



# WEB TECHNOLOGIES

## Express JS

---

**Shruthi L**

Department of Computer Science and Engineering.

# WEB TECHNOLOGIES

---

## Express JS MIDDLEWARE

**Shruthi L**

Department of Computer Science and Engineering

# WEB TECHNOLOGIES

## MIDDLEWARE

---



### Express JS

- Express is a framework that does minimal work and gets most of the job done by functions called *middleware*.
- A *middleware* is a function that takes in an HTTP request and response object, plus the next middleware function in the chain.
- Express.js Middleware are different types of functions that are invoked by the Express.js routing layer before the final request handler.
- Middleware appears in the middle between an initial request and final intended route,
- Middleware functions are always invoked in the order in which they are added.

# WEB TECHNOLOGIES

## MIDDLEWARE

---

- Middleware functions are functions that have access to the
  - request object (req),
  - response object (res), and
  - next function in the application's request-response cycle.
- The next function is a function in the Express router which, when invoked, executes the middleware succeeding the current middleware.
- Middleware functions can perform the following tasks:
  - Execute any code.
  - Make changes to the request and the response objects.
  - End the request-response cycle.
  - Call the next middleware in the stack.

# WEB TECHNOLOGIES

## MIDDLEWARE

---

### Express JS

- An Express application can use the following types of middleware:
  - Application-level middleware
  - Router-level middleware
  - Error-handling middleware
  - Built-in middleware
  - Third-party middleware

# WEB TECHNOLOGIES

## MIDDLEWARE

---



### Express JS

#### Application-level middleware

- Bind application-level middleware to an instance of the app object by using the `app.use()` and `app.METHOD()` functions.
- `METHOD` is the HTTP method of the request that the middleware function handles (such as GET, PUT, or POST).

```
var express = require('express')
var app = express()
app.use(function (req, res, next) {
  console.log('Time:', Date.now())
  next()
})
```

```
app.use('/user/:id', function (req, res, next) {
  console.log('Request Type:', req.method)
  next()
})
```

# WEB TECHNOLOGIES

## MIDDLEWARE

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

HTTP method for which the middleware function applies.

Path (route) for which the middleware function applies.

The middleware function.

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

Callback argument to the middleware function, called "next" by convention.

```
app.listen(3000);
```

HTTP **response** argument to the middleware function, called "res" by convention.

HTTP **request** argument to the middleware function, called "req" by convention.

# WEB TECHNOLOGIES

## MIDDLEWARE

---

- Order of Middleware Calls
- Middleware in Express is executed in the order in which they are written/included in your file, given that the route
- Use of middleware before and after route handler; also how a route handler can be used as a middleware itself.

```
var express = require('express');
var app = express();
//First middleware before response is sent
app.use(function(req, res, next){
  console.log("Start"); next();
});
//Route handler
app.get('/', function(req, res, next){
  res.send("Middle"); next();
});
app.use('/', function(req, res){
  console.log('End');
});
app.listen(3000);
```



# WEB TECHNOLOGIES

## MIDDLEWARE

---



### Router-level middleware

- Router-level middleware works in the same way as application-level middleware, except it is bound to an instance of `express.Router()`.
- Load router-level middleware by using the `router.use()` and `router.METHOD()` functions.

```
var express = require('express')
var app = express()
var router = express.Router()
router.use(function (req, res, next) {
  console.log('Time:', Date.now())
  next()
})
```

# WEB TECHNOLOGIES

## MIDDLEWARE

---

### Error handling middleware

- Defined in same way as other middleware functions, except with four arguments-(err, req, res, next):  

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

### Built in middleware

- `express.static` serves static assets such as HTML files, images, and so on.

### Third party middleware(Form data)

- `body-parser`

# WEB TECHNOLOGIES

## MIDDLEWARE

---



### Built-in middleware

- The ***express.static*** generator function generates one such middleware function.
- This middleware responds to a request by trying to match the request URL with a file under a directory specified by the parameter to the generator function.
- If a file exists,
  - It returns the contents of the file as the response; else
  - It chains to the next middleware function.
- The middleware is *mounted on the application using the ***application's use()**** method.

# WEB TECHNOLOGIES

## MIDDLEWARE

---

### Express JS

- The middleware generator takes the parameter static to indicate that this is the directory where all the static files reside.

```
const express = require('express');

const app = express();
app.use(express.static('static'));

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



**THANK YOU**

---

**Shruthi L**

Department of Computer Science Engineering