jwt

August 17, 2026

jwt

Kind: Function

Source: `src/middleware/jwt/jwt.ts`

Part of: Middleware

JWT Auth Middleware for Hono.

`jwt` creates Hono middleware that reads a JWT from the request, verifies it with the configured options, and stores the decoded payload in the request context. Routes protected by this middleware can read the payload from `c.get('jwtPayload')`.

Signature

```
function jwt(options: { secret: SignatureKey cookie?: | string | { key: string; secret?: string
```

Parameters

Name	Type
`options`	`{ secret: SignatureKey cookie?:

Returns: `MiddlewareHandler`

Diagram

```
graph LR
  Request[Incoming request] --> Middleware[jwt middleware]
  Middleware --> Token[Read JWT]
  Token --> Verify[Verify signature and claims]
  Verify -->|Valid| Context[Store jwtPayload in context]
  Context --> Handler[Route handler]
  Verify -->|Invalid or missing| Unauthorized[Unauthorized response]
```

Usage

```
import { Hono } from 'hono'
import { jwt } from 'hono/jwt'

const app = new Hono()

app.use(
  '/api/*',
  jwt({
    secret: 'my-secret-key',
  })
)

app.get('/api/profile', (c) => {
  const payload = c.get('jwtPayload')

  return c.json({
    userId: payload.sub,
  })
})

export default app
```

AI Coding Instructions

- Register `jwt` before route handlers that read `c.get('jwtPayload')`.
- Pass the same secret or public-key configuration used when signing tokens.
- Scope the middleware to protected route paths instead of applying it to public endpoints.
- Read claims from `jwtPayload` in handlers and validate application-specific fields before granting access.
- Return tokens through the expected authorization header format when calling protected routes.