Backend Development with Node.js

A Beginner-Friendly Guide That Actually Makes Sense

Bishal Rijal

Bishal Rijal

Digital Bishal

History of JavaScript

1. JavaScript was developed in 1995 by Brendan Eich at Netscape Communications in 10 days.
2. During development, it was first named Mocha.
3. Later, its name was changed to LiveScript.
4. Finally, it was renamed JavaScript for marketing purposes due to Java's popularity.
5. JavaScript was initially used in the Netscape Navigator web browser.
6. Microsoft created its own version of JavaScript called JScript for Internet Explorer.
7. This led to browser compatibility problems between different browsers.
8. To solve these issues, JavaScript was standardized as ECMAScript in 1997 by ECMA International.
9. ECMAScript 3 (1999) made JavaScript stable and widely usable on the web.
10. With the introduction of Node.js in 2009, JavaScript expanded from browsers to server-side development.

How website works?

A website works when a user enters a URL in a browser, the browser (frontend) sends a request to the server, the backend processes the request, interacts with the database to fetch or store data if needed, then sends a response back, and the frontend displays the content to the user.

HTTP request methods

GET: Used to fetch/read data from the server.
Example: Get a list of students from the database.
POST: Used to send new data to the server.
Example: Submit a registration form to create a new user.
PATCH: Used to update existing data partially.
Example: Update only the email address of a user.
DELETE: Used to remove data from the server.
Example: Delete a user account from the database.

One-line memory trick:

GET = Read | POST = Create | PATCH = Update | DELETE = Remove

HTTP response status codes

1xx – Informational

- 100 Continue → Request received, continue sending data

2xx – Success

- 200 OK → Request successful
- 201 Created → Resource created successfully

3xx – Redirection

- 301 Moved Permanently → URL changed permanently

4xx – Client Error

- 400 Bad Request → Invalid request
- 401 Unauthorized → Login required
- 403 Forbidden → Access denied
- 404 Not Found → Resource not foud

5xx – Server Error

- 500 Internal Server Error → Server crashed or failed
- 503 Service Unavailable → Server temporarily down

One-line memory trick:

2xx = Success | 3xx = Redirect | 4xx = Client Error | 5xx = Server Error

What is Node.js?

JavaScript run time environment.

It is used for sever side programming.

Note: It is not a programming language, library, or framework

Installation of Node.js

To check whether node is installed or not In your computer, open terminal and type node -v

Node Installation

For **Mac** Users : https://nodejs.org/en

For **Windows** Users : https://nodejs.org/en/download

Windows user can also refer this video : https://www.youtube.com/watch?v=EIzdQxMXcrc

Node REPL

Read, Evaluate, Print and Loop

.help give us command, global command

To open Node REPL:

i. Open CMD

ii. Type node

Node Files

fileName.js

To run file using Node.js, node fileName.js

Module.exports

Require: A built in function to include external module that exists in seperate file.

module.exports = A special object

NPM - Node Package Manager

What is NPM?

NPM is the standard package manger for NodeJs.

npm install <package name>

NPMJS.com

example: npm i figlet

package.json

The package.json file contains descriptive and functional metadata about a project, such as name, version, and dependencies.

npm init

node_modules

The node_modules folder contains every installed dependency for your project.

package-lock.json

It records the exact version of every installed dependency, including its sub-dependencies and their version.

Library vs Framework

Library

A library is a collection of pre-written code that can be used to perform specific tasks.

eg - axios

Framework

A framework is a set of pre-written code that provides a structure for developing software applications.

eg - express

express

Express is a Node.js web application framework that helps us to make web applications. It is used for server side programming.

Major usages

1. Listen for incoming requests.
2. Parse request
3. To match response with routes.
4. sent suitable response

Getting started with express

```
const express = require("express");
const app = express();

let port = 8080;

app.listen(port, () => {
console. log (`app listening on port ${port}');
});
```

Ports are the logical endpoints of a network connection that is used to exchange information between a web server and a web client.

Handling requests

```
app.use((req, res) => {
console. log ("new incoming request");
});
```

Sending response

In Express.js, sending a response refers to the process of the server completing the HTTP request-response cycle by sending data back to the client (e.g., a web browser or API consumer). This is primarily accomplished using methods on the res (response) object within a route handler.

A basic example using res.send() in a route handler:

```
const express = require('express');
const app = express();

app.get('/', (req, res) => {
  res.status(200).send('Hello World from Express!');
});

app.listen(3000, () => {
  console.log('Server listening on port 3000');
});
```

Routing

It is process of selecting a path for traffic in a network or between or across multiple networks.

```
app.get("/apple", (req, res) => {
res.send ({
name: "apple",
color: "red",
});
});
```

path parameter

req.params

```
app.get("/ig/:username", (req, res) => {
let { username } = req.params;
res.send(`This account belongs to @${username}');
```

query string

```
app.get("/search", (req, res) => {
let { q } = req.query;
if (!q) {
res. send ("No search query");

res. send (`These are the results for: ${q}');
});
```

Nodemon

To automaticall restart server with code changes.

npm install nodemon

Templating

EJS (Embeded JavaScript Templates)

npm i ejs

EJS is a simple templating language that lets you generate HTML markup with plain JavaScript.

EJS is a simple templating language that lets you generate HTML markup with plain JavaScript. No religiousness about how to organize things.

Using EJS

```
// Set EJS as templating engine
app.set("view engine", "ejs")

app.get("/", (req,res)=>{
    res.render("home.ejs")
})
```

Create home.ejs file inside views.

Views Directory

```
const path = require("path")
// Set path for views
app.set("views", path.join(__dirname+"/views"))
```

Interpolation Syntax

Interpolation refers to embedding expressions into marked-up text.

Tags

- `<%` 'Scriptlet' tag, for control-flow, no output
- `<%_` ‘Whitespace Slurping’ Scriptlet tag, strips all whitespace before it
- `<%=` Outputs the value into the template (HTML escaped)
- `<%-` Outputs the unescaped value into the template
- `<%#` Comment tag, no execution, no output
- `<%%` Outputs a literal '<%'
- `%>` Plain ending tag
- `-%>` Trim-mode ('newline slurp') tag, trims following newline
- `_%>` ‘Whitespace Slurping’ ending tag, removes all whitespace after it

Instagram EJS

Create a basic template for an Instagram page based on the following route :

/ig/:username

```
app.get("/instagram/:username", (req, res)=>{
    const username = req.params.username
    res.render("instagram.ejs", {data: username})
})
```

Create instagram.ejs file inside views.

Conditional statement in EJS

Adding condition inside ejs.

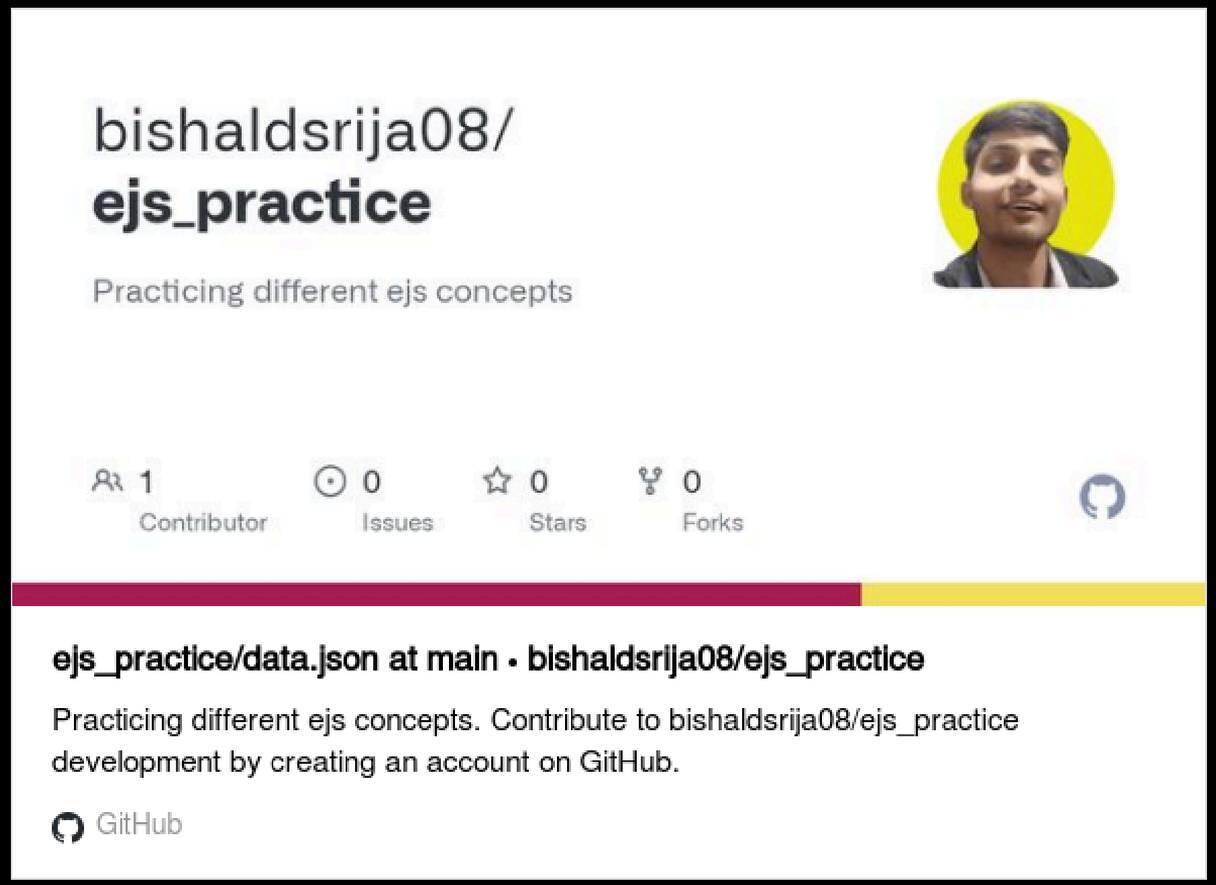
```
<% if (data == 6) {%>
    <p>Dice is six, roll again!</p>
<% } %>
```

Loops in EJS

Adding loops inside ejs.

```
<h2>Followers:</h2>
<% for(user of followers){ %>
<li><%= user%></li>
<% } %>
```

Instagram Page with EJS



data.json from here

Includes

To include one EJS template within another, you use the special EJS tag <%- include(...) %> within your main file. This function allows you to reuse common sections of HTML, such as headers and footers, across multiple pages.

<!-- Include the navbar -->
<%- include("./navbar.ejs") %>

Get and Post request

GET

> Used to GET some response

> Data sent in query strings
(limited, string data & visible in URL)

POST

> Used to POST something (for Create/ Write/ Update)

> Data sent via request body (any type of data)

Handling post request

Middlewares to parse incoming request bodies

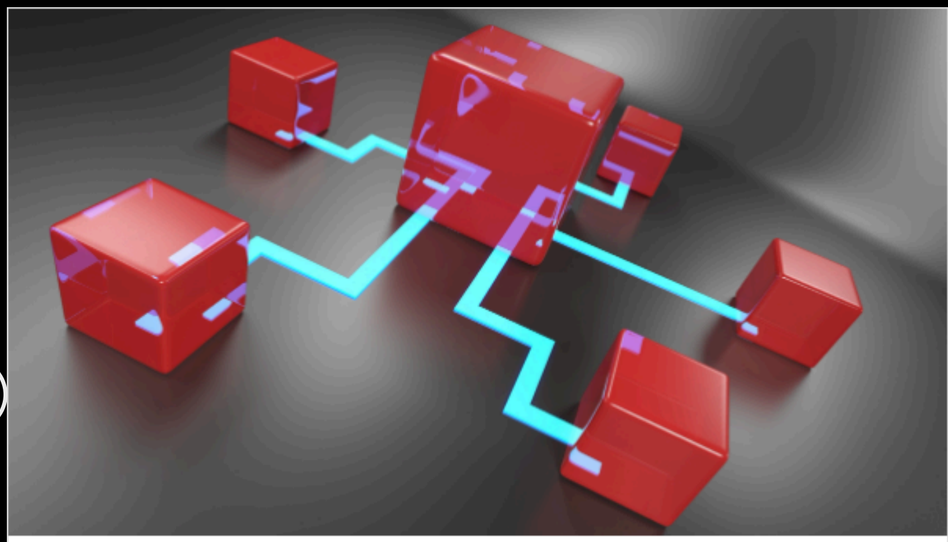
```
app.use(express.json()); // Middleware to parse JSON bodies
app.use(express.urlencoded({extended: true})); // Middleware to parse URL-encoded bodies
```

Rest

Representational State Transfer

REST is an architectural style that defines a set of constraints to be used for creating web services.

READ MORE



Best practices for REST API design - Stack Overflow

REST APIs are one of the most common kinds of web interfaces available today. They allow various clients including browser apps to communicate with services via the REST API. Therefore, it's very important to design REST APIs properly so tha…

stackoverflow / Mar 2, 2020

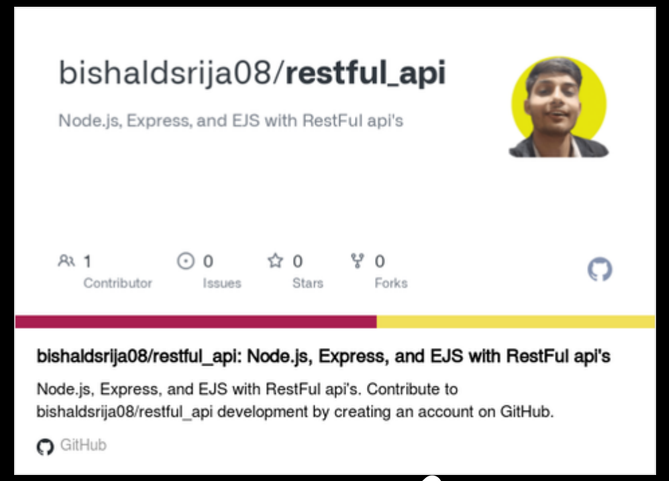
CRUD

Create, Read, Update, and Delete

GET retrieves resources.
POST submits new data to the server
PUT updates existing data
PATCH update existing data partially
DELETE removes data

Creating RESTful APIs

Get	/posts	To get all posts	Read
Post	/posts	To add a new post	Create
Get	/posts/:id	To get one post (using ID)	Read
Patch	/posts/:id	To update a specific post	Update
Delete	/posts/:id	To delete a specific post	Delete



Reference GitHub Repo

Implement: GET /posts

Index Route

GET	/posts	to get data for all posts

Implement: Post /posts

Create Route

POST	/posts	to add new post

2 routes:

Serve the form	GET	/posts/new
Add the new post	POST	/posts

Redirect

res.redirect(url)

```
// post api
app.post("/posts", (req, res) => {
    const { username, title } = req.body;
    posts.push({username, title})
    res.redirect("/");
})
```

Show Route

Implement: Get /post/:id

GET	/posts/:id	to get one post (using id)

Create id for Posts

UUID Package

Universally unique identifier

npm install uuid

Update Route

Implement: Patch /post/:id

Patch	/posts/:id	to update a specific post (using id)

Edit Route

Create form for Update

Get	/posts/:id/edit	Serve the edit form

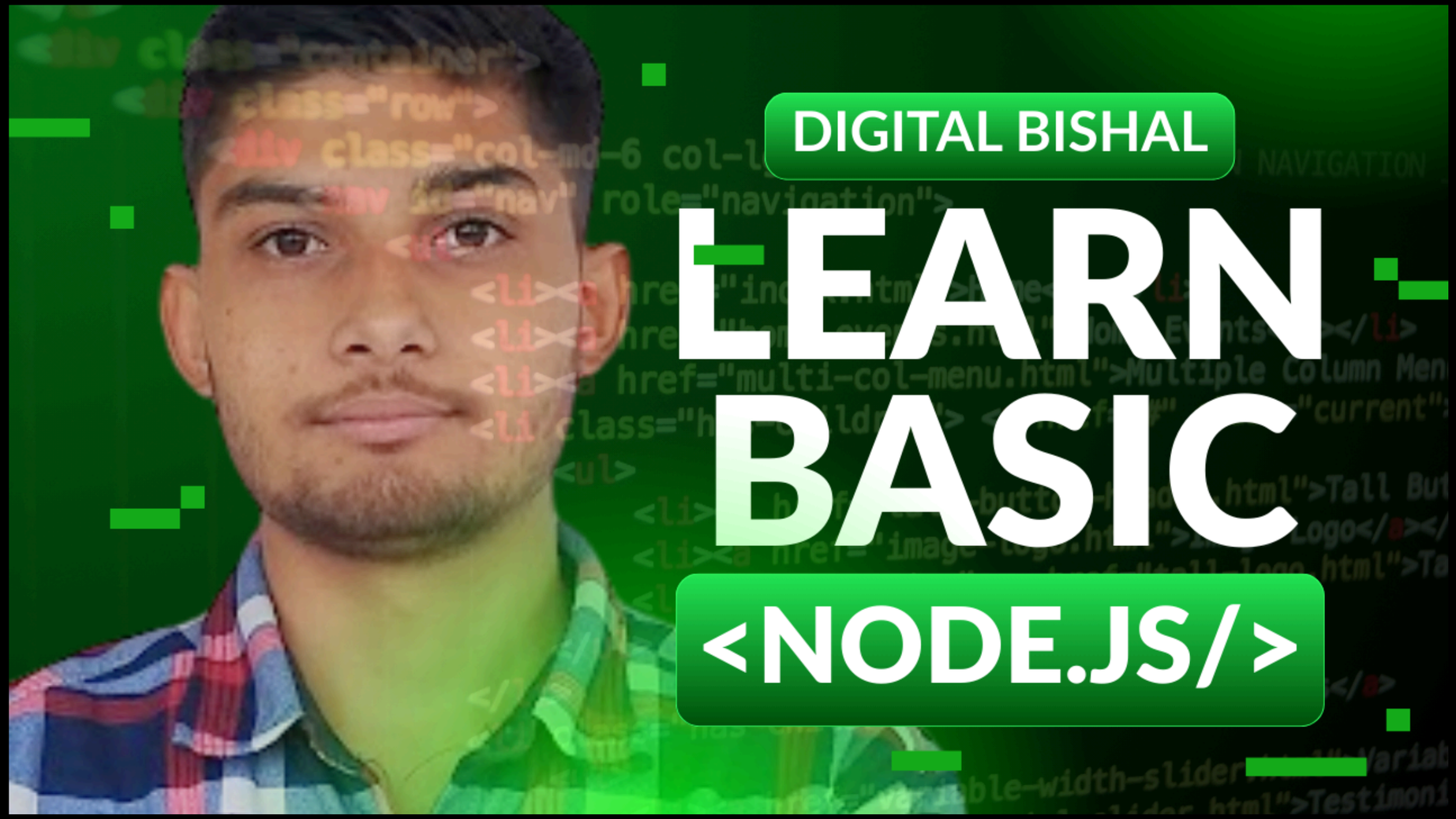
We can’t send a patch and a delete request from the form. We can only send get and POST requests from a form. To achieve delete and patch request, we use **npm i method-override** package.

Delete Route

Destroy route

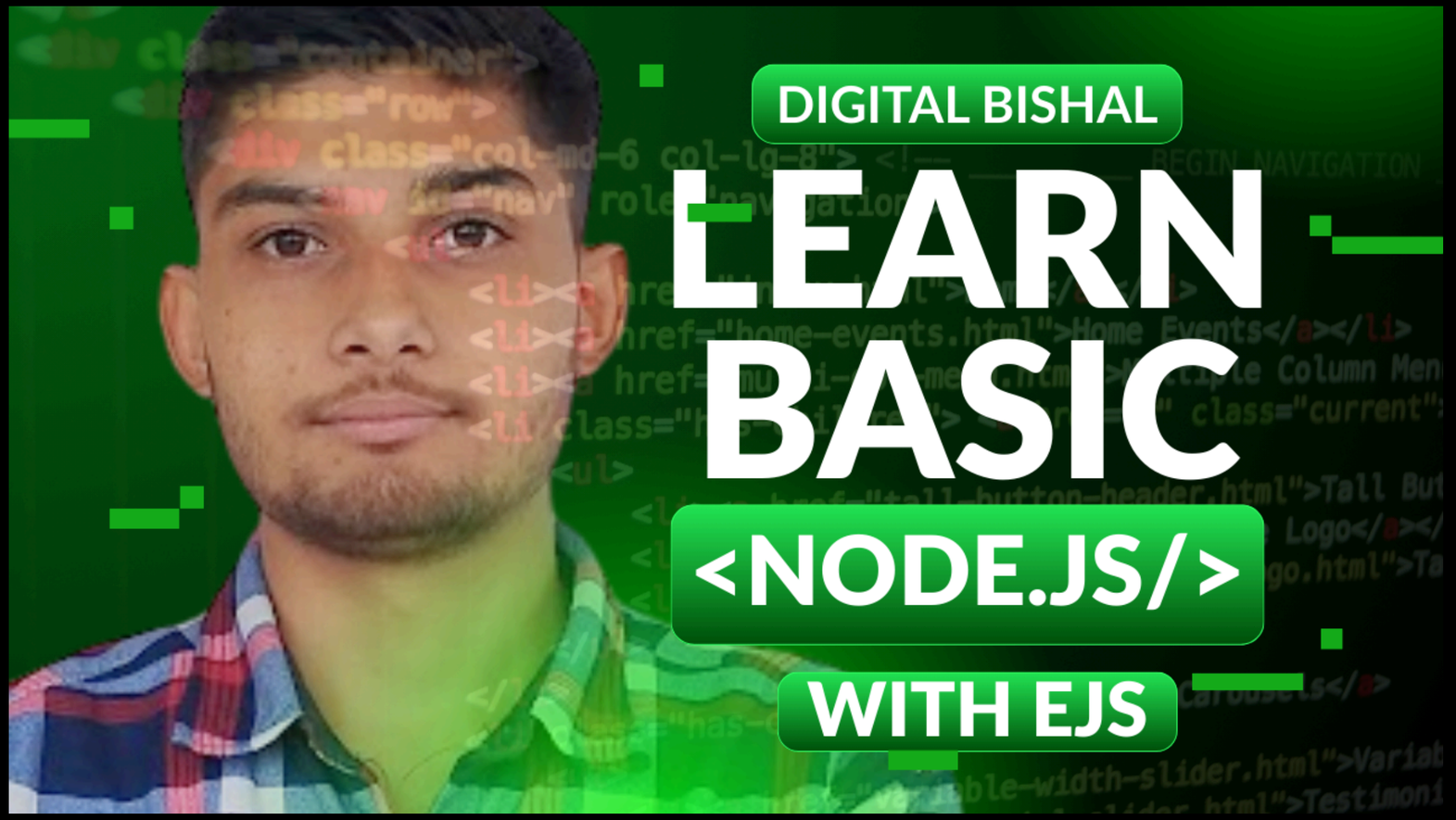
Delete	/posts/:id	To delete the specific post

Learn Node.js Step by Step for Beginners



Watch now

Learn Node.js with EJS Step by Step for Beginners



Watch now