Référence rapide GraphQL

Schémas, requêtes, mutations, types, fragments

Définition de schéma

Racine du schéma

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

Type objet

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

Requêtes

Requête simple

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

Requête nommée avec alias

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

Mutations

Mutation simple

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

Types d'entrée

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

Souscriptions

Souscription simple

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

Aperçu

`subscription`	Données en temps réel via WebSocket
`Server push`	Le serveur envoie des mises à jour au client
`Single field`	Un seul champ racine par souscription

Types

Types scalaires

`Int`	Entier signé 32 bits
`Float`	Nombre flottant double précision
`String`	Séquence de caractères UTF-8
`Boolean`	true ou false
`ID`	Identifiant unique (sérialisé comme String)

Modificateurs de type

`String`	Chaîne nullable
`String!`	Chaîne non nulle
`[String]`	Liste nullable de chaînes nullables
`[String!]!`	Liste non nulle de chaînes non nulles

Enum et Union

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

Arguments et variables

Variables

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

Valeurs par défaut

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

Fragments

Fragment nommé

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

Utilisation de fragments

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

Fragment en ligne

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

Directives

Directives intégrées

`@include(if: Boolean!)`	Inclure le champ uniquement si la condition est vraie
`@skip(if: Boolean!)`	Ignorer le champ si la condition est vraie
`@deprecated(reason: String)`	Marquer un champ comme déprécié

Exemple d'utilisation

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

Introspection

Introspection de type

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

Introspection de schéma

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

Champs d'introspection

`__schema`	Interroger les types et directives du schéma
`__type(name:)`	Interroger un type spécifique par son nom
`__typename`	Retourne le nom du type de tout objet

Modèles courants

Pagination (style Relay)

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

Gestion des erreurs

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