parseResponse

August 18, 2026

parseResponse

Kind: Function

Source: `src/client/utils.ts`

Part of: Client

Shortcut to get a consumable response from `hc`'s fetch calls (Response), with types inference.

Smartly parse the response data, throwing a structured error if the response is not `ok`. (DetailedError)

`parseResponse` accepts an `hc` fetch response and returns its parsed, consumable data with the response type inferred from the client call. It checks `response.ok` before parsing and throws a `DetailedError` when the request fails.

Signature

```
async function parseResponse(fetchRes: T | Promise<T>): Promise< FilterClientResponseByStatusCo
```

Parameters

Name	Type
`fetchRes`	`T

Returns: `Promise< FilterClientResponseByStatusCode< T, Exclude<ContentfulStatusCode, ClientErrorStatusCode | ServerErrorStatusCode> > extends never ? undefined : FilterClientResponseByStatusCode< T, Exclude<ContentfulStatusCode, ClientErrorStatusCode | ServerErrorStatusCode> > extends ClientResponse<infer RT, infer _, infer RF> ? RF extends 'json' ? RT : RT extends string ? RT : string : undefined >`

Diagram

```
graph LR
  A[hc client call] --> B[parseResponse]
  B --> C{response.ok}
  C -->|true| D[Parsed typed data]
  C -->|false| E[DetailedError]
```

Usage

```
import { hc, parseResponse } from 'hono/client'
import type { AppType } from './server'

const client = hc<AppType>('/api')

const user = await parseResponse(
  client.users[':id'].$get({
    param: { id: 'user-id' },
  })
)

console.log(user)
```

AI Coding Instructions

- Pass the `hc` client call directly to `parseResponse` to preserve inferred response types.
- Handle `DetailedError` where request failures need status or response error details.
- Do not call `response.json()`, `response.text()`, or another body reader before `parseResponse`; response bodies can only be consumed once.
- Use `parseResponse` at client-call boundaries so callers receive parsed data rather than raw `Response` objects.