Node.Js

Quick View

Luca Pau
luca.pau82@gmail.com

- https://github.com/felixge/node-style-guide

Perchè?

Node's goal is to provide an easy way to build scalable network programs.

nodejs.org

Che cos'è nodeJs

- JavaScript è ben conosciuto come linguaggio di programmazione client-side nel browser
- node.js è JavaScript eseguito server-side
- SSJS -> Server-Side JavaScript
    - Usare un linguaggio che conosciamo
    - Usare lo stesso linguaggio sia sul client che sul server
- Powered by Google V8 runtime
- Veloce, estensibile scalabile
- NON E' un web framework, NON E' un linguaggio

L’idea di NodeJs

- Eseguire processi asincroni in un thread singolo invece che il classico multithread processing, minimizzare l’hoverhead e la latenza, massimizzare la scalabilità
- Scalare orrizontalmente invece che verticalmente
- Ideale per applicazioni che servono moltissime richieste e non richiedono una grande computazione per ogni richiesta
- Non ideal per calcoli pesanti
- Minor problemi con la concorrenza

Come?

Keep slow operations from blocking other operations.

I/O Tradizionale

```
1 var data = file.read('fileMoltoCorposo.txt');
2 doSomethingWith(data);
```

I/O Tradizionale

```
1 var data = file.rea ( 'fi  MoltoCorpos  txt');

 // zzzZZzz

2 doSomethingWith(data);
```

Spreca il tempo di attesa di
lettura del file

I/O Async

```
1 file.read('fileMoltoCorposo.txt', function(data) {
2     doSomethingWith(data);
3 });
4 doSomethingElse();
```

Non aspetta la lettura del file,
nel frattempo fa altro.

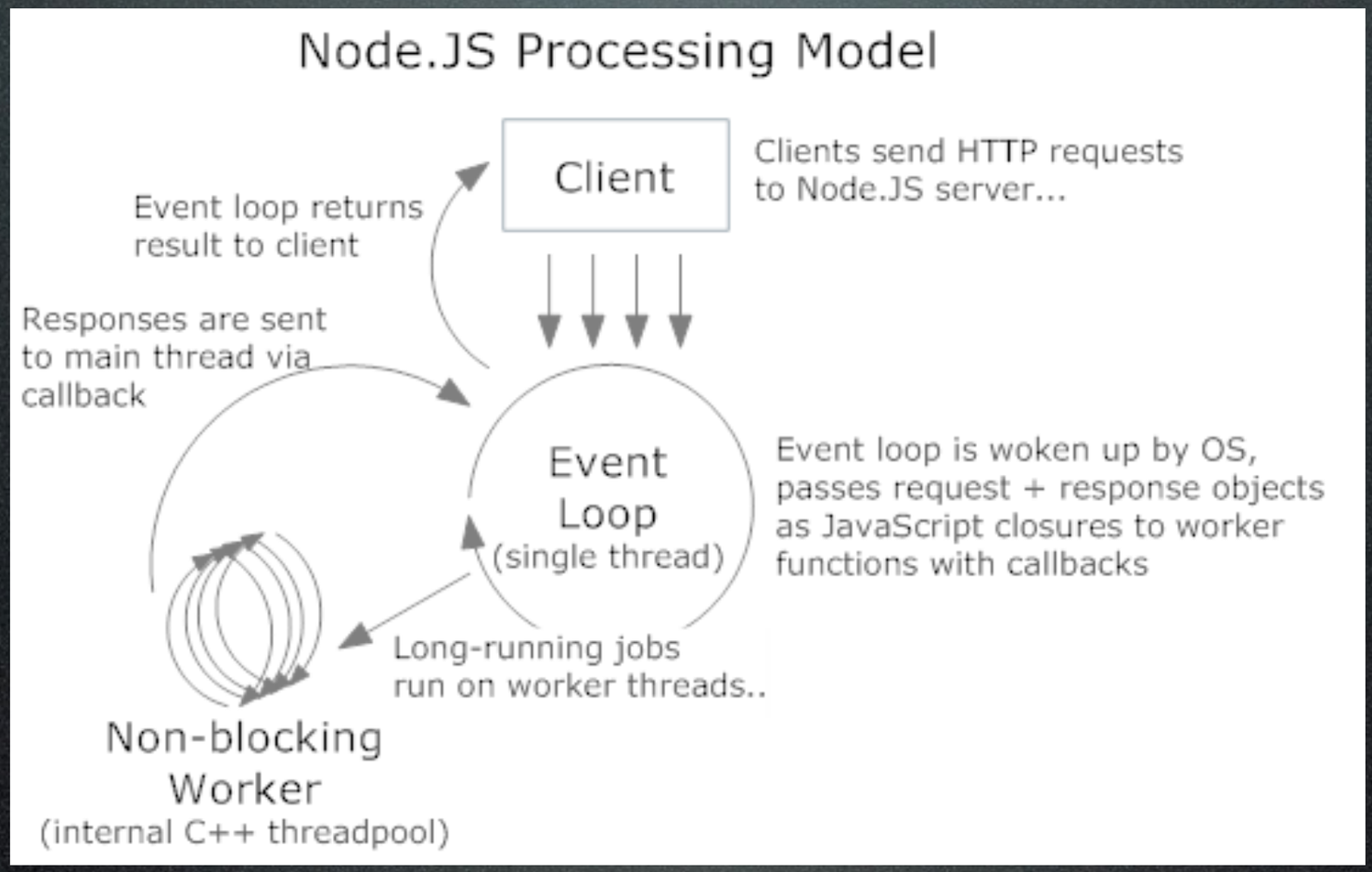
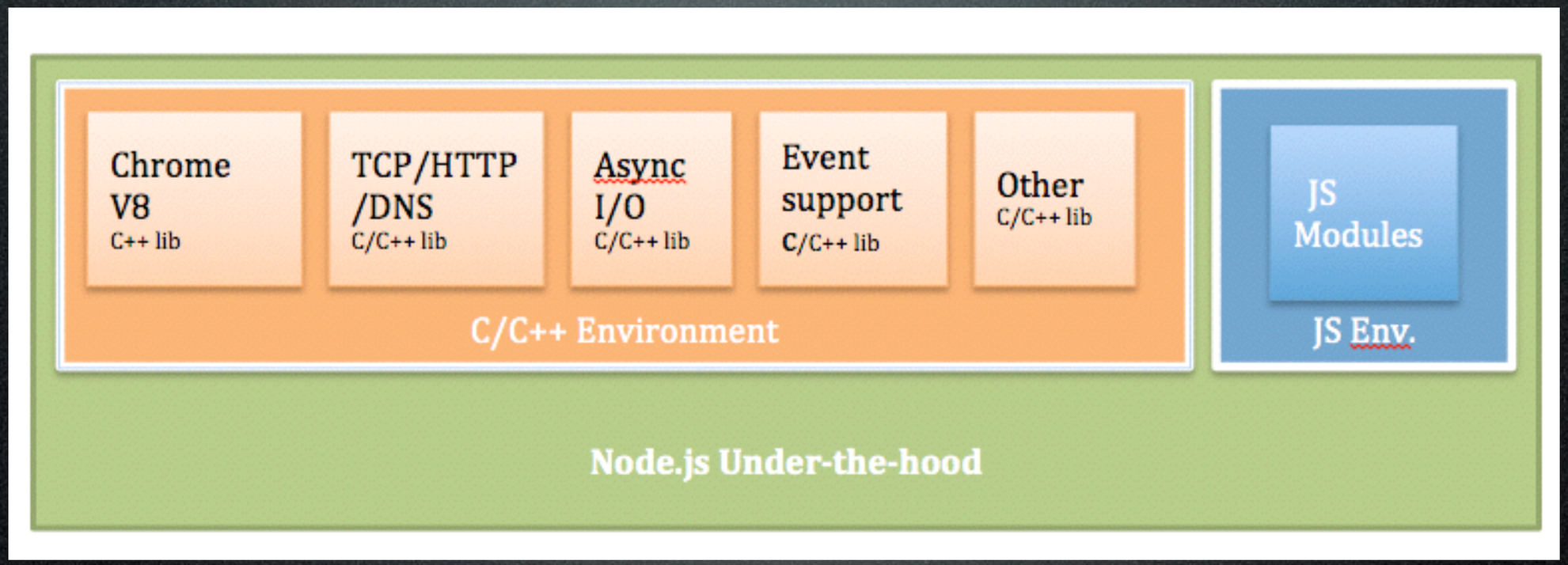


Diagram by Aaron Stannard
http://www.aaronstannard.com/post/2011/12/14/Intro-to-NodeJS-for-NET-Developers.aspx

Il rovescio della medaglia

- Evitare codice sincrono in quanto blocca l'event loop.
- callbacks, callbacks, and altre callbacks

Come è fatto?


Chrome V8
C++ lib
TCP/HTTP /DNS
C/C++ lib
Async I/O
C/C++ lib
Event support
C/C++ lib
Other
C/C++ lib
C/C++ Environment
JS Modules
JS Env.
Node.js Under-the-hood

Hello Node!!

app.js

```
1 console.log('Hello World');
```

```
$ node app.js
Hello World
```

Hello Node HTTP

Chiama il metodo per creare il nuovo oggetto server

Usa require per il caricamento del modulo

Definisce una callback anonima

```
var http = require('http');
http.createServer(function(req, res) {
    res.writeHeader(200, {'Content-Type': 'text/plain'});
    res.end('Hello world');
}).listen(3000);
```

Risponde alla chiamata HTTP

Si mette in ascolto sulla porta 3000

```
$ curl localhost:3000
Hello world
```

MonoThread

```
1 file.read('file.txt', function(data) {
2     // Will never fire
3 });
4 while (true) {
5     // this blocks the entire process
6 }
```

Node non è adatto a all'uso di algoritmi che fanno largo uso della CPU.

Server statico di file

```
1 var connect = require('connect'),
2     http = require('http'),
3     directory = '/path/to/Folder';
4
5 connect()
6     .use(connect.static(directory))
7     .listen(80);
8
9 console.log('Listening on port 80.');
```

Node.Js

Installazione e ambiente di esecuzione

Tool e ambiente di lavoro

- Node.js: http://nodejs.org
- Webstorm: http://www.jetbrains.com/webstorm/
- Un paio di terminali (Prompt dei comandi per i M$ oriented)

Node REPL

read-eval-print loop

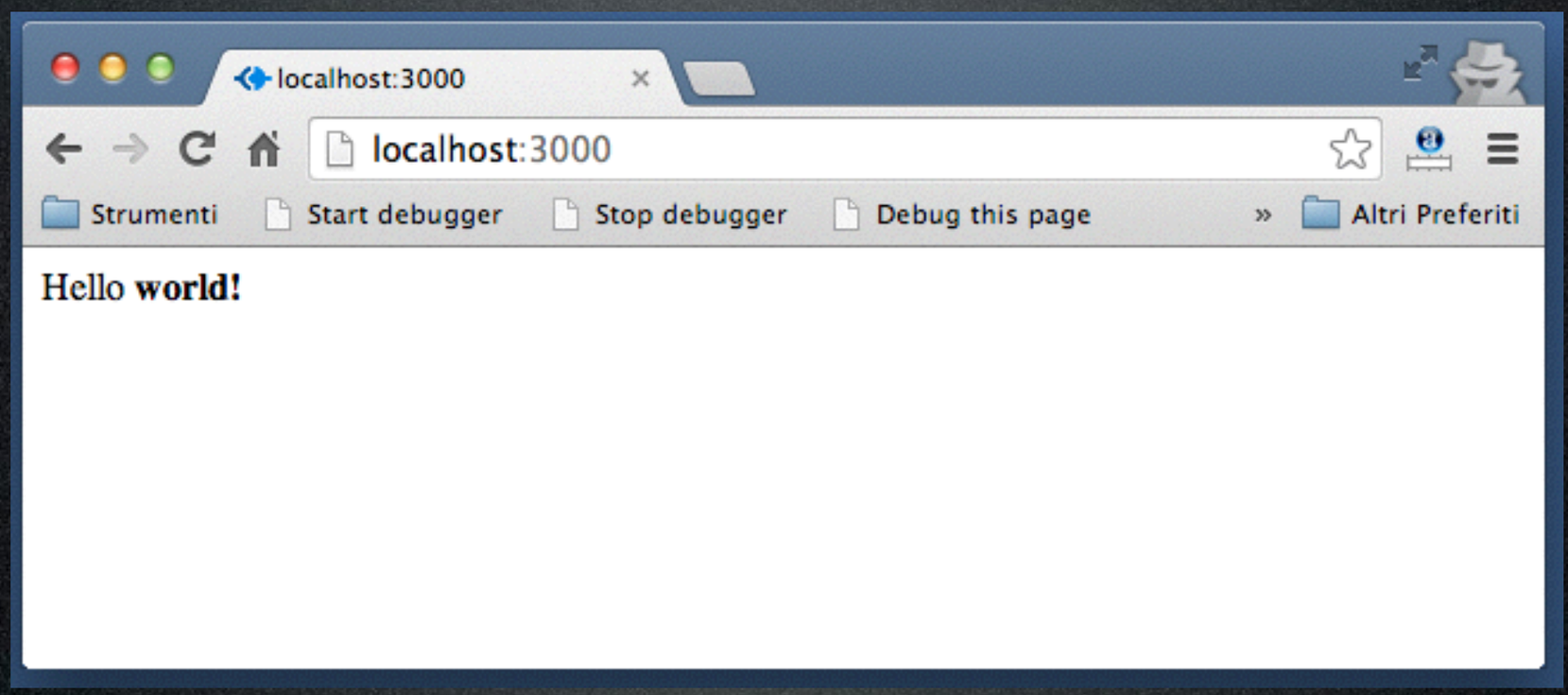
```
$ node
> Object.keys(global)
[ 'ArrayBuffer',
  'Int8Array',
  'Uint8Array',
....
  'clearInterval',
  'setImmediate',
  'clearImmediate',
  'console',
  'module',
  'require',
  '_' ]
>
```

Esecuzione di un file

app.js

```
1 var http = require('http');
2 var serv = http.createServer(function (req, res) {
3     res.writeHead(200, {'Content-Type': 'text/html'});
4 res.end('Hello <b>world!</b>');
5 });
6 serv.listen(3000);
```

```
$ node app.js
```



NPM
(Node Package Manager)

Permette di gestire con facilità i moduli in un progetto eseguendo il download, gestendo le dipendenze, eseguendo i test e installando utility

```
$ npm
Usage: npm <command>

where <command> is one of:
    add-user, adduser, apihelp, author, bin, bugs, c, cache,
...
$ npm --version
1.3.21
```

NPM
Installazione di un modulo

```
$ mkdir test
$ cd test
$ npm install color
$ ls -l
drwxr-xr-x  3 luca  staff  102  1 Apr 16:52 node_modules
```

app.js

```
1 require('colors');
2 console.log('hello world'.rainbow);
```

```
$ node app.js
hello world
```

NPM

Installazione di un applicazione

package.json

```
1 {
2     "name" : "test-project",
3     "version" : "0.0.1",
4     "dependencies" : {
5         "colors" : "0.5.0"
6     }
7 }
```

```
$ ls -l
-rw-r--r--  1 luca  staff   54  1 Apr 16:55 app.js
-rw-r--r--  1 luca  staff  113  1 Apr 17:01 package.json $
npm install
$ ls -l
-rw-r--r--  1 luca  staff   54  1 Apr 16:55 app.js
drwxr-xr-x  3 luca  staff  102  1 Apr 17:03 node_modules
-rw-r--r--  1 luca  staff  113  1 Apr 17:01 package.json
```

NPM
Comandi e utility

Creazione di un file package.json

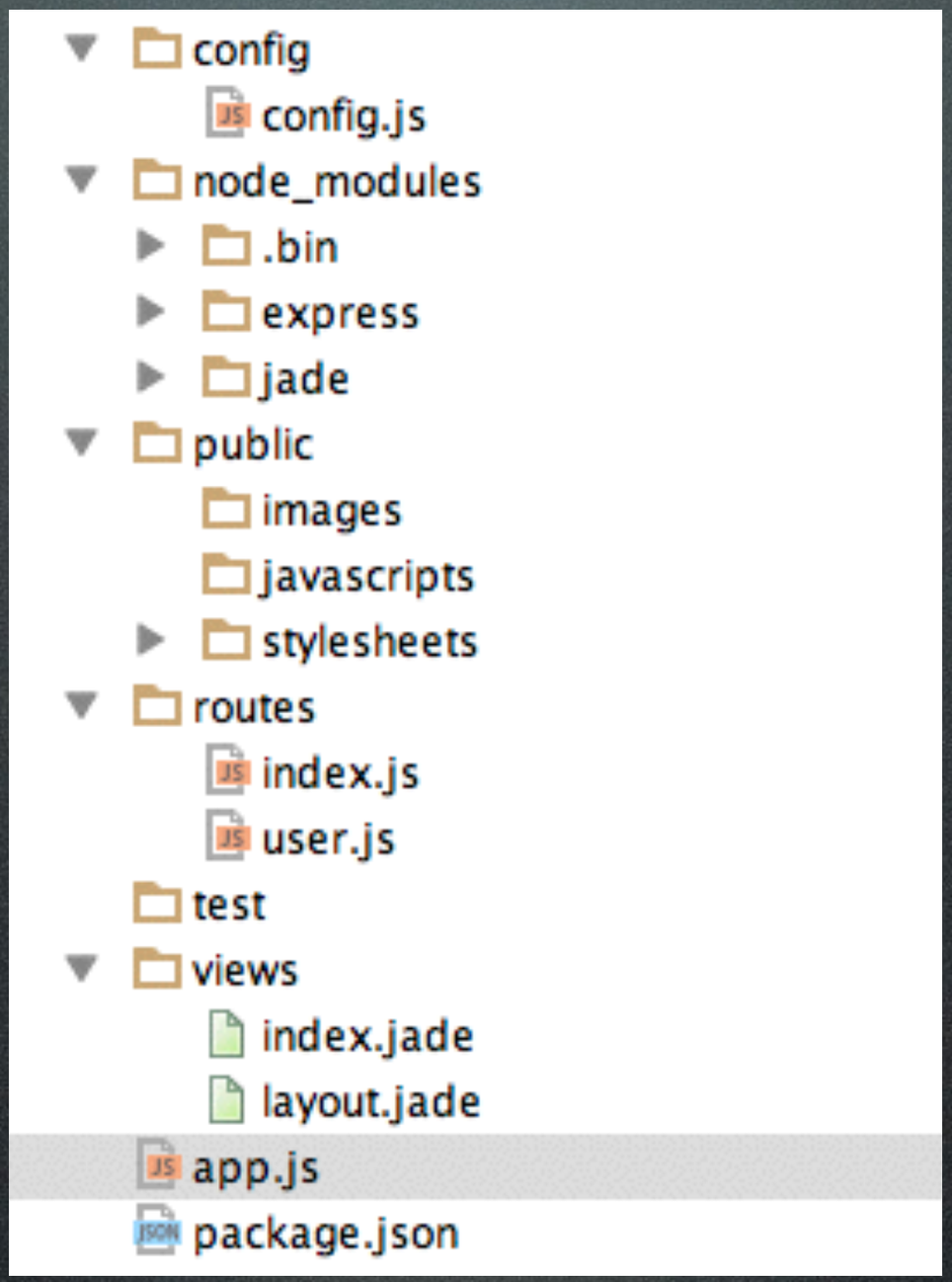
```
$ npm init
```

Installazione nel sistema di un package e relativi comandi

```
$ npm install -g nome-modulo
```

```
$ npm install -g express
$ mkdir test-express
$ cd test-express
$ express
```

Struttura di una app



Node.Js

Javascript for Node
Language core

The Global Object

Nei browser la window è l'oggetto globale.
Node espone invece due oggetti:

- global - Ogni variabile connessa a 'global' è accessibile da ovunque nel codice
- process - Contiene i riferimenti relativi al contesto di esecuzione

Non solo Javascript5

Per sopperire ad alcune mancanze di javascript nel core di NodeJs sono stata aggiunta:

- Gestione dei moduli
- Eventi
- Buffer

Module System

Browsers

Nei browser l'utilizzo di nuovi moduli avviene generalmente tramite inclusione e associazione del modulo all'oggetto globale

```
<script>
    jQuery(function () {
        alert('hello world!');
    });
</script>
```

- name collision
- pollution of global variable

Module System

Node.js

Node introduce un sistema semplice e molto potente

my-module.js

```
1 var count = 0;
2 exports.next = function() { return count++; }
```

app.js

```
1 var counter = require('./my-module.js');
2 console.log(counter.next());
```

Esempi uso moduli

Events

EventEmitter API implementa le funzionalità necessarie per la gestione degli eventi

```
1 var EventEmitter = require('events').EventEmitter;
2 var a = new EventEmitter();
3
4 a.on('event', function() {
5     console.log('event called');
6 });
7 a.emit('event');
```

Buffer

Con l'implementazione dell'oggetto Buffer
vengono gestiti i dati in formato binario

```
1 var myBuffer = new Buffer('==ii1j2i3h1i23h', 'base64');
2 require('fs').writeFile('logo.png', myBuffer);
```

Node.Js

Struttura e flusso di un programma Node.Js

Hello World!!

```
1 console.log('Hello World');
```

```
$ node app.js
Hello World
$
```

Hello World 2!!

```
1 var http = require('http');
2
3 http.createServer(function (req, res) {
4     res.writeHead(200, {'Content-Type': 'text/plain'});
5     res.end('Hello World\n');
6 }).listen(1337, '127.0.0.1');
7
8 console.log('Server running at http://127.0.0.1:1337/');
```

```
$ node app.js
Server running at http://127.0.0.1:1337/
```

Express

Connect

Connect is an extensible HTTP server framework for node, providing high performance "plugins" known as middleware.

- Connect wrap gli oggetti Server, ServerRequest, e ServerResponse del modulo HTTP di node.js

- Aggiunge la possibilità per l'oggetto server di utilizzare uno stack di middleware

Middleware

- I middleware sono delle funzioni che gestiscono richieste
- Un server creato con “connect.createServer” può utilizzarne uno stack
- Tre parametri (request, response, next)