Comprehensive Guide to Server Basics

What is Node.js?

Node.js is a runtime environment that enables executing JavaScript on the server-side, outside of web browsers. Built on Google's V8 JavaScript engine, Node.js excels at handling asynchronous operations and is widely used for building scalable, real-time applications.

What is NPM?

NPM (Node Package Manager) is the default package manager for Node.js. It manages the installation, updating, and configuration of JavaScript libraries, dependencies, and packages used in projects. It also hosts a vast registry of open-source packages.

What is a Server?

A server is a computer or software that provides functionality, services, or resources to other computers (clients) over a network. Examples include web servers (serving websites), database servers (handling database queries), and file servers (sharing files).

What is an API?

An API (Application Programming Interface) is a set of rules and protocols defining how software components should interact. APIs allow different software systems to communicate seamlessly, enabling developers to integrate third-party services into their applications.

What is a REST API?

REST (Representational State Transfer) is an architectural style used to design networked applications. REST APIs utilize HTTP methods and URLs to interact with resources, emphasizing stateless communication and a uniform interface. Each resource is represented by a unique URL.

REST API Best Practices

- Use consistent, clear endpoint structures.

- Employ meaningful HTTP methods appropriately (GET, POST, PUT, DELETE).
- Implement accurate and informative HTTP status codes.
- Ensure stateless communication.
- Provide thorough documentation and API versioning.
- Use nouns for resources, not verbs (e.g., /books not /getBooks).

What is ORM?

ORM (Object-Relational Mapping) simplifies the interaction between databases and programming languages by mapping database tables to objects within the application, making database operations more intuitive and efficient. Examples: Prisma, Sequelize, Mongoose (for MongoDB).

Form Validation

Form validation ensures that user input adheres to predefined formats and rules. It prevents erroneous or malicious data from being processed. Techniques include checking required fields, data types, lengths, regular expressions, and custom validators.

Schema & Model

- **Schema**: Defines the blueprint of data, specifying the structure, types, and constraints.
- **Model**: A representation of a database entity based on the schema, used to perform operations like create, read, update, and delete (CRUD).

What is Hashing?

Hashing transforms input data into a fixed-length string of characters (a hash). Hashing is one-way and secure, commonly used to store passwords securely, verify data integrity, and support authentication mechanisms. Common algorithms: SHA-256, bcrypt.

What is JWT?

JWT (JSON Web Token) is a secure, digitally signed token format used for transmitting information between two parties. JWTs are commonly utilized in authentication and authorization processes, allowing servers to verify a client's identity securely. A JWT consists of a header, payload, and signature.

Why Use Hashing?

Hashing is crucial for security purposes, such as storing sensitive information like passwords securely. It prevents unauthorized access by ensuring data remains confidential and inaccessible if compromised. Passwords should be hashed with salt to make brute-force attacks harder.

Services & Controllers

- **Service**: Contains business logic and data handling, isolating complex operations and logic from controllers.
- **Controller**: Handles incoming requests, invokes relevant services, processes data, and returns responses to the client. Keeps code modular and maintainable.

HTTP Methods

- **GET**: Retrieve resources or data.
- **POST**: Create new resources or submit data.
- **PUT**: Replace or update existing resources entirely.
- **PATCH**: Partially update existing resources.
- **DELETE**: Delete existing resources.

HTTP Status Codes

- **200 OK**: Successful request.
- **201 Created**: Resource successfully created.
- **204 No Content**: Request successful, no response body.

- **400 Bad Request**: Invalid or malformed request.
- **401 Unauthorized**: Authentication is required or has failed.
- **403 Forbidden**: Client lacks permission.
- **404 Not Found**: Resource not found.
- **409 Conflict**: Duplicate entry or conflict in request.
- **500 Internal Server Error**: General server-side error.

Middleware

Middleware are functions that sit between request and response cycles, performing tasks like authentication checks, logging, error handling, and request parsing. They are reusable and executed sequentially.

CORS (Cross-Origin Resource Sharing)

CORS enables controlled access to resources across different origins (domains). It helps prevent unauthorized cross-domain data sharing, thus enhancing application security. CORS headers tell the browser which origins are permitted.

Rate Limiting

Rate limiting restricts the number of requests a client can make within a certain timeframe, helping prevent abuse, ensuring fair usage, and protecting server resources from overload. Common in public APIs and login routes.

Environment Variables (ENV)

Environment variables securely store sensitive configuration data, such as database credentials, API keys, and other settings. They provide flexibility by allowing different configurations for development, testing, and production environments without exposing sensitive data in code.

Components

Components are reusable UI building blocks encapsulating logic, styling, and state. Commonly used in frameworks like React, they facilitate modular and maintainable application development. Components can be functional or class-based.

State

State refers to data managed within an application or component that can change over time, influencing the UI dynamically based on user interactions or other events. Managed using `useState`, Redux, or other libraries.

Context

Context provides a mechanism to share state globally across components in an application without prop drilling, simplifying state management and reducing complexity. `useContext` hook is commonly used in React.

Linting

Linting involves analyzing source code to detect syntax errors, enforce coding standards, maintain consistency, and improve overall code quality. Tools like ESLint and Prettier are commonly used for linting JavaScript applications.

Additional Concepts for Beginners

What is a Database?

A database is an organized collection of data. In web development, we commonly use relational databases (like PostgreSQL, MySQL) or NoSQL databases (like MongoDB) to store and retrieve data efficiently.

What is CRUD?

CRUD stands for Create, Read, Update, and Delete — the four basic operations performed on database records.

What is a Route?

A route defines the path and HTTP method to respond to client requests in the backend. For example, `GET /books` fetches all books.

What is MVC (Model-View-Controller)?

A software design pattern that separates logic:

- **Model** handles data.
- **View** is the UI (mostly in frontend).
- **Controller** processes input and returns responses.

What is JSON?

JSON (JavaScript Object Notation) is a lightweight data-interchange format used to exchange data between client and server.

What is Asynchronous Programming?

Asynchronous programming allows non-blocking code execution. In JavaScript, async/await and promises are used to handle operations like API calls and database queries without freezing the app.

What is a Promise?

A Promise represents a value that will be available in the future, either resolved (success) or rejected (error).

What is a Package.json?

This file holds metadata about a Node.js project, including scripts, dependencies, and versioning. It's used by NPM to manage the project.

What is a Build Process?

The build process refers to compiling, bundling, and optimizing code for production. Tools like Webpack, Babel, and Vite help with this.

What is Deployment?

Deployment is the process of making your web application accessible on the internet. Common platforms include Vercel, Heroku, Netlify, and AWS.

Version Control with Git

Git is a version control system that helps track changes in code. GitHub, GitLab, and Bitbucket are platforms to collaborate and manage repositories.