## what is node js?

* built over javascript
* uses v8 engine which was develop for google (writtem in CPP)
* JS runs in browsers
* NodeJS is javascript runtime (allow you to run JS on server) which means we can run node js without browser
* NodeJS uses V8 engine to convert JS code to machine code

## First Code

* check node version node -v
* to enter nodeJS interactive mode called REPL write node in cmd
* execute file using nodeJs like node HelloWorld.js

## NodeJS role in Web Development

* Run Server (create Server and listen to incoming request)
* Business logic (Handle request, validate input, connect to DB)
* responses (return responses (rendered HTML, JSON etc))

## JS Summary

* weakly typed language (no type assignment, data types can be switched dynamically)
* OO language
* versatile language (run in browser as well on local and server, can perform broad variety of task)

## const vs let vs var

* var is global scoped (if variable define outside function can be access anywhere in program)
* all three are function scope
* const and let are blocked scope (It can’t be accessible outside the particular block).
* hoisting (a variable can be declared after it has been used) only var

x = 5;

var x;

* redeclaration only in var
* reassignment in var and let

## objects

* object allow you to group key pair value

const person = {

name: 'Hassan' // name is the key hassan is the value where this is the property of object

age: 29

// three ways to declare fuction in object where this refer to object

greet1() {console.log('hi' + this.name)} //output h1 hassan

greet2: () => {console.log('hi' + this.name)} // Arrow functions do not bind their own "this" output will be error

greet3: function(){console.log('hi' + this.name)} //output h1 hassan

}

## primitive vs reference type

* arrays and objects are considered reference types, while primitive types include numbers, strings, booleans, null, and undefined
* Primitive types are immutable, meaning their values cannot be changed. When you assign a new value to a variable holding a primitive type,
* it creates a new copy of that value.
* Reference types are mutable, meaning their values can be changed. When you assign a reference type to a variable, the variable holds a reference (memory address) to the actual object or array. Modifying the object or array through one variable affects all references to it.

## spread operator vs rest operator

const a = 1;

const arr = [2,3,4,5]

const b = [...arr]; // spread operator extracts the collected elements to single element

const c = [a, ...arr]; // rest operator is known for destruction of element then it collects the leftover element to make it array or obj

## Destructuring

const person = {

name: 'Hassan'

age: 29

greet1() {console.log('hi' + this.name)}

}

//destructuring in function

const printName = ({name}) => console.log(name);

printName(person);

// object destructuring

const {name, age} = person;

// array destructuring

const hobbies = ['sports','cooking'];

const [hobby1, hobby2] = hobbies; // in array des depends on index

console.log(hobby1, hobby2) // output sports, cooking

## Async code

const fetchData = () => {

const promise = new Promise((resolve, reject) => {

setTimeout(() => {

resolve('Done!');

}, 1500);

});

return promise;

};

setTimeout(() => {

console.log('Timer is done!');

fetchData()

.then(text => {

console.log(text);

return fetchData();

})

.then(text2 => {

console.log(text2);

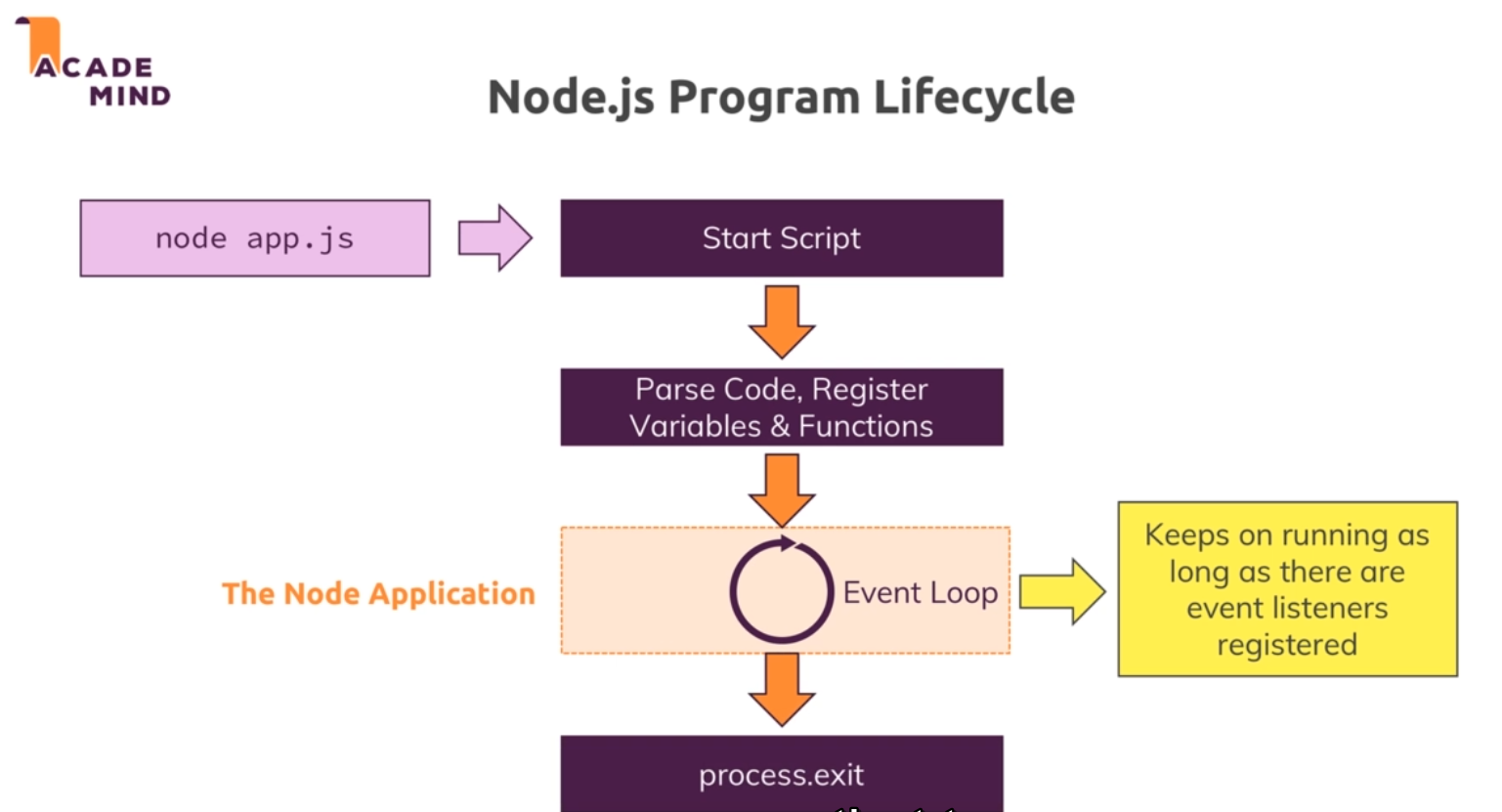
});

}, 2000);

console.log('Hello!');

console.log('Hi!');

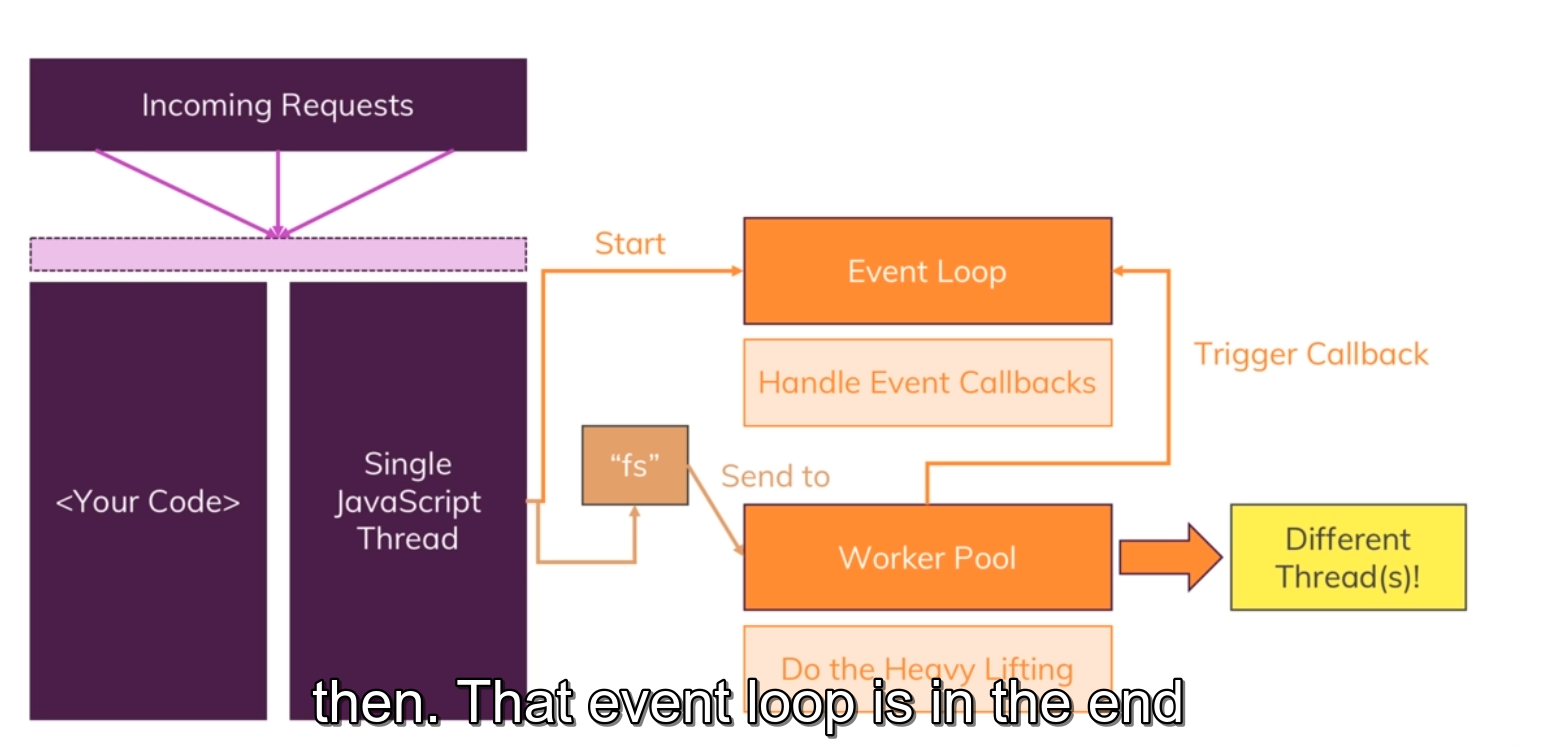
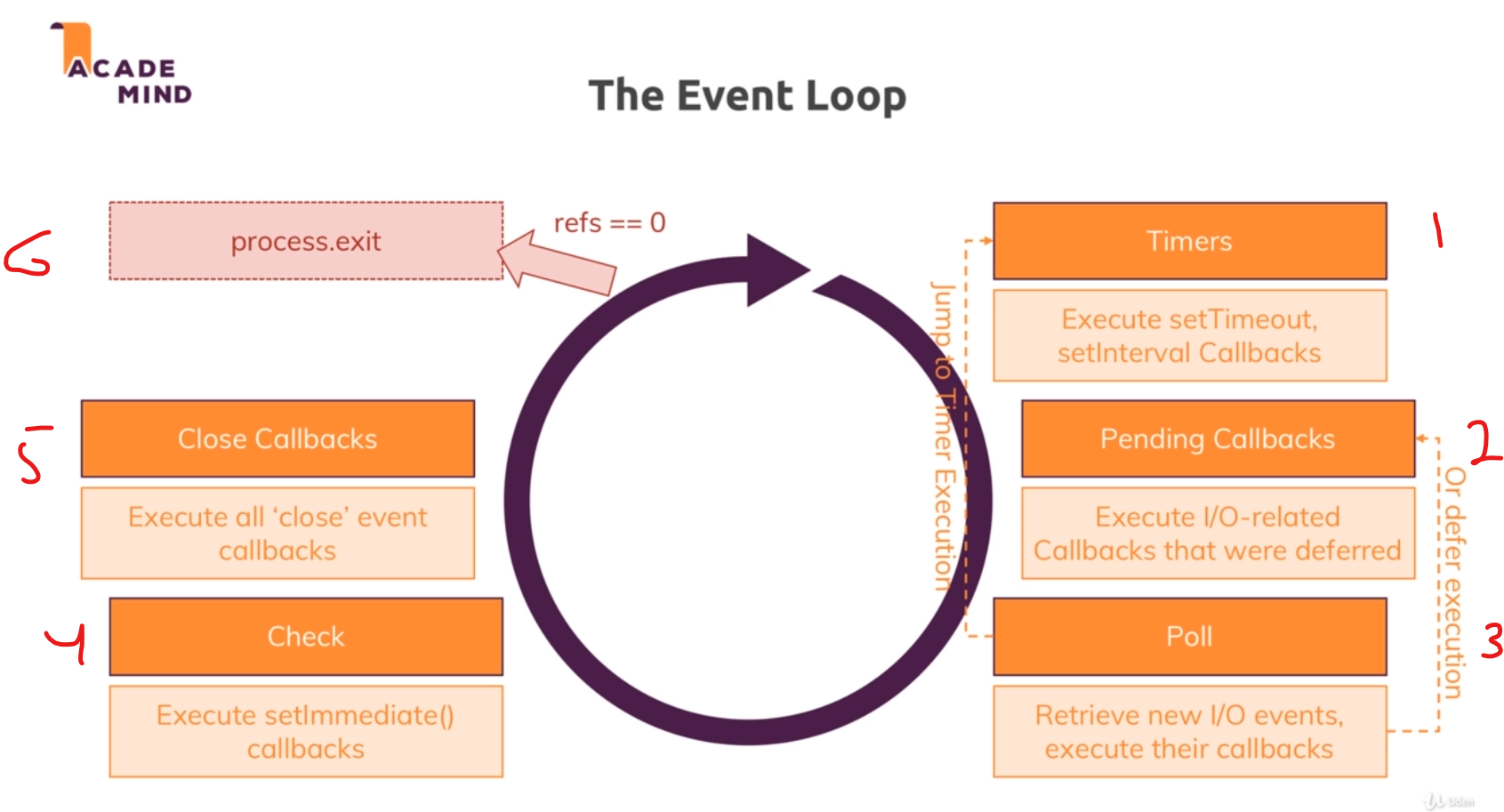
## Node program lifecycle

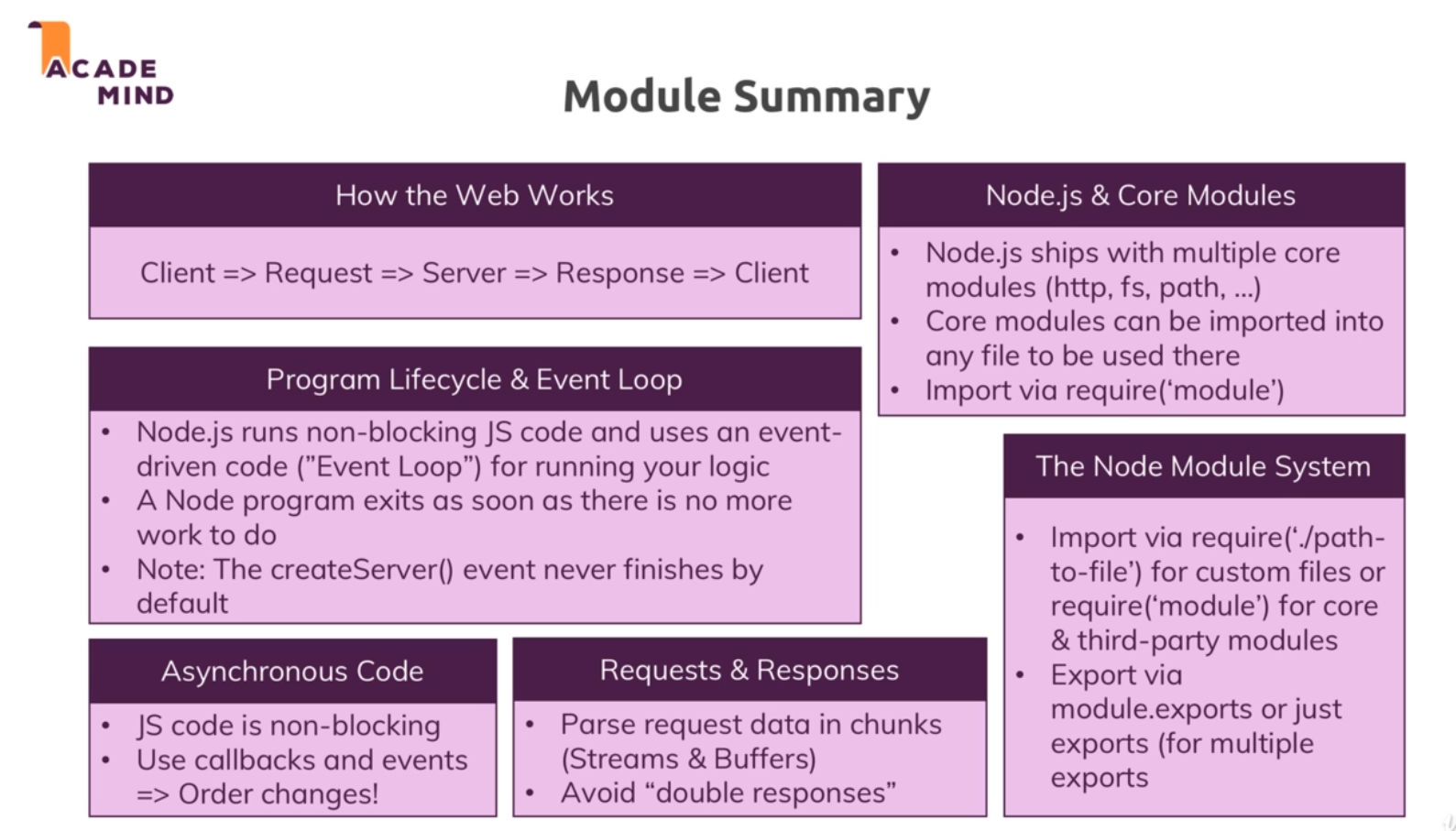


## Different Header

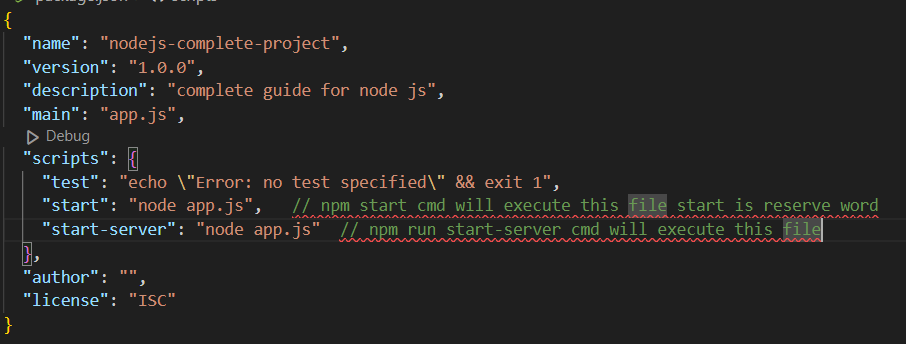
<https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers>

## NodeJS (single thread, event loop and code blocking)

* NodeJS use single JS thread (what happens if there are multiple requests?) it not entertain that request in this case event loop and worker plays an important role
* Event loop automatically starts as the program execution starts it is responsible for handling event callbacks
* Event loop will only execute those callbacks which can be executed immediately
* Heavy operation are send to worker pool which as also start automatically which do the heavy lifting and can execute different threads and detached from our code once done it will trigger the call back
* 
* 

NodeJS Basic module Summary  


Installing 3rd party packages

* Write npm init cmd in terminal
* 

## Local vs Global packages

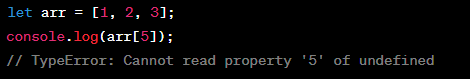
**If the package is installed globally, then that package can't have different versions for different applications**. By installing the package locally, we can ensure that each program can have its local package of the desired version.

## Npm install dev and prod dependencies

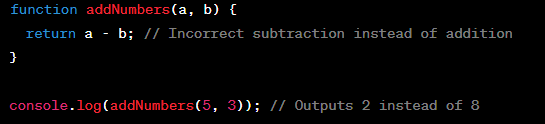
* For dev npm install nodemon –save-dev
* For prod npm install nodemon –save
* For globally npm install nodemon –g

## Error types

* Syntax error (typo error)
* Runtime error (try to execute code which will break)



* Logical error



## **What Is Express JS?**

Express is a node js web application framework that provides broad features for building web and mobile applications. It is used to build a single page, multipage, and hybrid web application.

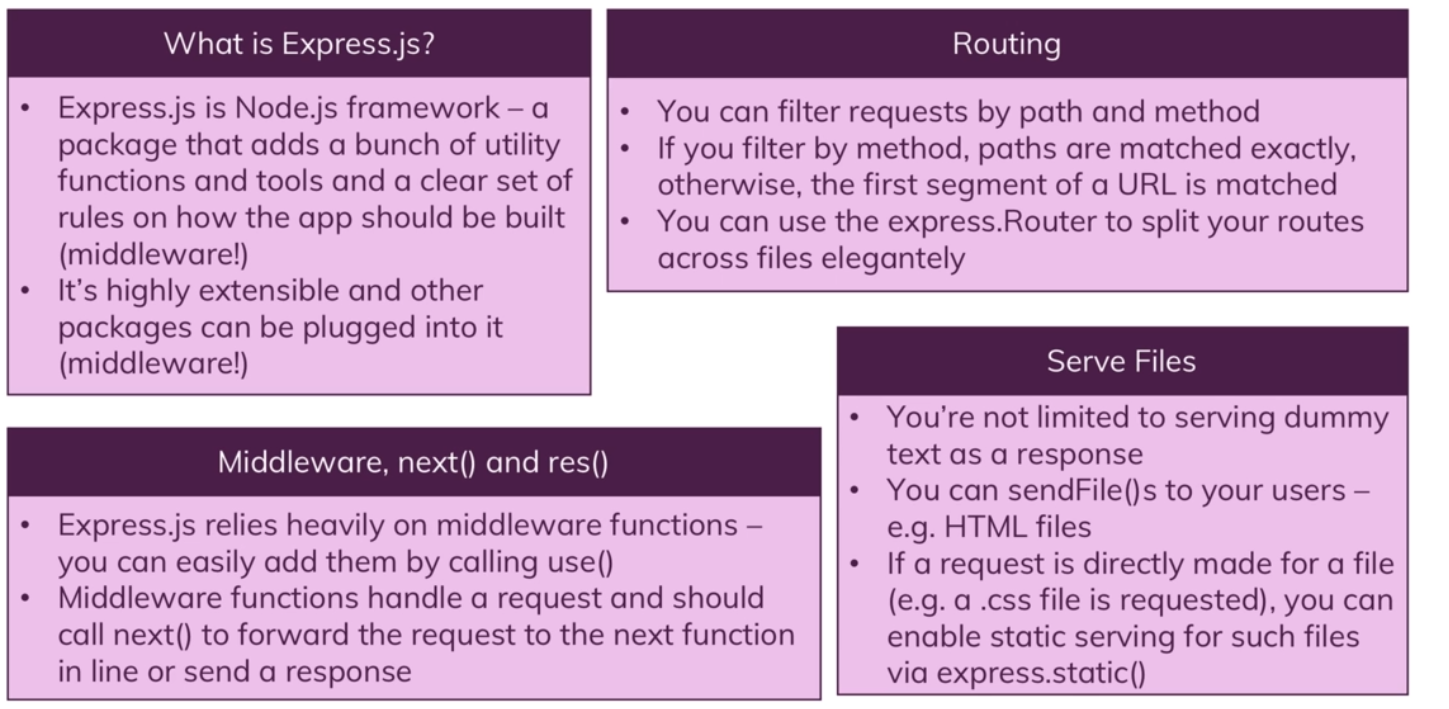
It's a layer built on the top of the Node js that helps manage servers and routes.

## **Why Express JS?**

* Express was created to make APIs and web applications with ease,
* It saves a lot of coding time almost by half and still makes web and
* mobile applications are efficient.
* Another reason for using express is that it is written in javascript as javascript is an easy language even if you don't have a previous
* knowledge of any language. Express lets so many new developers enter the field of web development.

The reason behind creating an express framework for node js is:

* Time-efficient
* Fast
* Economical
* Easy to learn
* Asynchronous



## What is MVC

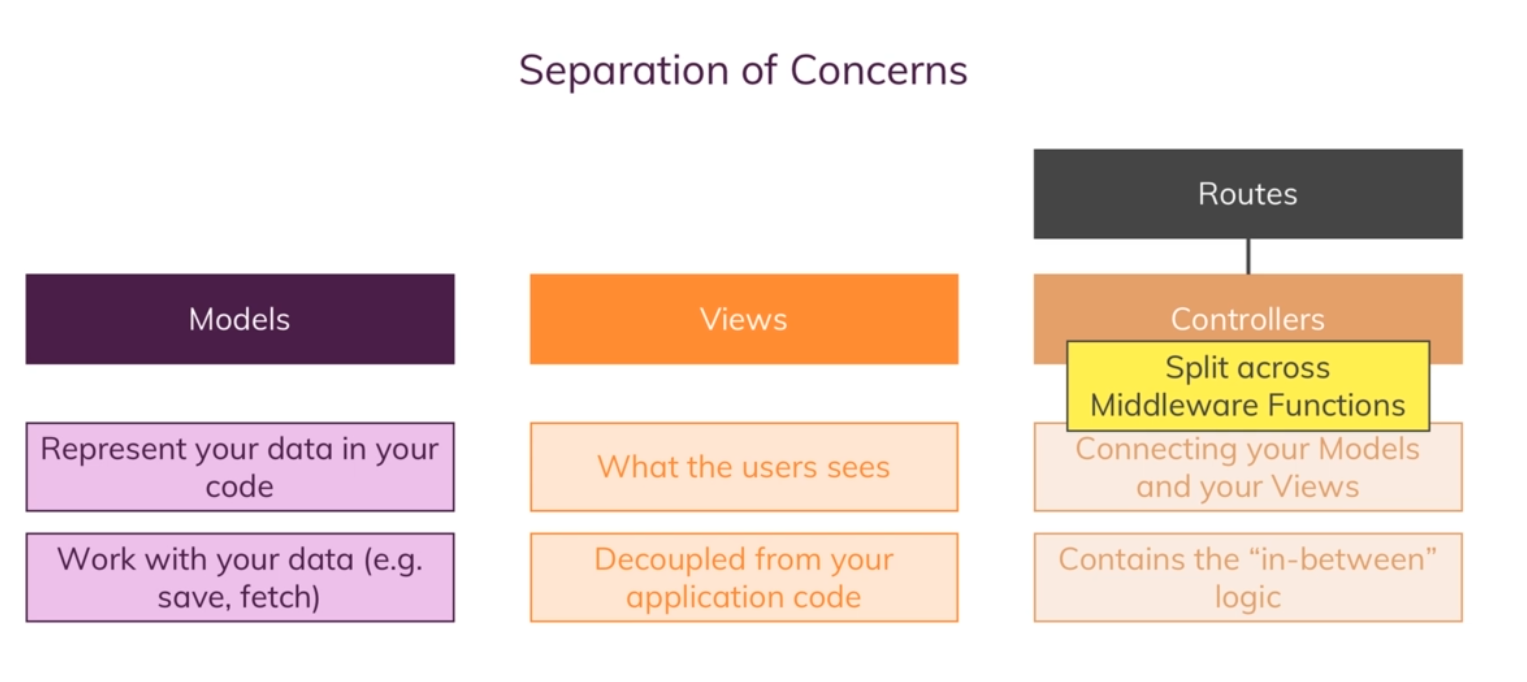
Separation of concerns means one part do one thing stand for model view controller

Models are basically objects which is responsible for representing data, working with data (eg save, fetch)

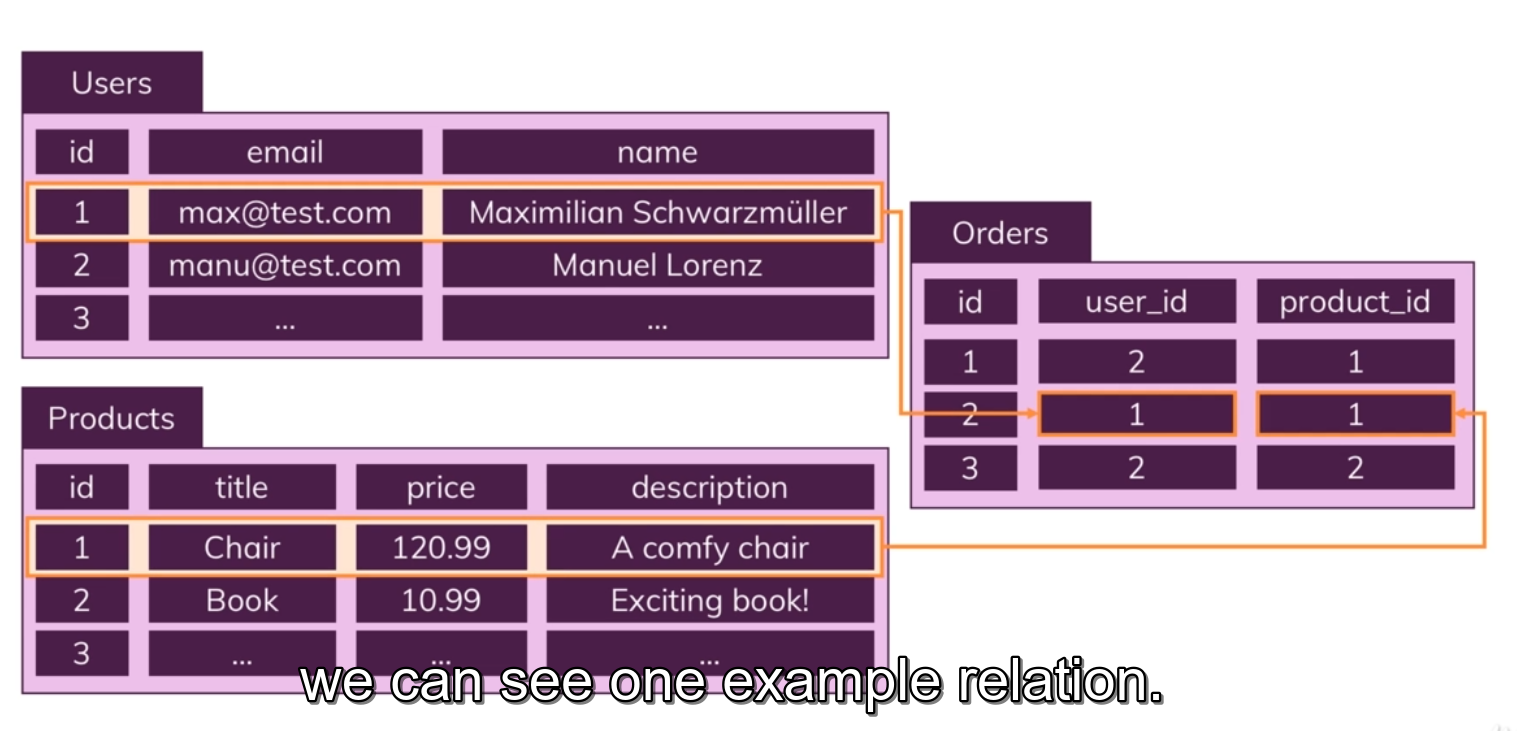
Views are to show that data to user and decoupled from your application code

Controller are connection btw models and views and contain in btw logic

Controller are also split across middleware functions



## SQL

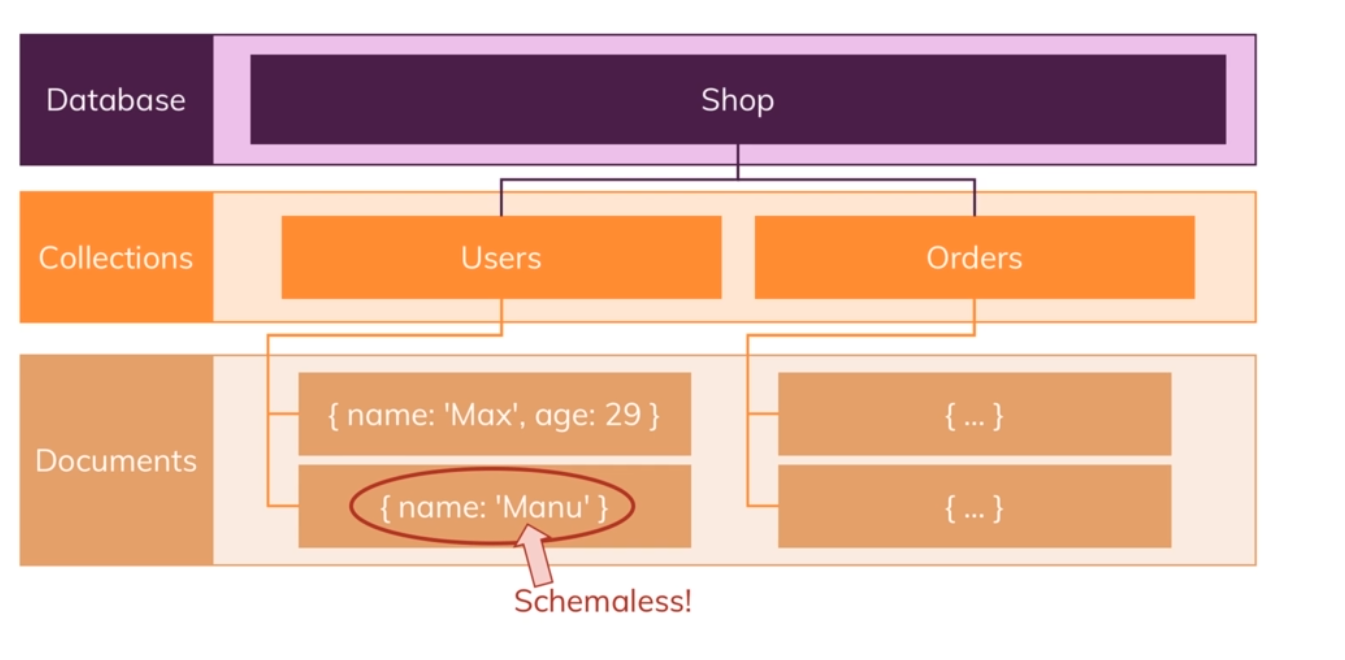
In SQL DB data is stored in tables each tables have columns or fields like id, name and email etc. data is stored in these fields are called records or rows. SQL DB allow you to relate tables 

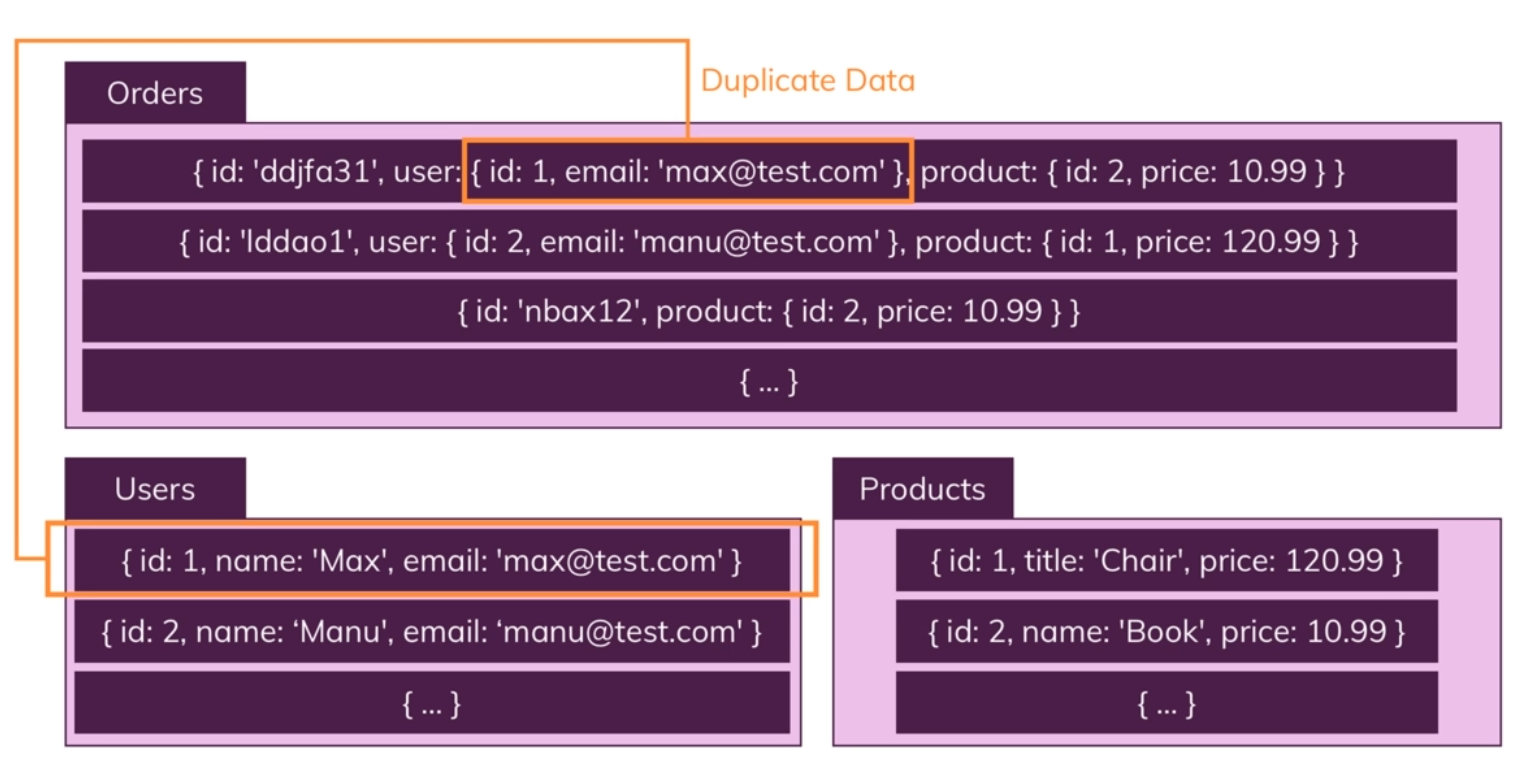
### Core Characteristics of SQL DB

* Clearly defined data schema. (All the data has to fit the schema in DB)
* Data Relations (tables are connected via one to one, one to many, many to many relationship)

## NoSQL

* In NOSQL we have DB example shop
* instead of tables we have collections like users and orders
* each collections have records called documents
* doesn’t have a strict schema
* relationship doesn’t exist





### Core Characteristics of NoSQL DB

* no data schema (no structure is required).
* No data relations (we can relate documents but we don’t have to and shouldn’t relate to much otherwise queries will become slow)

## Horizontal scaling

We add more servers and merge data into one DB

## Vertical scaling

Improve server capacity/ hardware

## SQL vs NoSQL

In SQL horizontal scaling is difficult vertical scaling in possible this causes limitation for lots of (thousands) read and write queries per seconds.

In NoSQL horizontal scaling and vertical Scaling is both possible great performance for both read and write operations

