Express.js

What is express.js? How we use?

What is Express.js?

Express.js is a minimal and flexible Node.js web application framework that

provides a robust set of features to develop web and mobile applications. It is

designed for building web applications and APIs.

Why Do We Need Express.js?

Simplicity: Simplifies the process of building server-side applications.

Flexibility: Allows the creation of complex applications with minimal effort.

Middleware: Facilitates the use of middleware for handling requests and responses.

Routing: Provides powerful routing mechanisms.

Scalability: Helps in building scalable and maintainable applications.

Setting Up Express.js

Install Node.js and npm: Ensure Node.js and npm are installed on your system.

Initialize a Project:

```bash
mkdir myapp
cd myapp
npm init -y
```

Install Express:

```bash
npm install express
```

Basic Express.js Application

Creating a Simple Server

Create app.js:

```
  javascript
Const express = require('express');
Const app = express();
Const port = 3000;

app.get('/', (req, res) => {
              res.send('Hello, World!');
});
app.listen(port, () => {
              console.log(
Server is running on http://localhost:${port}
);
});
```

Run the server:

```
bash
node app.js
```

Access the server:open a browser and navigate to http://localhost:3000.

Middleware in Express.js

What is Middleware?

Middleware functions are functions that have access to the request object (req), the response object (res), and the next middleware function in the application's request-response cycle.

Example of Middleware:

javascript

```
app.use((req, res, next) => {
                console.log(
${req.method} request for '${req.url}'
);
                next();
});
```

Handling POST Requests Using Body-Parser Middleware

Install Body-Parser:

bash

```
npm install body-parser
```

Use Body-Parser in app.js:

javascript

```
const bodyParser = require('body-parser');

app.use(bodyParser.urlencoded({ extended: true }));

app.post('/submit', (req, res) => {
        const { name, email } = req.body;
        res.send(
Name: ${name}, Email: ${email});
});
```

HTML Form:

html

```html
<form action="/submit" method="post">
  <input type="text" name="name" placeholder="Name">
  <input type="email" name="email" placeholder="Email">
  <button type="submit">Submit</button>
</form>
```

Routing in Express.js

Defining Routes

javascript

```
app.get('/', (req, res) => {
                res.send('Home Page');
});
app.get('/about', (req, res) => {
                res.send('About Page');
});
```

Parameters and Queries

javascript

```
app.get('/user/:id', (req, res) => {
                res.send(
User ID: ${req.params.id}
);
});

app.get('/search', (req, res) => {
                res.send(
Search Query: ${req.query.q}
);
});
```

Error Handling

Basic Error Handling

javascript

```
app.use((err, req, res, next) => {
        console.error(err.stack);
        res.status(500).send('Something went wrong!');
});
```

404 Error Handling

javascript

```
app.use((req, res, next) => {
        res.status(404).send('Sorry, page not found');
});
```

conclusion

Express.js is a powerful framework for building web applications and APIs.
It provides essential features such as routing, middleware, and scalability.
With Express.js, developers can create robust and maintainable server-side applications efficiently.