API GraphQL Cheat Sheet

by Demetrio via cheatography.com/175545/cs/36839/

Mutations: OnChangeResponse

```
{
    " id": " Que ry",
    " dat a": {
        " gen era lQu ery Nam e": [... UPDATED DATA ...], // For queries without pagina tio n/f ilters
        " -an oth erQ uer yNa me": null // For queries with pagination and/or filter
   }
}
```

- QueryName there is top level properties on your `Query` type.
- You should list all top level QueryNames that may be affected (data could be changed) after applying mutation

Entity / ID's

`id:` **`ID!`**	Main Entity id
`userId:` **`ID!`**	Foreign Entity id
`deletedByAdminId:` **`ID`**	Foreign Entity id with modifier
`deletedByAdminName:` **`String`**	Foreign Entity field with modifier

- `deletedBy` - is modifier/action
- `Admin` - is Foreign Type/Entity name
- `Id`/`Name` - is Foreign Entity field

Filters: Date / Time

```
startTime: DateTime
endTime: DateTime
startDate: Date
endDate: Date
createdTimeFrom: DateTime
createdTimeTo: DateTime
```

Know how to deprecate

```
type MyType {
   id: ID!
    old Field: String @depre cat ed( reason: " -
old Field is deprec ated. Use newField instea d.")
    new Field: String
    ano the rField: String @depre cated
}
```

For example `userId` in the `User` Type could be deprecated, because there is `id` have been added
https://spec.graphql.org/June2018/#sec--deprecated

Non Nullable

Ensure that all fields that always should have a value is marked as Non Nullable (!) in response

For example Nullable Boolean have a three state which makes a confusion.
Only really optional fields, may be Nullable

Group Fields

```
type GeoPoint {
   lat: Float!
   lon: Float!
}
type Address {
    street: String
    city: String
    cou ntry: String!
   geo: GeoPoint
}
type User {
    add ress: Address
}
```

Do not shy to group related fields.
That simplify reading/understanding of API, reusing the same Fragments, and write common utilities to work with that

API GraphQL Cheat Sheet

by Demetrio via cheatography.com/175545/cs/36839/

Filters: Nullability

`isActive:` **`Boolean`**	
`startTime:` **`DateTime`**	
`siteId:` **`ID!`**	**Required** filter field is **Not Nullable**
`limitNames:` **`[LimitName!]`**	No Nullable Entities inside array

- Fields is **Nullable**, because `null` means that we *opt-out* filter
- Empty array is also means that we *opt-out* filter

Pagination

`totalPages:` **`NonNegativeInt!`**	0 – ∞
`totalElements:` **`NonNegativeInt!`**	0 – ∞
`page:` **`PositiveInt!`**	1 – ∞
`size:` **`PositiveInt!`**	1 – ∞
`isFirst:` **`Boolean!`**	
`isLast:` **`Boolean!`**	
`content:` **`[SomeEntity!]!`**	If there is no entities - Empty Array is pretty good!

Translations

`period: " P1Y "`	
`period Name: "One Year"`	Translation for `period` values
`type: TypeEnum`	"TEMPORATY" | "PERMANENT"
`typeName: " Tem por - aty "`	Tranlation for `TypeEnum` values

Boolean Fields

`isActive:` **`Boolean!`**
`isDeleted:` **`Boolean!`**
`hasContent:` **`Boolean!`**

- Always start boolean fields with `is` / `has` prefix. *(That's just a good practice)*
- Do not forget that Nullable Boolean have a three state: `true`, `false` and `"I'm not sure"`, don't do that!

Use specific Scalar types

Bad	Good
`country:` **`String!`**	`country:` **`Country!`**
`pageContent:` **`String!`**	`pageContent:` **`HTML!`**
`currency:` **`String!`**	`currency:` **`Currency!`**
`metadata:` **`String!`**	`metadata:` **`JSON!`**
`value:` **`String!`**	`value:` **`Duration!`**
`addressType:` **`Int!`**	`addressType:` **`AddressTypeEnum!`**

- That allow you to implement validation for corresponding type once, and reuse it everywhere.
- For FE that also allows to validate the types, and improving readability and understanding of API