**NODE JS**

https://food-order-react-app-5d529-default-rtdb.firebaseio.com/meals.json

**Introduction**

1. Node JS is a Javascript Language but also support more feature than Javascript Itself. Features like Accessing File System, running on any machine***. (It uses Non-Blocking I/O and Asynchronous process and a Single Thread Application)***

***Node js is not preferred for CPU Intensive Work because its single thread application( for e.g complex problems like calculating factorial etc.)its used for I/o intensive work.***

***So Nodejs handles multiple requests using libuv library which built using kernel and kernel has multiple threads so behind the scenes it gets handled.in short node single thread does not wait for response it directly transfer request to this kernel.***

***OR***

***The event loop is the secret by which JavaScript gives us an illusion of being multithreaded even though it is single-threaded.***

2. Node Js is a runtime Environment (Interpreter ) which convert Source code to machine code/Binary code, Therefore its run on any Machine/Server rather then only in browsers and Node Js Runs on any Machine.

3.V8 is a Javascript Engine(Built by Google) that run JS in Browser.

4. Similarly Node also uses v8 engine.

***Synchronous:*** One at a Time (blocking code)

***Asynchronous***: Multiple things happen at the same time (parallel)

Three Methods to write Async code :

**1. callbacks**

**2. Promises**

**3. Async/Await**

Callback means : when you pass function as parameter inside another function and higher order function means those function itself where function as parameter is passed

Event Loop:

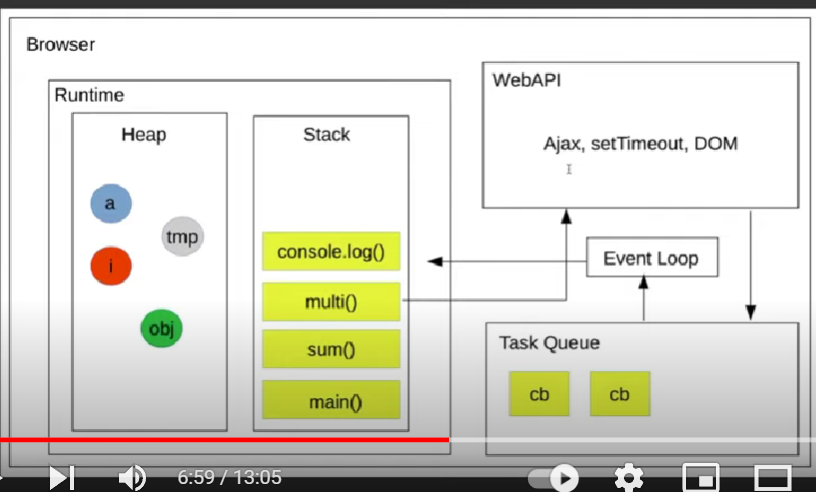
Every function on Javascript page on runtime goes intside stacks i.e (global execution context) and if they have inner functions they will get upon each other inside stack, when respective function execution overs those functions start getting popped out from stack.

Suppose if there is some Asynchronous functions are present then they will treated as “callbacks” and get stored in WebApi container. Suppose those callback function execution completes, then they will shift to task/Event Queue.

Here Event loop comes in picture, event loop first check and wait till all function in stack gets executed , after stacks get empty then only callback value returns from the queue.

It is called event loop because it continuesly( in loop ) checking the callStack and callback Queue.

Note: for every callback who give promise e.g fetch will store result in MicroQueue instead of callback Queue, and MicroQueue priority is higher than callback Queue.



In above picture as there is taskQueue similary there is different different task/event Queue like check, Timer etc.

**Nodemon**

Nodemon use to automatically reload the server if there is some changes in .js and .json file .

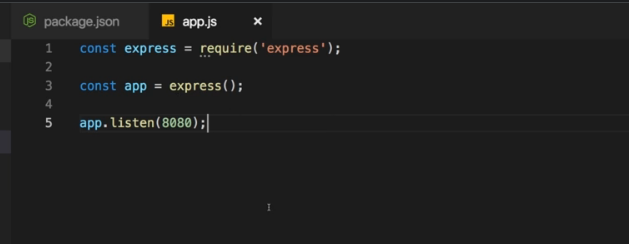
**Express JS**

Its is a JavaScript Framework use to easily handle different routes Requests.

There is a Middleware (line of Different Functions) where request get passed before sending Response.

In order to parse request (e.g req.body )in express we need to use different Body parser.

Simple express Server:



const bodyParser = require('body-parser')

const app = express();

app.use(bodyParser.urlencoded({extended:false}))

app.use('/add-products',(req, res, next) => {      //here app.use help to use middlewares

    res.send('<form action="/products" method="POST"><input type="text" name="title"><button type="submit">Add Product</button></form>')

})

app.post('/products',(req,res,next)=>{  //app.get and app.post use for filtering get and post requests however app.use works for every Incoming Requests And app.get and app.post trigger Only on Exact path match, however app.use trigger on every request when start of path match i.e(‘/’) or (‘/…’)

    console.log(req.body);  // Express automatically adds body in requests but we need if we Don’t use body parser req.body = undefined

    res.redirect('/')       //body parser to parse the requests

})

Templating Engine:

There are, Three Types of Tempalting Engine To Render Dynamic Content while using MVC Model.

1. EJS (<p> <%=name %> </p>) (Recommended)

2. Pug(Jade) ( p{#name} )

3.Handlebars ( <p>{{ name }}</p>)

MVC Pattern:

Model : Represent your any Model in code(e.g product, User),and work with your data (e.g Fetch, save)

View : What the users Sees.

Controller : contains the main logic, control the model and view to decide what to render or not.

Connecting to databases

// connecting to mysql

const mysql = require('mysql2');

//pool avoids us to everytime making and connection and closing a connection

const pool = mysql.createPool({ //adding sql server credentials to which we have

    host:'localhost', to connect

    user:'root',

    database:'nodeproject',

    password:'nodeproject',

})

module.exports = pool.promise();

const db = require('../util/database');

// executing  simple query inside function using db.excute that will return promise as we export pool.promise above!!

save() {

    return db.execute('INSERT INTO products (title,price,ImageUrl,description) VALUES (?,?,?,?)',

    [this.title,this.price,this.imageUrl,this.description])

  }

// calling function somewhere it requires

save().then((data return from database but for save no data because we done insert operation)=>{

// any logic

    res.redirect('/');

  }).catch( err=> console.log( err));

We will used sequelize a 3rd Part package that help to write sql query in terms of object

const Sequelize = require('sequelize')

const sequelize = new Sequelize('nodeproject','root','nodeproject',

{dialect:'sql2',host:'localhost'

})

module.exports = sequelize;

this setup behind the scene make the above connection

//Defining Model for Database e.g Product

const Sequelize = require('sequelize')

const sequelize = require('../util/database')

const Product = sequelize.define('product',{

  id:{

    type: Sequelize.INTEGER,

    primaryKey:true,

    allowNull : false,

    autoIncrement : true,

  },

  title:Sequelize.STRING, // his will directl specify Type

  imageUrl:{

    type:Sequelize.STRING,

    allowNull:false

  },

  description:{

    type:Sequelize.STRING,

    allowNull:false

  }

})

module.exports = Product;

const sequelize = require('./util/database');

// this will convert model to table by automatially create sequel query behind the scene

sequelize.sync()

.then((result)=>{

    console.log(result);

    app.listen(3000); // start serve if there is successful connection to Database

})

.catch( err =>  console.log(err))

Vertical Scalablity: Increasing ram and Power in single Server.

Horizontal Scalability : Increasing number of Servers

Using Databases:

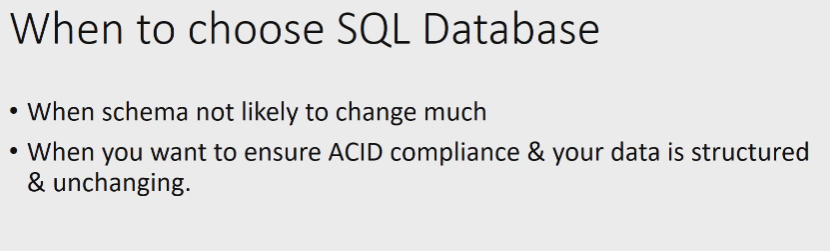
Fetching data from file takes more time because we have to read entire file and it takes time so we have to use database.

Two types of Databases:

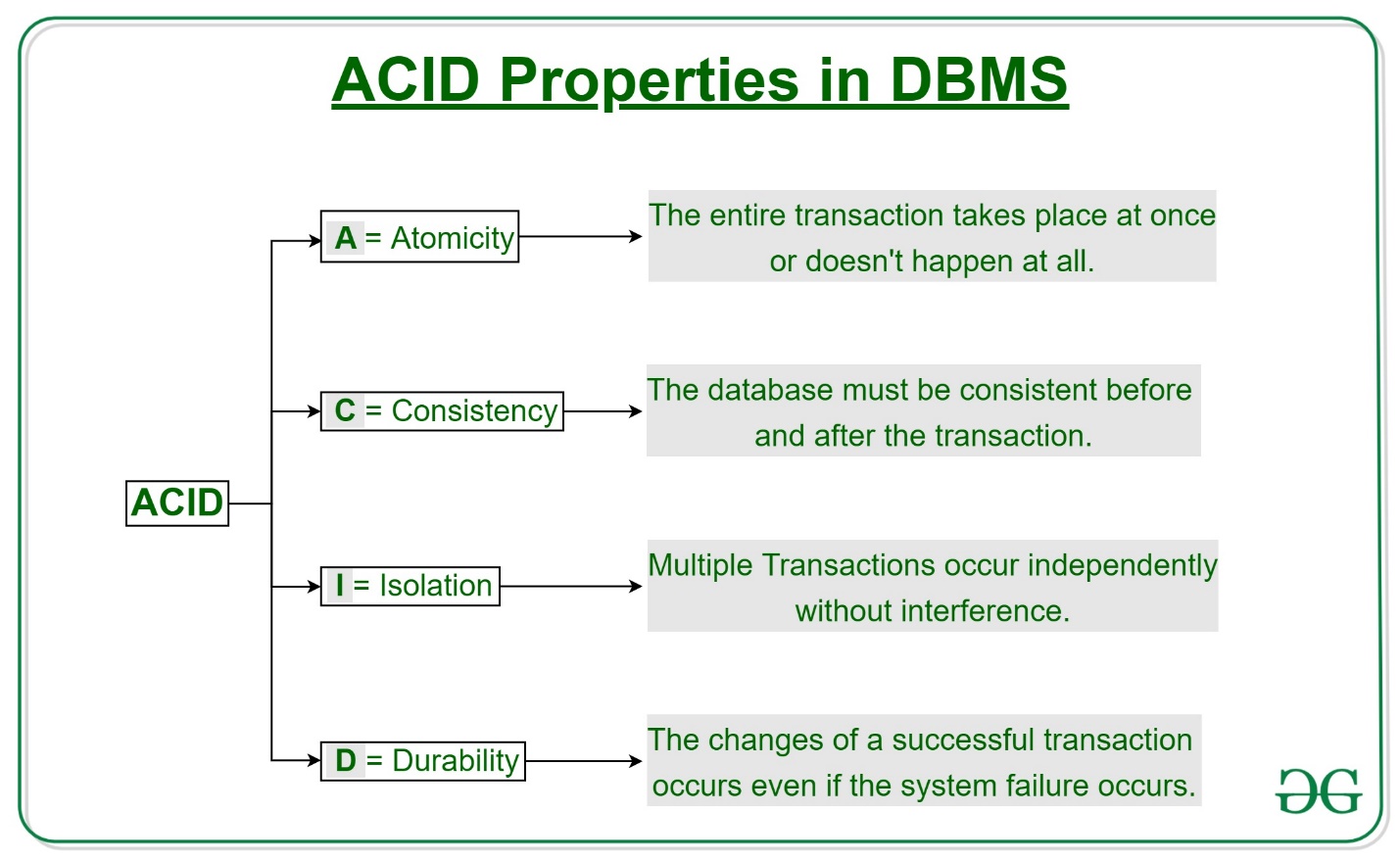
1.SQL(structured query language) -> Tables, relations between tables, required fixed scehma Like it must have 3 columns.

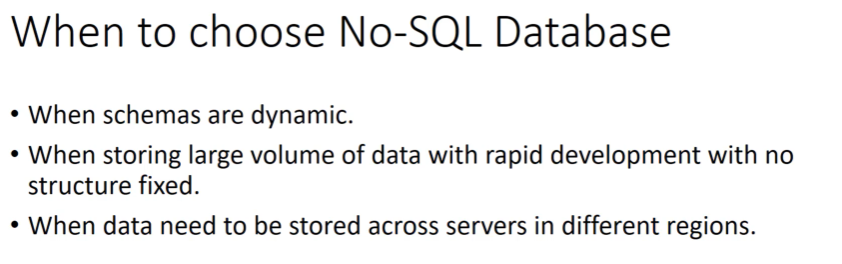
Schema means Student(studentId, studentName, ProjectId)

2.NOSQL ->unstructured data, not required fixed schema.



If there is transaction u must use sql databases because it also have ACID properties.





Cookies:

A cookie is a place where we can store Information like Is the user logged in or not, so that every time on reload user still stays logged in after authentication.

Cookies are set by the server in the client. But users can also manipulate Cookie in client, so we use sessionID to store this confidential information instead of direct Login Credentials.

Like on the Website there is a checkbox Remembered Me, if check/click on that then it will make a cookie in our browser.

Cookies are also sent to the server with requests

Size: 4kb

Cache: The main difference between Cache and Cookie is that Cache is used to store online page resources during a browser for the long run purpose or to decrease the loading time. On the other hand, cookies are used to store user choices/data such as browsing sessions to trace the user preferences.

A cache stores the contents like HTML pages, images, Javascript, CSS, etc.

While cookies store the contents like browsing sessions and temporary tracking data.ie s like userLoggedin or not



Local Storage:

If you’re building a static site (like a single page app, for instance), using something like local storage means your web pages can run independently of any web server/Backend. They don’t need any backend language or logic to store data in the browser: they can just do it as they please.

This is a pretty powerful concept and one of the main reasons that local storage is such a hit with developers. This is majorly used for testing purposes.

Local storage wasn’t designed to be used as a secure storage mechanism in a browser. It was designed to be a simple string-only key/value store that developers could use to build slightly more complex single-page apps. That’s it.

It does not even expire after the browser is closed

Session Storage:

Session storage is same as Local Storage however local storage never expires, we need to clear their data manually, and Session Storage Expires on Tab Close.

Cookie Storage:

Here using document.cookie data is string type data is stored, it also support expiration time.

There are two Types of Authentication:

1. Token based Authentication (JWT)Json Web Token

Token = Header+payload+Signature

1. Session-Basedsed Authentication

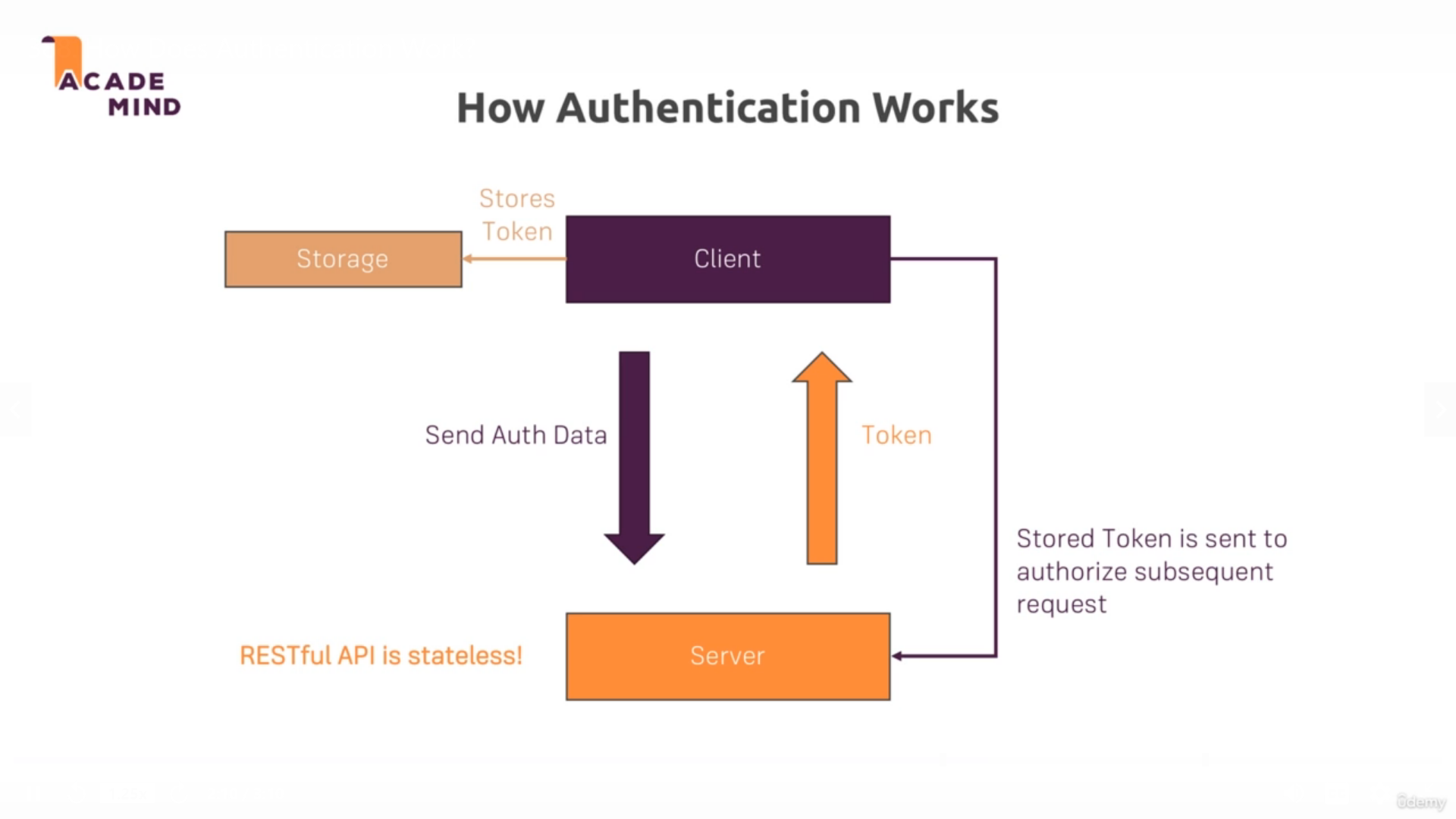
For Session-Based Authentication

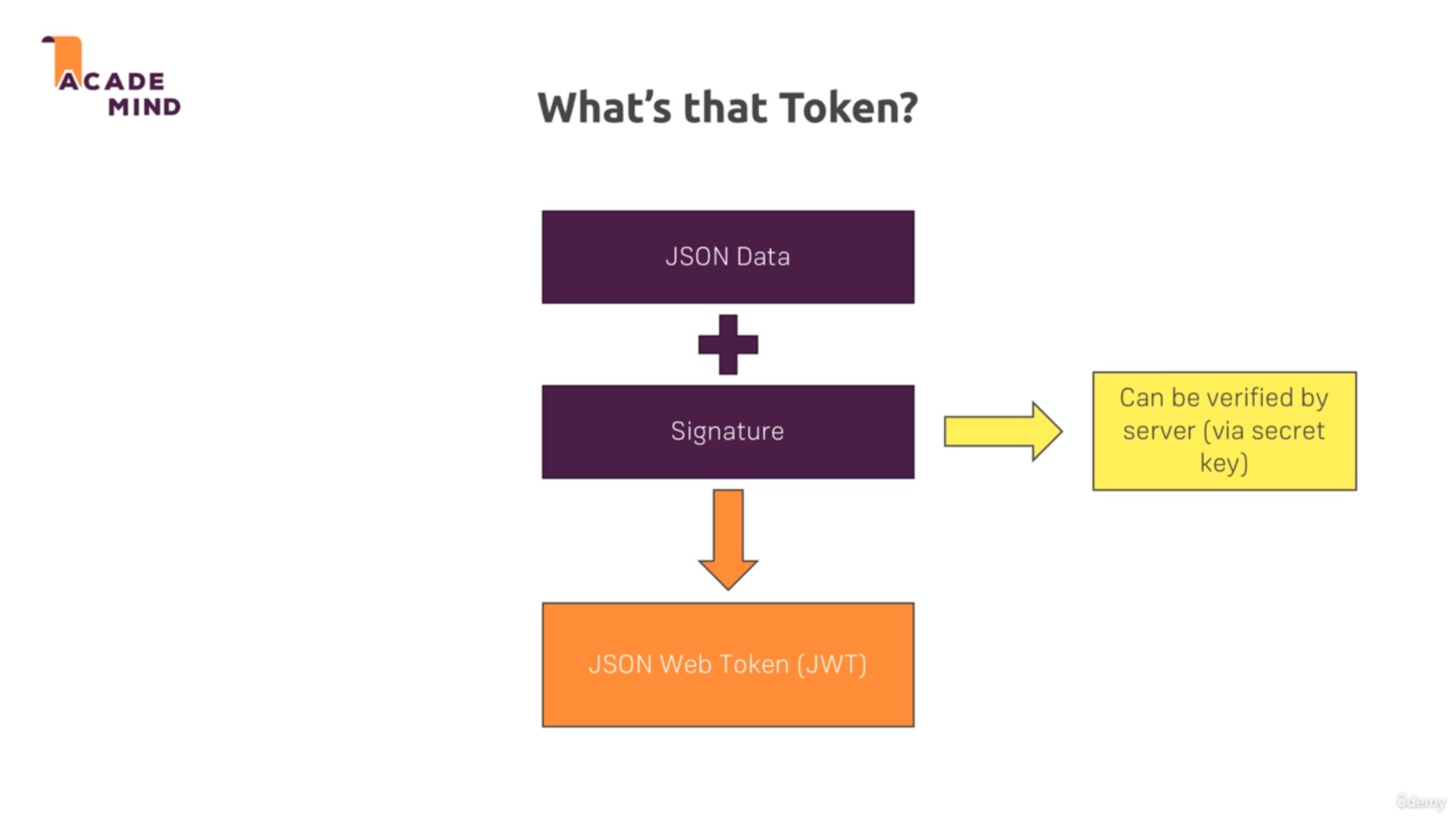
Session Storage:

When u enter Username and Password, it will create session, then this session will be sent to the server [Session are stored in Server], the server will create sessionID. Then send that sessionID back to client .However In cookie we store the session Id(HASHCODE), it is a hashcode so user cannot manipulate it. If user manipulates then server rejects it.

Session cookie expires when the browser closed

For JWT-Based Authenticati





Advantages of JWT is User Authentication Information is stored in Client only, so If there are multiple servers we just need to store secret key on that servers, meanwhile in previous session based authentication we need to create session on every server, because if user moved to 2nd server (with help of load balancer) because of traffic in already authenticated server, he need to login again.

In REST API , you don’t need use session and Cookies for Authentication of every request after LoggedIn. Here we use Token which server send to client and client stored and use that token in other requests after LoggedIn.

This is JWT Authentication Method. Json Web Token

Token=base64URlEncode( header) +base64UrlEncode(Payload) +Signature

Header: Contain type of Algorithm e.g(HS256) to hash Token Signature.

Payload: Data

Signature: This will contain the information of user, based on Header, Payload,

And Especially with SecretKey it will generate Signature.

When client sent back Token to the server, the server decode/dehashed the header and payload part of token and again with that help of stored secretKey it will encode it and check whether it matches the last part of token or not.

Server will not Authenticated based on token after token Expiration time.

HTTP STATUS CODE: status code help the browser to know the what happened to sended request

Success 2xx

1. 200 -> OK
2. 201-> Some resources created on server using POST request like creating user, adding some data on server using REST Api.
3. 204-> when you don’t want to return anything, e.g delete request.

Not Modified

1. 304-> Not modified, if certain data not modified on server after your last request then, we send 304 code, to save bandwidth, because the data of last request can be stored in cache.

Client Error

1. 400-> bad request, if the client does not provide sufficient information that needed for that Api, Api can send Bad request that indicated sufficient information is not provided. For e.g Api need username and email but client sends only username then server sends 400
2. 401->Unauthorize, if client Does not send Api key,or client is not Authenticated then server sends 401 code.( don’t know you are)
3. 403->Forbidden, if client sends Api key but client is just as basic User, but he tries to access admin feature then server sends 403 code. (know who you are but you don’t have permission to access this).
4. 404-> Not found, if users accessing API which does not exists then we get 404 error.
5. 422-> Validation failed for form.

Server Error

1. 500-> Internal Server Error, when there is something wrong happens on server( unexpected ) or Database down, then it returns 500 status code

*New Concept*

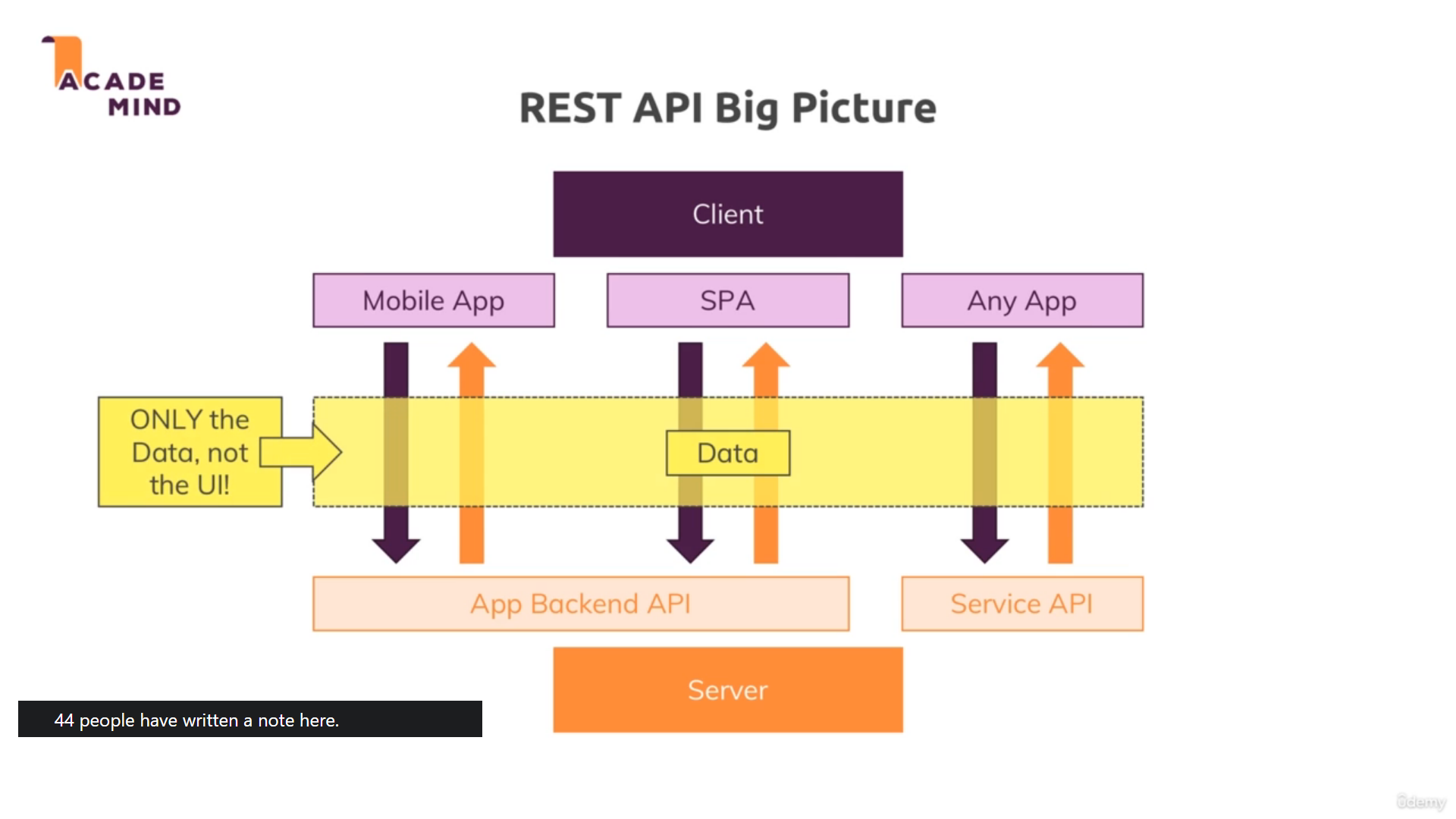
Sendgrid and Nodemailer is used for Emailing in Node 3rd party library.

You can do validation on both client side and Server side, but server Side validation is More Secure, because client side validation can be changed.

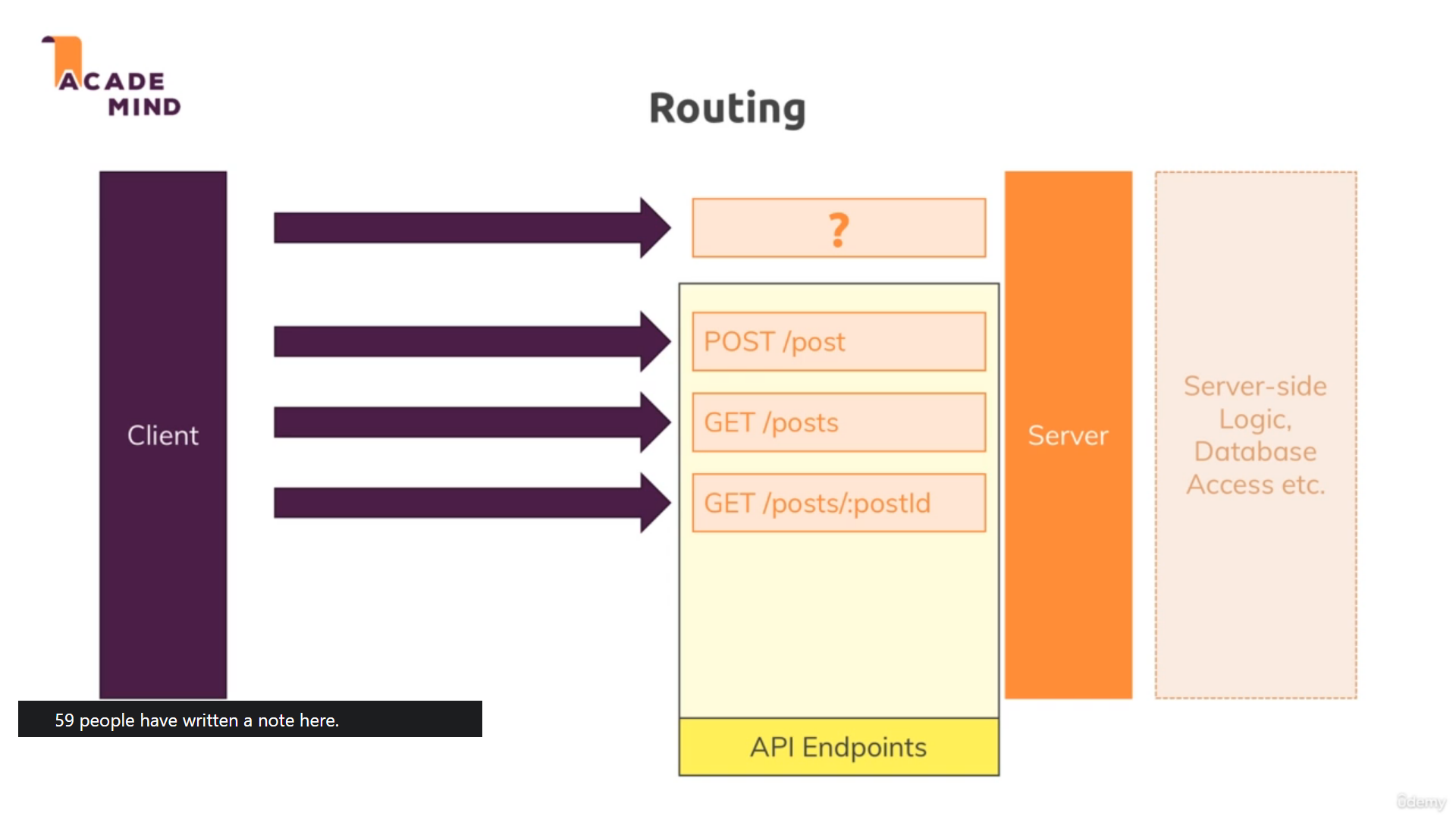
In express this validation can be done by express validator-check package.

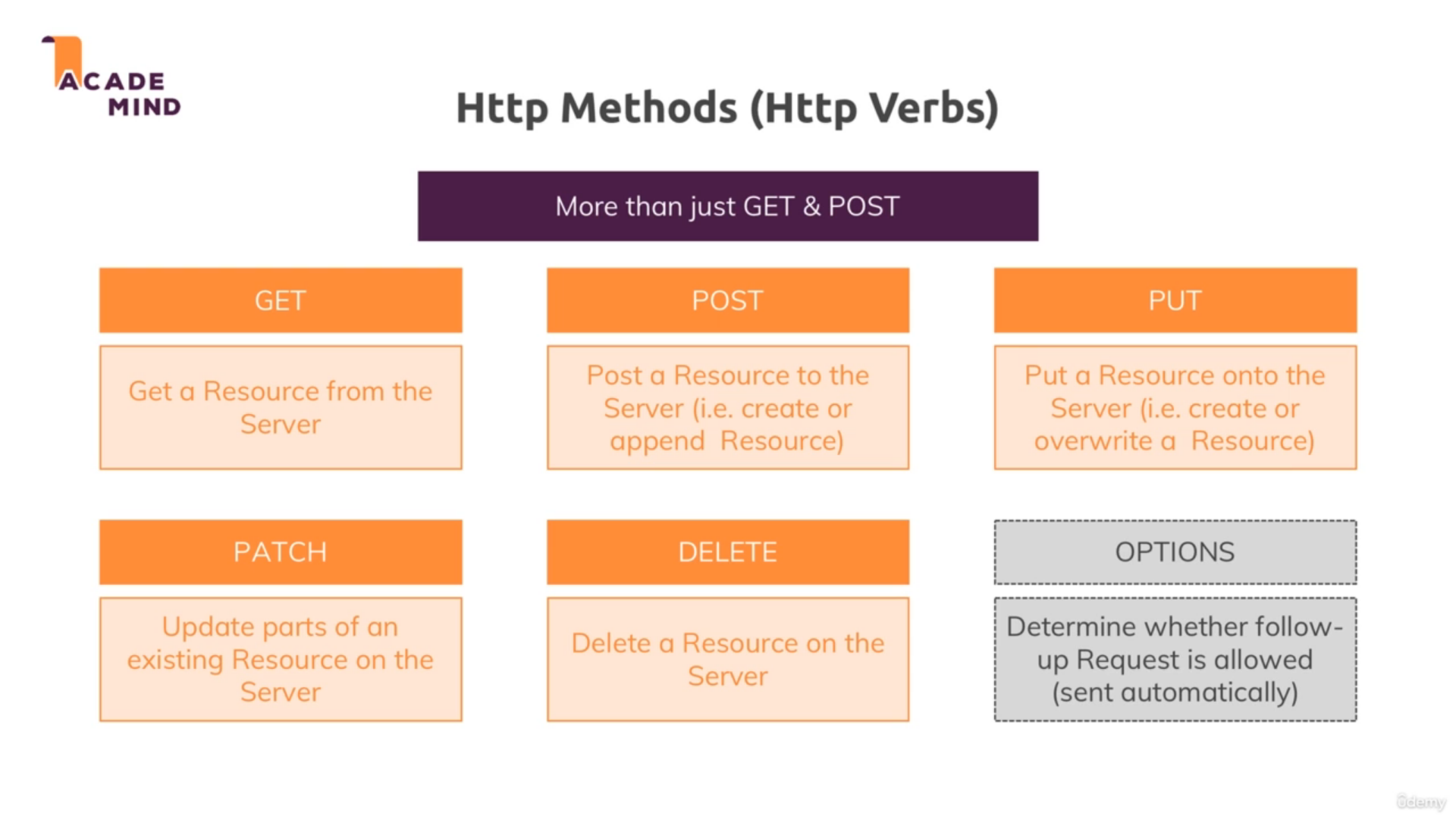
REST API (Representational State Transfer)

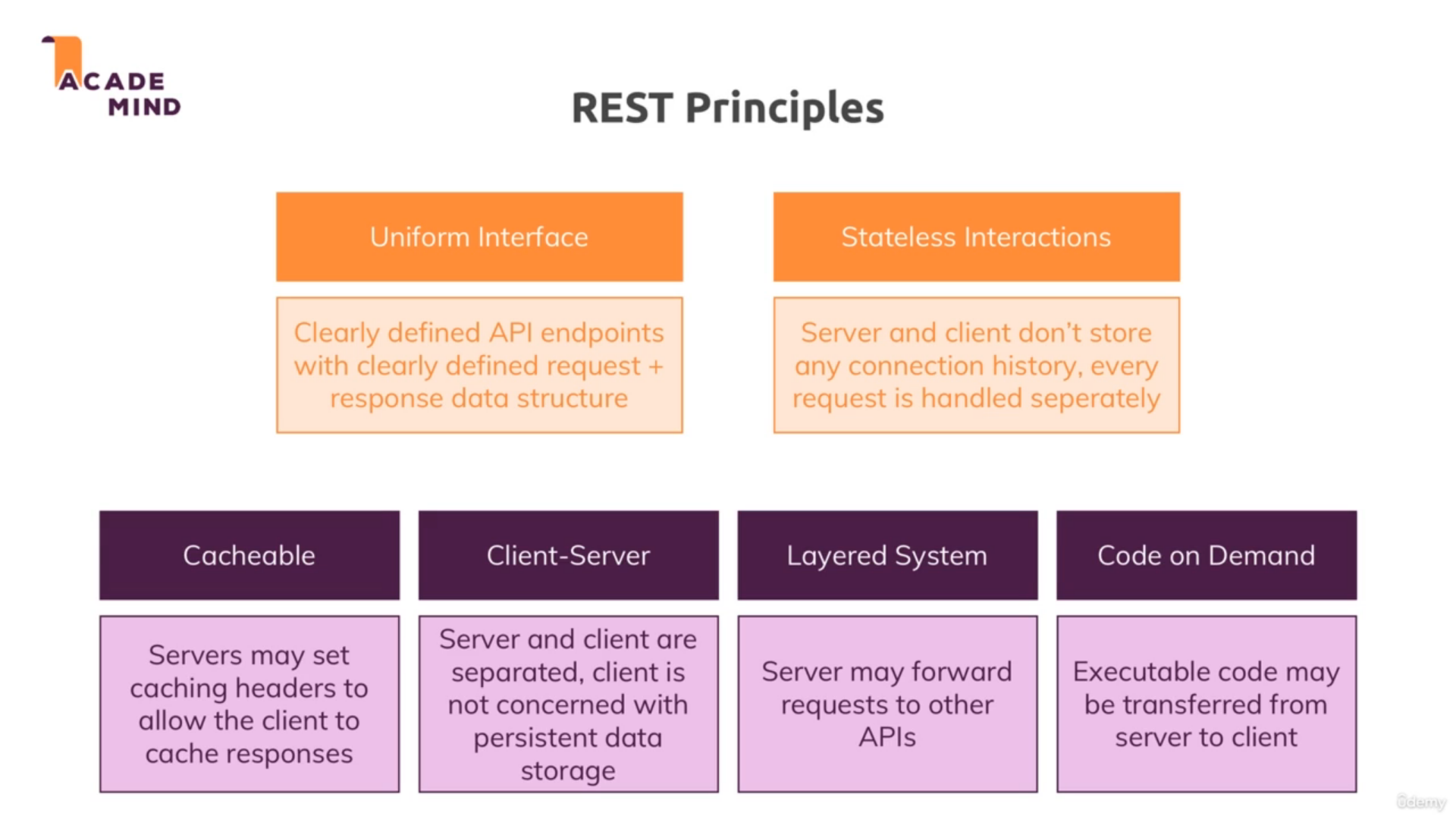
Rest API use to decoupled Frontent and ServerSide Logic. It help only to transfer data to frontend, it does not render whole Html Page on every request to API.



We use JSON(Javascript Object Notation) Format to transfer data from client to Server, However there are other formats Like XML(Extensible markup language) Similar to HTML, PlainText or HTML. But JSON format is extensively Used Everywhere.

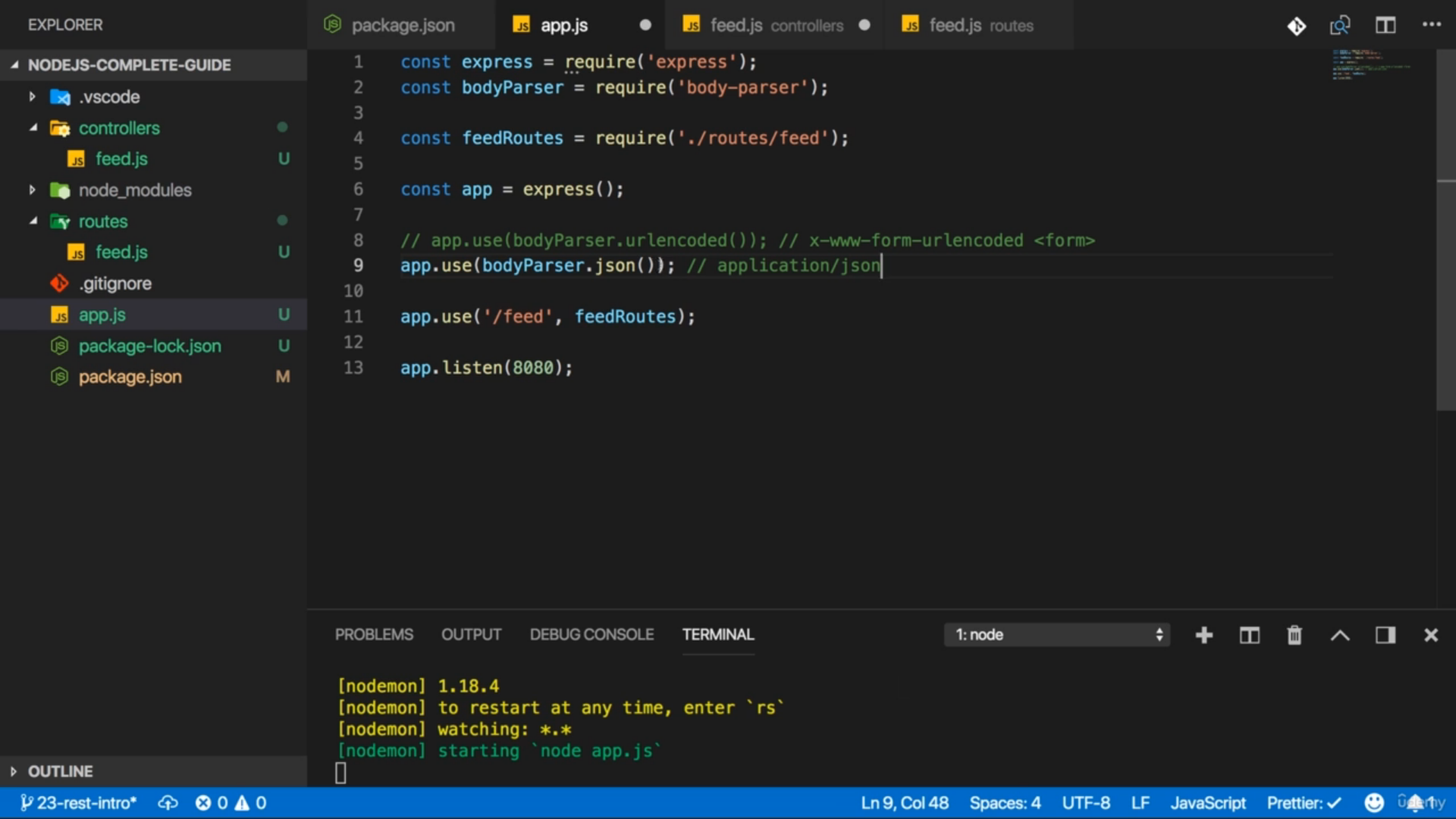


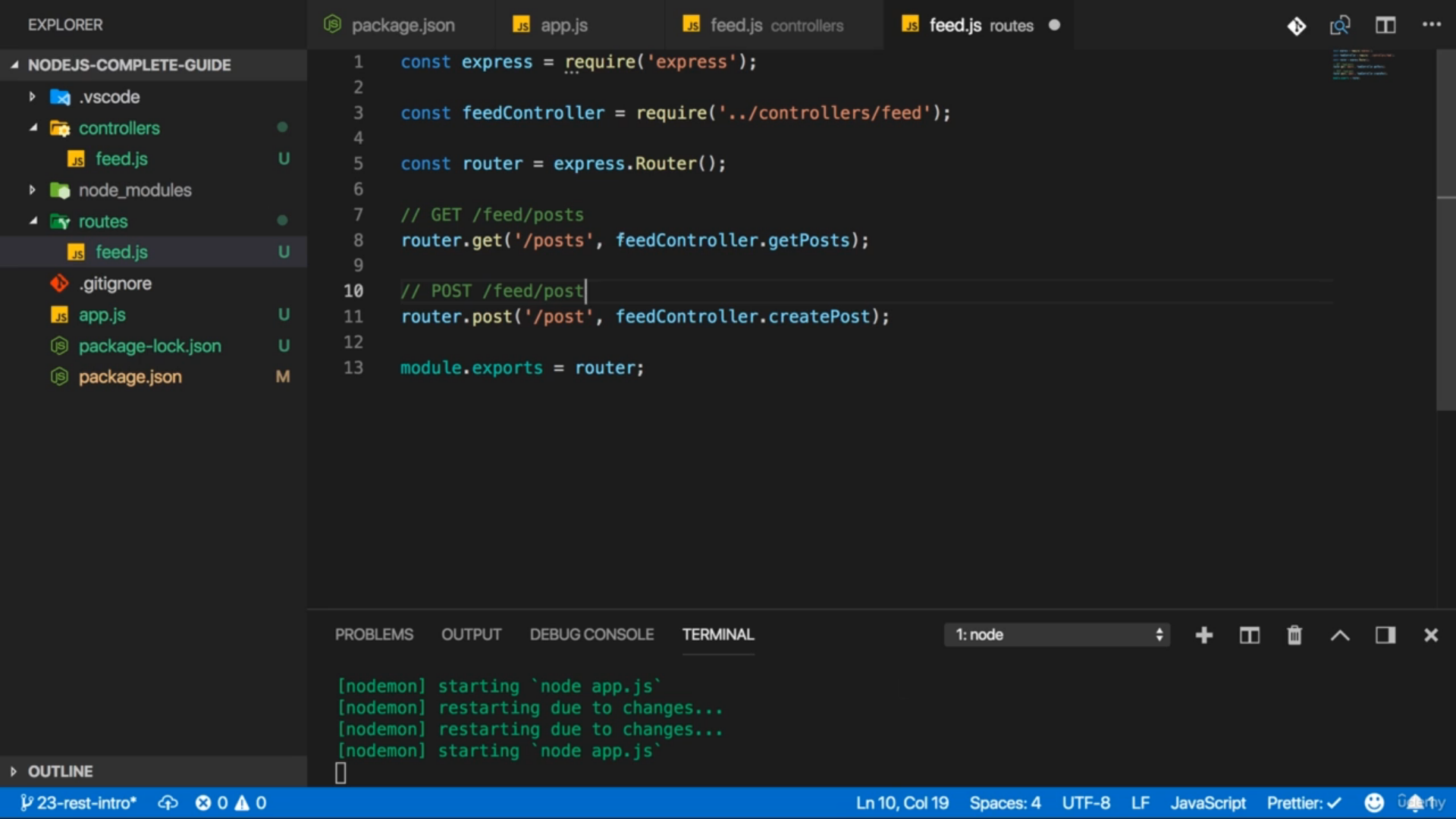




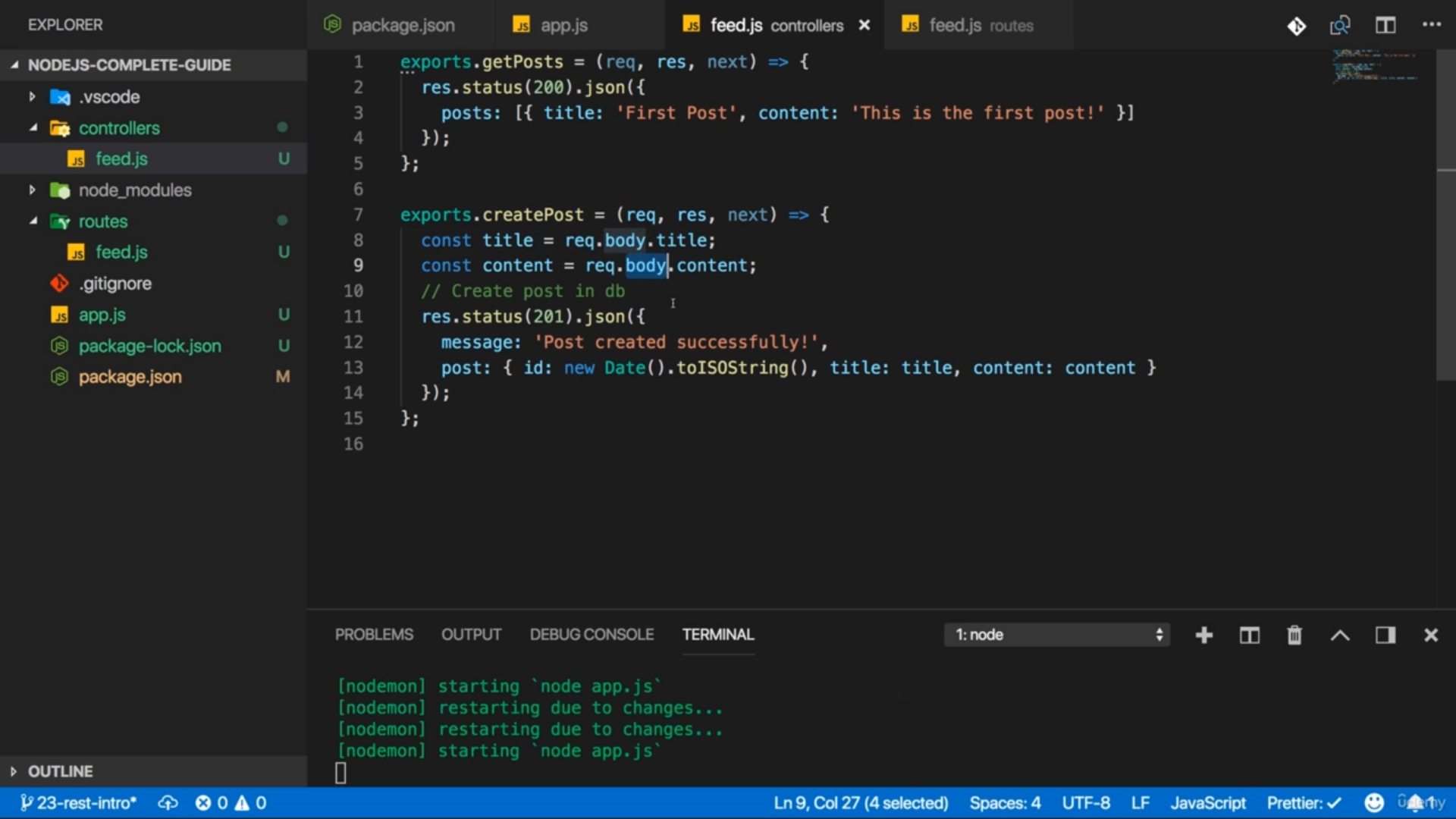
Creating simple Rest Endpoints, which will trigger controller logic present in controllers/feed.js

App.js

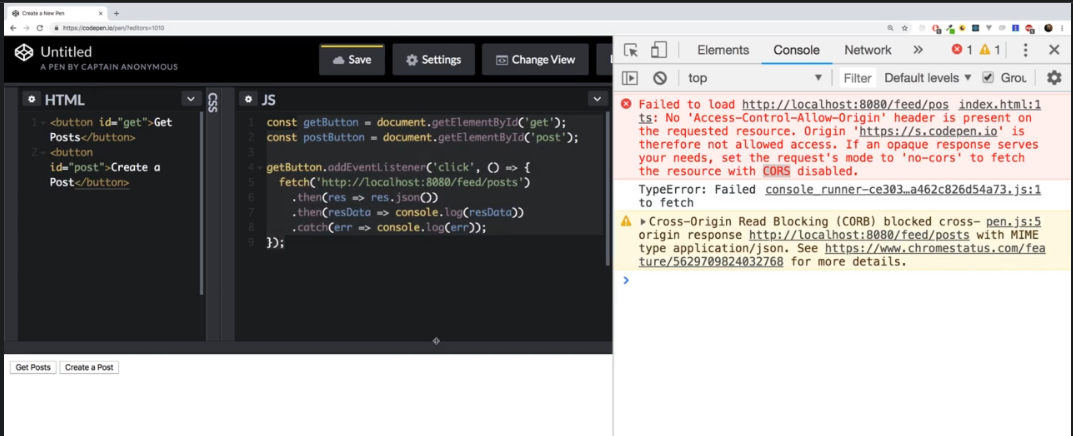


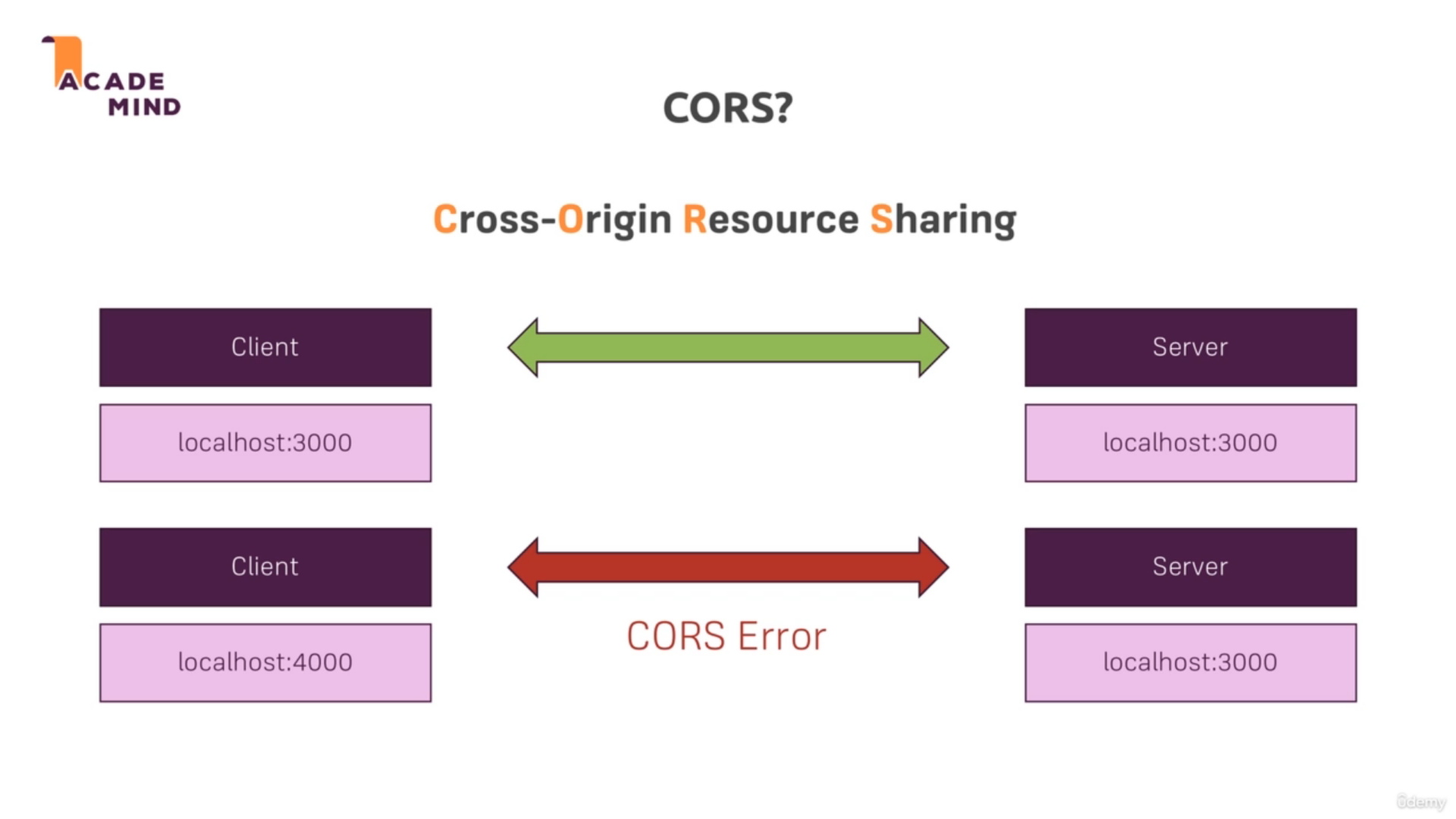


Feed.js

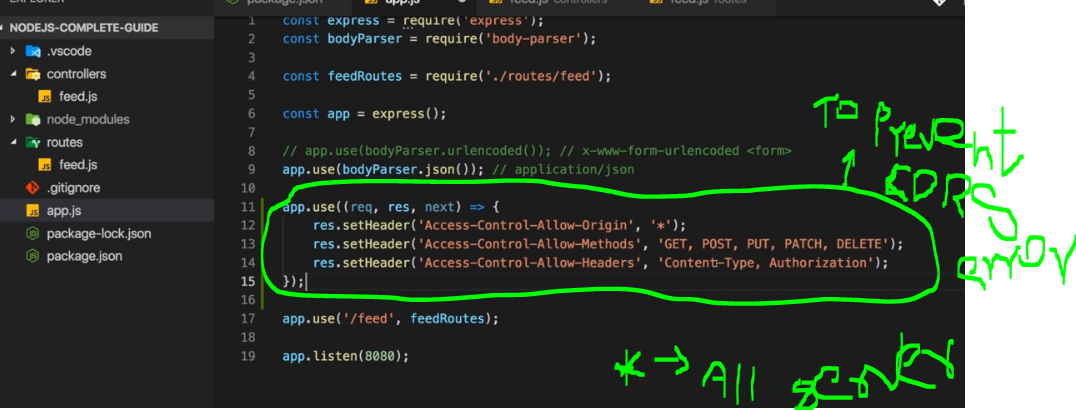


If client and server are on different port/Servers then browser prevent that and we can get CORS error.





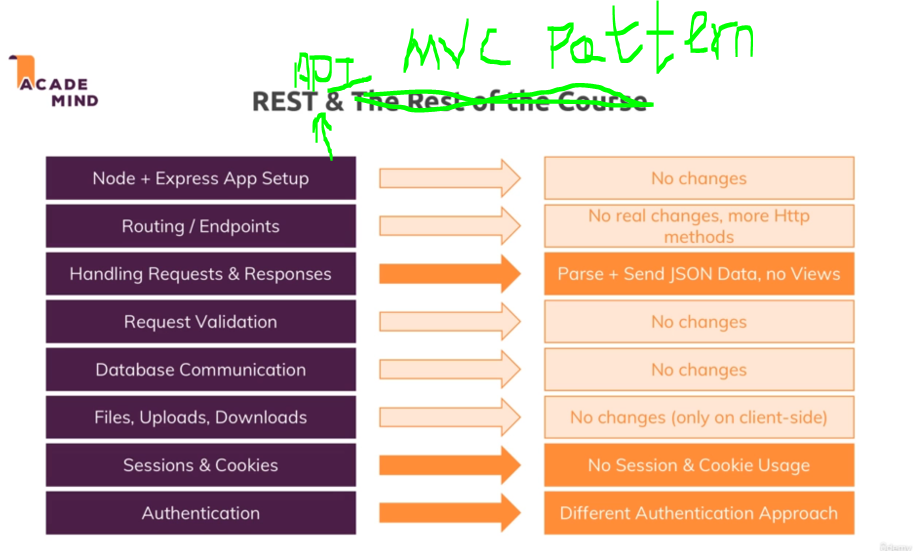
We do this to prevent



In REST API , you don’t need use session and Cookies for Authentication of every request after LoggedIn. Here we use Token which server send to client and client stored and use that token in other requests after LoggedIn.

This is JWT Authentication Method. Json Web Token

Token= header+Payload+signature



In MongoDb there are Collections instead of Table and documents instead of records.

Table change to -> collections

Rows change to -> document

It store in JSON/BSON –> BSON is basically Binary JSON itself

There is relationship achieved in Mongodb using Nested/Embeded Documents and References.

Connecting to MongoDb database using mongodbdriver:

Const mongodb = require(‘mongodb’)

Const MongoClient = mongodb.MongoClient

MongoClient.connect(“Mongodb database server Link”)

Mongoose (It help in elegant Mongodb Object Modeling )

It’s a document object Mapping library(ODM).

object data modeling (ODM) library that provides a modeling environment for your data. Used to interact with MongoDB, it makes life easier in managing data with various helper function and make writing query little bit easy.

Connection to Mongodb database using mongoose package

Const mongoose = require(‘mongoose’)

mongoose.connect(“database server link”)

with Mongoose we can work without Query.

CRUD operation in Mongoose.

For e.g we have direct save() function in mongoose, however using mongodb only we need to write insertQuery.

To fetch all data from database use find() method of mongoose.

To fetch single product use findById() method of mongoose.

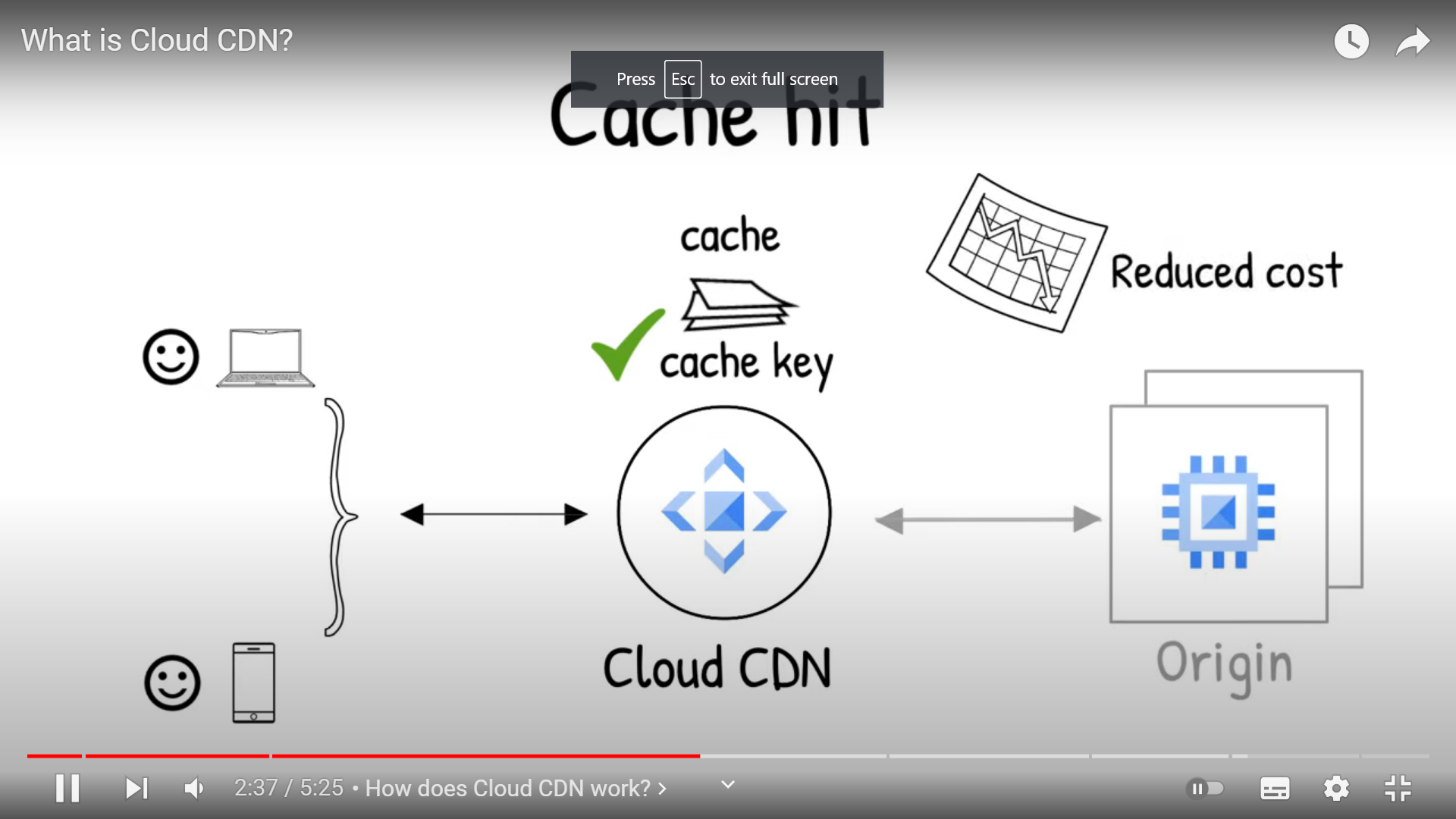
Update done by using findById and save() method of mongoose.

Delete done by using findByIdandRemove() method offer by Mongoose.

Payment Option Using Stripe payment gateway(3rd party)

Cloud CDN:

Content Delivery Network



Here origin is Backend Server

A content delivery network (CDN) refers to a geographically distributed group of servers which work together to provide fast delivery of Internet content.

A CDN allows for the quick transfer of assets needed for loading Internet content including HTML pages, javascript files, stylesheets, images, and videos.

Cloud CDN Brings the Servers close to the users, so that every request did not need to travel overseas servers, 1st request it will go to origin/original server through LoadBalancer then response will store in cloud CDN in Cache form, so next time request will receive data from cloud cdn instead of travelling to origin/original server back.

Load balancer: Every request will sent to server where load is minimum this is the function of Load Balancer.

What is NPM?

Npm is node package manager, it uses to manages package, dependency, libraries used in project., it help to install, update uninstall node packages.

There are also other package manager in Nodes like Yarn, pnpm

Dependency vs devDependency:

Dependency is used to run your application

devDependency is used while building your application which just make easy to build your application which provide useful resources, this don’t take part in main functioning of application. E.g Nodemon is devdependency

package.json: It records important metadata like dependencies / libraries used in a project

package.lock.json: If you're collaborating on a shared project with multiple developers, and you want to ensure that node dependency’s installations remain identical for all developers and environments, you need to use package-lock.json . package-lock.json is automatically generated for any operations where npm modifies either package.

### setImmediate vs setTimeout vs  process.nextTick() :

### priority of callbacks in the event Queue:

### process.nextTick() > setTimeout(cb,0) > setImmediate()

### setImmediate belongs to catch Queue like we have TaskQueue in event Loop.

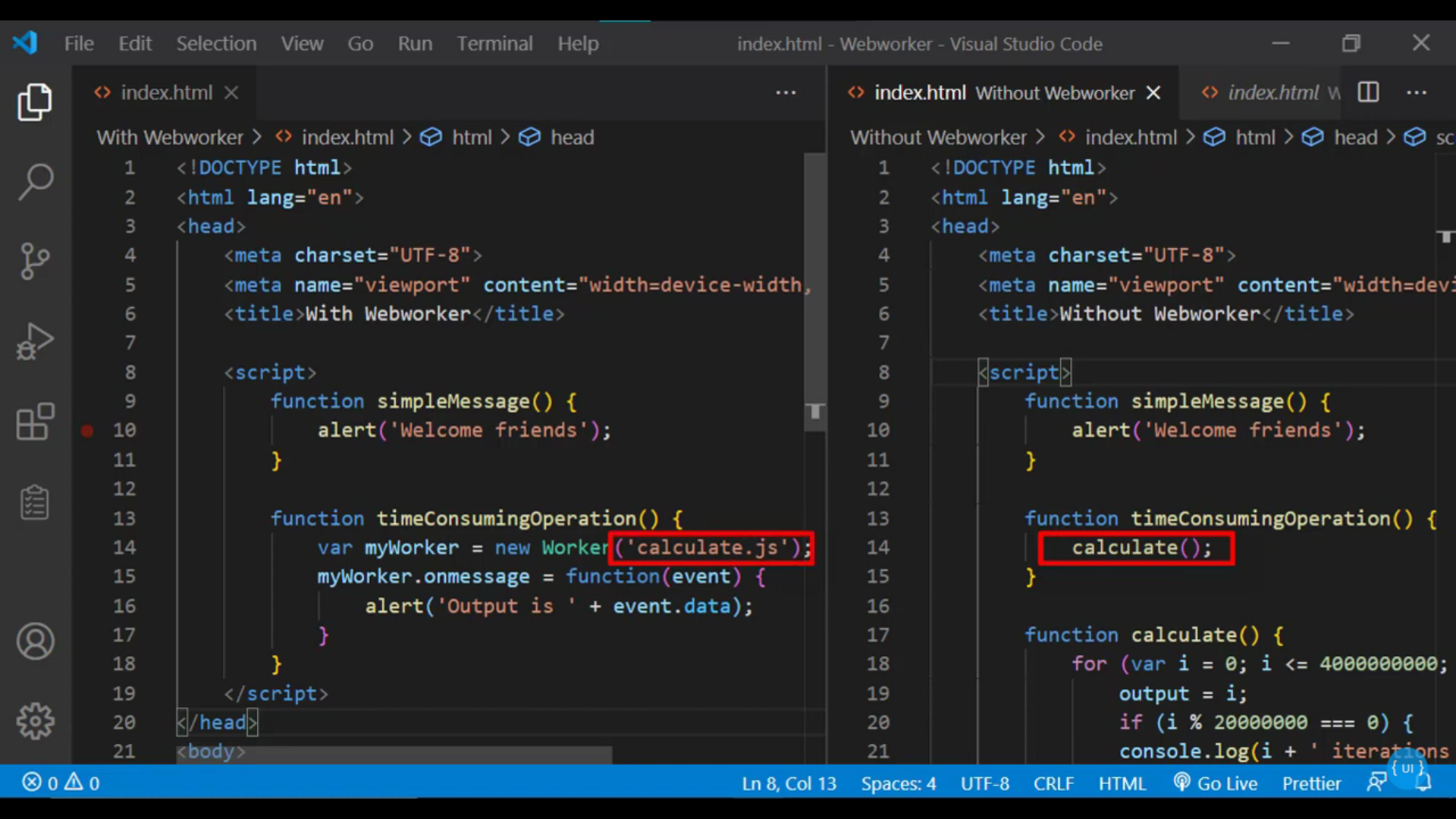
### 

Event Emmiter in NodeJs

As NodeJS is server, So click event in not there, so here events occur Like Opening or closing the files.

**What is a web worker used for?**

Web Workers are **a simple means for web content to run scripts in background threads**. The worker thread can perform tasks without interfering with the user interface.(e.g if some function in script is running and not completed ,till then u cant interfere the webpage it will throw error page unresponsive, so complex long calculation done by using web workers. It helps nodeJs to achieve multithreading.



FORK:

Although NodeJS is a **single-threaded** JavaScript runtime, it can create multiple subprocesses that run separately from each other, allowing you to **divide and run your application script from different NodeJS instances**.The child process/webworkers will have its own memory and runtime instance, so you need to have additional resources to allocate for each child process (Additional resources like making Multiple Request on server).

FORK is used when user hitting multiple requests on server, while web workers are used when some function in script takes too much time will make other UI interaction on browser fails.

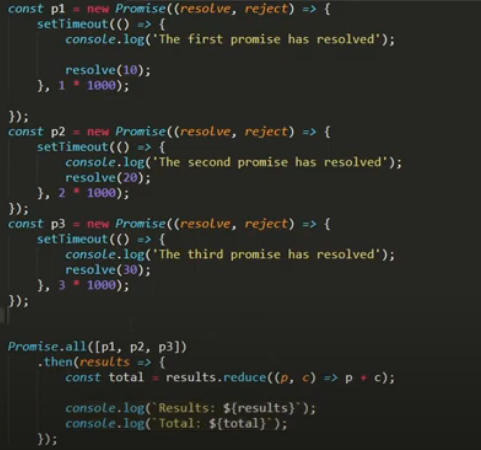
CSRF( cross site request forgery):

Suppose Hacker creates a fake final link of transfer money with all parameters (https:/bank.com?parameter1=”acntno”, paramter2=’ifsc code’) and send that link to the victim in form of image or button.

If victim already authenticated and loggedIn with that bank then, there will be session id stored in cookie. So at that time if user clicked on that fake link, then request of transfer money will get executed without further authentication, and money will get deposited in hacker account.

To prevent this attack, we generate CSRF Token using csrf library and send that token to Client, and in every Upcoming request client need to send that CSRF token to keep him as valid user.

Note: while Using JWT Authentication method the CSRF Issue is not there because there we already attaching JWT Token in headers of every request.



If we want to execute promise parallely, then we can use promise.all() ,

Note this will only helpful when app promise resolve and no one get reject

## Difference Between WebSocket and Socket.io

**WebSocket** is the communication Protocol that provides bidirectional communication between the Client and the Server over a TCP connection; WebSocket remains open all the time, so they allow real-time data transfer. When clients trigger the request to the server, it does not close the connection on receiving the response;

**Socket.IO** is a library that enables real-time and full-duplex communication between the Client and the Web servers. It uses the WebSocket protocol to provide the interface.