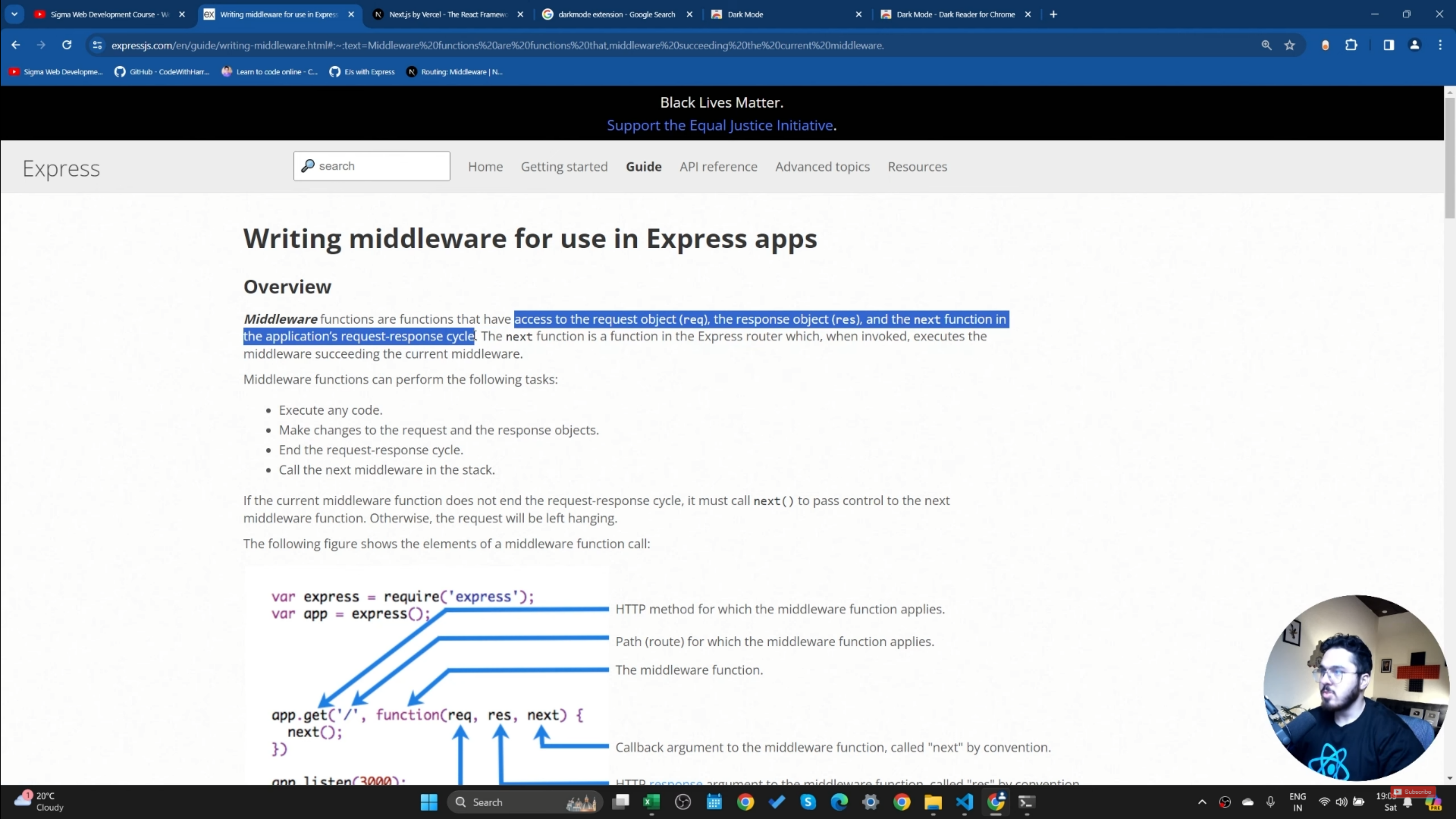

Sigma Web Development Course - W
Writing middleware for use in Expres
Next.js by Vercel - The React Framew
darkmode extension - Google Search
Dark Mode
Dark Mode - Dark Reader for Chrome
expressjs.com/en/guide/writing-middleware.html#:~:text=Middleware%20functions%20are%20functions%20that,middleware%20succeeding%20the%20current%20middleware.
Sigma Web Developme...
GitHub - CodeWithHarr...
Learn to code online - C...
EJs with Express
Routing: Middleware | N...

Black Lives Matter.
Support the Equal Justice Initiative.

Express
search
Home
Getting started
Guide
API reference
Advanced topics
Resources

Writing middleware for use in Express apps

Overview

Middleware functions are functions that have access to the request object (`req`), the response object (`res`), and the `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.

If the current middleware function does not end the request-response cycle, it must call `next()` to pass control to the next middleware function. Otherwise, the request will be left hanging.

The following figure shows the elements of a middleware function call:

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

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

app.listen(3000);
```

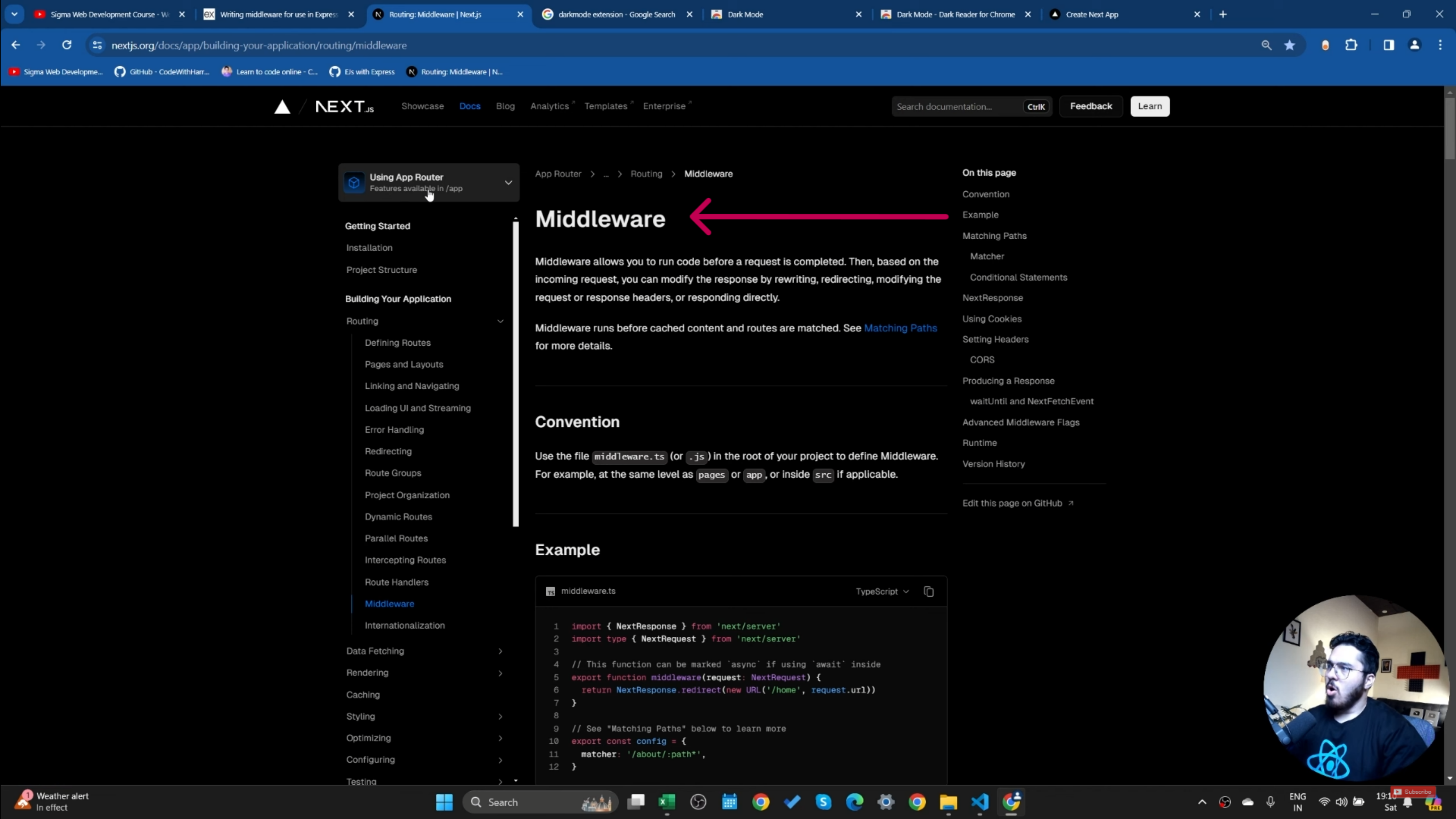
HTTP method for which the middleware function applies.

Path (route) for which the middleware function applies.

The middleware function.

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

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



Middleware

Middleware allows you to run code before a request is completed. Then, based on the incoming request, you can modify the response by rewriting, redirecting, modifying the request or response headers, or responding directly.

Middleware runs before cached content and routes are matched. See Matching Paths for more details.

Convention

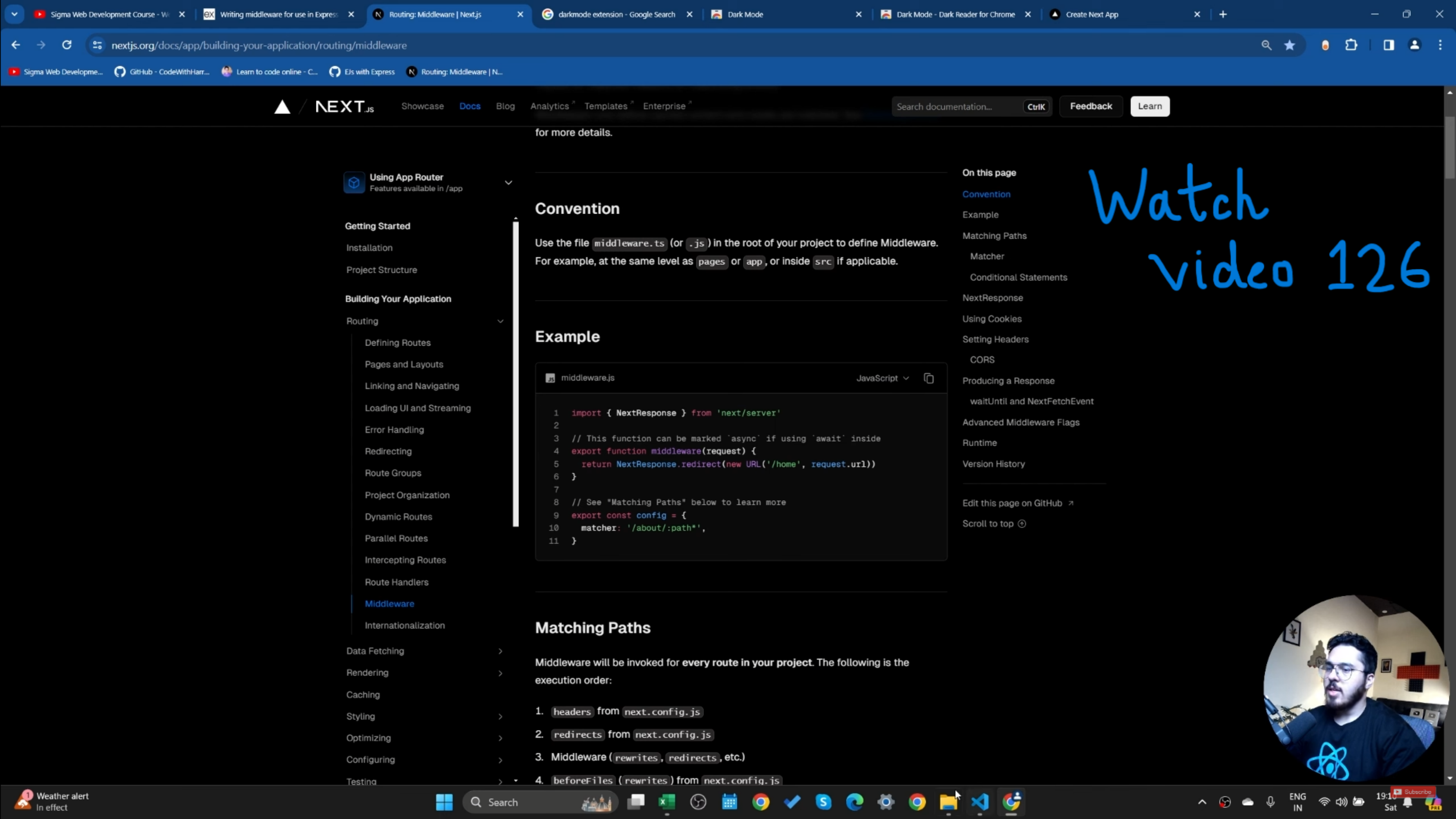
Use the file `middleware.ts` (or `.js`) in the root of your project to define Middleware. For example, at the same level as `pages` or `app`, or inside `src` if applicable.

Example

middleware.ts

TypeScript

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

// This function can be marked `async` if using `await` inside
export function middleware(request: NextRequest) {
  return NextResponse.redirect(new URL('/home', request.url))
}

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



for more details.

Convention

Use the file `middleware.ts` (or `.js`) in the root of your project to define Middleware. For example, at the same level as `pages` or `app`, or inside `src` if applicable.

Example

middleware.js

JavaScript

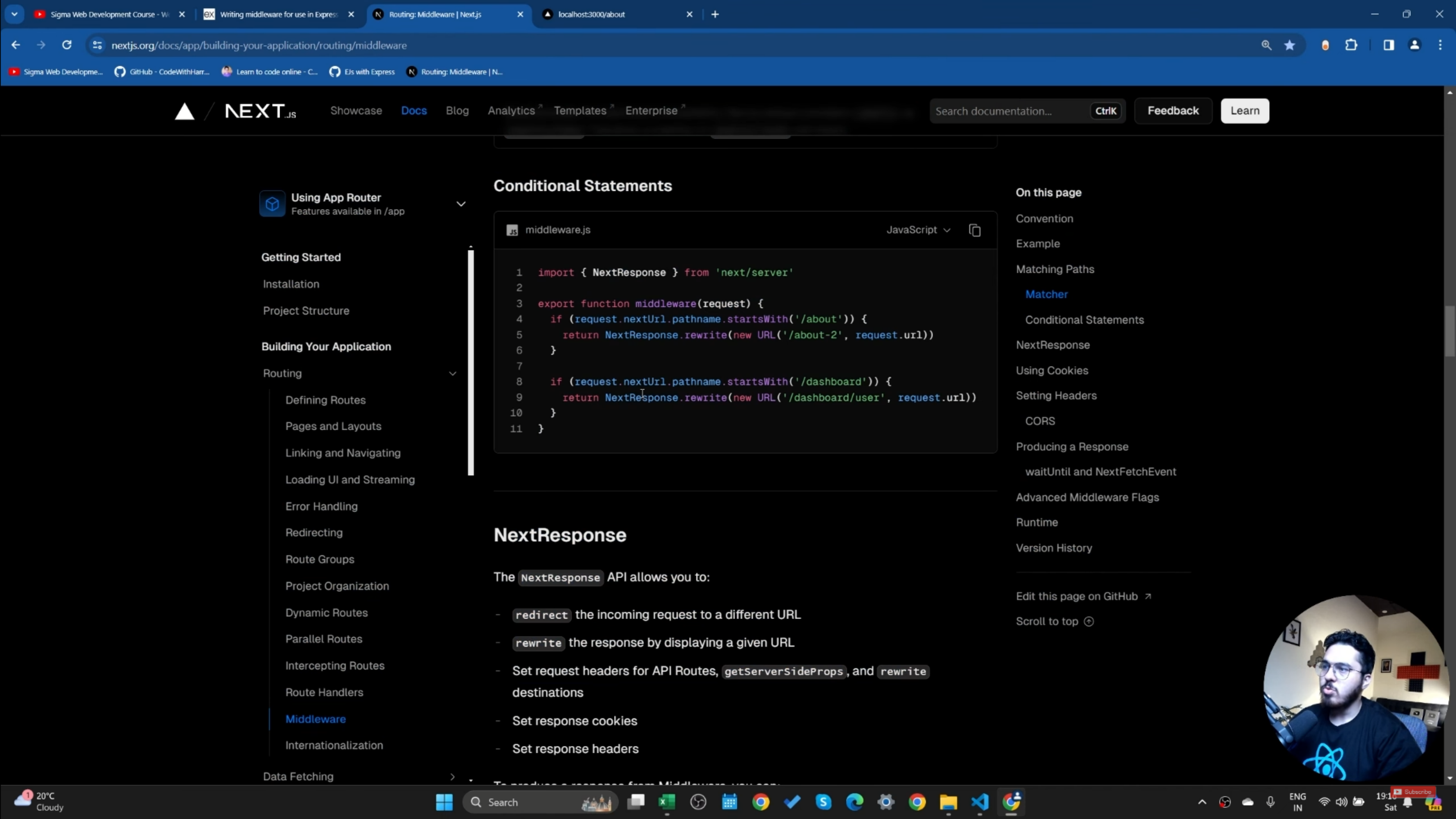
```javascript
import { NextResponse } from 'next/server'

// This function can be marked `async` if using `await` inside
export function middleware(request) {
  return NextResponse.redirect(new URL('/home', request.url))
}

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

Matching Paths

Middleware will be invoked for **every route in your project**. The following is the execution order:

1. `headers` from `next.config.js`
2. `redirects` from `next.config.js`
3. Middleware (`rewrites`, `redirects`, etc.)
4. `beforeFiles` (`rewrites`) from `next.config.js`

Watch
video 126



Conditional Statements

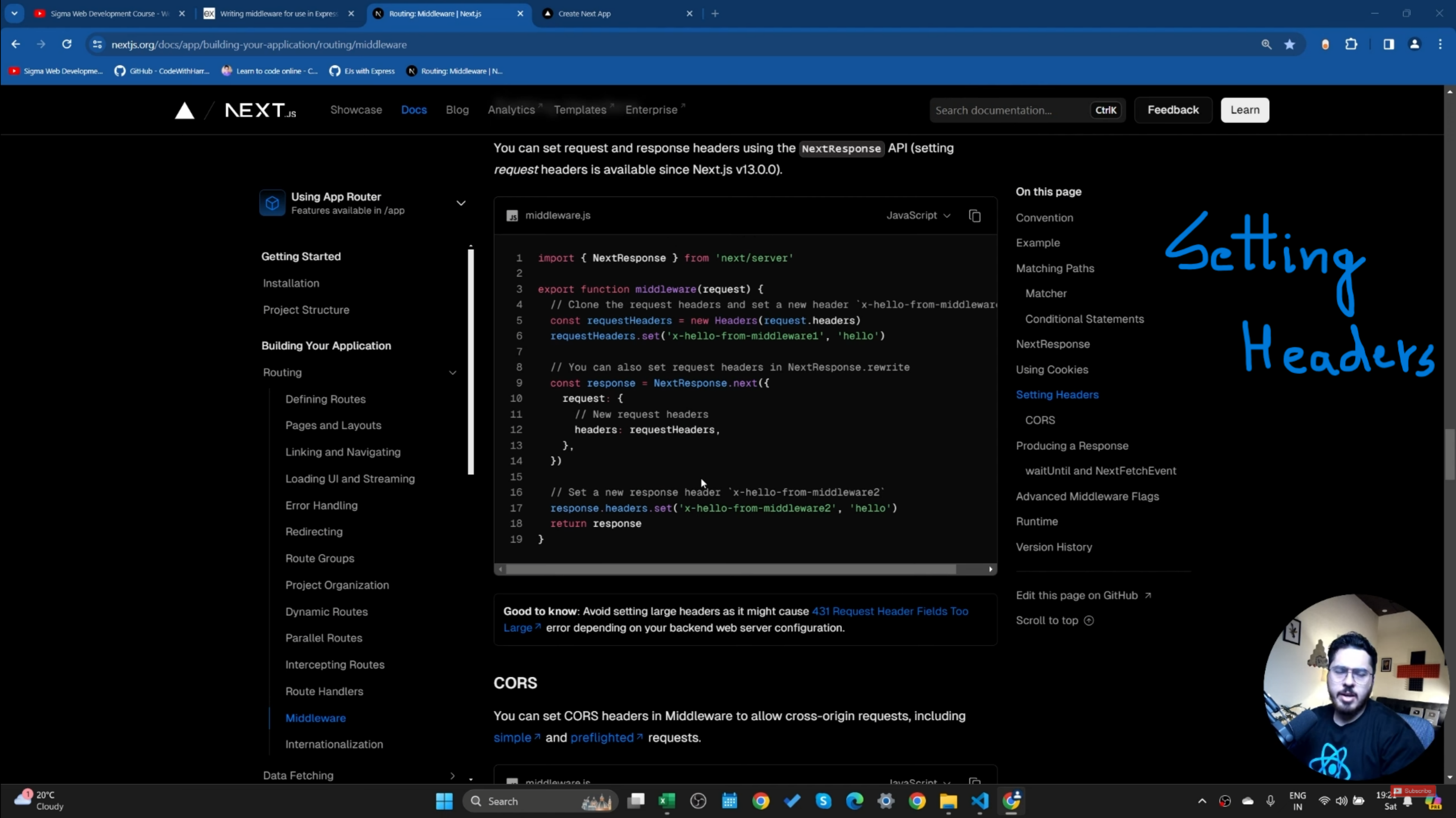
middleware.js

```javascript
import { NextResponse } from 'next/server'

export function middleware(request) {
  if (request.nextUrl.pathname.startsWith('/about')) {
    return NextResponse.rewrite(new URL('/about-2', request.url))
  }

  if (request.nextUrl.pathname.startsWith('/dashboard')) {
    return NextResponse.rewrite(new URL('/dashboard/user', request.url))
  }
}
```

NextResponse

The `NextResponse` API allows you to:

- `redirect` the incoming request to a different URL
- `rewrite` the response by displaying a given URL
- Set request headers for API Routes, `getServerSideProps`, and `rewrite` destinations
- Set response cookies
- Set response headers



You can set request and response headers using the `NextResponse` API (setting *request* headers is available since Next.js v13.0.0).

middleware.js

JavaScript

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

export function middleware(request) {
  // Clone the request headers and set a new header `x-hello-from-middlewar
  const requestHeaders = new Headers(request.headers)
  requestHeaders.set('x-hello-from-middleware1', 'hello')

  // You can also set request headers in NextResponse.rewrite
  const response = NextResponse.next({
    request: {
      // New request headers
      headers: requestHeaders,
    },
  })

  // Set a new response header `x-hello-from-middleware2`
  response.headers.set('x-hello-from-middleware2', 'hello')
  return response
}
```

Good to know: Avoid setting large headers as it might cause 431 Request Header Fields Too Large error depending on your backend web server configuration.

CORS

You can set CORS headers in Middleware to allow cross-origin requests, including simple and preflighted requests.

middleware.js

JavaScript

Setting Headers