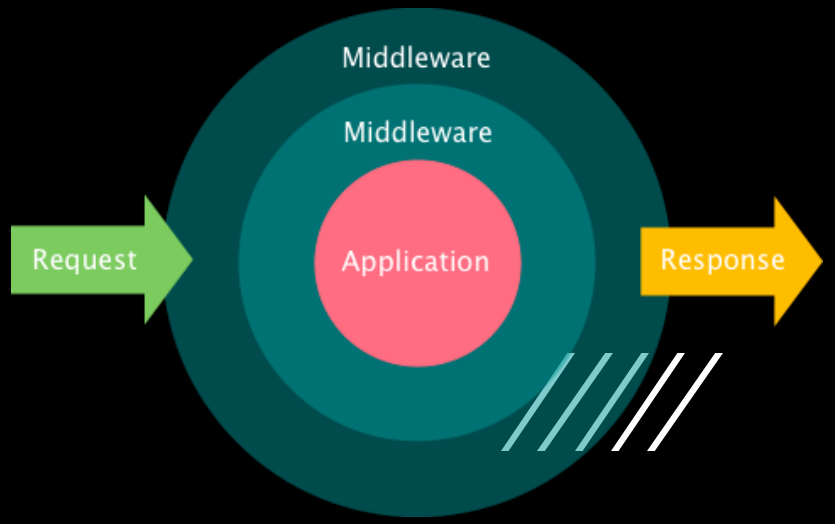
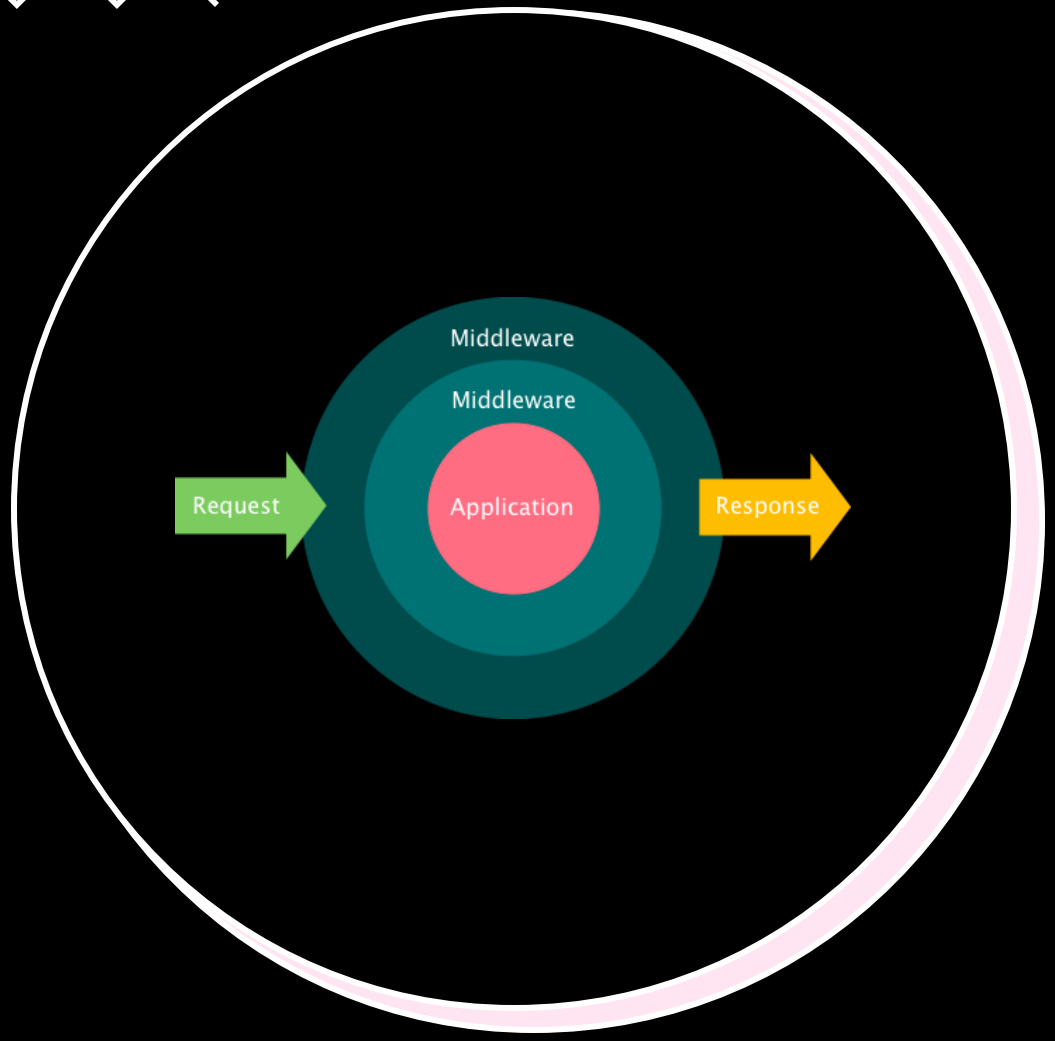
EXPRESS-ADVANCED TOPICS

MIDDLEWARE

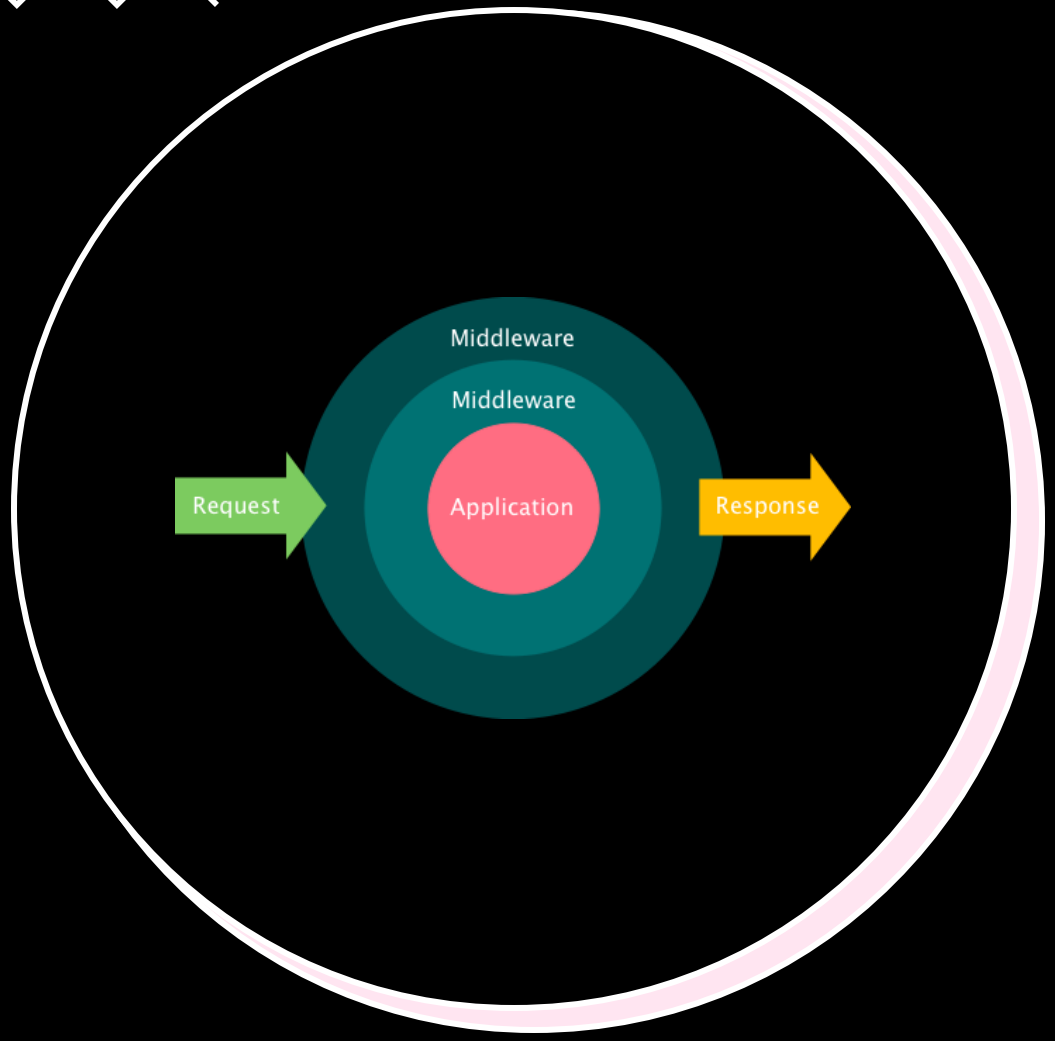


Middleware



A middleware function is a function that takes a request object and either terminates the request/response cycle or passes control to another middleware function

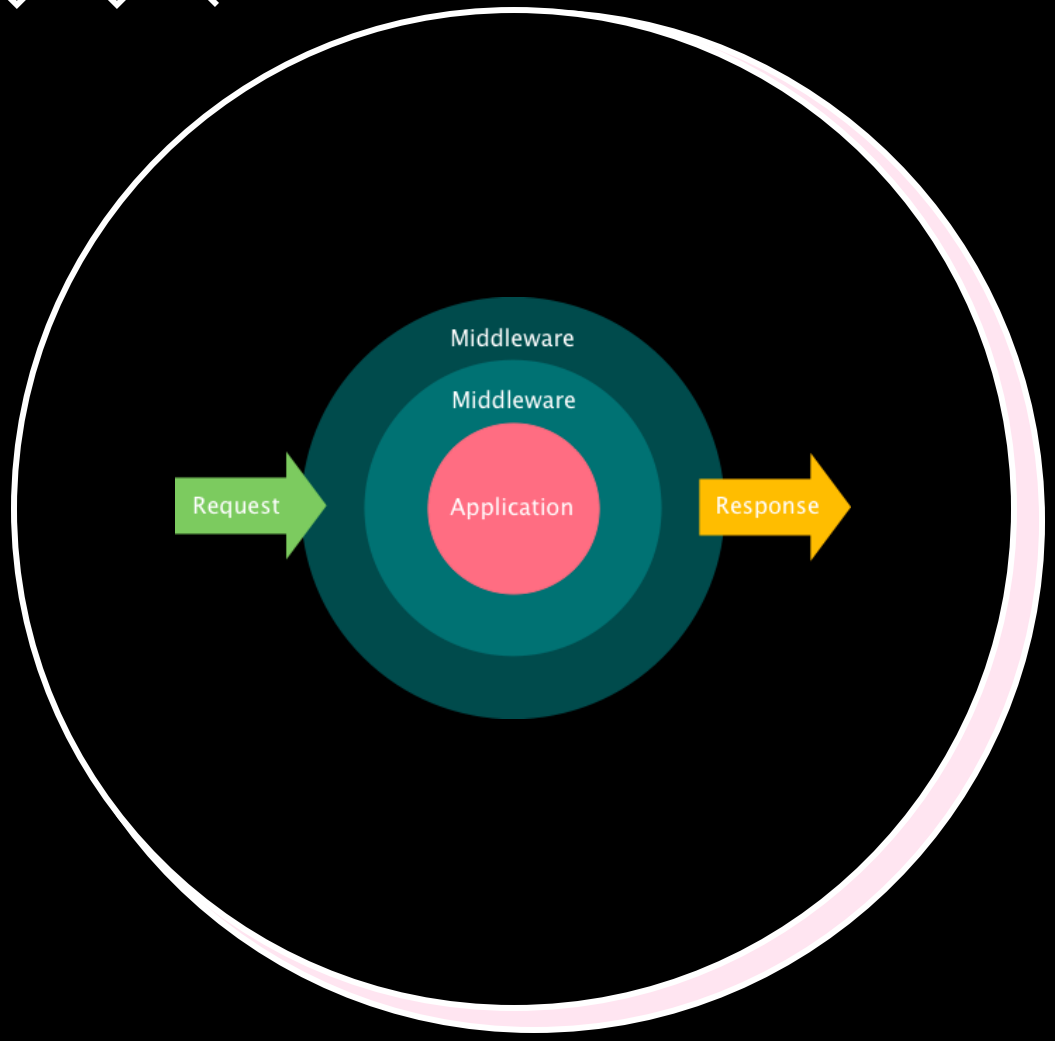
Middleware



Express has a few built-in middleware functions:

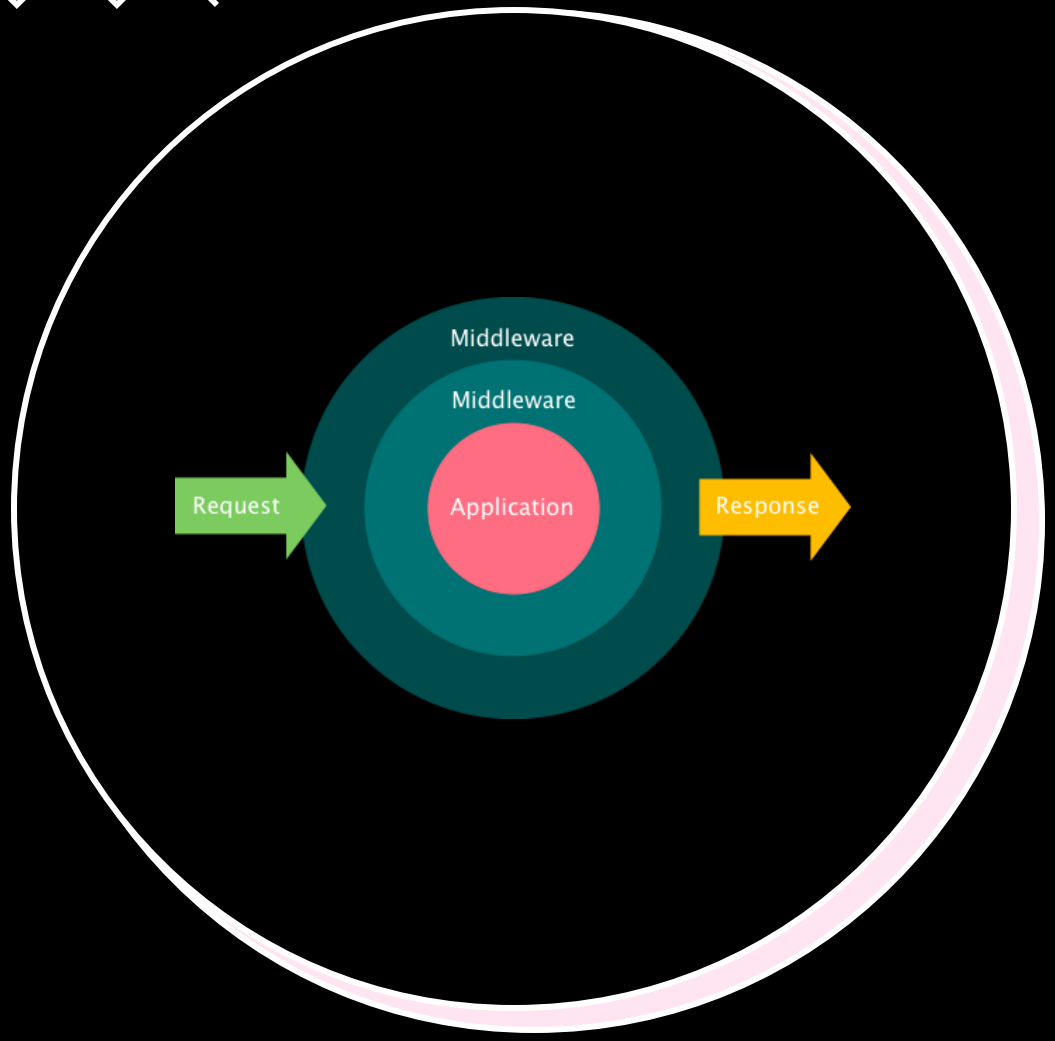
- ❑ json(): to parse the body of requests with a JSON payload
- ❑ urlencoded(): to parse the body of requests with URL-encoded payload
- ❑ static(): to serve static files

Middleware



You can create custom middleware for cross-cutting concerns, such as logging, authentication, etc.

Middleware

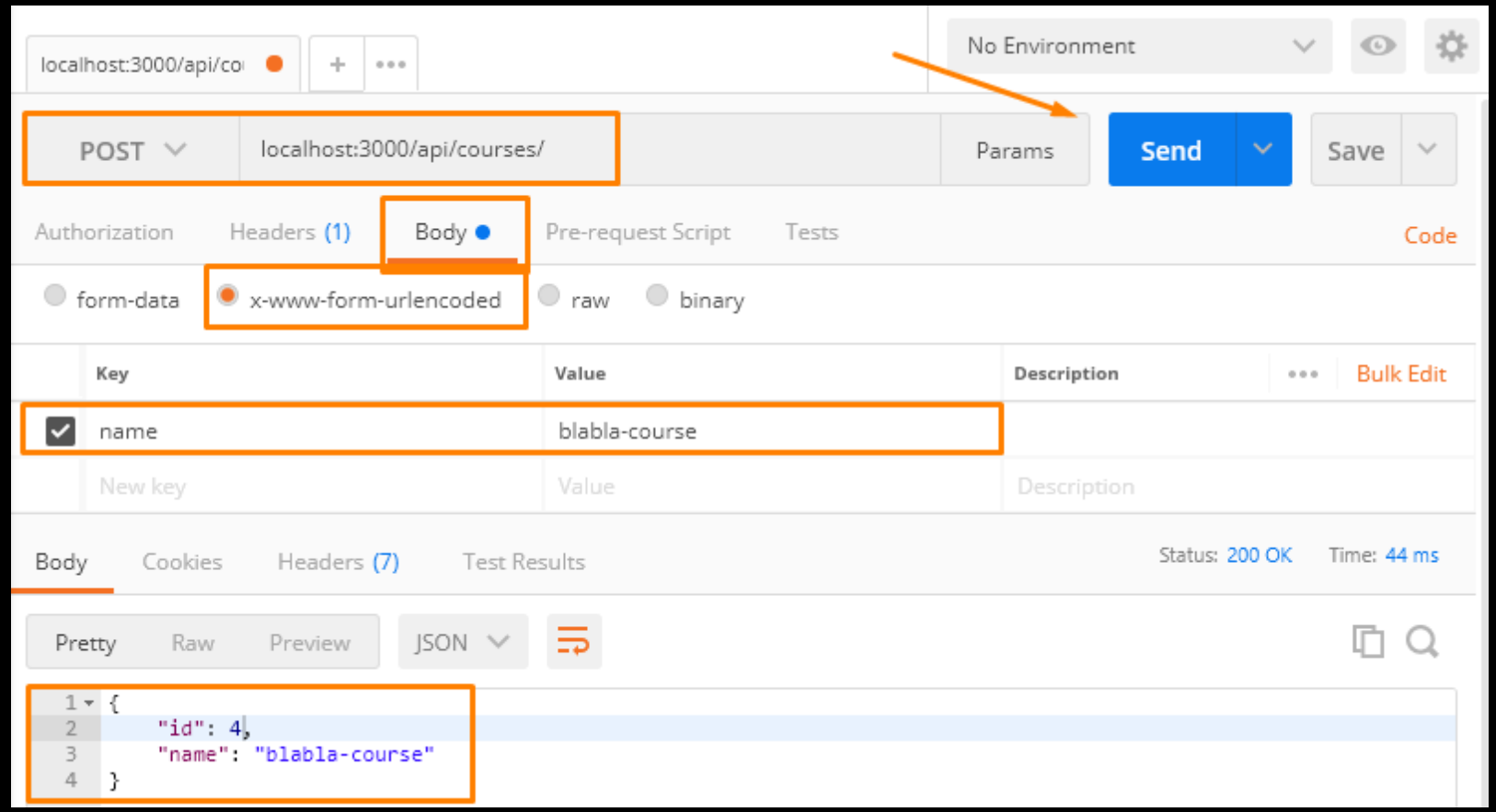



```
// Custom middleware (applied on all routes)

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

```
// Custom middleware (applied on routes starting
 with /api/admin)

app.use('/api/admin', function(req, res, next))
{
    // ...
    next();
}
```

Middleware

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

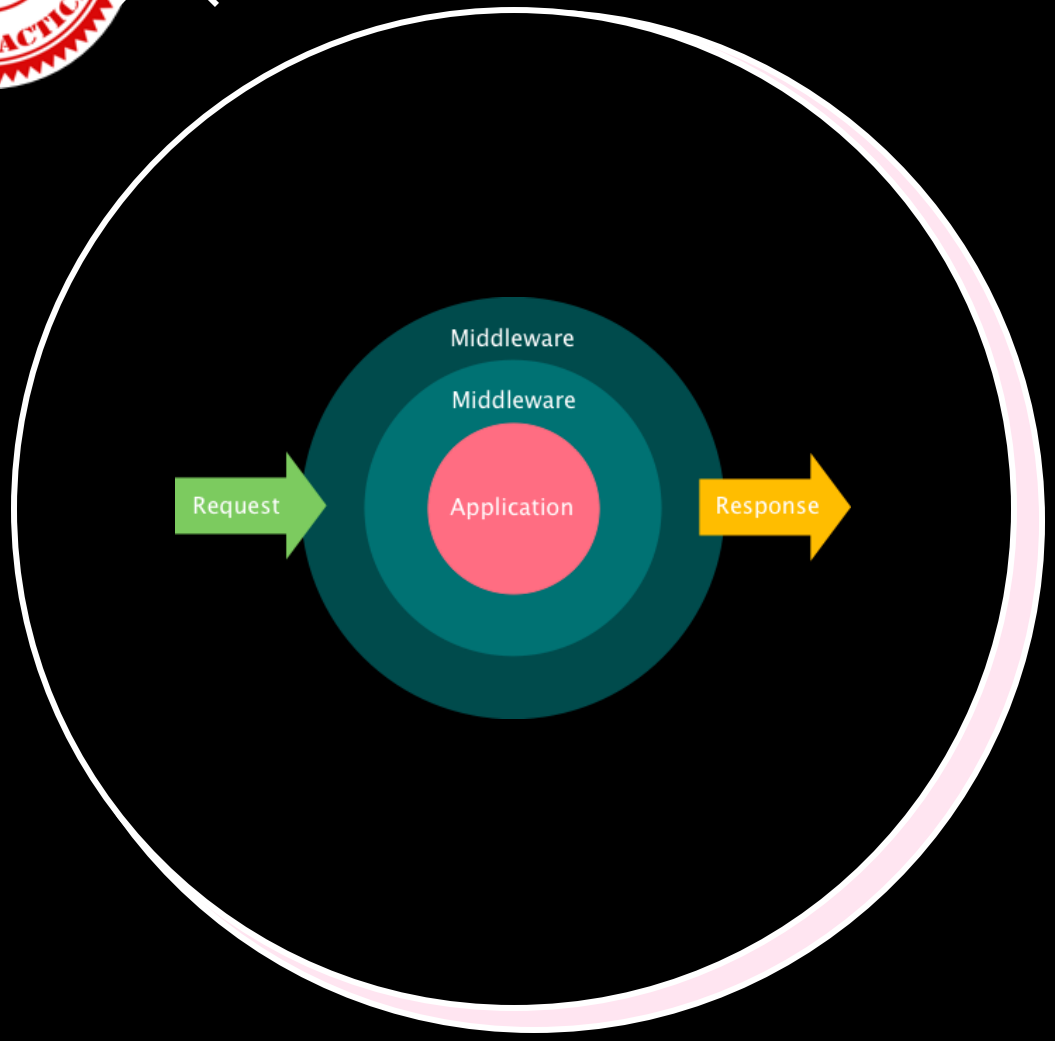


Middleware

```
app.use(express.static('public'));
```

Create folder "public"
Create file readme.txt into folder "public"

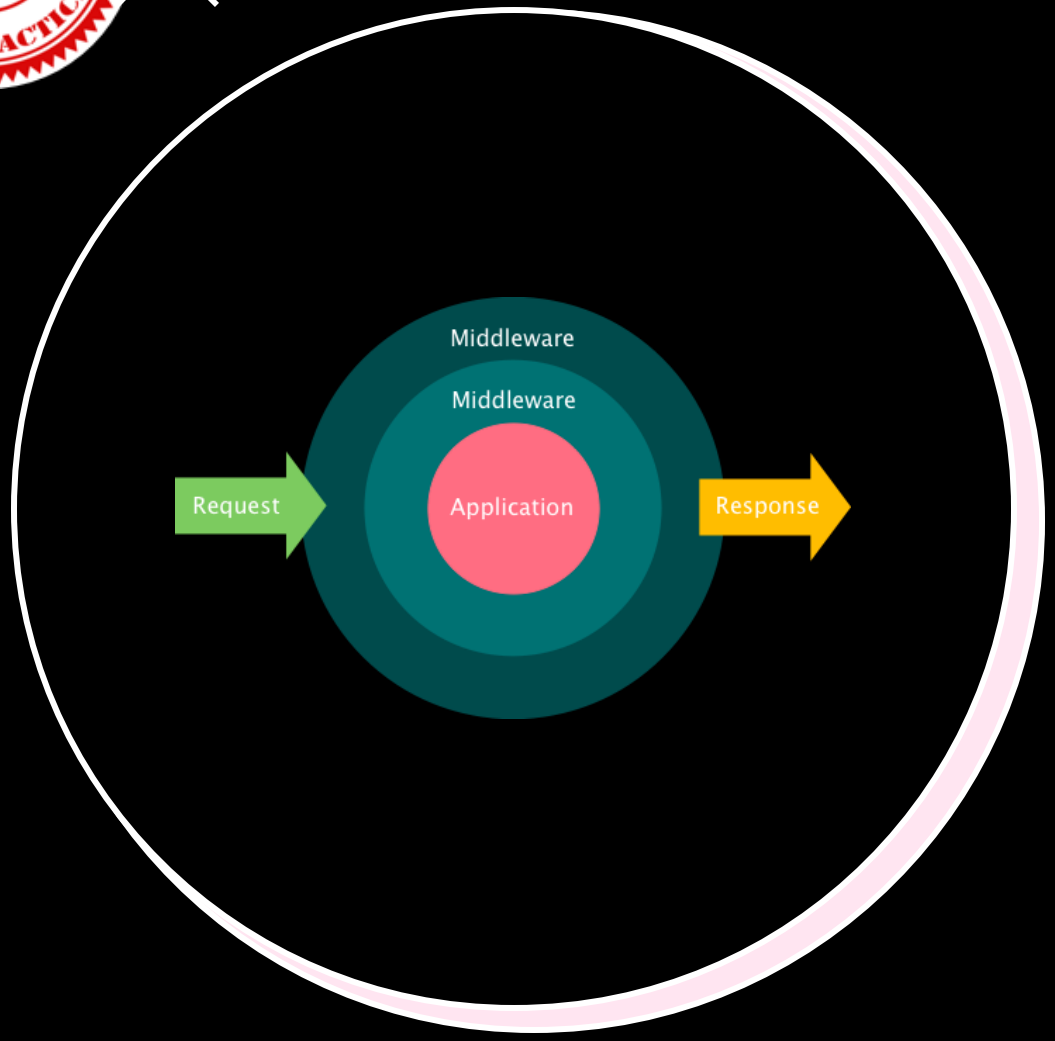
Middleware



Helmet helps you secure your Express apps by setting various HTTP headers.

It's not a silver bullet, but it can help!

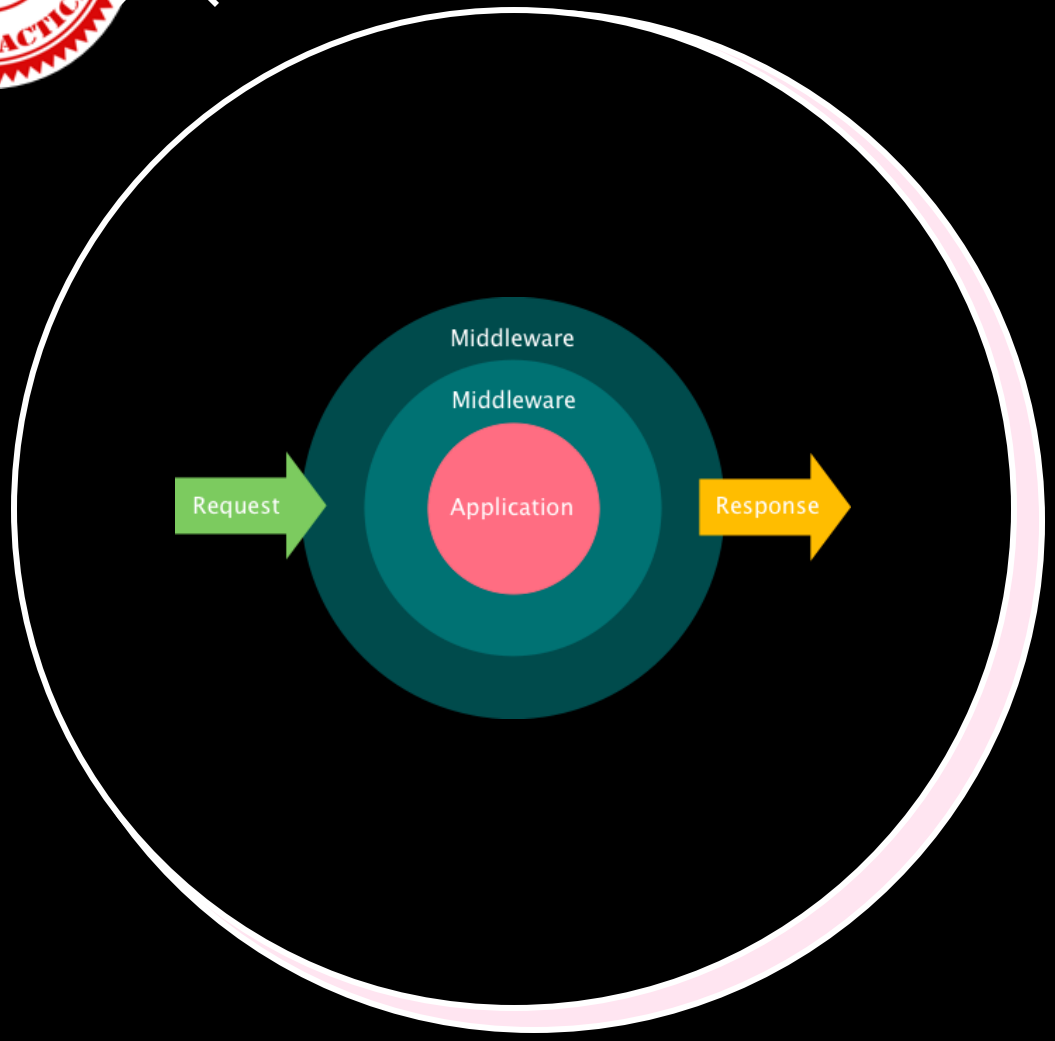
Middleware



Helmet helps you secure your Express apps by setting various HTTP headers.

It's not a silver bullet, but it can help!

Middleware



morgan

HTTP request logger middleware for node.js

```
PROBLEMS    OUTPUT    TERMINAL    DEBUG CONSOLE

Listening on port 3000...
Logging...
GET /api/courses/ 200 79 - 2.860 ms
```

Environments

We can detect the environment in which our Node application is running (development, production, etc) using process.env.NODE_ENV and app.get('env').

EXPRESS-ADVANCED TOPICS

CONFIGURATION

CONFIGURATION

The config package gives us an elegant way to store configuration settings for our applications

PowerShell

```
npm i config
```

CONFIGURATION

- Create folder config
- Create file default.json

```
{
    "name": "My Express App"
}
```

- Create file development.json

```
{
    "name": "My Express App - Development",
    "mail": {
        "host": "dev-mail-server"
    }
}
```

- Create file production.json

```
{
    "name": "My Express App - Production",
    "mail": {
        "host": "prod-mail-server"
    }
}
```

CONFIGURATION

PowerShell

```
set NODE_ENV=development
nodemon index.js
$env:NODE_ENV="production"
nodemon index.js

$env:app_password="12345678"
```

Create file custom-environment-variables.json

```
{
    "mail":{
        "password":"app_password"
    }
}
```

PowerShell

```
nodemon index.js
```

EXPRESS-ADVANCED TOPICS

DEBUGGING

DEBUGGING

We can use the debug package to add debugging information to an application. Prefer this approach to console.log() statements.

PowerShell

```
npm i debug
```

DEBUGGING

index.js

```
const startupDebugger = require('debug')('app:startup');
const dbDebugger = require('debug')('app:db');


 if (app.get('env') === 'development'){
     app.use(morgan('tiny'));
     startupDebugger('Morgan enabled...');
 }

 //DB work...
 dbDebugger('Connected to the Database...');
```

PowerShell

```
$env:DEBUG='app:startup'
$env:DEBUG=''
$env:DEBUG='app:*'
```

EXPRESS-ADVANCED TOPICS

TEMPLATING ENGINES

TEMPLATING ENGINES

PowerShell

```
npm i pug
```

- Create folder views
- Create file index.pug

```
html
  head
    title= title
  body
    h1= message
```

EXPRESS-ADVANCED TOPICS

STRUCTURING EXPRESS APPLICATIONS

STRUCTURING EXPRESS APPLICATIONS

- Create folder routes
- Create file courses.js
- Create file home.js

- Create folder middleware
- Move logger.js to middleware