link: null
title: 珠峰架构师成长计划
description: src\index.js
keywords: null
author: null
date: null
publisher: 珠峰架构师成长计划
stats: paragraph=216 sentences=1013, words=6082

1.React Query介绍

- react-query (https://react-query.tanstack.com/overview)基于Hooks,能够智能管理请求的一切内容，包括数据、状态、缓存，更新等
- 管理请求
    - 基本上axios等请求库封装，可以实现请求、轮询、失败重试、无限加载等功能
    - 可以在网络重连、窗口获得焦点等时机等向服务器发送请求同步状态
- 状态管理
    - 可以把服务器端的状态缓存在客户端的内存中，从而让任意组件获取这些状态
- comparison (https://react-query.tanstack.com/comparison)

2. 安装

```
npm install react-query axios --save
npm install express cors morgan --save
```

3.基本查询

src\index.js

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import App from './App';
import { QueryClient, QueryClientProvider } from 'react-query';
const queryClient = new QueryClient();
const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <QueryClientProvider client={queryClient}>
    <App />
  QueryClientProvider>);
```

src\App.js

```
import { useQuery } from 'react-query';
import request from './request';
function App() {
  const { data } = useQuery('users', () => request.get('/users'))
  return (
    (<ul>
      {
        data?.map((user) => <li key={user.id}>{user.name}li>)
      }
    ul>)
  )
}
export default App;
```

src\request.js

```
import axios from 'axios';
axios.interceptors.response.use(response => response.data);
axios.defaults.baseURL = 'http://localhost:8080';
export default axios;
```

```
const express = require('express');
const cors = require('cors');
const logger = require('morgan');
const app = express();
app.use(express.json());
app.use(logger('dev'));
app.use(cors({
  allowedHeaders: ["Content-Type"],
  allowMethods: ["GET", "POST", "PUT", "DELETE", "OPTIONS"]
}));
const users=new Array(10).fill(true).map((item, index) => ({ id: String(index + 1), name: `name${index + 1}` }))
app.use((req, res, next) => {
  setTimeout(next, 1000);
});
app.get('/users', (req, res) => {
  res.json(users);
});
app.listen(8080, () => console.log('started on port 8080'));
```

4.加载状态

字段 含义 取值 status 状态 loading、error、success isLoading 是否首次加载中 true、false isError 是否获取失败 isSuccess 是否获取成功 true、false

src\App.js

```
import { useQuery } from 'react-query';
import request from './request';
function App() {
+  const { data, isLoading, isError } = useQuery('users', () => {
+    throw new Error('用户列表加载失败!');
+    return request.get('/users');
  })
+ if (isLoading) return 加载中.......

+ if (isError) return 加载失败
  return (
    (
      {
        data?.map((user) => {user.name})
      }
    )
  )
}
export default App;
```

5.开发工具

src\App.js

```
import { useQuery } from 'react-query';
+import { ReactQueryDevtools } from 'react-query/devtools'
import request from './request';
+function Users() {
+  const { data, isLoading, isError } = useQuery('users', () => request.get('/users'))
+  if (isLoading) return 加载中.......

+  if (isError) return 加载失败
+  return (
+     (
+       {
+         data?.map((user) => {user.name})
+       }
+     )
+  )
+}
function App() {
  return (
+    <>
+
+
+    </>
  )
}
export default App;
```

6.refetchOnWindowFocus

字段 含义 取值 isFetching 是否正在请求 true、false

src\App.js

```
import { useQuery } from 'react-query';
import { ReactQueryDevtools } from 'react-query/devtools'
import request from './request';
function Users() {
+ const { data, isLoading, isError, isFetching } = useQuery('users', () => request.get('/users'), {
+   refetchOnWindowFocus: true
  })
  if (isLoading) return 加载中.......

  if (isError) return 加载失败
  return (
    (
      <>

          {
            data?.map((user) => {user.name})
          }

+       {isFetching && 更在更新数据...}
      </>
    )
  )
}
function App() {
  return (
    <>

    </>
  )
}
export default App;
```

7.staleTime

- `staleTime`可以理解为数据保质期，在保质期内遇到同 `key` 的请求，不会去再次获取数据，也就是从缓存中取，瞬间切换展示， `isFetching` 也一直为 `false`

字段 含义 取值 isStale 是否已经过期

立刻过期,

永不过期

src\App.js

```
import { useQuery } from 'react-query';
import { ReactQueryDevtools } from 'react-query/devtools'
import request from './request';
function Users() {
  const { data, isLoading, isError, isFetching } = useQuery('users', () => request.get('/users'), {
    refetchOnWindowFocus: true,
+   staleTime: 3000
  })
  if (isLoading) return 加载中.......

  if (isError) return 加载失败
  return (
    (
      <>

          {
            data?.map((user) => {user.name})
          }

        {isFetching && 更在更新数据...}
      </>
    )
  )
}
function App() {
  return (
    <>

    </>
  )
}
export default App;
```

8.cacheTime

- cacheTime是指未使用/非活动缓存数据保留在内存中的时间（以毫秒为单位）
- 当查询的缓存变得未使用或不活动时，该缓存数据将在此持续时间后被垃圾收集。当指定不同的缓存时间时，将使用最长的一个

src\App.js

```
import { useState } from 'react';
import { useQuery } from 'react-query';
import { ReactQueryDevtools } from 'react-query/devtools'
import request from './request';
function Users() {
  const { data, isLoading, isError, isFetching } = useQuery('users', () => request.get('/users'), {
    refetchOnWindowFocus: true,
+   staleTime: Infinity,
+   cacheTime: 5000
  })
  if (isLoading) return 加载中.......

  if (isError) return 加载失败
  return (
    (
      <>

          {
            data?.map((user) => {user.name})
          }

        {isFetching && 更在更新数据...}
      </>
    )
  )
}
function App() {
+ const [show, setShow] = useState(true);
  return (
    <>
+      setShow(!show)}>{show ? '隐藏' : '显示'}
+     {show && }

    </>
  )
}
export default App;
```

9.轮询

src\App.js

```
import { useState } from 'react';
import { useQuery } from 'react-query';
import { ReactQueryDevtools } from 'react-query/devtools'
import request from './request';
function Users() {
  const { data, isLoading, isError, isFetching } = useQuery('users', () => request.get('/users'), {
    refetchOnWindowFocus: true,
    staleTime: Infinity,
    cacheTime: 5000,
+   refetchInterval: 1000
  })
  if (isLoading) return 加载中.......

  if (isError) return 加载失败
  return (
    (
      <>

          {
            data?.map((user) => {user.name})
          }

        {isFetching && 更在更新数据...}
      </>
    )
  )
}
function App() {
  const [show, setShow] = useState(true);
  return (
    <>
       setShow(!show)}>{show ? '隐藏' : '显示'}
      {show && }

    </>
  )
}
export default App;
```

api.js

```
const express = require('express');
const cors = require('cors');
const logger = require('morgan');
const app = express();
app.use(express.json());
app.use(cors({
  allowedHeaders: ["Content-Type"],
  allowMethods: ["GET", 'POST', "PUT", "DELETE", "OPTIONS"]
}));
app.use(logger('dev'));
const users = new Array(10).fill(true).map((item, index) => ({ id: String(index + 1), name: `name${index + 1}` }))
app.use((req, res, next) => {
  setTimeout(next, 1000);
});
app.get('/users', (req, res) => {
   res.json(users.map(user => ({ ...user, name: user.name + '#' + new Date().toLocaleString() })));
});
app.listen(8080, () => console.log('started on port 8080'));
```

10.查询键去重

src\App.js

```
import { useState } from 'react';
import { useQuery } from 'react-query';
import { ReactQueryDevtools } from 'react-query/devtools'
import request from './request';
+function Users({ queryKey }) {
+ const { data, isLoading, isError, isFetching } = useQuery(queryKey, () => request.get('/users'))
  if (isLoading) return 加载中.......

  if (isError) return 加载失败
  return (
    (
      <>

          {
            data?.map((user) => {user.name})
          }

        {isFetching && 更在更新数据...}
      </>
    )
  )
}
function App() {
  return (
    <>
+
+

    </>
  )
}
export default App;
```

11.自定义hooks

src\App.js

```
import { useQuery } from 'react-query';
import { ReactQueryDevtools } from 'react-query/devtools'
import request from './request';
+function useUsers() {
+  return useQuery('users', () => request.get('/users'));
+}
+function Stats() {
+  const { data } = useUsers();
+  return data && 共计{data.length}用户
+}
function Users() {
+ const { data, isLoading, isError, isFetching } = useUsers();
  if (isLoading) return 加载中.......

  if (isError) return 加载失败
  return (
    (
      <>

          {
            data?.map((user) => {user.name})
          }

        {isFetching && 更在更新数据...}
      </>
    )
  )
}
function App() {
  return (
    <>

+


    </>
  )
}
export default App;
```

12.QueryObserver

src\App.js

```
import { useEffect, useState } from 'react';
+import { useQuery, QueryObserver, useQueryClient } from 'react-query';
import { ReactQueryDevtools } from 'react-query/devtools'
import request from './request';
function Stats() {
+ const [data, setData] = useState();
+ const queryClient = useQueryClient();
+ useEffect(() => {
+   const observer = new QueryObserver(queryClient, { queryKey: 'users' })
+   const unsubscribe = observer.subscribe(result => setData(result.data))
+   return unsubscribe;
+ }, []);
  return data && 共计{data.length}用户
}
function Users() {
+ const { data, isLoading, isError, isFetching } = useQuery('users', () => request.get('/users'))
  if (isLoading) return 加载中.......

  if (isError) return 加载失败
  return (
    (
      <>

          {
            data?.map((user) => {user.name})
          }

        {isFetching && 更在更新数据...}
      </>
    )
  )
}
function App() {
  return (
    <>

    </>
  )
}
export default App;
```

13.组合缓存键

src\App.js

```
import React from 'react';
import { useQuery } from 'react-query';
import { ReactQueryDevtools } from 'react-query/devtools'
import request from './request';
+function User({ userId }) {
+  const { data, isLoading, isError, error, isFetching } = useQuery(['user', userId], () => request.get('/user', {
+    params: {
+      userId
+    }
+  }))
+  if (isLoading) return 加载中.......

+  if (isError) return {error.message}
+  return (
+    (
+      <>
+        {data.id ? {data.id}:{data.name} : {userId}对应的用户不存在}
+        {isFetching && 更在更新数据...}
+      </>
+    )
+  )
+}
function App() {
+ const [userId, setUserId] = React.useState('');
  return (
    <>
+      setUserId(event.target.value)} />
+


    </>
  )
}
export default App;
```

api.js

```
const express = require('express');
const cors = require('cors');
const logger = require('morgan');
const app = express();
app.use(express.json());
app.use(cors({
  allowedHeaders: ["Content-Type"],
  allowMethods: ["GET", "POST", "PUT", "DELETE", "OPTIONS"]
}));
app.use(logger('dev'));
const users = new Array(10).fill(true).map((item, index) => ({ id: String(index + 1), name: `name${index + 1}` }))
app.use((req, res, next) => {
  setTimeout(next, 1000);
});
app.get('/users', (req, res) => {
  res.json(users.map(user => ({ ...user, name: user.name + '#' + new Date().toLocaleString() })));
});
+app.get('/user', (req, res) => {
+  const userId = req.query.userId;
+  const user = users.find(user => user.id === userId);
+  if (user)
+    res.json(user);
+  else
+    res.json({})
+});
app.listen(8080, () => console.log('started on port 8080'));
```

14.enabled

字段 含义 取值 isIdle 是否空闲 true、false

src\App.js

```
import React from 'react';
import { useQuery } from 'react-query';
import { ReactQueryDevtools } from 'react-query/devtools'
import request from './request';

function User({ userId }) {
  const { data, isLoading, isError, error, isFetching, isIdle } = useQuery(['user', userId], () => request.get('/user', {
    params: {
      userId
    }
+ }), { enabled: !!userId })
  if (isIdle) return null;
  if (isLoading) return 加载中.......

  if (isError) return {error.message}
  return (
    (
      <>
        {data.id ? {data.id}:{data.name} : {userId}对应的用户不存在}
        {isFetching && 更在更新数据...}
      </>
    )
  )
}
function App() {
  const [userId, setUserId] = React.useState('');
  return (
    <>
      setUserId(event.target.value)} />

    </>
  )
}
export default App;
```

15.retry

src\App.js

```
import React from 'react';
import { useQuery } from 'react-query';
import { ReactQueryDevtools } from 'react-query/devtools'
import request from './request';
function User({ userId }) {
  const { data, isLoading, isError, error, isFetching, isIdle } = useQuery(['user', userId], () => request.get('/user', {
    params: {
      userId
    }
  }), {
    enabled: !!userId,
+   retry: 1,
+   retryDelay: 1000,
+   retryDelay: attemptIndex => Math.min(1000 * 2 ** attemptIndex, 30000),
  })
  if (isIdle) return null;
  if (isLoading) return 加载中.......

  if (isError) return {error.message}
  return (
    (
      <>
        {data.id ? {data.id}:{data.name} : {userId}对应的用户不存在}
        {isFetching && 更在更新数据...}
      </>
    )
  )
}
function App() {
  const [userId, setUserId] = React.useState('');
  return (
    <>
       setUserId(event.target.value)} />

    </>
  )
}
export default App;
`
```

api.js

```
const express = require('express');
const cors = require('cors');
const logger = require('morgan');
const app = express();
app.use(express.json());
app.use(cors({
  allowedHeaders: ["Content-Type"],
  allowMethods: ["GET", "POST", "PUT", "DELETE", "OPTIONS"]
}));
app.use(logger('dev'));
const users = new Array(10).fill(true).map((item, index) => ({ id: String(index + 1), name: `name${index + 1}` }))
app.use((req, res, next) => {
  setTimeout(next, 1000);
});
app.get('/users', (req, res) => {
  res.json(users.map(user => ({ ...user, name: user.name + '#' + new Date().toLocaleString() })));
});
app.get('/user', (req, res) => {
  const userId = req.query.userId;
  const user = users.find(user => user.id
  if (user)
    res.json(user);
  else
+   res.sendStatus(404)
});
app.listen(8080, () => console.log('started on port 8080'));
```

16.取消请求

- axios发送请求时，添加cancelToken (http://www.axios-js.com/zh-cn/docs/#%E5%8F%96%E6%B6%88)配置项可以用于取消请求
- CancelToken有一个source静态方法，调用之后返回一个对象，该对象包含一个token属性，用于标记请求和一个cancel方法，用于取消请求
- cancel取消请求方法在调用取消请求的时候可以将取消原因 message字符串传递进去，这样请求在被取消之后会被 catch捕获，你可以在这里将取消原因打印出来或者提示给用户，比如提示用户不要频繁点击发送请求
- get的 cancelToken放置在第二个参数的对象里面， post的 cancelToken放置在第三个参数对象里面

src\App.js

```
import React from 'react';
import { useQuery } from 'react-query';
import { ReactQueryDevtools } from 'react-query/devtools'
+import request, { CancelToken } from './request';

function User({ userId }) {
  const { data, isLoading, isError, error, isFetching, isIdle } = useQuery(['user', userId], () => {
+   const source = CancelToken.source();
+   const promise = request.get('/user', {
+     params: { userId }, cancelToken: source.token
+   }).catch(error => {
+     if (request.isCancel(error)) {
+       console.log(error.message);
+     }
+   });
+   promise.cancel = () => source.cancel('请求被React Query取消');
+   return promise;
  }, {
    enabled: !!userId,
    retry: 3,
    retryDelay: 1000,
    retryDelay: attemptIndex => Math.min(1000 * 2 ** attemptIndex, 30000),
  })
  if (isIdle) return null;
  if (isLoading) return 加载中.......

  if (isError) return {error.message}
  return (
    (
      <>
        {data.id ? {data.id}:{data.name} : {userId}对应的用户不存在}
        {isFetching && 更在更新数据...}
      </>
    )
  )
}
function App() {
  const [userId, setUserId] = React.useState('');
  return (
    <>
       setUserId(event.target.value)} />

    </>
  )
}
export default App;
```

src\request.js

```
+import axios, { CancelToken } from 'axios';
axios.interceptors.response.use(response => response.data);
axios.defaults.baseURL = 'http://localhost:8080';
export default axios;
+export { CancelToken }
```

api.js

```
const express = require('express');
const cors = require('cors');
const logger = require('morgan');
const app = express();
app.use(express.json());
app.use(cors({
  allowedHeaders: ["Content-Type"],
  allowMethods: ["GET", "POST", "PUT", "DELETE", "OPTIONS"]
}));
app.use(logger('dev'));
const users = new Array(10).fill(true).map((item, index) => ({ id: String(index + 1), name: `name${index + 1}` }))
app.use((req, res, next) => {
+ setTimeout(next, 1000 * req.query.userId);
});
app.get('/users', (req, res) => {
  res.json(users.map(user => ({ ...user, name: user.name + '#' + new Date().toLocaleString() })));
});
app.get('/user', (req, res) => {
  const userId = req.query.userId;
  const user = users.find(user => user.id
  if (user)
    res.json(user);
  else
    res.sendStatus(404)
});
app.listen(8080, () => console.log('started on port 8080'));
```

17.依赖查询

src\App.js

```
import React from 'react';
import { useQuery } from 'react-query';
import { ReactQueryDevtools } from 'react-query/devtools'
import request, { CancelToken } from './request';

function User({ userId }) {
  const { data, isLoading, isError, error, isFetching, isIdle } = useQuery(['user', userId], () => {
    const source = CancelToken.source();
    const promise = request.get('/user', {
      params: { userId }, cancelToken: source.token
    }).catch(error => {
      if (request.isCancel(error)) {
        console.log(error.message);
      }
    });
    promise.cancel = () => source.cancel('请求被React Query取消');
    return promise;
  }, {
    enabled: !!userId,
    retry: 3,
    retryDelay: 1000,
    retryDelay: attemptIndex => Math.min(1000 * 2 ** attemptIndex, 30000),
  })
+ const postsResult = useQuery(['posts', data?.id],
+   () => request.get(`/posts`, { params: { userId: data?.id } }),
+   {
+     enabled: !!(data?.id)
+   });
  if (isIdle) return null;
  if (isLoading) return 加载中.......

  if (isError) return {error.message}
  return (
    (
      <>
        {data.id ? {data.id}:{data.name} : {userId}对应的用户不存在}
        {isFetching && 更在更新数据...}
+       {postsResult.data && 帖子数:{postsResult.data.length}}
+       {postsResult.isFetching && '正在更新贴子数据...'}
      </>
    )
  )
}
function App() {
  const [userId, setUserId] = React.useState('');
  return (
    <>
       setUserId(event.target.value)} />

    </>
  )
}
export default App;
```

```
const express = require('express');
const cors = require('cors');
const logger = require('morgan');
const app = express();
app.use(express.json());
app.use(cors({
  allowedHeaders: ["Content-Type"],
  allowMethods: ["GET", "POST", "PUT", "DELETE", "OPTIONS"]
}));
app.use(logger('dev'));
const users = new Array(10).fill(true).map((item, index) => ({ id: String(index + 1), name: `name${index + 1}` }))
+const posts = new Array(10).fill(true).map((user, index) => ({ id: String(index + 1), title: `title${index + 1}`, userId: String(index + 1) }))
app.use((req, res, next) => {
  setTimeout(next, 1000 * req.query.userId);
});
app.get('/users', (req, res) => {
  res.json(users.map(user => ({ ...user, name: user.name + '#' + new Date().toLocaleString() })));
});
app.get('/user', (req, res) => {
  const userId = req.query.userId;
  const user = users.find(user => user.id
  if (user)
    res.json(user);
  else
    res.sendStatus(404)
});
+app.get('/posts', (req, res) => {
+  const userId = req.query.userId;
+  res.json(posts.filter(post => post.userId === userId));
+});
app.listen(8080, () => console.log('started on port 8080'));
```

18.初始化数据

src\App.js

```
import React from 'react';
import { useQuery } from 'react-query';
import { ReactQueryDevtools } from 'react-query/devtools'
import request from './request';
const initialUser = { id: "1" }
const initialPosts = [];
function User({ userId }) {
  const { data, isLoading, isError, error, isFetching, isIdle } = useQuery(['user', userId], () => request.get('/user', {
    params: { userId }
  }), {
    enabled: !!userId,
    retry: 3,
    retryDelay: 1000,
    retryDelay: attemptIndex => Math.min(1000 * 2 ** attemptIndex, 30000),
    initialData: initialUser,
    initialStale: false,
    staleTime: 5000
  })
  const postsResult = useQuery(['posts', data?.id],
    () => request.get(`/posts`, { params: { userId: data?.id } }),
    {
      enabled: !!(data?.id),
      initialData: initialPosts,
      initialStale: false,
      staleTime: 5000
    });
  if (isIdle) return null;
  if (isLoading) return 加载中.......

  if (isError) return {error.message}
  return (
    (
      <>
        {data.id ? {data.id}:{data.name} : {userId}对应的用户不存在}
        {isFetching && 更在更新数据...}
        {postsResult.data && 帖子数:{postsResult.data.length}}
        {postsResult.isFetching && '正在更新贴子数据...'}
      </>
    )
  )
}
function App() {
  const [userId, setUserId] = React.useState('');
  return (
    <>
       setUserId(event.target.value)} />

    </>
  )
}
export default App;
```

19.并发查询

src\App.js

```
import React from 'react';
+import { useQuery, useQueries } from 'react-query';
import { ReactQueryDevtools } from 'react-query/devtools'
import request from './request';
function User() {
+ const [usersQuery, postsQuery] = useQueries([
+   { queryKey: ['users'], queryFn: () => request.get('/users') },
+   { queryKey: ['posts'], queryFn: () => request.get(`/posts`) },
+ ]);
  return (
    (
      <>
+       {usersQuery.data && 用户数:{usersQuery.data.length}}
+       {postsQuery.data && 帖子数:{postsQuery.data.length}}
      </>
    )
  )
}
function App() {
  return (
    <>

    </>
  )
}
export default App;
```

api.js

```
const express = require('express');
const cors = require('cors');
const logger = require('morgan');
const app = express();
app.use(express.json());
app.use(cors({
  allowedHeaders: ["Content-Type"],
  allowMethods: ["GET", "POST", "PUT", "DELETE", "OPTIONS"]
}));
app.use(logger('dev'));
const users = new Array(10).fill(true).map((item, index) => ({ id: String(index + 1), name: `name${index + 1}` }))
const posts = new Array(10).fill(true).map((user, index) => ({ id: String(index + 1), title: `title${index + 1}`, userId: String(index + 1) }))
app.use((req, res, next) => {
  setTimeout(next, 1000 * 1);
});
app.get('/users', (req, res) => {
  res.json(users.map(user => ({ ...user, name: user.name + '#' + new Date().toLocaleString() })));
});
app.get('/user', (req, res) => {
  const userId = req.query.userId;
  const user = users.find(user => user.id
  if (user)
    res.json(user);
  else
    res.sendStatus(404)
});
app.get('/posts', (req, res) => {
+  res.json(posts);
});
app.listen(8080, () => console.log('started on port 8080'));
```

20.列表和详情

src\App.js

```
import React from 'react';
import { useQuery } from 'react-query';
import { ReactQueryDevtools } from 'react-query/devtools'
import request from './request';
+function Users({ setUserId }) {
+  const usersResult = useQuery('users', () => request.get('/users'), { staleTime: 5000 });
+  if (usersResult.isLoading) {
+    return '用户列表加载中......';
+  }
+  return (
+    <>
+      用户列表
+
+        {
+          usersResult.data?.map(user =>  setUserId(user.id)}>{user.name})
+        }
+
+    </>
+  )
+}

+function User({ userId, setUserId }) {
+  const userResult = useQuery(['user', userId], () => request.get('/user', {
+    params: { userId }
+  }), { staleTime: 5000 });
+  if (userResult.isLoading) {
+    return '单个用户加载中......';
+  }
+  return (
+
+       setUserId(-1)}>返回
+      {userResult.data && ID:{userResult.data.id},NAME:{userResult.data.name}}
+
+  )
+}
function App() {
+ const [userId, setUserId] = React.useState(-1);
+ return (
+   <>
+     {
+       userId > -1 ? (
+
+       ) :
+     }
+
+   </>
+ )
}
export default App;
```

21.读取查询缓存

- QueryCache (https://react-query.tanstack.com/reference/QueryCache)是 React Query 的存储机制。它存储它包含的所有数据、元信息和查询状态
- 通常，您不会直接与 `QueryCache` 交互，而是使用 `QueryClient`

src\App.js

```
import React from 'react';
+import { useQuery, useQueryClient } from 'react-query';
import { ReactQueryDevtools } from 'react-query/devtools'
import request from './request';
function Users({ setUserId }) {
  const usersResult = useQuery('users', () => request.get('/users'), { staleTime: 5000 });
  if (usersResult.isLoading) {
    return '用户列表加载中......';
  }
  return (
    <>
      用户列表

        {
          usersResult.data?.map(user =>  setUserId(user.id)}>{user.name})
        }

    </>
  )
}

function User({ userId, setUserId }) {
  const queryClient = useQueryClient();
  const userResult = useQuery(['user', userId], () => request.get('/user', {
    params: { userId }
  }), {
    staleTime: 5000,
+   initialData: () => queryClient.getQueryData('users')?.find(user => user.id === userId),
+   initialStable: false
  });
  if (userResult.isLoading) {
    return '单个用户加载中......';
  }
  return (

       setUserId(-1)}>返回
      {userResult.data && ID:{userResult.data.id},NAME:{userResult.data.name}}

  )
}
function App() {
  const [userId, setUserId] = React.useState(-1);
  return (
    <>
      {
        userId > -1 ? (

        ) :
      }

    </>
  )
}
export default App;
```

22.预缓存

src\App.js

```
import React from 'react';
import { useQuery, useQueryClient } from 'react-query';
import { ReactQueryDevtools } from 'react-query/devtools'
import request from './request';
function Users({ setUserId }) {
+  const queryClient = useQueryClient();
+ const usersResult = useQuery('users', async () => {
+   const users = await request.get('/users');
+   users.forEach(user => {
+     queryClient.setQueryData(['user', user.id], user);
+   });
+   return users;
  }, { staleTime: 5000 });
  if (usersResult.isLoading) {
    return '用户列表加载中......';
  }
  return (
    <>
      用户列表

        {
          usersResult.data?.map(user =>  setUserId(user.id)}>{user.name})
        }

    </>
  )
}

function User({ userId, setUserId }) {
- const queryClient = useQueryClient();
  const userResult = useQuery(['user', userId], () => request.get('/user', {
    params: { userId }
  }), {
-    staleTime: 5000,
-    initialData: () => queryClient.getQueryData('users')?.find(user => user.id === userId),
-    initialStable: false
  });
  if (userResult.isLoading) {
    return '单个用户加载中......';
  }
  return (

       setUserId(-1)}>返回
      {userResult.data && ID:{userResult.data.id},NAME:{userResult.data.name}}

  )
}
function App() {
  const [userId, setUserId] = React.useState(-1);
  return (
    <>
      {
        userId > -1 ? (

        ) :
      }

    </>
  )
}
export default App;
```

23.副作用

src\App.js

```
import React from 'react';
import { useQuery, useQueryClient } from 'react-query';
import { ReactQueryDevtools } from 'react-query/devtools'
+import { ToastContainer, toast } from 'react-toastify';
+import 'react-toastify/dist/ReactToastify.css';
import request from './request';
function Users({ setUserId }) {
  const queryClient = useQueryClient();
  const usersResult = useQuery('users', async () => {
    const users = await request.get('/users');
    users.forEach(user => {
      queryClient.setQueryData(['user', user.id], user);
    });
    return users;
  }, {
+   onSuccess(data) {
+     //console.log('查询成功', data);
+     toast("查询成功!")
+   },
+   onError(error) {
+     //console.log('查询失败', error);
+     toast("查询失败!")
+   },
+   onSettled(data, error) {
+     console.log('查询结束');
+   }
  });
  if (usersResult.isLoading) {
    return '用户列表加载中......';
  }
  return (
    <>
      用户列表

        {
          usersResult.data?.map(user =>  setUserId(user.id)}>{user.name})
        }

    </>
  )
}

function User({ userId, setUserId }) {
  //  const queryClient = useQueryClient();
  const userResult = useQuery(['user', userId], () => request.get('/user', {
    params: { userId }
  }));
  if (userResult.isLoading) {
    return '单个用户加载中......';
  }
  return (

       setUserId(-1)}>返回
      {userResult.data && ID:{userResult.data.id},NAME:{userResult.data.name}}

  )
}
function App() {
  const [userId, setUserId] = React.useState(-1);
  return (
    <>
      {
        userId > -1 ? (

        ) :
      }
+

    </>
  )
}
export default App;
```

23.预查询

src\App.js

```
import React from 'react';
import { useQuery, useQueryClient } from 'react-query';
import { ReactQueryDevtools } from 'react-query/devtools'
import { ToastContainer, toast } from 'react-toastify';
import 'react-toastify/dist/ReactToastify.css';
import request from './request';
+function fetchUsers() {
+  return request.get('/users');
+}
function Users({ setUserId }) {
  const usersResult = useQuery('users', fetchUsers);
  if (usersResult.isLoading) {
    return '用户列表加载中......';
  }
  return (
    <>
      用户列表

        {
          usersResult.data?.map(user =>  setUserId(user.id)}>{user.name})
        }

    </>
  )
}

function App() {
+ const queryClient = useQueryClient();
+ const [show, setShow] = React.useState(false);
+ React.useEffect(() => {
+   queryClient.prefetchQuery('users', fetchUsers, { staleTime: 5000 });
+ })
  return (
    <>
+      setShow(!show)} onMouseOver={() =>
+       queryClient.prefetchQuery('users', fetchUsers, { staleTime: 5000 })}>show
+     {
+       show &&
+     }

    </>
  )
}
export default App;
```

24.变更操作

src\App.js

```
import React from 'react';
+import { useQuery, useQueryClient, useMutation } from 'react-query';
import { ReactQueryDevtools } from 'react-query/devtools'
import request from './request';
function fetchUsers() {
  return request.get('/users');
}
function Users({ setUserId }) {
  const usersResult = useQuery('users', fetchUsers);
  if (usersResult.isLoading) {
    return '用户列表加载中......';
  }
  return (
    <>
      用户列表

        {
          usersResult.data?.map(user =>  setUserId(user.id)}>{user.name})
        }

    </>
  )
}

function App() {
+ const nameRef = React.useRef();
+ const queryClient = useQueryClient();
+ const { mutate, isLoading, isError, isSuccess, error } = useMutation(
+   (values) => request.post('/users', values), {
+   onSuccess() {
+     //queryClient.invalidateQueries('users');
+   },
+   onError(error) {
+     //alert(error.response.data.message);
+   },
+   onSettled(data, error) {
+     queryClient.invalidateQueries('users');
+   }
+ });
+ const handleSubmit = (event) => {
+   event.preventDefault();
+   const name = nameRef.current.value;
+   const user = { name };
+   mutate(user);
+ }
  return (
    <>

+
+
+
+
+     {isError && {error.response.data.message}}

    </>
  )
}
export default App;
```

api.js

```
const express = require('express');
const cors = require('cors');
const logger = require('morgan');
const app = express();
app.use(express.json());
app.use(cors({
  allowedHeaders: ["Content-Type"],
  allowMethods: ["GET", "POST", "PUT", "DELETE", "OPTIONS"]
}));
app.use(logger('dev'));
const users = new Array(10).fill(true).map((item, index) => ({ id: String(index + 1), name: `name${index + 1}` }))
const posts = new Array(10).fill(true).map((user, index) => ({ id: String(index + 1), title: `title${index + 1}`, userId: String(index + 1) }))
app.use((req, res, next) => {
  setTimeout(next, 1000 * 1);
});
app.get('/users', (req, res) => {
  res.json(users.map(user => ({ ...user, name: user.name + '#' + new Date().toLocaleString() })));
});
app.get('/user', (req, res) => {
  const userId = req.query.userId;
  const user = users.find(user => user.id
  if (user)
    res.json(user);
  else
    res.sendStatus(404)
});
+app.post('/users', (req, res) => {
+  let user = req.body;
+  if (user.name) {
+    user.id = (users.length + 1) + '';
+    user.createdAt = new Date().toLocaleDateString();
+    users.push(user);
+    res.json(user);
+  } else {
+    res.status(400).send({ message: '用户名不能为空!' });
+  }
+});
app.get('/posts', (req, res) => {
  res.json(posts);
});
app.listen(8080, () => console.log('started on port 8080'));
```

25.乐观更新

- [useMutation (https://react-query.tanstack.com/reference/useMutation#_top)](https://react-query.tanstack.com/reference/useMutation#_top)
- onMutate函数将在突变函数被触发之前触发，并传递突变函数将接收的相同变量
- 用于对资源执行乐观更新以希望突变成功
- 如果发生突变失败，从此函数返回的值将传递给onError和onSettled函数，并且可用于回滚乐观更新

src\App.js

```
import React from 'react';
import { useQuery, useQueryClient, useMutation } from 'react-query';
import { ReactQueryDevtools } from 'react-query/devtools'
import request from './request';
function fetchUsers() {
  return request.get('/users');
}
function Users({ setUserId }) {
  const usersResult = useQuery('users', fetchUsers);
  if (usersResult.isLoading) {
    return '用户列表加载中......';
  }
  return (
    <>
      用户列表

        {
          usersResult.data?.map(user =>  setUserId(user.id)}>{user.name})
        }

    </>
  )
}

function App() {
  const nameRef = React.useRef();
  const queryClient = useQueryClient();
  const { mutate:saveUser, isLoading, isError, isSuccess, error } = useMutation(
    (values) => request.post('/users', values), {
+    onMutate(values) {
+      queryClient.setQueryData('users', oldUsers => [...oldUsers, { ...values, id: String(Date.now()) }]);
+    },
    onSuccess() {
      //queryClient.invalidateQueries('users');
    },
    onError(error) {
      //alert(error.response.data.message);
    },
    onSettled(data, error) {
      queryClient.invalidateQueries('users');
    }
  });
  const handleSubmit = (event) => {
    event.preventDefault();
    const name = nameRef.current.value;
    const user = { name };
    saveUser(user);
  }
  return (
    <>

      {isError && {error.response.data.message}}

    </>
  )
}
export default App;
```

```
const express = require('express');
const cors = require('cors');
const logger = require('morgan');
const app = express();
app.use(express.json());
app.use(cors({
  allowedHeaders: ["Content-Type"],
  allowMethods: ["GET", 'POST', "PUT", "DELETE", "OPTIONS"]
}));
app.use(logger('dev'));
const users = new Array(10).fill(true).map((item, index) => ({ id: String(index + 1), name: `name${index + 1}` }))
const posts = new Array(10).fill(true).map((user, index) => ({ id: String(index + 1), title: `title${index + 1}`, userId: String(index + 1) }))
app.use((req, res, next) => {
  setTimeout(next, 1000 * 1);
});
app.get('/users', (req, res) => {
+ res.json(users);
});
app.get('/user', (req, res) => {
  const userId = req.query.userId;
  const user = users.find(user => user.id
  if (user)
    res.json(user);
  else
    res.sendStatus(404)
});
app.post('/users', (req, res) => {
  let user = req.body;
  if (user.name) {
    user.id = (users.length + 1) + '';
    user.createdAt = new Date().toLocaleDateString();
    users.push(user);
    res.json(user);
  } else {
    res.status(400).send({ message: '用户名不能为空!' });
  }
});
app.get('/posts', (req, res) => {
  res.json(posts);
});
app.listen(8080, () => console.log('started on port 8080'));
```

26.失败回滚

src\App.js

```
import React from 'react';
import { useQuery, useQueryClient, useMutation } from 'react-query';
import { ReactQueryDevtools } from 'react-query/devtools'
import request from './request';
function fetchUsers() {
  return request.get('/users');
}
function Users() {
  const usersResult = useQuery('users', fetchUsers);
  if (usersResult.isLoading) {
    return '用户列表加载中......';
  }
  return (
    <>
      用户列表

        {
          usersResult.data?.map(user => {user.name})
        }

    </>
  )
}

function App() {
  const nameRef = React.useRef();
  const queryClient = useQueryClient();
+ const { mutate: saveUser, reset, isLoading, isError, isSuccess, error } = useMutation(
    (values) => request.post('/users', values), {
+   retry: false,
+   onMutate(values) {
+     queryClient.cancelQueries('users');
+     const oldUsers = queryClient.getQueryData('users');
+     queryClient.setQueryData('users', oldUsers => [...oldUsers, { ...values, id: String(Date.now+)) }]);
+     //return oldUsers;
+     return () => queryClient.setQueryData('users', oldUsers)
+   },
+   onSuccess(data) {
+     queryClient.setQueryData('users', oldUsers => oldUsers.map((user, index) => {
+       if (index === oldUsers.length - 1) {
+         return data;
+       }
+       return user;
+     }));
+     //queryClient.invalidateQueries('users');
+   },
+   onError(error, values, rollback) {
+     //alert(error.response.data.message);
+     //queryClient.setQueryData('users', rollbackValue);
+     console.log(rollback);
+     rollback && rollback();
+   },
+   onSettled(data, error) {
+     queryClient.invalidateQueries('users');
+     setTimeout(() => {
+       nameRef.current.value = ''
+       reset();
+     }, 3000)
+   }
  });
  const handleSubmit = (event) => {
    event.preventDefault();
    const name = nameRef.current.value;
    const user = { name };
    saveUser(user);
  }
  return (
    <>

      {isError && {error.response.data.message}}

    </>
  )
}
export default App;
```

27.分页查询

src\App.js

```
import React from 'react';
import { useQuery, useQueryClient, useMutation } from 'react-query';
import { ReactQueryDevtools } from 'react-query/devtools'
import request from './request';
+function fetchUsers({ queryKey: [_, { pageNumber }] }) {
+  return request.get('/users', {
+    params: {
+      pageNumber
+    }
+  });
+}
function Users() {
+ const queryClient = useQueryClient();
+ const [pageNumber, setPageNumber] = React.useState(1);
+ const usersResult = useQuery(['users', { pageNumber }], fetchUsers, {
+   onSuccess() {
+     queryClient.prefetchQuery(['users', { pageNumber: pageNumber + 1 }], fetchUsers);
+   }
+ });
  return (
    <>
      用户列表

        {
+         usersResult.data?.data.map(user => {user.name})
        }

+      { setPageNumber(pageNumber => pageNumber - 1)}>上一页}
+      {pageNumber}
+      {= usersResult.data?.totalPage} onClick={() => setPageNumber(pageNumber => pageNumber + 1)}>下一页}
    </>
  )
}

function App() {
  const nameRef = React.useRef();
  const queryClient = useQueryClient();
  const { mutate: saveUser, reset, isLoading, isError, isSuccess, error } = useMutation(
    (values) => request.post('/users', values), {
    retry: false,
    onMutate(values) {
      queryClient.cancelQueries('users');
      const oldUsers = queryClient.getQueryData('users');
      queryClient.setQueryData('users', oldUsers => [...oldUsers, { ...values, id: String(Date.now()) }]);
      //return oldUsers;
      return () => queryClient.setQueryData('users', oldUsers)
    },
    onSuccess(data) {
      queryClient.setQueryData('users', oldUsers => oldUsers.map((user, index) => {
        if (index
          return data;
        }
        return user;
      }));
      //queryClient.invalidateQueries('users');
    },
    onError(error, values, rollback) {
      //alert(error.response.data.message);
      //queryClient.setQueryData('users', rollbackValue);
      console.log(rollback);
      rollback && rollback();
    },
    onSettled(data, error) {
      queryClient.invalidateQueries('users');
      setTimeout(() => {
        nameRef.current.value = ''
        reset();
      }, 3000)
    }
  });
  const handleSubmit = (event) => {
    event.preventDefault();
    const name = nameRef.current.value;
    const user = { name };
    saveUser(user);
  }
  return (
    <>

      {isError && {error.response.data.message}}

    </>
  )
}
export default App;
```

api.js

```
const express = require('express');
const cors = require('cors');
const logger = require('morgan');
const app = express();
app.use(express.json());
app.use(cors({
  allowedHeaders: ["Content-Type"],
  allowMethods: ["GET", 'POST', "PUT", "DELETE", "OPTIONS"]
}));
app.use(logger('dev'));
+const users = new Array(30).fill(true).map((item, index) => ({ id: String(index + 1), name: `name${index + 1}` }))
const posts = new Array(30).fill(true).map((user, index) => ({ id: String(index + 1), title: `title${index + 1}`, userId: String(index + 1) }))
app.use((req, res, next) => {
  setTimeout(next, 1000 * 1);
});
app.get('/users', (req, res) => {
+ const pageNumber = Number(req.query.pageNumber);
+ const totalPage = Math.floor(users.length / 10);
+ const offset = (pageNumber - 1) * 10;
+ res.json({
+   data: users.slice(offset, offset + 10),
+   pageNumber,
+   totalPage
+ });
});
app.get('/user', (req, res) => {
  const userId = req.query.userId;
  const user = users.find(user => user.id
  if (user)
    res.json(user);
  else
    res.sendStatus(404)
});
app.post('/users', (req, res) => {
  let user = req.body;
  if (user.name) {
    user.id = (users.length + 1) + '';
    user.createdAt = new Date().toLocaleDateString();
    users.push(user);
    res.json(user);
  } else {
    res.status(400).send({ message: '用户名不能为空!' });
  }
});
app.get('/posts', (req, res) => {
  res.json(posts);
});
app.listen(8080, () => console.log('started on port 8080'));
```

28.无限分页

src\App.js

```
import React from 'react';
+import { useInfiniteQuery } from 'react-query';
import { ReactQueryDevtools } from 'react-query/devtools'
import request from './request';
+function fetchUsers({ pageParam = 1 }) {
+  return request.get('/users', {
+    params: {
+      pageNumber: pageParam
+    }
+  });
+}
function Users() {
+ const { data, hasNextPage, fetchNextPage } = useInfiniteQuery(['users'], fetchUsers, {
+   getNextPageParam: (lastPageData) => {
+     console.log(lastPageData);
+     return lastPageData.pageNumber < lastPageData.totalPage ? lastPageData.pageNumber + 1 : false;
+   }
+ });
  return (
    <>
      用户列表

+       {
+         data?.pages?.map((page, index) => {
+           return (
+
+               {
+                 page.data?.map(user => {user.id}:{user.name})
+               }
+
+           )
+         })
+       }

+      fetchNextPage()}>加载更多
    </>
  )
}

function App() {
  return (
    <>

    </>
  )
}
export default App;
```