DOCUMENT360

Webinar On

Documenting GraphQL APIs:
How is it different than REST?

Speaker

Mark Wentowski

API Documentation Specialist

Independent consultant

Why learn GraphQL?

- Growing popularity
- Facilitate communication: frontend /backend
- Write interactive examples
- Enhance API developer experience

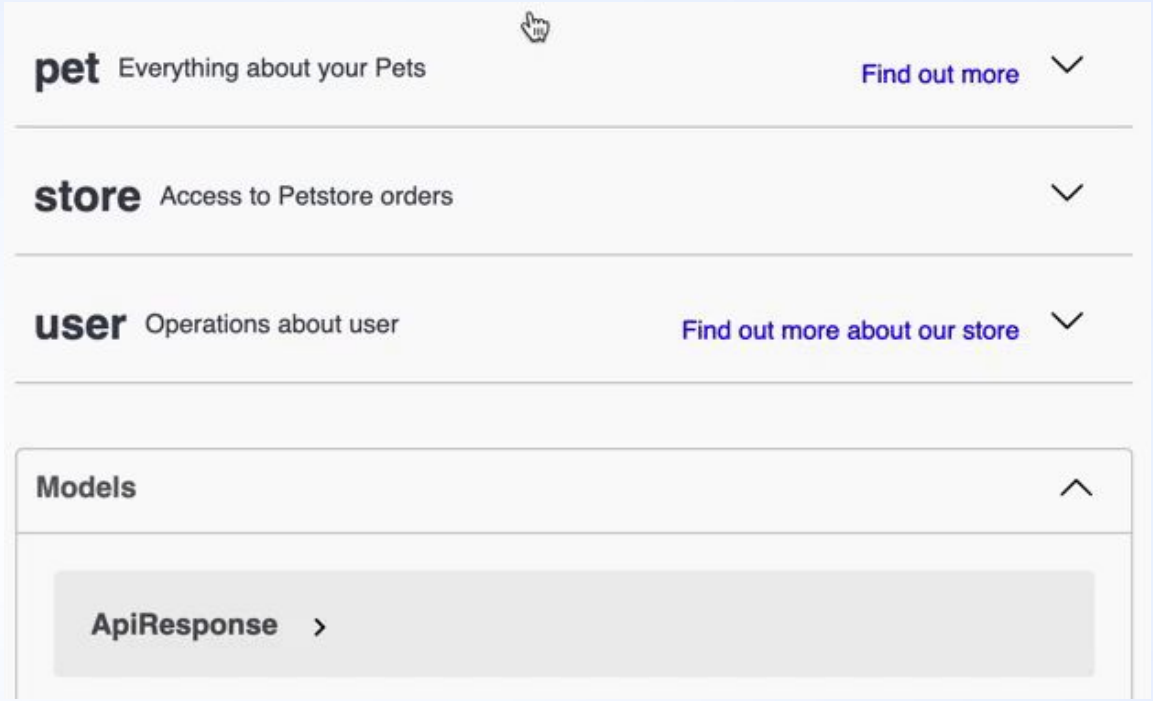
GraphQL vs. REST

- They both approaches to designing APIs
- They differ significantly in how they structure data and how clients interact with the API
- Not mutually exclusive

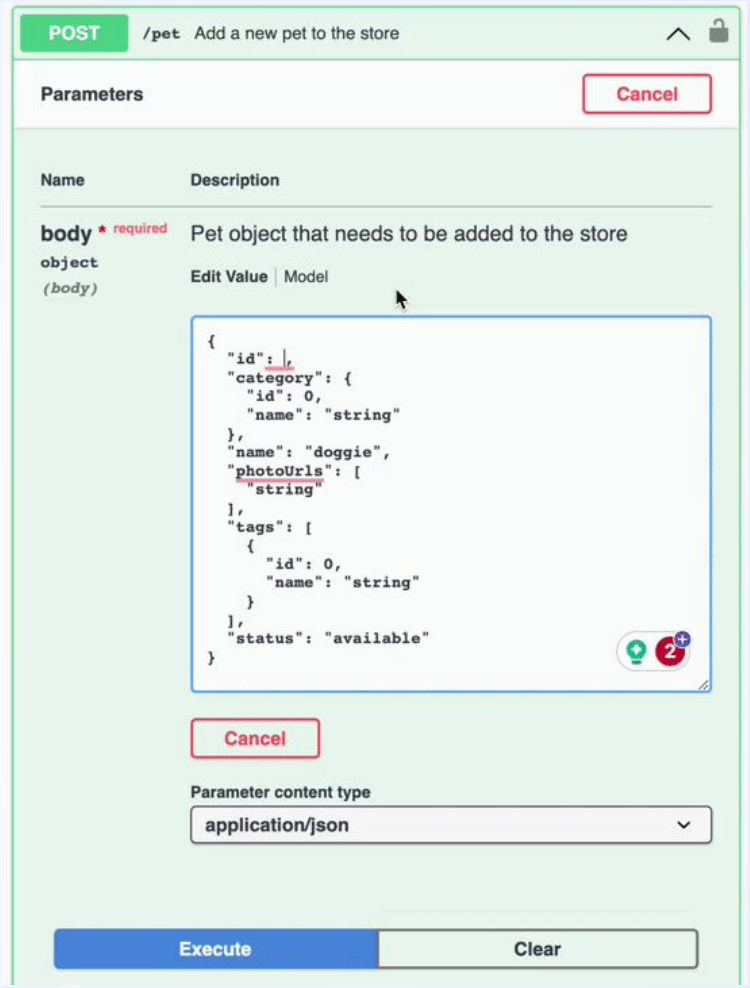
REST

- Architectural style for designing networked applications
- Manage information by using web addresses
- Strictly defined data structures

REST - 'endpoint-based'



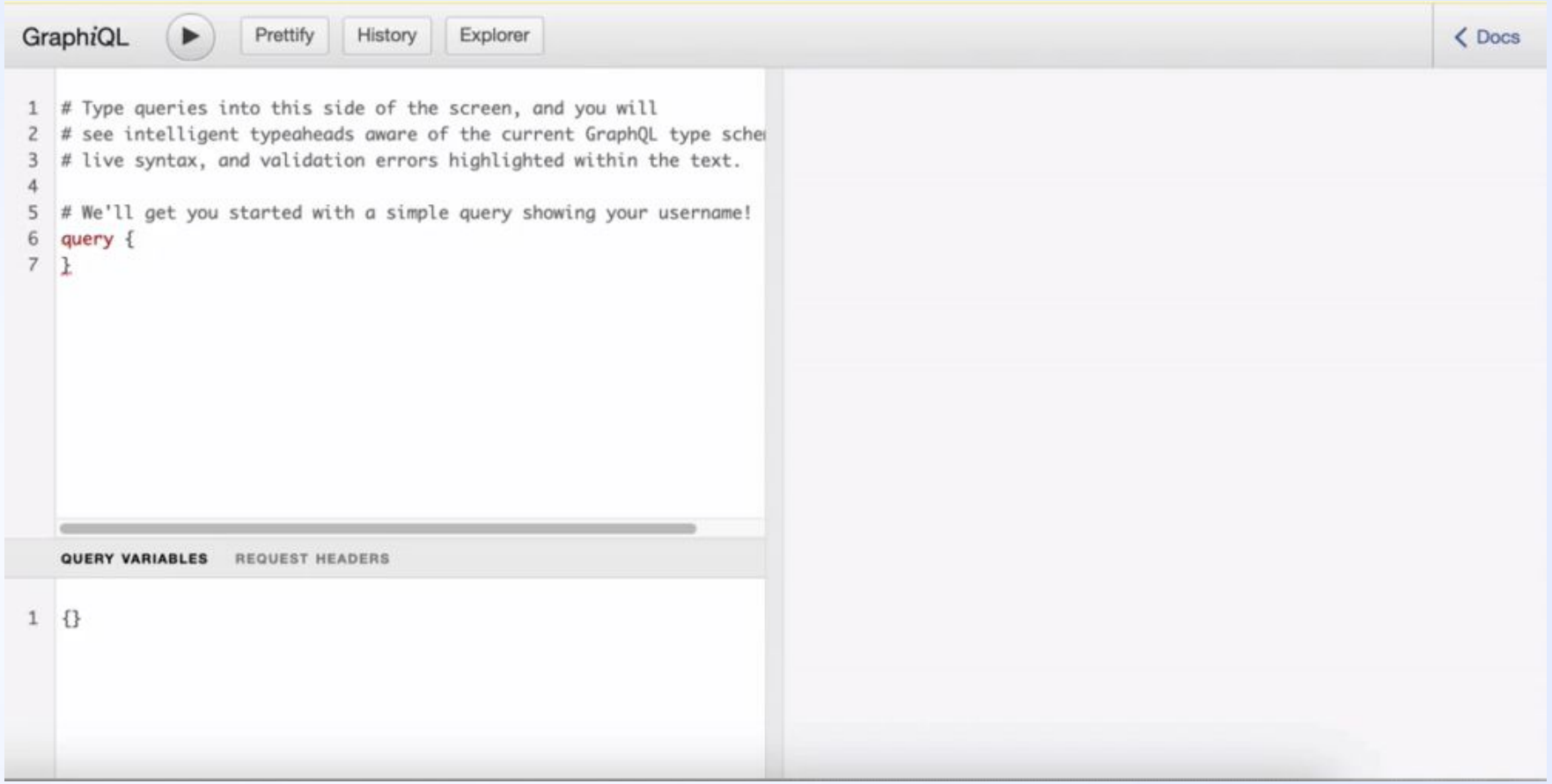
REST - 'fixed and structured'



GraphQL

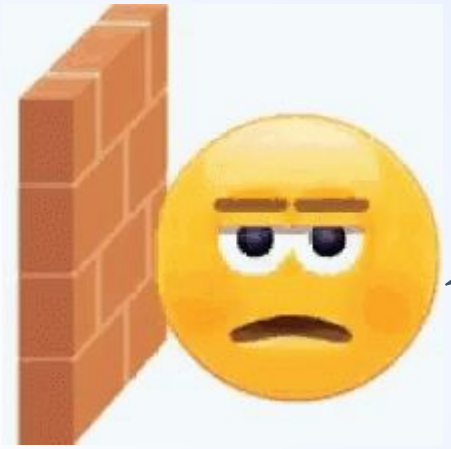
- GraphQL is a query language
- Schema-based approach to requesting data

GraphiQL – 'flexible'



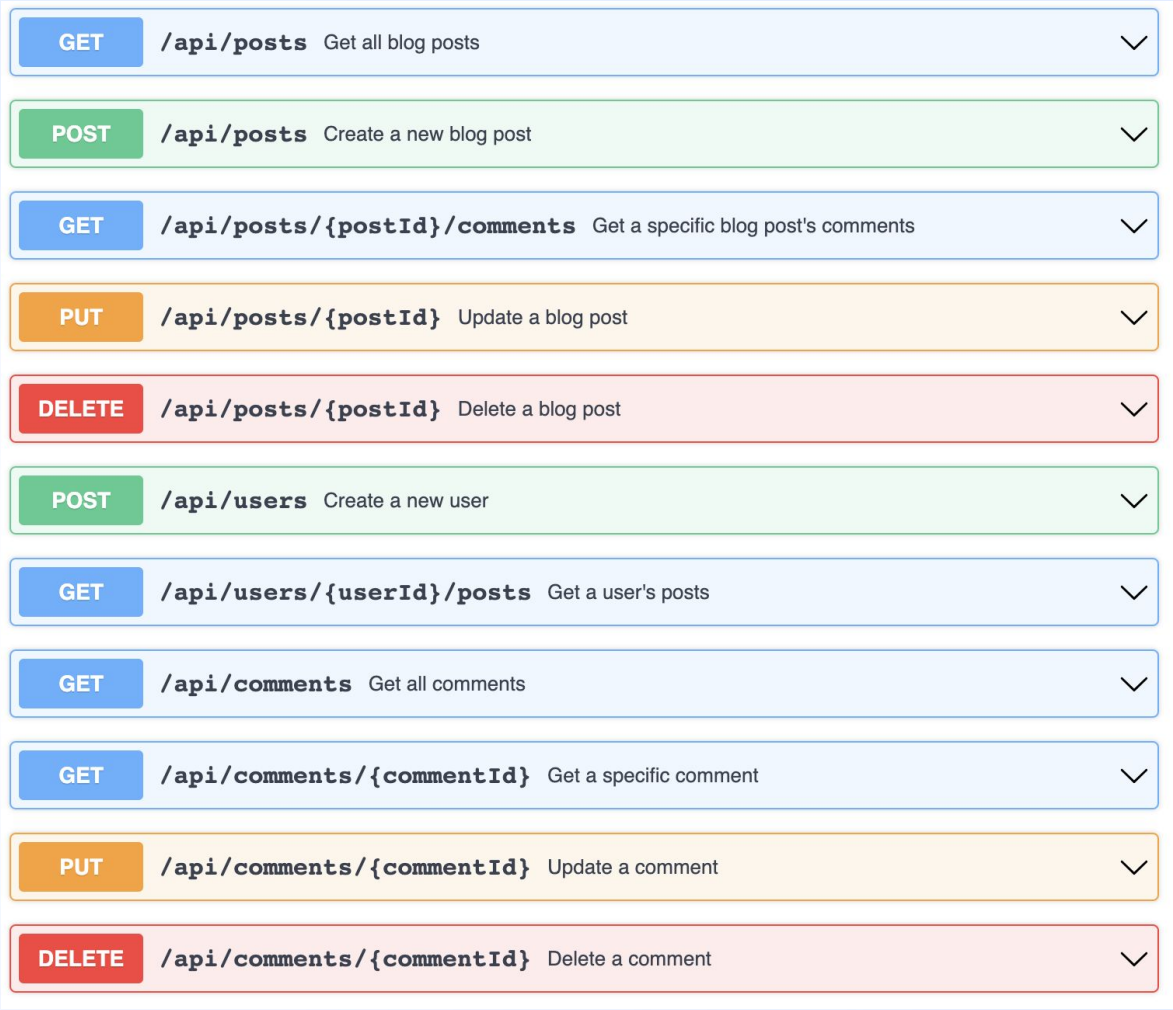
Underfetching / Overfetching

1. *Why can't I get all the data I need in one request?*



2. *Why do I have to get back all this data when I only need a subset?*

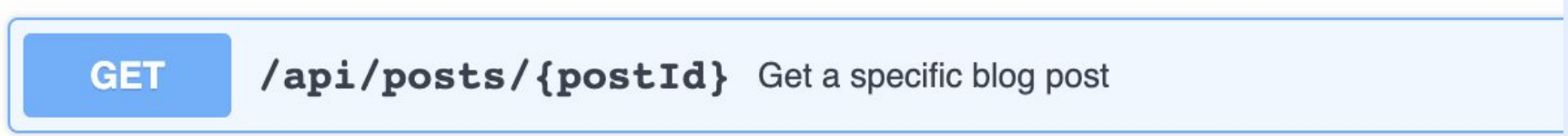
Fake Blog API



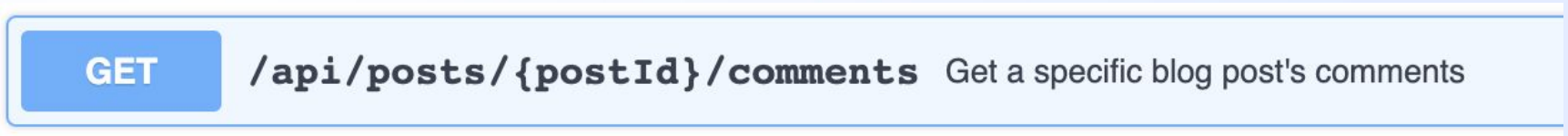
REST Example: Underfetching

I want details for a specific blog post and its comments.

Request #1:



Request #2:



GraphQL – Fetch with one request

Request

```graphql
query {
  post(id: 1) {
    id
    title
    content
    createdAt
    comments {
      id
      text
      createdAt
    }
  }
}
```

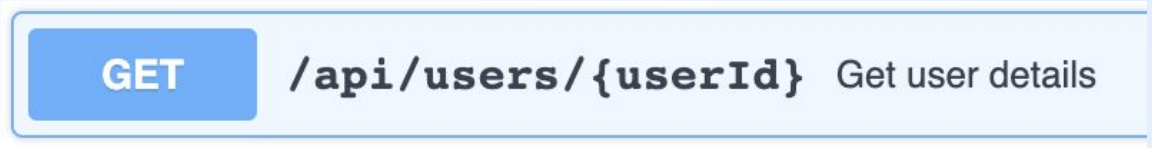
Response


```json
"data": {
  "post": {
    "id": 1,
    "title": "Sample Blog Post",
    "content": "This is a sample blog post content.",
    "createdAt": "2023-07-25T12:34:56Z",
    "comments": [
      {
        "id": 101,
        "text": "Great post!",
        "createdAt": "2023-07-25T14:00:00Z"
      },
      {
        "id": 102,
        "text": "Thanks for sharing!",
        "createdAt": "2023-07-25T15:30:00Z"
      }
```

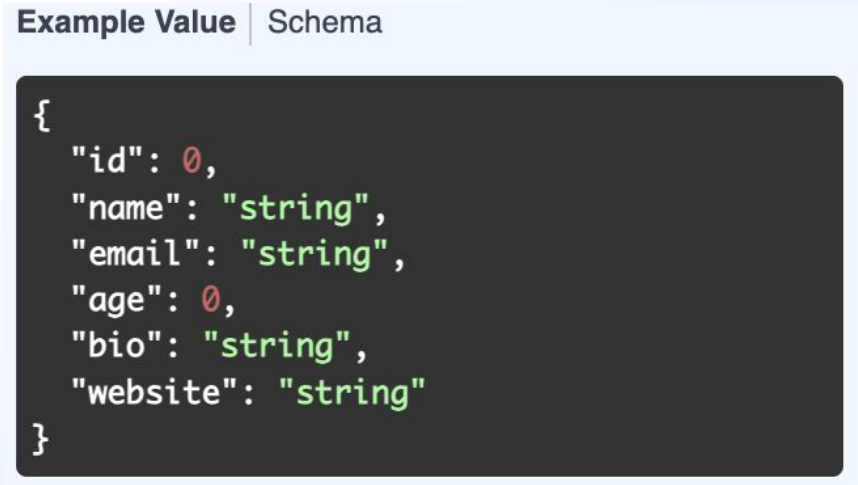
REST Example: Overfetching

I want specific user details.

Request:



Response:



GraphQL - Fetch subset of data

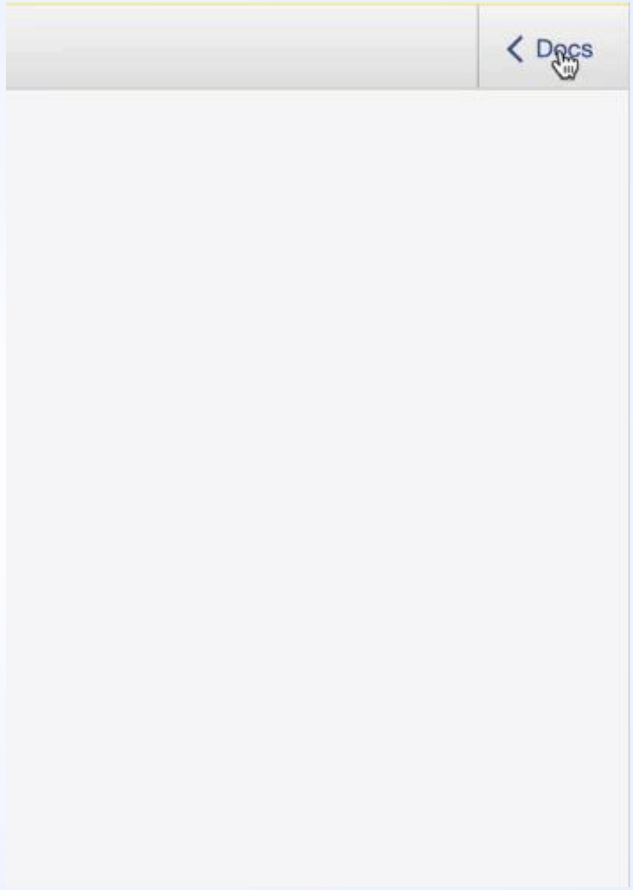
Request

```
query {
  user(id: 1) {
    email
    bio
  }
}
```

Response


```
{
  "data": {
    "user": {
      "email": "john.doe@example.com",
      "bio": "Passionate about blogging and technology."
    }
  }
}
```

Documenting GraphQL APIs - Field descriptions



DOCUMENT360

```graphql
type Book {
  id: ID!
  title: String!
  author: String!
}

type Query {
  books: [Book!]!
}
```

Code comments

Type	Field	Description
Book	id	The unique identifier of the book.
Book	title	The title of the book.
Book	author	The author of the book.
Query	books	

Google sheets + scripts

```json
{
  "Book": {
    "id": "The unique identifier of the book.",
    "title": "The title of the book.",
    "author": ""
  },
  "Query": {
    "books": "Get a list of all books in the library."
  }
}
```

JSON / YML + scripts

Documenting GraphQL APIs - Conceptual docs

- Knowledge bases
- Help authoring tools
- Static site generators (Markdown / git)

Single endpoint

Example	GraphQL endpoint: • https://fakeblogapi.com/graphql
Documentation Strategy	• Provide a code example demonstrating how to send queries to the GraphQL endpoint using libraries.

Query language focus

Example	"GraphQL is the query language used by the Blog API to allow clients to fetch data from the server."
Documentation Strategy	• Introduce GraphQL and key features • Compare GraphQL with traditional RESTful APIs

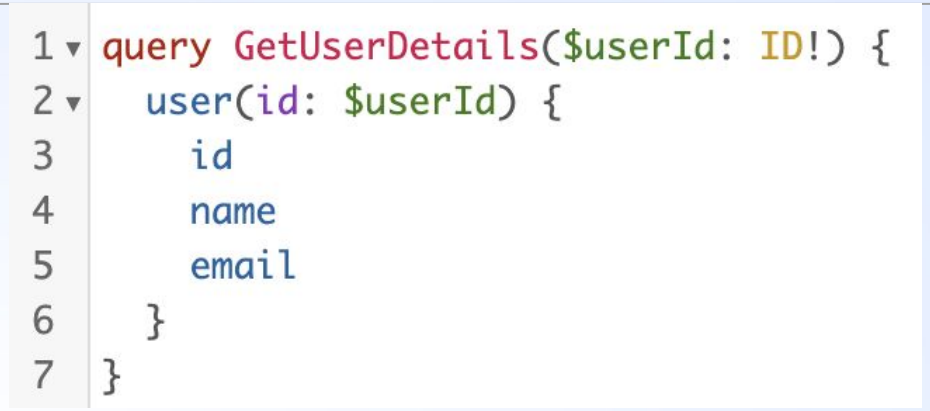
Schema documentation

Example	
Documentation Strategy	• Document types, fields and relationships.

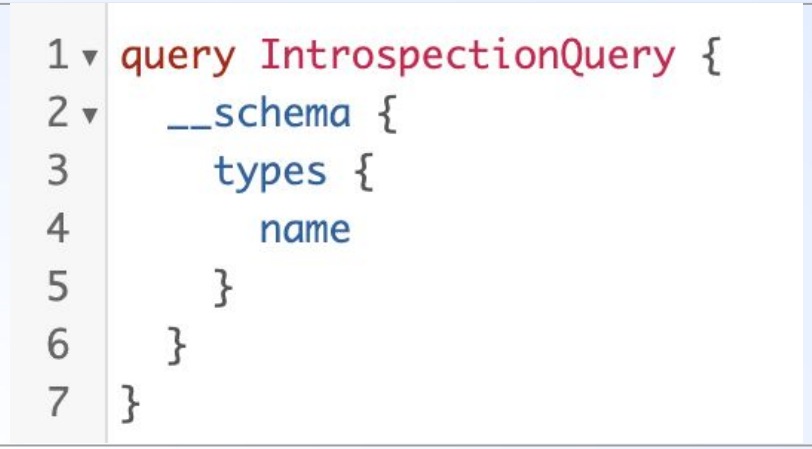
GraphiQL support

Example	“The Blog API supports GraphiQL, an interactive IDE for exploring and testing GraphQL queries."
Documentation Strategy	• Introduce GraphiQL • Access instructions • Testing examples

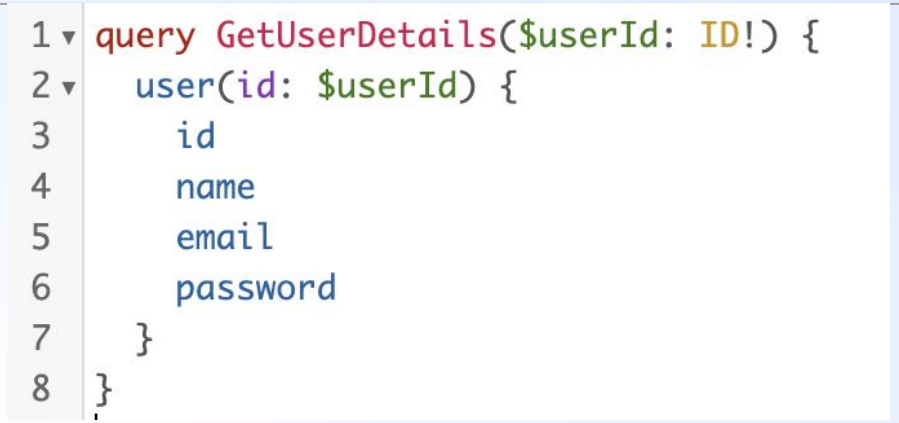
Query Variables

Example	
Documentation Strategy	• Describe role of query variables • Demonstrate how to use query variables

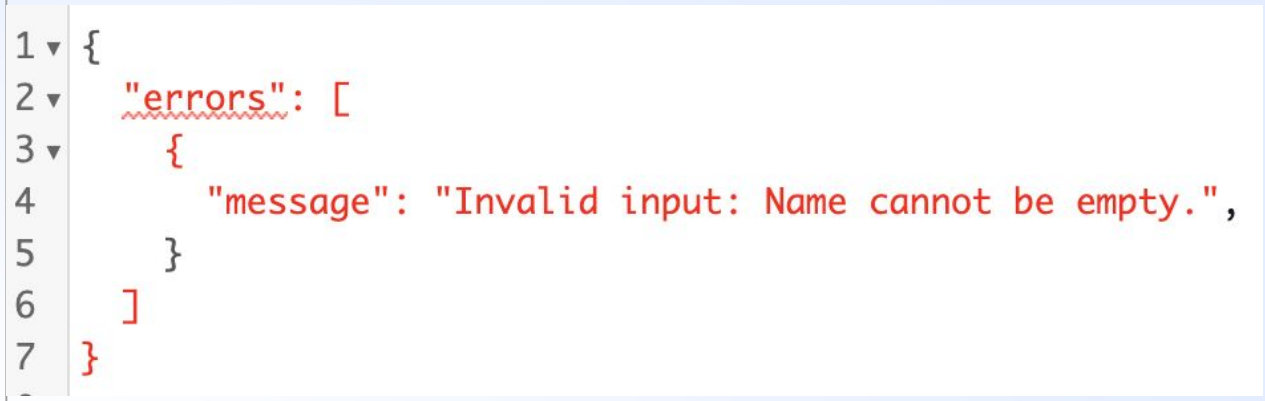
Introspection queries

Example	
Documentation Strategy	• Explain introspection in GraphQL • Provide examples that developers can execute to explore the API's schema.

Security

Example	
Documentation Strategy	• Explain that sensitive information should not be requested. • Provide best practices for ensuring confidential information is not exposed.

Error handling

Example	
Documentation Strategy	• Explain the structure of error responses • Document common error scenarios and how to handle them

Tutorials

Example	Describe the scenario where a user wants to create a new blog post through the API.
Documentation Strategy	Provide real-world examples and step-by-step guides, developers can understand how to interact with the API in practical scenarios.

Example tutorial

Create a new blog post

Explain the purpose of the mutation and its expected input fields (title, content, and authorId).

1. **Mutation Query**: Provide the mutation query with placeholders for the required variables.
2. **Query Variables**: Explain the purpose of each query variable ($title, $content, and $authorId) and their expected data types.
3. **Execution**: Show how to execute the mutation with actual values for the query variables.
4. **Response**: Explain the structure of the response and how to interpret the returned data (in this case, the id, title, and createdAt fields of the newly created post).

Sample request

```
mutation CreateNewPost($title: String!, $content: String!, $authorId: ID!) {
  createPost(title: $title, content: $content, authorId: $authorId) {
    id
    title
    createdAt
  }
}
```

Sample response

```
{
  "data": {
    "createPost": {
      "id": "123",
      "title": "New Blog Post",
      "createdAt": "2023-07-25T12:34:56Z"
    }
  }
}
```

Questions ?

Thank You!