Stack MEAN

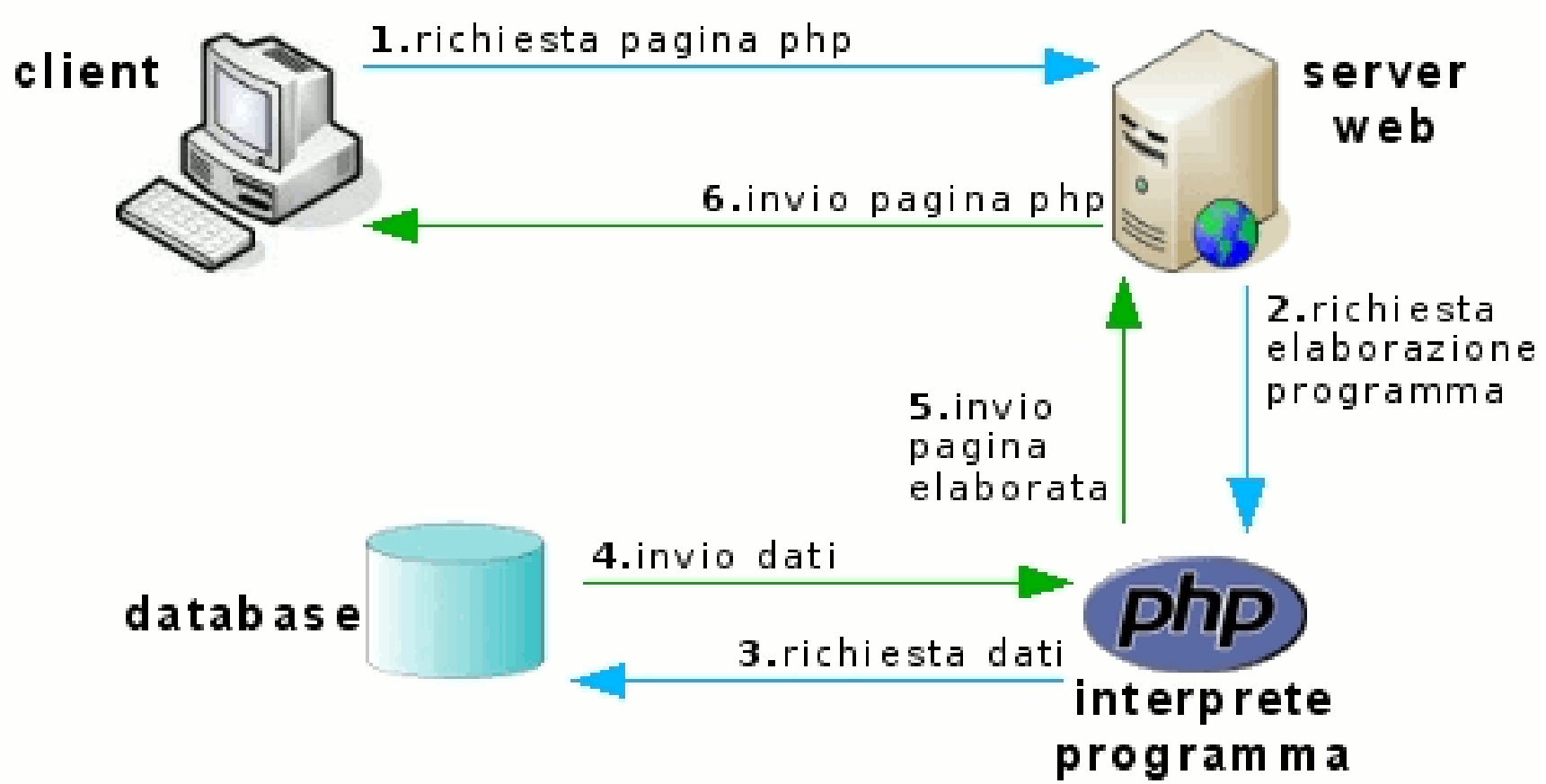
Summary

- Da LAMP a MEAN
- MEAN vs LAMP
  - Node.js
  - Express.js

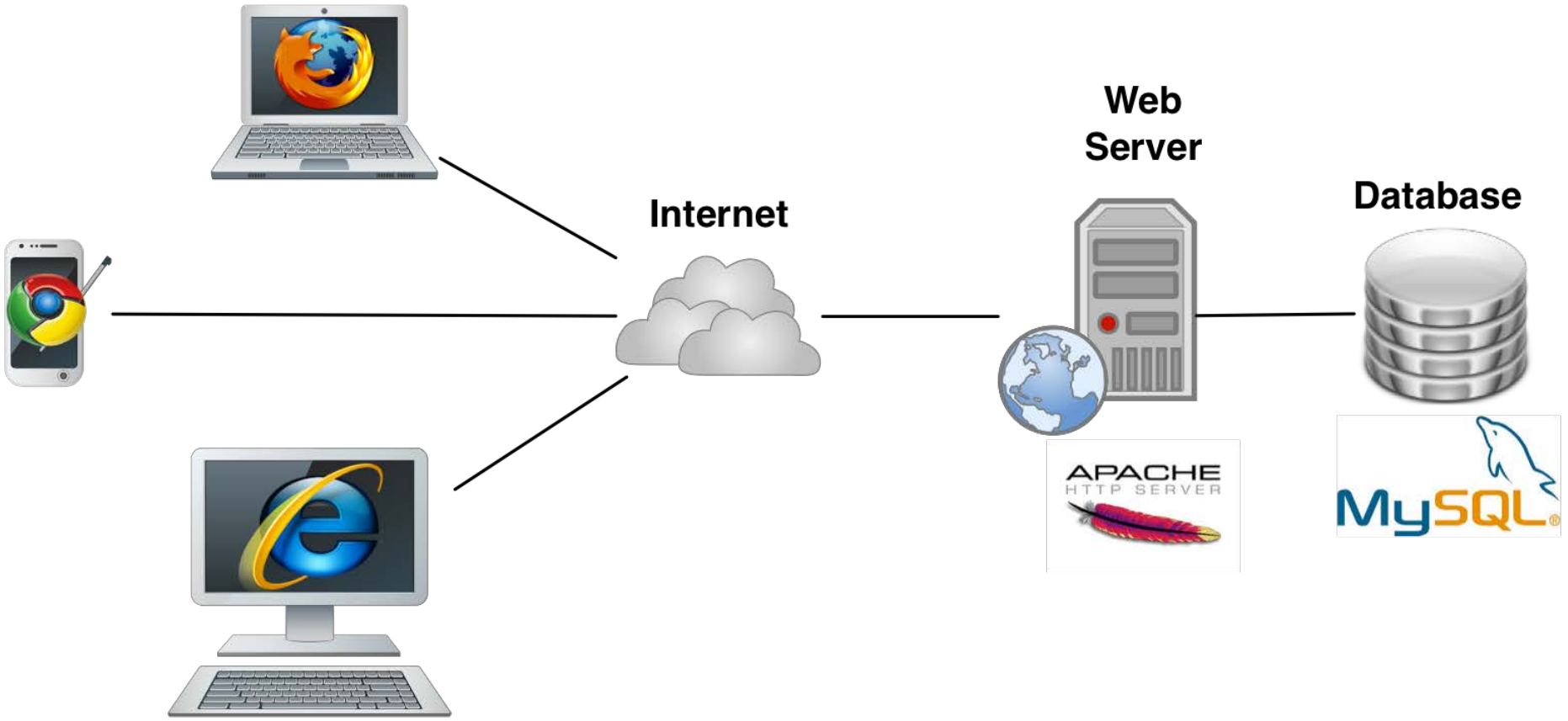
Architettura Client-Server

- Da Tecnologie Web avete imparato come sviluppare un'applicazione Web
- Tecnologie Client-side (oltre a *HTML* e *CSS*)
  - *Javascript*
  - *jQuery*
- Tecnologie Server-side
  - *PHP*
  - Alternative (Perl, Python, ASP.NET, Ruby, C#, …)
- Database
  - *MySQL* (alternative SQL server, ORACLE, …)

Architettura Client-Server



Architettura Client-Server



LAMP

- Linux
- Apache
- MySQL
- PHP

WEB APPLICATION DEVELOPMENT ENVIRONMENT

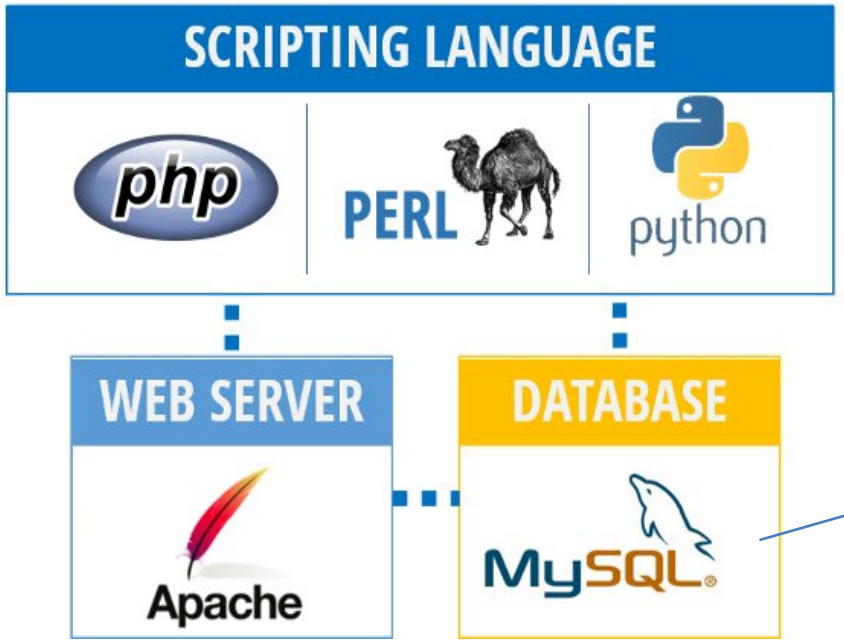
XAMPP LAMP WAMP

Cross Platform

Linux

Windows

Operating System



Oppure MariaDB

LAMP

- È molto utilizzato
- Le sue componenti sono open source
- È supportato da una grande comunità
- È personalizzabile
- Esempio:
  - **LEMP** – usa NGINX al posto di Apache for NGINX (viene usata la E, data la pronuncia "*engine-X*")
  - **LLMP** – usa *lighttpd* al posto di Apache
  - **LAPP** – usa PostgreSQL al posto di MySQL

Problemi

- Apache non è il web server più veloce
- È difficile scrivere buon codice PHP riusabile e veloce da eseguire
- Si usano linguaggi diversi per front end e back end
- Molte conversioni dei dati (ad esempio: da XML/JSON a PHP, da PHP a HTML)
- Non c'è una netta separazione nello sviluppo tra la parte client e quella server

Requisiti per un Web moderno

- Velocità e scalabilità
  - Siti Web veloci, per risposte veloci
- Nessuna pagina da ricaricare
- Tante richieste concorrenti quante richieste
- Caricamento condizionato
  - caricamento delle risorse solo quando richiesto
- Mobile/responsive UI

MEAN stack

- ***MEAN: Full-stack Javascript framework*** ottimizzato che permette di velocizzare e rendere più semplice (*user-friendly*) la creazione di applicazioni Web robuste e facilmente mantenibili

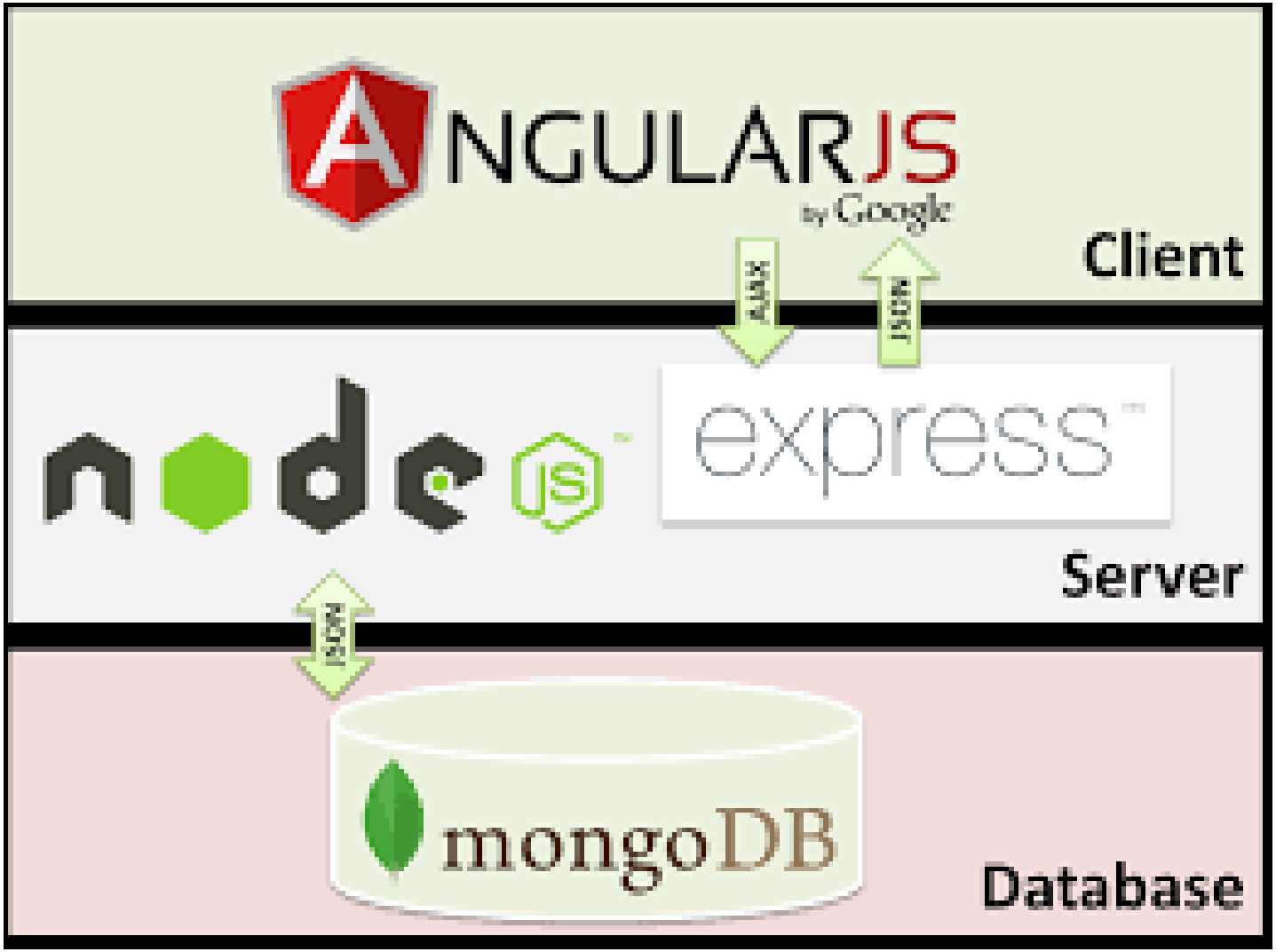
- È l'acronimo per:
  - MongoDB
  - Express.js
  - Angular
  - Node.js

Full Stack Javascript

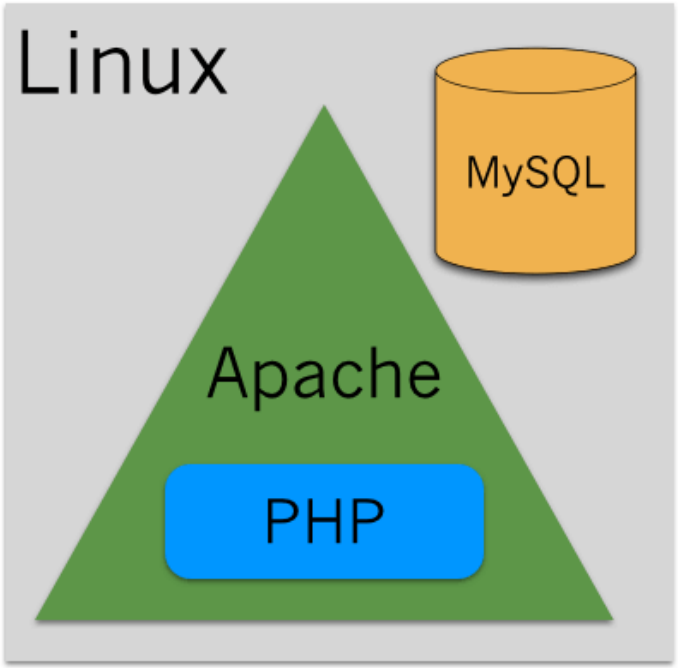
Q&A

- Ci sono domande?

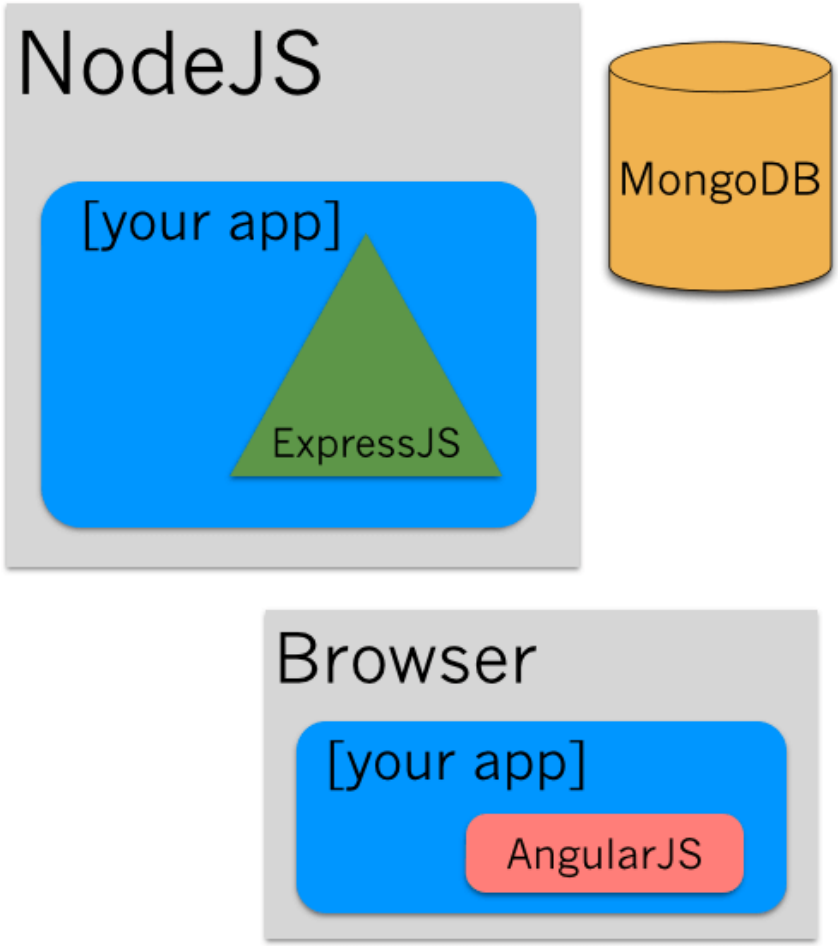
MEAN stack


ANGULARJS
by Google
Client
AJAX
JSON
express
Server
JSON
mongoDB
Database

LAMP


Linux
MySQL
Apache
PHP

MEAN

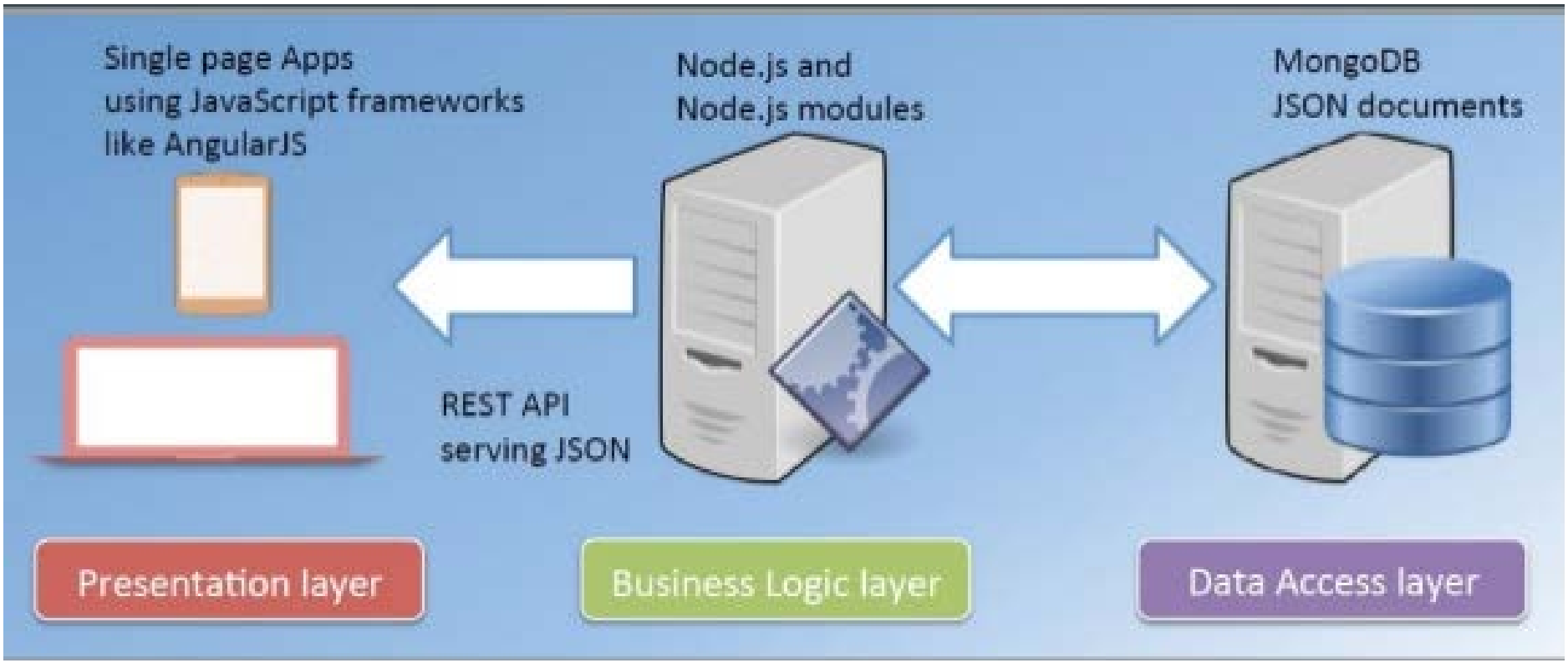

NodeJS
[your app]
ExpressJS
MongoDB
Browser
[your app]
AngularJS

Confronto MEAN-L(W)AMP

- LAMP e WAMP fanno riferimento a uno specifico sistema operativo. MEAN nasce **multipiattaforma**
- MEAN usa Node.js come ambiente di esecuzione delle applicazioni server-side, esegue codice JS al di fuori del browser (**paradigma «JavaScript everywhere»**)
- MEAN usa un DB **non relazionale** (MongoDB) al posto di MySQL, DB relazionale di L(W)AMP
- MEAN fornisce **due supporti di programmazione** (uno **client** e uno **server,** Angular e Express) basati sullo stesso linguaggio **Javascript**
- L(W)AMP si occupano di definire solo lo stack a lato server, usando per programmare PHP, Python e/o Perl

Nuovo modello di architettura

- MEAN rappresenta un cambiamento forte nel modello di architettura
  - Si passa da database relazionali a database documentali (NoSQL)
  - Si passa da pattern MVC (model-view-controller) lato server a single Web application lato client


Single page Apps
using JavaScript frameworks
like AngularJS
Node.js and
Node.js modules
MongoDB
JSON documents
REST API
serving JSON
Presentation layer
Business Logic layer
Data Access layer

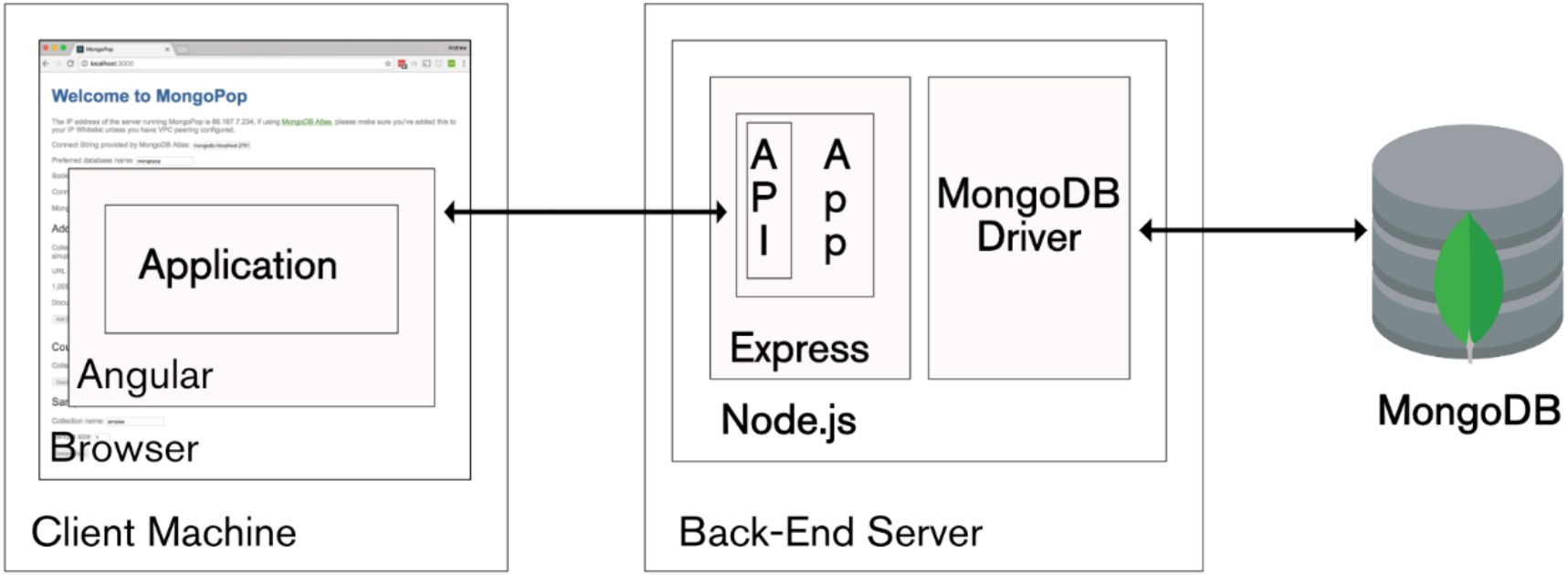
Vantaggi

- 100% open source
- 100% Javascript (+JSON +HTML)
- 100% Web standard
- Modello consistente tra back-end e front-end
- ***Uso di Javascript in tutto lo stack***
  - Javascript (il linguaggio del web)
  - JSON (data format del web)
  - Nessuna conversione necessaria per il database
- Low memory overhead
- Si può iniziare lo sviluppo con il frontend

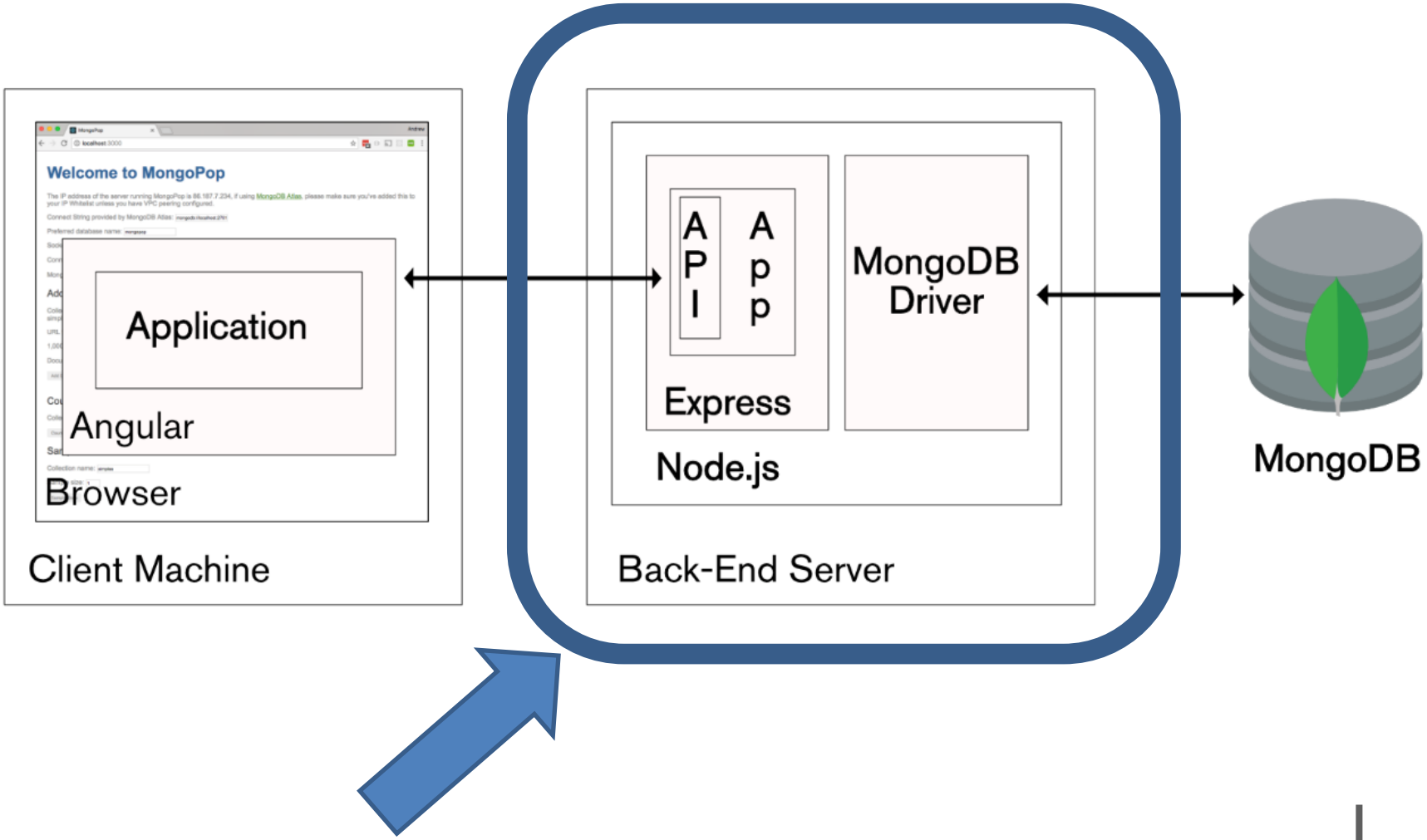
Svantaggi

- Non esistono ancora linee guida/best practice per la generazione di codice JS
  - Inoltre: Problematiche dovute al blocco nel browser dell'esecuzione di script
- MongoDB non è robusto quanto SQL server
  - In termini di sicurezza
- Una volta creato il primo sito Web con MEAN è difficile tornare indietro ai «vecchi» approcci

Logica dell'applicazione con MEAN



Logica dell'applicazione con MEAN



Q&A

- Ci sono domande?

node.js

- Una ***piattaforma software*** (non è un Web server né un linguaggio) cross-platform
  - Permette di creare il proprio Web server
  - Si possono creare Web application sopra di esso
- Costruito sulla base di Google Chrome V8 javascript engine
  - Esegue javascript server side
- Single-threaded
- Event driven architecture
- Esecuzione asincrona
- Non blocking I/O model

Link: https://nodejs.org/

node.js

- Node.js permette di avere Web application con connessioni real-time, two-way, dove non solo il client, ma anche il server può iniziare una connessione, permettendo di trasferire dati liberamente
  - Ciò è completamente in contrasto con applicazioni Web tradizionali

Benefici

- Permette di scrivere codice in Javascript
- Estremamente veloce e leggero
- Uso efficiente delle risorse
- Non c’è bisogno di eseguire un Web server separato
- Permette di avere un forte controllo della logica dell’app e dell’ambiente
- Permette di gestire diversi utenti con poche risorse

Dal sito ufficiale …

- *«Node.js è un runtime Javascript costruito sul **motore JavaScript V8** di Chrome»*
- *«Node.js usa un **modello** I/O non bloccante e ad eventi, che lo rende un framework leggero ed efficiente»*
- *«L'ecosistema dei pacchetti di Node.js, **npm**, è il più grande ecosistema di librerie open source al mondo»*

Elementi principali

- Motore JavaScript V8
- Modello di Node.js
- NPM

Dal sito ufficiale …

- *«Node.js è un runtime Javascript costruito sul **motore JavaScript V8** di Chrome»*
- *«Node.js usa un **modello** I/O non bloccante e ad eventi, che lo rende un framework leggero ed efficiente»*
- *«L'ecosistema dei pacchetti di Node.js, **npm**, è il più grande ecosistema di librerie open source al mondo»*

Motore JavaScript V8

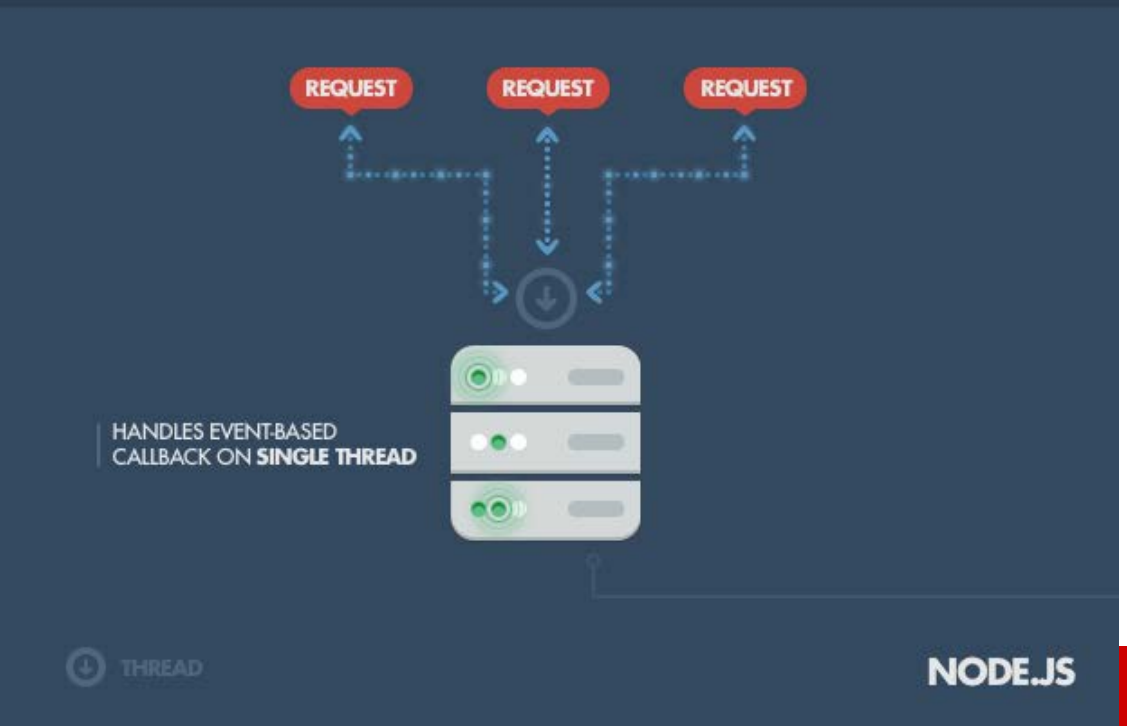
- V8 è un motore open source high-performance JavaScript
- È sviluppato da Google (infatti è utilizzato in Chrome e Node.js)
- È scritto in C++
- È multipiattaforma
- Supporta ECMAScript
- È attualmente incluso in Google Chrome
- Compilazione «Just-In-Time» di Javascript in linguaggio macchina, senza codice intermedio
  - Converte il codice Javascript in linguaggio macchina di basso livello o codice macchina che i microprocessori possono capire

Q&A

- Ci sono domande?

Dal sito ufficiale …

- *«Node.js è un runtime Javascript costruito sul motore JavaScript V8 di Chrome»*
- *«Node.js usa un **modello** I/O non bloccante e ad eventi, che lo rende un framework leggero ed efficiente»*
- *«L'ecosistema dei pacchetti di Node.js, npm, è il più grande ecosistema di librerie open source al mondo»*

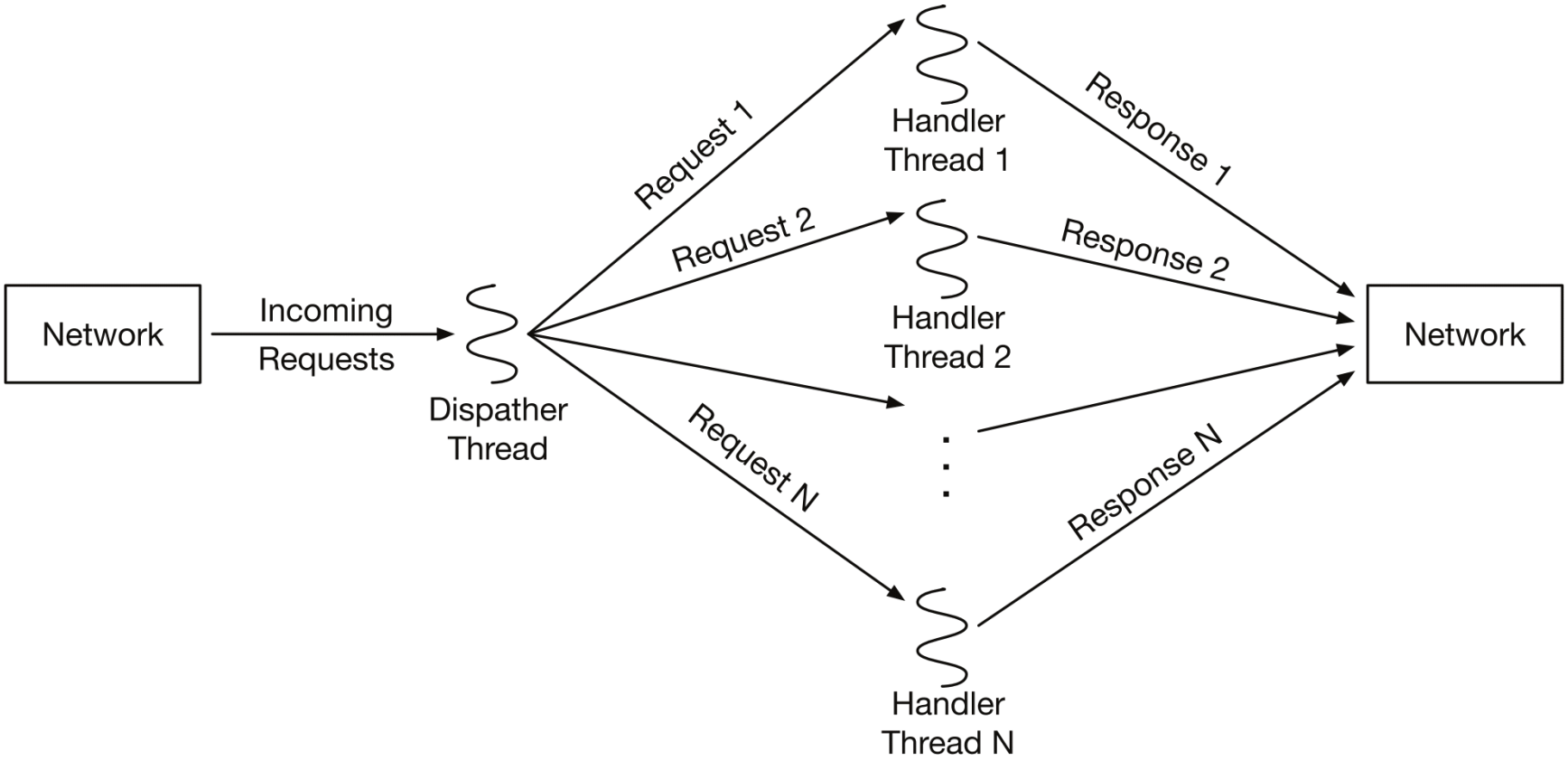


Traditional (Thread Based) VS Single Thread (Event-Driven)

Modello Thread-based (Traditional)

- Un dispatcher thread assegna ogni richiesta entrante a un worker thread per l'elaborazione
- In un dato momento, il numero di thread è uguale al numero di richieste
- Le operazioni di I/O sono bloccanti

Modello Thread-based (traditional)



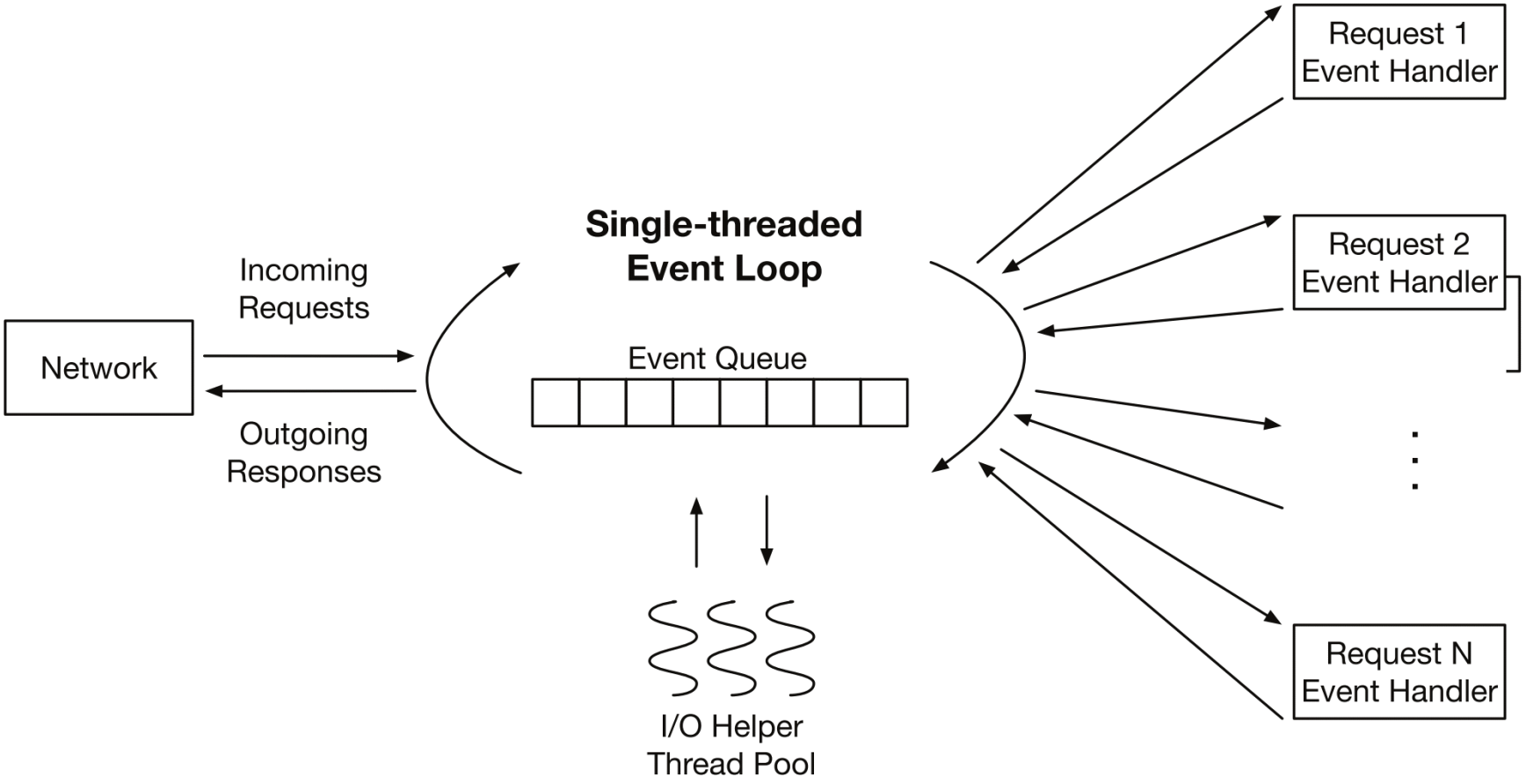
Svantaggi del modello Thread-based

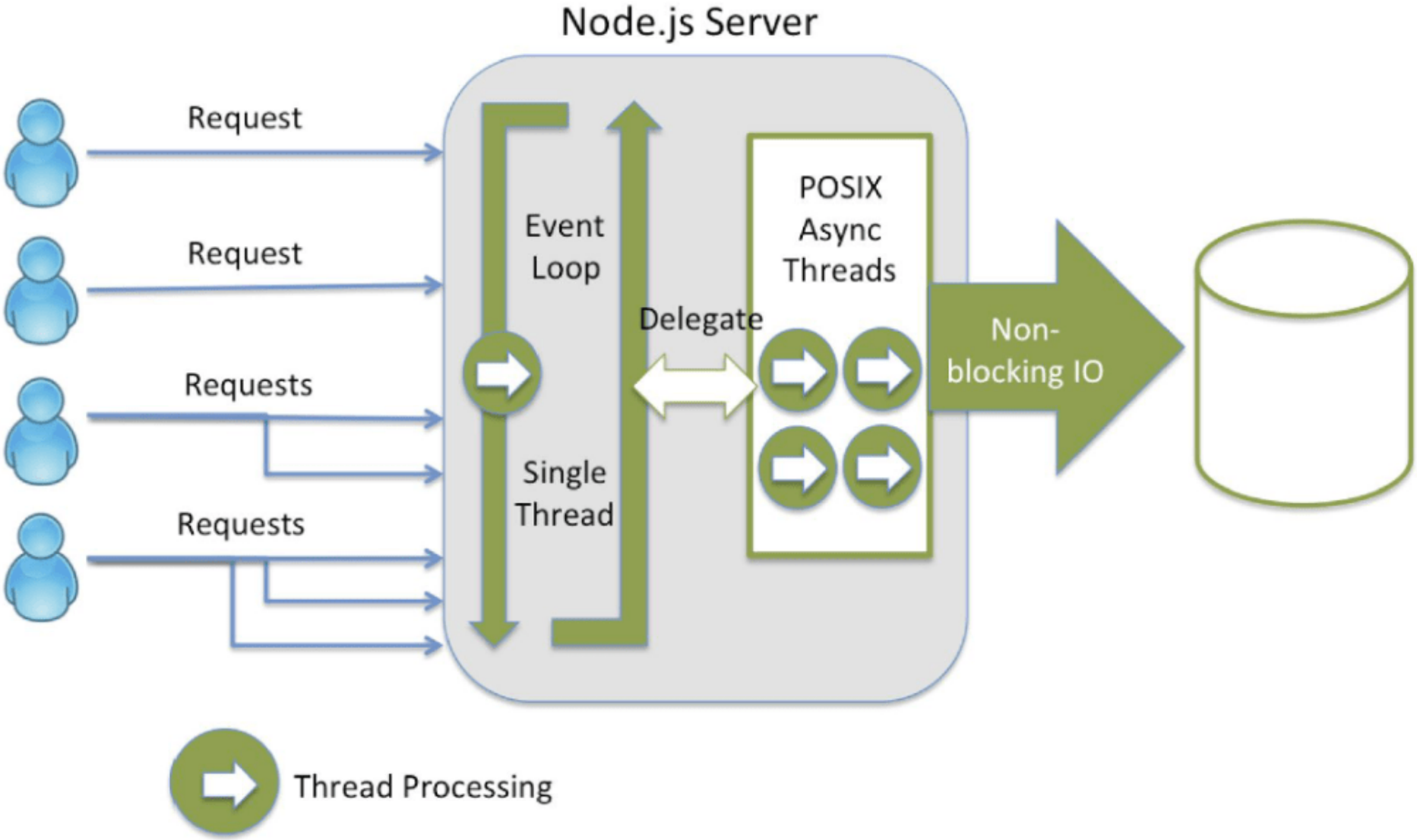
- Al crescere delle richieste, aumenta il numero di thread attivi. Limitazioni:
  - Overhead del sistema operativo dovuto a context switching
  - I sistemi operativi hanno un limite di thread massimo
- Questo limita notevolmente la scalabilità del server

Node.js -> Modello Event-Driven

- Le richieste vengono tradotte in eventi (ognuno dei quali è associato a un *event handler*) e inserite in una coda
- C'è un *event loop* ***single-threaded*** nel quale, ad ogni iterazione, si controlla se ci sono nuovi eventi nella coda e si esegue il relativo handler
- Si utilizzano dei thread aggiuntivi per le operazioni di I/O che vengono eseguite in maniera asincrona

Modello Event-Driven





Node.js runs single-threaded, non-blocking, and asynchronously

Logica Asincrona

- Tutte le richieste vengono servite da un singolo thread in modo concorrente e con logica asincrona
  - ***ATTENZIONE***: questo significa che occorre fare attenzione a come quel singolo thread viene gestito, perché una cattiva gestione può causare problemi a tutti i client
  - Per esempio: uno sleep si ripercuote sull'intero sistema
- Non si aspettano i risultati
  - Quando questi saranno pronti verranno inseriti nell'event loop

Logica Asincrona

- Ricapitolando
  - Non esegue un nuovo thread quando una richiesta arriva, ma inizia invece ad eseguire in modo asincrono
    - quindi è sempre pronto a gestire nuove richieste
  - Ciò permette di consumare meno memoria e permette al server di essere più veloce

Why the Hell Would You Use Node.js

- Per le sue caratteristiche Node.js è una piattaforma altamente indicata per data-intensive real-time application, che eseguono su diversi device distribuiti
  - Queste applicazioni vengono chiamate DIRT Application (Data-Intensive Real-Time Application)
- Infatti, node.js è capace di gestire un elevato numero di connessioni simultanee → elevata scalabilità

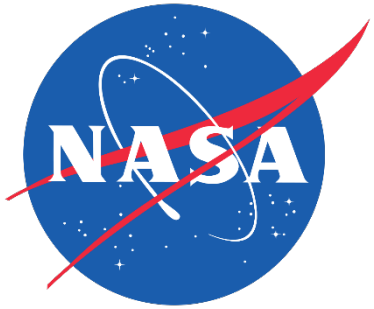
Quando usare Node.js

- Node.js si è affermato come principale soluzione nello sviluppo delle DIRT Application
  Esempi:
  - Trasmissione di contenuti multimediali in streaming
  - Videogiochi online
  - Chat
  - Sistemi di monitoraggio con IoT e smart object

Quando NON usare Node.js

- Applicazioni CPU-intensive
  - Computazioni lunghe e complesse
- Server per la gestione di file statici

Chi usa Node.js



Q&A

- Ci sono domande?

Dal sito ufficiale…

- *«Node.js è un runtime Javascript costruito sul motore JavaScript V8 di Chrome»*
- *«Node.js usa un modello I/O non bloccante e ad eventi, che lo rende un framework leggero ed efficiente»*
- *«L'ecosistema dei pacchetti di Node.js, **npm**, è il più grande ecosistema di librerie open source al mondo»*

Node Package Manager

- NPM è l'acronimo di Node Package Manager
- *npmjs.com* ospita migliaia di pacchetti gratuiti che è possibile scaricare ed usare
- Un pacchetto contiene tutti i file necessari per utilizzare un modulo
- Un modulo è una libreria che può essere inclusa nel progetto
- NPM viene installato direttamente con Node
- Permette agli utenti di «consumare»/distribuire moduli (pacchetto)

Installare Node.js

- https://nodejs.org/it/download/
- Latest LTS Version: **14.16.0** (includes npm **6.14.11**)
- Disponibili:
  - Codici sorgenti e pre-built installer per Windows, macOS e Linux
  - Altre piattaforme (SunOS, Docker Image, ecc)
- Per verificare l'installazione
  - node -v

Installare tramite package manager

- https://nodejs.org/en/download/package-manager/

Comandi Node

- È possibile richiamare node.js dalla console con il comando:

```
node <flag> <script> <argomenti script>
```

- Con il comando *node*, senza argomenti, si richiamerà Node.js in modalità interattiva

Principali comandi NPM

- **npm install <nome pacchetto>[@1.0] [--save] [-dev/-optional/-exact] [--global]**
- **npm update [--save] [-dev/-optional] [--global]**
- **npm list <nome pacchetto>**
- **npm uninstall <nome pacchetto> [--save] [-dev/-optional]**
- **npm publish**

Node Hello World

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

http.createServer(function (req, res) {
    res.writeHead(200, {'Content-Type':
'text/plain'});
    res.end('Hello World!');
    console.log("Response sent");
}).listen(8080);
console.log("A node web server is running!");
```

Q&A

- Ci sono domande?

Deploy

“If #nginx isn’t sitting in front of your node server, you’re probably doing it wrong”

- Bryan Hughes via Twitter

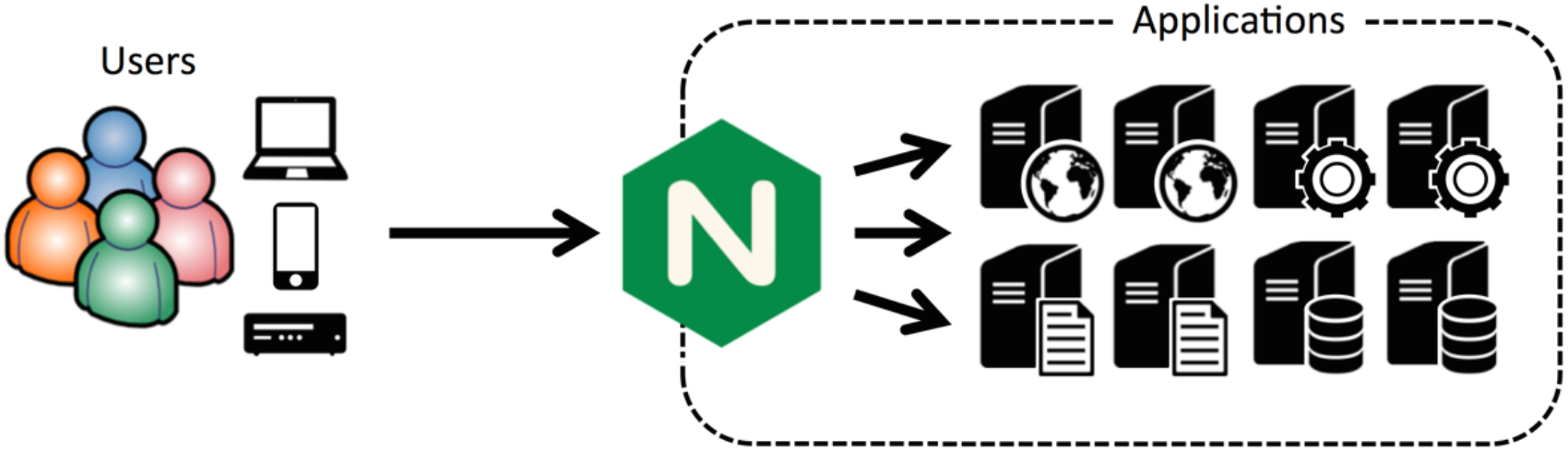
- Esempio di errore:
  - https://stackoverflow.com/questions/37242654/nginx-and-node-js-am-i-doing-it-wrong

NGINX

- È un **web server**
- Può essere usato anche come:
  - Reverse proxy
  - Load balancer
  - HTTP cache
- È gratuito e open source

NGINX e Node.js

- Si affianca a Node.js, nginx come Reverse Proxy Server



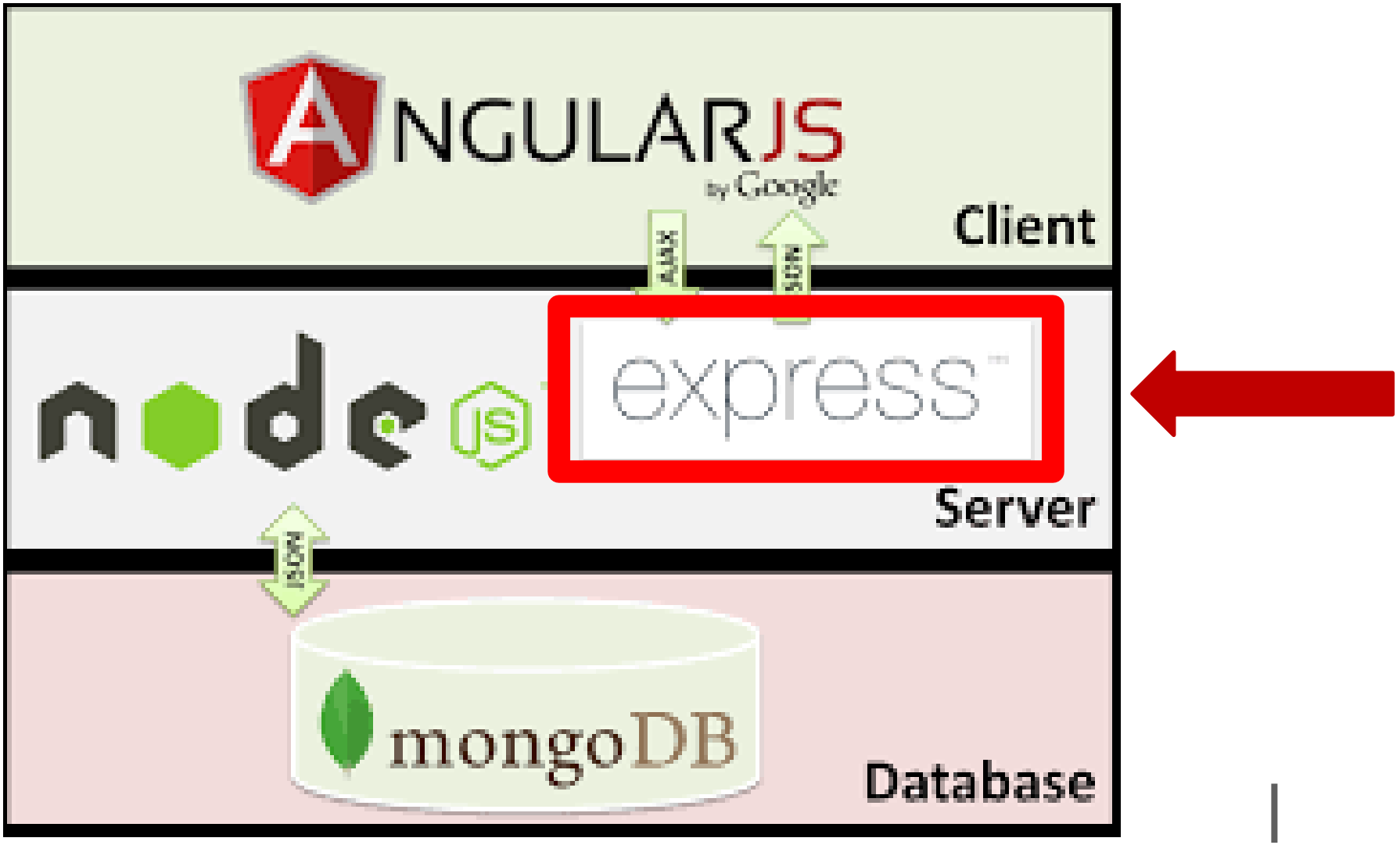
Reverse Proxy Server

- È un proxy server che solitamente si trova dietro il firewall in una rete privata e dirige le richieste dei client all'appropriato backend server
- È un ulteriore livello di astrazione
- Spesso usato per:
  - Load balancing
  - Web accellerator
  - Security and anonymity

Perché usare un Reverse Proxy?

- Semplificare la gestione di privilegi e porte
- Servire in modo più efficiente i file statici
- Cache dei file statici
- Nascondere l'identità del server Node.js
  - Migliore gestione dei crash
  - Mitigare attacchi DoS
- Load balancing

Express


ANGULARJS
by Google
Client
AJAX
JSON
express
Server
JSON
mongoDB
Database

Express

Express 4.16.4

Fast, unopinionated, minimalist web framework for Node.js

```
$ npm install express --save
```

Express

- È un Node.js web application framework server side, minimale e flessibile
- Fornisce un insieme di feature per costruire applicazioni web (e mobile):
  - Facilita l'uso di Node.js
  - ***Facilita l'implementazione di API REST***
  - Supporta diversi template engine
  - Il layer di funzionalità fornito non «oscura» le funzionalità di node.js

express

Installazione

- Avendo già installato node.js

```
$ npm install express --save
```

Express - Funzionalità principali

- Consente di definire middleware per rispondere a richieste HTTP
- Consente di definire una tabella di routing per eseguire azioni differenti in base a:
  - Metodo HTTP
  - URL

Express Hello World

```
var express = require('express');
var app = express();

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

app.listen(3000, function () {
  console.log('Example app listening on port
3000!');
});
```

Express Generator

- È possibile utilizzare lo strumento di creazione dell'applicazione, *Express*, per creare velocemente una struttura dell'applicazione
- Installazione:
  `npm install express-generator`
- Esempio utilizzo:
  `express --view=pug myapp`

Express Generator

Node | Express

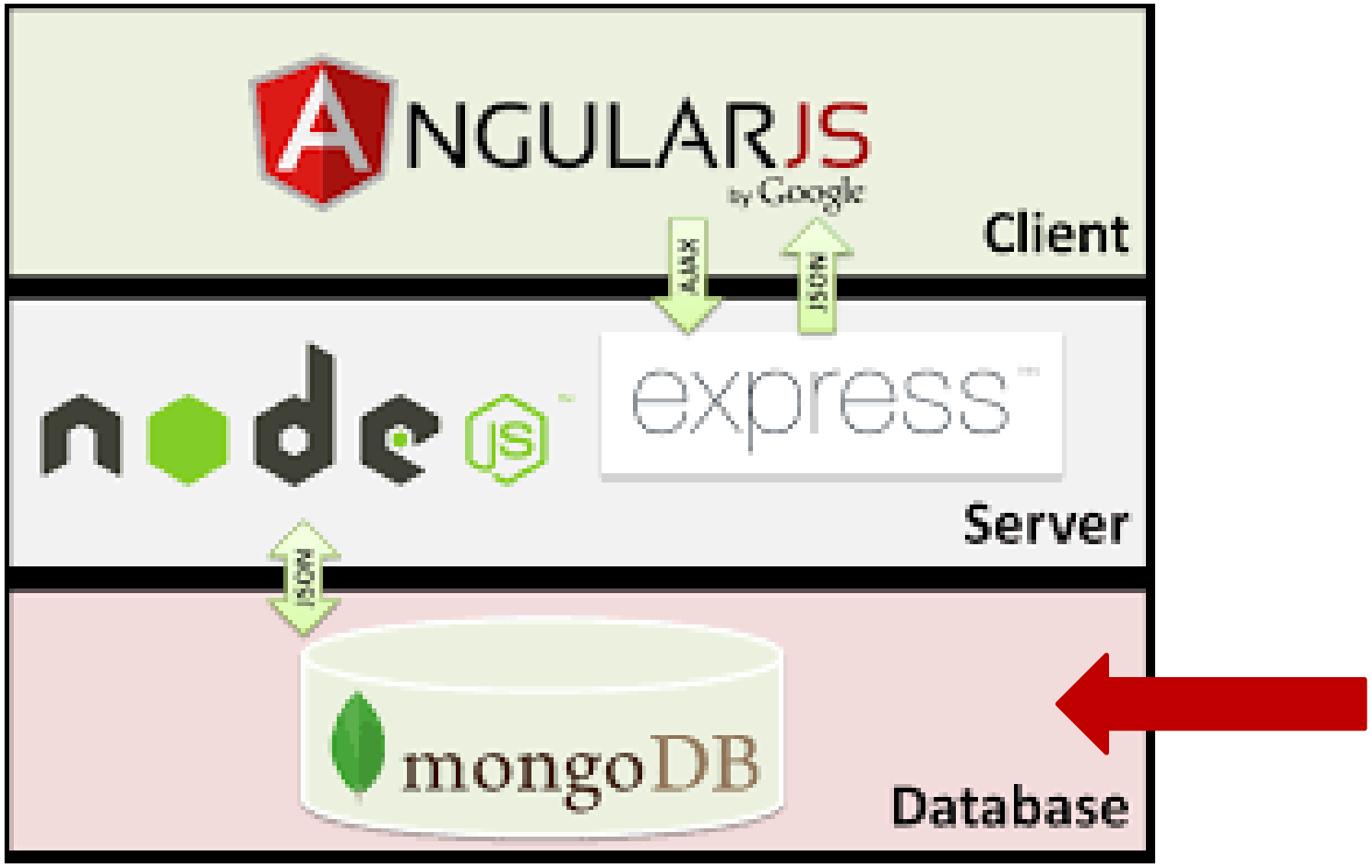
```
var http =
require('http');


http.createServer(functio
n (req, res) {
    res.writeHead(200,
{'Content-Type':
'text/plain'});
    res.end('Hello
World!');
    console.log("Response
sent");
}).listen(8080);
console.log("A node web
server is running!");
```

```
var express =
require('express');
var app = express();

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

app.listen(3000, function
() {
  console.log('Example app
listening on port 3000!');
});
```

Q&A

- Ci sono domande?

mongoDB


ANGULARJS
by Google
Client
AJAX
JSON
express
Server
JSON
mongoDB
Database

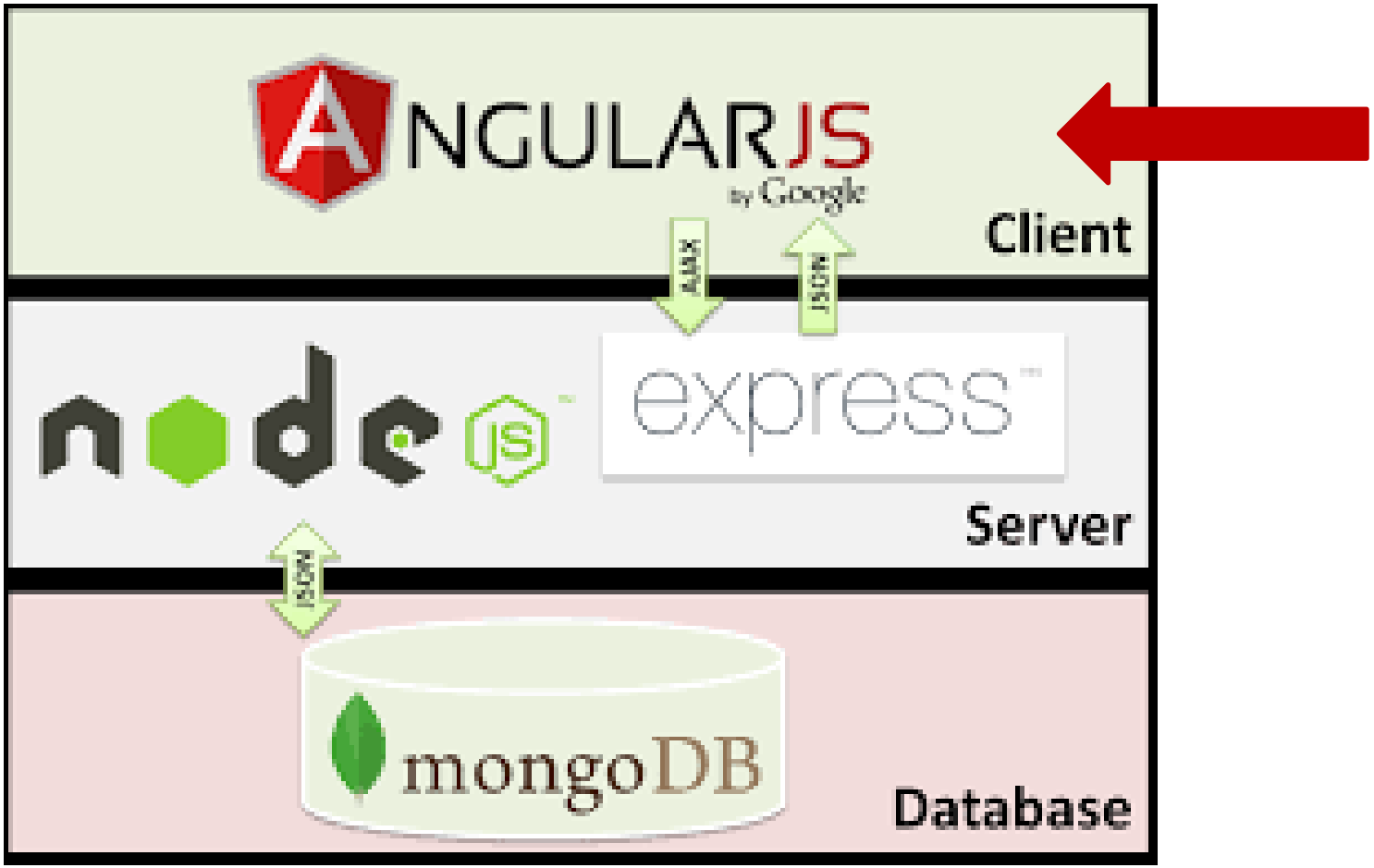
mongoDB

- È un database NON relazionale, classificato come NoSql database
  - No solite strutture a tabelle relazionali MA json-like document con schema dinamico
- È orientato ai documenti (document-oriented)
- È cross-platform

Confronto

MongoDB	RDBMS
Collection	Table
Document	Row
Index	Index
Embedded Document	Join
Reference	Foreign Key

Angular!?


ANGULARJS
by Google
Client
AJAX
JSON
express
Server
JSON
mongoDB
Database

Angular

- Framework JavaScript open source
- Mantenuto da Google
- Permette di creare applicazioni single-page
- …. (lo vedremo in dettaglio)

Angular … Per forza?

- Non necessariamente!
- Infatti noi vedremo non solo MEAN, ma anche le sue varianti «MEVN» (con VUE al posto di Angular) e «MERN» (con React al posto di Angular)
- Vedremo anche il confronto tra MEAN (con Angular) e le sue varianti MEVN (con VUE) e MERN (con React)

Signature Stack

- MEAN, LAMP, WAMP sono solution stack ***standard***
- E' possibile sempre usare versioni «ibride» dei solution stack, dette anche «*signature stack*»
- Ad esempio: Uber, Facebook, Reddit, Pinterest utilizzano un proprio signature stack, con varianti differenti

Uber Tech Stack

- Web server: NGINX, Apache
- Databases: MySQL, PostgreSQL, MongoDB, Cassandra
- Server-side framework: Node.js
- Programming languages: Python, Java, JavaScript, Objective-C

Reddit Tech Stack

- Server: NGINX
- Databases: PostgreSQL, Redis, Cassandra
- Server-side framework: Node.js
- Programming languages: JavaScript, Python

Pinterest Tech Stack

- Programming Languages: Python, Java, Go
- Framework: Django, Javascript MVC
- Databases: MySQL, Hadoop, HBase, Memcached, Redis
- Server: NGINX

Facebook Tech Stack

- Programming Languages: PHP, GraphQL, Hack
- Framework: Tornado
- Databases: Cassandra, RocksDB, Beringei, Memcached
- Server: custom/proprietary

Airbnb Tech Stack

- Programming Languages: JavaScript, Ruby, Java, Sass
- Framework: Rails
- Databases: MySQL, Amazon RDS, Hadoop, Redis
- Server: NGINX

Q&A

- Ci sono domande?

Link utili

- http://mean.io/
- https://nodejs.org/
- https://medium.com/the-node-js-collection/why-the-hell-would-you-use-node-js-4b053b94ab8e
- https://www.npmjs.com/
- https://expressjs.com/
- https://stackshare.io/feed