proxy

August 17, 2026

proxy

Kind: Function

Source: `src/helper/proxy/index.ts`

Part of: Helper

Fetch API wrapper for proxy.
The parameters and return value are the same as for `fetch` (except for the proxy-specific options).

The “Accept-Encoding” header is replaced with an encoding that the current runtime can handle.
Unnecessary response headers are deleted and a Response object is returned that can be returned
as is as a response from the handler.

`proxy` wraps the Fetch API for forwarding requests through a handler. It adjusts the `Accept-Encoding` header to match the current runtime, removes response headers that should not be forwarded, and returns a `Response` that can be returned directly from the handler.

Signature

```
async function proxy(input, proxyInit)
```

Parameters

Name	Type
`input`	`any`
`proxyInit`	`any`

Diagram

```
graph LR
  A[Incoming handler request] --> B[proxy]
  B --> C[Adjust Accept-Encoding]
  C --> D[Fetch upstream resource]
  D --> E[Remove response headers]
  E --> F[Return Response from handler]
```

Usage

```
import { proxy } from "./helper/proxy";

export async function handleRequest(request: Request): Promise<Response> {
  const url = new URL(request.url);
  const upstreamUrl = `https://api.example.com${url.pathname}`;

  return proxy(upstreamUrl, {
    method: request.method,
    headers: request.headers,
    body: request.body,
  });
}
```

AI Coding Instructions

- Treat `proxy` as a Fetch-compatible wrapper; pass standard fetch inputs unless proxy-specific options are needed.
- Return the resulting `Response` directly from the request handler rather than rebuilding its body or headers.
- Do not manually set `Accept-Encoding` before calling `proxy`; the wrapper selects an encoding supported by the current runtime.
- Preserve the incoming method, headers, and body when forwarding a request unless the handler intentionally changes them.

Relationships

- IMPORTS → `HTTPException`