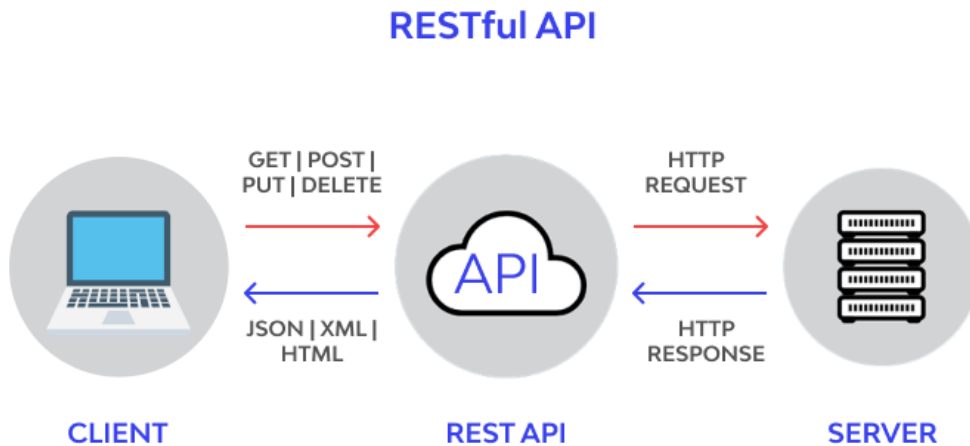


# Build your own RESTful API from scratch:



## Table of Contents :

1. Introduction
2. What is an API ?
  - a. API Endpoint
  - b. API Path
  - c. API Parameter
3. Using Weather Map API to fetch the temperature
4. What is the REST API ?
  - a. Rules to make an API → RESTful
  - b. Difference between REST and SOAP
5. Creating a Database using Robo3T
6. Methods of REST API
  - a. GET all articles
  - b. POST a new article
  - c. DELETE all articles
  - d. GET a specific article
  - e. PUT a specific article
  - f. PATCH a specific article
  - g. DELETE a specific article
7. References

## Introduction :

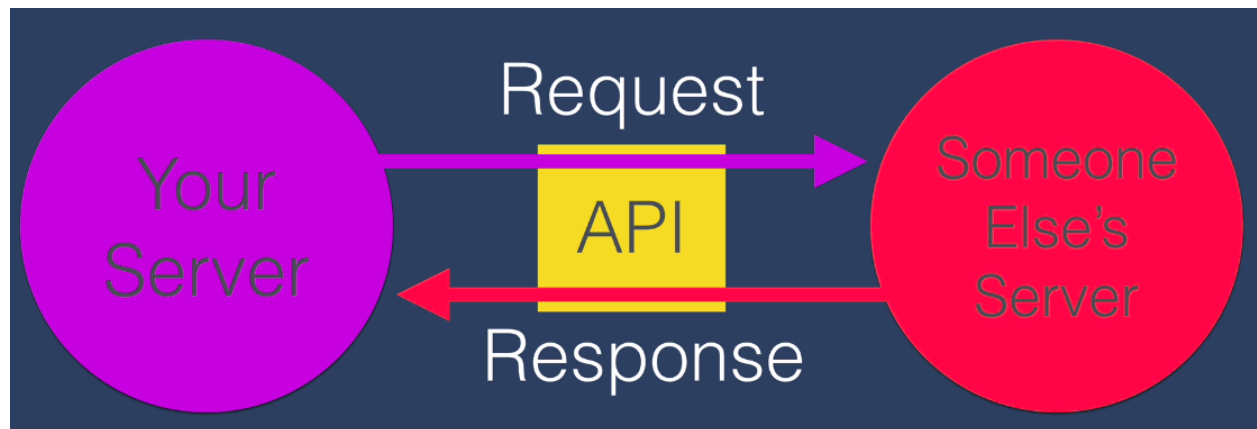
Since the invention of the internet, we have been using different applications and web pages to get data for various resources. However, where does this data come from?

**Well, it's the servers from where we get the data.** So in this article we will look into how a client communicates with the servers to extract the required information.

## What is an API ?

An **Application Programming Interface (API)** is a set of commands, functions, protocols, and objects that programmers can use to create software or interact with an external system.

APIs work using 'requests' and 'responses.'



## API Endpoint :

- When an API requests information from a web application or web server, it will receive a response. The place that APIs send requests and where the resource lives, is called an **endpoint**.
- An endpoint is one end of a communication channel.
- For APIs, an endpoint can include a URL of a server or service. Each endpoint is the location from which APIs can access the resources they need to carry out their function.
- Twitter API Endpoint example : <https://api.twitter.com/2/tweets/search/stream>

## API Path :

- An API URL Path is an address that allows us to access an API and its various features.
- There are 2 parts to any API URL:
  - a) **Base URL** : the base address for the specific API that we are using. Until we choose a specific Endpoint, though, the Base URL isn't going to do much.  
e.g <https://www.google.com> acts as the Base URL.
  - b) **Endpoint** : a specific "point of entry" in an API. We attach these to the end of our Base URL and get results depending on which Endpoint we choose.  
e.g <https://www.google.com/support> acts as the Endpoint.

## API Parameter :

- API Parameters are options that can be passed with the endpoint to influence the response.
- In GET requests, they're found in strings at the end of the API URL path.
- In POST requests, they're found in the POST body.
- In the world of APIs, these are like search filters. And depending on the parameters we set, we get a different response each time.

e.g [www.yoursite.com?myparam1=123&myparam2=abc&myparam2=xyz](http://www.yoursite.com?myparam1=123&myparam2=abc&myparam2=xyz)

Well, the stuff at the end, after the .com :

[?myparam1=123&myparam2=abc&myparam2=xyz](http://www.yoursite.com?myparam1=123&myparam2=abc&myparam2=xyz) are called parameters.

They are separated by &.

## Using Weather Map API to fetch the temperature :

```
const express = require("express");
const https=require("https");
const bodyParser = require("body-parser");

const app=express();
app.use(bodyParser.urlencoded({extended:true}));

app.get("/",function(request,response){
  response.sendFile(__dirname+"/index.html");
});

app.post("/",function(request,response){

  var place=request.body.place;

  const
```

```

url="https://api.openweathermap.org/data/2.5/weather?q="+place+"&appid=9c32
34d1801c308ec2d2aca43ef657e3&units=metric";
https.get(url,function(res)
{
    res.on("data",function(data)
    {
        const weatherData=JSON.parse(data);
        const temperature=weatherData.main.temp;
        const weatherDesc=weatherData.weather[0].description;
        const icon=weatherData.weather[0].icon;
        const
imageUrl="http://openweathermap.org/img/wn/"+icon+"@2x.png";

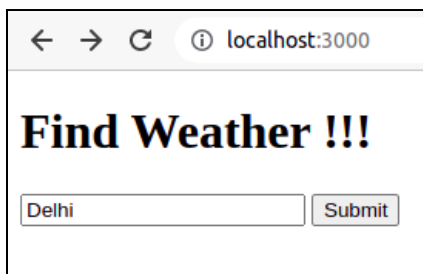
        response.write("<p>The weather is currently
"+weatherDesc+"</p>")
        response.write("<h1>Temperature of "+place "= "+temperature"
degree Celsius</h1>");
        response.write("");
        response.send();
    })
})

});

app.listen(3000,function(){
    console.log("Server started on port 3000");
});

```

#### Input the Place:

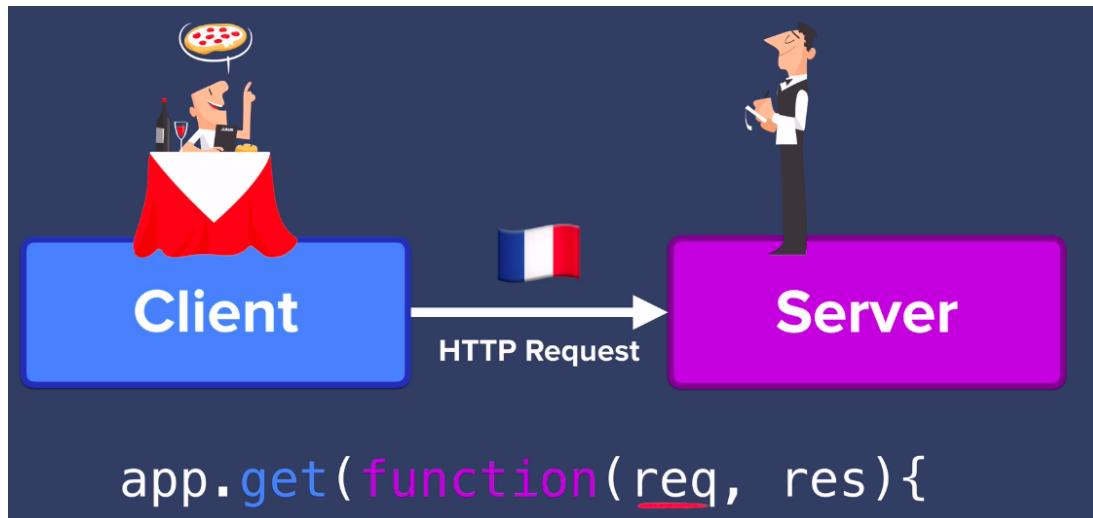


#### Output:



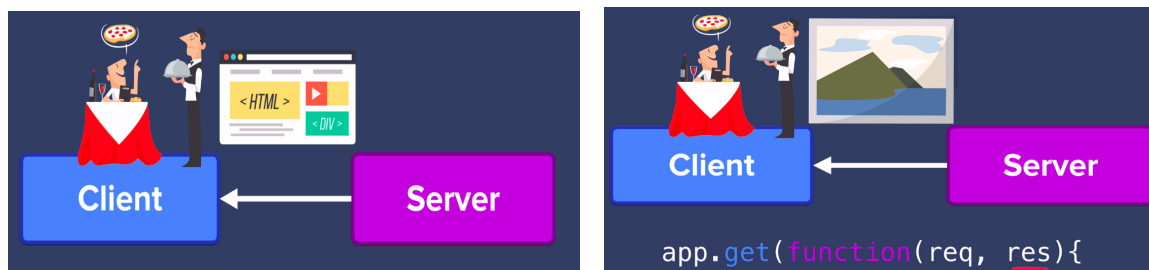
## What is the REST API ?

- The term REST stands for REpresentational State Transfer.
- It is an architectural style that defines a set of rules in order to create Web Services.
- In a client-server communication, REST suggests creating an object of the data requested by the client and send the values of the object in response to the user.

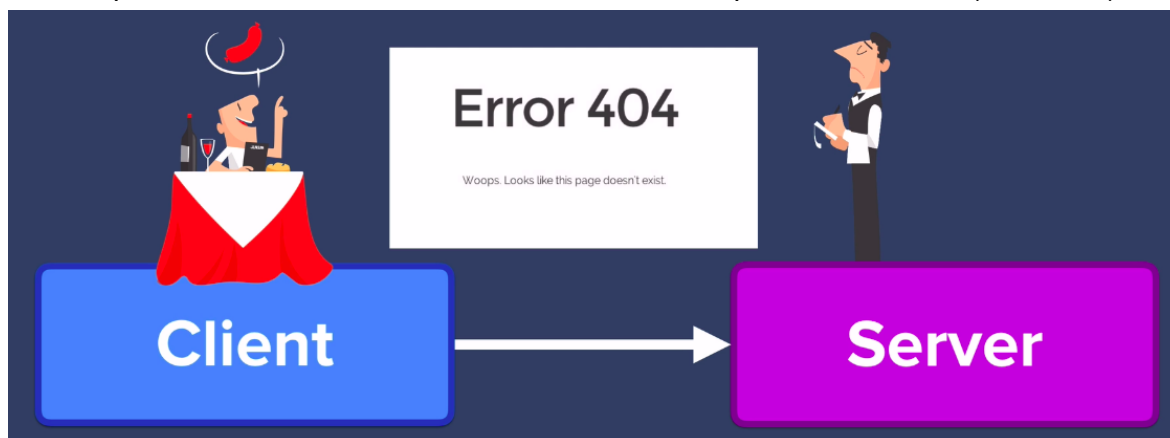


Client sends requests to the server using the `.get` function. It takes a function with request and response.

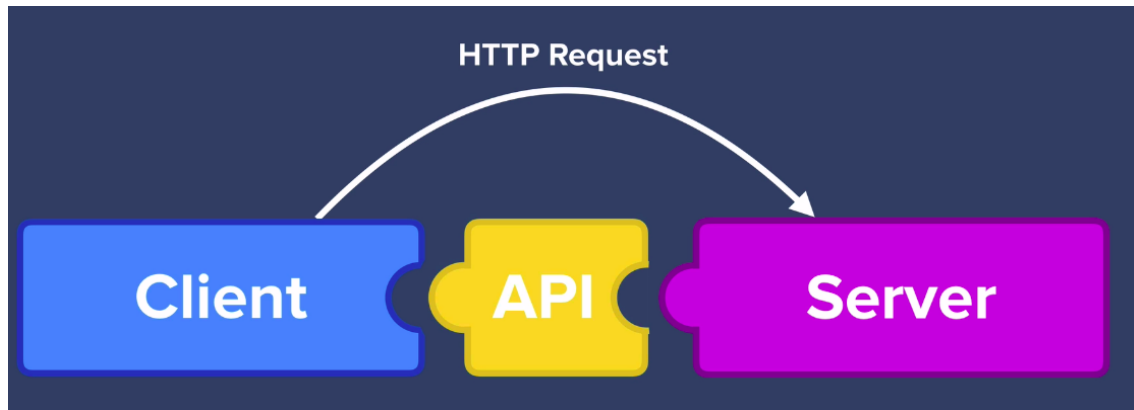
Server sends the response back : either HTML pages, or the images or data.



If the requested data is not found then the server will respond to Error 404 (Not found).



Now, the client interacts with the server using API's .

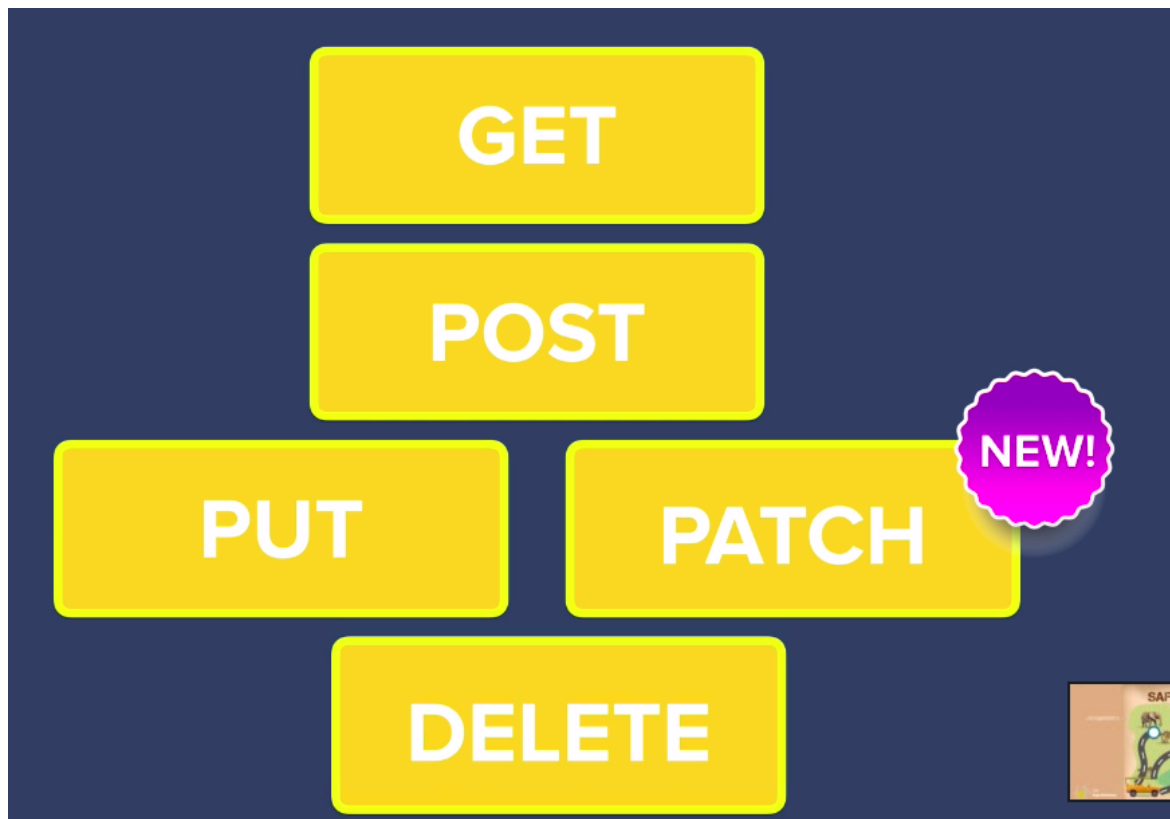


Rules to make an API → RESTful

Use HTTP Request Verbs ✓

Use Specific Pattern of Routes/Endpoint URLs ✓

To make an API restful, we need to use following **HTTP request verbs**:

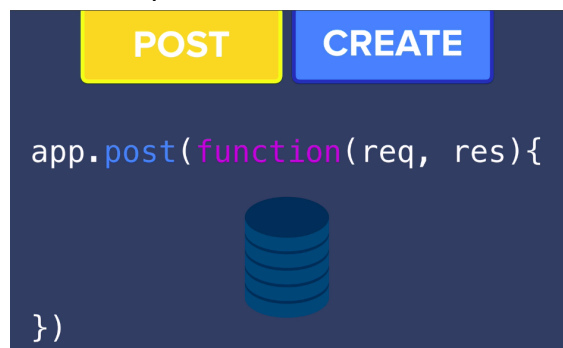


## CRUD Operations:

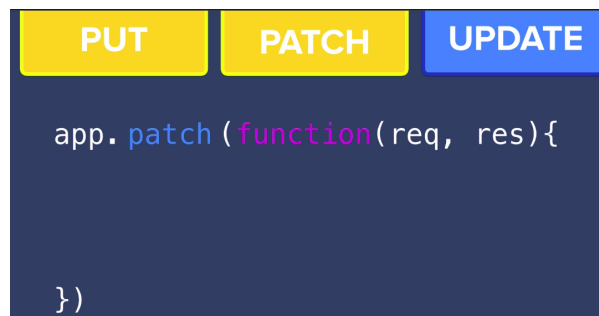
**GET** is equivalent to **READ**



**POST** is equivalent to **CREATE**



**PUT, PATCH** is equivalent to **UPDATE**



**DELETE** is equivalent to **DELETE**



**Summary:**



**RESTful Routing :**

# RESTful Routing

HTTP Verbs	/articles	/articles/jack-bauer
GET	Fetches <b>all</b> the articles	Fetches <b>the</b> article on jack-bauer
POST	Creates <b>one</b> new article	-
PUT	-	Updates <b>the</b> article on jack-bauer
PATCH	-	Updates <b>the</b> article on jack-bauer
DELETE	Deletes <b>all</b> the articles	Deletes <b>the</b> article on jack-bauer

## Difference between REST and SOAP:

REST	SOAP
<ul style="list-style-type: none"><li>• REST stands for REpresentational State Transfer.</li><li>• REST is an architectural style that doesn't follow any strict standard.</li><li>• REST is not restricted to XML and its the choice of implementer which Media-Type to use like XML, JSON, Plain-text.</li><li>• REST has SSL and HTTPS.</li><li>• REST can use SOAP.</li></ul>	<ul style="list-style-type: none"><li>• SOAP stands for Simple Object Access Protocol.</li><li>• Since SOAP is a protocol, it follows a strict standard to allow communication between the client and the server.</li><li>• SOAP uses only XML for exchanging information in its message format.</li><li>• SOAP has SSL( Secure Socket Layer) and WS-security.</li><li>• SOAP cannot use REST.</li></ul>

## Creating a database with Robo3T

```
cd /usr/local/bin/robo3t/bin/  
./robo3t
```

```
cd /usr/local/bin/robo3t/bin/  
./robo3t
```



## Server setup

### CodeWiki-API

- Create new Directory called Wiki-API
- Initialise NPM and install `body-parser`, `mongoose`, `ejs` and `express`
- Create a new file called `app.js`
- Inside `app.js` add `server code` (Write/Copy)
- Setup `MongoDB`:
  - DB name is `wikiDB`
  - Collection name is `articles`
  - Document has 2 fields: `title` and `content`

### Solution:

```
const express = require("express");
const bodyParser = require("body-parser");
const ejs = require("ejs");
const mongoose = require('mongoose');

const app = express();

app.set('view engine', 'ejs');

app.use(bodyParser.urlencoded({
  extended: true
}));
app.use(express.static("public"));

//TODO
mongoose.connect("mongodb://localhost:27017/wikiDB");
const articleSchema={
  title: String,
  content: String
}

const Article=mongoose.model("Article",articleSchema);

app.listen(3000, function() {
  console.log("Server started on port 3000");
});
```

## Database:



Key	Value	Type
(1) ObjectId("5c139771d79ac8eac11e7...")	{ 3 fields }	Object
_id	ObjectId("5c139771d79ac8eac11e754a")	ObjectId
title	API	String
content	API stands for Application Programming Interface. It is a set of subroutine definitions, com...	String
(2) ObjectId("5c1398aad79ac8eac11e7...")	{ 3 fields }	Object
_id	ObjectId("5c1398aad79ac8eac11e7561")	ObjectId
title	Bootstrap	String
content	This is a framework developed by Twitter that contains pre-made front-end templates for ...	String
(3) ObjectId("5c1398ecd79ac8eac11e75...")	{ 3 fields }	Object
_id	ObjectId("5c1398ecd79ac8eac11e7567")	ObjectId
title	DOM	String
content	The Document Object Model is like an API for interacting with our HTML	String

## GET All articles:



```
<ModelName>.find({conditions}, function(err, results){  
    //Use the found results docs.  
});
```

```
app.get("/articles",function(request,response){  
  Article.find(function(err,foundArticles){  
    if(!err)  
      response.send(foundArticles);  
    else  
      response.send(err);  
  });  
});
```

## Output:

Articles\_Rest\_API / GET all articles

GET http://localhost:3000/articles


Params Authorization Headers (9) Body ● Pre-request Script Tests Settings

Body Cookies Headers (6) Test Results Status: 200 OK

Pretty Raw Preview Visualize JSON

```
1 {
2   "_id": "5c139771d79ac8eac11e754a",
3   "title": "API",
4   "content": "API stands for Application Programming Interface. It is a set of subroutine definitions, communication protocols, and tools for building a set of clearly defined methods of communication among various components. A good API makes it easier to develop a computer program that uses pre-built blocks, which are then put together by the programmer."
5 },
6 {
7   "_id": "5c1398aad79ac8eac11e7561",
8   "title": "Bootstrap",
9   "content": "This is a framework developed by Twitter that contains pre-made front-end templates for web design"
10 },
11 {
12   "_id": "5c1398ecd79ac8eac11e7567",
13   "title": "DOM",
14   "content": "The Document Object Model is like an API for interacting with our HTML"
15 }
16 }
17 }
```

## POST a new article:



# CREATE

```
const <constantName> = new <ModelName> ({
  <fieldName> : <fieldData>,
  ...
});

<constantName>.save();
```

```
app.post("/articles",function(request,response){
  const newArticle=new Article({
    title: request.body.title,
    content: request.body.content
  });
```

```

newArticle.save(function(err){
  if(!err)
    response.send("Successfully saved Data");
  else
    response.send(err);
});
});

```


POST http://localhost:3000/articles

Params Authorization Headers (9) **Body** Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL

KEY	VALUE
<input checked="" type="checkbox"/> title	REST
<input checked="" type="checkbox"/> content	REST is short for REpresentational State Transfer. It's an archite
Key	Value

Body Cookies Headers (6) Test Results

Pretty Raw Preview Visualize HTML 

1 Successfully saved Data


GET http://localhost:3000/articles

Params Authorization Headers (9) **Body** Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL

KEY	VALUE
<input checked="" type="checkbox"/> title	REST
<input checked="" type="checkbox"/> content	REST is short for REpresentational State Transfer. It's an archite
Key	Value

Body Cookies Headers (6) Test Results

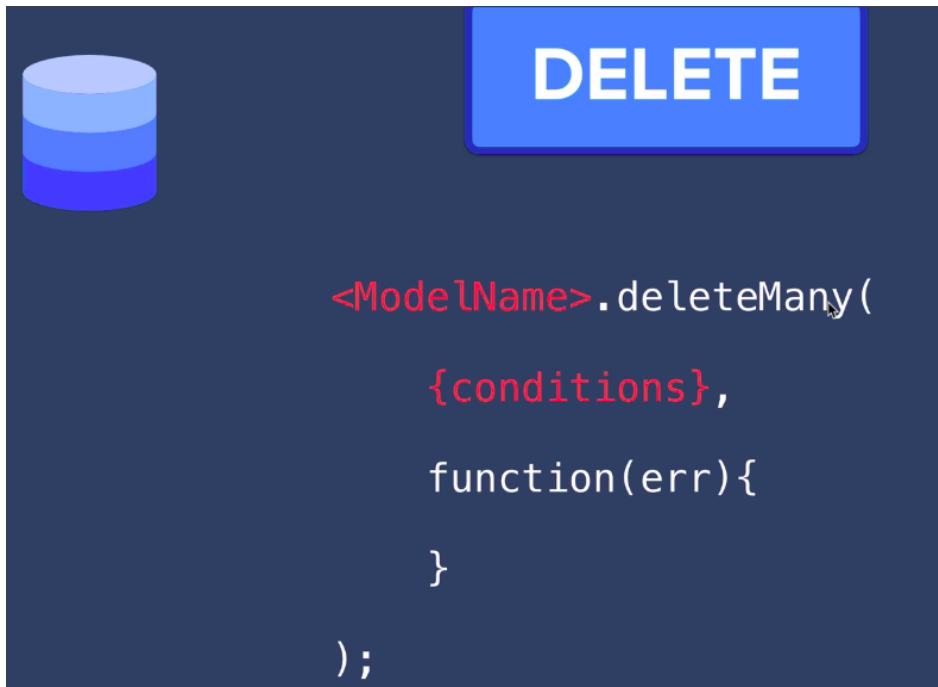
Pretty Raw Preview Visualize JSON 

```

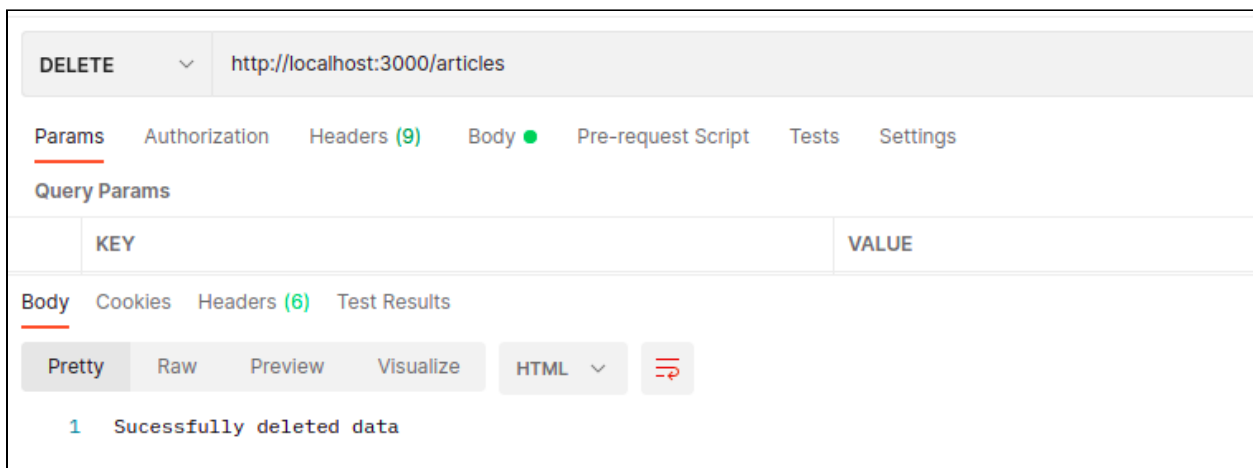
1 {
2   {
3     "_id": "5c139771d79ac8eac11e754a",
4     "title": "API",
5     "content": "API stands for Application Programming Interface. It is a set of subroutine definitions, communication p
        it is a set of clearly defined methods of communication among various components. A good API makes it easier to d
        blocks, which are then put together by the programmer."
6   },
7   {
8     "_id": "5c1398aad79ac8eac11e7561",
9     "title": "Bootstrap",
10    "content": "This is a framework developed by Twitter that contains pre-made front-end templates for web design"
11  },
12  {
13    "_id": "5c1398ecd79ac8eac11e7567",
14    "title": "DOM",
15    "content": "The Document Object Model is like an API for interacting with our HTML"
16  },
17  {
18    "_id": "61eae5ad2c364c638458e930",
19    "title": "REST",
20    "content": "REST is short for REpresentational State Transfer. IIIt's an architectural style for designing APIs.",
21    "_v": 0
22  }
23 }

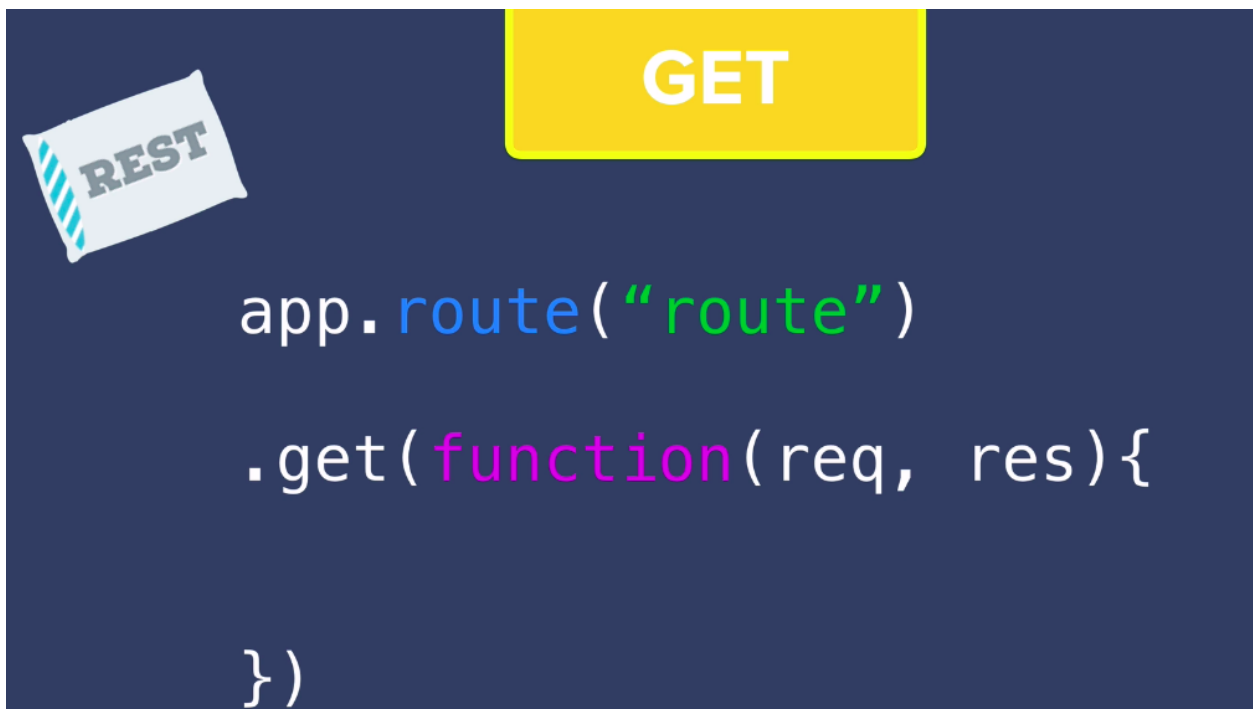
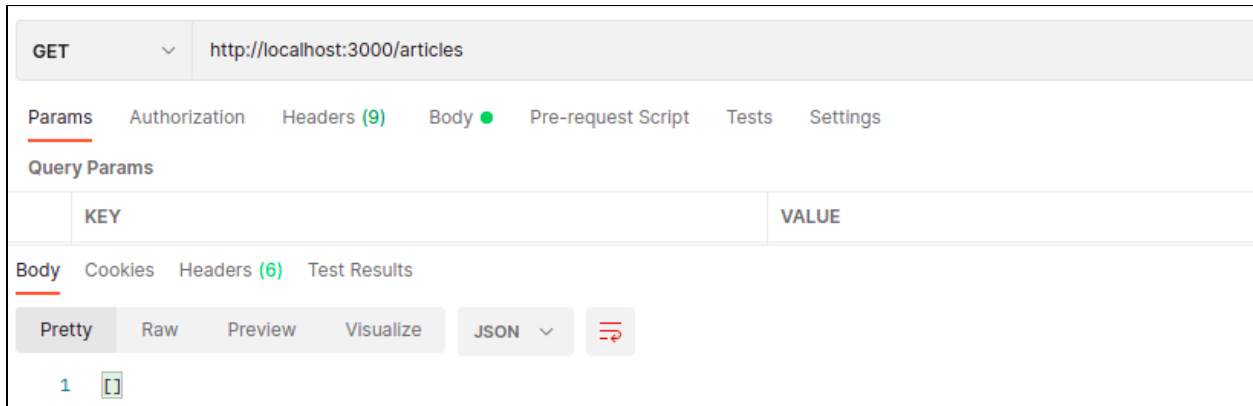
```

## DELETE All articles:



```
app.delete("/articles",function(request,response){  
    Article.deleteMany(function(err){  
        if(!err)  
            response.send("Successfully deleted data");  
        else  
            response.send(err);  
    });  
});
```





```

        title: request.body.title,
        content: request.body.content
    });
    newArticle.save(function(err){
        if(!err)
            response.send("Successfully saved Data");
        else
            response.send(err);
    });
})

.delete(function(request,response){
    Article.deleteMany(function(err){
        if(!err)
            response.send("Successfully deleted data");
        else
            response.send(err);
    });
});
});

```

## GET a specific article:



READ

```

<ModelName>.findOne(
    {conditions},
    function(err, result){
        //Use the found result.
    }
)

```

```

app.route("/articles/:articleTitle")

.get(function(request,response){
  Article.findOne({title:
request.params.articleTitle},function(err,foundArticle){
    if(foundArticle)
      response.send(foundArticle);
    else
      response.send("No matching article found");
  });
})
})

```

Articles\_Rest\_API / GET a specific article

GET ▼ http://localhost:3000/articles/DOM

Params Authorization Headers (7) Body Pre-request Script Tests Settings

Query Params

KEY	VALUE
-----	-------

Body Cookies Headers (6) Test Results

Pretty Raw Preview Visualize JSON ▼ ≡

```

1
2  "_id": "5c1398ecd79ac8eac11e7567",
3  "title": "DOM",
4  "content": "The Document Object Model is like an API for interacting with our HTML"
5

```

GET ▼ http://localhost:3000/articles/Jack%20Bauer

Params Authorization Headers (7) Body Pre-request Script Tests Settings

Query Params

KEY	VALUE
-----	-------

Body Cookies Headers (6) Test Results

Pretty Raw Preview Visualize JSON ▼ ≡

```

1
2  "_id": "5c18f35cde40ab6cc551cd60",
3  "title": "Jack Bauer",
4  "content": "Jack Bauer once stepped into quicksand. The quicksand couldn't escape and nearly drowned.",
5  "__v": 0
6

```



PUT a specific article:



UPDATE

```
<modelName>.update(  
    {conditions},  
    {updates},  
    {overwrite: true}  
    function(err, results){}  
);
```

```
.put(function(request,response){  
    Article.findOneAndUpdate(  
        {title: request.params.articleTitle},  
        {title: request.body.title,content: request.body.content},  
        {overwrite:true},  
        function(err){  
            if(!err)  
                response.send("Successfully updated data");  
            else  
                response.send(err);  
        });  
    });  
})
```

Articles\_Rest\_API / PUT a specific article

PUT ▼ http://localhost:3000/articles/DOM

Params Authorization Headers (9) **Body** ● Pre-request Script Tests Settings

☐ none ☐ form-data ☒ x-www-form-urlencoded ☐ raw ☐ binary ☐ GraphQL

	KEY	VALUE
<input checked="" type="checkbox"/>	title	DOM1
<input checked="" type="checkbox"/>	content	Testing DOM
	Key	Value

Body Cookies Headers (6) Test Results

Pretty Raw Preview Visualize HTML ▼

1 Successfully updated data

## PATCH a specific article:



```
<ModelName>.update(  
  {conditions},  
  {$set: updates},  
  function(err, results){}  
);
```

```

.patch(function(request,response){
  Article.findOneAndUpdate(
    {title: request.params.articleTitle},
    {$set: request.body},
    function(err){
      if(!err)
        response.send("Successfully updated data");
      else
        response.send(err);
    });
})

```

Articles\_Rest\_API / PATCH a specific article

PATCH ⌵ http://localhost:3000/articles/DOM1

Params Authorization Headers (9) **Body** ● Pre-request Script Tests Settings

● none ● form-data ● x-www-form-urlencoded ● raw ● binary ● GraphQL

	KEY	VALUE
<input checked="" type="checkbox"/>	title	DOM
	Key	Value

Body Cookies Headers (6) Test Results

Pretty Raw Preview Visualize HTML ⌵ ≡

1 Successfully updated data

## DELETE a specific article:



# DELETE

```
<ModelName>.deleteOne(  
    {conditions},  
    function(err){  
  
    }  
);
```

```
.delete(function(request,response){  
    Article.deleteOne(  
        {title: request.params.articleTitle},  
        function(err,foundArticle){  
            if(!err)  
                response.send("Successfully deleted data");  
            else  
                response.send(err);  
        });  
    });  
});
```

Articles\_Rest\_API / DELETE a specific article


DELETE ▼ http://localhost:3000/articles/DOM

Params Authorization Headers (7) Body Pre-request Script Tests Settings

Query Params

	KEY	VALUE
--	-----	-------

Body Cookies Headers (6) Test Results

Pretty Raw Preview Visualize HTML ▼ 

1 Successfully deleted data

## References:

- [What is an API URL Path? API URL Meaning \[Explained\] - Apipheny](#)
- <https://www.udemy.com/course/the-complete-web-development-bootcamp/learn/lecture/18125165#overview>
- [What Is REST API? | RESTful API Tutorial For Beginners | Edureka](#)
- <https://apipheny.io/what-are-api-parameters/>
- [Difference between REST API and SOAP API - GeeksforGeeks](#)

---

---