

Introduzione a Node.js

Copyright Università degli Studi di Milano

Cos' è Node.js?

- Node.js è un ambiente JavaScript multiplatforma lato server.
- JavaScript è il linguaggio utilizzato come collante per i moduli C++
- È basato sul motore JavaScript V8 di Google
- Node.js utilizza un modello di I/O non bloccante e guidato dagli eventi
- Node.js è come un browser web con un diverso insieme di moduli C++
- Fornisce moduli per interagire con le risorse del sistema, come i processi, il file system, la rete, ecc...

Installare Node.js

- Node.js può essere installato sui principali sistemi operativi.
- Per scaricare il programma di installazione: <http://www.nodejs.org>

Copyright Università degli Studi di Milano

Versioni di Node.js

- Il programma di installazione è disponibile:
 - in una versione Long-Term Support
 - in una versione Corrente (Current).
- La versione Long-Term Support è la migliore per le applicazioni in produzione.
- La versione Current è la migliore per lavorare con le funzioni più recenti.

Demo: Installare Node.js

Copyright Università degli Studi di Milano

Node.js Programmi

- I programmi Node.js possono essere creati utilizzando file JavaScript che devono essere eseguiti tramite Node.js.
- I file contengono codice JavaScript e hanno estensione '.js'.
- Per eseguire un programma, è necessario eseguire Node.js e passare il nome del file (con o senza l'estensione JS) come primo parametro

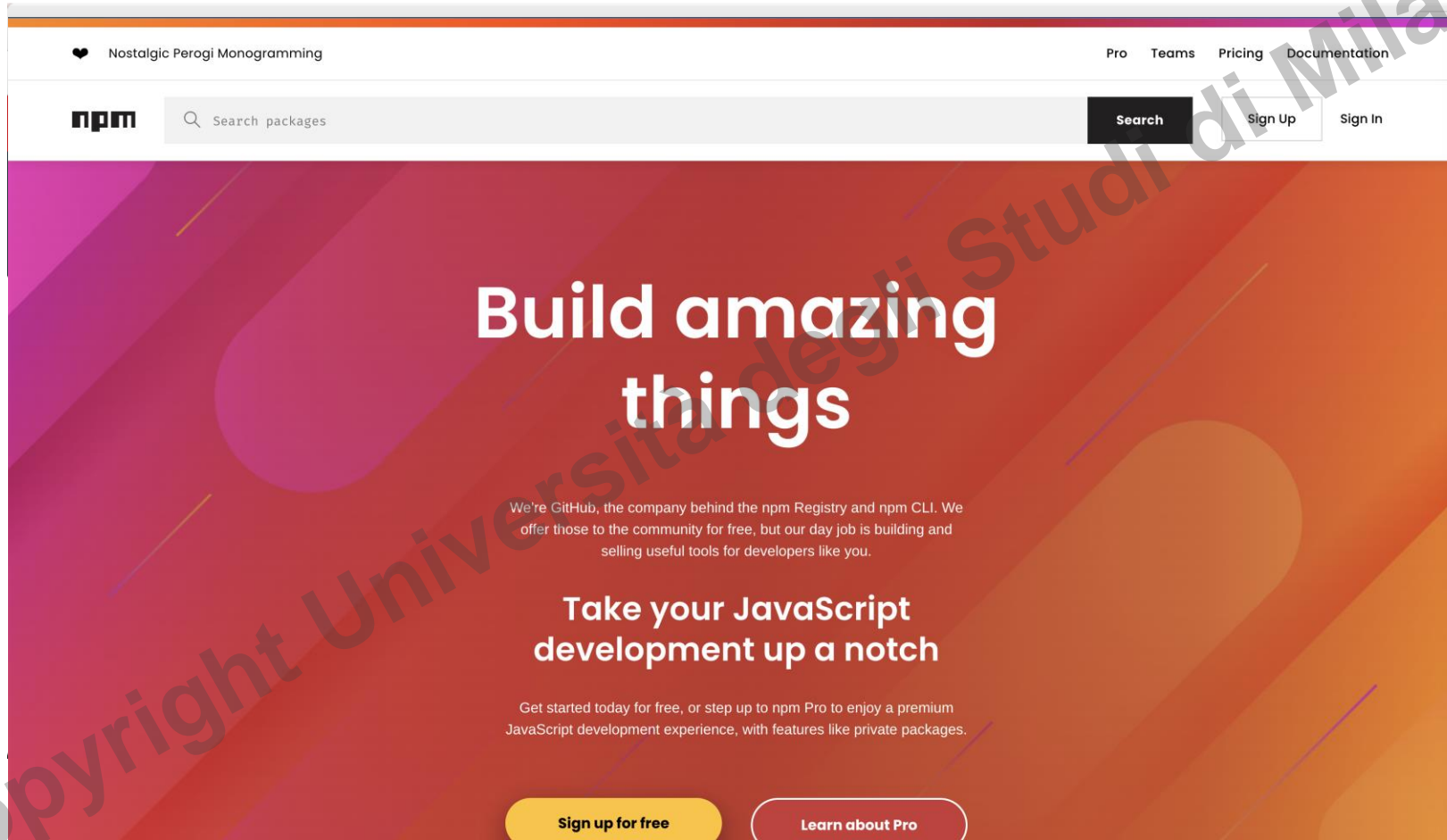
DEMO: Eseguire un programma Node.js

Copyright Università degli Studi di Milano

Gestire Pacchetti con NPM

- NPM è l'acronimo di Node.js Package Manager (Gestore di pacchetti Node.js). Fornisce:
 - NPM fornisce un repository pubblico di pacchetti
 - una specifica per la creazione di pacchetti
 - uno strumento a riga di comando per lavorare con i pacchetti.
- La società npm, Inc. sviluppa e mantiene NPM.
- Node.js distribuisce npm insieme a Node.js, ma in realtà si tratta di un programma separato con un proprio versioning

NPM Repository pubblica



<https://www.npmjs.com/>

Pacchetti Locali vs Globali

- I pacchetti possono essere installati localmente o globalmente.
- I pacchetti locali sono memorizzati localmente nella cartella `node_modules`
 - In genere, i pacchetti locali sono librerie di codice utilizzate dal progetto
- I pacchetti globali sono memorizzati globalmente nel sistema
- I pacchetti locali sono disponibili solo all'interno del progetto specifico, mentre quelli globali sono disponibili in tutto il sistema.

Installing & Uninstalling Packages

- Il programma npm viene utilizzato per gestire i pacchetti
- Il primo argomento del programma npm è il comando da eseguire
- I pacchetti possono essere installati con il comando install e disinstallati con il comando uninstall.
- Esistono molti altri comandi disponibili per NPM

```
$ npm install --global express
```

```
$ npm install express
```

```
$ npm uninstall --global express
```

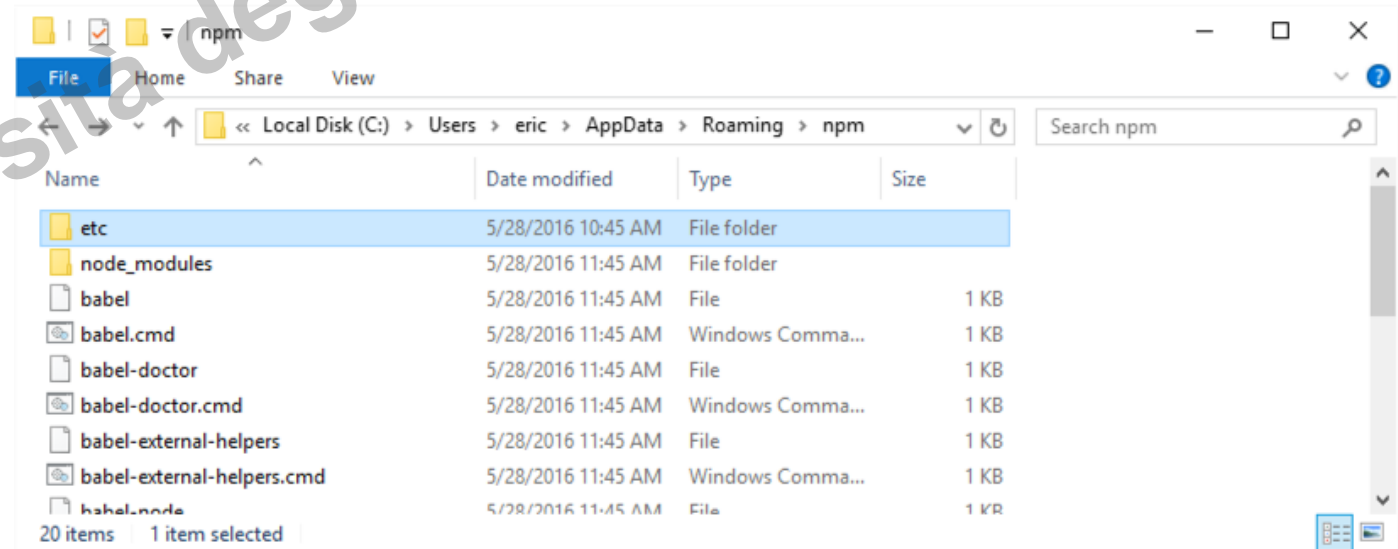
```
$ npm uninstall express
```

- Il flag **--global** installa e disinstalla il pacchetto a livello globale; senza il flag global, i pacchetti vengono installati e disinstallati localmente.

Pacchetti Globali su Windows

- Sono globali per l'utente, non per il sistema
- Non richiede privilegi amministrativi

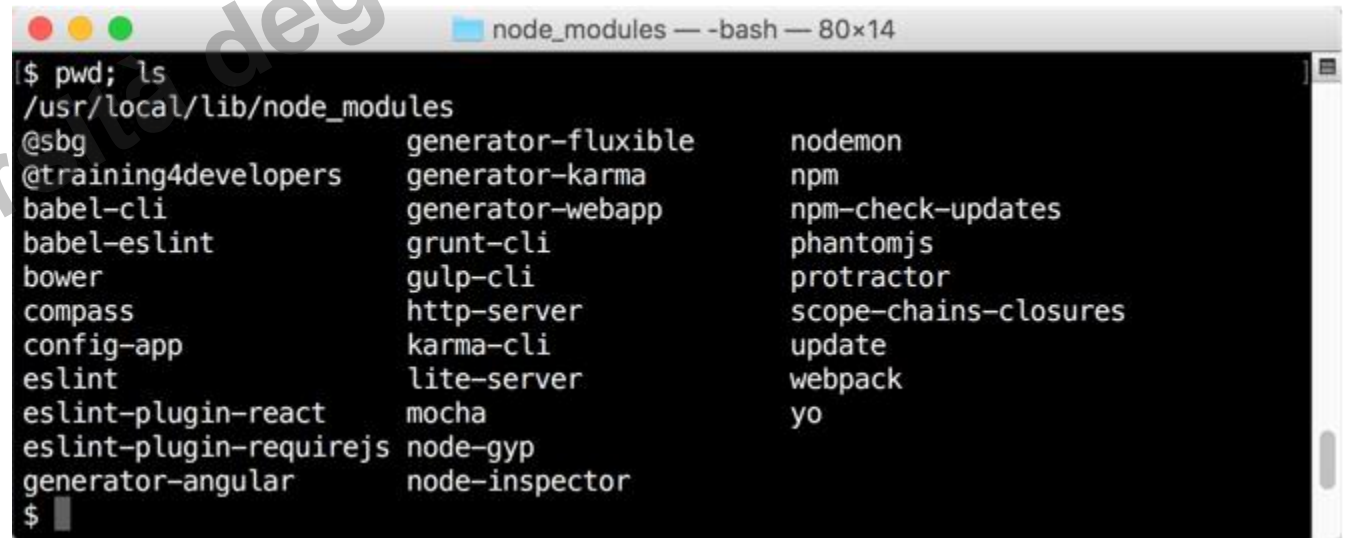
Globally Installed Packages are stored in:
C:\Users\<username>\AppData\Roaming\npm



Pacchetti Globali su Mac e Linux

- Per impostazione predefinita, sono globali per il sistema, non solo per l'utente
- Richiede privilegi amministrativi

Su un Mac per impostazione predefinita, i pacchetti installati globalmente sono memorizzati in:



```
node_modules — -bash — 80x14
$ pwd; ls
/usr/local/lib/node_modules
@sbg          generator-fluxible  nodemon
@training4developers generator-karma      npm
babel-cli     generator-webapp    npm-check-updates
babel-eslint  grunt-cli           phantomjs
bower         gulp-cli            protractor
compass      http-server         scope-chains-closures
config-app    karma-cli           update
eslint        lite-server         webpack
eslint-plugin-react mocha               yo
eslint-plugin-requirejs node-gyp
generator-angular node-inspector
$
```

DEMO: Gestione dei pacchetti con NPM

Creare un semplice Server Web

- Uno dei moduli principali per Node.js è il modulo HTTP, che fornisce un server Web e un client Web
- Il server web è molto flessibile, ma richiede molto codice boilerplate per costruire anche le applicazioni più semplici
- Comunemente, altri pacchetti come Express vengono utilizzati per configurare il server web
- I server Web sono applicazioni ad alta intensità di I/O che li rendono adatti per Node.js
- Node.js è ottimo per i server Web grazie alla sua facile gestione dei dati JSON

DEMO: Creare un semplice webserver

Copyright Università degli Studi di Milano

Leggere I File Dal Disco

- Node.js consente l'accesso completo al sistema a differenza di un browser Web che consente solo l'accesso in modalità sandbox
- L'accesso alle risorse del file system può avvenire in modo sincrono e asincrono
- L'accesso sincrono può essere utilizzato per il caricamento iniziale del programma, ma solo l'accesso asincrono deve essere utilizzato durante il funzionamento del programma
- Sia il testo che i dati binari possono essere letti e scritti

Creare un pacchetto

- Tutti i progetti (che sono anche pacchetti) devono essere configurati per funzionare con NPM
- Il comando `npm init` viene utilizzato per configurare un progetto
 - Verrà posta una serie di domande, tutte con risposte predefinite, utilizzate per creare e inizializzare un file `package.json`
- Il file `package.json` contiene metadati sul progetto, nonché un elenco di dipendenze dell'applicazione e dello sviluppo
- Quando i pacchetti NPM sono installati, NPM li salva nel file `package.json`

Salvare le dipendenze di un pacchetto

- La semplice installazione dei pacchetti non salva la dipendenza nel file `package.json`
- Oltre all'installazione, è necessario specificare flag aggiuntivi:
 - `--save` o `-S` salverà il pacchetto come dipendenza dell'applicazione
 - `--save-dev` o `-D` salverà il pacchetto come dipendenza per lo sviluppo
- Le dipendenze dell'applicazione vengono utilizzate dal programma Node.js durante l'esecuzione (esempio comune potrebbe essere Express)
- Le dipendenze per lo sviluppo vengono utilizzate per sviluppare il programma Node.js (esempio comune potrebbe essere Grunt)

Salvare le dipendenze di un pacchetto

Terminal Commands

```
$ npm init -y
```

```
$ npm install -S express
```

```
$ npm install -D grunt
```

```
$ more package.json
```

- I comandi del terminale a sinistra, produrranno un file package.json simile a quello a destra.

- Il file è un file JSON e può essere modificato manualmente
- Name è il nome del pacchetto
- La versione segue lo schema SEMVER
- Viene monitorata anche la versione di ogni dipendenza
- Main è il file principale importato quando si richiede il modulo
-

Package.json

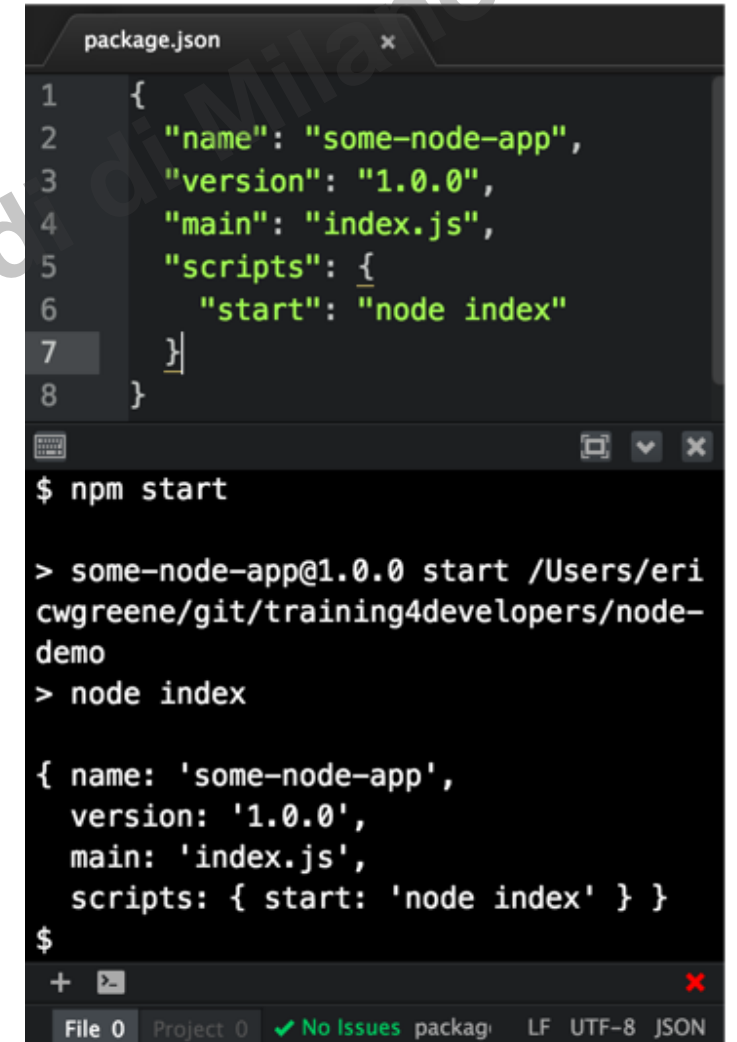
```
{  
  "name": "demo",  
  "version": "1.0.0",  
  "description": "",  
  "main": "Gruntfile.js",  
  "dependencies": {  
    "express": "^4.13.4"  
  },  
  "devDependencies": {  
    "grunt": "^1.0.1"  
  }  
}
```

DEMO: Creare un pacchetto

Copyright Università degli Studi di Milano

Use NPM per eseguire un pacchetto

- Oltre ad essere gestiti NPM, i pacchetti possono essere eseguiti con NPM
- Per configurare l'esecuzione dei pacchetti Node, viene configurata l'opzione scripts del file package.json
- Per eseguire un programma Node.js, lo script di avvio è configurato



The screenshot shows a code editor window titled 'package.json' with the following content:

```
1 {  
2   "name": "some-node-app",  
3   "version": "1.0.0",  
4   "main": "index.js",  
5   "scripts": {  
6     "start": "node index"  
7   }  
8 }
```

Below the editor is a terminal window. The first command is `$ npm start`. The output shows the command being executed: `> some-node-app@1.0.0 start /Users/ericwgreene/git/training4developers/node-demo`, followed by `> node index`. The terminal then displays the package.json content as JSON:

```
{ name: 'some-node-app',  
  version: '1.0.0',  
  main: 'index.js',  
  scripts: { start: 'node index' } }
```

The terminal prompt returns to `$`. At the bottom of the editor, a status bar shows 'File 0', 'Project 0', 'No Issues', 'packag', 'LF', 'UTF-8', and 'JSON'.

DEMO: Usare NPM per eseguire un pacchetto