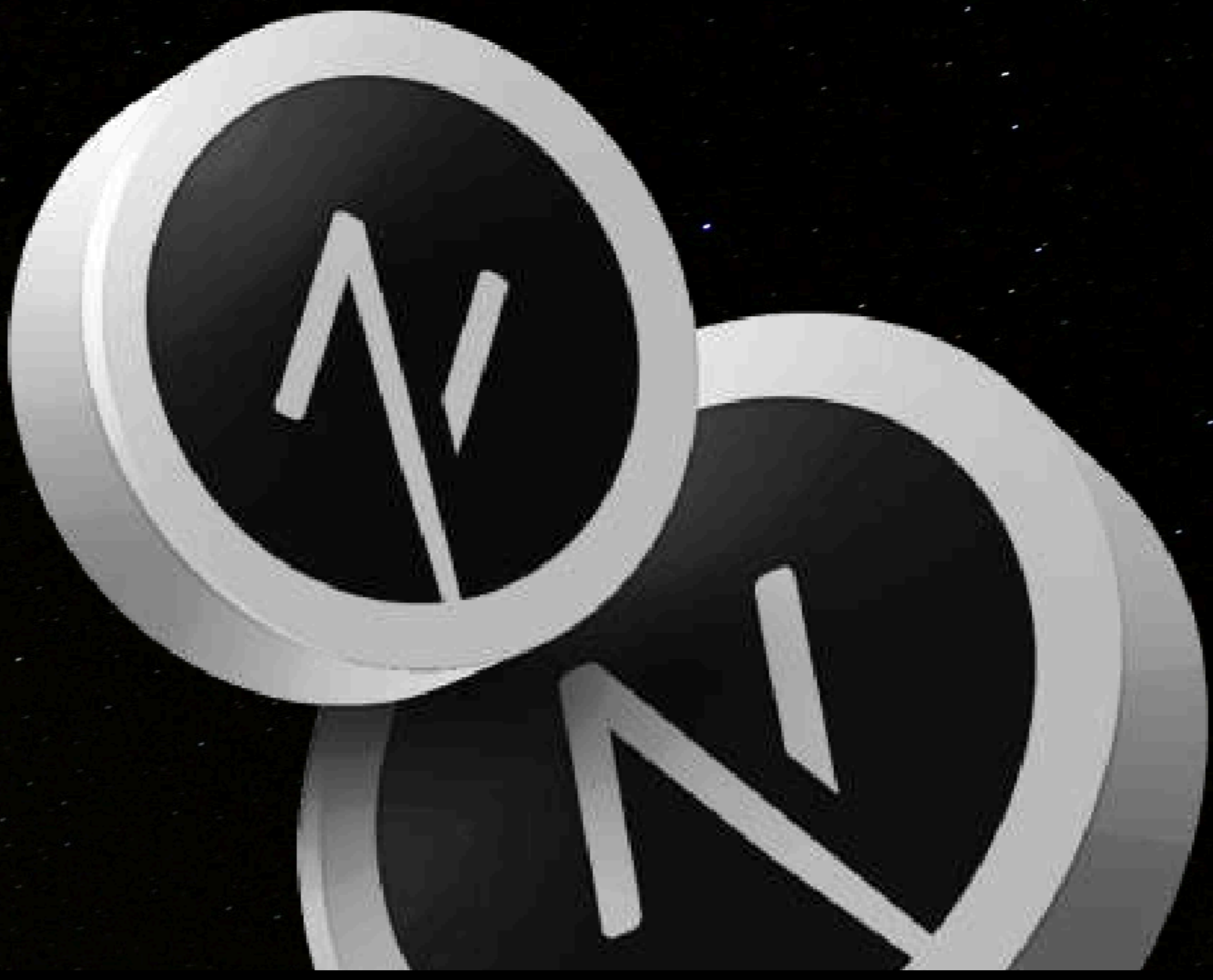
FREEL
Zohaib Asghar

shadowctrl.me

Middleware.js

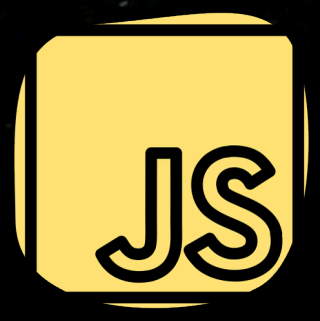
In Next.js



FREELANCER

What Is?

Your App's Powerful Gatekeeper



Middleware is like the superhero sidekick of Next.js, swooping in before a request wraps up! It hands you the reins to control routing, authentication, and even give those requests and responses a little makeover. Talk about power!

@shadowctrl
Freelance Developer

FREELANCER

Authenticate

🔒 Securing Your Next.js App

```
// Your code goes here

export function middleware(request) {
const token = request.cookies.get('auth_token')
if (!token) return NextResponse.redirect('/login')
}
```

Essential for authentication, functioning like a bouncer that protects routes, removes unauthorized users, and manages access, all while keeping your code organized.

@shadowctrl
Freelance Developer

FREELANCER

Performance

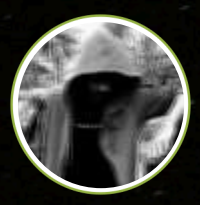
Performance and Routing Optimization

```
// Your code goes here

export function middleware(request) {
// Geolocation routing example
if (request.geo.country === 'US') {
    return NextResponse.rewrite('/us-version')
 }
}
```

Performance Techniques:

- Implement geolocation-based routing
- Add custom headers
- Rewrite or redirect requests
- Cache control and optimization strategies



@shadowctrl
Freelance Developer

FREELANCER

Patterns

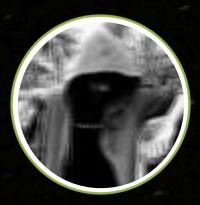
Advanced Middleware Patterns

```
// Your code goes here
export function middleware(request) {
const isAdmin = checkAdminStatus(request)
const isProtectedRoute = request.nextUrl.pathname.startsWith('/admin')

  if (isProtectedRoute && !isAdmin) {
    return NextResponse.redirect('/unauthorized')
 }
}
```

Advanced Strategies:

- Chaining multiple middleware functions
- Dynamic route protection
- Logging and monitoring requests
- Complex conditional routing



@shadowctrl
Freelance Developer

FREELANCER

Pitfalls

Best Practices and Common Pitfalls

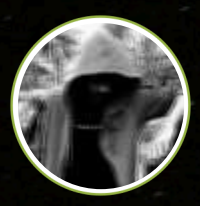
Middleware Wisdom:

- Keep middleware lightweight
- Avoid complex logic in middleware
- Use for global concerns, not page-specific logic
- Understand performance implications
- Test thoroughly across different scenarios

Common Mistakes to Avoid:

- Overcomplicating middleware functions
- Blocking critical request paths
- Neglecting error handling
- Ignoring performance overhead

Pro Tip: Always measure and profile your middleware's impact!



@shadowctrl
Freelance Developer

Did you find it Useful?

Leave a comment!

FOLLOW

Zohaib Asghar

Like Comment Repost