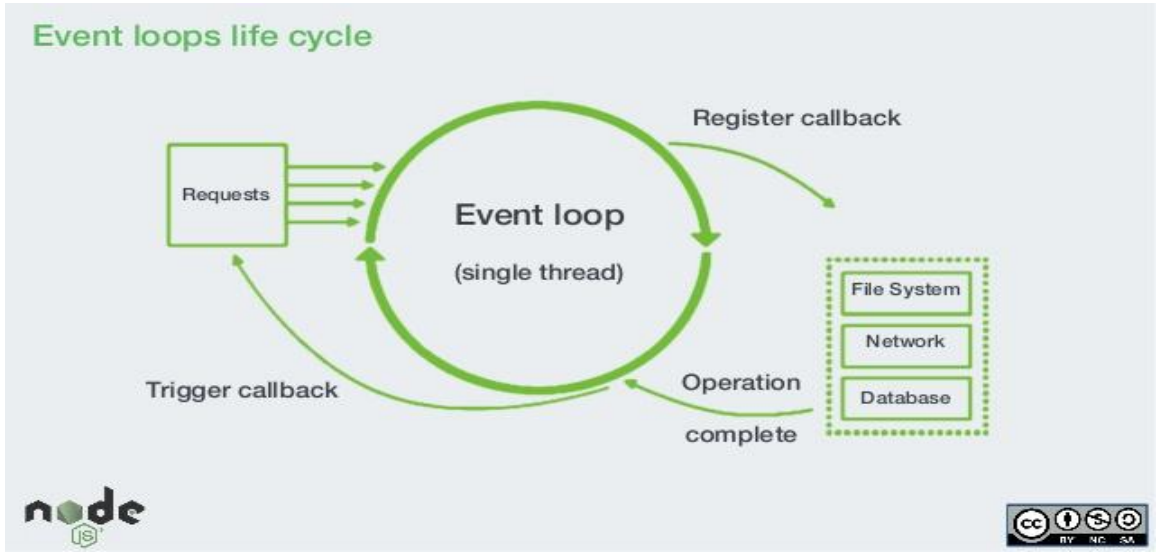
Response :Header Status, content/type ...
Body : contenuto risposta server (HTML,XML,FILE,JSON…..)

Su IBMi **Apache** e **Ngnix** (5733-OPS Option 11)

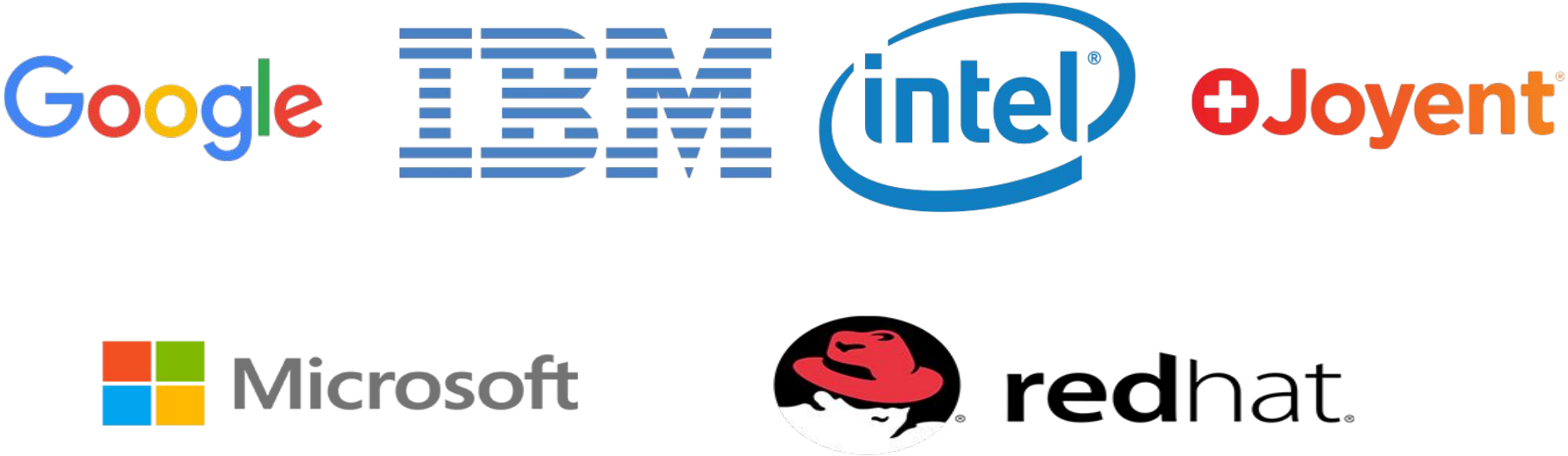
www.w3c.com

Node.js

Node.js è una piattaforma di sviluppo, creata nel 2009, basata sul motore Javascript V8 di Google. Implementa un modello I/O non bloccante, event-driven e asincrono



Node.js Fundation Platinum Sponsor



Node.js Perchè?

Tempi di sviluppo ridotti

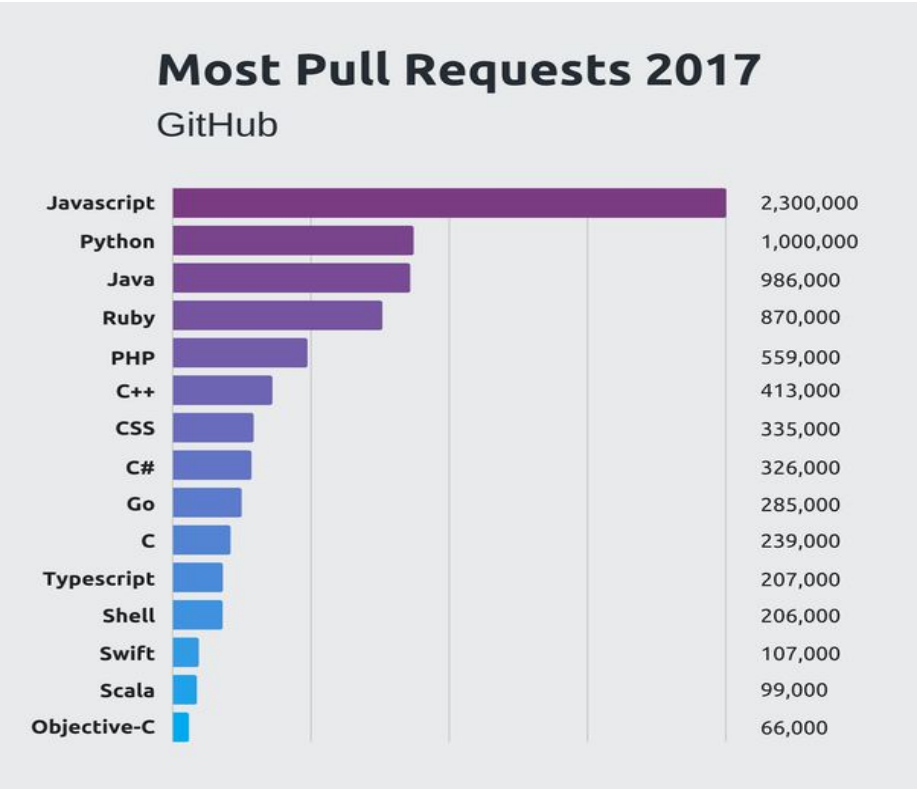
Costi di infrastruttura contenuti

Performante per realizzare applicazioni real-time

Javascript lato server e lato client

Numeri : **1.960** contributors, **47.642** star ghitub, **17.400.000** risultati Google, **221.520** (**74.968** Nov 2017) questions StackOverflow
QUORA

Node.js Perchè?



Filosofia rispetto altri linguaggi:

- RPG JAVA ecc cercano di includere tutto quello che serve
- Node.js include il minimo
  - piccoli moduli che fanno una sola cosa e bene
  - moduli che lavorano assieme ad altri moduli
  - moduli che gestiscono stream ed eventi
- Node.js un ecosistema unico

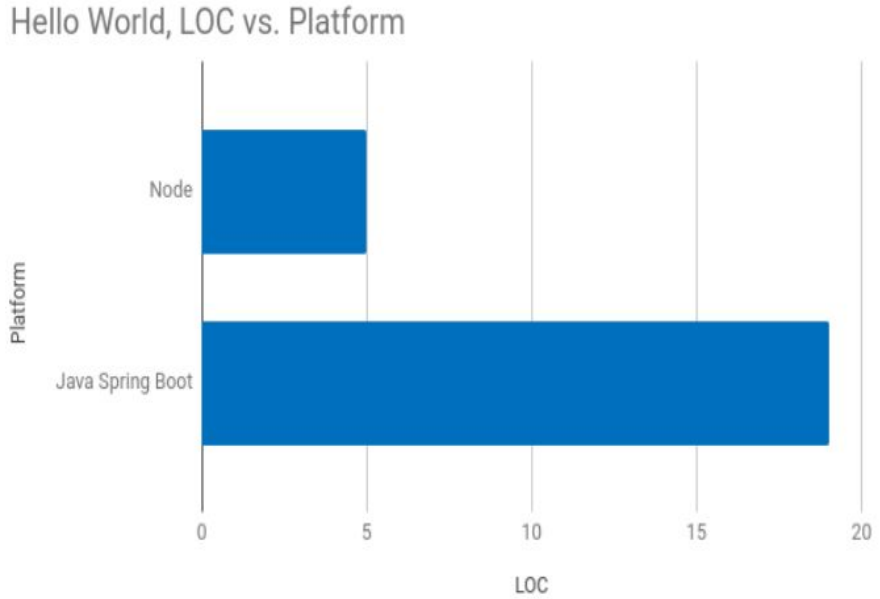
Java Spring Boot Hello World

```java
package hello;

import org.springframework.web.bind.annotation.RestController;
import org.springframework.web.bind.annotation.RequestMapping;

@RestController
public class HelloController {
    @RequestMapping("/")
    public String index() {
        return "Greetings from Spring Boot!";
    }
}
```

```java
package hello;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class Application {
    public static void main(String[] args) {
        SpringApplication.run(Application.class, args);
    }
}
```

Node.js Hello World

```javascript
const app = require('express')()

app.get('/', (req, res) => res.send('Hello World from Node.js!'))

app.listen(3000, () => console.log('Listening on 3000!'))
```



Javascript Client e Server

```
$(‘#mybutton’).html(‘Click me!’);                    //jQuery
$(‘#mybutton’).on(‘click’ ,function(){
        $(‘#mybutton’).html(‘Done!!!!’);
});
```

```
http.createServer(function(req, res) {                    //Node.js
  res.writeHead(200, {'Content-Type': 'text/plain'})
  res.end('Hello World')
}).listen(port, '0.0.0.0')
```

Node.js Quando?

API RESTful per applicazioni web

Applicazioni e strumenti di sviluppo

Uso avanzato degli stream

Applicazioni scalabili

Node.js quando NON usarlo?

Calcoli pesanti e lunghi

Quando si perde la caratteristica non bloccante

La parte di calcolo supera la parte di I/O

Per rimpiazzare il software già esistente

Servire file statici

Node.js

Versioni : SemVer MAJOR.MINOR.PATCH

MAJOR version when you make incompatible API changes,

MINOR version when you add functionality in a backwards-compatible manner

PATCH version when you make backwards-compatible bug fixes.

Node.js Storia

Nato nel 2009 fork progetto nel 2015 io.js. Riunione in unico progetto spiega salto di versioni da 0.x.y a 4.x.y Versioni LTS Node release

Da Marzo su IBM i Con Open source RPM's (Tecnology Preview)
8.9.1 'Carbon' rilasciata 31/10/2017 - End Dicembre 2019

Su IBM i 5733-OPS Option 10
6.9.1 'Boron' rilasciata 18/10/2016 - End Aprile 2019 (99% ES6)

ES6 engine Javascript V8 compatibility

Javascript variabili

```
var a,b;
a='String';
console.log(typeof a); // > string
b=1;
console.log(typeof b); // > number
b=true;       //boolean
console.log(typeof b); // > boolean
a=[];
console.log(typeof a); // > object
```

Primitivi

- Numbers
- String
- Boolean

Strutturati

- Function
- Object (array)

LITMIS
SPACES

Node.js command line REPL

REPL : Read, Eval, Print, and Loop Console interattiva
Digitare **node** per lanciarla
.exit per uscire.
Posso scrivere o copiare codice e interrogare oggetti per capirne il funzionamento.

```
var a =4;
typeof a;
'number'
var a ='yty';
typeof a;
'string'
var a =[1,2,3,4]
typeof a;
'object'
```

```
a[0];
1
a[5];
undefined
a.push('fr');
5
a;
[ 1, 2, 3, 4, 'fr' ]

a.splice(4,1);
```

```
a={'nome':'Mario','cognome':'Rossi','indirizzo':'Via Verdi, 25'};
{ nome: 'Mario', cognome: 'Rossi', indirizzo: 'Via Verdi, 25' }
a.indirizzo;
'Via Verdi, 25'
```

LITMIS
SPACES

Node.js Hello World

Creiamo un file server.js

Da shell **node server.js**

http://spaces.litmis.com:<port>/

```
node server.js
Server runnig at
http://spaces.litmis:63843
crtl+c for exit
```

```
//File server.js
var http = require('http');
var port = process.env.LITMIS_PORT_DEVELOPMENT;

http.createServer(function(req, res) {
  res.writeHead(200, {'Content-Type':
'text/plain'});
  res.end('Hello World');
}).listen(port, '0.0.0.0');
console.log('Server running at
http://spaces.litmis.com:%d crtl+c for exit',
port);
```

Javascript function

LITMIS SPACES

```
//normal
function func1(a,b){
return a+b;
};
var result = func1(5,3);
//anonymus
var func1 = function(a,b){
   return func1(a,b);
};
var result = func1(5,3);
```

Da notare :

- Una variabile può essere una funzione
- I parametri a e b vivono solo nelle funzione
- Simili a subprocedure RPG

Asincrono

LITMIS SPACES

```
// File sayHello.js
console.log("Hello.");
// Say "Goodbye" two seconds from now.
setTimeout(function() {
            console.log("Goodbye!");
            }, 2000 );
// Say "Hello again!"
console.log("Hello again!");
```

`setTimeout` funzione javascript che ritarda esecuzione per intervallo prestabilito

Output:

> Hello

> Hello again!

> Goodbye!

```
cd javascript
node sayHello.js
.....
```

LITMIS SPACES

Javascript self-invoked

```
var func1= (function () {
    // private
    var counter = 0;
    console.log('Module init '+counter);
    return {
        add: function () {
            return counter += 1;
        },
        del: function () {
            return counter -= 1;
        }
    }
})();
```

Viene ‘eseguita’ una sola volta grazie a **();** alla fine
Ho variabili **private** : possibile modificare `counter` solo richiamando `funct1.add()` o `funct1.del()`
w3cschool

```
node selfInvoked.js
.....
```

Callback

```
//File javascript/readFile.js
var fs = require('fs');
fs.readFile('dati.txt',
"utf8",function(err,data){
    console.log('-- Callback --');
    if(err) {console.log('Error reading file
%s',err);}
    else    {console.log(data);}
});
```

Funziona come parametro.
Usate per chiamate **async** : che richiedono tempo.
Quando termina I/O del file viene eseguito il codice della callback.
Permette ‘sincronismo’.
callbackhell.com

```
node readFile.js
.....
```

Promise : alternativa alla callback

```
fs.readFileAsync = function (filename) {
    return new Promise(function (resolve, reject) {
        try {fs.readFile(filename, function(err, buffer){
                console.log('-- Promise --');
                if (err) reject(err); else
resolve(buffer);
        });
        } catch (err) {reject(err);}
    });
};
fs.readFileAsync('dati.txt', "utf8").then(function(data) {
   console.log('Data %s',data);}).catch(function (err){
console.log('Error reading data});
```

Aggiungiamo a readFile.js
L'oggetto Promise evita annidamento callback su chiamate asincrone consecutive.

Oggetto ES6 deve ritornare resolve o reject

AWAIT da node 8

```
var rf = await
fs.readFile('./dati.txt')
```

LITMIS
SPACES

Node.js moduli

```
//File javascript/module/funct.js
function func1(p1, p2){
  return p1 + p2
}
exports.func1 = func1
```

```
//File javascript/app.js
var f = require('./modules/funct')
var result = f.func1(2, 3)
console.log('result:' + result)
```

Esempio :
file funct.js ha una funzione e con **export** la rendo disponibile
File app.js recupero il mio modulo con un **require**

```
node app.js
result : 5
```

Node Package Manager : npm

E’ già installato con Node.js

https://www.npmjs.com/

% **npm help**

% **npm init**

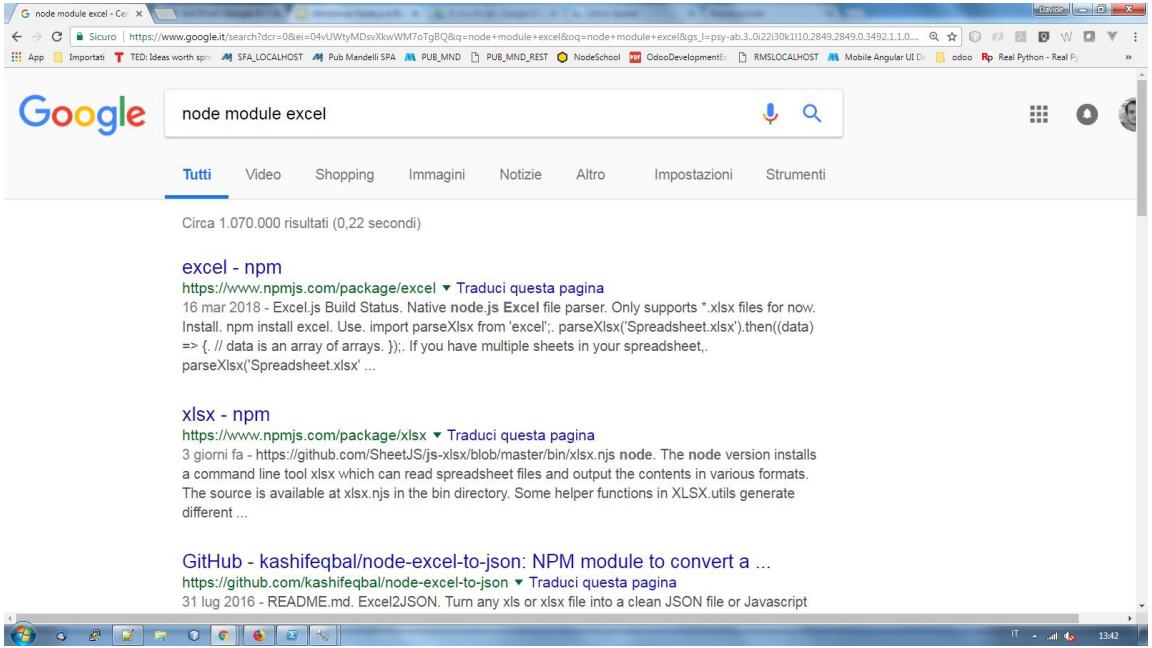
Crea file **package.json** che racchiude tutte le dipendenza (altri moduli) necessari alla mia applicazione nella cartella **node_modules**

installiamo il package express (--save aggiung il package all’elenco in **package.json**)

```
mkdir app
cd app
npm init
….Domande Applicazione
cat package.json
….
npm install express --save
cat package.json
……
ls ….
```

Trovare i moduli

Google npm module <nome_modulo>



Express.js

E’ il più famoso web framework per Node.js
Fornisce una serie di utilità routing, middleware, template, error handling, ecc ecc
http://expressjs.com/
Generatori : impostano struttura app
express-generator

```
//File app/index1_helloworld.js
var express = require('express')
var app = express()

app.get('/', function(req, res) {
  res.send('Hello World! (Express)');
})

var port = process.env.PORT ||
process.env.LITMIS_PORT_DEVELOPMENT
app.listen(port, function() {
  console.log('Running on port %d', port)
})
```

LITMIS
SPACES

Connessione al DB2

Non serve npm: il modulo è già presente sulla nostra macchina.
Nel nostro esempio copiamo file nodejs_workshop/app/db.js nella nostra cartella app

```
node db.js
.........
```

[DB2 driver and toolkit](#) [API Manual](#)

```
//Esempio utilizzo
const db =
require('/QOpenSys/QIBM/ProdData/OPS/
Node6/os400/db2i/lib/db2a')

const dbconn = new db.dbconn()
dbconn.conn("*LOCAL")
const stmt = new db.dbstmt(dbconn)

stmt.exec(sql, function(result,err){
.................
});
```

LITMIS
SPACES

Query DB

```
//File app/indexQuery.js
….
app.get('/', function(req, res) {
  stmt.exec(`SELECT * FROM
${schema}.CUSTOMER`, function(results,
err) {
   res.json(results)
  })
})
….
```

Creiamo file app/index.js
Copiamo contenuto nodejs_workshop/app/index_query.js

```
node index.js
Server runnig at
http://spaces.litmis:<port>
crtl+c for exit
```

Template per HTML

Esistono template che permettono a Node.js di produrre anche il livello presentazione, tuttavia per questa esigenza esistono anche framework lato ‘client’ molto utilizzati e potenti (es Angular.js, React.js, Vue.js ecc ecc).

In questi casi Node si ‘limita’ a generare JSON che poi viene elaborato e presentato nei browser dai vari framework.

Nel nostro esempio utilizziamo un framework Node.js chiamato pug che partendo da una sua sintassi particolare genera HTML dinamico.

Template per HTML

```
npm install pug --save
mkdirs view && cd view
```

Copio index3_view.js app.set

```
//File app/index.js
app.set('views', __dirname + '/views')
app.set('view engine', 'pug')
app.get('/', function(req, res) {
 res.render('index', { title: 'Node.js
Workshop', message: 'Hey I\'m on IBMi!'})
})
```

Nuovo File index.pug

```
//File app/view/index.pug
html
  head
    title!= title
  body
    h1!= message
```

```
cd..
node index.js
..http://spaces.litmis:.
```

Vista Lista (customers.pug + modifica index)

```
//File view/customers.pug
h1=title
  table
    thead
      tr
        th Last Name
        th Customer Number
    tbody
      each row in results
        tr
          td=row.LSTNAM
          td=row.CUSNUM
```

```
//File app/index.js
app.get('/customers', function(req, res) {
  stmt.exec(`SELECT LSTNAM, CUSNUM FROM
${schema}.CUSTOMER`, function(results) {
    res.render('customers', { title:
'Customers', results: results})
  })
})
```

```
cd..
node index.js
```

Vista dettaglio (customer.pug + modifica index)

```
//Edit File view/customers.pug
...
        td=a(href=`/customer/${row.CUSNUM}`)=row.CUSNUM
```

```
//File app/index.js
app.get('/customer/:id', function(req, res) {
  var sql = `SELECT * FROM ${schema}.CUSTOMER WHERE
CUSNUM=` + req.params.id;
  stmt.exec(sql, function(result, err) {
    res.render('customer', { title: 'Customer', result:
result[0]})
  });
});
```

- Pug href modifica colonna CUSTID
- Node leggere parametri in url

File customer.pug in nodejs_workshop/app/view/

```
cd..
node index.js
```

CRUD Create Read Update Delete

```
pwd
/home/USRGM8CO/app
cd view
mkdir customers
```

copy paste da sorgenti…
../view/customer/index.pug
../view/customer/show.pug

Creo nuove viste pug
new.pug , edit.pug e _form.pug
_<nome> convenzione per contenuto parziale

```
//File app/view/customers/new.pug
h1 New Customer
include _form ← COME /COPY e non serve.pug
a(href='/customers') Back
```

Npm Body Parser (POST data in JSON)

```
cd
cd app
npm install body-parser --save
……...
```

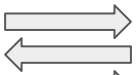
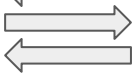
permette di leggere comodamente body delle request:
req.body.<Nome>

```
//File app/index.js
…….
var express = require('express'),
    bodyParser = require('body-parser');

var app = express();
app.use(bodyParser.json())
   .use(bodyParser.urlencoded({ extended:
false }));
…….
```

Web Socket

Connessione FULL-DUPLEX attraverso una singola connessione TCP

Handsake http: o https:// poi protocollo ws:// o wss://
Perchè : con socket il sever può parlare per primo e a molti

HTTP : A:request  B:response
A:request  B:response
……..

Websocket : A:request  B:response
A  B

Web Socket

```
cd ~
mkdir websock
cd websock
npm init
npm install express --save
npm install socket.io --save
touch index.js index.html
……...
```

```
//File app/index.js
…….
var express = require('express'),
    bodyParser = require('body-parser');

var app = express();
app.use(bodyParser.json())
   .use(bodyParser.urlencoded({ extended:
false }));
…….
```

System call

Dalla shell posso lanciare comandi di sistema Parola chiave system
| pipe output
grep <regular expression filter>

```
system WRKACTJOB

.........

system WRKACTJOB | grep DAVIDE

.......
```

Oggetti Nativi

IBMi Native Object

Accesso a Oggetti nativi, profili utente, code, Comandi e Programmi RPG

```
cd /QOpenSys/QIBM/ProdData/OPS/Node6/os400/xstoolkit/lib/
ls -l
-rw-r--r--    1 qsys     0             5666 Oct 25 17:28 idataq.js
-rw-r--r--    1 qsys     0            12866 Oct 25 17:28 inetwork.js
-rw-r--r--    1 qsys     0            32950 Oct 25 17:28 iobj.js
.....
```

Utilizza XML services for IBMi (PTF per 7.1 e 7.2) Documentazione

iToolkit.js

Questa libreria permette di eseguire programmi in QSH, ILE o comandi.
Nel materiale del corso si trova cartella itoolkit con 3 esempi chiamate CL
e una chiamata ad un programma RPG.

Tips

Start openssh su IBMi

```
(STRTCPSVR SERVER(*SSHD))
```

Start Node in production on IBMi

```
SBMJOB CMD(QSH CMD('/www/myapp/start.sh'))
```

start.sh esegue comandi shell vedi materiale corso.

Utilizzare materiale corso su altri sistemi 1

```
git clone https://github.com/DlcF4/nodejs-Workshop.git
cd nodejs-Workshop/app
npm install
cd ../websocket
npm install
cd ../itoolkit
npm install
```

Dopo aver scaricato sorgenti recuperiamo **dipendenze** descritte in package.json con npm per entrambe le ‘app’

Utilizzare materiale corso su altri sistemi 2

Variabili d'ambiente: nel codice utilizziamo 2 variabili legate all'ENV:

```
const schema = process.env.LITMIS_SCHEMA_DEVELOPMENT
var port = process.env.LITMIS_PORT_DEVELOPMENT
```

Per poter utilizzare il codice è sufficente ricreare le variabili d'ambiente oppure modificare il codice indicando lo `schema` (Libreria) dove trovare i file e la porta HTTP dove far girare Node.js.

Esempio :

```
const schema = <my_IBMi_library>
var port = 3000
```

Utilizzare materiale corso su altri sistemi 3

DB : dopo aver correttamente impostato la libreria, possiamo generare la tabella di esempio e far partire la nostra applicazione.

```
node db.js
node index.js
```

IBMi dashboard

Version modificata per node.js V6 dentro git workshop nella cartella **`nodejs-Workshop/ibmidash`**. Link e istruzioni node.js v4 qui

```
cd imbidash
npm install
openssl genrsa -out ibmidash-key.pem 2048
openssl req -new -sha256 -key ibmidash-key.pem -out ibmidash-csr.pem
openssl x509 -req -in ibmidash-csr.pem -signkey ibmidash-key.pem -out
ibmidash-cert.pem
node index.js
```

CCSID

Impostare CCSID

```
system DSPJOB | grep CCSID
system "CHGUSRPRF USRPRF(DAVIDE) CCSID(1144)"
system 'CHGJOB CCSID(1144)'
system DSPJOB | grep CCSID
exit
E' necessario rieffettuare login su bash.
```

iWorkshop Node.js e IBMi

FINE