Using React Query to improve UX

Nicolas Jardim

UI Engineer at Mollie

What is this talk?

- A beginner's introduction to RQ
- Some features of RQ
- Cool demo

React Query

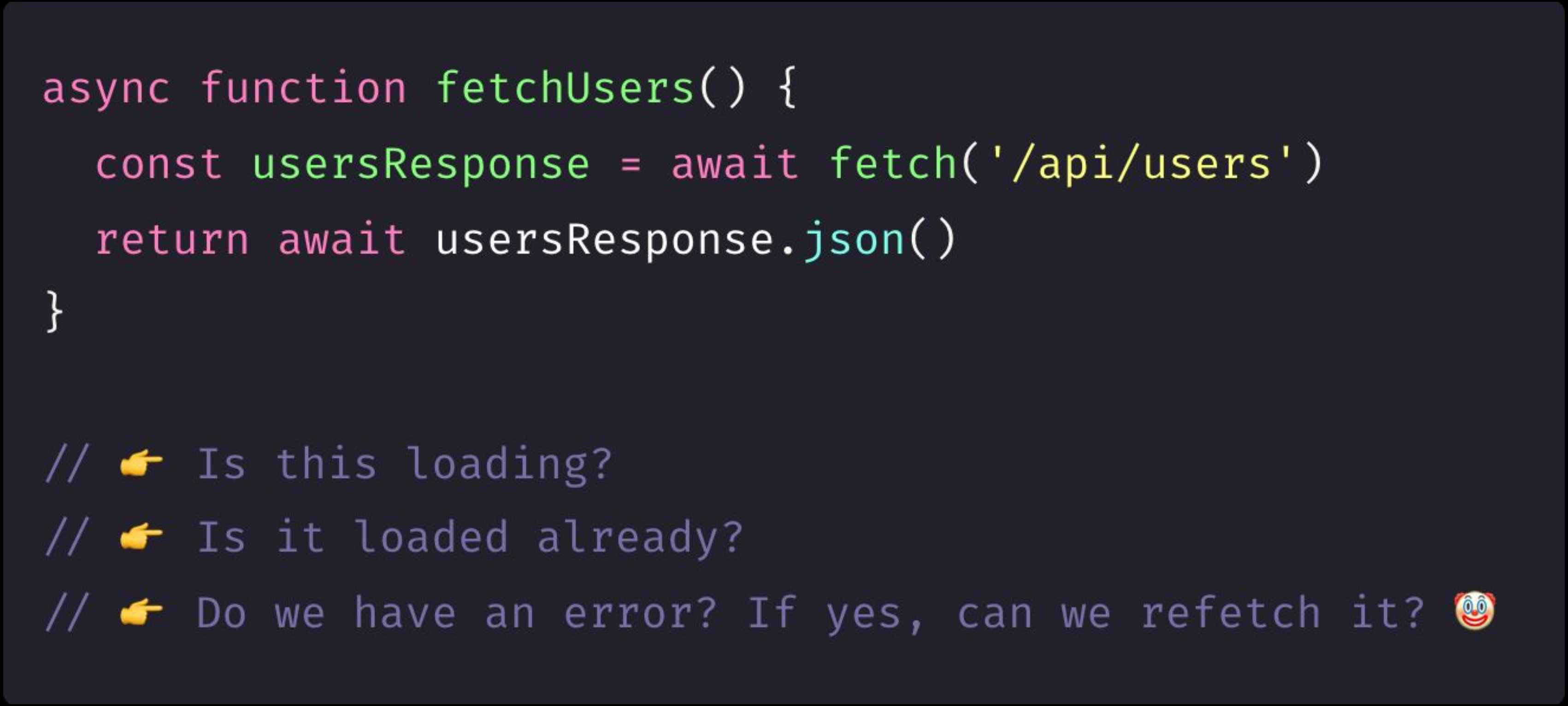
🔁

TanStack Query

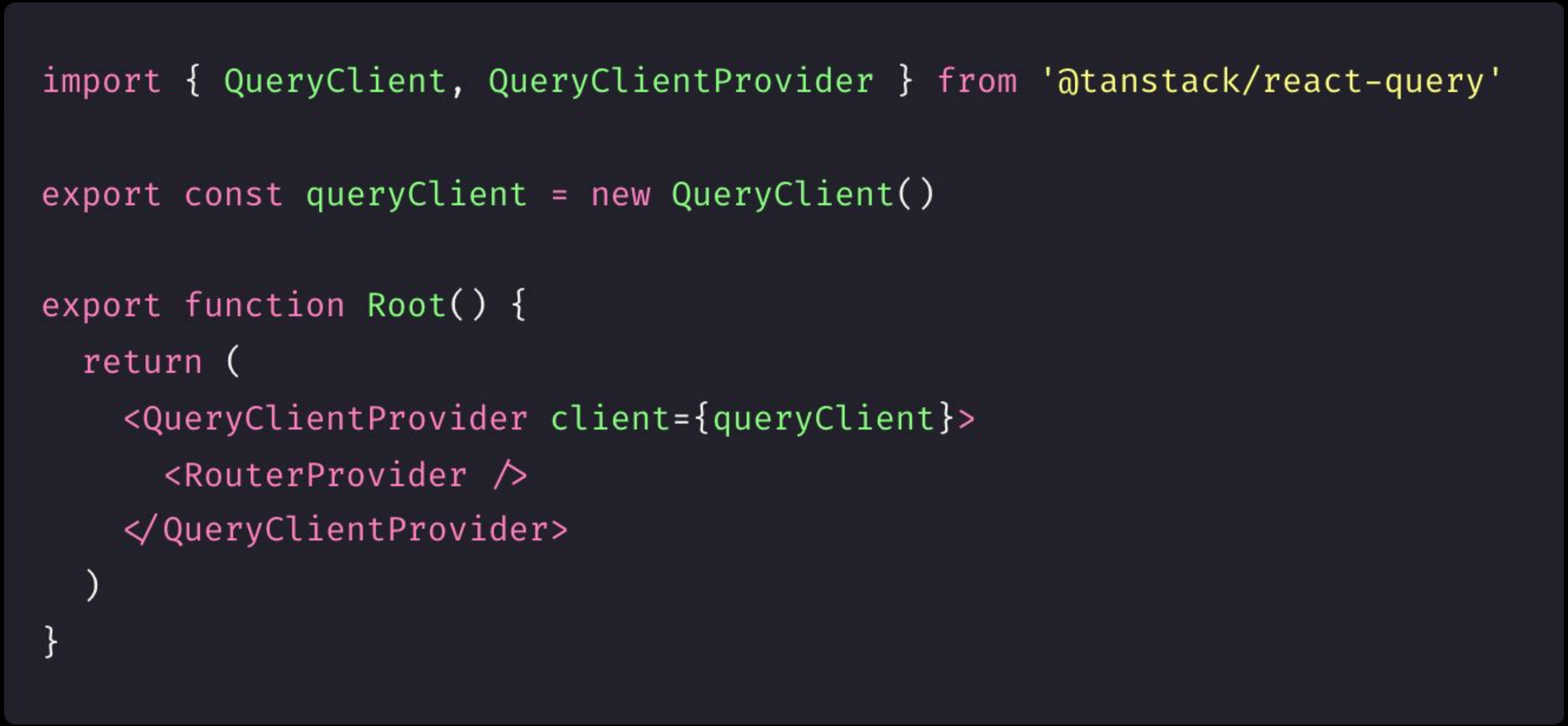
What is React Query?

Fetch,cache and synchronize data from a server.

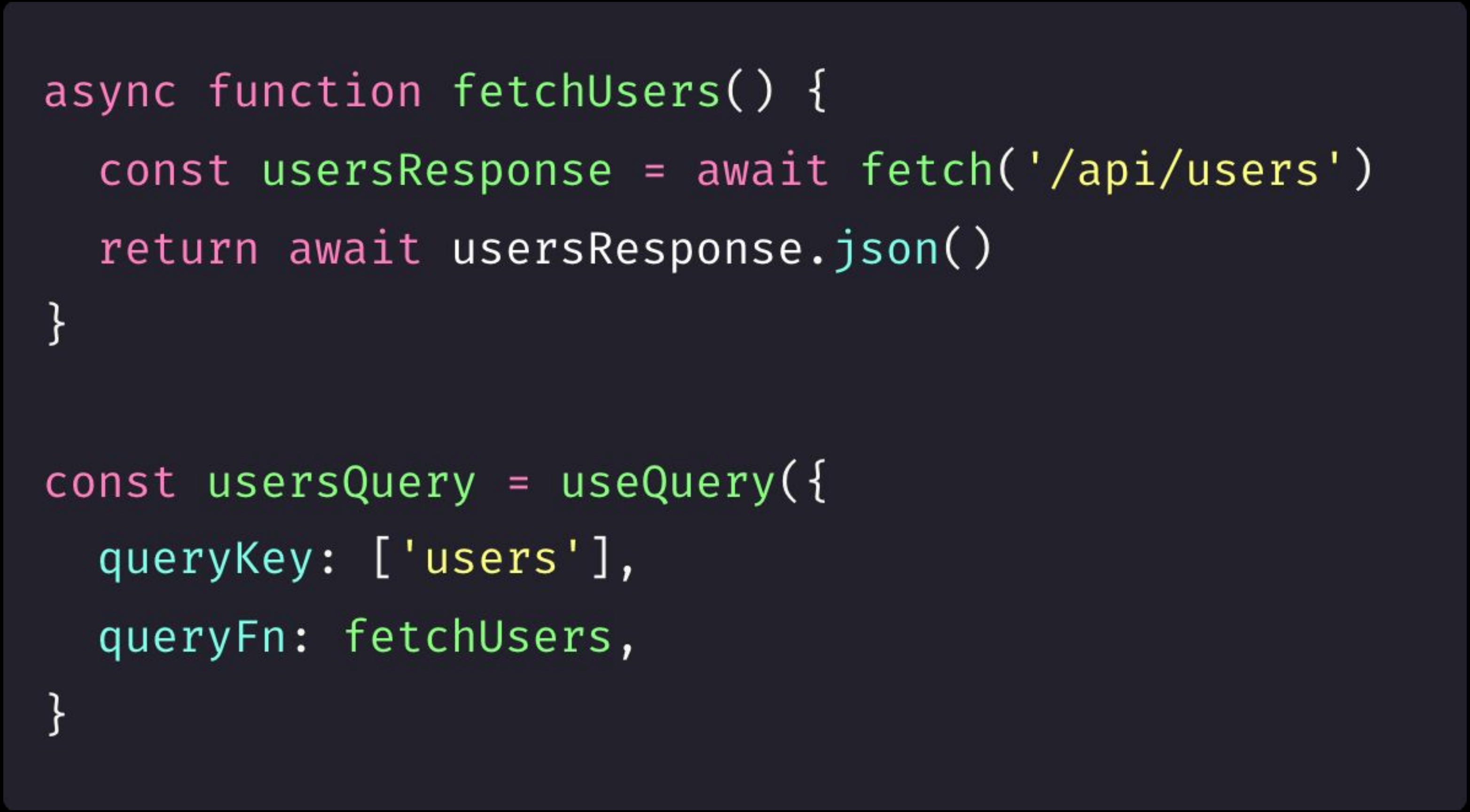
Async State Manager



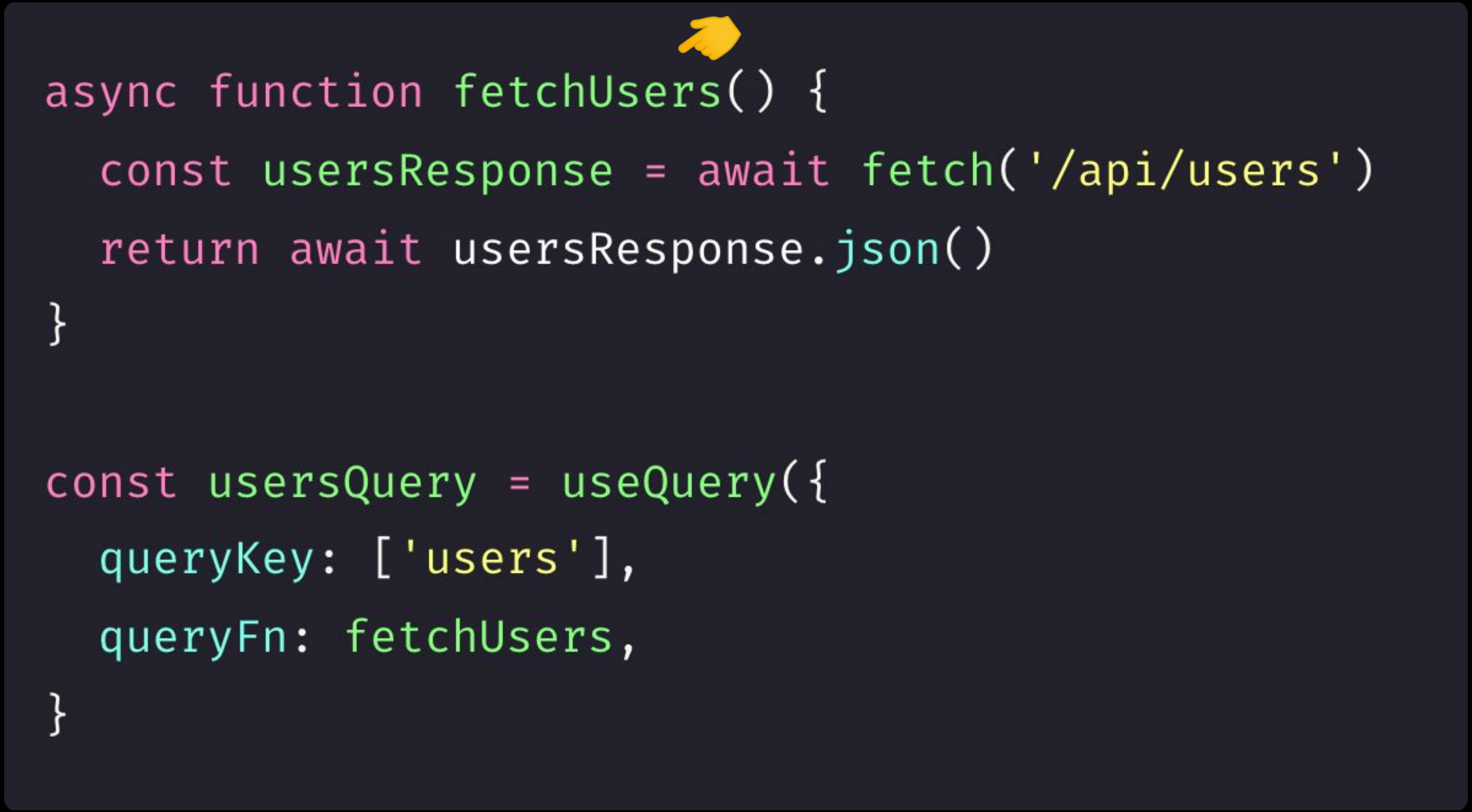
```
async function fetchUsers() {
  const usersResponse = await fetch('/api/users')
  return await usersResponse.json()
}


// 👉 Is this loading?
// 👉 Is it loaded already?
// 👉 Do we have an error? If yes, can we refetch it? 🤡
```



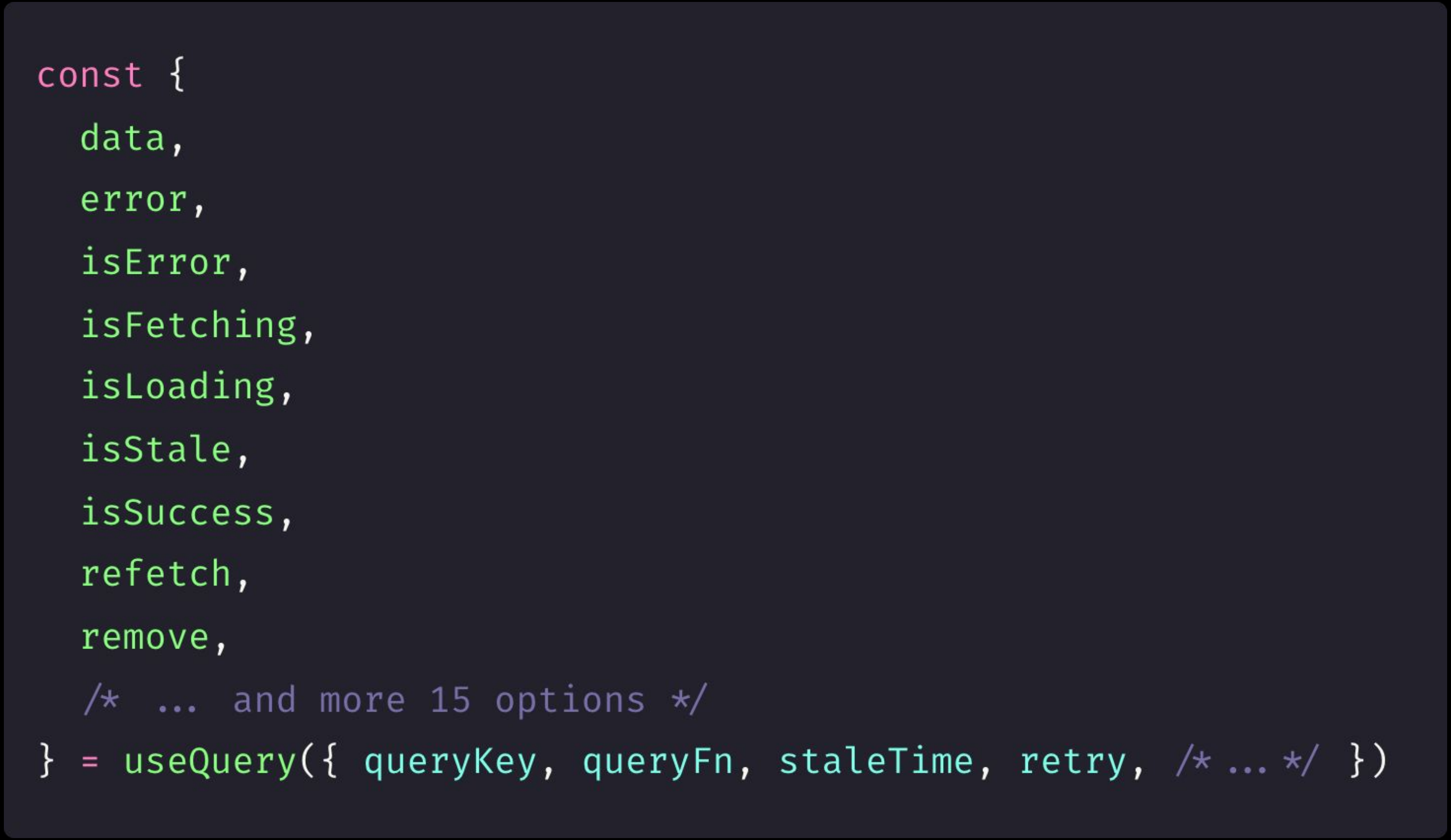
```
import { QueryClient, QueryClientProvider } from '@tanstack/react-query'

export const queryClient = new QueryClient()

export function Root() {
  return (
    <QueryClientProvider client={queryClient}>
      <RouterProvider />
    </QueryClientProvider>
  )
}
```



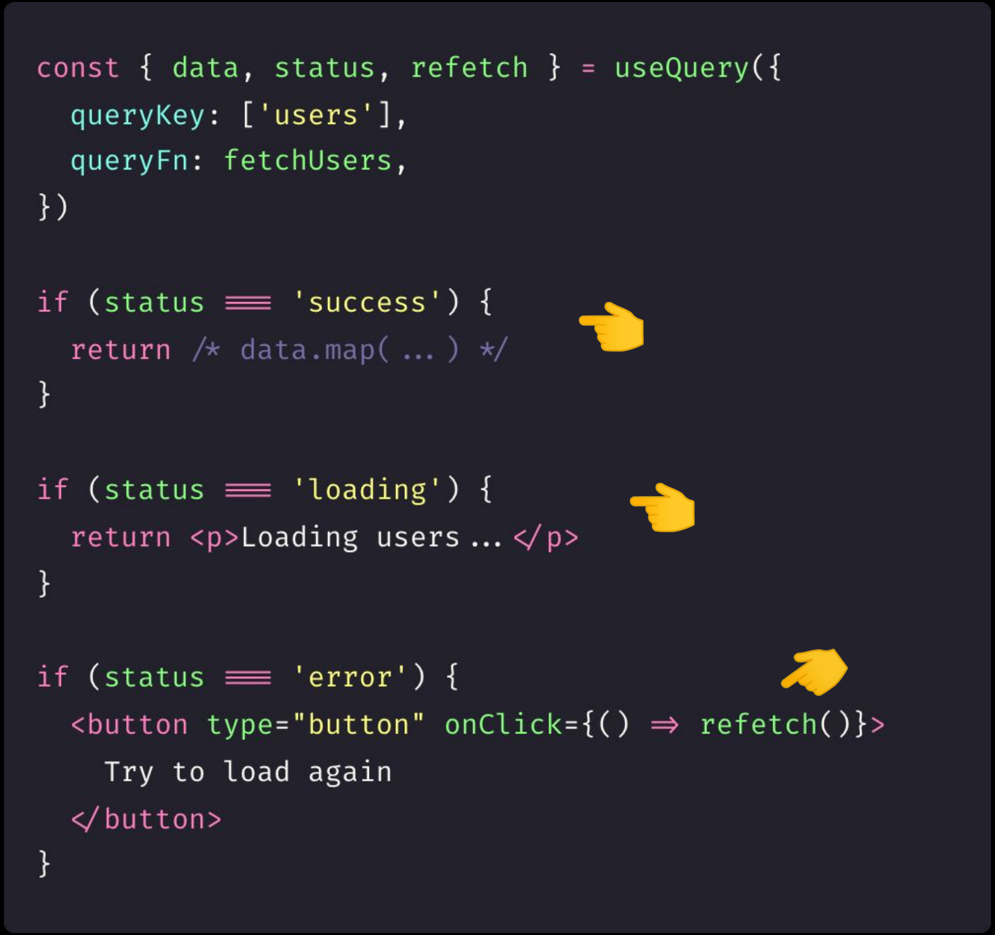
```
async function fetchUsers() {
  const usersResponse = await fetch('/api/users')
  return await usersResponse.json()
}

const usersQuery = useQuery({
  queryKey: ['users'],
  queryFn: fetchUsers,
}
```



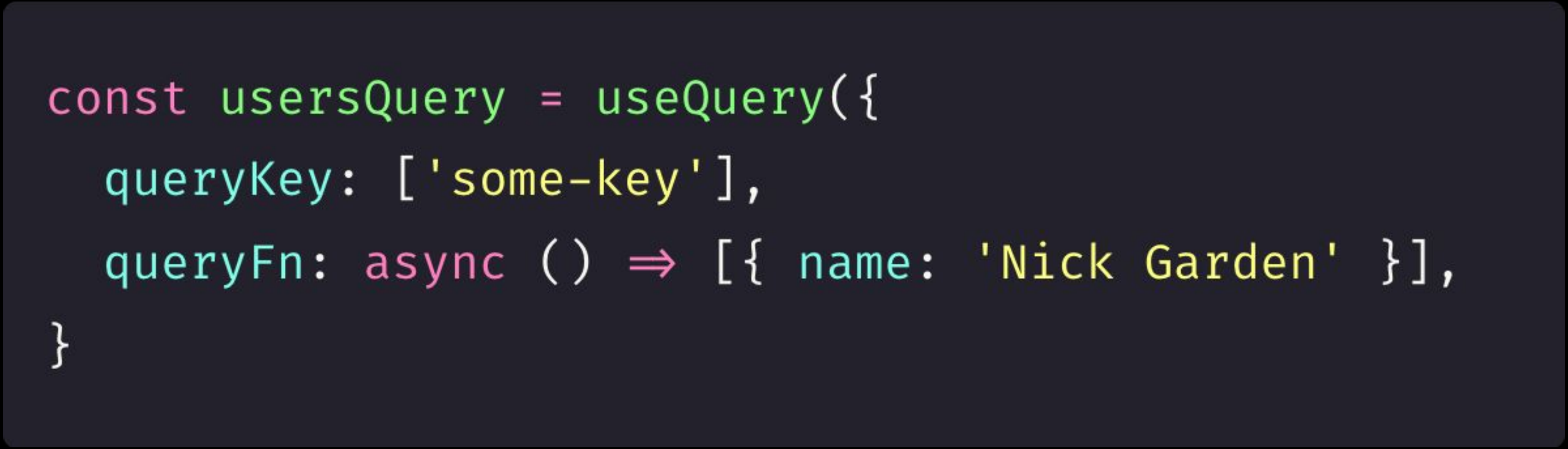
```
async function fetchUsers() {
  const usersResponse = await fetch('/api/users')
  return await usersResponse.json()
}

const usersQuery = useQuery({
  queryKey: ['users'],
  queryFn: fetchUsers,
}
```



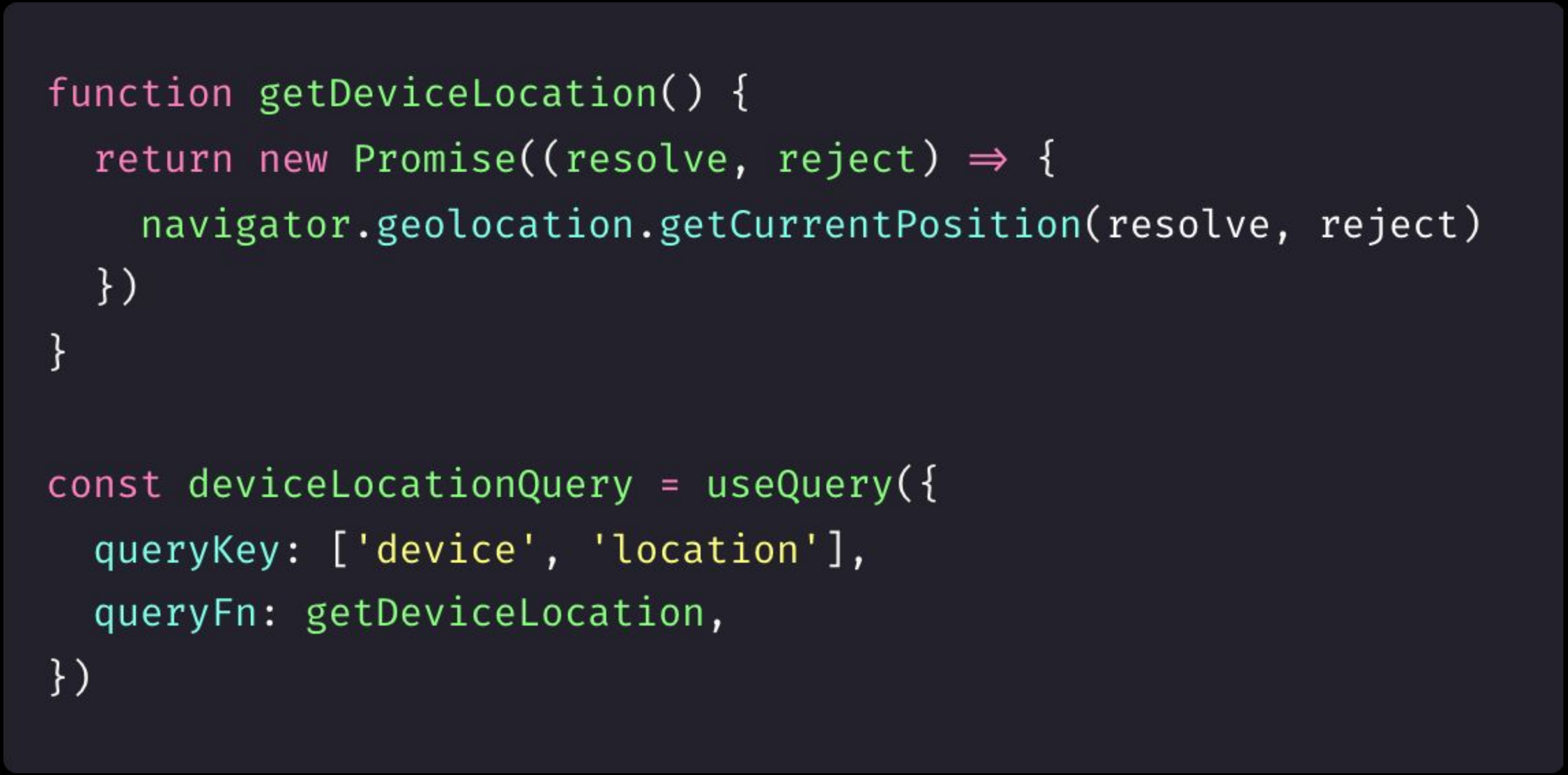
```
const {
  data,
  error,
  isError,
  isFetching,
  isLoading,
  isStale,
  isSuccess,
  refetch,
  remove,
  /*  ...  and more 15 options */
} = useQuery({ queryKey, queryFn, staleTime, retry, /* ... */ })
```



```
const { data, status, refetch } = useQuery({
  queryKey: ['users'],
  queryFn: fetchUsers,
})

if (status === 'success') {
  return /* data.map( ... ) */
}

if (status === 'loading') {
  return <p>Loading users...</p>
}

if (status === 'error') {
  <button type="button" onClick={() => refetch()}>
    Try to load again
  </button>
}
```



```
const usersQuery = useQuery({
  queryKey: ['some-key'],
  queryFn: async () => [{ name: 'Nick Garden' }],
}
```



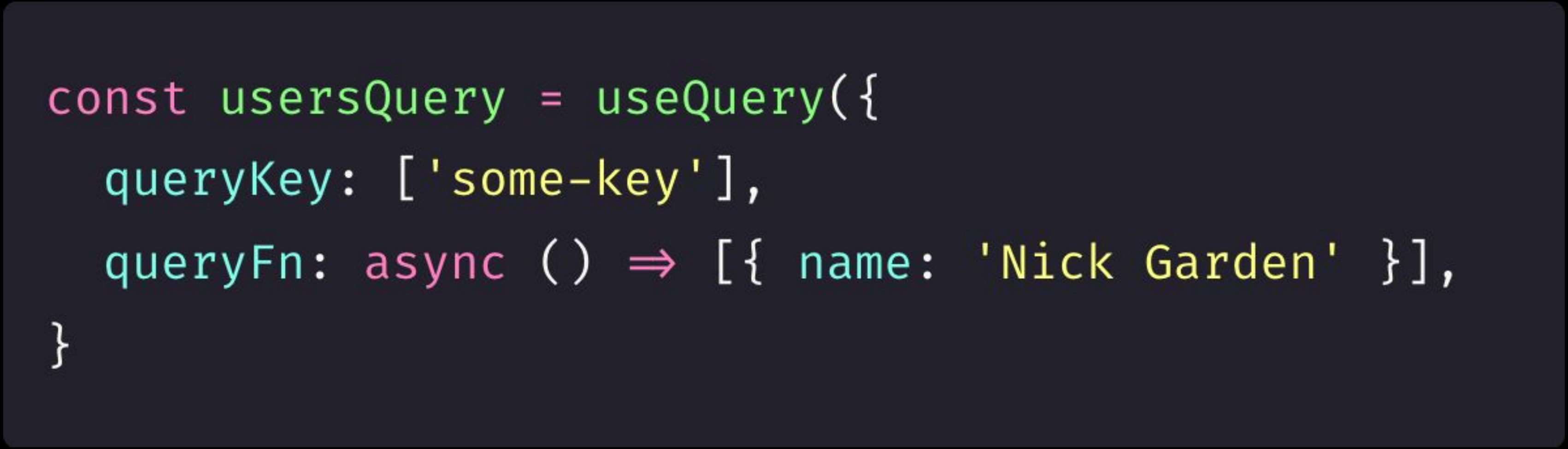
```javascript
function getDeviceLocation() {
  return new Promise((resolve, reject) => {
    navigator.geolocation.getCurrentPosition(resolve, reject)
  })
}

const deviceLocationQuery = useQuery({
  queryKey: ['device', 'location'],
  queryFn: getDeviceLocation,
})
```

Demo

Stack

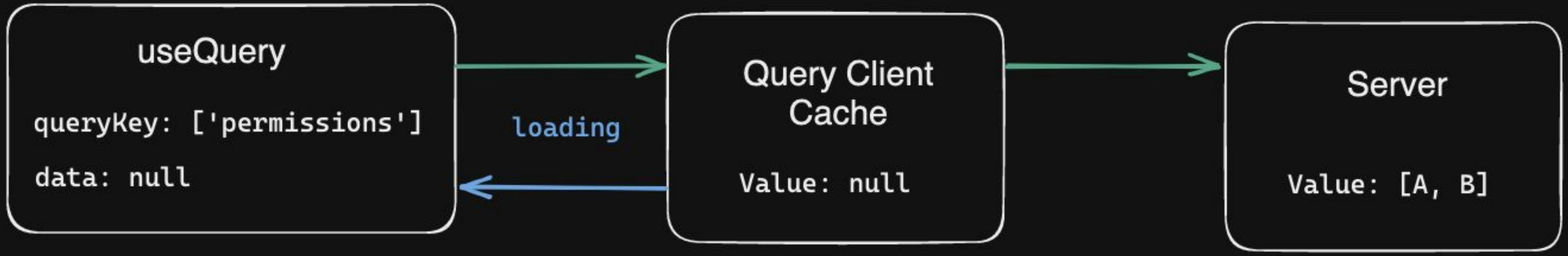
- React (of course)
- Vite
- Fetch API
- Serverless functions
- Database on Supabase

How it works



```
const usersQuery = useQuery({
  queryKey: ['some-key'],
  queryFn: async () => [{ name: 'Nick Garden' }],
}
```

First load: No data in cache



useQuery

queryKey: ['permissions']

data: null

loading

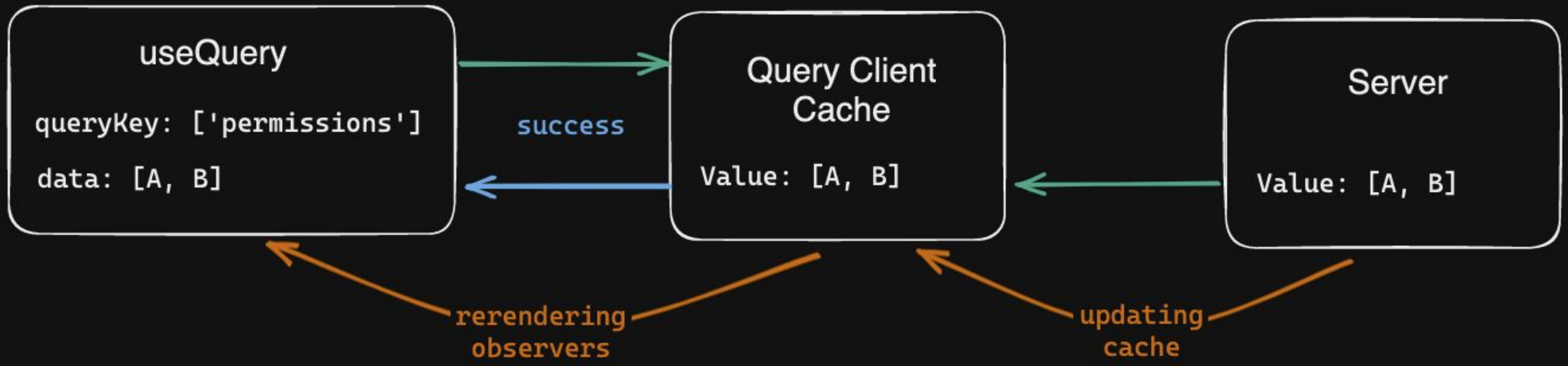
Query Client Cache

Value: null

Server

Value: [A, B]

First load: No data in cache



useQuery

queryKey: ['permissions']

data: [A, B]

success

Query Client Cache

Value: [A, B]

Server

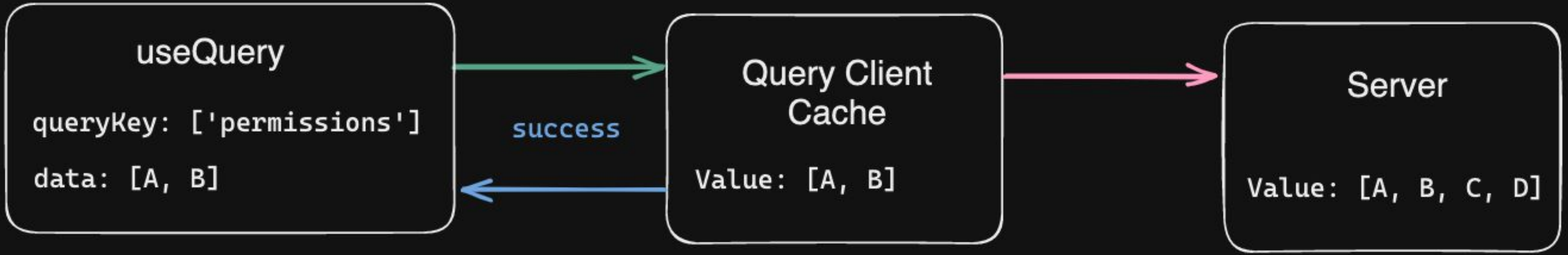
Value: [A, B]

rerendering observers

updating cache

Data is in cache, but stale ("outdated")

no loading :)



useQuery

queryKey: ['permissions']

data: [A, B]

success

Query Client Cache

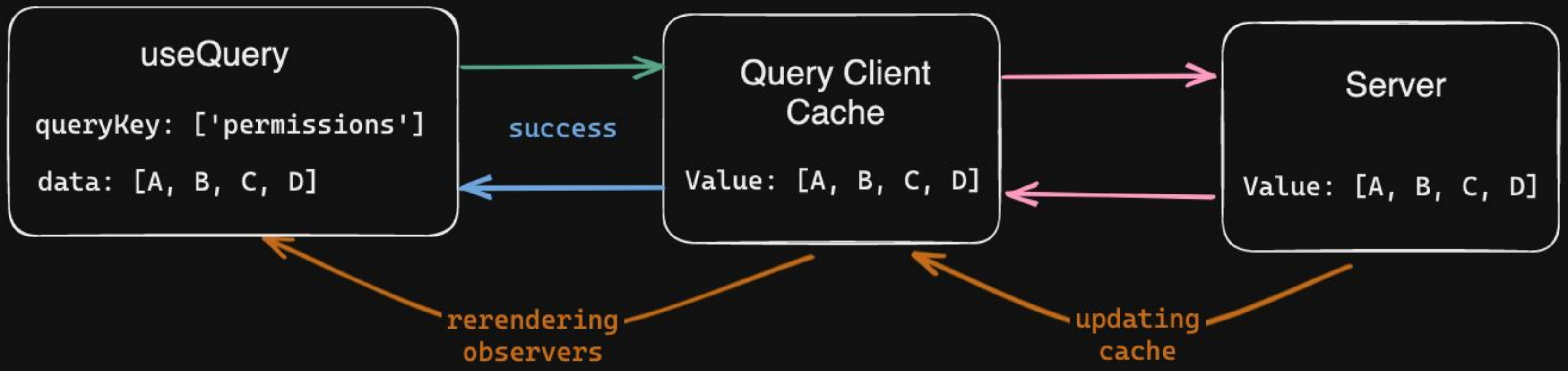
Value: [A, B]

Server

Value: [A, B, C, D]

Data is in cache, but stale ("outdated")

no loading :)



useQuery

queryKey: ['permissions']

data: [A, B, C, D]

success

Query Client Cache

Value: [A, B, C, D]

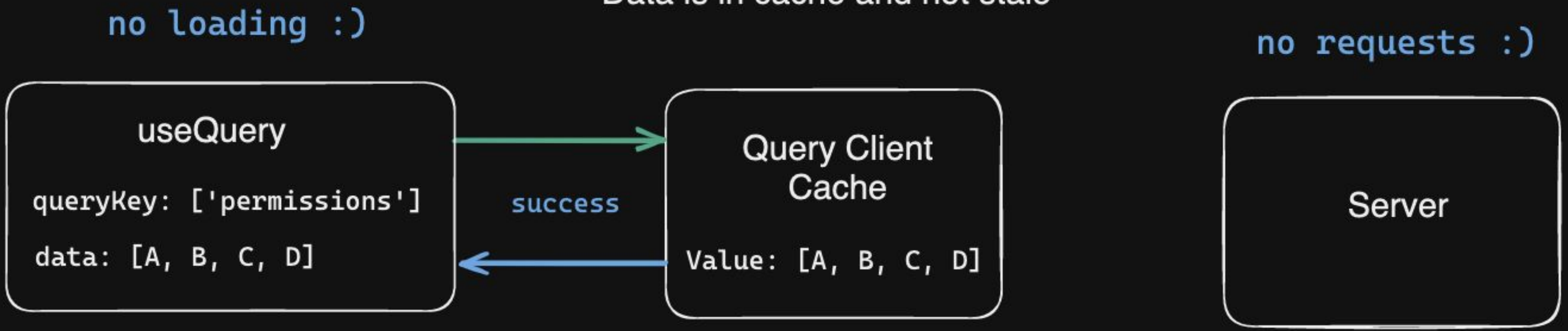
Server

Value: [A, B, C, D]

rerendering observers

updating cache

Data is in cache and not stale



no loading :)

useQuery

queryKey: ['permissions']

data: [A, B, C, D]

success

Query Client Cache

Value: [A, B, C, D]

no requests :)

Server

Query re-run when

- On mount
- On window focus
- On reconnect

Wrapping up

Thank you

Questions?

Resources 👇