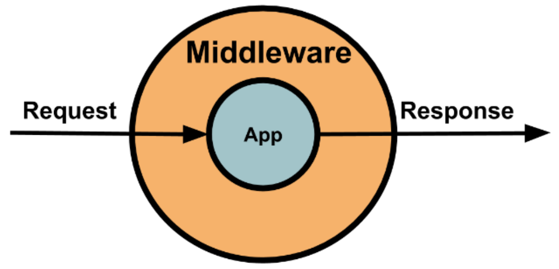
Middle ware

#Node JS Notes

Middle ware

What is Middleware

- Middleware is a middle layer that is called between request and response. When the request is called middleware is called and it called before it sends response.
- The next middleware function is commonly denoted by a variable named next.
- Or
- 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. These functions are used to modify req and res objects for tasks like parsing request bodies, adding response headers, etc.



Why use middleware?

- On most websites, any time you interact with the browser, it requests information from a web server.
- The web server then sends a response back to the browser which it then converts and shows it to you as text, image, video, etc.

Middleware function can perform the following tasks.

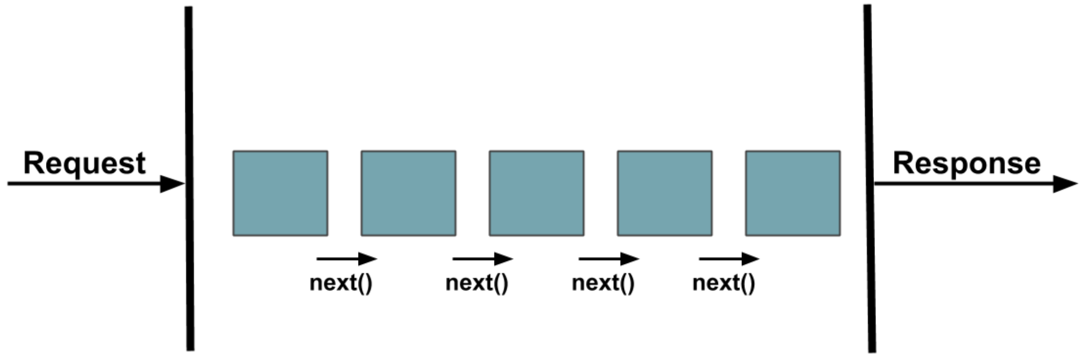
- Can execute any code.
- Can make changes to the request and the response objects.
- Can end the request-response cycle.
- Can call the next middleware function in the stack.

Express.js middleware

- **Express.js** is a routing and Middleware framework for handling the different routing of the webpage and it works between the request and response cycle.

What is next()?

- A middleware function typically receives three arguments: req, res and next . next is the next function which will be executed after the current one.



References

- https://expressjs.com/en/guide/writing-middleware.html
- https://expressjs.com/en/guide/using-middleware.html

Types of middlewares

1. Application level middleware
2. Router-level middleware
3. Error handling middleware
4. Built in middleware
5. Third Party middleware

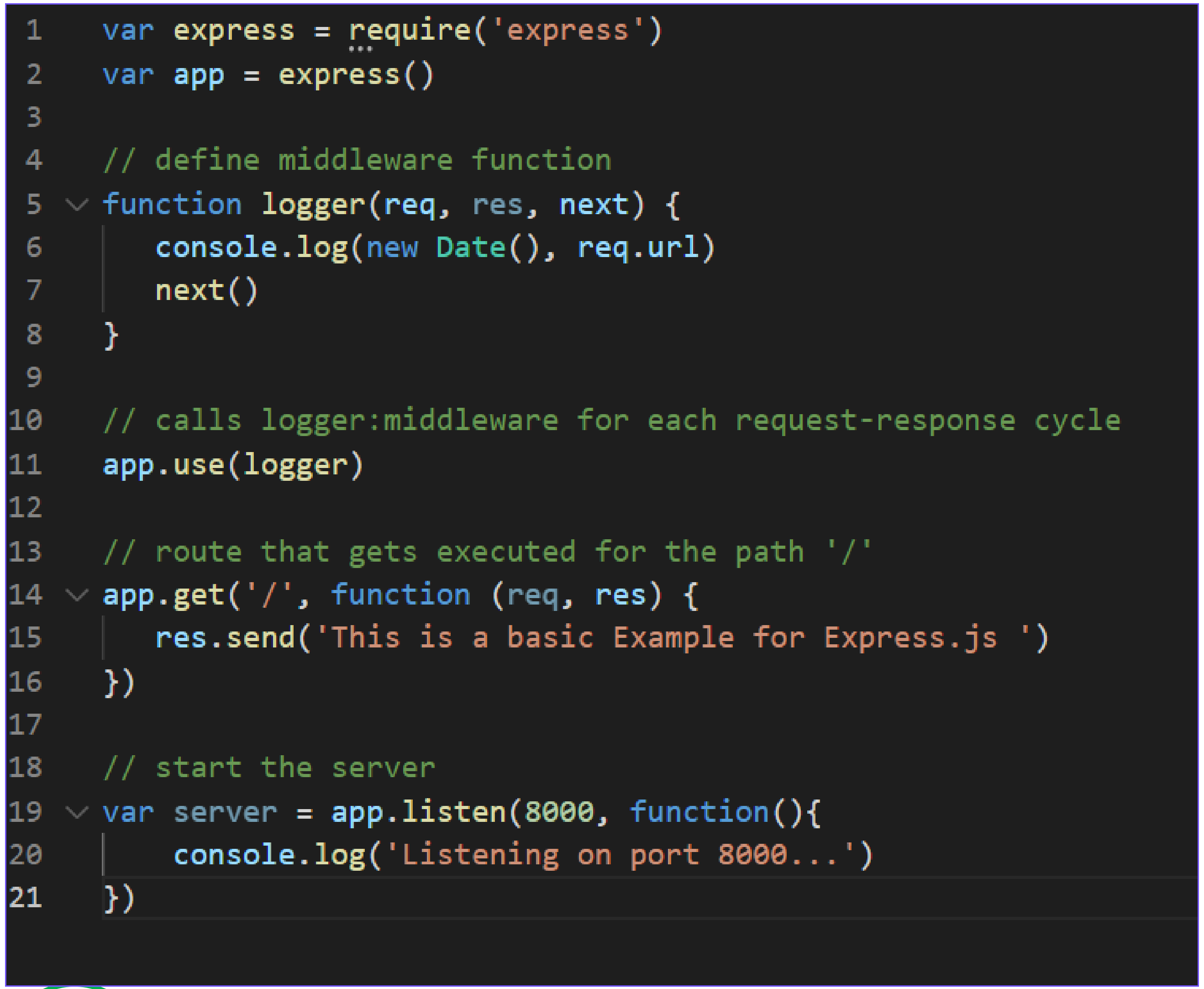
Application Level Middleware

Application level middleware

- We use app.use() or app.METHOD() to bind the app instance to the application middleware.
- For instance, in the following code, we have created a middleware that serves no particular URL. The middleware runs for every request.

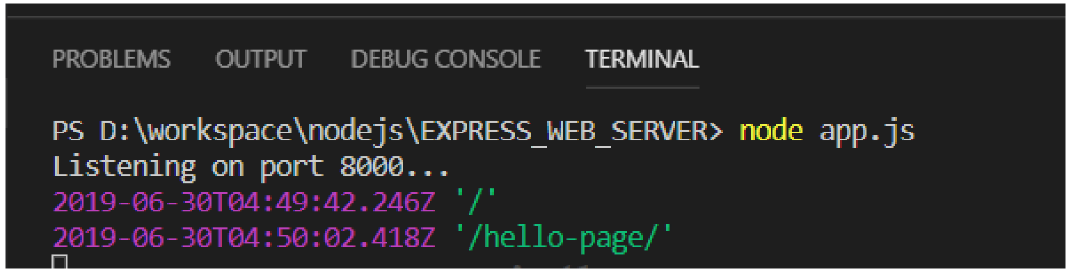
```
var app = express()

app.use(function (req, res, next) {
  console.log('Hello from the middleware !')
  next()
})
```

Example



```
var express = require('express')
var app = express()

// define middleware function
function logger(req, res, next) {
    console.log(new Date(), req.url)
    next()
}

// calls logger:middleware for each request-response cycle
app.use(logger)

// route that gets executed for the path '/'
app.get('/', function (req, res) {
    res.send('This is a basic Example for Express.js ')
})

// start the server
var server = app.listen(8000, function(){
     console.log('Listening on port 8000...')
})
```

- http://localhost:8000/
- http://localhost:8000/hello-page/



```
PS D:\workspace\nodejs\EXPRESS_WEB_SERVER> node app.js
Listening on port 8000...
2019-06-30T04:49:42.246Z '/'
2019-06-30T04:50:02.418Z '/hello-page/'
```

```
// define middleware function
function logger(req, res, next) {
  console.log(`Logged  ${req.url}  ${req.method} -- ${new Date()}`)
  next()
}

// calls logger:middleware for each request-response cycle
app.use(logger)
```

Cont.

- logger is the function name, req is the HTTP request object, res is the Node Response Object and next is the next function in request-response cycle.
- You can access all the properties and methods of request object req.
- Similarly, you can access all the properties and methods of response object res.
- Calling next() function inside the middleware function is optional. If you use next() statement, the execution continues with the next middleware function in request-response cycle. If you do not call next() function, the execution for the given request stops here.

following code, this middleware handles request for the route /data

```
app.use('/data', function (req, res, next) {
  console.log('This middleware handles the data route')
  next()
})
```

Router Level

Router level middleware

- Router level middlewares are similar to application-level middlewares except that they are bound to an instance of express.Router().
- The below middleware executes for every request to the router.

Example

- Load router-level middleware by using the router.use() and router.METHOD() functions.

```
var app = express()
var router = express.Router();

router.use(function (req, res, next) {
  console.log('This is a Router middleware');
  next();
})
```

Error Handling

Error handling middlewares

- Express JS comes with default error handling params, define error-handling middleware functions in the same way as other middleware functions, except error-handling functions have four arguments instead of three:
- To call an error-handling middleware, we need to send the error object in the next() method

Example

```
app.use(function (err, req, res, next) {
  console.error("Error found !");
  res.status(500).send('Something very wrong happened !')
})
```

Built-in middlewares

Built-in middlewares

- Express also has some built-in middlewares such as -

- express.static() - serves static assets such as images, HTML files etc.
- express.json() - parses the incoming request object with JSON payloads.
- express.urlencoded() - parses incoming requests with URL-encoded payloads

- List :
- https://expressjs.com/en/resources/middleware.html

- Express provides you with middleware to deal with the (incoming) data (object) in the body of the request.

- a. express.json() is a method inbuilt in express to recognize the incoming Request Object as a JSON Object. This method is called as a middleware in your application using the code: app.use(express.json());

- b. express.urlencoded() is a method inbuilt in express to recognize the incoming Request Object as strings or arrays. This method is called as a middleware in your application using the code: app.use(express.urlencoded());

Third party middleware

Third party middleware

- We may use several third party middlewares to add functionality to our Express Apps
- To use third party modules, we need to install them first and then include it in our app
  - Morgan
  - body-parser

morgan

- morgan is a very popular logging middleware. It lets us see the requested data right in the console. To use it, install it first

- npm install morgan

Body Parser

- Body-parser is the Node.js body parsing middleware. It is responsible for parsing the incoming request bodies in a middleware before you handle it.
- Parse incoming request bodies in a middleware before your handlers, available under the req.body property.

```
const bodyparser = require('body-parser')

// Body-parser middleware
app.use(bodyparser.urlencoded({extended:false}))
app.use(bodyparser.json())

Console.log(req.body)
```

Calling middleware using app.use()

- Middleware is called before every request in the web page.

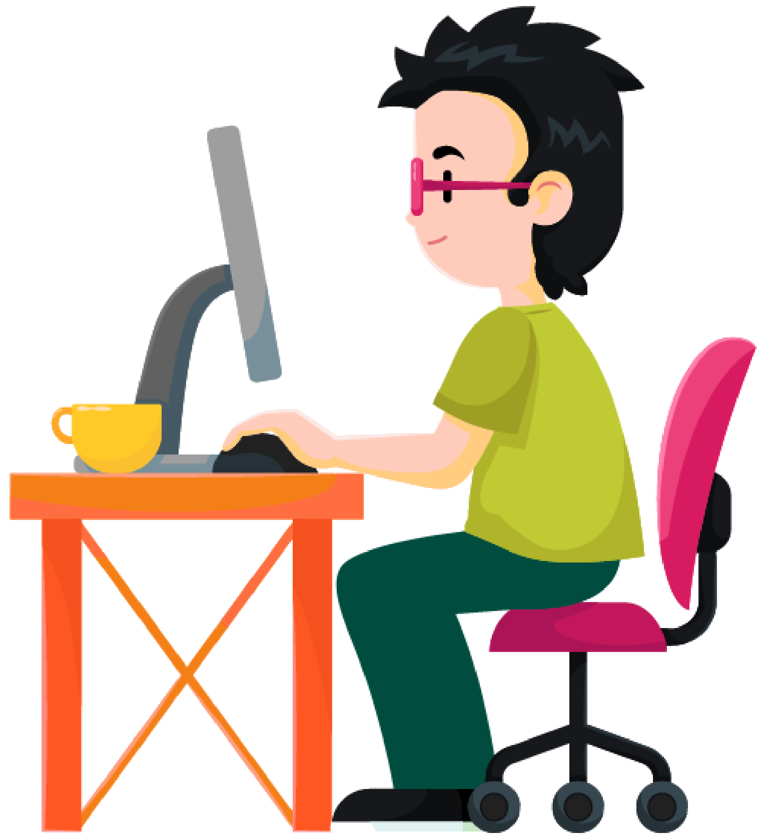
```
var express = require('express');
var app = express();

app.use(function (req, res, next) {
   console.log('Method is: ' + req.method + ' URL is: ' + req.url);
   next();
});

app.get('/', function (req, res) {
   res.send('Hello This is home page');
});

app.listen(3000, function () {
   console.log('App listening on port 3000!');
});
```

Calling for a specific route

- Middleware is called before a /students route. In this way, we call middleware before any specific routes.

```
var express = require('express');
var app = express();

// Middleware get called before the original route and send the request
// This route is only called before /students route
app.use('/students', function (req, res, next) {
   console.log('Method is: ' + req.method + ' URL is: ' + req.url);
   next();
});

// Before the route is called middleware get activated
app.get('/students', function (req, res) {
   res.send('Get all Student Data');
});

app.listen(3000, function () {
   console.log('App listening on port 3000!');
});
```

Get Exclusive Video Tutorials



www.aptutorials.com

https://www.youtube.com/user/Akashtips

Get More Details

www.akashsir.com

If You Liked It !

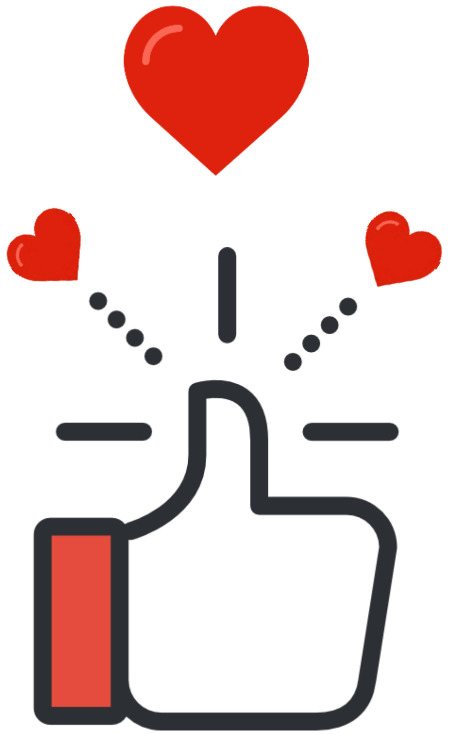
Rating Us Now

Just Dial

https://www.justdial.com/Ahmedabad/Akash-Technolabs-Navrangpura-Bus-Stop-Navrangpura/079PXX79-XX79-170615221520-S5C4_BZDET

Sulekha

https://www.sulekha.com/akash-technolabs-navrangpura-ahmedabad-contact-address/ahmedabad



Connect With Me

Akash Padhiyar
#AkashSir

www.akashsir.com
www.akashtechnolabs.com
www.akashpadhiyar.com
www.aptutorials.com

# Social Info

Akash.padhiyar
Akashpadhiyar
Akash_padhiyar
+91 99786-21654

#Akashpadhiyar
#aptutorials