Cheat Sheet for comprensive GraphQL

Comprehensive GraphQL Cheat Sheet

1. Introduction to GraphQL

- **What is GraphQL?**

- A query language for APIs.
- Provides a complete and understandable description of the data in your API.
- Gives clients the power to ask for exactly what they need and nothing more.

---

2. Core Concepts

Schema Definition Language (SDL)

- **Schema Definition**

- Defines the types and operations in your API.
- Example:

```
type Query {
  user(id: ID!): User
}

type User {
  id: ID!
  name: String!
  email: String!
}
```

Queries

- **Basic Query**

- Retrieves data from the server.
- Example:

```
query {
  user(id: "1") {
    id
    name
    email
```

```
  }
}
```

- **Nested Queries**

- Example:

```
query {
  user(id: "1") {
    id
    name
    posts {
      title
      content
    }
  }
}
```

Mutations

- **Basic Mutation**

- Modifies data on the server.
- Example:

```
mutation {
  createUser(name: "John Doe", email: "john@example.com") {
    id
    name
    email
  }
}
```

Subscriptions

- **Basic Subscription**

- Real-time updates from the server.
- Example:

```
subscription {
  newUser {
    id
    name
    email
```

```
  }
}
```

---

3. Types and Fields

Scalar Types

- **Built-in Scalar Types**

- `Int`: A signed 32-bit integer.
- `Float`: A signed double-precision floating-point value.
- `String`: A UTF-8 character sequence.
- `Boolean`: `true` or `false`.
- `ID`: A unique identifier, often serialized as a string.

Object Types

- **Definition**

- Example:

```
type User {
  id: ID!
  name: String!
  email: String!
}
```

Interface Types

- **Definition**

- Example:

```
interface NamedEntity {
  name: String!
}

type User implements NamedEntity {
  id: ID!
  name: String!
  email: String!
}
```

Union Types

- **Definition**

- Example:

```
union SearchResult = User | Post

type Query {
  search(query: String!): [SearchResult!]!
}
```

Input Types

- **Definition**

- Example:

```
input UserInput {
  name: String!
  email: String!
}

type Mutation {
  createUser(input: UserInput!): User!
}
```

---

4. Directives

- **Built-in Directives**

- `@include(if: Boolean)`: Includes a field if the argument is `true`.
- `@skip(if: Boolean)`: Skips a field if the argument is `true`.
- Example:

```
query {
  user(id: "1") {
    id
    name @include(if: $showName)
    email @skip(if: $hideEmail)
  }
}
```

---

5. Execution

Query Execution

- **Execution Flow**

- The server processes the query and returns the requested data.
- Example:

```
query {
  user(id: "1") {
    id
    name
  }
}
```

Mutation Execution

- **Execution Flow**

- The server processes the mutation and returns the modified data.
- Example:

```
mutation {
  createUser(name: "John Doe", email: "john@example.com") {
    id
    name
    email
  }
}
```

Subscription Execution

- **Execution Flow**

- The server establishes a real-time connection and pushes updates to the client.
- Example:

```
subscription {
  newUser {
    id
    name
    email
  }
}
```

---

6. Error Handling

- **Error Types**

- **GraphQL Errors**: Errors returned in the `errors` array of the response.

- **Network Errors**: Errors related to the network or server communication.

- **Example Error Response**

```
{
  "errors": [
    {
      "message": "User not found",
      "locations": [{"line": 2, "column": 3}],
      "path": ["user"]
    }
  ],
  "data": null
}
```

---

7. Best Practices

- **Designing Schemas**

- Use clear and descriptive names.
- Avoid over-nesting.

- **Writing Queries**

- Only request the fields you need.
- Use fragments for reusable pieces of queries.

- **Handling Errors**

- Provide meaningful error messages.
- Use custom error types when necessary.

---

8. Tools and Libraries

- **GraphQL Clients**

- **Apollo Client**: A comprehensive state management library for JavaScript.

- **Relay**: A framework for building data-driven React applications.

- **GraphQL Servers**

- **Apollo Server**: A flexible, community-driven GraphQL server.

- **GraphQL Yoga**: A fully-featured GraphQL server with focus on easy setup.

- **GraphQL IDEs**

- **GraphiQL**: An in-browser IDE for exploring GraphQL.

- **Altair**: A feature-rich GraphQL client for all platforms.

---

9. Advanced Topics

Fragments

- **Definition**

- Reusable units of query logic.
- Example:

```
fragment UserDetails on User {
  id
  name
  email
}

query {
  user(id: "1") {
    ...UserDetails
  }
}
```

Variables

- **Definition**

- Dynamic values passed to queries and mutations.
- Example:

```
query GetUser($userId: ID!) {
  user(id: $userId) {
    id
    name
```

```
  }
}
```

Aliases

- **Definition**

- Rename fields in the response.
- Example:

```
query {
  user1: user(id: "1") {
    id
    name
  }
  user2: user(id: "2") {
    id
    name
  }
}
```

Pagination

- **Definition**

- Efficiently handle large datasets.
- Example:

```
query {
  users(first: 10, after: "cursor") {
    edges {
      node {
        id
        name
      }
    }
    pageInfo {
      endCursor
      hasNextPage
    }
  }
}
```

Caching

- **Definition**

- Optimize performance by storing and reusing data.
- Example:

```
query {
  user(id: "1") {
    id
    name
  }
}
```

Authentication and Authorization

- **Definition**

- Secure your GraphQL API.
- Example:

```
type Mutation {
  createPost(title: String!, content: String!): Post!
    @auth(requires: USER)
}
```

This cheat sheet provides a comprehensive overview of GraphQL, covering its core concepts, types, execution, error handling, best practices, tools, and advanced topics. Use this as a quick reference to master GraphQL.

By Ahmed Baheeg Khorshid

ver 1.0