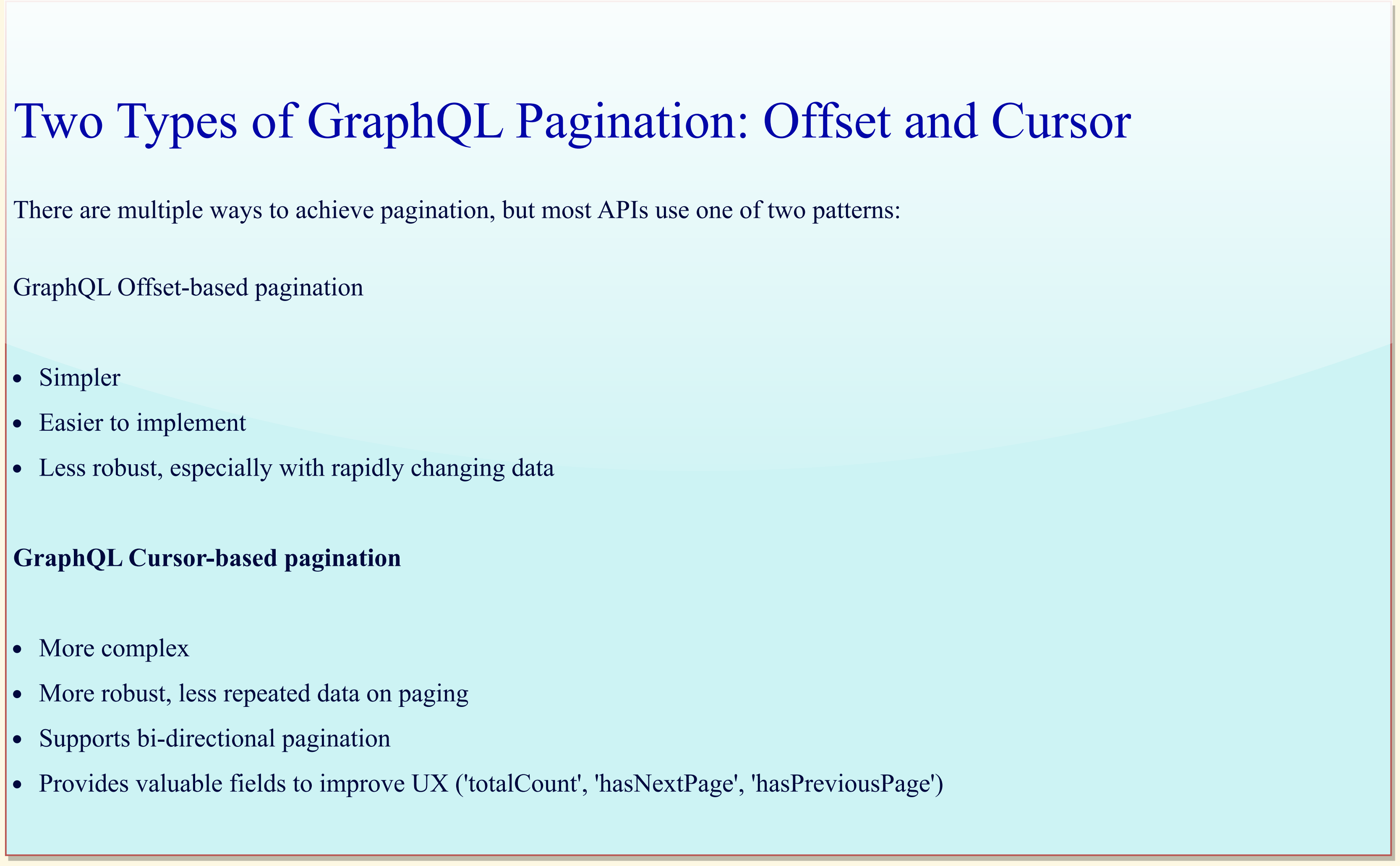


Two Types of GraphQL Pagination: Offset and Cursor

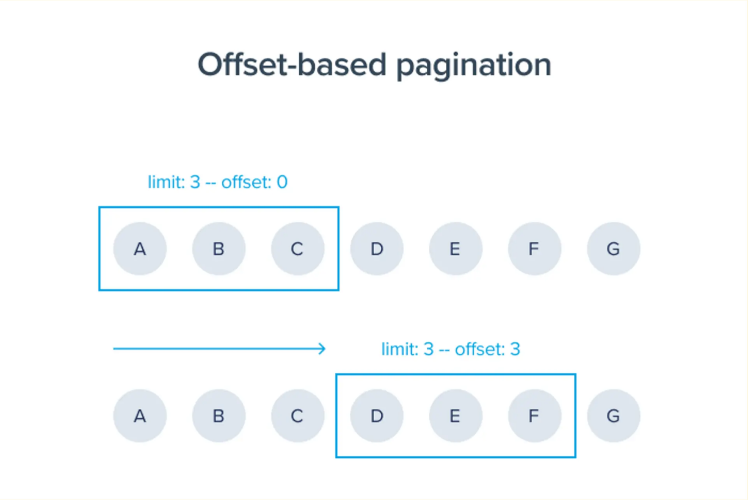
There are multiple ways to achieve pagination, but most APIs use one of two patterns:

GraphQL Offset-based pagination

- Simpler
- Easier to implement
- Less robust, especially with rapidly changing data

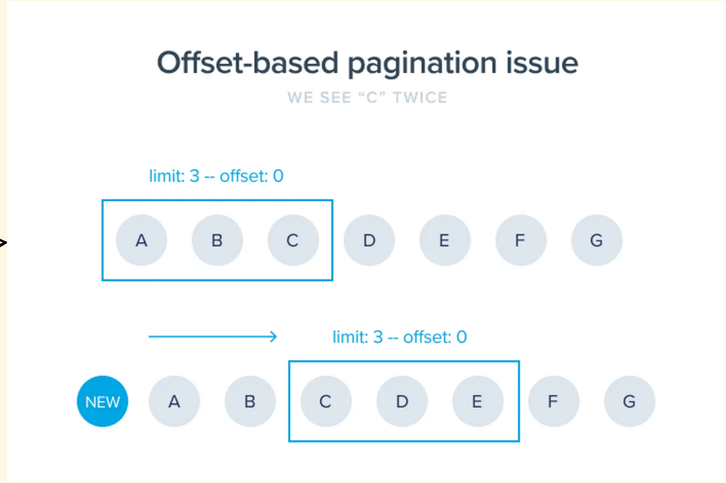
GraphQL Cursor-based pagination

- More complex
- More robust, less repeated data on paging
- Supports bi-directional pagination
- Provides valuable fields to improve UX ('totalCount', 'hasNextPage', 'hasPreviousPage')



Pro
Very simple to implement

Cons
- repeated data in certain situations specially if you add a new row while paging - No way to retrieve the last page (this is a commonly desired feature for table views of data) - No way to know if there are more pages (for disabling the next page button) - No way to know the total number of items



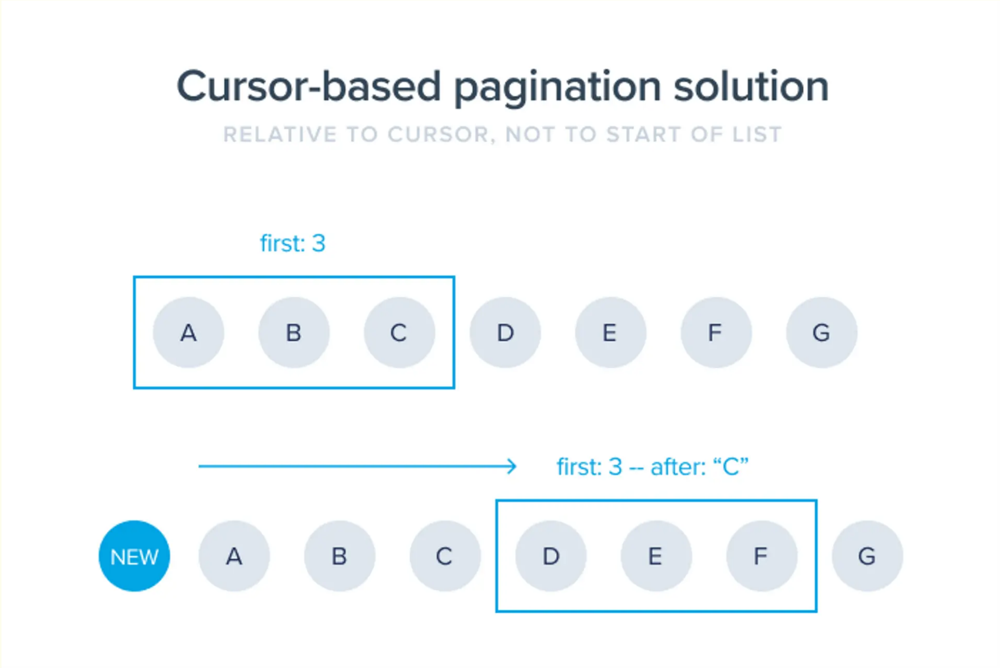
Offset-based is preferable where, for example:

You want to keep your app development simple
Data doesn't change very often, or seeing duplicates is not a deal breaker
You need to retrieve (i.e., skip to) a specific page at the start

courses → find() -> limit it to passed limit

courses → limit,offset

Relay - styled



Cursor-based pagination solution

RELATIVE TO CURSOR, NOT TO START OF LIST

Pro

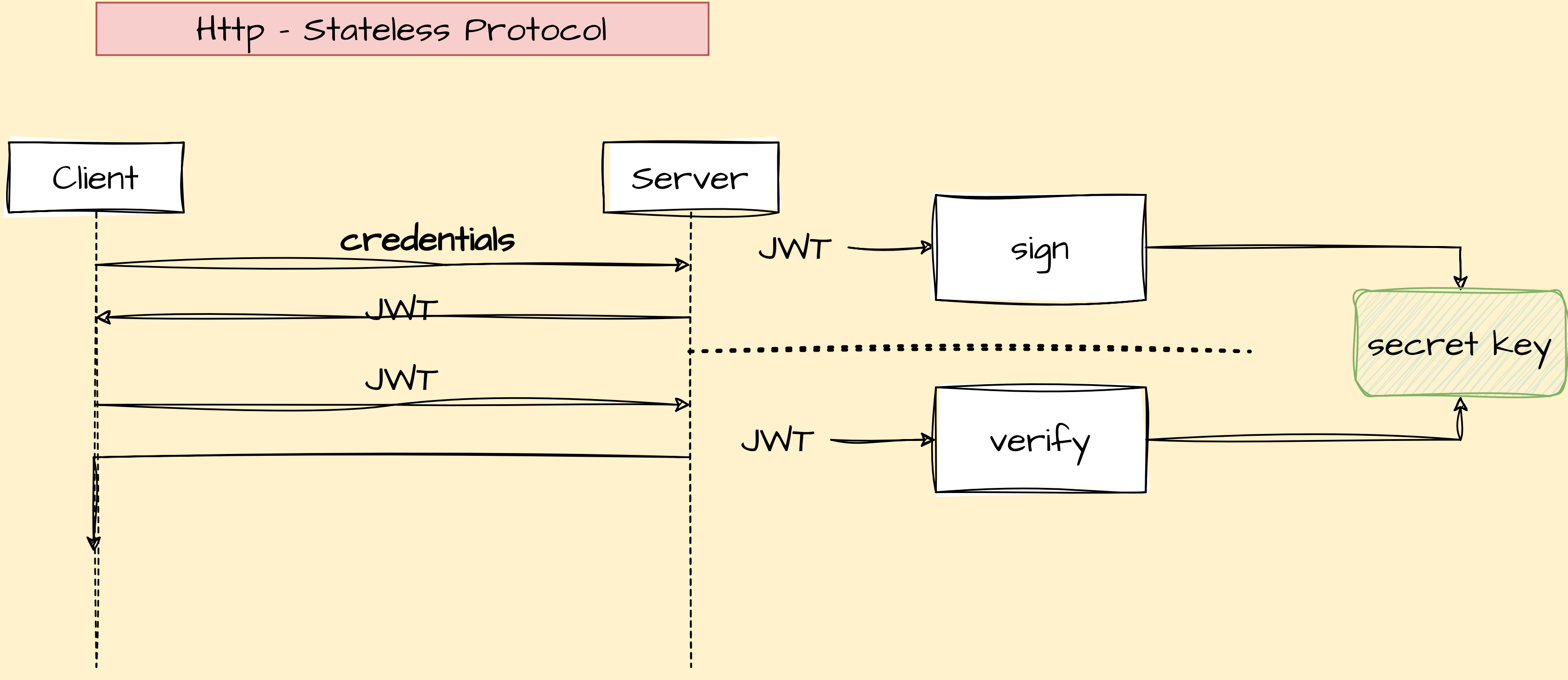
- provides a lot more data which can be helpful for UX
- Supports reverse pagination
- Pagination is relative to specific rows to avoid issues around dynamic data

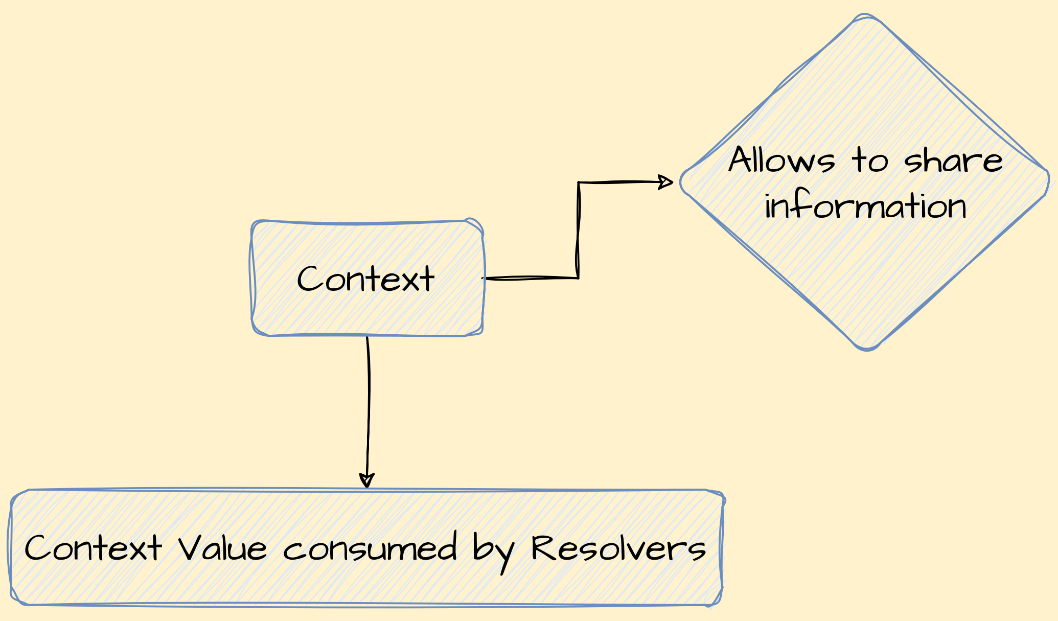
Cons

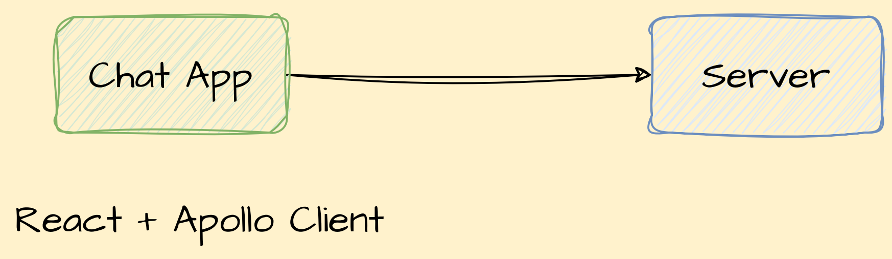
- More verbose than Offset-based pagination
- Results in larger and more nested queries
- No way to grab an arbitrary page to start (for example, you can't start on page 3, you need to get each page to retrieve the cursors)

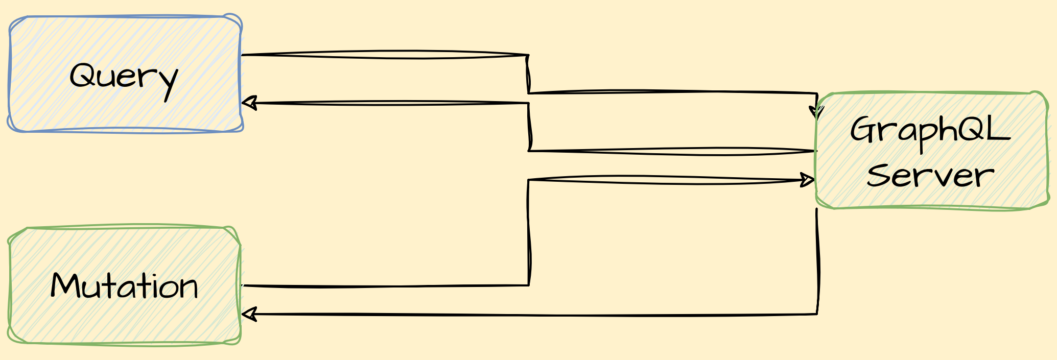
cursor-based is preferred where:

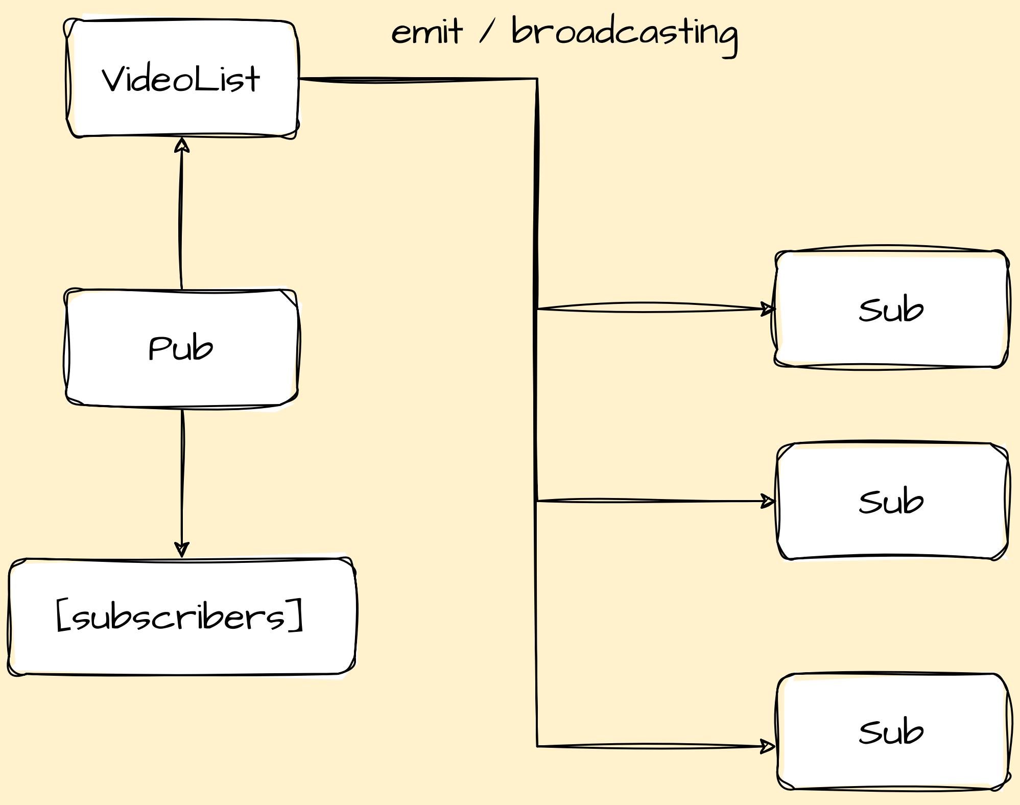
Your lists are dynamic, changing often (e.g., a newsfeed)
Users want to navigate back and forth, or jump to the very first or last page
You want a professional pagination experience (i.e., no duplicate entries, additional data like total entries/pages, etc.)

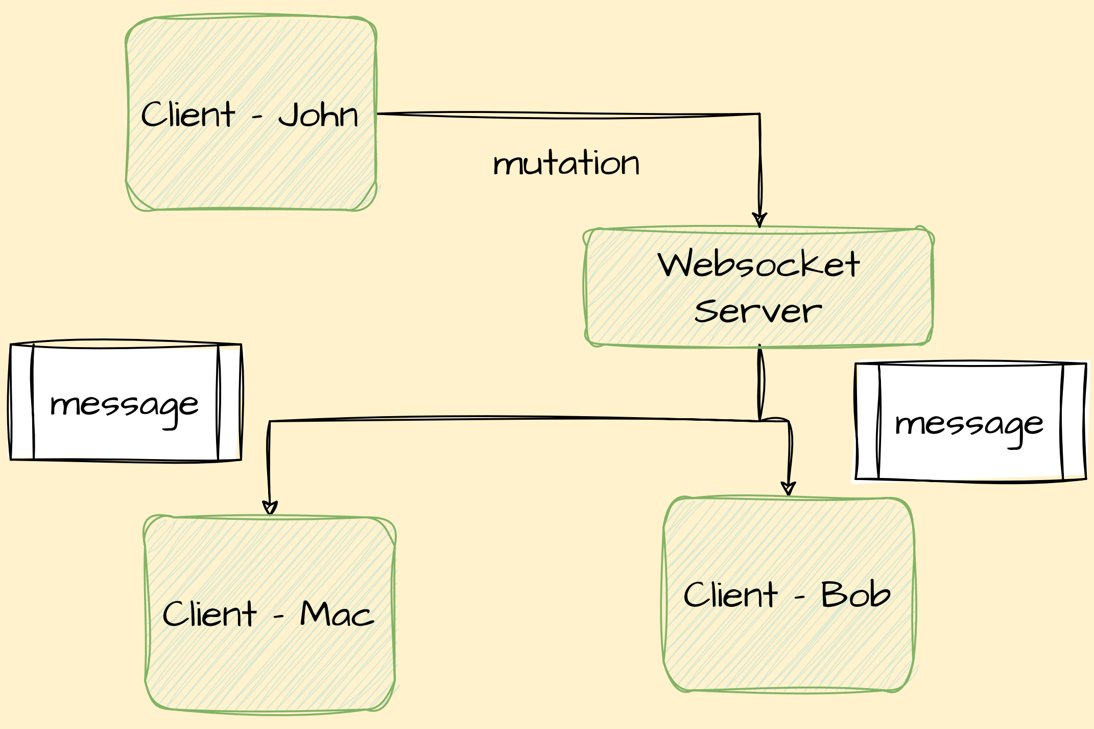

Http - Stateless Protocol
Client
Server
credentials
JWT
JWT
JWT
sign
secret key
JWT
verify

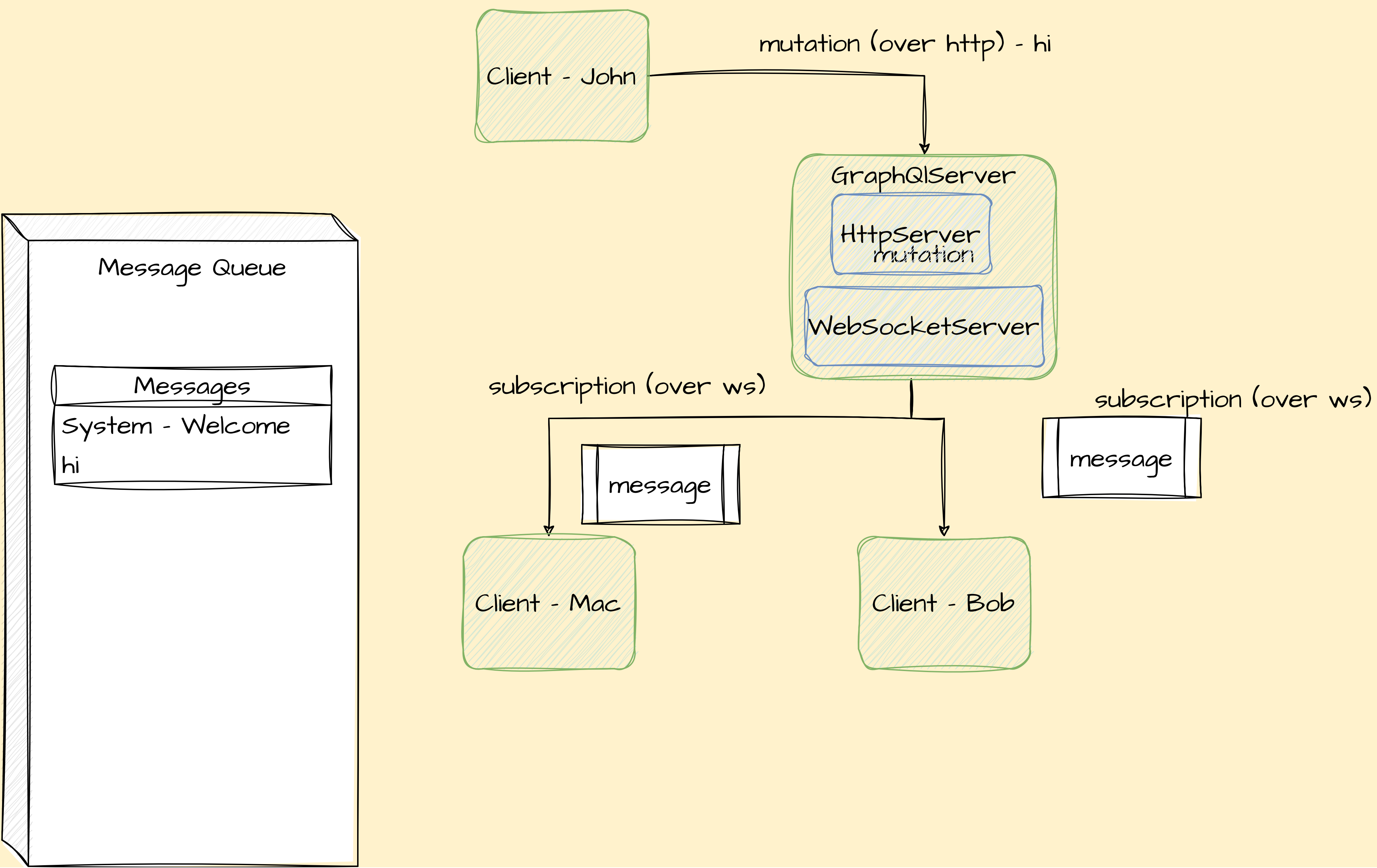

Allows to share information
Context
Context Value consumed by Resolvers


Chat App
Server
React + Apollo Client


Query
GraphQL Server
Mutation


emit / broadcasting
VideoList
Sub
Pub
Sub
[subscribers]
Sub


Client - John
mutation
Websocket Server
message
message
Client - Mac
Client - Bob


Client - John
mutation (over http) - hi
GraphQlServer
HttpServer
mutation
WebSocketServer
Message Queue
Messages
System - Welcome
hi
subscription (over ws)
subscription (over ws)
message
message
Client - Mac
Client - Bob