Schema

Basic schemas

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

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

See: sogko/graphql-shorthand-notation-cheat-sheet

Enums

```
enum DIRECTION {
  LEFT
  RIGHT
}

type Root {
  direction: DIRECTION!
}
```

Built in types

Scalar types	
`Int`	Integer
`Float`	Float
`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

Mutations

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

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

Queries

Basic query

```
{ status }
```

↓

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

Lookups

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

↓

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

Mutations

Query

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

Variables

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

↓

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

Mutations are just fields that do something when queried.

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" } }
```

Multiple types

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

Great for searching.

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.

Variables

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