BIBLIOTECA
REACT-QUERY

Qual vai ser o foco?

Apenas envolvendo o nosso caso de uso

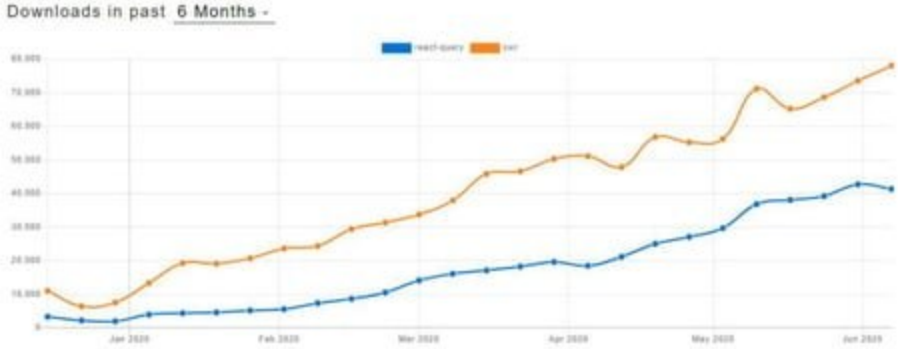
Agenda

Apresentação

O que pode ser feito

Uso no Maila

Comparando números



Stats

	stars	forks	issues	updated	created	size
react-query	7,895	289	19	Jun 20, 2020	Sep 10, 2019	minzipped size: 5.2 KB
swr	10,593	333	108	Jun 20, 2020	Oct 28, 2019	minzipped size: 4.4 KB

README.MD


README.md
React Query
React Query
Hooks for fetching, caching and updating asynchronous data in React
Enjoy this library? Try them all! React Table, React Form, React Charts
Quick Features
Transport/protocol/backend agnostic data fetching (REST, GraphQL, promises, whatever!)
Auto Caching + Refetching (stale-while-revalidate, Window Refocus, Polling/Realtime)

Quem usa?

Features de propaganda

Quick Features

- Transport/protocol/backend agnostic data fetching (REST, GraphQL, promises, whatever!)
- Auto Caching + Refetching (stale-while-revalidate, Window Refocus, Polling/Realtime)
- Parallel + Dependent Queries
- Mutations + Reactive Query Refetching
- Multi-layer Cache + Automatic Garbage Collection
- Paginated + Cursor-based Queries
- Load-More + Infinite Scroll Queries w/ Scroll Recovery
- Request Cancellation
- React Suspense + Fetch-As-You-Render Query Prefetching
- Dedicated Devtools (React Query Devtools)
- 4kb - 6kb (depending on features imported) minzipped size 6.2 KB

O desafio que veio para resolver

The Challenge

Tools for managing "global state" are plentiful these days, but most of these tools:

- Mistake **server cache state** for **global state**
- Force you to manage async data in a synchronous way
- Duplicate unnecessary network operations
- Use naive or over-engineered caching strategies
- Are too basic to handle large-scale apps or
- Are too complex or built for highly-opinionated systems like Redux, GraphQL, [insert proprietary tools], etc.
- Do not provide tools for server mutations
- Either do not provide easy access to the cache or do, but expose overpowered foot-gun APIs to the developer

Solução encontrada

The Solution

React Query exports a set of hooks that address these issues. Out of the box, React Query:

- Separates your **server cache state** from your **global state**
- Provides async aware APIs for reading and updating server state/cache
- Dedupes both async and sync requests to async resources
- Automatically caches data, invalidates and refetches stale data, and manages garbage collection of unused data
- Scales easily as your application grows
- Is based solely on Promises, making it highly unopinionated and interoperable with any data fetching strategy including REST, GraphQL and other transactional APIs
- Provides an integrated promise-based mutation API
- Opt-in Manual or Advance cache management

Inspiração

▼ Inspiration & Hat-Tipping

A big thanks to both [Draqula](https://github.com/vadimdemedes/draqula) for inspiring a lot of React Query's original API and documentation and also [Zeit's SWR](https://github.com/zeit/swr) and its creators for inspiring even further customizations and examples. You all rock!

- https://github.com/vadimdemedes/draqula
- https://github.com/zeit/swr

Diferenças de React-Query para SWR

▼ How is this different from Zeit's SWR?

Zeit's SWR is a great library, and is very similar in spirit and implementation to React Query with a few notable differences:

- Automatic Cache Garbage Collection - React Query handles automatic cache purging for inactive queries and garbage collection. This can mean a much smaller memory footprint for apps that consume a lot of data or data that is changing often in a single session
- `useMutation` - A dedicated hook for handling generic lifecycles around triggering mutations and handling their side-effects in applications. SWR does not ship with anything similar, and you may find yourself reimplementing most if not all of `useMutation`'s functionality in user-land. With this hook, you can extend the lifecycle of your mutations to reliably handle successful refetching strategies, failure rollbacks and error handling.
- Prefetching - React Query ships with 1st class prefetching utilities which not only come in handy with non-suspenseful apps but also make fetch-as-you-render patterns possible with React Query. SWR does not come with similar utilities and relies on `<link rel='preload'>` and/or manually fetching and updating the query cache
- Query cancellation integration is baked into React Query. You can easily use this to wire up request cancellation in most popular fetching libraries, including but not limited to fetch and axios.
- Query Key Generation - React Query uses query key generation, query variables, and implicit query grouping. The query key and variables that are passed to a query are less URL/Query-based by nature and much more flexible. All items supplied to the query key array are used to compute the unique key for a query (using a stable and deterministic sorting/hashing implementation). This means you can spend less time thinking about precise key matching, but more importantly, allows you to use partial query-key matching when refetching, updating, or removing queries in mass eg. you can refetch every query that starts with a `todos` in its key, regardless of variables, or you can target specific queries with (or without) variables, and even use functional filtering to select queries in most places. This architecture is much more robust and forgiving especially for larger apps.

DevTools

https://codesandbox.io/s/github/tannerlinsley/react-query/tree/master/examples/playground

Demonstrando a lib. no ____

Informação sensível, apenas permitida dentro do domínio da Zup!

Indo a fundo de como funciona

https://youtu.be/seU46c6Jz7E

Osmar Petry

Frontend

CONTATO

osmar.petry@zup.com.br

+55 47 99148-XXXX

https://github.com/osmarpetry

Obrigado!