25-09-2021

Phase 3 :

13 session

Node JS

1. Overview of Node
2. Node JS modules : core module, local module external module
3. Fs module
4. Url module
5. http module
6. express module
7. creating web application using Http and express module
8. Mongo DB database
9. Connection mongo db database using node js with help of mongodb and mongoose module
10. Connecting Express JS with Mongoose with Mongo Db Database
11. Express JS MVC application
12. MEAN CRUD Operation -🡪 Angular -🡪Express -🡪 Mongoose -🡪 Mongo DB using MVC style.
13. Socket programming
14. Authentication and Authorization
15. Interceptor

Html

Css

JavaScript

jQuery : jQuery is a external library which provide lot of pre-defined function which internally connected to each other to do read, write and update operation very easily.

Library and framework

Backbone js

Coffee js

Ext JS

Node JS

Angular JS

Angular Framework

React JS

Vue JS

Node JS is not a library or not a framework. It is a run time environment for JavaScript programs.

Front end Team backend Team

HTML / HTML5 Java or Spring boot

CSS/CSS3 Asp.net

JavaScript with ES5 and ES6 Php

Bootstrap Python

jQuery CGI

Node JS

After node JS using JavaScript we can create client side as well as server side program or application.

Node JS provide lot of pre-defined module or external module which help to do the program using JavaScript like doing file handling programs, creating dynamic web application, creating REST API, connecting database etc.

Node JS module we can’t run on the browser.

Inside Node JS program we can’t use DOM(Document Object Model) and BOM (Browser Object Model).

Node REPL (Read Eval Print and Loop)

Node JS modules

Module in node JS is a simple or complex functionality organized in a single or multiple JavaScript file which can be re-use throughout the Node JS application.

3 types

1. Core module
2. Local module or user-defined module
3. Third party or external module

npm install typescript

npm install @angular/cli

syntax to load the module

var/let referencName = require(“moduleName”);

fs : file system module

fs is a type of core module which provide pre-defined API which help to do sync as well as async file handling program using JavaScript.

fs.writeFile(filePath,data,callback);

Read and write Json Data using JS module.

Node JS provide two pre-defined global object is

console

process

26-09-2021

Taking the value through keywords in Node JS

readline : readline is a core module which help to take the value through keyboards.

readline-sync : This is external module which help to take the value through keyboards as

synchronously.

npm install readline-sync –g

http module

node js provide predefine core module ie http module which help to create the server using Node JS application.

Non node js server side application like Java, Asp.net, Php or Python. These language will take the help of other server like tomcat, web logic, IIS, JBoss are server.

Node JS itself provide the server to deploy the application.

Normal Server Vs Node JS Server

Gmail or google or any other application running on server.

Thread : it is small execution of a code within a process. Number client means number of thread will created.

1st Client

2nd Client

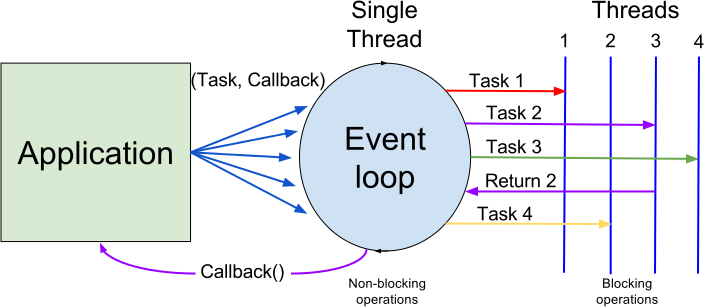
3rd Client

Tomcat server can takes 1000 request concurrently.

All server are thread base when number of client increate the thread get block or lock.

Node JS improve this concept using Event loop driven architecture framework.

JavaScript and Node JS is single thread application.



http : it is pre-defined core module which help to create server side application using node js as well as we can run this application on node js server.

First load the module

let http = require(“http”);

using reference call createServer function which takes callback function as a parameter with two parameter request and response. Request is use to receive the request from a client and response is use to give the response back to client.

In createServer function we can pass callback function in function style or arrow style.

Sample application

simple\_http\_app.js

let http = require("http");

let server = http.createServer(function(req,res){

res.end("Welcome to Simple Http Client Application");

});

server.listen(9090,()=>console.log("Server running on port number 9090"));

node simple\_http\_app.js

After run the application

Open the browser and type as a URL

<http://localhost:9090/>

url module : node js provide core module ie url module which help to provide the URL Details.

url.parse(urlInfo) : here is query consider a string.

url.parse(urlInfo,true) : here the query consider a reference. Where query property consider as reference then we can extract property from a query.

http route : base upon the URL path we want to display the different message.

<http://localhost:9090/aboutus>

<http://localhost:9090/contactus>

<http://lcocalhost:9090/login>

http is a base or core module which provide very basic core functionality.

Node js provided lot of pre-defined third partly modules which help to create dynamic

Web page using Node JS.

Express JS

Koa JS

Hapijs

Geddy js

Socket io

Etc

All third party framework internally use Http core module they wrap http module and provided extra functionality.

Express JS

MEAN Stack : Express JS

MEAN , MERN, MEVN

Express JS is a open source web application framework for Node JS application. It provides various extra features that makes web application development very fast and easy otherwise more time take using Http module.

Html, css and Javascript

DOM and BOM

In Node JS application we will create package.json file which contains all node js application configuration details.

Syntax to create the package.json file

npm init

It will ask the package name : please give the packagename(provide meaningful name ie one word).

Then enter the key continuously and at the last hit yes.

npm install express –g globally

npm install express locally

express module support http protocol methods

get

app.get(“path”,callback): callback with request and response parameter.

Node js provided pre-defined global property ie \_\_dirname : This property provide use current directory path.

post

in Express JS application if we pass the data through post method by default express js can’t receive the value.

To enable data from the request we have to take the help of one of the external module ie body-parser. This body-parser we have to use as a middleware (between client and server application).

In old version body-parser separately we have install as local module but in new version express js module contains body-parser module.

Express with SignIn and SignUp with FS Modules

Create the package.json file using npm init

Then install express modules

npm install express

03-10-2021

In Plain HTML or Normal HTML web page we can’t use any dynamic code.

Express JS provided view engine or template : If we use these view engine or template.

In This view page we can do some dynamic task.

Jade

If we use Express JS view engine the view technologies become tightly coupled on backend

Technologies.

View (HTML) or Jade -🡪 Express (HSBC ) --- Banking Application

Or online shopping application

Express ---- > Payment

PayTm Java

Google pay Asp.net

Phone Pay Python

Net banking Php

Two web application they can’t communicate to each other if they develop in different language. Like Java, Python, Node Express JS or Aps.net etc

Web Service : Giving the service for web application when two application running using different technologies in same os or different OS.

2 types

1. SOAP Web Service
2. RestFull Web Service

SOAP : Simple Object Access Protocol. SOAP web service is base upon SOA(Service Oriented Architecture). SOAP is standard.

SOA (optional)

SB (Service Broker) google, just dial

Service Requester Service Provider

Or

Service Consumer

(optional)

SB (Service Broker) google, just dial

WSDL

Service Requester Service Provider

Or

Service Consumer SOAP Req and SOAP Response

It is only in XML format.

Function or

Node Express Method

Java Writing in

Asp.net or Java or python

WSDL Web Service Description language : It is a type of xml file which provide our service details

SR download the WSDL file and using some tools they create respective language code and call the service. Using that code we will send the request and service provider and get the response from service provider.

Limitation of SOAP Web service we can consume and produce the data only in the form of XML. XML is heavy to transfer the data from one application to another application.

JSON : Java Script Object Notation.

If we want to share the data between two technologies we have to use xml or json.

REST full Web service Representational State Transfer. So we can consume and produce the data between two technologies in any format like xml or json or plain text or html or etc. json is light weighted.

REST Web service using Http protocol all method to consume and produce the data.

Creating REST API using Express JS : So if we make Express JS application as a REST API any other technologies can invoke or call or consume or produce our application like Java, Asp.net, Python, Angular or React or Any REST Client application can call to consume as well as produce the data.

Swagger is use to create the documentation for the REST API.

Postman it is very simple plugin to test our rest api using browser.

Create new project REST API using Express JS

Then create the package.json file using npm init.

Then install express js locally

npm install express

According the REST API Resources means any entity like Customer, Account, Bank, Product, Employee, Manager

REST API with GET method

Select query

Get the resources

1. Return simple text message
2. Return simple json data
3. Return object in json format
4. Return array of object in json format
5. Pass value using get method
   1. Query param
      1. URL?key=value : single value
      2. URL?key=value&key=value&key=value

By default html form with get method internally use query param concept to send the data through URL.

* 1. Path param
     1. URL/value1 : single value
     2. URL/value1/value2/value3 : multiple value

Get Employee details, Get all customer info, Get Employee information base upon id, get order details base upon range date etc.

REST API with POST method

Post method is use to create the resources. Storing the entity details, Store Employee records, Order details, Product information.

Insert query

We can’t send the data for express js post method through URL.

1. We have to use form with method post
2. Browser Plugin : postman, rest client for chrome or etc.

In Express JS by default we can’t consume the data from a body part so we have enable using middleware.

Put method : Put method is use to update the resource

Using employee id update salary, using employee id update employee age, using product id update product price.

In post method we have to pass full object. In put method we can pass partial object.

Update query in database.

Delete method : delete method is use to delete the resource

Using employee id delete employee records, using order id delete order information, using customer id delete customer details etc.

Delete query in database

09-10-2021

Create folder product management system

Backend -🡪 REST API using Array or FS module

Create backend folder

Create package.json file using npm init

Install express js module

npm install express

now create app.js

create API

1. Get all products
2. Get product by id
3. Store product
4. Update product price
5. Delete product using id

Front end -🡪 Angular

In Frontend folder create new angular project

ng new angular-product-operation

ng g c product : this command use to create component

ng g s product : This command is use to create service

ng g class product : This command is use to create produce model class

we are running two server

Angular application running on port number : 4200

Express JS Application running on port number 9090

Two domain or server going to communicate to each others.

CORS : Cross Origin Resource Sharing

In backend technologies develop in any language they have to enable cors policy.

Node JS provide external module ie cors

So we have to install this module

And add as middleware module in express js

npm install cors

// this class map to json data retrieve from backend technologies using any language

export class Product {

    constructor(public pid:number,

        public pname:string,

        public price:number,

        public address:Address){

    }

}

class Address {

    constructor(public city:string,public state:string){}

}

10-10-2021

Angular provide lot pre-defined pipe (filter). Which help to filter the data while displaying in template.

Uppercase

Lowercase

Date

Currency

Json

Async

Syntax

{{variableName or objectName | pipeName}}

16-10-2021

File system limitation

1. File system allow data redundancy (duplicate records we can store).
2. Data consistence in file system( .txt, .doc, .pdf)
3. Security

Database

Data : raw facts.

Information : processed data or meaningful data.

Database : storing the data in table format using row and columns.

DBMS : Database Management System : it is a software which help to store the data in table format.

Excel : is a type of DBMS

RDBMS : Relational Database Management System.

Trainer\_Students

DBMS

TId Name Tech SId SName Age …………..

1 Raj Java 100 Seeta 21

1 Raj Java 101 Reeta 22

1 Raj Java 102 Meeta 23

2 Ravi Python 103 Veeta 24

Dr EF . Codd’s Rules 12 rules

Trainer

PK(primary key)

TId TName Tech

1 Raj Java

2 Ravi Python

Student

PK(primary key) FK

SId SName Age TSId (this column is use to refer to pk to other table)

100 Seeta 21 1

101 Reeta 22 1

102 Meeta 23 1

103 Veeta 24 2

Family table

PK FK

MemberId Name Age ReportId

1 Dad 70 null

2 Mom 65 1

3 Sister 42 1

4 Brother 38 2

5 Me 34 2

RDBMS provide SQL (Structure query language) which help to interact with the database.

MySQL, Oracle, Postgres SQL, Db2 , SQL Server 2020 etc they all are RDBMS database

These all databases use common language ie SQL.

90% query are same but 10 to 15% change.

create database databaseName;

**Limitation of SQL Databases**

1. All SQL database are schema bases. Before storing the records in all RDBMS database first we have to create the table with number of columns(column names and their data types). and values to store.
2. Once we created the table if we want to change the table structure it is very difficult.

Employee Table

Id Name Salary Age Phnumber City

1 Raj 12000 null null null

2 Reeta 14000 null null null

3 Meeta 16000 21 null null

4 Teeta 18000 null 99001234 null

5 Keeta 22000 21 null Bangalore

No SQL Databases

Key-value : redis

Graph database : Neo4j

Document Oriented : Mongo DB

Column family : Cassandra, HBase

In databases we can store the data in semi structure format.

MonoDB is a type of No SQL database which is use to store the data in document format.

Mongo DB is schema less database. We can store the data using document concept.

Front end side : Angular or React JS know how to consume JSON Database.

Node JS or Java or Python or Php or Asp.net if they are REST API then known how to consume and produce the JSON data.

If we use RDBMS database then we have to convert our JSON data into table format and vice-versa.

Mongo DB provided great features for us we can store the data in mongo DB in JSON format.

RDBMS MongoDB

Database Database

Table Collection

Record or tuple document

Column or field fields

Window user in C Drive

Create one folder with name data and inside data folder create db folder

Window user :

Open the command prompt in this location

C:\Program Files\MongoDB\Server\5.0\bin

mongod : This command is use to run monodb sever.

Mac User

only run mongod command in terminal

Window user open another command prompt in same location and type as

mongo This command is use to run the mongo terminal

mac user in another or same terminal run the mongo command.

All mongo DB commands

show dbs

Or

show databases

These command is use to display all the databases present in mongo DB database.

To create the database we can use the command as

use databaseName : if database already present it will switch to that database else it will create and switch to that database.

show collections

Or

show tables;

These command is use to check all collections (tables) in current database.

db.createCollection(“Sample”);

This command is use to create the collection.

db.collectionName.insert({key1:value1,key2:value2,key3:value3})

key can be double quote or without quote (by default quote consider). Value may be type of number, string, boolean, primitive array type, complex type or complex array type.

db.Sample.insert({name:”Raj”});

**view the document from a collection**

db.collection.find();

In RDBMS Primary key optional So we can create table in RDMBS database without primary key and we can store duplicate.

Bu in Mongo DB default every document consider as a unique document if we store same types of fields with same values. By default for every document mongo db internally it will create \_id as a field to maintain the unique ness between two documents. So \_id is a like a primary key in mongo db database.

So if we didn’t pass the value for \_id it automatically create the unique id value using ObjectId data types. if you want to pass custom unique value we can pass but field name can’t change.

Employee

\_id Name Age City

In mongo DB we can insert the document in collection. If collection is not present it automatically create the collection and add the document. If collection is present it will document to existing collection.

If you want to retrieve specific document using index position

db.Employee.find()[0];

db.Employee.find()[1];

retrieve specific fields value using index position.

db.Employee.find()[1].city;

db.Employee.find()[2].\_id;

if we want to retrieve more than one fields values.

db.CollectionName.find({condition},{specificFields1,specificField2….});

db.Employee.find({},{name:1}) : This query display all name and \_id(pre-defined property by default it will display)

db.Employee.find({},{name:1,\_id:0}) : This display all document names fields only

db.Employee.find({},{name:1,\_id:0,age:1}) : This display all document name and age fields

display more than one field with specific index position

db.Employee.find({},{name:1,age:1,\_id:0})[1];

db.Employee.find({},{name:1,age:1,\_id:0})[2];

skip() and limit() functions

db.Employee.find().skip(2); This query is use to skip starting n documents from collections

db.Employee.find().limit() : This query is use to display the n number of document from a collections.

sort() functions

db.Employee.find().sort({age:-1}) : it display all the document age descending order.

db.Employee.find().sort({age:1}) : it display all the document age ascending order.

Multi field sort example

db.Employee.find().sort({city:1,age:-1})

retrieve the documents from collection using conditions.

Using equal operator

db.CollectionName.find({condition});

db.Employee.find({\_id:100});

db.Employee.find({name:"Raju"});

db.Employee.find({city:"Bangalore"});

relational operator

db.CollectionName.find({field:{operator:value}});

db.Employee.find({age:{$gt:25}}); >

db.Employee.find({age:{$gte:25}}); >=

db.Employee.find({age:{$lt:25}}); <

db.Employee.find({age:{$lte:25}}); <=

**db.Employee.find({age:{$eq:25}}); or db.Employee.find({age:25}) =**

**db.Employee.find({age:{$ne:25}}); !=**

**to check both conditions true or false**

**$and $or operator**

**db.Employee.find({$and:[{name:"Raj"},{age:21}]}); : both condition must be true**

**db.Employee.find({$or:[{name:"Raj"},{age:25}]}); : any one condition must be true**

**17-10-2021**

**Update the documents**

**db.CollectionName.update({condition},{$set:{key:value}});**

**db.Employee.update({\_id:100},{$set:{age:30}}); : update only one value**

**db.Employee.update({\_id:101},{$set:{name:'Raju Patel',age:32}}); : update more than value fields value**

**db.Employee.update({$and:[{\_id:100},{age:30}]},{city:”Mysore”}); : it check conditions if the condition is satisfies then for existing document add 2nd curly braces fields and remove old fields.**

**db.Employee.update({$and:[{\_id:100},{age:30}]},{$set:{city:"Mysore"}}); : both conditions must be satisfies then update one value**

**db.Employee.update({\_id:100},{$set:{name:"Ram Kumar",age:21}});**

**$set operator change the fields value base upon the condition if fields already present in existing document if fields not present then it will add that fields to existing documents.**

**Remove the fields from a specific document using conditions.**

**db.Employee.update({\_id:106},{$unset:{city:1}});**

**db.Employee.update({age:{$gt:24}},{$set:{city:"Delhi"}});**

**if query satisfies more than one document but still it will update only one document.**

**Ie first documents where the conditions satisfies.**

**So if you want to update more than one document then you can use**

**db.Employee.update({age:{$gt:24}},{$set:{city:"Delhi"}},{multi:true});**

**or**

**db.Employee.updateMany({age:{$gt:24}},{$set:{city:"Delhi"}});**

**We will make collection with array values.**

**db.Students.update({\_id:102},{$push:{skillSet:"Vue JS"}});**

**this command is use to value of array fields for the existing documents.**

**db.Students.update({\_id:102},{$pop:{skillSet:1}});**

**This command is use to remove last element or top most elements.**

**Conditions for fields with contains array values**

**db.Students.find({"skillSet":"C"}); : C value may be in any position in skillset field**

**db.Students.find({"skillSet.0":"C"}); C value must in 0 position in skillset field**

**db.Students.find({"skillSet.1":"C"}); C value must in 1 position in skillset field**

**db.Students.count(); This command is use to find number of document present in collections.**

**db.Students.find({\_id:100}).count(); This command is use to find number of document with conditions.**

**Remove query**

**db.CollectionName.remove({condition});**

**db.Employee.remove({\_id:106});**

**db.Employee.remove({$and:[{\_id:105,name:’Ravi’}]});**

**db.Employee.remove({age:{$gt:21}});**

**Mongo DB relationship**

**In Mongo DB we can achieve the relationship using 2 ways**

1. **Embedded style relationship : in this style we will store all records in single collection.**
2. **Linking style relationship : in this style we will store all records in more than one collections.**

**Relationship**

**One to one**

**One to many**

**Many to one**

**Many to Many**

**Embedded style relationship**

**One to one relationship**

**One to many relationship**

**Employee Address**

**Id, name, age city,state pincode**

**{**

**\_id:100,name:”Ravi”,age:21,address:{city:”Bangalore”,state:”Kar”,pincode:560096},**

**\_id:101,name:”Ramesh”,age:24,address:{city:” Mumbai”,state:”Mh”,pincode:41004},project:{pid:1122,tech:Java},**

**\_id:102,name:”Ajay”,age:26,address:{city:” Mumbai”,state:”Mh”,pincode:41004},project:[{pid:1122,tech:”Java”},{pid:1133,tech:”Python”}]**

**}**

**First document one to one relationship**

**Second document one to one with two embedded collection**

**Third document one to one and one to many**

**db.Employees.insert({\_id:100,name:"Ravi",age:21,address:{city:"Bangalore",state:"Kar"}});**

**db.Employees.insert({\_id:101,name:"Ramesh",age:23,address:{city:"Mumbai",state:"Mh"},project:{pid:1111,tech:"Java"}});**

**db.Employees.insert({\_id:102,name:"Raju",age:25,address:{city:"Mumbai",state:"Mh"},project:{pid:2222,tech:["Java","Oracle"]}});**

**db.Employees.insert({\_id:104,name:"Ajay",age:25,address:{city:"Mumbai",state:"Mh"},project:[{pid:111,tech:"java"},{pid:3333,tech:"Python"}]});**

**db.Employees.find().pretty();**

**This query to apply the condition of complex property**

**db.Employees.find({"address.city":"Mumbai"}).pretty();**

**db.Employees.find({"project.tech":"Java"}).pretty();**

**Linking Style relationship**

**Trainer**

**\_id TName tech**

**100 Raj Java**

**101 Ram Python**

**102 Raju Angular**

**Student1 : only storing trainer id as a foreign key(Mongo DB doesn’t support FK)**

**\_id SName Age TSID**

**1 Seeta 21 100**

**2 Meeta 22 100**

**3 Veeta 23 101**

**4 Keeta 24 102**

**5 Leeta 25 [102,100]**

**Student2 : Storing the complete trainer details in Students**

**\_id SName Age TSID**

**1 Seeta 21 {100 Raj Java}**

**2 Meeta 22 {100 Raj Java}**

**3 Veeta 23 {101 Ram Python }**

**4 Keeta 24 {102 Raju Angular }**

**5 Leeta 25 [{102 Raju Angular },{100 Raj Java}]**

**Trainer collection**

**> db.Trainer.insert({\_id:100,tname:"Raj",tech:"Java"});**

**> db.Trainer.insert({\_id:101,tname:"Ram",tech:"Python"});**

**> db.Trainer.insert({\_id:102,tname:"Raju",tech:"Angular"});**

**Student1 Collection : Storing TrainerId as a foreign key.**

**> db.Student1.insert({\_id:1,sname:"Seeta",age:21,tsid:db.Trainer.find()[0].\_id});**

**> db.Student1.insert({\_id:2,sname:"Meeta",age:22,tsid:db.Trainer.find()[0].\_id});**

**> db.Student1.insert({\_id:3,sname:"Veeta",age:23,tsid:db.Trainer.find()[1].\_id});**

**> db.Student1.insert({\_id:4,sname:"Keeta",age:24,tsid:db.Trainer.find()[2].\_id});**

**> db.Student1.insert({\_id:5,sname:"Leeta",age:25,tsid:[db.Trainer.find()[0].\_id,db.Trainer.find()[2].\_id]});**

**In this approach if you want to retrieve documents from more than one collection we have to use aggregate operator or functions.**

**Student2 Collection: Storing complete trainer details in Student documents.**

> db.Student2.insert({\_id:1,sname:"Seeta",age:21,tsid:db.Trainer.find()[0]});

> db.Student2.insert({\_id:2,sname:"Meeta",age:22,tsid:db.Trainer.find()[0]});

> db.Student2.insert({\_id:3,sname:"Veeta",age:23,tsid:db.Trainer.find()[1]});

> db.Student2.insert({\_id:4,sname:"Keeta",age:24,tsid:db.Trainer.find()[2]});

> db.Student2.insert({\_id:5,sname:"Leeta",age:25,tsid:[db.Trainer.find()[0],db.Trainer.find()[2]]});

db.Employee.find({name:{$regex:"^R"}}); start with R character

**db.Employee.find({name:{$regex:"h$"}}); end with h character**

23-10-2021

Aggregate function or aggregate operators

db.StudentInfo.insertMany([

{\_id:100,name:"Raj",salary:45000,deptId:100,city:"Bangalore"},

{\_id:101,name:"Ravi",salary:25000,deptId:101,city:"Mumbia"},

{\_id:102,name:"Ramesh",salary:42000,deptId:102,city:"Delhi"},

{\_id:103,name:"Rajesh",salary:35000,deptId:100,city:"Bangalore"},

{\_id:104,name:"Lokesh",salary:35000,deptId:100,city:"Bangalore"},

{\_id:105,name:"Ajay",salary:15000,deptId:101,city:"Mumbai"},

{\_id:106,name:"Vijay",salary:22000,deptId:102,city:"Delhi"},

{\_id:107,name:"Mahesh",salary:26000,deptId:102,city:"Delhi"},

{\_id:108,name:"Dinesh",salary:28000,deptId:103,city:"Mumbai"},

{\_id:109,name:"Raghu",salary:32000,deptId:103,city:"Mumbai"},

{\_id:110,name:"Nagesh",salary:46000,deptId:100,city:"Bangalore"}

]);

Aggregate function is use to group multiple document and them perform aggregation function or operation on and it return a single or multiple result depending upon the operators.

db.Trainer.aggregate([{$lookup:{from:"Student1",localField:"\_id",foreignField:"tsid",as:"StudentDetails"}}]).pretty();

$group : this operator is use to combine more than one document property values.

db.StudentInfo.aggregate([{$group:{\_id:"$city"}}]);

db.StudentInfo.aggregate([{$group:{\_id:"$deptId"}}]);

Total

db.StudentInfo.aggregate([{$group:{\_id:"$city",totalSalary:{$sum:"$salary"}}}]);

max

db.StudentInfo.aggregate([{$group:{\_id:"$city",maxSalary:{$max:"$salary"}}}]);

min

db.StudentInfo.aggregate([{$group:{\_id:"$city",minSalary:{$min:"$salary"}}}]);

avg

db.StudentInfo.aggregate([{$group:{\_id:"$city",avgSalary:{$avg:"$salary"}}}]);

count

db.StudentInfo.aggregate([{$group:{\_id:"$city",totalEmp:{$sum:1}}}]);

$match operator like a having clause in RDBMS

db.StudentInfo.aggregate([{$match:{city:"Bangalore"}},{$group:{\_id:"$city",totalEmp:{$sum:1}}}]);

db.StudentInfo.aggregate([{$match:{city:{$in:["Bangalore","Delhi"]}}},{$group:{\_id:"$city",totalEmp:{$sum:1}}}]);

db.StudentInfo.aggregate([{$match:{city:{$nin:["Bangalore","Delhi"]}}},{$group:{\_id:"$city",totalEmp:{$sum:1}}}]);

Index

Index concept in SQL and No SQL Database mainly use to improve the performance.

Collection -🡪 100000

Index is a type of special data structure that store small portion of the collection data set in an easy way to search the data.

db.StudentInfo.getIndexes();

{ "v" : 2, "key" : { "\_id" : 1 }, "name" : "\_id\_" } ]

db.StudentInfo.createIndex({deptId:1});

db.studentDetails.createIndex({phnumber:1},{unique:true}); This syntax is use to create the

unique index

Connecting Mongo DB Database using Node JS Application

Node JS provides External Module ie

1. mongodb : mongodb is type of external module which help to connect the Mongo DB database and we can do Insert, Delete, Update and retrieve documents from collections.

Mongodb is external module we have to install using npm command

Create package.json file using command as

npm init

npm install mongodb