

# Express Middlewares



# What Are Middlewares in Express?

Middlewares in Express are functions that execute during the request-response cycle.

They have access to the request object (`req`), the response object (`res`), and the next function to pass control to the next middleware.

```
function myMiddleware(req, res, next) {  
  console.log('Request URL:', req.url);  
  next();  
}
```



# Types of Middlewares

- **Application-Level Middleware:** Bound to an instance of Express.
- **Router-Level Middleware:** Bound to an instance of `express.Router()`.
- **Error-Handling Middleware:** Handles errors in the application.
- **Built-in Middleware:** Comes with Express (e.g., `express.json()`).
- **Third-Party Middleware:** External libraries like `morgan` or `cors`.

```
// Application-Level Middleware
app.use((req, res, next) => {
  console.log('Time:', Date.now());
  next();
});
```



# Creating and Using Middlewares

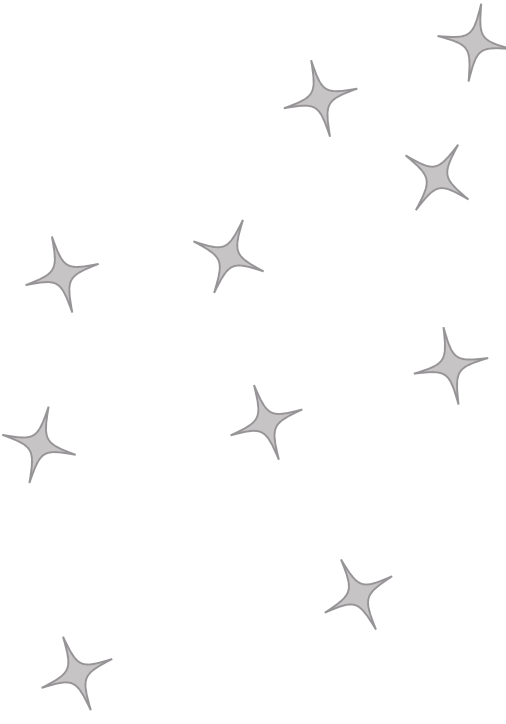
Creating a Middleware: Define a function with req, res, and next.

Custom middlewares can be created and applied globally or to specific routes.

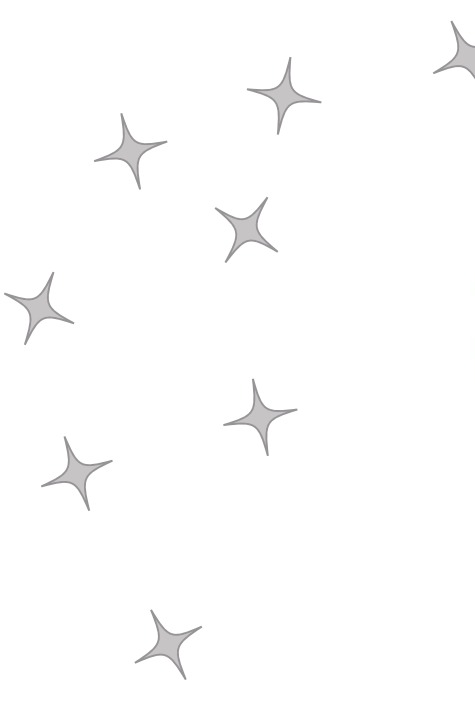
```
function requestLogger(req, res, next) {  
  console.log(`${req.method} ${req.url}`);  
  next();  
}  
  
// Using Middleware  
app.use(requestLogger);
```



# Built-in Middlewares in Express



1. `express.json()`: Parses incoming requests with JSON payloads.
2. `express.urlencoded()`: Parses incoming requests with URL-encoded payloads.
3. `express.static()`: Serves static files.



```
app.use(express.json());  
app.use(express.urlencoded({ extended: true }));
```



# Third-Party Middlewares

- **Morgan:** HTTP request logger.
- **Cors:** Enables Cross-Origin Resource Sharing.
- **Helmet:** Secures Express apps by setting HTTP headers.

```
const morgan = require('morgan');
const cors = require('cors');

app.use(morgan('dev'));
app.use(cors());
```





# Error-Handling Middlewares

Error-handling middlewares catch and handle errors in your application. They are defined with four arguments: `err`, `req`, `res`, and `next`.

```
function errorHandler(err, req, res, next) {  
  console.error(err.stack);  
  res.status(500).send('Something broke!');  
}  
  
app.use(errorHandler);
```

Error-handling middlewares ensure that errors are caught and handled gracefully.

