Data Fetching & Caching.

SPA State (Data)

- **Client state (aka App State).**
  - **e.g. Menu selection, UI theme, Text input, logged-in user id.**

  - **Characteristics:**
    - **Client-owned; Not shared; Not persisted (across sessions); Up-to-date.**
    - **Accessed synchronously.**
    - useState**() hook**
    - **Management - Private to a component or Global state (Context).**

SPA State (Data)

- **Server state (The M in MVC).**
  - **e.g. list of ‘discover’ movies, movie details, friends.**

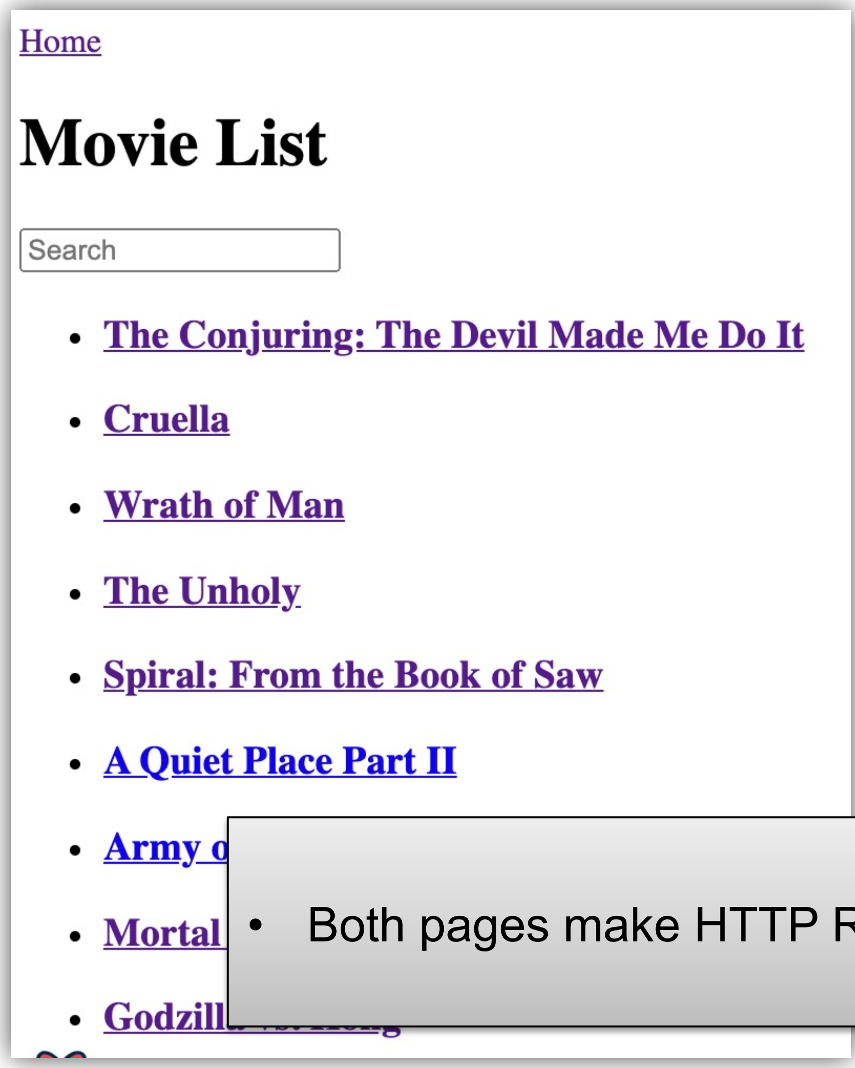
  - **Characteristics:**
    - **Persisted remotely. Shared ownership.**
    - **Accessed asynchronously → Impacts User experience.**
    - **Can change without client’s knowledge → Client can be ‘out of date’.**
    - useState **+** useEffect **hooks.**

SPA Server State.

- **Server state characteristics (contd.).**
  - **Management options:**
    1. **Private to a component →**
       - **Good separation of concerns.**
       - **Unnecessary re-fetching.**
    2. **Global state (Context).**
       - **No unnecessary re-fetching.**
       - **Fetching data before its required.**
       - **Poor separation of concerns.**
    3. **3rd party library – e.g. Redux**
       - **Same as 2 above.**

- **We want the best of 1 and 2, if possible.**

Sample App.

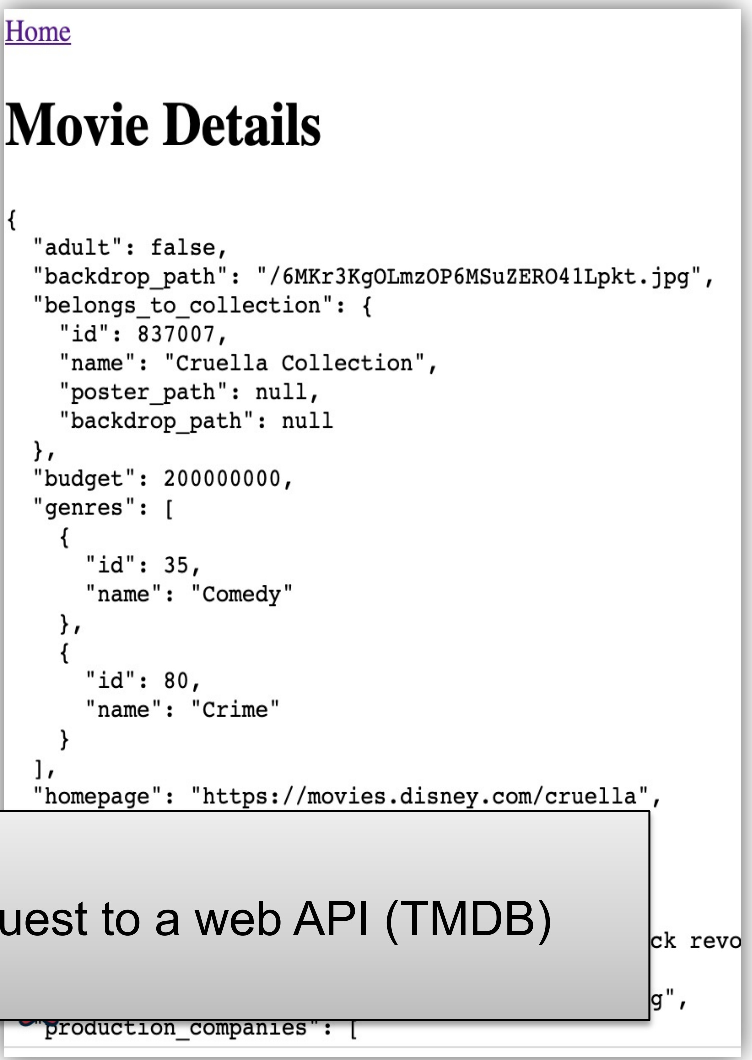


Home

Movie List

Search

- The Conjuring: The Devil Made Me Do It
- Cruella
- Wrath of Man
- The Unholy
- Spiral: From the Book of Saw
- A Quiet Place Part II
- Army o
- Mortal
- Godzill



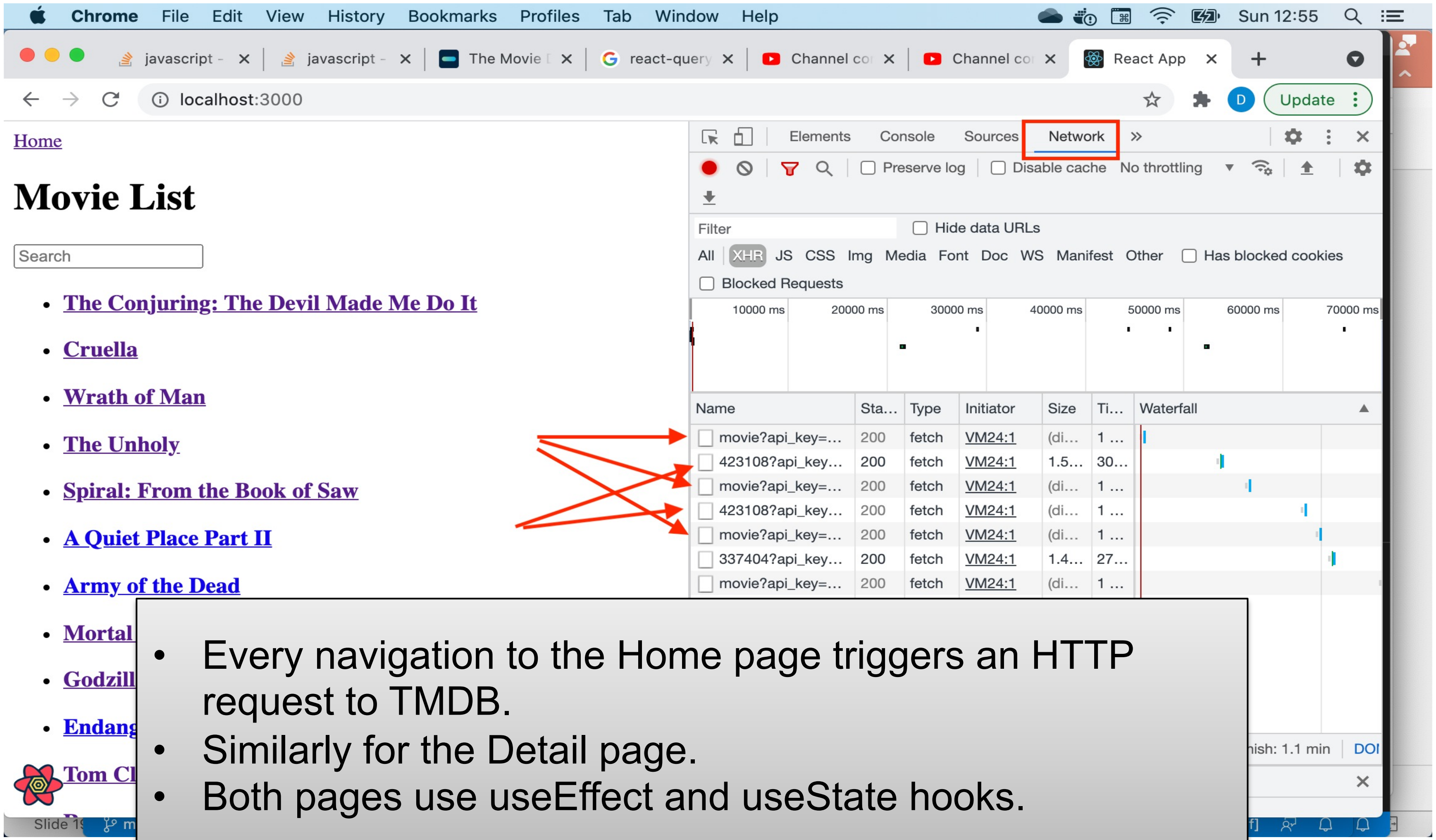
Home

Movie Details

```
{
  "adult": false,
  "backdrop_path": "/6MKr3KgOLmzOP6MSuZERO41Lpkt.jpg",
  "belongs_to_collection": {
    "id": 837007,
    "name": "Cruella Collection",
    "poster_path": null,
    "backdrop_path": null
  },
  "budget": 200000000,
  "genres": [
    {
      "id": 35,
      "name": "Comedy"
    },
    {
      "id": 80,
      "name": "Crime"
    }
  ],
  "homepage": "https://movies.disney.com/cruella",
ck revo
g",
  "production_companies": [
```

- Both pages make HTTP Request to a web API (TMDB)

Sample App – The Problem.


Chrome File Edit View History Bookmarks Profiles Tab Window Help
Sun 12:55
javascript
javascript
The Movie
react-query
Channel co
Channel co
React App
localhost:3000
Update
Elements Console Sources Network
Preserve log
Disable cache
No throttling
Filter
Hide data URLs
All XHR JS CSS Img Media Font Doc WS Manifest Other
Has blocked cookies
Blocked Requests
10000 ms 20000 ms 30000 ms 40000 ms 50000 ms 60000 ms 70000 ms

Home

Movie List

Search

- The Conjuring: The Devil Made Me Do It
- Cruella
- Wrath of Man
- The Unholy
- Spiral: From the Book of Saw
- A Quiet Place Part II
- Army of the Dead
- Mortal
- Godzill
- Endang
- Tom Cl

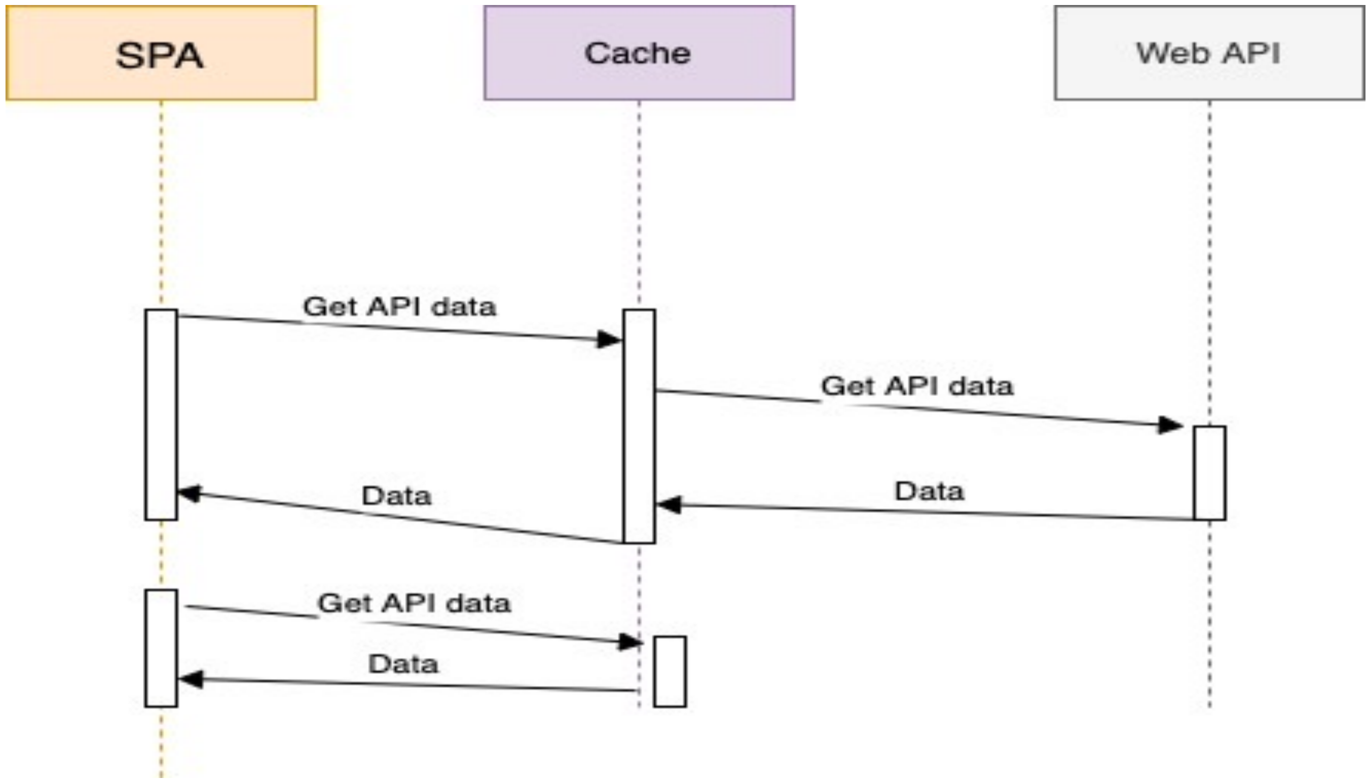
Name	Sta…	Type	Initiator	Size	Ti…	Waterfall
movie?api_key=…	200	fetch	VM24:1	(di…	1 …	
423108?api_key…	200	fetch	VM24:1	1.5…	30…	
movie?api_key=…	200	fetch	VM24:1	(di…	1 …	
423108?api_key…	200	fetch	VM24:1	(di…	1 …	
movie?api_key=…	200	fetch	VM24:1	(di…	1 …	
337404?api_key…	200	fetch	VM24:1	1.4…	27…	
movie?api_key=…	200	fetch	VM24:1	(di…	1 …	

ish: 1.1 min

- Every navigation to the Home page triggers an HTTP request to TMDB.
- Similarly for the Detail page.
- Both pages use useEffect and useState hooks.

Sample App – The Solution. .

- **Cache the API data locally in the browser.**
- **Caches are in-memory datastores with high performance, low latency.**
- **Helps reduce the workload on the backend for read intensive workloads.**



Caching (General).

- **Caches are** key-valu**e datastores.**

  key1: value, key2: value, …….

  - **Keys must be <u>unique</u>.**
  - **Value can be any <u>serializable</u> data type – JS Object, JS array, Primitive.**
- **Cache hit – The requested data is in the cache.**
- **Cache miss - The requested data is not in the cache.**
- **Caches have a simple interface:**

  serializedValue = cache.get(key)

  cache.delete(key)

  cache.purge()
- **Cache entries should have a <u>time-to-live</u> (TTL).**

The react-query library

- **3rd party JavaScript (React) caching library.**
  - **Provides a set of hooks.**

  e.g. const { data, error, isLoading, isError } =
  useQuery(key, queryFunction);

  - data – from the cache or returned by the API.
  - error – error response from API.
  - isLoading(boolean) – true while waiting for API response.
  - isError (boolean) – true when API response is an error status.

- **It causes a component to re-render on query completion.**
- **It replaces your** useState **and** useEffect **hooks.**

The query key.

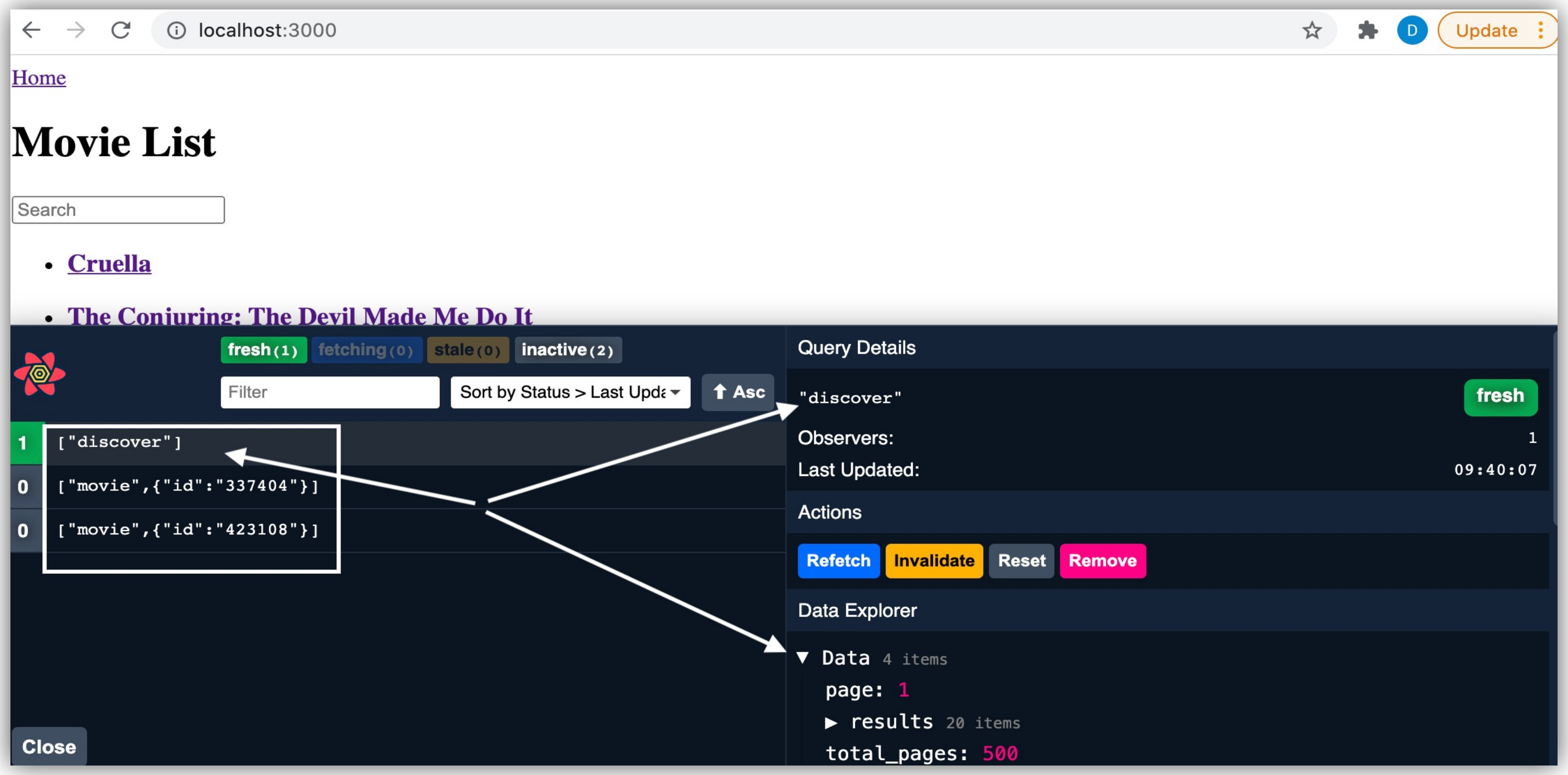
- *“Query keys can be as simple as a string, or as complex as an array of many strings and nested objects. As long as the query key is serializable, and **unique to the query's data** …..”*

```
e.g.  const { ……, } =
              useQuery( ["movie", { id: 123456 }],  getMovie);

export const getMovie = async (args) => {
   const [ prefix, { id: id }] = args.queryKey;
  …. Do HTTP GET using movie id of 123456
```

react-query DevTools.

- **Allows us <u>observe</u> the current state of the cache data store – great when debugging.**



localhost:3000

Home

Movie List

Search

- Cruella
- The Conjuring: The Devil Made Me Do It

fresh (1) fetching (0) stale (0) inactive (2)

Filter | Sort by Status > Last Upda | Asc

```
1 ["discover"]
0 ["movie",{"id":"337404"}]
0 ["movie",{"id":"423108"}]
```

Query Details

"discover" fresh

Observers: 1

Last Updated: 09:40:07

Actions

Refetch Invalidate Reset Remove

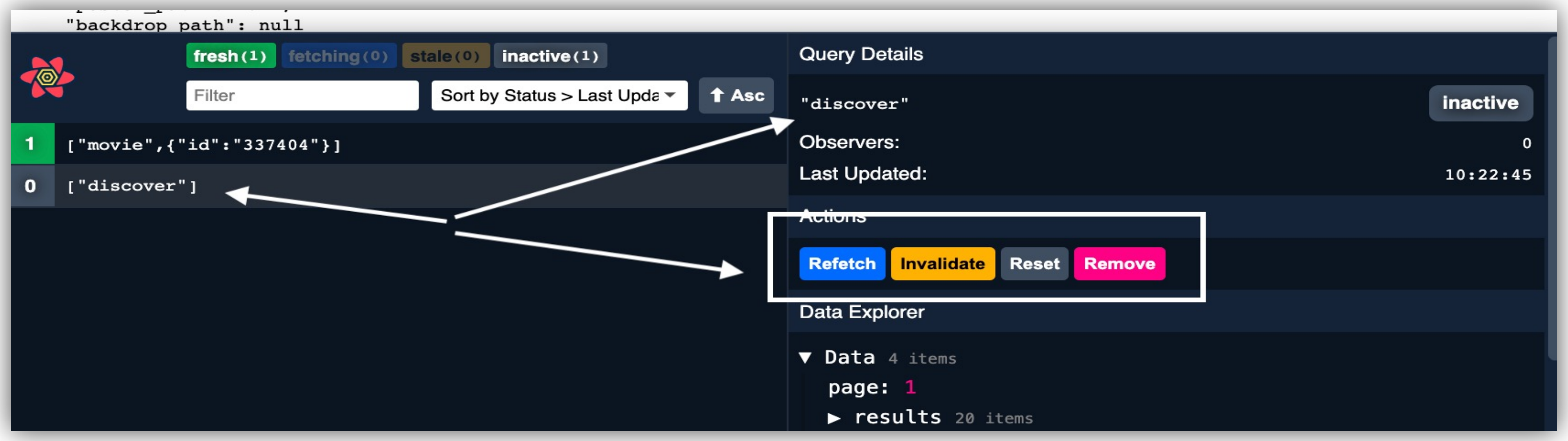
Data Explorer

```
▼ Data 4 items
  page: 1
  ▶ results 20 items
  total_pages: 500
```

Close

react-query DevTools.

- **Allows us <u>manipulate</u> cache entries.**



  - **Refresh – force cache to re-request (update) data from web API immediately.**
  - **Invalidate – Set entry as ‘stale’. Cache will request update from web API when required by the SPA.**
  - **Reset – only applies when app can update data.**
  - **Remove – remove entry from cache immediately.**

Summary

- **State Management - The M in MVC**

- **State:**
  1. **Client/App state.**
  2. **Server state.**

- **Cache server state locally in the browser.**
  - **Reduces unnecessary HTTP traffic → Reduce page loadtime**
  - **Be aware of cache entry staleness → Use TTL.**

- **The react-query library**
  - **A set of hooks for cache interaction.**

The End