cache

August 18, 2026

cache

Kind: Function

Source: `src/middleware/cache/index.ts`

Part of: Middleware

Cache Middleware for Hono.

`cache` creates Hono middleware that checks a named Cache API store before running a route handler. For cache misses, it runs the handler, sets the configured `Cache-Control` header, and stores the response for later matching requests.

Signature

```
function cache(options: { cacheName: string | ((c: Context) => Promise<string> | string) wait?:
```

Parameters

Name	Type
`options`	`{ cacheName: string

Returns: `MiddlewareHandler`

Diagram

```
graph LR
  Request[Incoming request] --> Middleware[cache middleware]
  Middleware --> CacheStore[Named Cache API store]
  CacheStore -->|Cache hit| CachedResponse[Cached response]
  CacheStore -->|Cache miss| Handler[Route handler]
  Handler --> Response[Response with Cache-Control]
  Response --> CacheStore
```

```
  Response --> Client[Client]
  CachedResponse --> Client
```

Usage

```ts
import { Hono } from 'hono'
import { cache } from 'hono/cache'

const app = new Hono()

app.get(
  '/articles/:id',
  cache({
    cacheName: 'articles',
    cacheControl: 'max-age=3600',
    wait: true,
  }),
  async (c) => {
    const article = await getArticle(c.req.param('id'))
    return c.json(article)
  }
)

export default app
```

AI Coding Instructions

- Apply `cache` before the route handler whose response should be stored.
- Set `cacheName` to separate cached responses for different application areas.
- Set `cacheControl` to match the intended client and edge caching policy.
- Use `wait: true` when cache writes should run through the execution context rather than delay the response.
- Ensure cached routes return responses that are safe to reuse for matching requests.