What are middlewares?

Middlewares are code that runs before / after your request handler.

It's commonly used for things like

1. Analytics
2. Authentication
3. Redirecting the user

src > TS index.ts > ...

```ts
import express from "express";

const app = express();

let requestCount = 0;

app.use(
  function middleware(req, res, next) {
    requestCount++;
    next()
  }
);

app.get("/", (req, res) => {
  res.send("Hello world");
})

app.get("/requestCount", (req, res) => {
  res.json({
    requestCount
  })
})

app.listen(3000);
```

▼ Code

Copy

```ts
import express from "express";

const app = express();

let requestCount = 0;

app.use(
  function middleware(req, res, next) {
    requestCount++;
    next()
  }
```

```
);

app.get("/", (req, res) => {
  res.send("Hello world");
})

app.get("/requestCount", (req, res) => {
  res.json({
    requestCount
  })
})

app.listen(3000);
```

src > TS index.ts > ...

```
1
2   import express from "express";
3   import jwt from "jsonwebtoken";
4
5   const app = express();
6
7   //@ts-ignore
8   async function authMiddleware(req, res, next) {
9     const token = req.headers.authorization.split(" ")[1];
10    const decoded = jwt.verify(token, "secret");
11    if (decoded) {
12      next();
13    } else {
14      res.status(401).send("Unauthorised");
15    }
16  }
17
18  app.get("/", authMiddleware, (req, res) => {
19    res.send("You are logged in");
20  })
21
22  app.listen(3000);
```

▼ Code

Copy

```
import express from "express";
import jwt from "jsonwebtoken";

const app = express();

//@ts-ignore
async function authMiddleware(req, res, next) {
```

```
    const token = req.headers.authorization.split(" ")[1];
    const decoded = jwt.verify(token, "secret");
    if (decoded) {
      next();
    } else {
      res.status(401).send("Unauthorised");
    }
  }

  app.get("/", authMiddleware, (req, res) => {
    res.send("You are logged in");
  })

  app.listen(3000);
```

middlewares + Next

Ref - https://nextjs.org/docs/app/building-your-application/routing/middleware

Middleware allows you to run code before a request is completed.

Then, based on the incoming request, you can modify the response by

1. rewriting
2. redirecting
3. modifying the request or response headers
4. or responding directly.

Use cases

- Authentication and Authorization: Ensure user identity and check session cookies before granting access to specific pages or API routes.
- Logging and Analytics: Capture and analyze request data for insights before processing by the page or API.

Code

Create a request count middleware

- Create an empty NextJS project

```
npx create-next-app
```

Copy

```
→  Projects npx create-next-app
Need to install the following packages:
create-next-app@14.1.4
Ok to proceed? (y) y
✔ What is your project named? … middleware-test
✔ Would you like to use TypeScript? … No / Yes
✔ Would you like to use ESLint? … No / Yes
✔ Would you like to use Tailwind CSS? … No / Yes
✔ Would you like to use `src/` directory? … No / Yes
✔ Would you like to use App Router? (recommended) … No / Yes
✔ Would you like to customize the default import alias (@/*)? … No / Yes
Creating a new Next.js app in /Users/harkiratsingh/Projects/middleware-test.
```

- Create middleware.ts in the root folder

Note: While only one `middleware.ts` file is supported per project, you can still organize your middleware logic modularly. Break out middleware functionalities into separate `.ts` or `.js` files and import them into your main `middleware.ts` file. This allows for cleaner management of route-specific middleware, aggregated in the `middleware.ts` for centralized control. By enforcing a single middleware file, it simplifies configuration, prevents potential conflicts, and optimizes performance by avoiding multiple middleware layers.

- Add code to track the number of requests

```
import { NextResponse } from 'next/server'
import type { NextRequest } from 'next/server'

let requestCount = 0;
export function middleware(request: NextRequest) {
  requestCount++;
  console.log("number of requests is " + requestCount);
  return  NextResponse.next()
}
```

Copy

- Try visiting the website

Code #2

Create a request count middleware to track only requests that start with `/api`

- Add a dummy API route (`api/user/route.ts`)

```
import { NextResponse } from "next/server";

export function GET() {
    return NextResponse.json({
        message: "Hi there"
    })
}
```

Copy

- Update `middleware.ts`

```
import { NextResponse } from 'next/server'
import type { NextRequest } from 'next/server'

let requestCount = 0;
export function middleware(request: NextRequest) {
  requestCount++;
  console.log("number of requests is " + requestCount);
  return  NextResponse.next()
}

// See "Matching Paths" below to learn more
export const config = {
  matcher: '/api/:path*',
}
```

Copy

Selectively running middlewares

```
import { NextResponse } from 'next/server'
import type { NextRequest } from 'next/server'

export function middleware(request: NextRequest) {
  console.log(request.nextUrl.pathname)
  if (request.nextUrl.pathname.startsWith('/admin')) {
    return NextResponse.redirect(new URL('/signin', request.url))
  }

  if (request.nextUrl.pathname.startsWith('/dashboard')) {
    return NextResponse.next()
  }
}
```

Copy

Ref - https://github.com/code100x/cms/blob/main/src/middleware.ts