

GraphQL cheatsheet

— Proudly sponsored by —

Gitcoin  Use your Javascript skills to grow
Open Source, while getting paid.
ethical ads via CodeFund

Queries

Basic query

```
{ status }
```

↓

```
{ status: 'available' }
```

Lookups

```
{  
  hero(id: "1000") { id name }  
}
```

↓

```
{ hero:  
  { id: "1000",  
    { name: "Luke Skywalker" } }  
}
```

Mutations

Nesting

```
{ hero { name height } }
```

↓

```
{ hero:  
  { name: "Luke Skywalker",  
    height: 1.74 } }
```

Aliases

```
{  
  luke: hero(id: "1000") { name }  
  han: hero(id: "1001") { name }  
}
```

↓

```
{ luke:  
  { name: "Luke Skywalker" },  
  han:  
    { name: "Han Solo" } }
```

Lists

```
{ friends { name } }
```

↓

```
{ friends:  
  [ { name: "Luke Skywalker" },  
    { name: "Han Solo" },  
    { name: "R2D2" } ] }
```

GraphQL queries look the same for both single items or lists of items.

Operation names and variables

Query

```
query FindHero($id: String!) {  
  hero(id: $id) { name }  
}
```

Just to make things less ambiguous. Also, to use variables, you need an operation name.

Query

```
{ createReview($review) { id } }
```

Variables

```
{ review: { stars: 5 } }
```

↓

```
{ createReview: { id: 5291 } }
```

Mutations are just fields that do something when queried.

Multiple types

```
{
  search(q: "john") {
    id
    ... on User { name }
    ... on Comment { body author { name } }
  }
}
```

Great for searching.

Variables

```
{ id: '1000' }
```

Over HTTP

GET

```
fetch('http://myapi/graphql?query={ me { name } }')
```

POST

```
fetch('http://myapi/graphql', {
  body: JSON.stringify({
    query: '...',
    operationName: '...',
    variables: { ... }
  })
})
```

Schema

Basic schemas

```
type Query {
  me: User
  users(limit: Int): [User]
}
```

Built in types

Scalar types

Int	Integer
Float	Float

Mutations

```
type Mutation {
  users(params: ListUsersInput) [User]!
}
```

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

See: [sogko/graphql-shorthand-notation-cheat-sheet](https://sogko.github.io/graphql-shorthand-notation-cheat-sheet/)

Enums

```
enum DIRECTION {  
  LEFT  
  RIGHT  
}  
  
type Root {  
  direction: DIRECTION!  
}
```

String	String
Boolean	Boolean
ID	ID
Type definitions	
scalar	Scalar type
type	Object type
interface	Interface type
union	Union type
enum	Enumerable type
input	Input object type
Type modifiers	
String	Nullable string
String!	Required string
[String]	List of strings
[String]!	Required list of strings
[String!]!	Required list of required strings

Interfaces

```
interface Entity {  
  id: ID!  
}  
  
type User implements Entity {  
  id: ID!  
  name: String  
}
```

Unions

```
type Artist { ... }  
type Album { ... }  
  
union Result = Artist | Album  
  
type Query {  
  search(q: String) [Result]  
}
```

References

<http://graphql.org/learn/queries/>

<http://graphql.org/learn/serving-over-http/>

- **0 Comments** for this cheatsheet. [Write yours!](#)

0 Comments

Cheatsheets

 **Login** ▾

 **Recommend**

 **Tweet**

 **Share**

Sort by Best ▾




Start the discussion...

LOG IN WITH

OR SIGN UP WITH DISQUS[?]

Name

Be the first to comment.

 **Subscribe**

 **Add Disqus to your site**

 **Add Disqus**

 **Disqus' Privacy Policy**

 **Privacy Policy**

 **Privacy Policy**

devhints.io / **Search 378+ cheatsheets**



Over 378 curated cheatsheets,
by developers for developers.

[Devhints home](#)

Other Databases cheatsheets

SQL joins
cheatsheet

MySql
cheatsheet

PostgreSQL
cheatsheet

Knex
cheatsheet

PostgreSQL JSON
cheatsheet

Top cheatsheets

Elixir
cheatsheet

React.js
cheatsheet

Vim
cheatsheet

ES2015+
cheatsheet

Vimdiff
cheatsheet

Vim scripting
cheatsheet