



MIDDLEWARE

DetectorOptions

Kind: Interface

VERSION
5

UPDATED
12 August 2026

DOCUMENT
reference-subsystem-src-middleware/interface-detectoroptions

CONTENTS

1	Properties	2	4	AI Coding Instructions	4
2	Diagram	3	5	How it works	4
3	Usage	3			

Kind: Interface

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

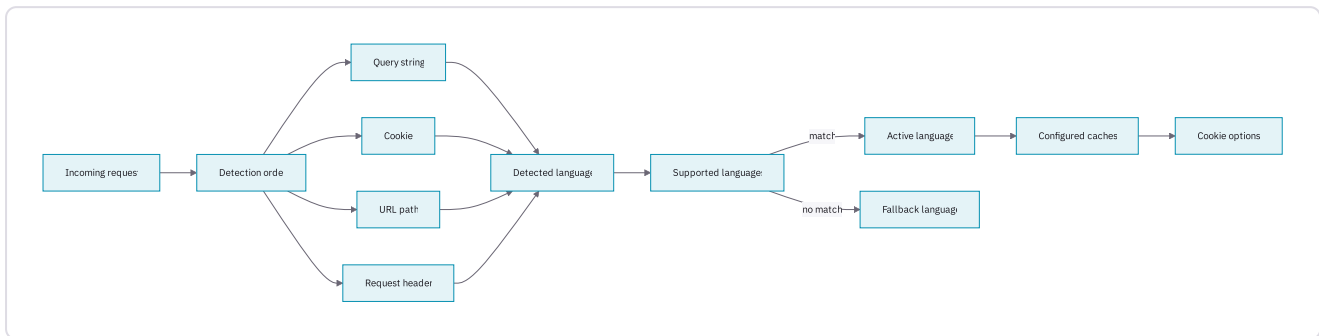
Part of: [Middleware](#)

`DetectorOptions` configures how language middleware detects, validates, and stores the active language. It defines detection sources, cache behavior, cookie settings, matching rules, fallback behavior, and the languages accepted by the application.

1 Properties

Property	Type
<code>order</code>	<code>DetectorType[]</code>
<code>lookupQueryString</code>	<code>string</code>
<code>lookupCookie</code>	<code>string</code>
<code>lookupFromPathIndex</code>	<code>number</code>
<code>lookupFromHeaderKey</code>	<code>string</code>
<code>caches</code>	<code>`CacheType[]</code>
<code>cookieOptions</code>	<code>`{ domain?: string path?: string sameSite?: 'Strict'</code>
<code>ignoreCase</code>	<code>boolean</code>
<code>fallbackLanguage</code>	<code>string</code>
<code>supportedLanguages</code>	<code>string[]</code>
<code>convertDetectedLanguage</code>	<code>(lang: string) => string</code>
<code>debug</code>	<code>boolean</code>

2 Diagram



3 Usage

```

import type {
  CacheType,
  DetectorOptions,
  DetectorType,
} from './src/middleware/language/language'

function createDetectorOptions(
  lookupFromPathIndex: number,
): DetectorOptions {
  const order: DetectorType[] = ['querystring', 'cookie', 'path', 'header']
  const caches: CacheType[] = ['cookie']

  return {
    order,
    lookupQueryString: 'lang',
    lookupCookie: 'language',
    lookupFromPathIndex,
    lookupFromHeaderKey: 'accept-language',
    caches,
    cookieOptions: {
      path: '/',
      sameSite: 'Lax',
      httpOnly: true,
      secure: true,
    },
    ignoreCase: true,
    fallbackLanguage: 'en',
    supportedLanguages: ['en', 'fr', 'de'],
  }
}

```

4 AI Coding Instructions

- Keep `order` aligned with the request sources supported by the language middleware.
- Use `supportedLanguages` to limit accepted detected values, and set `fallbackLanguage` to a supported value.
- Set `caches` to `false` when detected languages must not be persisted between requests.
- Match `lookupQueryString`, `lookupCookie`, and `lookupFromHeaderKey` with the names used by application routes and clients.
- Configure `cookieOptions` for the deployment environment, including `secure`, `sameSite`, and cookie scope.

5 How it works

`DetectorOptions` is the exported TypeScript interface that defines the complete configuration shape used by language detection functions and by `languageDetector`. Its required fields select detection sources, configure their lookups, define language matching, caching, and fallback behavior; `cookieOptions`, `convertDetectedLanguage`, and `debug` are optional.

[src/middleware/language/language.ts:13-45](#)

- `order` is an array of detector names: `'path'`, `'querystring'`, `'cookie'`, or `'header'`. The middleware tries these in array order and stops at the first detector that returns a language. [src/middleware/language/language.ts:10-10](#) [src/middleware/language/language.ts:241-258](#)
- `lookupQueryString` names the query parameter read by the query-string detector. [src/middleware/language/language.ts:16-17](#) [src/middleware/language/language.ts:137-140](#)
- `lookupCookie` names both the request cookie read by the cookie detector and the response cookie written when cookie caching is active. [src/middleware/language/language.ts:18-19](#) [src/middleware/language/language.ts:145-148](#) [src/middleware/language/language.ts:221-228](#)
- `lookupFromPathIndex` selects the zero-based non-empty URL path segment examined by the path detector. [src/middleware/language/language.ts:20-21](#) [src/middleware/language/language.ts:176-180](#)
- `lookupFromHeaderKey` names the request header passed to the Accept-header parser by the header detector. [src/middleware/language/language.ts:22-23](#) [src/middleware/language/language.ts:153-166](#)
- `caches` is either `false` or an array whose declared member type is `'cookie'`. Cookie caching writes only when the value is an array containing `'cookie'`; the cookie helper appends a `Set-Cookie` response header. [src/middleware/language/language.ts:11-11](#) [src/middleware/language/language.ts:25-25](#) [src/middleware/language/language.ts:221-228](#) [src/helper/cookie/index.ts:99-102](#)
- `cookieOptions` can set `domain`, `path`, `sameSite`, `secure`, `maxAge`, and `httpOnly` for that cache cookie. The middleware merges a supplied object with the default cookie options before use.

[src/middleware/language/language.ts:26-34](#)[src/middleware/language/language.ts:292-300](#)

- `ignoreCase` controls whether detected and supported codes are compared in lowercase or unchanged. A successful match returns the original configured entry from `supportedLanguages`, not the normalized input. [src/middleware/language/language.ts:35-36](#)
[src/middleware/language/language.ts:98-106](#)
- `supportedLanguages` is the accepted language-code list, and `fallbackLanguage` is returned when no configured detector yields an accepted language. [src/middleware/language/language.ts:37-40](#)
[src/middleware/language/language.ts:260-266](#)
- `convertDetectedLanguage`, when present, receives the trimmed detected value before case handling and supported-language matching. Errors during normalization result in that candidate being rejected. [src/middleware/language/language.ts:41-42](#) [src/middleware/language/language.ts:92-131](#)
- `debug` causes successful detections and detector or cookie-cache exceptions to be sent to `console.log` or `console.error`; it does not alter the selected language.
[src/middleware/language/language.ts:43-44](#) [src/middleware/language/language.ts:227-232](#)
[src/middleware/language/language.ts:244-256](#)

Language candidates are trimmed, optionally converted, then checked first for an exact configured match. If that fails, a longer detected tag may match the longest configured prefix immediately followed by `-`; otherwise it is rejected. [src/middleware/language/language.ts:92-131](#) For header detection, parsed `Accept-Language` entries are examined in parser order, which is sorted by descending quality only when the parser observes a later entry with a higher quality value.

[src/middleware/language/language.ts:153-167](#) [src/utils/accept.ts:211-237](#)

`languageDetector` accepts `Partial<DetectorOptions>`, overlays it on `DEFAULT_OPTIONS`, and separately merges nested cookie settings. The defaults check query string `lang`, cookie `language`, then header `accept-language`; accept `en`; fall back to `en`; cache in a cookie; ignore case; and set cache-cookie defaults of `SameSite=Strict`, `Secure`, one-year `maxAge`, and `HttpOnly`.

[src/middleware/language/language.ts:51-68](#) [src/middleware/language/language.ts:292-300](#)

Before returning middleware, construction throws `Error` when the fallback is absent from `supportedLanguages`, the path index is negative, or `order` contains a name outside the detector map. [src/middleware/language/language.ts:204-216](#) On each request, the resulting language is stored as `ctx` variable `language` before downstream middleware runs.

[src/middleware/language/language.ts:304-308](#)