Cheat Sheet for comprehensive GraphQL

GraphQL Basics

Schema Definition Language (SDL)

- **Scalar Types**: \`Int\`, \`Float\`, \`String\`, \`Boolean\`, \`ID\`
- **Object Types**:

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

- **Interfaces**:

```
interface NamedEntity {
  name: String!
}
```

- **Enums**:

```
enum UserRole {
  ADMIN
  USER
}
```

- **Input Types**:

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

Queries and Mutations

- **Queries**: Fetch data

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

- **Mutations**: Modify data

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

- **Subscriptions**: Real-time updates

```
subscription {
  newMessage {
    content
    sender
  }
}
```

Advanced Features

Fragments

- **Inline Fragments**:

```
query {
  user(id: "1") {
    ... on Admin {
      permissions
    }
    ... on User {
      name
    }
  }
}
```

- **Named Fragments**:

```
fragment UserDetails on User {
  name
  email
}
query {
  user(id: "1") {
    ...UserDetails
  }
}
```

Directives

- **@include**: Include field if condition is true

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

- **@skip**: Skip field if condition is true

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

Variables

- **Defining Variables**:

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

- **Using Variables**:

```
{
  "userId": "1"
}
```

Best Practices

Naming Conventions

- **Types**: PascalCase (e.g., `UserProfile`)
- **Fields**: camelCase (e.g., `userName`)
- **Enums**: UPPER_CASE (e.g., `USER_ROLE`)

Error Handling

- **GraphQL Errors**: Returned in the `errors` array in the response
- **Custom Errors**: Use custom error types in resolvers

Performance Optimization

- **Batching**: Use `DataLoader` to batch requests
- **Caching**: Implement caching strategies at the resolver level

Tools and Libraries

GraphQL Clients

- **Apollo Client**: Comprehensive state management
- **Relay**: Optimized for performance and pagination
- **urql**: Lightweight and extensible

GraphQL Servers

- **Apollo Server**: Full-featured server
- **Express GraphQL**: Simple integration with Express
- **GraphQL Yoga**: Easy setup with subscriptions and file uploads

Examples

Full Example: Query with Variables

```
query GetUser($userId: ID!, $showEmail: Boolean!) {
  user(id: $userId) {
    name
    email @include(if: $showEmail)
  }
}
```

Variables:

```
{
  "userId": "1",
  "showEmail": true
}
```

Full Example: Mutation

```
mutation CreateUser($input: UserInput!) {
  createUser(input: $input) {
    id
    name
  }
}
```

Variables:

```
{
  "input": {
    "name": "John",
    "email": "john@example.com"
  }
}
```

Tips and Tricks

Debugging

- **GraphiQL**: In-browser IDE for exploring GraphQL
- **Playground**: Interactive environment for testing queries

Security

- **Rate Limiting**: Implement rate limiting on mutations
- **Input Validation**: Validate inputs to prevent injection attacks

Documentation

- **Schema Introspection**: Use introspection to explore the schema
- **Comments**: Add comments in SDL for better documentation

Conclusion

- **GraphQL is powerful**: Leverage its flexibility and efficiency

- **Keep learning**: Stay updated with new features and best practices
- **Community support**: Engage with the GraphQL community for help and resources

By Ahmed Baheeg Khorshid

ver 1.0