acceptsOptions

August 17, 2026

acceptsOptions

Kind: Interface

Source: `src/helper/accepts/accepts.ts`

Part of: Helper

`acceptsOptions` defines the contract for matching a list of `Accept` values against an `acceptsConfig`. Its `match` method receives the available accept entries and configuration, then returns the selected value as a string.

Properties

Property	Type
`match`	`(accepts: Accept[], config: acceptsConfig) => string`

Diagram

```
graph LR
  A[Accept[]] --> C[acceptsOptions.match]
  B[acceptsConfig] --> C
  C --> D[string]
```

Usage

```
import type { acceptsOptions } from './src/helper/accepts/accepts';

const options: acceptsOptions = {
  match(accepts, config) {
    const match = accepts.find((accept) => accept.type === config.type);

    return match?.type ?? '';
  },
};
```

```
const selected = options.match(accepts, config);
```

AI Coding Instructions

- Implement `match` with both the `Accept[]` input and `acceptsConfig` input in mind.
- Return a string for every match path, including when no `Accept` entry matches.
- Keep matching rules consistent with the `Accept` and `acceptsConfig` types used by the accepts helper.
- Avoid mutating the incoming `accepts` array or `config` object during matching.

How it works

`acceptsOptions` is the options interface accepted by `accepts(c, options)`. It extends `acceptsConfig`, so callers must set:

- `header`: one of `'Accept'`, `'Accept-Charset'`, `'Accept-Encoding'`, `'Accept-Language'`, `'Accept-Patch'`, `'Accept-Post'`, or `'Accept-Ranges'`. src/helper/accepts/accepts.ts:11-18 src/utils/headers.ts:345-352
- `supports`: an array of strings against which parsed accept-value `type` fields are compared. src/helper/accepts/accepts.ts:11-15 src/helper/accepts/accepts.ts:21-24
- `default`: the string returned when the configured request header is absent or when the selected matcher finds no supported type. src/helper/accepts/accepts.ts:14 src/helper/accepts/accepts.ts:42-44 src/helper/accepts/accepts.ts:21-24

It optionally accepts `match`, a function with the signature `(accepts: Accept[], config: acceptsConfig) => string`. src/helper/accepts/accepts.ts:17-19 `accepts` reads `options.header` from the request, parses its value, and calls this function with the parsed values and the same options object when `match` is set; otherwise it calls `defaultMatch`. src/helper/accepts/accepts.ts:40-48

Each element passed as the first `match` argument has `type`, `params`, and numeric `q` fields. src/helper/accepts/accepts.ts:5-9 The parser initializes missing quality values to `1`, reads a `q` parameter when present, and returns an array of parsed values. src/utils/accept.ts:163-167 src/utils/accept.ts:211-237

Without a custom `match`, `defaultMatch` sorts the parsed array by descending `q`, returns the first entry whose `type` occurs in `supports`, and otherwise returns `default`.

src/helper/accepts/accepts.ts:21-25 A custom matcher can choose different selection rules; the test demonstrates one that sorts ascending by `q` and returns the first supported type. src/helper/accepts/accepts.test.ts:89-112

There is no runtime validation of `header`, `supports`, `default`, or the return value of `match` in this code. If the request header is falsy, `accepts` returns `default` immediately and does not parse the header or call `match`. src/helper/accepts/accepts.ts:41-48