Express JS

Express JS is a node.js web application framework, designed for building single-page, multi-page, and hybrid web applications. Several popular Node.Js frameworks are built on Express. Ex: Nestjs, feathers etc.

Express.js is a popular web application framework for Node.js, which is a JavaScript runtime environment. It simplifies the process of building web applications and APIs by providing a set of features and tools to handle routing, middleware, and other common web development tasks.

In simpler terms, think of Express.js as a toolbox that helps you create web applications using JavaScript. It provides pre-built functions and structures to make your job easier when building websites or web services. With Express.js, you can easily define routes (URL paths) for handling different requests from users, manage data sent and received from the server, and integrate with databases or other external services.

Imagine you're building a house. Express.js is like having a set of blueprints and tools ready for you to use. It saves you time and effort by providing a framework so you can focus on building your application without having to reinvent the wheel.

Why learn Express ?

Express makes the development of Node application very easy and it is very simple to use. It provides a simple and efficient way to build web applications and APIs using JavaScript. It helps Node to handling routes, requests, and responses, making it easier for you to create robust and scalable applications. As it is very flexible, lightweight and easy to learn and contains a ton of middleware option making it an excellent choice to learn and use Express in your application.

Express Advantages:

- **Simplicity and Minimalism:** Express JS has very simple design, that makes it easy to learn and use. With its simple structure you can quickly set up a server, define routes, and handle HTTP requests which makes it an excellent choice for building web applications efficiently.
- **Flexibility and Customization:** Express JS is a flexible framework that allows you to structure the application based on your preferences. It does have a strict application architecture so you can organize your code according to your preference.
- **Middleware Ecosystem:** Express JS has a large numbers of middleware that can be easily integrated into applications. Middleware functions increases the functionality of Express by allowing you to handle various tasks such as authentication, logging, and error handling.
- **Scalability:** Express JS is designed to be lightweight and scalable, which makes it suitable for building both small projects and large-scale applications. It is asynchronous and has event-driven architecture which allows you to handle a large number of requests.
- **Active Community Support:** Express JS has a large active community who contribute to its growth and improvement. Because of them the framework is regularly updated and well-documented.

Installing Express

The Express.js package is available on npm package repository. Let us install express package locally in an application folder named Express.

```
D:\express> npm init
D:\express> npm install express --save
```

The above command saves the installation locally in the node_modules directory and creates a directory express inside node_modules.

Request & Response

Express application uses a callback function whose parameters are request and response objects.

```
app.get('/', function (req, res) {

 // --

 })
```

Request: It's like a message sent from a client (e.g., web browser) to a server, asking for something (e.g., a webpage).

Response: After the server processes the request, it sends back a message to the client with the requested information (e.g., the webpage) or an indication of success or failure.

You can print req and res objects which provide a lot of information related to HTTP request and response including cookies, sessions, URL, etc.

Hello world Example

We are going to create basic Express app which starts a server and listens on port 5000 for connection. This app responds with Hello World! for requests to the homepage. For every other path, it will respond with a 404 Not Found.

ExpressJs Routing

Basic Routing We have seen a basic application which serves HTTP request for the homepage. Routing refers to determining how an application responds to a client request to a particular endpoint, which is a URI (or path) and a specific HTTP request method (GET, POST, and so on).