acceptsConfig

August 19, 2026

acceptsConfig

Kind: Interface

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

Part of: Helper

`acceptsConfig` defines the configuration required for Accept header handling. It stores the parsed `AcceptHeader`, a list of supported content types, and the fallback content type returned when no supported match is found.

Properties

Property	Type
`header`	`AcceptHeader`
`supports`	`string[]`
`default`	`string`

Diagram

```
graph LR
  Config[acceptsConfig]
  Header[header: AcceptHeader]
  Supports[supports: string[]]
  Default[default: string]

  Config --> Header
  Config --> Supports
  Config --> Default
```

Usage

```ts
import type { acceptsConfig } from './helper/accepts/accepts';

const config: acceptsConfig = {
  header: requestAcceptHeader,
  supports: ['application/json', 'text/html'],
  default: 'application/json',
};

function selectContentType(config: acceptsConfig): string {
  return config.default;
}
```

AI Coding Instructions

- Pass a parsed `AcceptHeader` value through `header`; do not replace it with a raw header string.
- Keep `supports` limited to content types the calling code can return.
- Set `default` to a value that also exists in `supports`.
- Use `acceptsConfig` when passing Accept negotiation settings between request handling code and content-type selection logic.

How it works

`acceptsConfig` is an exported TypeScript interface that describes the required configuration shared by the `accepts` helper and its matching functions. It has three required fields: `header`, `supports`, and `default`. [src/helper/accepts/accepts.ts:11-15]

- `header` selects the request header that `accepts` reads. Its type is `AcceptHeader`, limited to `'Accept'`, `'Accept-Charset'`, `'Accept-Encoding'`, `'Accept-Language'`, `'Accept-Patch'`, `'Accept-Post'`, or `'Accept-Ranges'`. [src/helper/accepts/accepts.ts:12] [src/utils/headers.ts:345-352]
- `supports` is a `string[]` of candidate values for matching. [src/helper/accepts/accepts.ts:13]
- `default` is the `string` returned when the selected header is absent or when the default matcher finds no supported accepted type. [src/helper/accepts/accepts.ts:14] [src/helper/accepts/accepts.ts:21-25] [src/helper/accepts/accepts.ts:40-48]

`acceptsOptions` extends this interface with an optional `match` callback. That callback receives parsed accept entries and the same `acceptsConfig` object, and returns a

string. [src/helper/accepts/accepts.ts:17-19] When no callback is supplied, `accepts` uses `defaultMatch`. [src/helper/accepts/accepts.ts:45-48]

The default matcher sorts parsed entries by descending `q` value, selects the first entry whose `type` occurs in `supports`, and returns that type; otherwise it returns `default`. [src/helper/accepts/accepts.ts:21-25] The matcher calls `.sort()` on its `accepts` argument. [src/helper/accepts/accepts.ts:23]

`accepts` reads `c.req.header(options.header)`. If that value is falsy, it returns `options.default`; otherwise, it parses the header and invokes either `options.match` or `defaultMatch` with the parsed entries and configuration. [src/helper/accepts/accepts.ts:40-48]

No runtime validation or explicit error handling for the three configuration fields appears in this file. [src/helper/accepts/accepts.ts:11-19]