# NodeJS

## Training Agenda

- JavaScript Overview
- NodeJS Overview
- Modules System
- File System
- Streams & Event
- Express
- Data Persistence
- View Engines
- NodeJS Securities
- Unit Testing

## Prerequisites:

- Basic knowledge of HTML, CSS & JavaScript
- Interest to Learn

## JavaScript Overview

LIGHT-
WEIGHT

SCRIPTING
LANGUAGE

INTERPRETER
BASED

EVENT
DRIVEN

SINGLE
THREADED

# JavaScript Building Blocks

## Functions

- Function Expressions
- HOF
- Nested Functions
- IIFE

## Objects Creation Methods

- Literal
- Constructor
- Instance

## NodeJS

“Node.js is a platform built on Chrome's JavaScript runtime for easily building fast and scalable network applications. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices.”

## NodeJS : Features

- Extremely fast
- I/O is Asynchronous
- Event Driven
- Single threaded
- Highly Scalable
- Open source

## NodeJS Process Model

Node.js runs in a single process and the application code runs in a single thread.

All the user requests to web application will be handled by a single thread and all the I/O work or long running job is performed asynchronously for a request.

An event loop is constantly watching for the events to be raised for an asynchronous job and executing callback function when the job completes.

Internally, Node.js uses libuv for the event loop which in turn uses internal C++ thread pool to provide asynchronous I/O.

# NodeJS : Module System

**Use of imports/exports for importing and exporting the modules in application.**

**Following are a few salient points of the module system:**

Each file is its own module.

Each file has access to the current module definition using the module variable.

The export of the current module is determined by the module.exports variable.

To import a module, use the globally available require function.

## NodeJS : Module System

- If something is a core module, return it.
- If something is a relative path (starts with ‘./’ , ‘../’) return that file OR folder.
- If not, look for node_modules/filename or node_modules/foldername each level up until you find a file OR folder that matches something.
- If it matched a file name, return it. If it matched a folder name and it has package.json with main, return that file.
- If it matched a folder name and it has an index file, return it.

# NodeJS : Require Function

The Node.js require function is the main way of importing a module.

The require function blocks further code execution until the module has been loaded.

Call require() based on some condition and therefore load the module on-demand

After the first time a require call is made to a particular file, the module.exports is cached.

Module allows you to share in-memory objects between modules.

Treats module as an object factory

## NodeJS : Few Core Modules

- path
- os
- event
- utils
- http
- fs
- and many more

## Task – Notes App

- **Notes App should provide following features -**
  - User should be able to run command from command prompt to create, read, delete and list the notes.
  - Should use local file storage
  - Should use terminal colors

- **Commands to run -**

```
> node index.js add --title="New Title" --body="New Title Body"
> node index.js read --title="Some Title"
> node index.js remove --title="Some Title"
> node index.js list
```

# NodeJS : Built-in Global Variables

**Console**

The console plays an important part in quickly showing what is happening in your application when you need to debug it.

**Timers**

setTimeout only executes the callback function once after the specified duration. But setInterval calls the callback repeatedly after every passing of the specified duration.

**__filename and __dirname**

These variables are available in each file and give you the full path to the file and directory for the current module.

**Process**

Use the process object to access the command line arguments. Used to put the callback into the next cycle of the Node.js event loop.

**Buffer**

Native and fast support to handle binary data

## Node Package Manager

NPM is the eco-system to manage the project dependencies

| Commands | Description |
|---|---|
| npm install | Install the packages/ dependencies |
| npm uninstall | Uninstall the packages/dependencies |
| npm config get/set | Get /set the npm eco-system |
| npm update | Update the project dependencies |
| npm ls | List down the dependencies |
| npm search | Search the listed package on npm registry |
| npm init | Generates package.json file in local project directory |
| npm outdated | List down the outdated package |

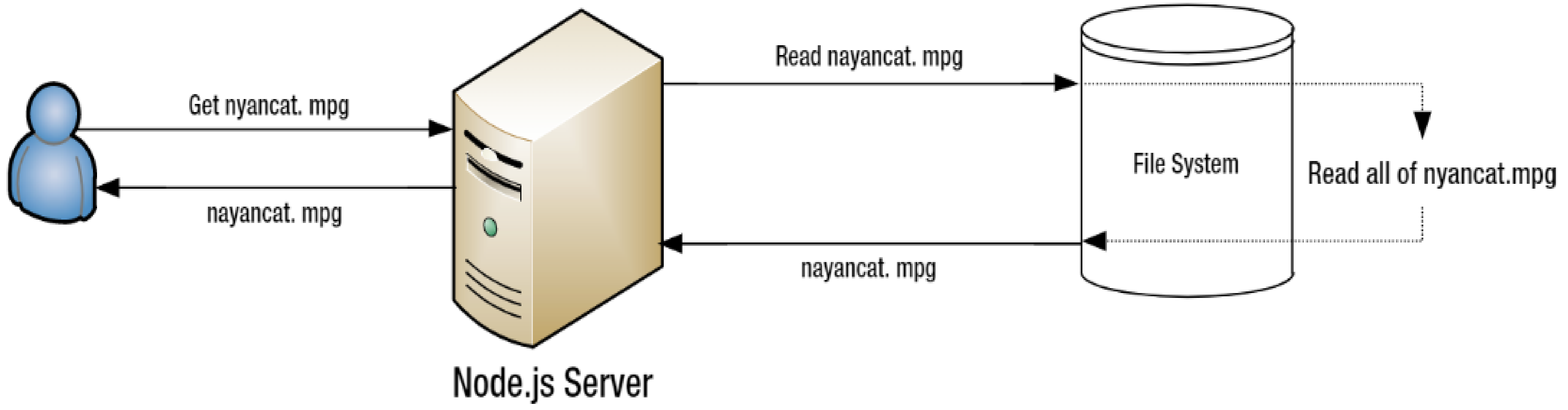
## NodeJS : Event System

- EventEmitter is a class designed to make it easy to emit events and subscribe to raised events.
  - Subscribing :
    - built-in support for multiple subscribers is one of the advantages of using events.
  - Unsubscribing :
    - EventEmitter has a removeListener function that takes an event name followed by a function object to remove from the listening queue.
- EventEmitter provides a function `once` that calls the registered listener only once.
- EventEmitter has a member function, listeners, that takes an event name and returns all the listeners subscribed to that event.
- Memory Leak : A common cause of memory leak is forgetting to unsubscribe for the event.
- A number of classes inside core Node.js inherit from EventEmitter.

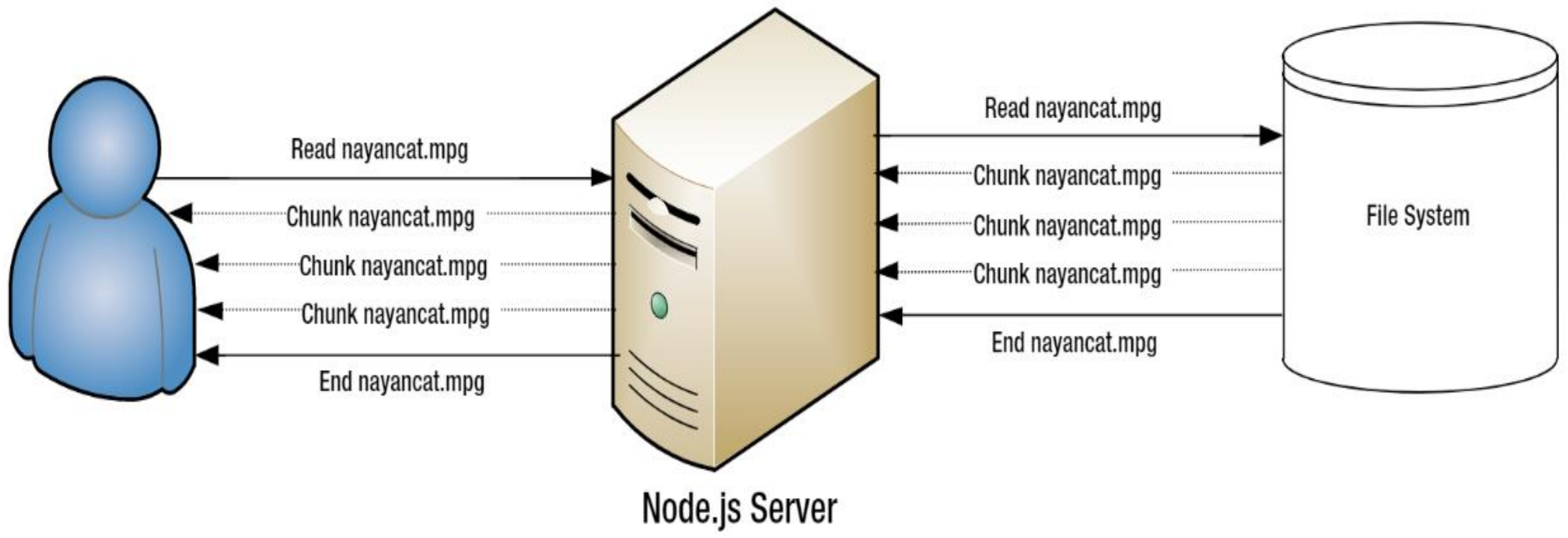
## Buffers & Streams

- Streams play an important role in creating performant web applications.
- Main motivation behind steams -
  - Improvement in user experience
  - better utilization of server resources
- All of the stream classes inherit from a base abstract Stream class which in turn inherits from EventEmitter.
- All the streams support a pipe operation that can be done using the pipe member function.

**Buffered Web Response**



**Streaming Web Response**



# NodeJS : Streams (Cntd...)

| | |
|---|---|
| Readable | A readable stream is one that you can read data from but not write to.<br>Process. stdin, which can be used to stream data from the standard input. |
| Writable | A writable stream is one that you can write to but not read from.<br>Process.stdout, which can be used to stream data to the standard output. |
| Transform | A transform stream is a special case of a duplex stream where the output of the stream is in some way computed from the input.<br>Encryption and compression streams. |
| Duplex | A duplex stream is one that you can both read from and write to.<br>Network socket. You can write data to the network socket as well as read data from it. |

# Express

## Fast, unopinionated, minimalist web framework

WEB APPLICATION FRAMEWORK FOR NODE APPS.

EXPRESS CAN ACCEPT MIDDLEWARE USING THE ‘USE’ FUNCTION WHICH CAN BE REGISTERED WITH HTTP.CREATESERVER.

EXPRESS REQUEST / RESPONSE OBJECTS ARE DERIVED FROM STANDARD NODEJS HTTP REQUEST / RESPONSE

REQUEST OBJECT CAN HANDLE URL : ROUTE PARAMETER & QUERYSTRINGS

EXPRESS ROUTER IS USED TO MOUNT MIDDLEWARES AND ACCESS ALL REST APIS

## Data Persistence

**Why NoSQL ?**

– Scalability

– Ease of Development

**NoSQL servers can be placed into four broad categories:**

Document databases (e.g. MongoDB)

Key-value databases (e.g. Redis)

Column-family databases (e.g. Cassandra)

Graph databases (e.g. Neo4J)

## MongoDB

- A MongoDB deployment consists of multiple databases.
- Each database can contain multiple collections.
- A collection is simply a name that you give to a collection of documents.
  - Each collection can contain multiple documents.
  - A document is effectively a JSON document

## Mongoose

- Mongoose provides a straight-forward, schema-based solution to model your application data.
- It includes built-in type casting, validation, query building, business logic hooks and more, out of the box.

```
> npm install mongoose
```

# Task – REST API : Todo App

**Todo App should provide following API**

| METHOD | API | DESCRIPTION |
|---|---|---|
| GET | /todos | Get all todo items |
| POST | /todos | Create todo item |
| GET | /todos/{id} | Get single todo item |
| PATCH | /todos/{id} | Update single todo item |
| DELETE | /todos/{id} | Delete single to item |

# View Engines

**Jade**

Uses whitespace and indentation as a part of the syntax.

**Handlebar**

Developers can’t write a lot of JavaScript logic inside the templates.

**EJS**

Follows JavaScript-ish syntax. Embed JavaScript code in template. Commonly used.

**Vash**

Vash is a template view engine that uses Razor Syntax. Look familiar to people who have experience in ASP.Net MVC.

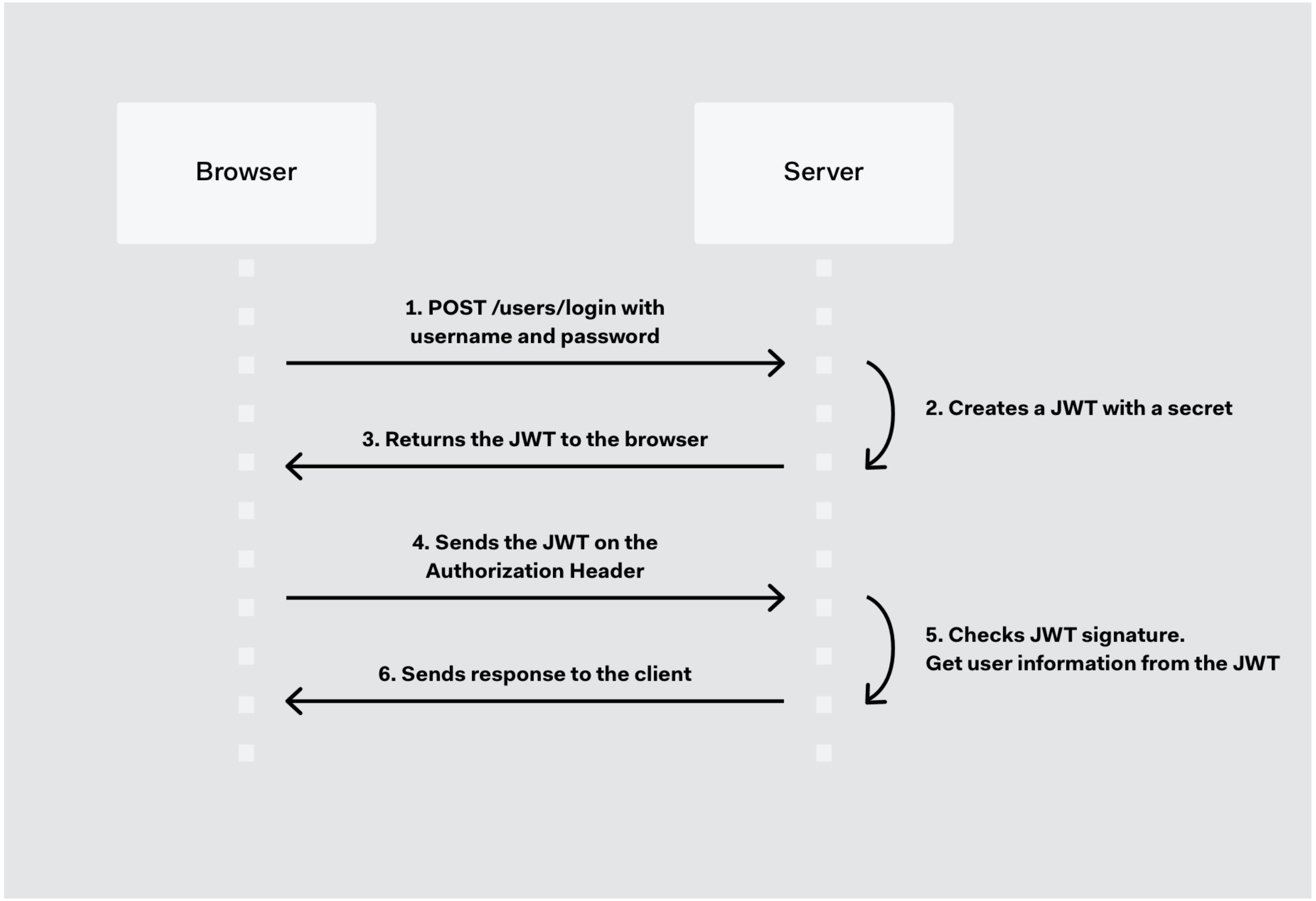
## NodeJS : Securities

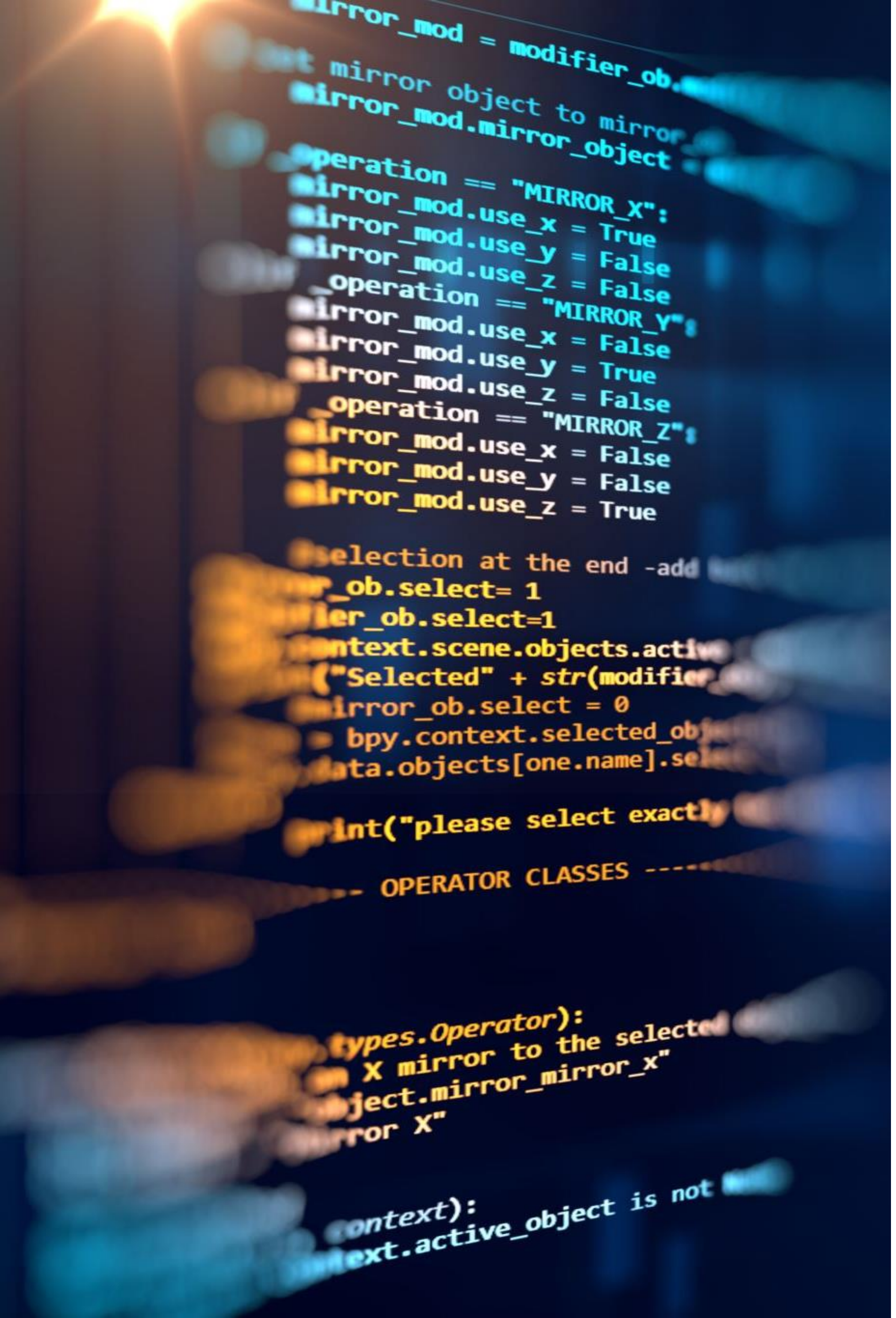
A JSON Web Token (JWT), defines an explicit, compact, and selfcontaining secured protocol for transmitting restricted information.

The JWT Claims Set represents a compact URL-safe JSON object, that is base64url encoded and digitally signed and/or encrypted.

The JSON object consists of zero or more name/ value pairs, where the names are strings and the values are arbitrary JSON values.

## JWT : Understanding


Browser
Server
1. POST /users/login with username and password
2. Creates a JWT with a secret
3. Returns the JWT to the browser
4. Sends the JWT on the Authorization Header
5. Checks JWT signature. Get user information from the JWT
6. Sends response to the client



## Testing

- Mocha
  - Feature-rich JavaScript test framework
  - Simple Asynchronous Testing API
- Chai
  - Chai is a BDD / TDD assertion library for node
  - Can be paired with any Javascript testing framework

## Microservices

Microservices are an architectural approach based on building an application as a collection of small services.

Microservices-based architecture simply breaks down the functionality into a small suite of services integrating the application via APIs.

Allows us to build, operate and manage services independently.

Microservices have their own load balancer and execution environment to execute their functionalities.

Microservices offer language and platform freedom.

# Microservices

Each service is isolated

Boundaries with API

Developing single service is faster

Testing single service is easy

Maintaining single service is easy

More failure resilient

## References

### Books

NodeJS by Basarat Ali Syed

Node.JS Web Development by David Herron

### Web

https://nodejs.org

http://npmjs.com

https://mongoosejs.com/docs/api.html

http://expressjs.com/