What is Node JS ?

What is framework ?

Framework

- It provides a set of tools, libraries, and functionalities that you can leverage to develop your software without having to write everything from scratch.

Express

Express

- Node.js provides a low-level API for creating web servers
- But Express.js simplifies this process by abstracting away many of the common tasks involved in setting up a web server, such as routing, middleware management, and handling HTTP requests and responses.

Installation and Simple Program

Routes

Routing

- Routing is about showing your app how to respond to different URLs.
- It involves associating HTTP methods with specific functions to handle requests. This helps organize and control the flow of your web application.

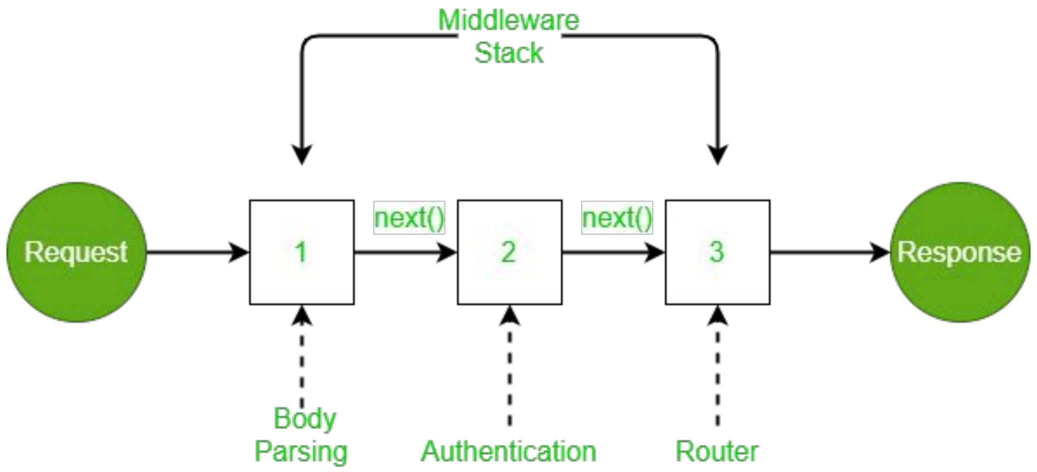
Demo Program

Get Request

Post request

Delete request

Middleware


Middleware Stack
Request
1
next()
2
next()
3
Response
Body Parsing
Authentication
Router

Middleware

- Middleware in Express.js acts like a series of security checkpoints or processing stations that a request from a client has to go through before reaching the final destination (the server) and getting a response back.

Middleware

- Logging
- Security Checks
- Processing
- Error Handling

Middleware

- Application-level middleware: Bound to the entire application
- Router-level middleware: Associated with specific routes
- Error-handling middleware: Handles errors during the request-response cycle.
- Built-in middleware: Provided by Express
- Third-party middleware: Developed by external packages

Demo Program

Cookies

Cookies

- Cookies are small pieces of data that a web server stores on a user's computer or device. They are used to remember information about the user, such as their preferences, login status, or items in their shopping cart.

Why cookies are used

- Personalization
- Session Management
- Shopping Carts
- Analytics

Cookie-parser

- The cookie-parser middleware is a third-party library that simplifies handling cookies in Express.js applications.

Url encoding

Url-Encoding

- When you submit a form on a website, the data you enter is sent to the server. This data needs to be in a format that the server can easily read and understand.
- URL-encoded bodies are one such format. They are commonly used when the form's method attribute is set to POST.

Session Storage

Session Storage

- Session storage refers to the mechanism of storing user session data on the server-side. Each session is identified by a unique session ID, which is sent to the client in a cookie

Difference Between Session Storage and Cookies

- Storage Location:
    - Session Storage: Data is stored on the server. The client only holds a session ID in a cookie.
    - Cookies: Data is stored on the client's browser.

Difference Between Session Storage and Cookies

- Security:
  - Session Storage: More secure as sensitive data is not exposed to the client.
  - Cookies: Less secure since data is stored on the client and can be accessed or manipulated.

Difference Between Session Storage and Cookies

- Capacity:
  - Session Storage: Can store larger amounts of data since it is stored on the server.
  - Cookies: Limited to around 4KB of data.

Difference Between Session Storage and Cookies

- Expiration:
  - Session Storage: Sessions can expire after a specified time or when the browser is closed.
  - Cookies: Can have a persistent lifespan defined by the expires or max-age attribute.

Difference Between Session Storage and Cookies

- Session Expiry:
    - Session Timeout: Sessions can be configured to expire after a certain period of inactivity. For example, a session might be set to expire if there has been no activity for 30 minutes.

Thank you.