

# Lecture-15

Coding Blocks - Kartik Mathur

**Middleware**s

## Class Agenda

01

Middlewares

02

Module-Exports

03

Todo app

04

-

05

-

06

-

07

-

# Middlewares

1



# Middlewares

An Express application can use the following types of middleware:

- Application-level middleware
- Built-in middleware
- Third-party middleware
- Error-handling middleware
- Router-level middleware: We will see this later on

# Middlewares

1. It is a generic middleware it runs for every request that comes to the express server.

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

2. It's another path specific middleware

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

3. Path fixed middlewares

```
app.get('/user/:id', (req, res) => {  
  res.send('hello, user!')  
})
```

# Application level middlewares

# Built in middleware

Express has the following built-in middleware functions:

- [express.static](#) serves static assets such as HTML files, images, and so on.
- [express.json](#) parses incoming requests with JSON payloads.
- [express.urlencoded](#) parses incoming requests with URL-encoded payloads.

Add them inside `app.use( express.json() );`

## Middlewares

# Error handling middleware

```
app.use((err, req, res, next) => {  
  console.error(err)  
  res.send('Something not right')  
})
```

## Middleware

# Third party middleware

```
// load the body-parser middleware  
npm i body-parser
```

```
// We can also use this instead of express.json() inbuilt middleware.  
app.use(bodyParser.json())
```

## HTTP Requests



# File-Dependency

2



# Dependency Map

```
script.js x lib1.js lib2.js
3_moduleAdvanced > script.js > ...
1 const lib1 = require('./lib1');
2 const lib2 = require('./lib2');
3
4 console.log(lib1);
5 console.log(lib2);
```

The screenshot shows a code editor with three files: script.js, lib1.js, and lib2.js. The code in lib1.js and lib2.js creates a circular dependency. Handwritten notes in the center explain the 'Infinite Loop' and the execution order.

```
lib1.js
1 const lib2=require('./lib2.js');
2
3 console.log("Inside file-1")
4 let a = 10;
5 function add(a,b){
6     return a+b;
7 }
8
9 module.exports={
10     a,
11     add
12 }
```

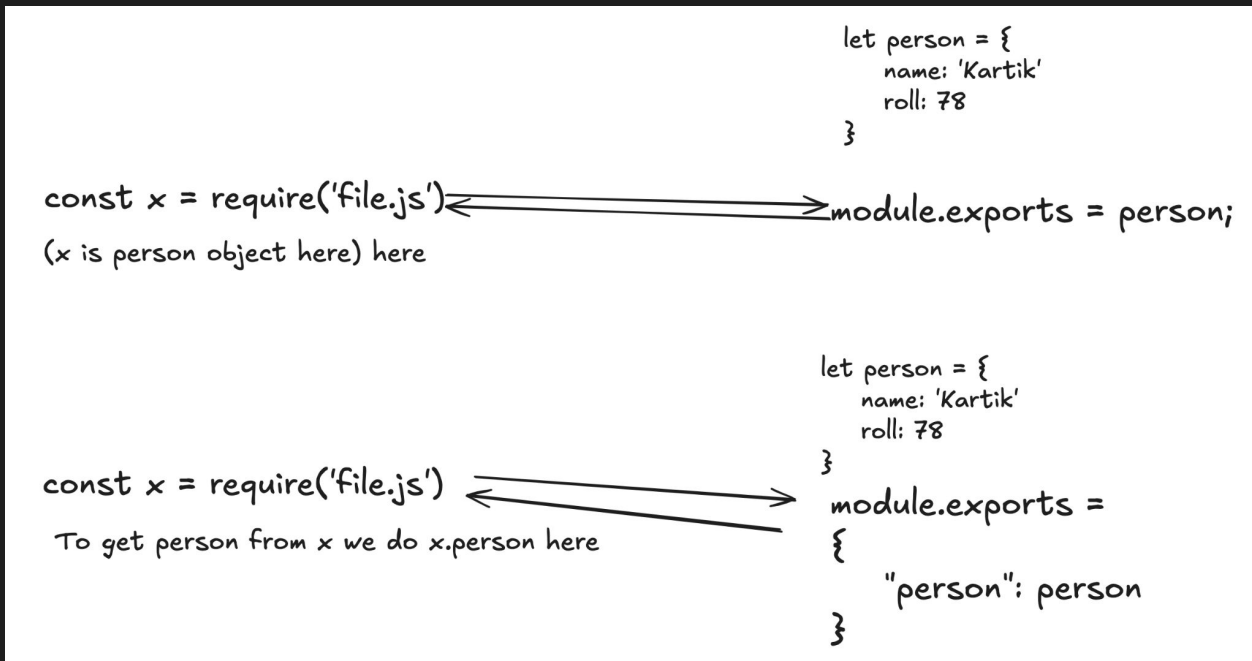
```
lib2.js
1 const lib1 = require('./lib1');
2 console.log("Inside file-2");
3
4 let b = 20;
5 function sub(a, b) {
6     return a - b;
7 }
8
9 module.exports = {
10     b,
11     sub
12 }
```

**Infinite Loop**

(#) Only at first require file gets executed uske baad nahi karega execute

How it works?

# Difference in module.exports



## How it works?

# Building TODO APP

3

