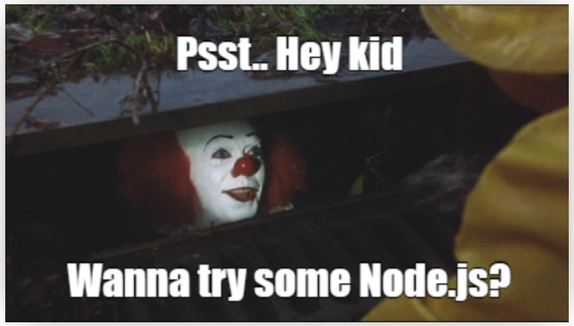
# Introduzione a Node.js

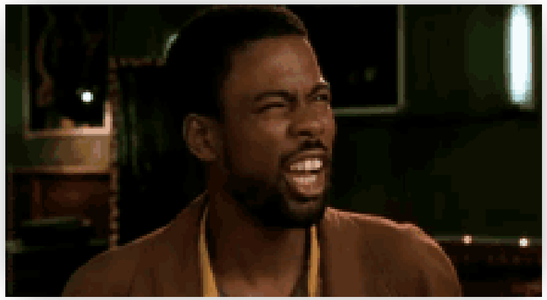
Che cos'è e perchè dovresti usarlo

# Che cos'è Node.js



# Cos'è Node.js

**Node è un runtime evironment per eseguire javascript al di fuori dai browser**



# Cosa NON è Node.js

~~Un linguaggio di programmazione~~

~~Un framework~~

# Cosa posso fare con Node.js?

YEOMAN

Gulp

{ REST }

- Utilities di sviluppo (Grunt, Gulp, Yeoman ecc)
- Webserver
- API Rest

# Perchè usare Node.js?

- Codebase più consistente
- Ecosistema di librerie gratuite con NPM
- Division of concerns maggiore

# Chi usa Node.js?

# Chi usa Node.js?



Grandi aziende che usano Node

PayPal Uber LinkedIn ebay Walmart

# Case study Paypal

- Costruito due volte più velocemente, con meno persone
- 33% linee di codice in meno
- 40% di files in meno
- Ha servito il doppio delle richieste al secondo
- 35% response time in meno
- In media utenti serviti 200ms più velocemente

# Come funziona Node.js?

# Come funziona Node.js?



Node è basato sul JavaScript Engine di Google Chrome (V8), embeddato all'interno di un programma C++

Creato da Ryan Dahl nel 2009

# Come funziona Node.js?



## Non presenti in Node.js

```
document.getElementById();
window.alert();
```

# Come funziona Node.js?



## Presenti in Node.js

```
const fs = require('fs')

fs.readFile("file.json",(error,data) => {
  (!error)?console.log(data):console.log("error: ", error)
});
```

# Come funziona Node.js?



```
const http = require('http');
const hostname = '127.0.0.1';
const port = 3000;

const server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hello World\n');
});

server.listen(port, hostname, () => {
  console.log(`Server running at http://${hostname}:${port}/`);
});
```

# Come funziona Node.js?



Ha una struttura Asincrona e non bloccante

# Come funziona Node.js?



**Immaginiamo di entrare in un ristorante...**

# Come funziona Node.js?

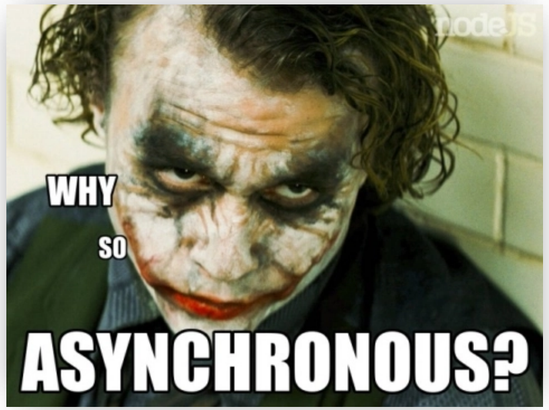


Ci sediamo al tavolo e la cameriera viene a prenderci l'ordine. Una volta preso l'ordine la cameriera porta la comanda in cucina e aspetta finchè l'ordine non è stato preparato del tutto, quindi ci porta la nostra ordinazione e passa al cliente successivo.

# Server PHP

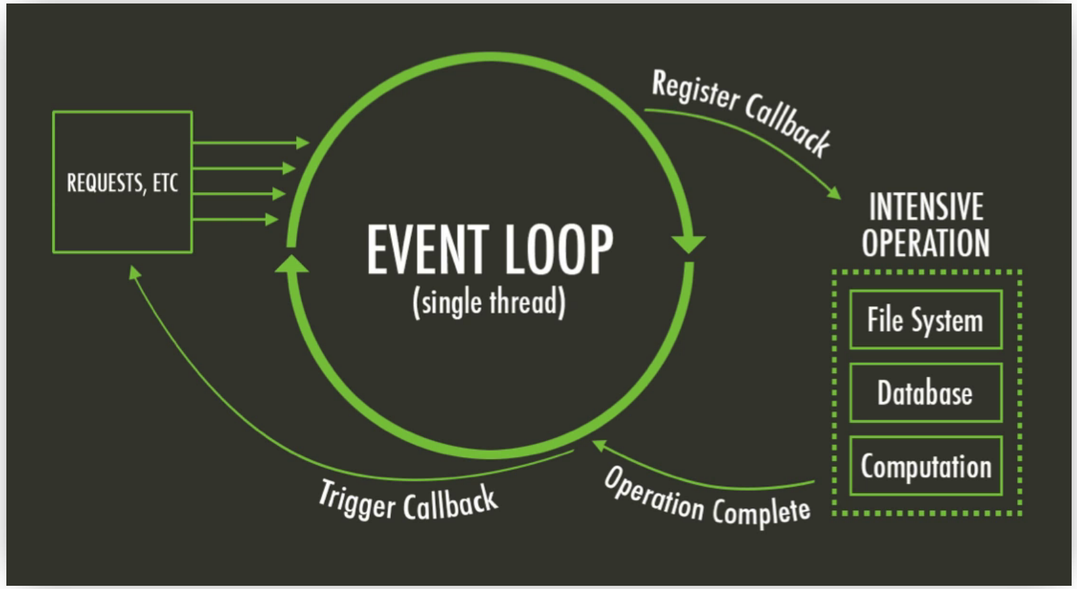
- Un client esegue una richiesta
- Viene allocato un thread e il codice viene eseguito
- C'è una chiamata ad un db
- Aspettiamo la risposta del db
- Il thread viene rilasciato solamente nel momento in cui l'esecuzione del codice è terminata.

# Server Node.js



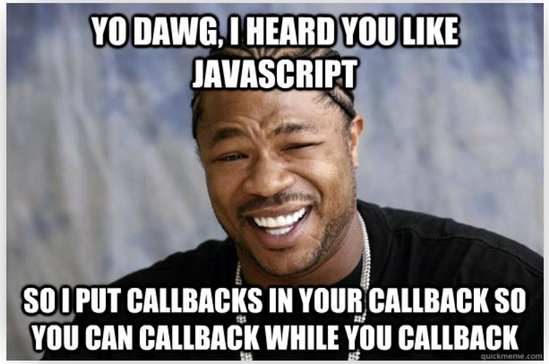

- Un client esegue una richiesta
- Il codice viene eseguito sul thread di Node.js
- C'è una chiamata ad un db
- Effettuo la chiamata e registro un evento (callback) per quando il db risponderà
- Libero il thread e servo eventuali altre richieste
- Il db risponde
- Il thread viene occupato dalla prima richiesta e esegue ciò che è indicato nella callback
- Il thread viene liberato

# Event Queue


REQUESTS, ETC
EVENT LOOP
(single thread)
Register Callback
INTENSIVE OPERATION
File System
Database
Computation
Trigger Callback
Operation Complete

# Come funziona Node.js?






Questa struttura asincrona fa si che Node.js sia perfetto per sviluppare applicazione I/O intensive, applicazioni in real time e web server altamente scalabili che possono servire un grosso numero di uteni contemporaneamente.

# Cosè NPM

# NPM



## NPM è un gestore di pacchetti per JavaScript

Permette di gestire versioni, aggiungere rimuovere pacchetti, tutti i pacchetti vengono salvati nella cartella **node_modules**

# Alcuni pacchetti per Node.js

- Express - Framework completo per sviluppo di applicazioni web
- Mongoose - Wrapper per interagire con MongoDB (NoSQL database)
- Redis - Chache manager e NoSQL database
- Socket.io - Componente server-side per interagire con web sockets
- Nodemon - Utility che effettua auto reload del server in caso di cambiamenti
- PM2 -Process manager e load balancer

# Cos'è Express

# Express



**Express è un framework per applicazioni web Node.js flessibile e leggero che fornisce una serie di funzioni avanzate per le applicazioni web e per dispositivi mobili.**

# Elementi fondamentali di Express

# Routes

I percorsi di route, in combinazione con un metodo di richiesta, definiscono gli endpoint a cui possono essere rivolte richieste. I percorsi di route possono essere stringhe, modelli di stringa o espressioni regolari.

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

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

app.listen(3000);
```

# Middlewares

Un middleware è una funzione che viene richiamata dal livello di routing Express prima dell'handler di richiesta finale, pertanto si trova al centro tra una richiesta base e la route prevista.

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

app.use(function (req, res, next) {
  console.log('Hi! I\'m a middleware!');
  next();
});

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

app.listen(3000);
```

# Cos'è PM2

# PM2



**PM2 è un process manager avanzato per Node.js**

# Gestione delle istanze di Node.js

PM2 permette di avviare, stoppare e gestire le istanze di Node.js. Questo comprende anche il monitoraggio delle istanze, ossia il riavvio delle istanze nel caso di cambiamenti al filesystem o in caso di errori fatali.

# Load balancing e altro

PM2 inoltre permette di avviare più istanze di node contemporaneamente, in maniera **cluster**, ossia spalmato su più CPUs. Permette anche la gestione delle variabili di **environment**

# File di configurazione

Tutto questo viene condensato in un file di configurazione (YML o JSON)

```yaml
apps:
  - script   : ./app.js
    name     : 'server'
    instances: max
    exec_mode: cluster
    watch: true
    env    :
      NODE_ENV: development
    env_production:
      NODE_ENV: production
    port: 3000
```

# Grazie