Apollo: Performance Management Playbook

USING APOLLO CLIENT FOR REDUX LIKE STATE MANAGEMENT

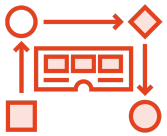
Peter Kellner
DEVELOPER, CONSULTANT AND AUTHOR

@pkellner linkedin.com/in/peterkellner99 ReactAtScale.com

Apollo Client Features for Managing State

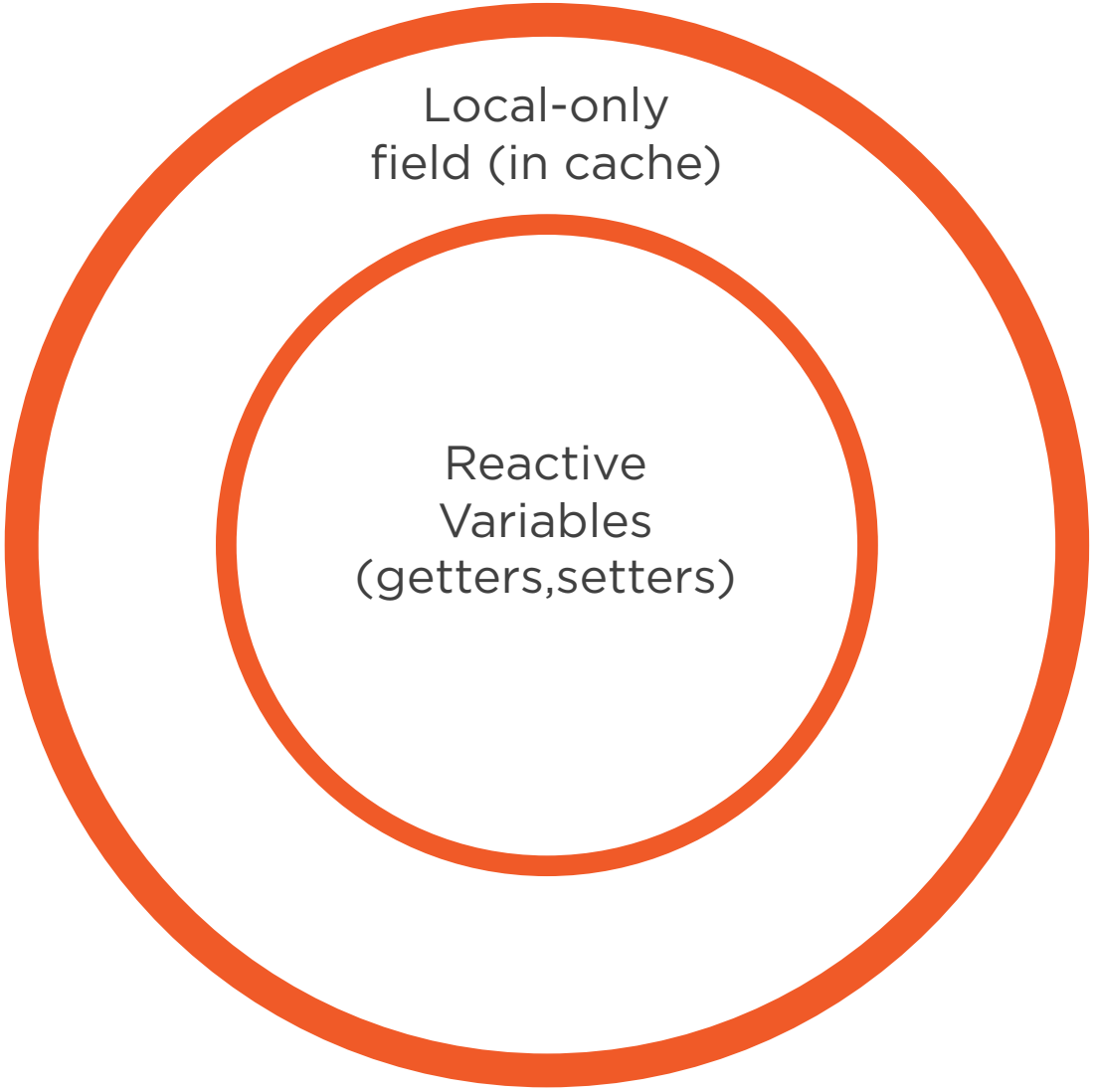
Creating and updating virtual or Local-only fields in a GraphQL schema

Creating and updating global or Reactive variables stored the Apollo Client instance



Fetching Local-only fields that include Reactive variables resulting in full state management in your app

State Management with Apollo Client



GraphQL queries

speakers {id,first...

speakers {id,favorite...

speakers {id,localonly...

Processed through useQuery

State data returned from **useQuery** reflected in UI

The Apollo 3 client supports virtual or Local-only schema fields.

```
type Speaker {
   id: ID!
   first: String
   last: String
   favorite: Boolean
 }
```

Apollo Server Schema

Type definition for Speaker

Reactive variables just like other variables except running app can observe and react to changes.

React Context API vs Apollo GraphQL Provider

React Context API

```
const App = () => {
  return (
    <ConfigContext.Provider
      value={configValue}>
      <div>Components</div>
    </ConfigContext.Provider>
  );
};
```

Apollo GraphQL Provider

```
const App = () => {
  return (
    <ApolloProvider
        client={apolloClient}>
      <div>Components</div>
    </ApolloProvider>
  );
};
```

Why Use Cache in Apollo GraphQL Client

Good for server

Fewer roundtrips from client to server

Less computational load required

Less network traffic

Lower cost

Good for user

User interface more responsive

Works better on slower connections

Enables optimistic UI

Leverage local compute power

```
const apolloClient = new ApolloClient({
  uri: "http://localhost:4000",
  cache: new InMemoryCache(),
});
```

Apollo Client Configuration