React Query

Terms

Caching
Mutations
Infinite queries
Paginated queries
Parameterized queries
Optimistic updates
Query key
Query function

Summary

- React Query is a library that simplifies fetching, caching, and updating data in React applications.
- It provides two primary hooks for fetching and updating data: **useQuery** and **useMutation**.
- To create a query object, we provide a **queryKey** that identifies the query and a **queryFn** that fetches the data.
- Query objects have three important properties: **data**, **error**, and **isLoading**.
- To provide better separation of concerns and reuse opportunities, we can encapsulate our queries in custom hooks.
- React Query provides a set of DevTools that can be used to inspect and debug your queries.
- React Query provides a number of options that can be used to customize the behavior of our queries, such as caching options and query retry behavior. These options can be set globally or on a per-query basis.

- React Query supports parameterized queries, where query parameters can be passed as arguments to the **useQuery** hook.
- React Query provides built-in support for paginated and infinite queries.
- To create a mutation object, we provide a **mutationFn** that mutates the data.
- Mutation objects have callbacks such as **onSuccess**, **onError** and **onMutate** (used for optimistic updates).

CREATING A QUERY

```
const { data, error, isLoading } = useQuery<Todo[], Error>({
  queryKey: ['todos'],
  queryFn: () => axios.get('').then((res) => res.data),
});
```

CUSTOMIZING QUERY SETTINGS

```
// Globally
const client = new QueryClient({
  defaultOptions: {
    queries: {
      retry: 3,
      cacheTime: 300_000, // 5m
      staleTime: 10 * 1000, // 10s
    },
  },
});

// Per query
const { data } = useQuery({
  staleTime: 10 * 1000, // 10s
});
```

PARAMETERIZED QUERIES

```
const { data } = useQuery<Todo[], Error>({
  queryKey: ['users', userId, 'todos'],
});
```


CREATING A MUTATION

```
const queryClient = useQueryClient();

const { error, isLoading } = useMutation<Todo, Error, Todo>({
  mutationFn: () => axios.post('').then((res) => res.data),
  onSuccess: (savedTodo, newTodo) => {
    // Approach 1: Invalidate the cache
    queryClient.invalidateQueries({ queryKey: ['todos'] });

    // Approach 2: Update the cache
    queryClient.setQueryData<Todo[]>(['todos'], (todos = []) => [
      savedTodo,
      ...todos,
    ]);
  },
});
```

