GraphQL 快速参考

Schema、查询、变更、类型、片段

Schema 定义

Schema 根类型

```
schema {
  query: Query
  mutation: Mutation
}
```

对象类型

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

查询

基础查询

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

命名查询与别名

```
query GetUsers {
  admin: user(role: ADMIN) { name }
  guest: user(role: GUEST) { name }
}
```

变更

基础变更

```
mutation {
  createUser(input: { name: "Alice" }) {
    id
    name
  }
}
```

Input 类型

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

订阅

基础订阅

```
subscription {
  messageAdded(channel: "general") {
    text
    sender { name }
  }
}
```

概览

subscription	通过 WebSocket 实现实时数据
Server push	服务端主动向客户端推送更新
Single field	每个订阅只有一个根字段

类型

标量类型

Int	有符号 32 位整数
Float	双精度浮点数
String	UTF-8 字符序列
Boolean	true 或 false
ID	唯一标识符（序列化为 String）

类型修饰符

String	可空字符串
String!	非空字符串
[String]	可空列表（元素可空）
[String!]!	非空列表（元素非空）

枚举、联合与接口

```
enum Role { ADMIN USER GUEST }
union SearchResult = User | Post
interface Node { id: ID! }
```

参数与变量

变量

```
query GetUser($id: ID!) {
  user(id: $id) {
    name
  }
}
# Variables: { "id": "123" }
```

默认值

```
query GetUsers($limit: Int = 10) {
  users(limit: $limit) { name }
}
```

片段

命名片段

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

使用片段

```
query {
  user(id: "1") { ...UserFields }
  admin: user(id: "2") { ...UserFields }
}
```

内联片段

```
query {
  search(text: "a") {
    ... on User { name }
    ... on Post { title }
  }
}
```

指令

内置指令

@include(if: Boolean!)	条件为真时包含该字段
@skip(if: Boolean!)	条件为真时跳过该字段
@deprecated(reason: String)	将字段标记为废弃

使用示例

```
query GetUser($withEmail: Boolean!) {
  user(id: "1") {
    name
    email @include(if: $withEmail)
  }
}
```

自省

类型自省

```
query {
  __type(name: "User") {
    name
    fields { name type { name } }
  }
}
```

Schema 自省

```
query {
  __schema {
    types { name kind }
    queryType { name }
  }
}
```

自省字段

__schema	查询 schema 的类型和指令
__type(name:)	按名称查询特定类型
__typename	返回任意对象的类型名

常用模式

分页（Relay 风格）

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

错误处理

```
{
  "data": { "user": null },
  "errors": [{ "message": "Not found",
    "path": ["user"] }]
}
```

最佳实践

命名操作	总是为查询和变更命名
使用变量	绝不将值插值到查询字符串中
只请求所需字段	精准选择，避免过度获取
使用片段	在查询间共享字段集合