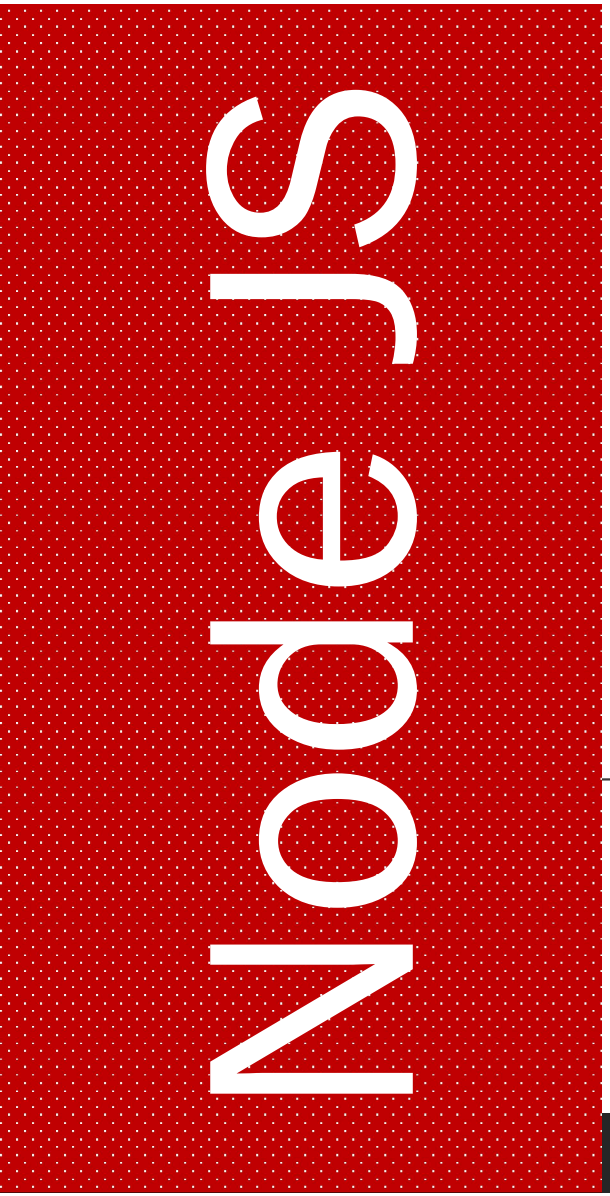

Learning

Node JS

PRAVEEN NAIR

Top Backend Technologies

Node.js - is an open-source, cross-platform JavaScript framework that is used to build server-side and networking applications.

Django - is an open-source framework based on Python. It is a web framework from the server's side. Django follows the Model Template View (MTV) architecture.

Spring Boot - is an open-source web framework based on Java that allows developers to build production-grade and standalone applications.

ASP.NET - is an open source web framework, created by Microsoft, for building modern web apps and services that run on macOS, Linux, Windows, and Docker.

Laravel - is a PHP framework that provides a built-in user interface, flexibility, API support, creativity, and an extensive range of various libraries that help in the development process of secure web applications

Introduction to Node JS

Node.js was invented for server side scripting and uses JavaScript.

Node.js is free and open source server environment.

Node.js runs on various OS(Windows, Linux, etc.)

Node.js continues with the next request in case current request is delayed.

Node.js can collect data submitted by front end users.

Node.js can add, delete, modify data in your database

Visit https://nodejs.org/en to download and install node js

Introduction to Express JS

Express.js is fast and minimalist web framework for Node.js. An Express application is essentially a series of middleware function calls.

Express.js has powerful models for URL routing (matching an incoming URL with a server function), and handling HTTP requests and responses.

By making XML HTTP Requests (XHRs) or GETs or POSTs from your React.js front end, you can connect to Express.js functions that power your application.

Request-Response Cycle

Client

- Web Browser
- Mobile App
- User Sends Request

HTTP REQUEST

- GET /users
- Headers
- Body (optional)

HTTP RESPONSE

- Status Code
- Headers
- Data (JSON, HTML)

Web Server

- Processes Request
- Accesses Database
- Sends Response

Create and setup server

Open terminal and type the following command
npm init –y //this will create package.json file

Add following line in package.json, below name attribute
"type":"module",

Create .gitignore file (Add /node_module and .gitignore in the file)

npm install express

index1.js - creating web server

```
import express from 'express'

const app = express()

app.listen(8080)

.....,,,,,,,,,,,,,,,,,,,,,,
```

http://localhost:8080

Cannot GET /

API testing tool

Postman - One of the most popular API testing platforms.

index2.js – req/res object

```
import express from "express";
const app = express();
app.listen(8080, () => console.log("Web Server has been started"));
app.get("/", (req, res) => {
  console.log(req.url);
  console.log(req.method)
  console.log(req.body)
  res.send("response from server");
});
```

index3.js – multiple routes

```
import express from "express";
const app = express();
app.listen(8080, () => console.log("Web Server has been started"));
app.get("/", (req, res) => {
  console.log(req.url);
  res.send("response from server for route /");
});

app.get("/home", (req, res) => {
  console.log(req.url);
  res.send("response from server for route /home");
});
```

index4.js – req.params

```
app.get("/:name", (req, res) => {
  res.send(req.params.name);
});
app.get("/:name/:age", (req, res) => {
  res.send(req.params.name + req.params.age);
});
app.get("/name/:name/age/:age", (req, res) => {
  res.send(req.params.name + req.params.age);
});
```

index5.js – req.headers

```
app.get("/", (req, res) => {
  console.log(req.headers);
  console.log(req.headers.host);
  console.log(req.headers.authorization);
  res.send("response from server for route /");
});
```

index6.js – req.query

```
//http://localhost:8080/?name=amy&age=23

app.get("/", (req, res) => {
  res.send(req.query.name+req.query.age);
});
```

index7.js – res.json

```
app.get("/", (req, res) => {
  const user = {
    name: "John",
    email: "john@gmail.com",
    role: "student",
  };
  res.json(user);
});
```

index8.js – res.json – array of objects

```
const users = [
  {id:1,name: "John",email: "john@gmail.com",role: "student"},
  {id:2,name: "Admin",email: "admin@gmail.com",role: "admin"},
];
app.get("/", (req, res) => {
  res.json(users);
});
```

index9.js – signup/login/bcrypt

```
app.post("/signup", async (req, res) => {
  const user = req.body;
  const hashedPwd = await bcrypt.hash(req.body.password, 10);
  user.password = hashedPwd
  users.push(user);
  res.json(users);
});
.....
app.post("/login", async (req, res) => {
  const { email, password } = req.body;
  const found = users.find((user) => user.email === email);
  if (found) {
    const validate = await bcrypt.compare(password, found.password);
    if (validate) {
      res.send("Hello " + found.name);
    } else {
      res.send("Invalid Password");
    }
  } else {
    res.send("User not found");
  }
});
```

What is REST API?

A **REST API** (Representational State Transfer Application Programming Interface) is a way for different applications to communicate with each other over HTTP using standard web methods.

Stateless
Every request must contain all required information
Server does NOT remember previous requests.
Example:
If user logs in, client must send token with each request.

REST vs Traditional Website

Traditional Website	REST API
Returns HTML	Returns JSON
Server renders UI	Client renders UI
Full page reload	Data only