

# GraphQL

#Trương Thành Tài

#Frontend Technical Lead **ekino.**

> `npx truthtai`

## WHAT IS?

**GraphQL is a query language for APIs and a runtime for fulfilling those queries with your existing data**

- Provides a complete and understandable description of the data in your API
- Gives clients the power to ask for exactly what they need and nothing more
- Makes it easier to evolve apis over time
- Enables powerful developer tools

## FIELD & List of Items

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

```
{  
  "data": {  
    "hero": {  
      "name": "R2-D2"  
    }  
  }  
}
```

```
{  
  hero {  
    name  
    # Queries can have comments!  
    friends {  
      name  
    }  
  }  
}
```

```
{  
  "data": {  
    "hero": {  
      "name": "R2-D2",  
      "friends": [  
        {  
          "name": "Luke Skywalker"  
        },  
        {  
          "name": "Han Solo"  
        },  
      ]  
    }  
  }  
}
```

## VARIABLE

```
{  
  human(id: "1000") {  
    name  
    height  
  }  
}
```

```
{  
  "data": {  
    "human": {  
      "name": "Luke Skywalker",  
      "height": 1.72  
    }  
  }  
}
```


## WHAT IS FRAGMENT?

```
{
  leftComparison: hero(episode: EMPIRE) {
    ...comparisonFields
  }
  rightComparison: hero(episode: JEDI) {
    ...comparisonFields
  }
}

fragment comparisonFields on Character {
  name
  appearsIn
  friends {
    name
  }
}
```

```
{
  "data": {
    "leftComparison": {
      "name": "Luke Skywalker",
      "appearsIn": [
        "NEWHOPE",
        "EMPIRE",
        "JEDI"
      ],
      "friends": [
        {
          "name": "Han Solo"
        },
        {
          "name": "Leia Organa"
        },
        {
          "name": "C-3PO"
        }
      ]
    }
  }
}
```

# WHAT IS QUERY?



The screenshot shows the GraphQL Playground interface. The top bar includes the 'GraphiQL' logo, a play button, and buttons for 'Prettify' and 'History'. A '< Docs' link is on the right. The main area is split into two panes. The left pane contains a query: 

```
1 query AllArticles {  
2   articles {  
3     id  
4     title  
5     author {  
6       id  
7       username  
8     }  
9   }  
10 }
```

 Below the query is a section titled 'QUERY VARIABLES' with a single variable '1'. The right pane displays the JSON response: 

```
{  
  "data": {  
    "articles": [  
      {  
        "id": 1,  
        "title": "Hello world",  
        "author": {  
          "id": 1,  
          "username": "g00glen00b"  
        }  
      },  
      {  
        "id": 2,  
        "title": "Foo",  
        "author": {  
          "id": 2,  
          "username": "admin"  
        }  
      },  
      {  
        "id": 3,  
        "title": "Writing GraphQL mutations with Spring boot",  
        "author": {  
          "id": 1,  
          "username": "g00glen00b"  
        }  
      }  
    ]  
  }  
}
```

## WHAT IS MUTATION?

```
1 mutation ($title:String, $body: String, $userId: Int) {  
2   create(title:$title, body:$body, userId:$userId) {  
3     title,  
4     body  
5   }  
6 }
```

### QUERY VARIABLES

```
1 {  
2   "title": "Test title",  
3   "body": "Test post body",  
4   "userId": 1  
5 }
```

```
{  
  "data": {  
    "create": {  
      "title": "Test title",  
      "body": "Test post body"  
    }  
  }  
}
```

## LAB: Playground

<https://min-shop.herokuapp.com/graphql>



## APOLLO CLIENT

Apollo Client is a fully-featured caching GraphQL client with integrations for React, Angular, and more. It allows you to easily build UI components that fetch data via GraphQL.

# APOLLO HOOK

1. ApolloProvider
2. ApolloConsumer
3. useQuery
4. useLazyQuery
5. useMutation
6. useSubscription
7. useApolloClient

<https://www.apollographql.com/docs/react/api/react/hooks/>

## LAB

Use Apollo GraphQL Hook to get:

- Product List
- Product Detail