



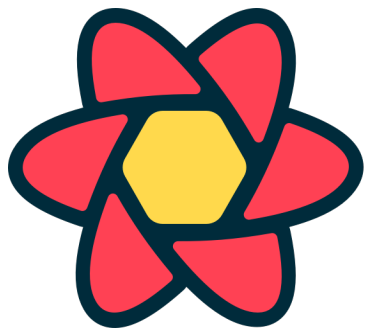
# Web 3

## React Query

**HO**  
**GENT**

# Gebruikte technologieën

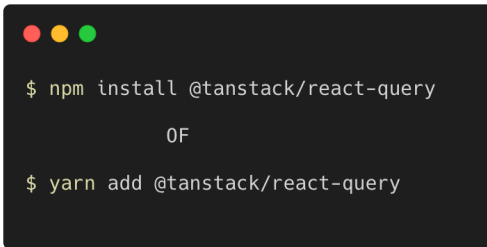
- fetch
- Axios
- TanStack Query



# React Query

**HO  
GENT**

# React Query

A dark-themed terminal window with three colored window control buttons (red, yellow, green) in the top-left corner. It displays two commands: '\$ npm install @tanstack/react-query' followed by '0F' on the next line, and '\$ yarn add @tanstack/react-query' on the third line.

```
$ npm install @tanstack/react-query
0F
$ yarn add @tanstack/react-query
```

- **Open-source JavaScript bibliotheek** die het **makkelijker** maakt om **gegevens op** te **halen** en te **beheren** vanuit een **API** in React (Native) applicaties
- **Eenvoudige** en **krachtige API** voor het **uitvoeren** van **HTTP-verzoeken** en het **verwerken** van de **resultaten**
- **Data beheren** in een **centrale cache**, aantal verzoeken verminderd en prestaties verbeterd
- Reeks handige functies: paginering, sortering, filteren, ...

<https://tanstack.com/query/latest/docs/react/overview>

# React Query

## QueryClient (Provider)

- **App** of **component** **wrappen** met de **QueryClientProvider**
- **QueryClient** **aanmaken** en **meegeven** aan de **provider**

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

const queryClient = new QueryClient();

const App = () => {
  return (
    <QueryClientProvider client={queryClient}>
      ...
    </QueryClientProvider>
  )
}
```

# React Query

## useQuery

- **Hook** belangrijk om data op te halen maar ook te versturen
- **Drie parameters: queryKey, queryFn, options**
  - **QueryKey: unieke id** voor de **request** die je aan het doen bent.  
**OPGELET** altijd in een **array**
  - **QueryFn: de functie** die de **query** moet **uitvoeren** (fetch API / Axios).
  - **Options: gedrag aanpassen** van de **query** bvb. cacheTime, enabled, ...

```
import { useQuery, useQueryClient } from '@tanstack/react-query';

const fetchPost = async (postId) => {
  const response = await fetch(`https://api.example.com/posts/${postId}`);
  return response.json();
}

const Post = ({ postId }) => {
  const queryClient = useQueryClient();
  const {data, isLoading, error} = useQuery(['post', postId], () => fetchPost(postId));
  ...
}
```

# React Query

## useMutation

- **Hook** belangrijk om **mutaties** uit te voeren op **serverdata**: **maken**, **bijwerken** of **verwijderen** (POST, PUT, DELETE)
- **mutate methode** om de **mutatie effectief uit** te **voeren**

```
import { useMutation, useQueryClient } from '@tanstack/react-query';

const deletePost = async (postId) => {
  const response = await fetch('https://api.example.com/posts/${postId}', { method: 'DELETE' });
  if(!response.ok) {
    throw new Error("Error deleting post");
  }
}

const DeleteButton = ({ postId }) => {

  const queryClient = useQueryClient();
  const mutation = useMutation({
    mutationFn: deletePost,
    onSuccess: () => {
      queryClient.invalidateQueries('posts');
    },
  });

  const handleDelete = () => {
    mutation.mutate(postId);
  }

  ...
}
```

# React Query

## prefetching

- **Mogelijkheid om vooraf data op te halen**, waardoor **gebruikersinteracties vlotter** verlopen
- `queryClient.prefetchQuery` **methode**

```
import { useQuery, useQueryClient } from '@tanstack/react-query';

const fetchPost = async (postId) => {
  const response = await fetch('https://api.example.com/posts/${postId}');
  return response.json();
};

function PostList({ posts }) {
  const queryClient = useQueryClient();

  const handleMouseEnter = async (postId) => {
    await queryClient.prefetchQuery(['post', postId], () => fetchPost(postId));
  };

  return (
    <ul>
      {posts.map((post) => (
        <li key={post.id} onMouseEnter={() => handleMouseEnter(post.id)}>
          {post.title}
        </li>
      ))}
    </ul>
  );
}
```