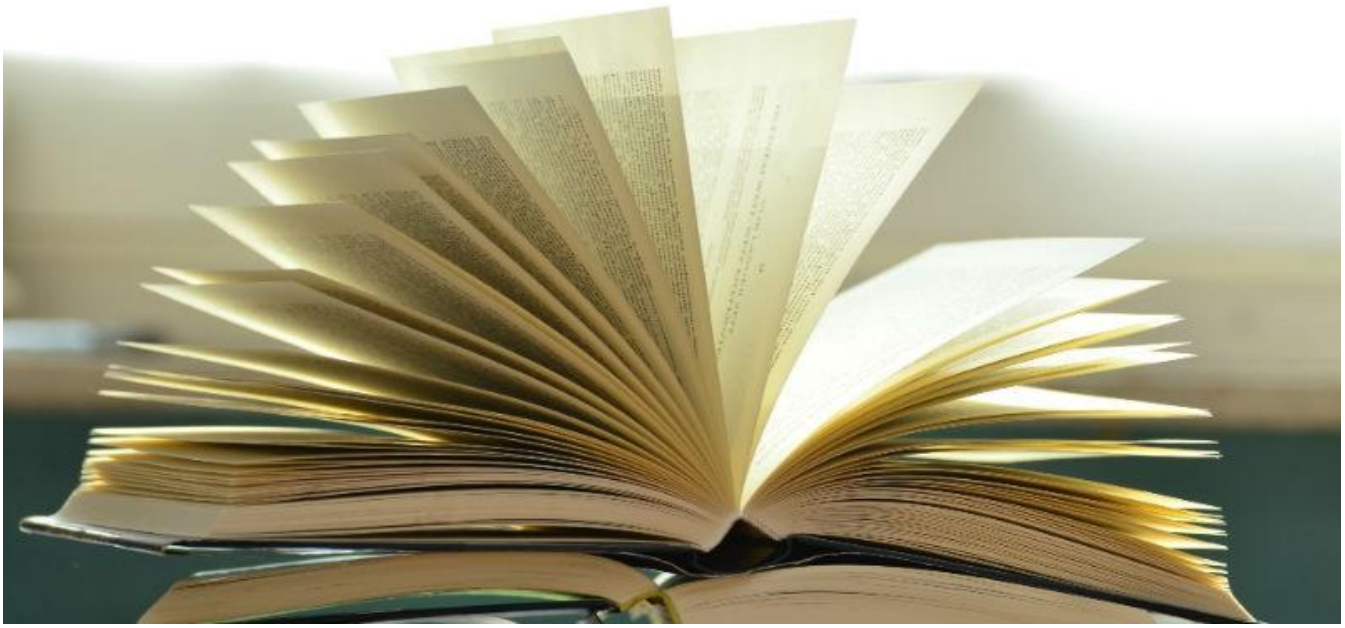
GraphQL Pagination By Example: Part 2

John Tucker
Aug 21, 2018 · 4 min read

Offset paging; server and client implementation.



This article is part of a series starting with *GraphQL Pagination By Example: Part 1*.

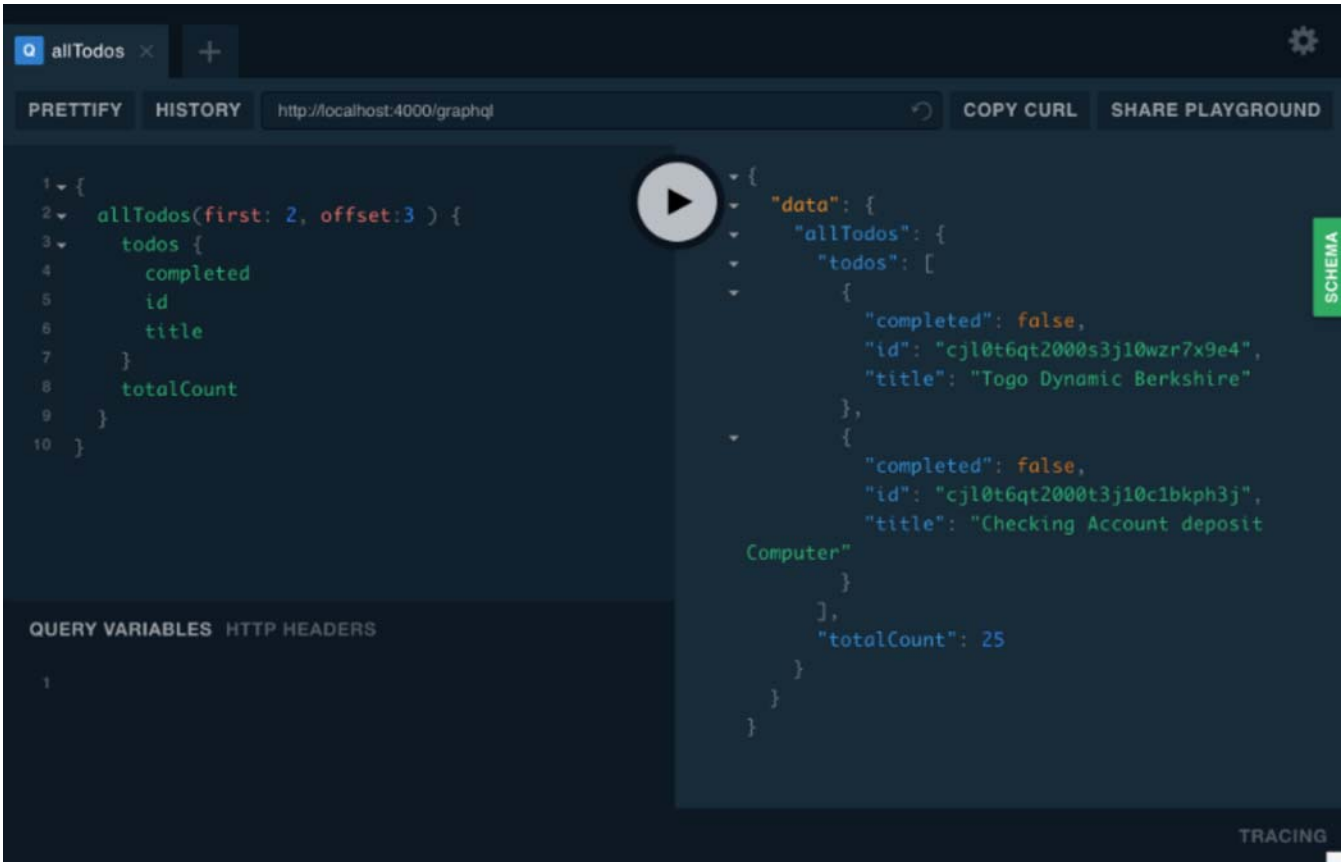
The final example for this series is available for download.

Offset Paging

The official *GraphQL* documentation has a section, *Pagination*, discussing the options for paginating data; the simplest being offset paging (which is quickly dismissed).

The problem with offset paging is only really suitable when the data (being paginated) is not expected to change while the user is consuming it. The article, *Pagination: You're (Probably) Doing It Wrong*, does a good job of illustrating this problem.

That being said, let us first implement offset paging in our example.



Observations:

- The *allTodos* query accepts two optional parameters; *first* and *offset*
- The *first* parameter limits the number of todos to return, e.g., limit to 2. If unspecified, the query will return all of them.
- The *offset* parameter indicates which todo (zero-based index) to return first., e.g., begin with todo with index of 3 (actually the fourth todo). If unspecified, the query will begin with the first todo (index of 0).
- The result consists of *todos* (the array of returned todos) and *totalCount* (the total number of todos)

The Code

The *GraphQL* type definitions need to be updated to support offset paging.

src / server.ts

```
...
const typeDefs = gql`
  type Todo {
    id: ID!
```

```
    title: String!
    completed: Boolean!
  }
  type TodosResult {
    todos: [Todo]
    totalCount: Int
  }
  type Query {
    allTodos(
      first: Int,
      offset: Int
    ): TodosResult
  }
`;
...
```

then we refactor the resolver to support it.

src / todos.ts

```
interface Todo {
  completed: boolean;
  id: string;
  title: string;
}
interface TodosResult {
  totalCount: number;
  todos: Todo[];
}
...
export const allTodos = (_: any, { first, offset = 0 }: { first: number, offset: number}): Todo
  const totalCount = data.length;
  const todos = first === undefined ?
    data.slice(offset) :
    data.slice(offset, offset + first);
  const result = {
    todos,
    totalCount,
  };
  return result;
};
```

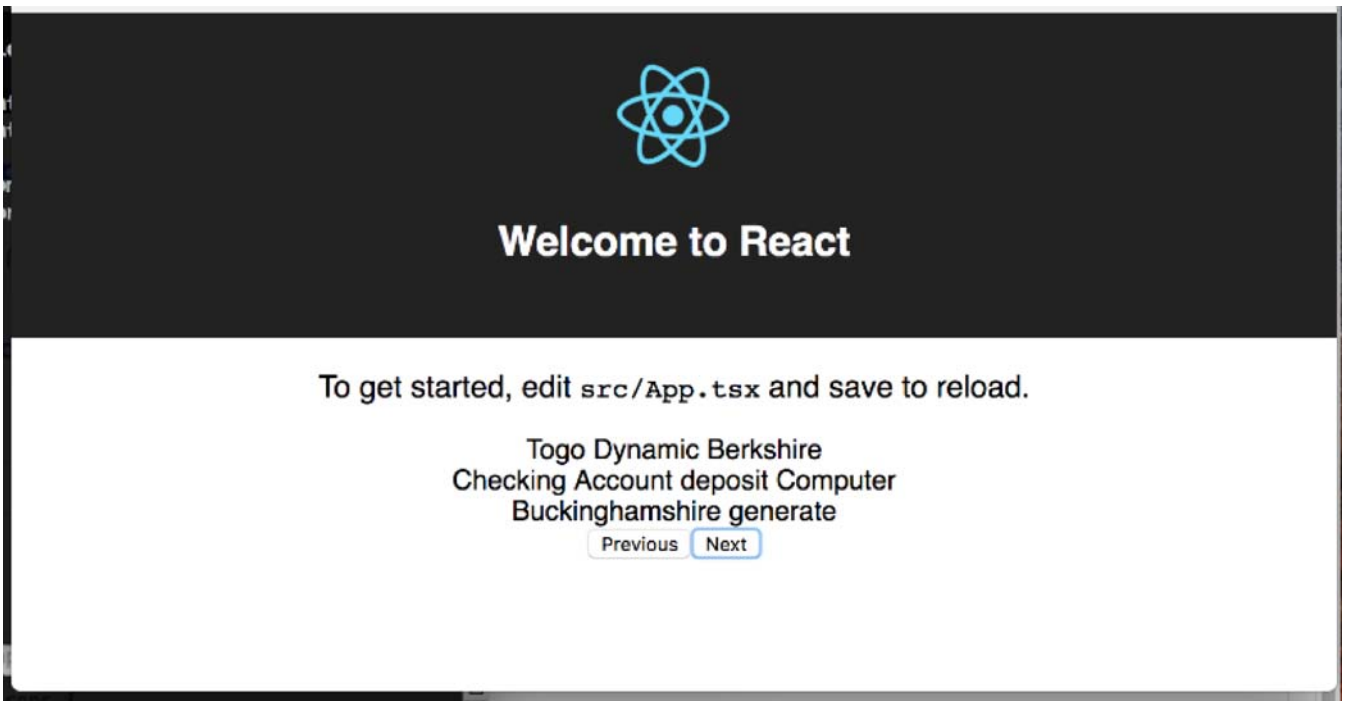
Offset Paging in Apollo Client

To complete the circle, we will write a *React + Apollo Client* application to consume this offset paging API.

note: You can skip this section if you are neither interested in *Apollo Client* or familiar with it. This implementation is based on the final article, *GraphQL and Apollo Client by Example: Part 8*, in a series I wrote on *Apollo Client*.

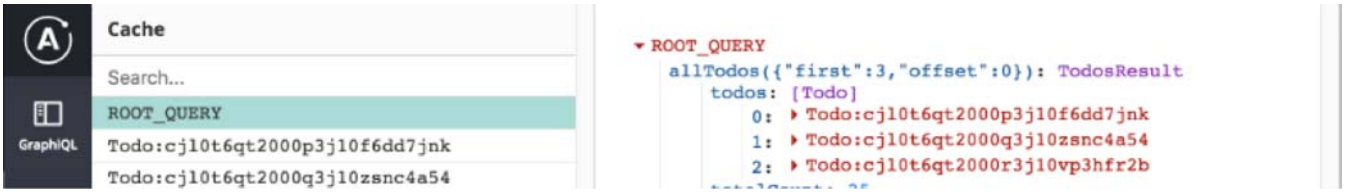
The complete version of the offset client example is available for download.

We start with the download from the *Apollo Client* article and update it to a paginated version; downloads and displays three todos at a time.



Observations:

- The *Previous* and *Next* buttons are disabled at the beginning and end of the pagination respectively
- Looking at *Apollo Client Developer Tools* tools; one will observe that indeed todos are downloaded three at a time; also once downloaded they are served from the *Apollo Client* cache (when using the *Previous* button); below we inspect the cache while viewing the fourth page of todos.



Code highlights:

- Did not use the *fetchMore* function as described in the official *Apollo Client* documentation on pagination; it did not support this use-case of paginating (only showing three todos at a time)
- We first implement a *Pagination* component that maintains the state of the *offset* and *totalCount* (the *FIRST* value is fixed to 3). While this example uses *React* state, one could use a global state management solution, e.g., *Redux* or *Apollo Client* local state
- The *Pagination* component also implements the *Previous* and *Next* buttons and renders the *Todos* component
- The *Todos* component executes the parameterized (*first* and *offset*) *allTodos* query and renders the *TodosView* and *TodoCount* components
- The *TodosView* component renders the todos
- The less obvious component, *TodoCount*, is used to initialize the *totalCount* state of the *Pagination* component using the *setTotalCount* function

Next Steps

In the next article, *GraphQL Pagination By Example: Part 3*, we explore cursor paging.

codeburst.io

> [Web Development](#).

React GraphQL Web Development Nodejs Typescript