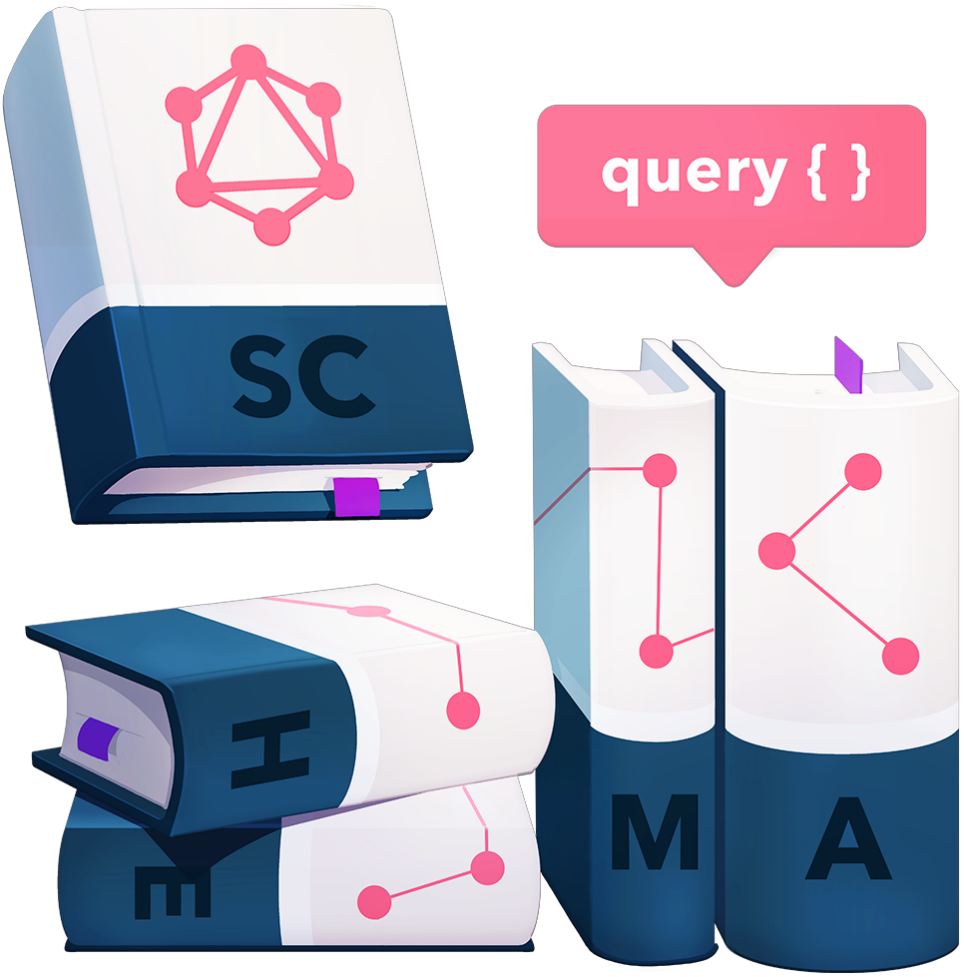
egghead.io

GraphQL Query Language

Cheat Sheet by Eve Porcello



Variables

You can pass arguments to a GraphQL query.

```
query ($category: PetCategory $status: PetStatus) {
      allPets(category: $category status: $status) {
        id
      }
    }
```

And in Query Variables panel (in json format):

```
{
  "category": "DOG",
  "status": "AVAILABLE"
}
```

If a value **is not** provided (eg. by user input or in Query Variables panel), you can set up a **default value**:

```
query ($category: PetCategory=CAT) {
      allPets(category: $category) {
        id
      }
    }
```

Operation Names

If there are multiple GraphQL queries in the same query document, you need to give the query an operation name.

```
query PetPage {
  ...
}
query CustomerPage {
  ...
}
```

Aliases

Use GraphQL Aliases to query the same field with different arguments and solve the naming collisions problem.

```
query {
      available: totalPets(status: AVAILABLE)
      checkedOut: totalPets(status: CHECKEDOUT)
    }
```

Subscription

Whenever you have any real-time needs in your application, you're going to use a subscription. Then you can use a mutation to trigger that change.

```
subscription {
      petReturned {
        pet {
          name
        }
      }
    }
```

```
mutation {
      checkIn(id: "C-2") {
        pet {
          name
        }
      }
    }
```

Fragments

Fragments are selection sets that can be used across multiple queries.

Define a PetDetails fragment:

```
fragment PetDetails on Pet {
      name
        weight
        category
        status
    }
```

Use spread `...` syntax to push all of the `PetDetails` into this query:

```
query {
      allPets(category: RABBIT, status: AVAILABLE) {
        ...PetDetails
      }
    }

    fragment PetDetails on Pet {
      name
        weight
        category
        status
    }
```

Mutations

Mutations give you the ability to invoke backend functions from the client.

Use an Input Type to Create an Account with a GraphQL Mutation:

```
mutation ($input: CreateAccountInput!) {
      createAccount(input: $input) {
        username
        name
      }
    }
```

in Query Variables panel:

```
{
  "input": {
    "name": "Eve Porcello",
    "username": "ep123",
    "password": "pass"
  }
}
```

Authenticate a User with a GraphQL Mutation:

```
mutation {
      login(username: "ep123" password: "pass") {
        customer {
          name
        }
        token
      }
    }
```

Union

Use unions whenever you want to return a list of multiple types.

```
query {
  familyPets {
    __typename
    ... on Cat {
      name
      sleepAmount
    }
    ... on Dog {
      name
      good
    }
  }
}
```

Eve Porcello

moonhighway.com · twitter.com/eveporcello

I'm the co-owner of Moon Highway, a training and curriculum development company focused on JavaScript, React, and GraphQL. We teach classes to engineers in-person and online, and we wrote O'Reilly's Learning GraphQL and Learning React. I live in Tahoe City, California and love to ski, backpack, hike, and look at trees and lakes as much as possible.