

---

# JAVASCRIPT

## CSS-FRONT-END BACK-END

### (API WITH PGSQL & MONGODB)

---

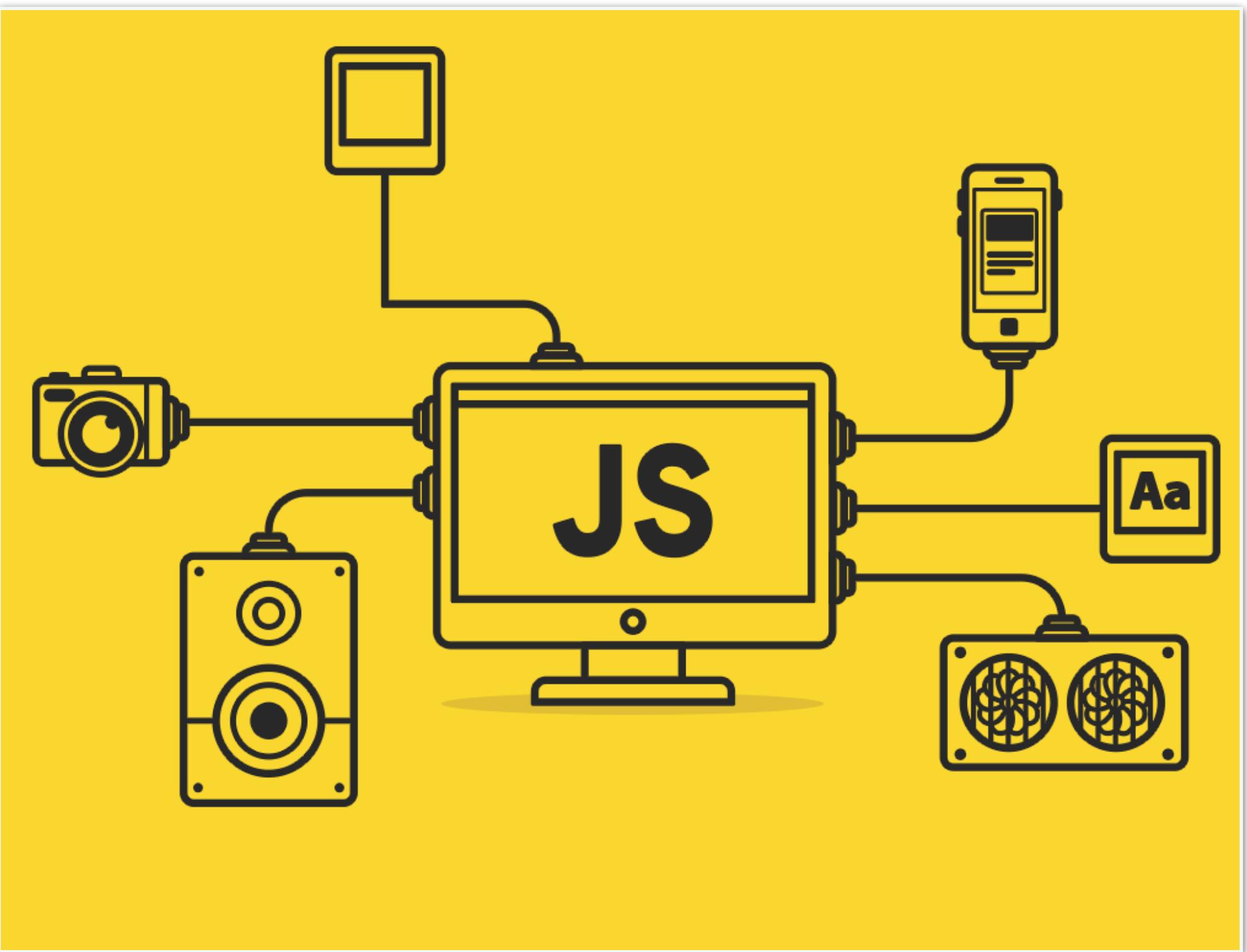
Anirach Mingkhwani

[Anirach.m@fitm.kmutnb.ac.th](mailto:Anirach.m@fitm.kmutnb.ac.th)



# OUTLINE

- CSS-Front-End
- PostgreSQL
- MongoDB
- Postman Test
- Front-End connect with Back-End



---

# CSS FRONT-END

---



# Books

Title	Author	Actions
Runway	planB	<a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a>
Ghost	John.D	<a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a>
Dr. Who	Kong	<a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a>
Firework	Dobb	<a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a>
<a href="#">Create New Book</a>		

The screenshot shows a web application interface for managing books. At the top, there is a header bar with icons for back, forward, refresh, and home, followed by the URL "Not Secure | node40732-noderest.proen.app.ruk-com.cloud:11503". To the right of the URL are several small icons: a download arrow, a star, a red square labeled "SC", a green circle labeled "G", a blue square labeled "TEX", and a gear icon.

The main content area has a title "Books" centered at the top. Below it is a table with the following data:

TITLE	AUTHOR	ACTIONS
The Simpsons	Bart Simson	<a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a>
7000 Stars	Anirach	<a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a>
Water Mark	Jimmy	<a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a>
Dr. Who	Robot	<a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a>
Runway	planB	<a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a>

At the bottom center of the table, there is a link labeled "Create New Book".

# Update Book

**Title:**

The Simpsons

**Author:**

Bart Simson

**Update**

[View](#)   [Delete](#)   [Back to List](#)

```
app/
-- views/
---- home.ejs
-- public/
---- css/
----- style.css
---- js/
----- script.js
-- node_modules/
-- package.json
-- server.js
```

In this structure:

- The `views` directory contains EJS templates for each page of the application. In this example, there is only one template, `home.ejs`, but there could be additional templates for other pages of the application.
- The `public` directory contains static assets that will be served by the web server. In this example, there are subdirectories for `css` and `js`, which contain the CSS and JavaScript files for the application.
- The `node\_modules` directory contains the installed Node.js modules that are used by the application.
- The `package.json` file contains information about the application and its dependencies.
- The `server.js` file is the main file of the application, which sets up the server and defines the routes and middleware.

The screenshot shows the Visual Studio Code interface with the following details:

- EXPLORER** pane on the left:
  - OPEN EDITORS**: package.json
  - FRONT-END-NODE** folder:
    - node\_modules
    - public
      - css (style.css)
      - images
      - views
        - book.ejs
        - books.ejs
        - create.ejs
        - update.ejs
    - routes
      - .gitignore
      - index.js
      - package-lock.json
  - package.json** is selected in the Explorer.
  - EDITOR** pane on the right:
    - package.json** (You, 2 days ago | 1 author (You))
    - Content of package.json:

```
1 {   "name": "front-end-node",  
2   "version": "1.0.0",  
3   "description": "Front-End for the API Project Node.js",  
4   "main": "index.js",  
5     ▶ Debug  
6   "scripts": {  
7     "start": "node index.js"  
8   },  
9   "repository": {  
10    "type": "git",  
11    "url": "git+https://github.com/Anirach/Front-End-Node.git"  
12  },  
13  "keywords": [],  
14  "author": "",  
15  "license": "ISC",  
16  "bugs": {  
17    "url": "https://github.com/Anirach/Front-End-Node/issues"  
18  },  
19  "homepage": "https://github.com/Anirach/Front-End-Node#readme",  
20  "dependencies": {  
21    "axios": "^1.3.4",  
22    "body-parser": "^1.20.2",  
23    "ejs": "^3.1.8",  
24    "express": "^4.18.2",  
25    "http": "^0.0.1-security"  
26  }  
27}  
28
```

The screenshot shows the VS Code interface with the Explorer and Editor panes.

**EXPLORER** pane:

- OPEN EDITORS: style.css (public/css)
- FRONT-END-NODE:
  - node\_modules
  - public
    - css (style.css)
    - images
    - views
      - book.ejs
      - books.ejs
      - create.ejs
      - update.ejs
  - routes
    - .gitignore
    - index.js
    - package-lock.json
    - package.json
  - README.md

**Editor pane:** style.css

```
/* Set font family and size for the whole page */
body {
    font-family: 'Open Sans', sans-serif;
    font-size: 16px;
}

/* Center the main heading */
h1 {
    text-align: center;
    margin-top: 30px;
    margin-bottom: 50px;
    font-size: 36px;
    color: #2C3E50;
}

/* Style the table */
table {
    border-collapse: collapse;
    width: 100%;
}

th, td {
    text-align: left;
    padding: 16px;
}
```

You, yesterday | 1 author (You)

The screenshot shows the Visual Studio Code interface. The left sidebar contains icons for file operations like copy, search, and refresh. The Explorer pane displays the project structure:

- OPEN EDITORS: style.css (public/css)
- FRONT-END-NODE:
  - node\_modules
  - public
    - css
      - style.css
    - images
    - views
      - book.ejs
      - books.ejs
      - create.ejs
      - update.ejs
  - routes
    - .gitignore
    - index.js
    - package-lock.json
    - package.json
    - README.md

The Editor pane shows the 'style.css' file content:

```
public > css > style.css > input[type="submit"]  
25 }  
26  
27 th {  
28     background-color: #2C3E50;  
29     color: white;  
30     font-weight: 600;  
31     text-transform: uppercase;  
32 }  
33  
34 tr:nth-child(even) {  
35     background-color: #f5f5f5;  
36 }  
37  
38 /* Style the form */  
39 form {  
40     width: 50%;  
41     margin: auto;  
42     border: 2px solid #2C3E50;  
43     padding: 20px;  
44     border-radius: 5px;  
45 }  
46  
47 label {  
48     display: block;  
49     font-weight: bold;  
50     margin-bottom: 10px;  
51 }  
52 }
```

The screenshot shows a code editor interface with a dark theme. On the left is a sidebar with various icons for file operations like copy, search, and navigation. The main area has two tabs: "style.css" which is currently active, and another tab for "style.css" under "public/css". The code editor displays the "style.css" file located at "public > css > style.css". The file contains CSS rules for input fields and links, with line numbers from 52 to 93. A status bar at the bottom indicates "You, yesterday • add css to front-end".

```
public > css > style.css > ...
```

```
52
53 input[type="text"] {
54   width: 100%;
55   padding: 8px;
56   box-sizing: border-box;
57   border-radius: 5px;
58   border: none;
59   background-color: #f2f2f2;
60 }
61
62 input[type="submit"] {
63   background-color: #2C3E50;
64   color: white;
65   border: none;
66   padding: 10px 16px;
67   font-size: 16px;
68   cursor: pointer;
69   border-radius: 5px;
70   margin-top: 10px;
71 }
72 You, yesterday • add css to front-end
73 /* Style the links */
74 a {
75   display: inline-block;
76   margin: 10px;
77   color: #2C3E50;
78   text-decoration: none;
79   font-size: 18px;
80   font-weight: 600;
81   border-bottom: 2px solid transparent;
82   transition: border-bottom-color 0.2s;
83 }
```

The screenshot shows the VS Code interface with the following details:

- EXPLORER View:** Shows the project structure:
  - FRONT-END-NODE folder
    - node\_modules
    - public
      - css (selected)
      - images
      - views
        - book.ejs
        - books.ejs
        - create.ejs
        - update.ejs
    - routes
  - .gitignore
  - index.js
  - package-lock.json
  - package.json
  - README.md
- OPEN EDITORS View:** Shows the file `style.css` is open in the editor.
- Editor View:** Displays the content of `style.css` with line numbers and syntax highlighting.

```
public > css > style.css > ...  
83 }  
84  
85 a:hover {  
86 border-bottom-color: #2C3E50;  
87 }  
88  
89 .link-center {  
90 text-align: center;  
91 margin-top: 20px;  
92 }  
93  
94 .link-center a {  
95 display: inline-block;  
96 margin: 10px;  
97 color: #2C3E50;  
98 text-decoration: none;  
99 font-size: 18px;  
100 font-weight: 600;  
101 border-bottom: 2px solid transparent;  
102 transition: border-bottom-color 0.2s;  
103 }  
104  
105 .link-center a:hover {  
106 border-bottom-color: #2C3E50;  
107 }  
108 }
```

The screenshot shows the Visual Studio Code interface. On the left is the Explorer sidebar with a dark theme. It lists the project structure under 'OPEN EDITORS' and 'FRONT-END-NODE'. The 'style.css' file in the 'css' folder is selected in both the Explorer and the main editor area. The main editor shows the CSS code for styling a table.

```
public > css > style.css > ...
```

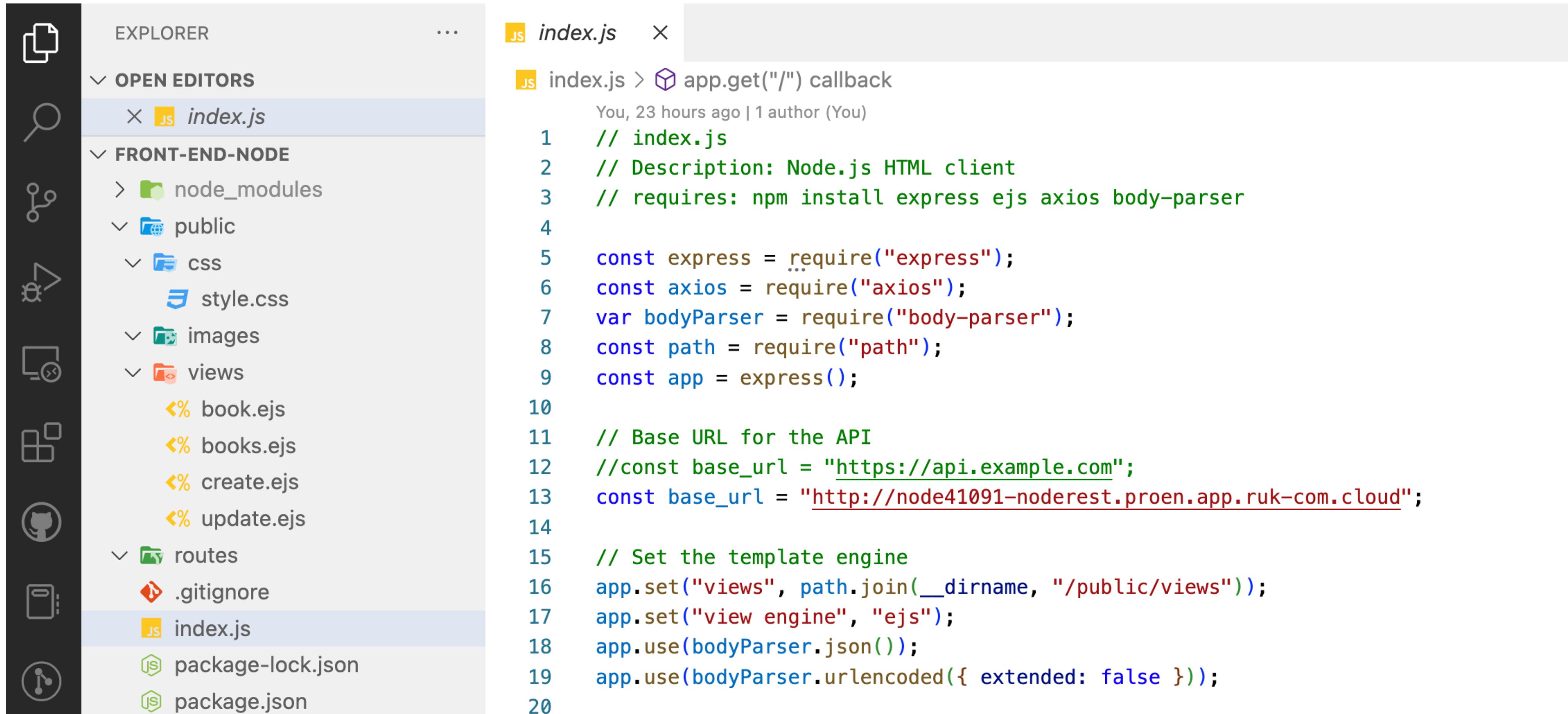
```
107 }  
108  
109 /* Style the table */  
110 .book-table {  
111 border-collapse: collapse;  
112 width: 75%;  
113 margin: 0 auto; /* center the table */  
114 margin-bottom: 20px;  
115 border-bottom: 2px solid #2C3E50;  
116 }  
117  
118 th, td {  
119 text-align: left;  
120 padding: 16px;  
121 }  
122
```

The screenshot shows the Visual Studio Code interface. On the left is the Explorer sidebar with a dark theme, displaying the project structure:

- FRONT-END-NODE
  - node\_modules
  - public
    - css (style.css)
    - images
    - views
      - book.ejs (selected)
      - books.ejs
      - create.ejs
      - update.ejs
  - routes
    - .gitignore
    - index.js
    - package-lock.json
    - package.json
    - README.md

The right side shows the code editor for the selected file, book.ejs:

```
public > views > book.ejs > html > head
You, yesterday | 1 author (You)
1 <!-- book.ejs -->
2 <!DOCTYPE html>
3 <html>
4   <head>
5     <title>Book</title>
6     <link rel="stylesheet" href="../css/style.css" /> You,
7   </head>
8   <body>
9     <h1>View Book</h1>
10    <table class="book-table">
11      <tr>
12        <th>Title</th>
13        <th>Author</th>
14      </tr>
15      <tr>
16        <td><%= book.title %></td>
17        <td><%= book.author %></td>
18      </tr>
19    </table>
20    <div class="link-center">
21      <a href="/update/<%= book.id %>">Edit</a>
22      <a href="/delete/<%= book.id %>">Delete</a>
23      <a href="/">Back to List</a>
24    </div>
25  </body>
26</html>
27
```



The screenshot shows the Visual Studio Code interface. The left sidebar (Explorer) displays the project structure:

- OPEN EDITORS: index.js
- FRONT-END-NODE:
  - node\_modules
  - public
    - css (style.css)
    - images
    - views
      - book.ejs
      - books.ejs
      - create.ejs
      - update.ejs
  - routes
    - .gitignore
    - index.js- package-lock.json
- package.json

The right pane (Editor) shows the content of index.js:

```
JS index.js  X  
JS index.js > app.get("/") callback  
You, 23 hours ago | 1 author (You)  
1 // index.js  
2 // Description: Node.js HTML client  
3 // requires: npm install express ejs axios body-parser  
4  
5 const express = require("express");  
6 const axios = require("axios");  
7 var bodyParser = require("body-parser");  
8 const path = require("path");  
9 const app = express();  
10  
11 // Base URL for the API  
12 //const base_url = "https://api.example.com";  
13 const base_url = "http://node41091-noderest.proen.app.ruk-com.cloud";  
14  
15 // Set the template engine  
16 app.set("views", path.join(__dirname, "/public/views"));  
17 app.set("view engine", "ejs");  
18 app.use(bodyParser.json());  
19 app.use(bodyParser.urlencoded({ extended: false }));  
20
```

---

# API WITH PGSQL

---

src >  PGSQLCRUD.js >  Book >  author

```
1  const express = require('express');
2  const Sequelize = require('sequelize');
3  const app = express();
4  // parse incoming requests
5  app.use(express.json());
6
7  // set db url
8  const dbUrl = 'postgres://webadmin:MAIgsf81141@node40729-noderest.proen.app.ruk-com.cloud:11478/Books'
9
10 // create a connection to the database
11 const sequelize = new Sequelize(dbUrl);
```

**pg****DT**

8.10.0 • Public • Published 5 days ago

[Readme](#)[Code](#) Beta[7 Dependencies](#)[8,224 Dependents](#)[215 Versions](#)

# node-postgres

[build](#) **passing** [npm](#) v8.10.0 [downloads](#) 18M/month

Non-blocking PostgreSQL client for Node.js. Pure JavaScript and optional native libpq bindings.

## Install

```
$ npm install pg
```

## ★ Documentation ★

## Features

- Pure JavaScript client and native libpq bindings share the same API

## Install

```
> npm i pg
```

## Repository

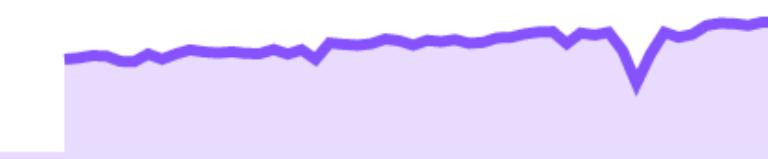
[github.com/brianc/node-postgres](#)

## Homepage

[github.com/brianc/node-postgres](#)

## ↓ Weekly Downloads

4,274,097



Version

8.10.0

License

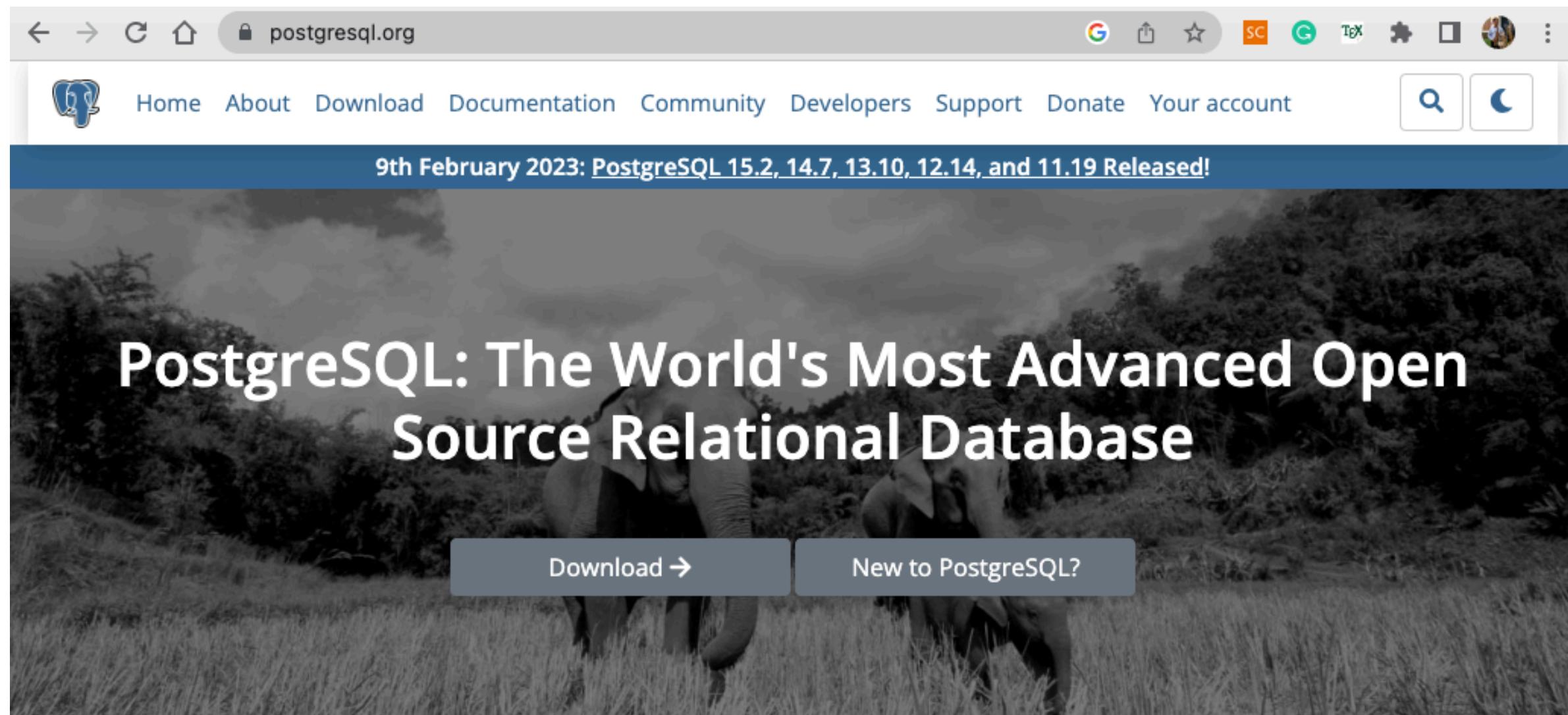
MIT

<https://www.npmjs.com/package/pg>

---

# PGSQL-DB

---



## New to PostgreSQL?

PostgreSQL is a powerful, open source object-relational database system with over 35 years of active development that has earned it a strong reputation for reliability, feature robustness, and performance.

There is a wealth of information to be found describing how to [install](#) and [use](#) PostgreSQL through the [official documentation](#). The [open source community](#) provides many helpful places to become familiar with PostgreSQL, discover how it works, and find career opportunities. Learn more on how to [engage with the community](#).

[Learn More](#)



## Latest Releases

**2023-02-09 - PostgreSQL 15.2, 14.7, 13.10, 12.14, and 11.19 Released!**

The PostgreSQL Global Development Group has [released an update](#) to all supported versions of PostgreSQL, including 15.2, 14.7, 13.10, 12.14, and 11.19. This release closes one security vulnerability and fixes over 60 bugs reported over the last several months.

For the full list of changes, please review the [release notes](#). You can get the updates on the [download](#) page.

**15.2 · 2023-02-09 · Notes**

**14.7 · 2023-02-09 · Notes**

**13.10 · 2023-02-09 · Notes**

**12.14 · 2023-02-09 · Notes**

<https://www.postgresql.org/>



## PostgreSQL Tutorial

Welcome to the PostgreSQLTutorial.com website! This **PostgreSQL tutorial** helps you understand PostgreSQL quickly. You'll master PostgreSQL very fast through many practical examples and apply the knowledge in developing applications using PostgreSQL.

If you are...

- Looking for learning PostgreSQL fast.
- Developing applications using PostgreSQL as the back-end database management system.
- Migrating from other database management systems such as MySQL, Oracle, and Microsoft SQL Server to PostgreSQL.

You'll find all you need to know to get started with PostgreSQL quickly and effectively here on this website.

PostgreSQL tutorial demonstrates many unique features of PostgreSQL that make it the most advanced open-source database management system.



### Getting Started with PostgreSQL

This section helps you get started with PostgreSQL by showing you how to install PostgreSQL on Windows, Linux, and macOS. You also learn how to connect to PostgreSQL using the psql tool as well as how to load a sample database into the PostgreSQL for practicing.

### Basic PostgreSQL Tutorial



Search ...

#### POSTGRESQL QUICK START

[What is PostgreSQL?](#)

[PostgreSQL Sample Database](#)

[Install PostgreSQL on Windows](#)

[Connect to Database](#)

[Load Sample Database](#)

[Install PostgreSQL on macOS](#)

[Install PostgreSQL on Linux](#)

#### ADVERTISEMENTS



Try the PostgreSQL editor with the world's highest user satisfaction.\*



\* According to G2.com

[Download for free](#)

#### POSTGRESQL FUNDAMENTALS

<https://www.postgresqltutorial.com/>



A new PostgreSQL server was successfully added to your NodeREST environment in Ruk-Com.

Please use the following data to access and manage the added node:

**phpPgAdmin:** <https://node40729-noderest.proen.app.ruk-com.cloud>

**Login:** webadmin

**Password:** MAIgsf81141

The screenshot shows a cloud-based management interface with a dark-themed sidebar and main content area.

**Top Bar:**

- SQL Databases:** PostgreSQL 14.6 (Node ID: 40729, 14.6)
- NoSQL Databases:** MongoDB 4.0.2 (Node ID: 40731, 4.0.2)
- ThaiClassify:** thaiclassify.proen.app.ruk-com.cloud (Stopped)

**Left Sidebar (Settings):**

- Custom Domains
- Custom SSL
- SSH Access
- Endpoints** (selected)
- Firewall
- Load Alerts
- Auto Horizontal Scaling
- Collaboration
- Change Owner
- Migration
- Export
- Info

**Main Content Area:**

Here you can add and manage endpoints, which can be used by other resources for communication.

**Edit Endpoint**

Node:	Node ID: 40729
Name:	DBAccess
Private Port:	5432
Protocol:	TCP
Public Port:	11478
Access URL:	node40729-noderest.proen.app.ruk-com

**Bottom Status Bar:**

- SQL Databases: PostgreSQL 14.6 (Node ID: 40729)
- DBAccess: 5432 TCP 11478 node40729-noderest.proen.app.ruk-com.cloud:11478

The screenshot shows a browser window with the URL `node40729-noderest.proen.app.ruk-com.cloud` in the address bar. The page title is "phpPgAdmin 7.13.0". On the left, there is a sidebar with icons for "Servers" and "PostgreSQL". The main content area displays the "Introduction" page of phpPgAdmin, featuring the title "phpPgAdmin 7.13.0 (PHP 7.4.33)" in large bold letters. Below it are dropdown menus for "Language" (set to English) and "Theme" (set to Default). A welcome message "Welcome to phpPgAdmin." is followed by a list of links:

- [phpPgAdmin Homepage](#)
- [PostgreSQL Homepage](#)
- [Report a Bug](#)
- [View online FAQ](#)
- [Selenium tests](#)

phpPgAdmin

PostgreSQL 14.6 running on localhost:5432 -- You are logged in as user "webadmin"

SQL | History | Find | Logout

phpPgAdmin: PostgreSQL ?:

Databases? Roles? Tablespaces? Export

Database	Owner	Encoding	Collation	Character Type	Tablespace	Size	Actions	Comment
postgres	postgres	UTF8	en_US.UTF-8	en_US.UTF-8	pg_default	8465 kB	<a href="#">Drop</a> <a href="#">Privileges</a> <a href="#">Alter</a>	default administrative connection database

Actions on multiple lines  
Select all / Unselect all --->

Create database

Servers

PostgreSQL

postgres

Schemas

public

Tables

Views

Sequences

Functions

Full Text Search

Domains

phpPgAdmin

PostgreSQL 14.6 running on localhost:5432 -- You are logged in as user "webadmin"

phpPgAdmin: PostgreSQL ?

Create database?

Name	Books
Template	template1 ▾
Encoding	UTF8 ▾
Collation	
Character Type	
Comment	For Node-REST Project

Create Cancel

The screenshot shows the phpPgAdmin interface for managing a PostgreSQL database. On the left, there's a tree view of the database structure under the 'PostgreSQL' server and 'postgres' database, including 'Schemas' like 'public', 'Tables', 'Views', 'Sequences', 'Functions', 'Full Text Search', and 'Domains'. The main area is titled 'Create database?' and contains a form with the following fields:

- Name:** Books
- Template:** template1
- Encoding:** UTF8
- Collation:** (empty)
- Character Type:** (empty)
- Comment:** For Node-REST Project

At the bottom of the dialog are 'Create' and 'Cancel' buttons.

phpPgAdmin

PostgreSQL 14.6 running on localhost:5432 -- You are logged in as user "webadmin"

SQL | History | Find | Logout

phpPgAdmin: PostgreSQL?

Databases? Roles? Tablespaces? Export

Database created.

Database	Owner	Encoding	Collation	Character Type	Tablespace	Size	Actions			Comment
Books	webadmin	UTF8	en_US.UTF-8	en_US.UTF-8	pg_default	8465 kB	<a href="#">Drop</a>	<a href="#">Privileges</a>	<a href="#">Alter</a>	For Node-REST Project
postgres	postgres	UTF8	en_US.UTF-8	en_US.UTF-8	pg_default	8465 kB	<a href="#">Drop</a>	<a href="#">Privileges</a>	<a href="#">Alter</a>	default administrative connection database

Actions on multiple lines  
Select all / Unselect all --->

Create database

The screenshot shows the phpPgAdmin interface for PostgreSQL 14.6. The left sidebar displays the server structure under 'Servers' for 'PostgreSQL'. It shows two databases: 'Books' and 'postgres'. Each database has a 'Schemas' folder containing a 'public' schema, which in turn contains 'Tables', 'Views', 'Sequences', 'Functions', 'Full Text Search', and 'Domains'. The main panel shows a message 'Database created.' above a table of databases. The table has columns: Database, Owner, Encoding, Collation, Character Type, Tablespace, Size, Actions (Drop, Privileges, Alter), and Comment. It lists two databases: 'Books' (owner webadmin, comment 'For Node-REST Project') and 'postgres' (owner postgres, comment 'default administrative connection database'). Below the table is an 'Actions on multiple lines' section with 'Select all / Unselect all' checkboxes and a 'Execute' button. At the bottom is a 'Create database' button.

**phpPgAdmin**

PostgreSQL 14.6 running on localhost:5432 – You are logged in as user "webadmin"

SQL | History | Find | Logout

phpPgAdmin: PostgreSQL: Books: public: books:

Columns Browse Select? Insert? Indexes? Constraints? Triggers? Rules? Admin Info Privileges? Import Export

Column	Type	Not Null	Default	Constraints	Actions				Comment
id	integer	NOT NULL	nextval('books_id_seq'::regclass)	key	Browse	Alter	Privileges	Drop	
title	character varying(255)	NOT NULL			Browse	Alter	Privileges	Drop	
author	character varying(255)	NOT NULL			Browse	Alter	Privileges	Drop	
createdAt	timestamp with time zone	NOT NULL			Browse	Alter	Privileges	Drop	
updatedAt	timestamp with time zone	NOT NULL			Browse	Alter	Privileges	Drop	

Browse | Select | Insert | Empty | Drop | Add column | Alter

The screenshot shows the Postman application interface. At the top, there is a header bar with the method **POST**, the URL **localhost:3000/books**, and a **Send** button. Below the header, there are tabs for **Params**, **Authorization**, **Headers (9)**, **Body** (which is selected and highlighted in green), **Pre-request Script**, **Tests**, and **Settings**. To the right of these tabs are **Cookies**, **</>**, and a gear icon. Under the **Body** tab, there are options for **none**, **form-data**, **x-www-form-urlencoded**, **raw** (which is selected and highlighted in orange), **binary**, **GraphQL**, and **JSON** (with a dropdown arrow). A **Beautify** button is also present. The main body area contains the following JSON payload:

```
1 {
2   "title": "Water Mark",
3   "author": "Jimmy"
4 }
```

Below the request section, there is a navigation bar with tabs for **Body**, **Cookies**, **Headers (7)**, **Test Results**, and **Security**. The **Headers (7)** tab is selected and highlighted in green. To the right of the navigation bar, there is a status indicator showing **200 OK**, **325 ms**, and **360 B**, along with a **Save Response** button. Under the **Body** tab, there are buttons for **Pretty**, **Raw**, **Preview**, and **Visualize**, with **Pretty** selected and highlighted in grey. There is also a **JSON** dropdown with a dropdown arrow. The response body area contains the following JSON data:

```
1 {
2   "id": 1,
3   "title": "Water Mark",
4   "author": "Jimmy",
5   "updatedAt": "2023-03-04T08:09:54.356Z",
6   "createdAt": "2023-03-04T08:09:54.356Z"
7 }
```

GET ▼ localhost:3000/books Send ▼

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

**Query Params**

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
	Key	Value	Description		

**Body** Cookies Headers (7) Test Results Security 200 OK 23 ms 487 B Save Response ▼

Pretty Raw Preview Visualize JSON ≡

```

1  [
2    {
3      "id": 1,
4      "title": "Water Mark",
5      "author": "Jimmy",
6      "createdAt": "2023-03-04T08:09:54.356Z",
7      "updatedAt": "2023-03-04T08:09:54.356Z"
8    },
9    {
10      "id": 2,
11      "title": "White Noise",
12      "author": "Luke",
13      "createdAt": "2023-03-04T08:11:01.466Z",
14      "updatedAt": "2023-03-04T08:11:01.466Z"
15    }
16 ]

```

**EVENT** TIME

- Prepare 3.84 ms
- Socket Initialization 0.79 ms
- DNS Lookup 0.19 ms
- TCP Handshake 0.2 ms
- Transfer Start 20.52 ms
- Download 0.94 ms
- Process 0.04 ms

**Total** 26.5 ms

**phpPgAdmin**

PostgreSQL 14.6 running on localhost:5432 -- You are logged in as user "webadmin"

SQL | History | Find | Logout

phpPgAdmin: PostgreSQL? Books? public? books?

Columns Browse Select? Insert? Indexes? Constraints? Triggers? Rules? Admin Info Privile

```
SELECT * FROM "public"."books";
```

Submit

Actions	id	title	author	createdAt	updatedAt
Edit Delete	1	Water Mark	Jimmy	2023-03-04 08:09:54.356+00	2023-03-04 08:09:54.356+00
Edit Delete	2	White Noise	Luke	2023-03-04 08:11:01.466+00	2023-03-04 08:11:01.466+00

2 row(s)

Expand | Insert | Refresh

The screenshot shows the phpPgAdmin interface for PostgreSQL 14.6. On the left, a sidebar displays the database schema tree. It starts with 'Servers', then 'PostgreSQL', followed by 'Books' and 'Schemas'. Under 'Books', there is a 'public' schema which contains 'Tables', 'Columns', 'Browse', 'Select', 'Insert', 'Indexes', 'Constraints', 'Triggers', 'Rules', and 'Admin' options. The 'Tables' node under 'public' has a single child 'books'. On the right, the main panel shows the title 'PostgreSQL 14.6 running on localhost:5432 -- You are logged in as user "webadmin"'. Below the title is a breadcrumb trail: 'phpPgAdmin: PostgreSQL? Books? public? books?'. A toolbar below the breadcrumb includes links for 'Columns', 'Browse', 'Select?', 'Insert?', 'Indexes?', 'Constraints?', 'Triggers?', 'Rules?', 'Admin', 'Info', and 'Privile'. A SQL query editor window contains the command 'SELECT \* FROM "public"."books";'. Below the query is a 'Submit' button. The results section shows a table with two rows of data. The table has columns: Actions, id, title, author, createdAt, and updatedAt. The first row has id 1, title 'Water Mark', author 'Jimmy', createdAt '2023-03-04 08:09:54.356+00', and updatedAt '2023-03-04 08:09:54.356+00'. The second row has id 2, title 'White Noise', author 'Luke', createdAt '2023-03-04 08:11:01.466+00', and updatedAt '2023-03-04 08:11:01.466+00'. Each row has 'Edit' and 'Delete' buttons in the 'Actions' column. The text '2 row(s)' is displayed below the table. At the bottom of the results panel are links for 'Expand', 'Insert', and 'Refresh'.

---

# MONGO-DB

---

MongoDB named a Leader in the 2022 Gartner® Magic Quadrant™ for Cloud Database Management Systems – learn more

MongoDB Products Solutions Resources Company Pricing

Try Free

I MONGODB

# Build the next big thing

The developer data platform that provides the services and tools necessary to build distributed applications fast, at the performance and scale users demand.

[Start Free](#) [Documentation →](#)

**37k+** **100+** **140k+** **#1**

Customers → Regions across AWS, Azure, and GCP → Developers join every month → Most used modern database →

```
Connecting to: mongodb+srv://cluster0.ab123.mongodb.net/
Using MongoDB: 6.0

North America - westus2
South America - brazilsouth
Asia Pacific - asia-east1
Europe - westeurope
Europe - northeurope
Europe - europe-west1
Middle East - europe-west1
Africa - europe-west1
North America - northcentralus
North America - centralus
North America - southcentralus
North America - eastus
```

<https://www.mongodb.com/>

## MongoDB Tutorial

- MongoDB - Home
- MongoDB - Overview
- MongoDB - Advantages
- MongoDB - Environment
- MongoDB - Data Modeling
- MongoDB - Create Database
- MongoDB - Drop Database
- MongoDB - Create Collection
- MongoDB - Drop Collection
- MongoDB - Data Types
- MongoDB - Insert Document
- MongoDB - Query Document
- MongoDB - Update Document
- MongoDB - Delete Document
- MongoDB - Projection
- MongoDB - Limiting Records
- MongoDB - Sorting Records
- MongoDB - Indexing
- MongoDB - Aggregation
- MongoDB - Replication
- MongoDB - Sharding
- MongoDB - Create Backup
- MongoDB - Deployment

MongoDB is a cross-platform, document oriented database that provides, high performance, high availability, and easy scalability. MongoDB works on concept of collection and document.

### Database

Database is a physical container for collections. Each database gets its own set of files on the file system. A single MongoDB server typically has multiple databases.

### Collection

Collection is a group of MongoDB documents. It is the equivalent of an RDBMS table. A collection exists within a single database. Collections do not enforce a schema. Documents within a collection can have different fields. Typically, all documents in a collection are of similar or related purpose.

### Document

A document is a set of key-value pairs. Documents have dynamic schema. Dynamic schema means that documents in the same collection do not need to have the same set of fields or structure, and common fields in a collection's documents may hold different types of data.

The following table shows the relationship of RDBMS terminology with MongoDB.

RDBMS	MongoDB
Database	Database
Table	Collection
Tuple/Row	Document
column	Field
Table Join	Embedded Documents
Primary Key	Primary Key (Default key _id provided by MongoDB itself)
Database Server and Client	
mysqld/Oracle	mongod

Home    HTML    CSS    JAVASCRIPT    SQL    PYTHON    JAVA    PHP    BOOTSTRAP    HOW TO    W3.CSS    C    C++    C#    REACT    R

# MongoDB Tutorial

[MongoDB HOME](#)

[MongoDB Get Started](#)

[MongoDB Query API](#)

[MongoDB Create Database](#)

[MongoDB Create Collection](#)

[MongoDB Insert](#)

[MongoDB Find](#)

[MongoDB Update](#)

[MongoDB Delete](#)

[MongoDB Query Operators](#)

[MongoDB Update Operators](#)

[MongoDB Aggregations](#)

[MongoDB Indexing/Search](#)

[MongoDB Validation](#)

[MongoDB Data API](#)

[MongoDB Drivers](#)

[MongoDB Node.js Driver](#)

[MongoDB Charts](#)

## MongoDB

### Exercises

[MongoDB Exercises](#)

# MongoDB Tutorial

[« Home](#) [Next »](#)

MongoDB is a document database. It stores data in a type of JSON format called BSON.

If you are unfamiliar with JSON, check out our [JSON tutorial](#).

A record in MongoDB is a document, which is a data structure composed of key value pairs similar to the structure of JSON objects.

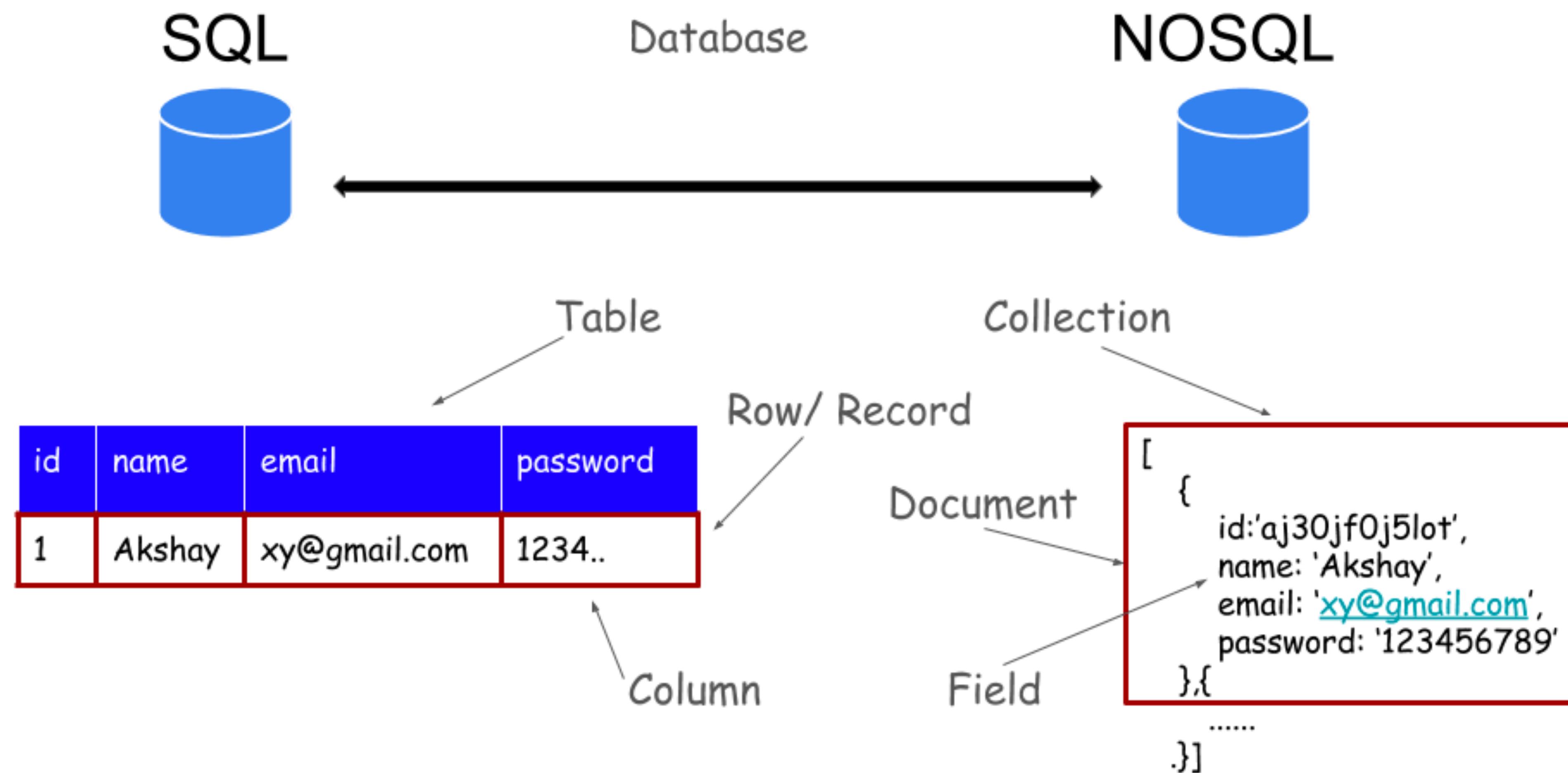
[Start learning MongoDB now »](#)

## A MongoDB Document

Records in a MongoDB database are called documents, and the field values may include numbers, strings, booleans, arrays, or even nested documents.

### Example Document

<https://www.w3schools.com/mongodb/>





A new MongoDB server was successfully added to your NodeREST environment in Ruk-Com.

Please use the following data to access and manage the added node:

**Admin Panel:** <http://node40731-noderest.proen.app.ruk-com.cloud/>

**Login:** admin

**Password:** SBFsqa14913

[Connect to MongoDB](#) from your application code using the linked instruction and the following details:

The screenshot shows the Mongo Express web application. At the top, there's a header bar with a warning icon (Not Secure), the URL node40731-noderest.proen.app.ruk-com.cloud, and several small icons for file operations, search, and help.

The main title "Mongo Express" is followed by a "Database" dropdown menu.

# Mongo Express

## Databases

	Database Name	Create Database
View	admin	Del
View	config	Del
View	local	Del
View	test	Del

## Server Status

Hostname	node40731-noderest.proen.app.ruk-com.cloud	MongoDB Version	4.0.2
Uptime	1645024 seconds (19 days)	Server Time	Mon, 06 Mar 2023 05:53:33 GMT

# Mongo Express

## Databases

Database Name

+ Create Database

View

admin

Del

View

config

Del

View

local

Del

View

test

Del

## Server Status

Hostname	node40731-noderest.proen.app.ruk-com.cloud	MongoDB Version	4.0.2
Uptime	1645024 seconds (19 days)	Server Time	Mon, 06 Mar 2023 05:53:33 GMT

The screenshot shows the Rukus Cloud interface for managing NoSQL databases. At the top, there's a navigation bar with a tree view for 'NoSQL Databases' (MongoDB 4.0.2), a search bar ('Node ID: 40731'), and various icons for database management. Below this is a list of databases, with one entry for 'ThaiClassify' (status: Stopped). A modal window titled 'Settings' is open, specifically for the 'Endpoints' section. The modal contains instructions: 'Here you can add and manage endpoints, which can be used by other resources for communication'. It has fields for 'Edit Endpoint':

- Node: Node ID: 40731
- Name: MongoDB
- Private Port: 27017
- Protocol: TCP
- Public Port: 11344
- Access URL: node40731-noderest.proen.app.ruk-com

At the bottom right of the modal are 'Cancel' and 'Apply' buttons. The main dashboard below the modal shows the same information: Node ID: 40731, MongoDB, 27017, TCP, 11344, and the access URL node40731-noderest.proen.app.ruk-com.cloud:11344.

## mongoose TS

7.0.1 • Public • Published 5 days ago

 Readme

 Code Beta

 7 Dependencies

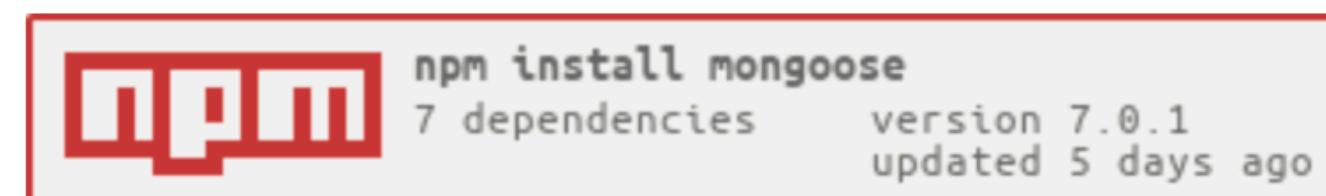
 13,580 Dependents

 772 Versions

# Mongoose

Mongoose is a **MongoDB** object modeling tool designed to work in an asynchronous environment. Mongoose supports **Node.js** and **Deno** (alpha).

 Test passing  npm package 7.0.1  deno.land/x 6.10.2  deno.land/x Published



## Documentation

The official documentation website is [mongoosejs.com](https://mongoosejs.com).

Mongoose 7.0.0 was released on February 27, 2023. You can find more details on [backwards breaking changes in 7.0.0 on our docs site](#).

## Support

### Install

> npm i mongoose

### Repository

 [github.com/Automattic/mongoose](https://github.com/Automattic/mongoose)

### Homepage

 [mongoosejs.com](https://mongoosejs.com)

 Fund this package

### Weekly Downloads

2,136,278



Version

7.0.1

License

MIT

<https://www.npmjs.com/package/mongoose>

The screenshot shows the Visual Studio Code interface. The left sidebar contains various icons for file operations like Open Editors, Find, and Save. The Explorer pane on the left shows a hierarchical view of a project named 'NODEREST'. Inside 'NODEREST', there are folders for 'Database', 'node\_modules', 'NodeExpressREST', and 'src'. The 'src' folder contains files such as 'CRUDBookNoDB.js', 'CRUDBookSQLite.js', 'index.js', 'MongoDBCRUD.js', 'PGSqlCRUD.js', 'SequelizeSQLiteCR...', '.env', '.gitignore', 'index.js', 'ncase.js', 'package-lock.json', and 'package.json'. The 'package.json' file is currently selected in the Explorer. The main editor area on the right displays the JSON content of the 'package.json' file. The code is color-coded, with numbers on the left indicating line numbers. The JSON structure includes fields for name, version, description, main file, scripts (with start, dev, and format options), repository URL, keywords, author, license, bugs URL, homepage URL, dev dependencies (nodemon and prettier), and dependencies (body-parser, dotenv, express, mongoose, sequelize, and sqlite3).

```
1 {  
2   "name": "noderest",  
3   "version": "1.0.0",  
4   "description": "",  
5   "main": "index.js",  
6   "scripts": {  
7     "format": "prettier --write src/*.js",  
8     "start": "node index.js",  
9     "dev": "nodemon index.js"  
10    },  
11    "repository": {  
12      "type": "git",  
13      "url": "git+https://github.com/Anirach/NodeREST.git"  
14    },  
15    "keywords": [],  
16    "author": "",  
17    "license": "ISC",  
18    "bugs": {  
19      "url": "https://github.com/Anirach/NodeREST/issues"  
20    },  
21    "homepage": "https://github.com/Anirach/NodeREST#readme",  
22    "devDependencies": {  
23      "nodemon": "^2.0.20",  
24      "prettier": "^2.8.3"  
25    },  
26    "dependencies": {  
27      "body-parser": "^1.20.2",  
28      "dotenv": "^16.0.3",  
29      "express": "^4.18.2",  
30      "mongoose": "^7.0.0",  
31      "sequelize": "^6.29.0",  
32      "sqlite3": "5.0.2"  
33    }  
34  }  
35 }
```

```
src > js MongoDBCRUD.js > ...
Anirach, 6 minutes ago | 2 authors (You and others)

1 // Description: REST API with MongoDB
2 // npm install express mongoose body-parser
3 // Run this file with node MongoDBREST.js
4 // Test with Postman
5
6 const express = require("express");
7 const mongoose = require("mongoose");
8 const bodyParser = require("body-parser");
9
10 // Database connection
11 mongoose.connect(
12   "mongodb://admin:SBFsqa14913@node40731-noderest.proen.app.ruk-com.cloud:11344",
13   {
14     useNewUrlParser: true,
15     useUnifiedTopology: true,
16   }
17 );
18
19 const Book = mongoose.model("Book", {
20   id: {
21     type: Number,
22     unique: true, // Ensures uniqueness of the "id" field
23     required: true, // If you want "id" to be required
24   },
25   title: String,
26   author: String,
27 });
28
```

```
28
29  const app = express();
30  app.use(bodyParser.json());
31
32 // Create
33 app.post("/books", async (req, res) => {
34   try {
35     // Get the last book record to determine the next ID
36     const lastBook = await Book.findOne().sort({ id: -1 });
37     const nextId = lastBook ? lastBook.id + 1 : 1;
38
39     // Create a new book with the next ID
40     const book = new Book({
41       id: nextId, // Set the custom "id" field
42       ...req.body, // Include other book data from the request body
43     );
44
45     await book.save();
46     res.send(book);
47   } catch (error) {
48     res.status(500).send(error);
49   }
50 });
51
```

```
52 // Read all
53 app.get("/books", async (req, res) => {
54   try {
55     const books = await Book.find();
56     res.send(books);
57   } catch (error) {
58     res.status(500).send(error);
59   }
60 });
61
62 // Read one
63 app.get("/books/:id", async (req, res) => {
64   try {
65     const book = await Book.findOne({id:req.params.id});
66     res.send(book);
67   } catch (error) {
68     res.status(500).send(error);
69   }
70 });
71
```

```
71
72 // Update
73 app.put("/books/:id", async (req, res) => {
74   try {
75     const book = await Book.findOneAndUpdate({id:req.params.id}, req.body, {
76       new: true,
77     });
78     res.send(book);
79   } catch (error) {
80     res.status(500).send(error);
81   }
82 });
83
84 // Delete
85 app.delete("/books/:id", async (req, res) => {
86   try {
87     const book = await Book.findOneAndDelete({id:req.params.id});
88     res.send(book);
89   } catch (error) {
90     res.status(500).send(error);
91   }
92 });
```

```
93
94 // Start the server
95 const PORT = process.env.PORT || 3000;
96 app.listen(PORT, () => {
97   console.log(`Server started at http://localhost:${PORT}`);
98 });
99
```

POST  Send

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

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

```
1 {  
2   "title": "By The Tree",  
3   "author": "Kate"  
4 }
```

Body Cookies Headers (7) Test Results Security 200 OK 30 ms 322 B Save Response

Pretty Raw Preview Visualize **JSON**

```
1 {  
2   "title": "By The Tree",  
3   "author": "Kate",  
4   "_id": "64057edcd6b2bdfc6e481b49",  
5   "id": 3,  
6   "__v": 0  
7 }
```

Mongo Express Database: test ▾ ➔ Collection: books ▾

## Viewing Collection: books

New Document New Index

Simple Advanced

Key Value String Find

Delete all 3 documents retrieved

_id	title	author	id	_v
64057eadd6b2bdfc6e481b43	Who care	Dave	1	0
64057ec0d6b2bdfc6e481b46	Blue Ride	Bob	2	0
64057edcd6b2bdfc6e481b49	By The Tree	Kate	3	0

localhost:5500

Dimensions: Responsive ▾ 616 x 664 100% ▾ No throttling ▾

**Books**

TITLE	AUTHOR	ACTIONS
Who care	Dave	<a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a>
Who care	Dave	<a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a>
Who care	Dave	<a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a>

Create New Book

Elements Console Sources Network Performance Memory Application

```

<!-- books.ejs -->
<!DOCTYPE html>
<html>
  <head> ...
  </head>
  <body data-new-gr-c-s-check-loaded="14.1098.0" data-gr-ext-installed> == $0
    <div id="MathJax_Message" style="display: none;"></div>
    <h1>Books</h1>
    <table class="book-table">
      <tbody>
        <tr> ...
        <tr>
          <td>Who care</td>
          <td>Dave</td>
          <td>
            <a href="/book/1">View</a>
            <a href="/update/1">Edit</a>
            <a href="/delete/1">Delete</a>
          </td>
        </tr>
        <tr> ...
        <tr> ...
      </tbody>
    </table>
    <div class="link-center"> ...
    <script type="text/javascript" src="https://cdnjs.cloudflare.com/ajax/libs/mathjax/2.7.0/MathJax.js?config=TeX-AMS-MML_HTMLorMML&delayStartupUntil=configured"></script>
    <script id="texAllTheThingsPageScript" type="text/javascript" src="chrome-extension://cbimabofgmfdkicghcadidpemeenbfffn/js/pageScript.js" inlinemath="[[["$, "$"], [";", ";"]]]" displaymath="[[["$, "$"], ["\\[", "\\]"]]]" skiptags="["script", "noscript", "style", "textarea", "pre", "code"]" ...
  </body>

```

Styles Computed Layout

Filter :hov .cls +

element.style { }

body { style.css:2 font-family: 'Open Sans', sans-serif; font-size: 16px; }

body { user agent stylesheet display: block; margin: 8px; }

margin 8  
border -  
padding -  
8 - 964x430 - 8

---

# THANK YOU

---

- CSS-Front-End
- PostgreSQL
- MongoDB
- Postman Test
- Front-End connect with Back-End