NodeJS Basics

09 September 2023 12:29

Backend consists of:

1. Server: system that provides services, resources, or functionality to other computers or clients over a network.
2. Database: structured collection of data organized for efficient storage, retrieval, and manipulation.
3. API: set of rules and protocols that allows different software applications to communicate and interact with each other.

NodeJS:

1. Server side JS runtime environment
2. Built on V8 Javascript engine

=> Characteristics:

1. Asynchronous and Non-blocking (can handle multiple concurrent operations without blocking the execution of other code)
2. Event-driven (relies heavily on events and callbacks)
3. Server-side development (use to build server side applications, such as web servers, APIs..)
4. Package management (has lots of open source libraries)
5. Cross-platform (runs on various OS, Windows, MacOS, Linux..)
6. Single Language (JS can be used for both client and server side dev)

=> How to create a NodeJS server:

1. Create a folder (eg. server)
2. cd folder_name
3. npm init -y (this initializes a package.json file)
4. Install nodemon:
   npm install -g nodemon
   -> Nodemon automatically restarts nodejs application whenever changes are detected in code
   -> We don't have to restart the server everytime manually
   -> -g means --global (globally installed in all the applications we will create in future)
5. Create a new file index.js
6. To run the server:
   -> nodemon index.js
   -> If nodemon not installed, node index.js
7. To run the server in another way:
   -> package.json file
      -> In "scripts", remove "test": "echo…"
      -> Add:
         "start": "node index.js",
         "dev": "nodemon index.js"
   -> In terminal, we can write **npm start** to run node index.js and **npm run dev** for nodemon index.js.

=> Modules

1. NodeJS follows module system.
2. Definition:
   A module is a self-contained unit of code in a programming language that encapsulates a set of related functions, variables, or classes, often used for code organization, reusability, and maintaining clean and modular code.
3. Types:
   i. Core modules:
      Built-in modules that come with NodeJS
      Can use them without installing
      Examples: http, path, fs…
         const http = require('http') OR import http from "http"
   ii. Third party modules:
      We can install them through npm
      Examples: express, nodemon…
         import express from "express"
   iii. File based/ custom modules:
      Created by self (developer) to organize code into reusable pieces
      Example:

```
name.js

const myName = "Khushi"
```

```
name.js

const myName = "Khushi"

export default myName
```

```
index.js

import myName from "./name.js

console.log(myName)
```

To use import export instead of require, we have to add "type": "module" after "main" in package.json file.

ExpressJS basics

09 September 2023 16:11

Express:
=> Framework for NodeJS
=> Designed to make it easier to develop server-side applications and APIs in Node by providing a simple and intuitive interface for handling HTTP requests, routing, middleware and more.

=> Characteristics:
1. Routing (allows to define routes, specifying how different HTTP requestes such as GET, POST, DELETE, PUT should be handled)
2. Middleware
3. Templates (eg EJS, handlebards..)
4. Static files
5. Request and response handling
6. Session management
7. RESTful Routing
8. Integration (Can easily integrated with databases like MongoDB, MySQL…)

=> index.js

```
import express from "express"

const app = express()

// methods
app.get("/", (req, res) => {
    res.send("Success")
})

app.listen(5000, () => {
    console.log("Server is running")
}
```

EJS

09 September 2023 18:21

EJS:

1. Stands for Embedded JavaScript
2. Popular templating engine for JavaScript.
3. It allows developers to embed JavaScript code directly into HTML templates, making it easier to generate dynamic content on web pages.
4. Often used in web development frameworks like Express for NodeJS applications.

=> Characteristics:

1. Templating:
   -> EJS provides a way to create templates with placeholders for dynamic data. These placeholders are typically enclosed in `<% %>` tags and can contain JavaScript code.
   -> Example: <h1>Welcome to <%= pageTitle %></h1>

2. Rendering:
   -> To render EJS templates, you typically use a JavaScript engine (like NodeJS) along with an EJS module. You can pass data to the template when rendering it, and EJS will replace the placeholders with the actual values.
   -> Example:

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

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

app.get('/', (req, res) => {
  const data = { pageTitle: 'My EJS App' };
  res.render('index', data);
});

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

3. Control Structures:
   -> EJS allows you to use control structures like `if` statements and loops within your templates to conditionally render content or iterate over data.

4. Partial Templates

5. Escaping