Introduzione

Non ho memoria della prima volta che ho usato Node.js, forse tra il 2012 e il 2013 su consiglio di un amico, o forse più tardi in un'altra occasione. Quello che però ricordo bene sono le tante volte in cui Node.js mi ha permesso di trasformare quello che avevo in mente in software funzionante. Non dovrebbe esserci nulla di speciale in quest'affermazione poiché questo è (in teoria) lo scopo di tutte le tecnologie che usiamo. Node.js, però, rientra nella ristretta cerchia di quegli strumenti che permettono di farlo senza sforzo e che, una volta provati, è difficile abbandonare.

Creare software è un'attività simile al disegnare: partiamo da un'idea, gli diamo una forma, vediamo se ci piace, la aggiustiamo da una parte e poi dall'altra, sistemiamo i dettagli fino a raggiungere il risultato che cercavamo. Non tutti i linguaggi e le tecnologie ci permettono di creare e trasformare il codice in modo così veloce, ma Node.js sì, grazie alle sue API e all'utilizzo di JavaScript. Questo dinamismo, la velocità e la flessibilità lo rendono la scelta ideale per chi vuole creare software in modo agile, ottenendo subito un risultato che potrà poi essere ampliato e perfezionato quando necessario. Anche se il nostro approccio al software è più tradizionale e basato su lunghe fasi di analisi, progettazione e implementazione, non sarà di certo Node.js a impedirci di continuare su questa strada.

Essendo nato nell'era di Internet, Node.js ha sposato fin dall'inizio la Rete e, come vedremo poi, questa fa parte di esso in modo indissolubile. Per creare un server HTTP o per stabilire connessioni TCP non servono moduli aggiuntivi: tutto questo è integrato e disponibile fin dalla sua nascita nel core di Node.js. Ciò ha fatto sì che l'intero modello d'esecuzione di Node.js sia stato plasmato dalle necessità dei protocolli di rete; ne scopriremo i risultati positivi nel corso del volume.

Tutte le sue caratteristiche lo hanno ben presto trasformato nella scelta ideale per chi cercava (e cerca ancora) lo strumento giusto per realizzare uno script, delle API, un'applicazione web, un sistema distribuito o semplicemente la tecnologia giusta da usare per costruire la sua startup tecnologia; per esperienza diretta posso affermare che Node.js brilla in tutti i casi elencati, e non solo. Questa potrebbe sembrare un'affermazione di parte se fatta dall'autore di un volume dedicato all'argomento ma, per fortuna, lo sterminato elenco di progetti e aziende che da quasi quindici anni usano Node.js parla da solo; è sufficiente effettuare una ricerca online per scoprire i nomi di coloro che grazie a Node.js hanno realizzato casi di successo.

Nonostante tutto questo, quando mi è stato proposto di scrivere un libro su Node.js sono rimasto perplesso per diversi motivi. Il primo è dovuto alla mole di risorse già

esistenti: decine di volumi sull'argomento, centinaia di corsi e uno sterminato numero di tutorial. Guardando le cose un po' più da vicino mi sono accorto però che la maggior parte di questi contenuti, oltre a essere in inglese, seguivano quasi tutti lo stesso copione. La struttura di queste guide è spesso basata sull'utilizzo di framework che nascondono le vere capacità di Node.js e che, in alcuni casi, lo limitano. Inoltre, una volta spiegato il funzionamento di alto livello è raro che il discorso continui verso le API di Node.js e la sua struttura interna. Tutto questo, per me, rappresenta un debito di conoscenza, il cui pagamento verrà richiesto non appena sarà necessario fare qualcosa che vada oltre quanto previsto dal framework o quando si verificherà un problema non previsto. A quel punto, il conto da pagare sarà alto e non è detto che saremo in grado di gestirlo.
Con questo volume ho quindi cercato di offrire un percorso alla scoperta di Node.js che seguisse una strada diversa: un passo alla volta, partendo dal basso e dalle sue funzioni più semplici, come la lettura di un file, fino ad arrivare a quelle più complesse come la creazione di server HTTP e TCP. Il movimento di questo viaggio è però tanto verso l'alto quanto verso il basso, perché tanto più saliremo per scoprire le astrazioni offerte da Node.js, tanto più affonderemo nei suoi meccanismi interni per comprenderle. Questo ci permetterà di avere una visione completa e chiara dello strumento che abbiamo di fronte e di capire come tutte le sue caratteristiche siano correlate tra loro in modo armonico. La tecnologia non deve mai essere magia.

A chi è rivolto

È difficile identificare una sola categoria di persone che può beneficiare dalla conoscenza di Node.js. Grazie alla sua flessibilità e al suo sterminato ecosistema di librerie disponibili, Node.js è uno strumento in grado di offrire qualcosa a ognuno di noi. Nonostante sia una tecnologia ormai nella fase della maturità, la sua evoluzione continua senza sosta e le possibilità che ci offre aumentano ogni giorno.
Questo volume si rivolge quindi a tutte le persone che non conoscono Node.js e vogliono impararlo ma anche a chi già lo utilizza e vuole consolidare o ampliare le proprie conoscenze. Per molte persone Node.js è rappresentato dal framework che utilizzano, oltre il quale non vanno mai. È per questo motivo che anche chi utilizza Node.js da anni potrebbe scoprire in questo volume cose che non conosce o che non ha mai avuto modo di approfondire. Una su tutte, anche se molto inflazionata, "cos'è e come funziona veramente l'event loop?".

Prerequisiti

Per poter leggere e comprendere appieno questo volume è necessario conoscere i fondamenti della programmazione. In questo caso è utile avere una conoscenza base di JavaScript. Se non la avete, potete recuperare usando i tanti libri disponibili su questo argomento nel catalogo Apogeo oppure attraverso i tanti siti disponibili online. Se siete alla ricerca di un volume in italiano è consigliato *JavaScript la guida definitiva* di David Flanagan (Apogeo, 2021), traduzione del best seller inglese dal titolo omonimo, poiché copre tutte le caratteristiche moderne di JavaScript che useremo in queste pagine. Nel corso del volume saranno comunque evidenziate, e spiegate, eventuali caratteristiche

di JavaScript fondamentali per la comprensione di quello che stiamo facendo ma che potrebbero non essere chiare a tutti.

L'approccio

In ogni capitolo del libro affronteremo gli argomenti che ci interessano usando esempi pratici: non singoli blocchi di codice senza contesto ma veri e propri progetti realizzati da zero. La loro dimensione andrà da poche righe di codice fino ad applicazioni composte da diversi file. L'approccio sarà quindi sempre incentrato sulla pratica poiché nulla si può sostituire a essa. Solo leggendo il codice mostrato e scrivendone altro si potrà trasformare quanto letto in conoscenza propria da riutilizzare per creare software basati su Node.js. Per le prime 200 pagine del volume non useremo alcun modulo esterno oltre a quelli forniti nel core di Node.js. Quando negli ultimi capitoli installeremo alcune librerie esterne e useremo un framework, il loro funzionamento non ci colpirà né ci lascerà senza parole: grazie a tutto il lavoro fatto in precedenza riusciremo a comprenderne il funzionamento interno anche senza vederne il codice. La scelta di limitare al massimo l'utilizzo di moduli esterni non è casuale: questo libro voleva essere incentrato su Node.js e non sulle librerie del suo ecosistema. Dopo aver imparato a muoverci tra le sue API potremo usare con facilità Node.js e, allo stesso modo, tutto quello che è basato su di esso.

Il codice

Tutto il codice usato nel libro è scaricabile all'indirizzo `https://github.com/lifeisfoo/node-js`. Nel repository troverete tante cartelle quante i capitoli del libro: ogni cartella contiene i file creati nel capitolo corrispondente. Nella maggior parte dei casi i file all'interno di ogni cartella sono numerati in modo crescente, così da poterli facilmente individuare e distinguere: le spiegazioni e le implementazione saranno fatte in modo iterativo e quindi ogni file corrisponderà a una versione intermedia. Negli ultimi capitoli, quando le applicazioni realizzate richiederanno più file, useremo anche sottocartelle e file con un numero di versione: in ogni caso nel testo compaiono sempre i riferimenti al file utilizzato. Se nuovi rilasci di Node.js o delle librerie usate richiederanno cambiamenti al codice, il repository sarà aggiornato appena possibile. Per ricevere una notifica ogni volta che il codice viene modificato, potete usare la funzione "watch" nel repository di GitHub. Tutti gli esempi mostrati nel testo sono eseguiti su un sistema UNIX, quindi su sistemi Windows, se non usate WSL, potrebbe essere necessario aggiungere qualche .exe o adeguare la struttura dei path mostrati. Tutto il codice usato nel libro è scaricabile anche dal sito di Apogeo all'indirizzo `https://bit.ly/apo-nj`.

La struttura

Questo libro è composto da 10 capitoli, ognuno dei quali ha un suo scopo ed è dedicato a uno o più aspetti di Node.js. In alcuni capitoli vengono introdotti concetti che poi sono ripresi e approfonditi in capitoli successivi: questo implica che, come già detto, il

libro dovrebbe essere letto dall'inizio alla fine, così da non perdere nessuna delle spiegazioni fornite.
Nel *Capitolo 1* ripercorreremo la storia di Node.js e della strada che ha fatto per arrivare fino a noi. Capiremo perché è stato creato, in quale contesto e il suo legame con JavaScript. Scopriremo anche perché è così diverso dalle altre tecnologie e quali sono le caratteristiche che lo rendono così speciale. Se questa che state leggendo è l'introduzione al libro, il Capitolo 1 è l'introduzione a Node.js.
Nel *Capitolo 2* ci occuperemo di un aspetto che è spesso trascurato: le versioni di Node.js e il loro ciclo di vita. Cosa cambia tra una versione e l'altra? Cosa potrebbe cambiare nella prossima versione? Oltre a rispondere a queste domande vedremo anche i molti modi con cui è possibile installare Node.js e come gestire l'installazione di più versioni senza creare conflitti tra loro.
Nel *Capitolo 3* creeremo la nostra prima applicazione Node.js e faremo la conoscenza con le API per l'accesso al filesystem. Vedremo come effettuare operazioni base come la stampa di un log, la lettura dei parametri passati all'applicazione dalla console e come terminare l'esecuzione dell'applicazione. Avremo anche un primo contatto con le diverse modalità di programmazione asincrona: callback, Promise e `async/await`.
Nel *Capitolo 4* toccheremo con mano il protocollo HTTP, scoprendone i dettagli e le caratteristiche. Questo ci permetterà di vedere da vicino alcuni dei meccanismi che stanno alla base delle richieste che ogni giorno partono dai nostri browser o dalle API. Gli header, il routing e i tipi di contenuto accettati sono alcune delle cose con cui lavoreremo realizzando il nostro primo server HTTP usando solo le API di Node.js.
Nel *Capitolo 5* scenderemo più in basso e arriveremo al protocollo TCP. Quindi vedremo come il testo e i dati vengono trasformati in byte durante le trasmissioni e come funzionano i meccanismi di conversione. Sfruttando i *socket* creeremo una chat di rete che comunica usando TCP e che possiamo utilizzare direttamente dal terminale attraverso strumenti creati decenni fa. Facendo tutto questo scopriremo gli eventi e la classe `EventEmitter` di Node.js, che è alla base del suo funzionamento.
Nel *Capitolo 6* faremo conoscenza di una delle astrazioni più potenti di Node.js, gli *stream*. Vedremo come attraverso di essi sia possibile far fluire i dati da una parte all'altra in modo efficienze ed eventualmente trasformandoli. Questo meccanismo è un altro di quelli usati in modo pervasivo all'interno di Node.js e che incontreremo spesso. Per comprendere meglio gli stream realizzeremo un server web di file statici.
Nel *Capitolo 7* entreremo nel cuore di Node.js, scoprendo il modo in cui funziona l'*event loop* e come questo influenza l'esecuzione del nostro codice. Vedremo tutte le sue fasi e come alcune funzioni di Node.js ci permettano di interagire con esso, sfruttandolo. Dopo averne compreso il funzionamento, scopriremo i suoi limiti e come superarli attraverso la creazione di un'applicazione che verifica se un numero è primo. Questo stress test ci permetterà di usare i *child process*, il modulo `cluster` e i *worker thread*.
Nel *Capitolo 8* vedremo da vicino come funzionano i due meccanismi di gestione dei moduli supportati da Node.js: CommonJS ed EcmaScript Modules. Qui scopriremo le differenze tra i due e i diversi modi in cui il codice può essere esportato e importato. Infine, vedremo come possiamo utilizzare dei moduli creati da altri attraverso npm e come questo sia legato a Node.js. Per fare tutto questo realizzeremo diversi moduli locali, che uniremo ad altri scaricati, con la possibilità così di creare un feed reader da terminale.
Nel *Capitolo 9* faremo la conoscenza con quello che per molti è "Node.js", cioè Express. Vedremo come questo permetta di creare applicazioni web che restituiscono pagine utilizzando template dinamici, e per l'occasione creeremo un clone di APOD. Conti-

nueremo a utilizzarlo per creare un feed reader via API e questo ci permetterà di vedere come utilizzare Node.js con MongoDB attraverso Mongoose.
Nel *Capitolo 10* riscriveremo il nostro feed reader via API usando Fastify per comprendere il funzionamento di questo framework ad alte prestazioni. Basandoci sul progetto realizzato in precedenza potremo vedere con chiarezza le differenze tra i due framework. Infine, modificheremo ancora il progetto così da sfruttare alcune delle caratteristiche che rendono Fastify unico e che ci permettono di creare API in modo semplice e veloce.
Infine, nell'*Appendice* vedremo quali strade possiamo seguire dopo aver completato questo volume e cosa fare per continuare in modo ottimale il percorso iniziato con Node.js.

Le risorse extra

Alla pagina `https://node-js.miliucci.org` potete trovare ulteriori risorse fornite dall'autore per completare, ed estendere, quanto descritto in questo volume. Quello che è possibile fare con Node.js è virtualmente illimitato, mentre lo spazio a disposizione in un libro no: per questo motivo online troveranno spazio alcuni degli argomenti che non è stato possibile inserire in queste pagine ma che torneranno utili a tutti coloro che intendono proseguire il viaggio con Node.js. Qui troverete anche l'elenco di tutti gli URL citati nel libro divisi per capitolo, così da poterli aprire in modo veloce senza doverli trascrivere.

Come leggere questo libro

Il libro è stato scritto per essere letto in modo sequenziale: per questo motivo, saltando uno o più capitoli si perderanno concetti e dettagli essenziali alla comprensione del contenuto. Ognuno ha un proprio modo di leggere libri tecnici, ma un buon approccio alla lettura è leggere un capitolo alla volta lontano dal PC, cercando di seguire il codice stampato e le spiegazioni collegate. Leggere e riuscire a comprendere quello che un libro vuole comunicare è un'attività a due: il lettore e il libro. Ogni altro elemento aggiuntivo non farà che distrarvi e deconcentrarvi.
Alla fine della lettura di ogni capitolo, prendete il codice relativo e provatelo, modificatelo ed estendetelo: usando la documentazione di Node.js online e le spiegazioni contenute in questo volume avrete modo di sperimentare ed esplorare un po' alla volta quanto visto fino a quel punto. La lettura, seguita dalla pratica diretta, è il miglior modo per padroneggiare una tecnologia e sentirsi a proprio agio usandola.
Se durante la lettura avrete dubbi o idee, appuntatele e verificate poi, alla fine del volume, se le risposte che cercavate sono arrivate o no. Per evitare di sovraccaricare il lettore con spiegazioni troppo complicate o troppo voluminose, alcuni concetti sono spiegati prima in modo parziale per poi essere completati più avanti. Se entro la fine del libro ci sono dei dubbi da sciogliere, potete chiedere un chiarimento seguendo le istruzioni presenti nel paragrafo "Errori e feedback" alla fine di questa introduzione.

Convenzioni tipografiche

Il codice sorgente viene mostrato usando un carattere tipografico dedicato: per esempio, `const node = true`. In alcuni casi il codice compare direttamente nel testo, in altri casi in

modo separato all'interno di specifici listati. Alcuni di questi riportano l'intero contenuto del file, altri solo una parte per brevità o perché il resto del codice è già stato mostrato in precedenza e non è cambiato: in questi casi troverete una linea con il commento `//...`. Il simbolo `$` viene usato per indicare il prompt della console quando vengono eseguiti dei comandi dal terminale. Il nome di moduli, metodi e funzioni viene scritto in due formati, completo e abbreviato: per esempio, `net.Socket` è quello completo ma altre volte, quando non c'è ambiguità, viene usato solo `Socket`.

Terminologia

Il nome dei concetti legati a Node.js, o più in generale legati allo sviluppo e a Internet, sono riportati in forme diverse, alcune volte in inglese mentre altre in italiano. Quindi, per esempio, potreste trovare scritto *percorso* in una pagina, *path* in un'altra o addirittura *route*. Il motivo di questa scelta è dato dalla mancanza di una terminologia standard, in una lingua o l'altra, e alle sfumature di significato che queste parole assumono in base al contesto. Ho preferito quindi usare le forme più comuni in base all'argomento trattato cercando di rendere la lettura più semplice.

Errori e feedback

Potete segnalare errori, affermazioni non chiare o passaggi complicati creando una issue nel repository pubblico con il codice usato nel libro oppure inviandomi un'email all'indirizzo `lifeisfoo@gmail.com`. Cercherò di rispondervi e chiarire per voi e per gli altri lettori tutto quello che non è comprensibile dal testo.
Sono benvenuti, e ben accetti, commenti, pareri o messaggi di qualsiasi tipo riguardo il volume. Se volete aiutare gli altri a trovare questo libro, potete lasciare una recensione nel negozio online dove l'avete comprato, sul social network che utilizzate o semplicemente consigliarlo alle persone che conoscete.

Ringraziamenti

Queste pagine sono frutto di un lavoro durato parecchi mesi, tra alti e bassi, tra momenti sì e altri no. La scrittura è stata solo l'ultima fase del mio viaggio con Node.js perché è stata preceduta da anni in cui l'ho usato per fare praticamente ogni cosa. Oltre a essere stato argomento centrale di alcuni miei talk e workshop, Node.js è stato anche lo strumento principale che, negli ultimi anni, mi ha permesso di realizzare il software che volevo. Se però questo volume esiste è anche grazie ad alcune persone che, in modo diretto o indiretto, mi hanno permesso di realizzarlo.
Grazie quindi a Guido Frascadore che, qualche anno fa, mi ha fornito la scusa per usare Node.js in grande e scommettere su di esso, utilizzandolo per costruire insieme una startup. Grazie a tutti i colleghi e le colleghe con i quali ho avuto modo di lavorare con Node.js: ogni vostro problema ha contribuito ad approfondire certi argomenti e avere così spiegazioni migliori da fornire. Tra questi, grazie a Giovanni Pinto: insieme abbiamo creato molte applicazioni con Node.js, divertendoci.

Da un po' di anni RomaJS (https://romajs.org) è la mia seconda casa: qui ogni mese diventa possibile incontrare amici, conoscenti e persone nuove che coltivano la stessa passione e con i quali condividere conoscenze ed esperienze. L'incontro mensile si è ormai arricchito anche di un aperitivo e di una cena dove poter scambiare idee e pareri in modo decisamente più conviviale. Quindi grazie a tutte le persone che dal 2013 portano avanti quella che forse è la più longeva community italiana: senza di voi il mio percorso sarebbe stato diverso (di sicuro meno divertente).
Grazie a Matteo Manchi, Alessio Biancalana, Luciano Mammino, Manuel Spigolon, Matteo Collina, Alessandro (Cirpo) Cinelli, Roberto Mascherucci, Emanuele De Cupis, Lorenzo De Santis, Marco Ippolito, Paolo Insogna e tutte le altre persone che avrò sicuramente dimenticato, per aver dedicato un po' del loro tempo nel fornirmi pareri e informazioni utili ad alcune parti del libro.
Grazie a Guido D'Orsi per i feedback e le piccole osservazioni che hanno portato a una discussione sulle API di Node.js di ben altre dimensioni.
Grazie a Fabrizio Vitale per la perizia con cui ha revisionato i capitoli trovando problemi impossibili da vedere e fornendo consigli per migliorare la completezza delle spiegazioni.
Infine, un ringraziamento speciale va a Serena Sensini per la costanza, il tempo e l'impegno dedicato alla revisione di ogni capitolo e per le righe che avete letto nella Prefazione.

Qualsiasi errore o imprecisione rimane responsabilità unica dell'autore.