DEV ACADEMY
TE KURA HANGARAU
O AOTEAROA

React Query

Week 6, Day 1

What is React Query?

React Query is a library that helps you **fetch**, *cache* and *update* data in your React applications.

I want to fetch Sharks:

```
const [sharks, setSharks] = useState<Shark[]>([])

useEffect(() => {
  async function fetchSharks() {
    const sharks = await getSharks()
    setSharks(sharks)
  }

  fetchSharks()
}, [])

return (/* ... */)
```

What happens if `fetchSharks` fails?

```
const [sharks, setSharks] = useState<Shark[]>([])
const [error, setError] = useState('')

useEffect(() => {
  async function fetchSharks() {
    setError('')
    try {
      const sharks = await getSharks()
      setSharks(sharks)
    } catch (err) {
      setError(err.message)
    }
  }

  fetchSharks()
}, [])

if (error !== '') {
  return (/* ... display error message ... */)
}

return (/* ... */)
```

What about a loading spinner?

```
const [sharks, setSharks] = useState<Shark[]>([])
const [error, setError] = useState('')
const [isLoading, setIsLoading] = useState(true)

useEffect(() => {
  async function fetchSharks() {
    setIsLoading(true)
    try {
      const sharks = await getSharks()
      setSharks(sharks)
    } catch (err) {
      setError(err.message)
    } finally {
      setIsLoading(false)
    }
  }

  fetchSharks()
}, [])

// ...
```

Phew... that was a lot of work...

What if I told you there was a better way?

Enter, React Query.

```
const { data: sharks, error, isLoading } =
useQuery(
  { queryKey: ['sharks'], queryFn: getSharks }
)

if (error) {
  return (/* ... display error message ... */)
}

if (isLoading) {
  return (/* ... display loading spinner ... */)
}

return (/* ... */)
```

Anatomy of a Query function

```
const { data, error, isLoading } = useQuery({
  queryKey: ['key'],
  queryFn: () => getSharks()
})
```

Query Key: A unique identifier for the query

Query Function: A function that returns a promise (usually an API function `` `getSharks` ``, `` `getSharkById` ``, etc.)

Let's Demo

Quick guide

Installation:

```
npm i @tanstack/react-query
```

Create a new QueryClient instance in `index.tsx` and wrap your app inside it:

```
const queryClient = new QueryClient()

<QueryClientProvider client={queryClient}>
    <App /> {/* or <RouterProvider router={router} /> with react routin
</QueryClientProvider>
```

Make a query:

```
  const { isLoading, isError, data: sharks, error } = useQuery({
    queryKey: ['sharks'],
    queryFn: fetchSharks,
  })
```

Stuck? Check the Docs!