**Explanation is here but source code is inside todo app folder.**

**Instance, class**

Class is just structure. It may have constructor, method, property, variable.

Instance: when we use new keyword with class we get a instance of this class. Inside the instance we can access all the property of this class. If we use p1 = new Person(); this class with p1 instance. Suppose we have some method inside the Person class. If we use any method of this class via p1 instance. Then this method is called instance method. But this is not vanilla JS object.

Instance method is having inside class.

**In mongoose, take an example**

Const todoSchema = new mongoose.Schema({ title: String});

Const Todo = new mongoose.model(‘Todo’, todoSchema);

Const todo = new Todo({title:’learn nodejs’});

**Note:** mongoose.Schema is a class.

Mongoose.model is also a class.

Create a schema => pass this schema to the mongoose.model class. Now we have a model instance.

In mongodb each table name is collection. Each record is called document in mongodb.

When we create a model. Actually this is a document structure to insert data into collection. This is mainly a class.

Using this model or document structure we can create actual document. We use new keyword to create actual document using model structure. After creating new actual document, now it’s a instance variable of the document structure model. We can access all the method or anything of this model class.

**Here is summary steps:**

* Create a schema from mongoose.Schema class
* Create a model ( which is an another class ) from mongoose.model class & pass schema to it
* Create a document using model class
* Call necessary model instance method using document

**Note:** we can not use arrow function inside schema instance, static chaining, query helper function. Because this will not identified by the arrow function. So better use function keywords.

**Instance Method:**

After creating new schema and modeling. We can create own methods inside the model class. Generally when we create a model class and create instance of this model. We can access all the method of this class easily. We can use our own methods by creating. When we create a model. We need to pass the schem to this function. Mainly model work based on schema. So custom methods need to pass to the schema.

**Custom instance method:**

todoSchema.methods = {

findActive: function () {

return mongoose.model(“Todo”).find({status: “active”});

}}

Now we can access this methods as an instance method.

**Example of custom method into ‘todoSchema’.**

const mongoose = require("mongoose");

const todoSchema = new mongoose.Schema({

  title: {

    type: String,

    required: true,

  },

  description: String,

  status: {

    type: String,

    enum: ["active", "inactive"],

  },

  date: {

    type: Date,

    default: Date.now(),

  },

});

// we have created a custom method

todoSchema.methods = {

  findActiveStatus: function () {

    return mongoose.model("Todo").find({ status: "active" });

  },

  findInActiveStatus: function () {

    return mongoose.model("Todo").find({ status: "inactive" });

  },

};

const Todo = mongoose.model("Todo", todoSchema);

// model return a class

module.exports = Todo;

**in controller:**

// find active status todo

const findActiveStatus = async (req, res, next) => {

  const todo = new Todo();

  console.log(await todo.findActiveStatus());

};

**In routes:**

// find active status of all tasks

router.get("/active", findActiveStatus);

**Static in class**

If we use static in our class with any method, property. Then we can access these method, property directly by using class name with dot operator. No need to create any instance or object of this class. When we declare any method as static. Then we can access this method by class name. Person.details().

Here Person is class and details was a static method. When we use static methods. These method goto the main class. So inside static method. We can access all the method using this keywords. Because they are in same class.

// custom static methods

// when we use statics. we can use this keyword to access the todoSchema instance methods

// mainly static method will be written in Todo class.so direactly access all the inbuilt methods

todoSchema.statics = {

  callStaticfunction: function () {

    return "this is a static method inside todo schema";

  },

  callInactiveStaticThiskeywordFunc: function () {

    return this.find({ status: "inactive" });

  },

};

**Query helper express JS**

**How chaining helper method works behind the scene?**

When we see method chaining, basically when we call a method of a class. It return again the class. That’s why we can use again other methods to make next chain. Second times it also return class and so on.

**how can we make method chaining ?**

we need to add query helper function into schema.

todoSchema.query = {

function….

}

**Note:** when we write regular express. /js/I this means behind the scene regular express run a constructor like this: new RegEx(‘/js/I’); when we want to use variable value for regex. We must use constructor method. Otherwise if we do like /variableValue/i… regular express take it as string.

// query helper

// new RegExp("string", "flag") is similar to /js/i

todoSchema.query = {

  byLanguage: function (language) {

    return this.find({ status: new RegExp(language, "i") });

  },

};

**Controller code:**

// find active status todo

const findActiveStatus = async (req, res, next) => {

  // instance method

  // const todo = new Todo();

  // console.log(await todo.findActiveStatus());

  // static chaining method

  // console.log(await Todo.callInactiveStaticThiskeywordFunc());

  // query method

  console.log(await Todo.find().byLanguage("active"));

};