



00 2/3

1st

Alfredo Perez



bit.ly/tiny-query





users-page.component.ts

```
export class UsersPageComponent {

  #usersApiService = inject(UsersApiService);
  currentPage = signal(1);

  usersRequestOptions = computed(() => {
    return {
      pagination: {
        limit: 20,
        page: this.currentPage(),
      },
      orderBy: 'createdAt',
    } as RequestOptions;
  });

  onPageChange(event: PaginatorState) {
    this.currentPage.update(
      () => (event.page ? event.page + 1 : 1));
  }
}
```




users-page.component.ts

```
export class UsersPageComponent {

  #usersApiService = inject(UsersApiService);
  currentPage = signal(1);

  usersRequestOptions = computed(() => {
    return {
      pagination: {
        limit: 20,
        page: this.currentPage(),
      },
      orderBy: 'createdAt',
    } as RequestOptions;
  });

  usersQuery = injectQuery(() => ({
    queryKey: ['users', this.usersRequestOptions()],
  }));

  onPageChange(event: PaginatorState) {
    this.currentPage.update(
      () => (event.page ? event.page + 1 : 1));
  }
}
```



users-page.component.ts

```
export class UsersPageComponent {

  #usersApiService = inject(UsersApiService);
  currentPage = signal(1);

  usersRequestOptions = computed(() => {
    return {
      pagination: {
        limit: 20,
        page: this.currentPage(),
      },
      orderBy: 'createdAt',
    } as RequestOptions;
  });

  usersQuery = injectQuery(() => ({
    queryKey: ['users', this.usersRequestOptions()],
    queryFn: () =>
      this.#usersApiService.fetchPage(
        this.usersRequestOptions()),
  }));

  onPageChange(event: PaginatorState) {
    this.currentPage.update(
      () => (event.page ? event.page + 1 : 1));
  }
}
```




users-page.component.html

```
@if (usersQuery.isPending()) {  
  <p>Loading...</p>  
} @else if (usersQuery.isError()) {  
  <span> {{ usersQuery.error()?.message }} </span>  
} @else {  
  <ag-grid-angular  
    class="ag-theme-alpine border-round"  
    [rowData]="usersQuery.data()?.items"  
    [columnDefs]="columnDefs"  
  />  
}
```



users-page.component.html

```
@if (usersQuery.isPending()) {  
  <p>Loading...</p>  
} @else if (usersQuery.isError()) {  
  <span> {{ usersQuery.error()?.message }} </span>  
} @else {  
  <ag-grid-angular  
    class="ag-theme-alpine border-round"  
    [rowData]="usersQuery.data()?.items"  
    [columnDefs]="columnDefs"  
  />  
}
```

Cache



Cache

Cache

"user", { page: 1 }

[Users]

Cache

"user", { page: 1 }

[Users]

Cache

"user", { page: 1 }

[Users]

"user", { page: 2 }

[Users]

Cache

"user", { page: 1 }

[Users]

"user", { page: 2 }

[Users]

Cache

"user", { page: 1}

[Users]

"user", { page: 2}

[Users]

"user", { page: 1}

Cache

"user", { page: 1}

[Users]

"user", { page: 2}

[Users]

[Users]

"user", { page: 1}


Automatic Re-fetch

Focus Event on the window

Gets Online


New Observers

Updates on Query Key


```
onPageChange(event: PaginatorState) {  
  this.currentPage.update(  
    ()  (event.page ? event.page + 1 : 1));  
  }  
}
```

```
usersRequestOptions = computed(() => {  
  return {  
    pagination: {  
      limit: 20,  
      page: this.currentPage(),  
    },  
    orderBy: 'createdAt',  
  } as RequestOptions;  
});
```


```
usersQuery = injectQuery(() => ({  
  queryKey: ['users', this.usersRequestOptions()],  
  queryFn: () =>  
    this.#usersApiService.fetchPage(  
      this.usersRequestOptions()),  
}));
```

```
onPageChange(event: PaginatorState) {  
  this.currentPage.update(  
    ()  (event.page ? event.page + 1 : 1));  
  }  
}
```

```
usersRequestOptions = computed(() => {  
  return {  
    pagination: {  
      limit: 20,  
      page: this.currentPage(),  
    },  
    orderBy: 'createdAt',  
  } as RequestOptions;  
});
```

```
usersQuery = injectQuery(() => ({  
  queryKey: ['users', this.usersRequestOptions()],  
  queryFn: () =>  
    this.#usersApiService.fetchPage(  
      this.usersRequestOptions()),  
}));
```

```
onPageChange(event: PaginatorState) {  
  this.currentPage.update(  
    () => (event.page ? event.page + 1 : 1));  
}  
}
```

```
usersRequestOptions  computed(() => {  
  return {  
    pagination: {  
      limit: 20,  
      page: this.currentPage(),  
    },  
    orderBy: 'createdAt',  
  } as RequestOptions;  
});
```

```
usersQuery = injectQuery(() => ({  
  queryKey: ['users', this.usersRequestOptions()],  
  queryFn: () =>  
    this.#usersApiService.fetchPage(  
      this.usersRequestOptions()),  
}));
```

Sinergy

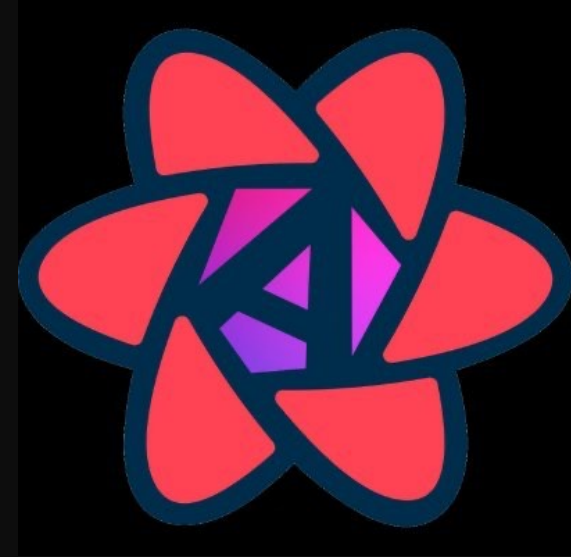




Component State



**Component
State**



**Server (Async)
State**

Server (Async) State

**Placeholder
Data**

Prefetching



bit.ly/tiny-query



