React – Apollo Client

A short introduction

What is Apollo Client?

- **Apollo Client** is a comprehensive state management library for JavaScript that enables you to manage both local and remote data with GraphQL. Use it to fetch, cache, and modify application data, all while automatically updating your UI.

What is Apollo Client?

- Apollo Client helps you structure code in an economical, predictable, and declarative way that's consistent with modern development practices. The core @apollo/client library provides built-in integration with React, and the larger Apollo community maintains integrations for other popular view layers.

Apollo Client Features

- **Declarative data fetching:** Write a query and receive data without manually tracking loading states.
- **Designed for modern React:** Take advantage of the latest React features, such as hooks.
- **Incrementally adoptable:** Drop Apollo into any JavaScript app and incorporate it feature by feature.
- **Universally compatible:** Use any build setup and any GraphQL API.
- **Community driven:** Share knowledge with thousands of developers in the GraphQL community.

Apollo Client Features

- **Declarative data fetching:** Write a query and receive data without manually tracking loading states.
- **Designed for modern React:** Take advantage of the latest React features, such as hooks.
- **Incrementally adoptable:** Drop Apollo into any JavaScript app and incorporate it feature by feature.
- **Universally compatible:** Use any build setup and any GraphQL API.
- **Community driven:** Share knowledge with thousands of developers in the GraphQL community.

Apollo Client Recommended docs

- Queries and Mutations. These are the read and write operations of GraphQL.
- Caching overview. Apollo Client's normalized cache enables you to skip network requests entirely when data is already available locally.
- Managing local state. Apollo Client provides APIs for managing both remote and local data, enabling you to consolidate all of your application's state.
- Basic HTTP networking. Learn how to send custom headers and other authentication metadata in your queries.

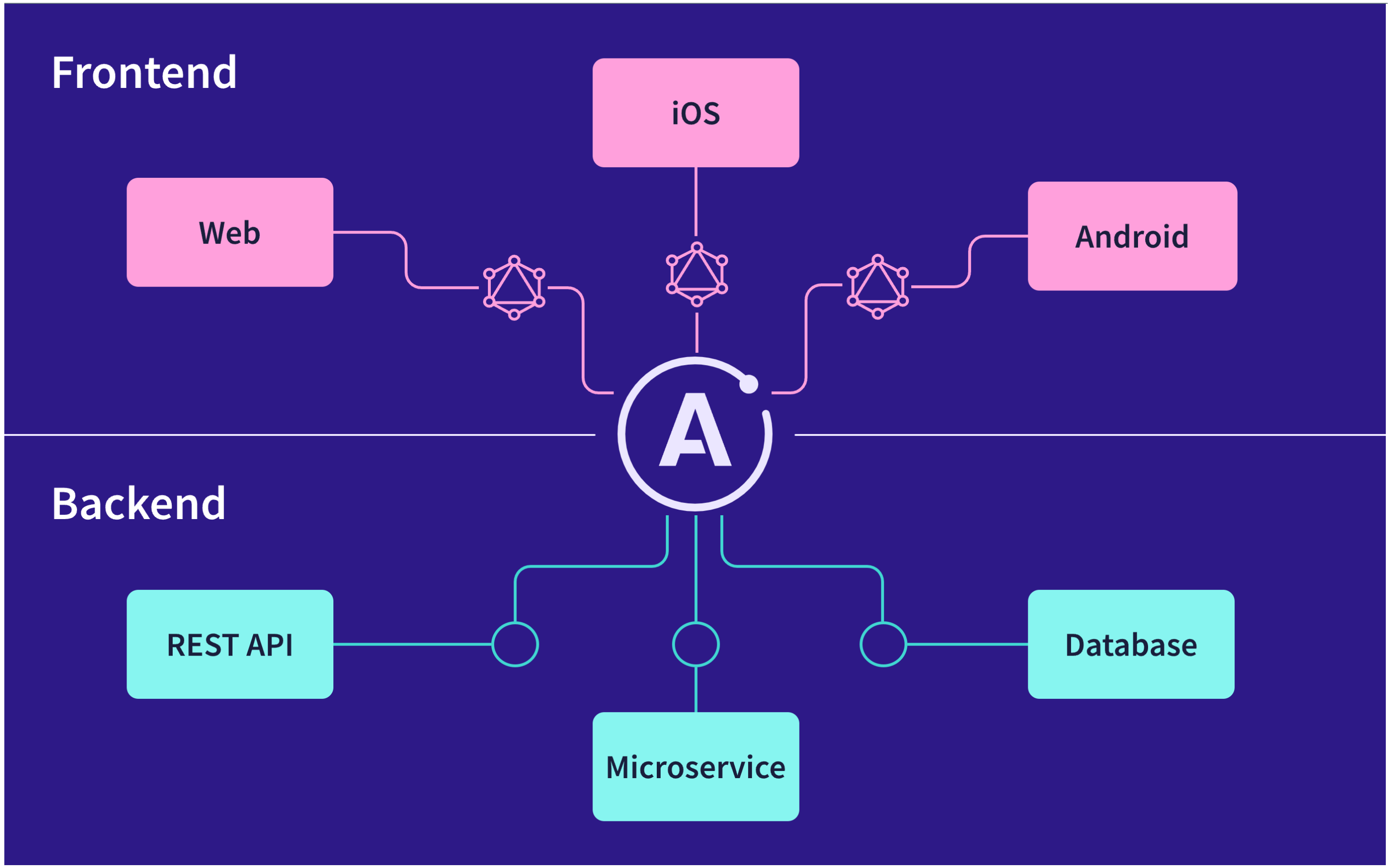
Further reading

Further reading

- https://www.apollographql.com/docs/react/
- https://www.apollographql.com/docs/intro/platform

React – Apollo Server

Apollo Server


Frontend
iOS
Web
Android
Backend
REST API
Database
Microservice

What is Apollo Server?

- **Apollo Server is an open-source, spec-compliant GraphQL server** that's compatible with any GraphQL client, including Apollo Client. It's the best way to build a production-ready, self-documenting GraphQL API that can use data from any source.

You can use Apollo Server as:

- A gateway for a federated supergraph (https://www.apollographql.com/docs/federation/)
- The GraphQL server for a subgraph in a federated supergraph
- A stand-alone GraphQL server, including in a serverless environment
- An add-on to your application's existing Node.js middleware (such as Express or Fastify)

Apollo Server provides:

- **Straightforward setup**, so your client developers can start fetching data quickly
- **Incremental adoption**, allowing you to add features as they're needed
- **Universal compatibility** with any data source, any build tool, and any GraphQL client
- **Production readiness**, enabling you to ship features faster

Getting Started:

- **https://www.apollographql.com/docs/apollo-server/getting-started**
- https://codesandbox.io/s/github/apollographql/docs-examples/tree/main/apollo-server/v3/getting-started?fontsize=14&hidenavigation=1&theme=dark&file=/index.js

GraphQL schema basics

- Your GraphQL server uses a **schema** to describe the shape of your available data. This schema defines a hierarchy of **types** with **fields** that are populated from your back-end data stores. The schema also specifies exactly which **queries** and **mutations** are available for clients to execute.

The schema definition language

- The GraphQL specification defines a human-readable **schema definition language** (or **SDL**) that you use to define your schema and store it as a string.

Further reading

Further reading

- https://www.makeuseof.com/what-are-web-components/
- https://www.codeinwp.com/blog/react-best-practices/

- https://reactjs.org/docs/hooks-reference.html#usereducer

- https://redux.js.org/tutorials/fundamentals/part-3-state-actions-reducers