GRAPHQL クイックリファレンス

スキーマ、クエリ、ミューテーション、型、フラグメント

スキーマ定義

スキーマルート

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

オブジェクト型

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

クエリ

基本クエリ

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

名前付きクエリとエイリアス

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

ミューテーション

基本ミューテーション

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

入力型

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

サブスクリプション

基本サブスクリプション

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

概要

subscription	WebSocket 経由のリアルタイムデータ
サーバープッシュ	サーバーがクライアントに更新を送信
単一フィールド	サブスクリプションごとにルートフィールドは 1 つのみ

型

スカラー型

Int	符号付き 32 ビット整数
Float	倍精度浮動小数点
String	UTF-8 文字列
Boolean	true または false
ID	一意識別子（文字列としてシリアライズ）

型修飾子

String	null 許容文字列
String!	null 非許容文字列
[String]	null 許容文字列の null 許容リスト
[String!]!	null 非許容文字列の null 非許容リスト

enum、union、interface

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

引数と変数

変数

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

デフォルト値

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

フラグメント

名前付きフラグメント

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

フラグメントの使用

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

インラインフラグメント

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

ディレクティブ

組み込みディレクティブ

@include(if: Boolean!)	条件が true のときのみフィールドを含める
@skip(if: Boolean!)	条件が true のときフィールドをスキップ
@deprecated(reason: String)	フィールドを非推奨としてマーク

使用例

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

イントロスペクション

型のイントロスペクション

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

スキーマのイントロスペクション

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

イントロスペクションフィールド

__schema	スキーマの型とディレクティブをクエリ
__type(name:)	名前で特定の型をクエリ
__typename	任意のオブジェクトの型名を返す

よくあるパターン

ページネーション（Relay スタイル）

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

エラーハンドリング

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

ベストプラクティス

オペレーションに名前を付ける	クエリとミューテーションには常に名前を付ける
変数を使用する	クエリ文字列に値を直接補間しない
必要なフィールドのみ要求する	正確なフィールド選択でオーバーフェッチを避ける
フラグメントを使用する	クエリ間でフィールドセットを共有する