Express.js Router Module

In Express.js, the Router module provides a powerful mechanism for organizing and structuring your application's routes.

Key Concepts:

- Creating a Router:
  JavaScript

```javascript
const express = require('express');
const router = express.Router();
```

- Defining Routes:
  - You define routes within the router object using the same methods as with the main app object:
    JavaScript

```javascript
router.get('/', (req, res) => {
  // ...
});
router.post('/users', (req, res) => {
  // ...
});
```

- Mounting the Router:
  - You mount the router onto a specific path in your main application:
    JavaScript

```javascript
app.use('/api', router);
```

    - This means that all routes defined within the router will be prefixed with /api.
    - For example, if you have a route router.get('/users'), it will become accessible at /api/users in the main application.

Benefits of Using Routers:

- Improved Code Organization:
  - Routers help you break down your application into smaller, more manageable modules.
  - You can group related routes together (e.g., user routes, product routes) within separate router files.

- Increased Reusability:
    - Routers can be reused in different parts of your application or even in other projects.
- Enhanced Maintainability:
    - Makes your code easier to understand, modify, and debug.

Example:

userRouter.js:

JavaScript

```javascript
const express = require('express');
const router = express.Router();
router.get('/', (req, res) => {
  res.send('Get all users');
});
router.get('/:userId', (req, res) => {
  const userId = req.params.userId;
  res.send(`Get user with ID: ${userId}`);
});
router.post('/', (req, res) => {
  res.send('Create a new user');
});
module.exports = router;
```

app.js:

JavaScript

```javascript
const express = require('express');
const userRouter = require('./userRouter'); // Import the user router
const app = express();
// Mount the user router
app.use('/users', userRouter);
// ... other routes and middleware ...
app.listen(3000, () => {
  console.log('Server listening on port 3000');
});
```

In this example:

- The userRouter defines routes for handling user-related requests (/users, /users/:userId).
- The userRouter is mounted to the /users path in the main application.

By using routers, you can create a more modular and organized structure for your Express.js applications, making them easier to develop, maintain, and scale.