GraphQL Cheatsheet

Overview of unique features

- Query language for APIs
- Allows clients to specify exactly what data they need
- Supports introspection and self-documentation
- Can be used for building efficient and flexible APIs

Querying

```
# Query for a specific field
{
  user(id: "123") {
    name
    email
  }
}

# Query for a specific field with an alias
{
  john: user(id: "123") {
    name
  }
  jane: user(id: "456") {
    name
  }
}

# Query for a field with arguments
{
  users(role: "admin") {
    name
  }
}

# Query for a field with nested objects
{
  user(id: "123") {
    name
    posts {
      title
      content
    }
  }
}
```

Mutations

```graphql
# Create a new user
mutation {
  createUser(input: {name: "John", email: "john@example.com"}) {
    id
    name
    email
  }
}

# Update an existing user
mutation {
  updateUser(id: "123", input: {name: "John Doe"}) {
    id
    name
    email
  }
}

# Delete a user
mutation {
  deleteUser(id: "123") {
    id
    name
    email
  }
}
```

Fragments

```graphql
# Define a fragment
fragment UserInfo on User {
  name
  email
}

# Use a fragment in a query
{
  user(id: "123") {
    ...UserInfo
  }
}

# Define a fragment with variables
```graphql
# Define a fragment with variables
fragment UserInfo on User {
  name
  email
  posts(first: $count) {
```
```

```
    title
    content
  }
}

# Use a fragment with variables
query($count: Int!) {
  user(id: "123") {
    ...UserInfo
  }
}
```

Directives

```
# Use a directive to conditionally include a field
{
  user(id: "123") {
    name
    email
    posts @include(if: $withPosts) {
      title
      content
    }
  }
}
```

Schema

```
# Define a schema
type User {
  id: ID!
  name: String!
  email: String!
  posts: [Post!]!
}

type Post {
  id: ID!
  title: String!
  content: String!
  author: User!
}

type Query {
  user(id: ID!): User
  posts: [Post!]!
}
```

```
type Mutation {
  createUser(input: CreateUserInput!): User!
  updateUser(id: ID!, input: UpdateUserInput!): User!
  deleteUser(id: ID!): User!
}

input CreateUserInput {
  name: String!
  email: String!
}

input UpdateUserInput {
  name: String
  email: String
}
```

Resources

- GraphQL documentation
- GraphQL tutorial
- GraphQL forum for community support and troubleshooting.