TESTING QUERIES IN GRAPHIQL

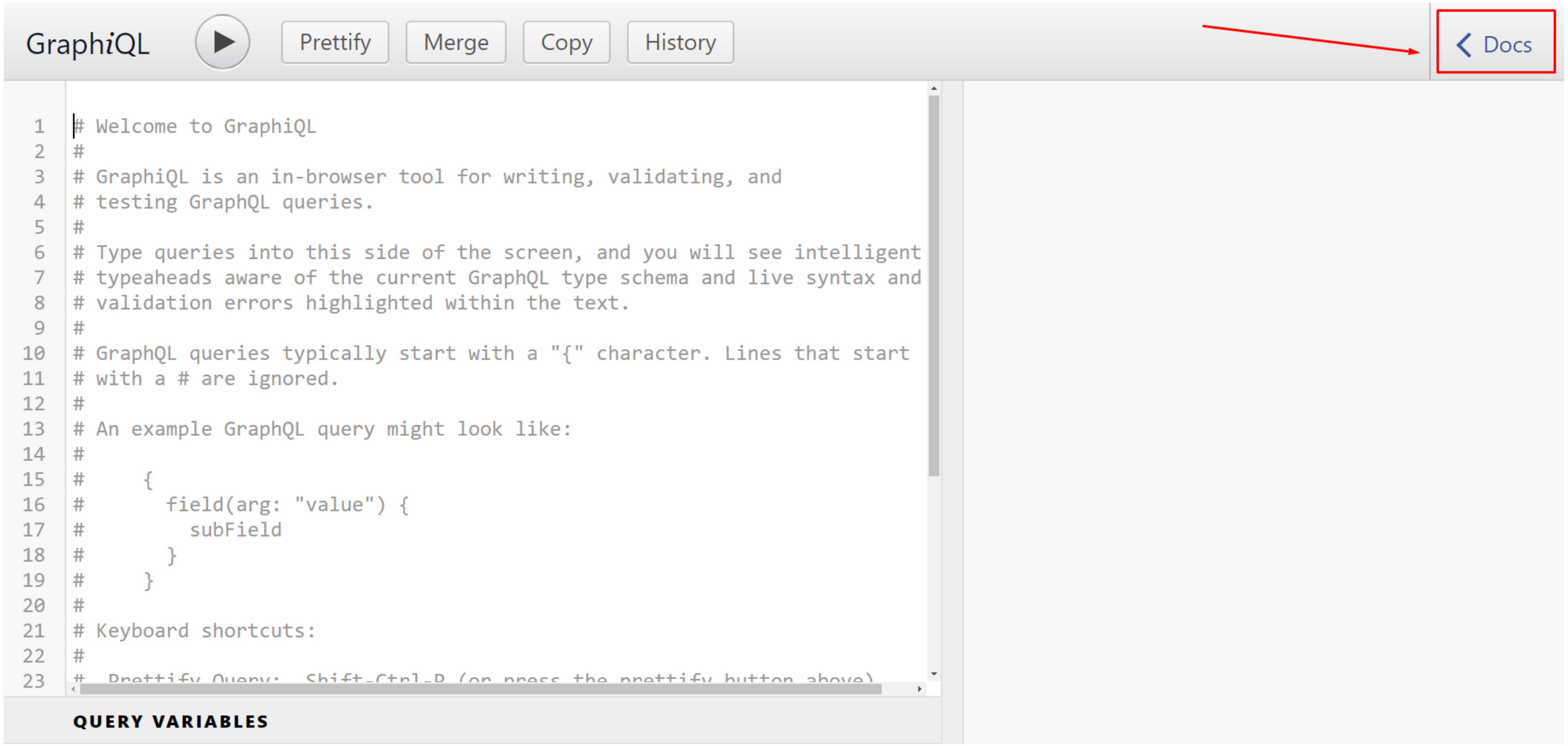
in this GraphQL tutorial I'll be explaining how we can use that front-end querying tool called Graphiql, to test out our GraphQL server.

app.js

```
const express = require('express');
const { graphqlHTTP } = require('express-graphql');
const schema = require('./schema/schema');

const app = express();

// bind express with graphql
app.use('/graphql', graphqlHTTP({
    schema,
    graphiql: true
}));

app.listen(4000, () => {
    console.log('now listening for requests on port 4000');
});
```

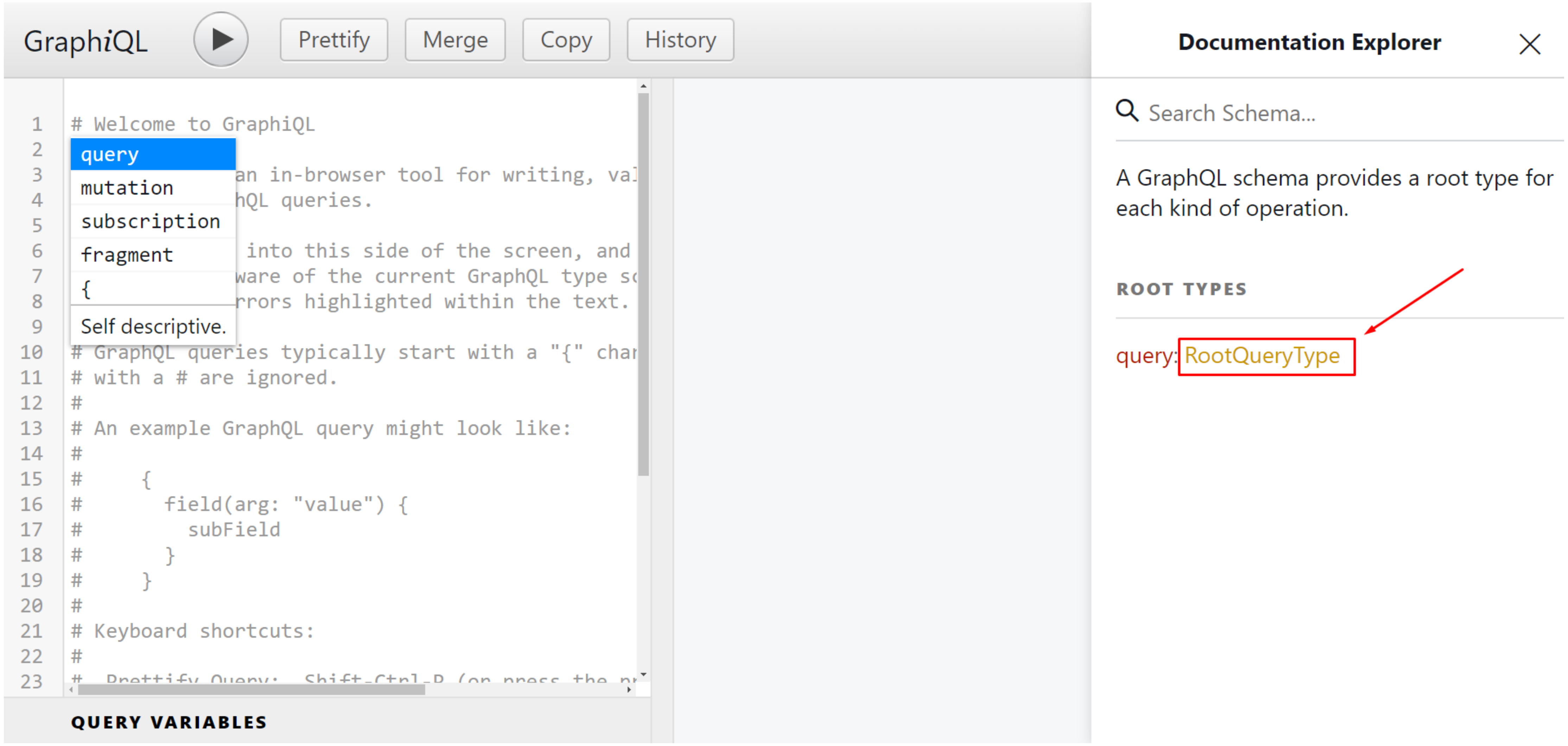
Insert code

Go to http://localhost:4000/graphql



```
# Welcome to GraphiQL
#
# GraphiQL is an in-browser tool for writing, validating, and
# testing GraphQL queries.
#
# Type queries into this side of the screen, and you will see intelligent
# typeaheads aware of the current GraphQL type schema and live syntax and
# validation errors highlighted within the text.
#
# GraphQL queries typically start with a "{" character. Lines that start
# with a # are ignored.
#
# An example GraphQL query might look like:
#
#     {
#       field(arg: "value") {
#         subField
#       }
#     }
#
# Keyboard shortcuts:
#
#  Prettify Query:  Shift-Ctrl-P (or press the prettify button above)
```

Go to http://localhost:4000/graphql



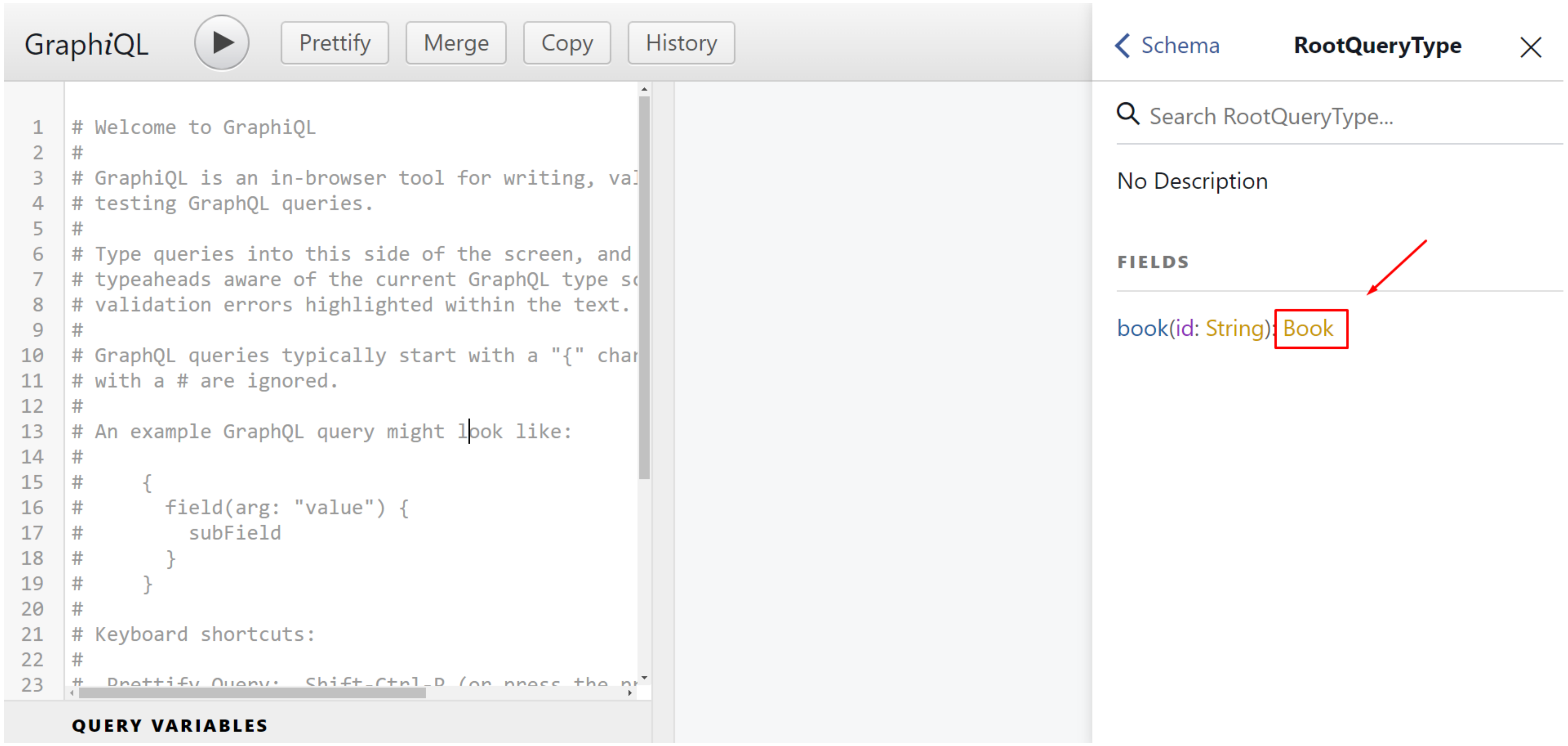
```
# Welcome to GraphiQL
#
#                   an in-browser tool for writing, val
#                   hQL queries.
#
#                    into this side of the screen, and
#                   ware of the current GraphQL type sc
#                   rrors highlighted within the text.
#
# GraphQL queries typically start with a "{" char
# with a # are ignored.
#
# An example GraphQL query might look like:
#
#     {
#       field(arg: "value") {
#         subField
#       }
#     }
#
# Keyboard shortcuts:
#
#  Prettify Query:  Shift-Ctrl-P (or press the pr
```

A GraphQL schema provides a root type for each kind of operation.

ROOT TYPES

query: RootQueryType

Go to http://localhost:4000/graphql



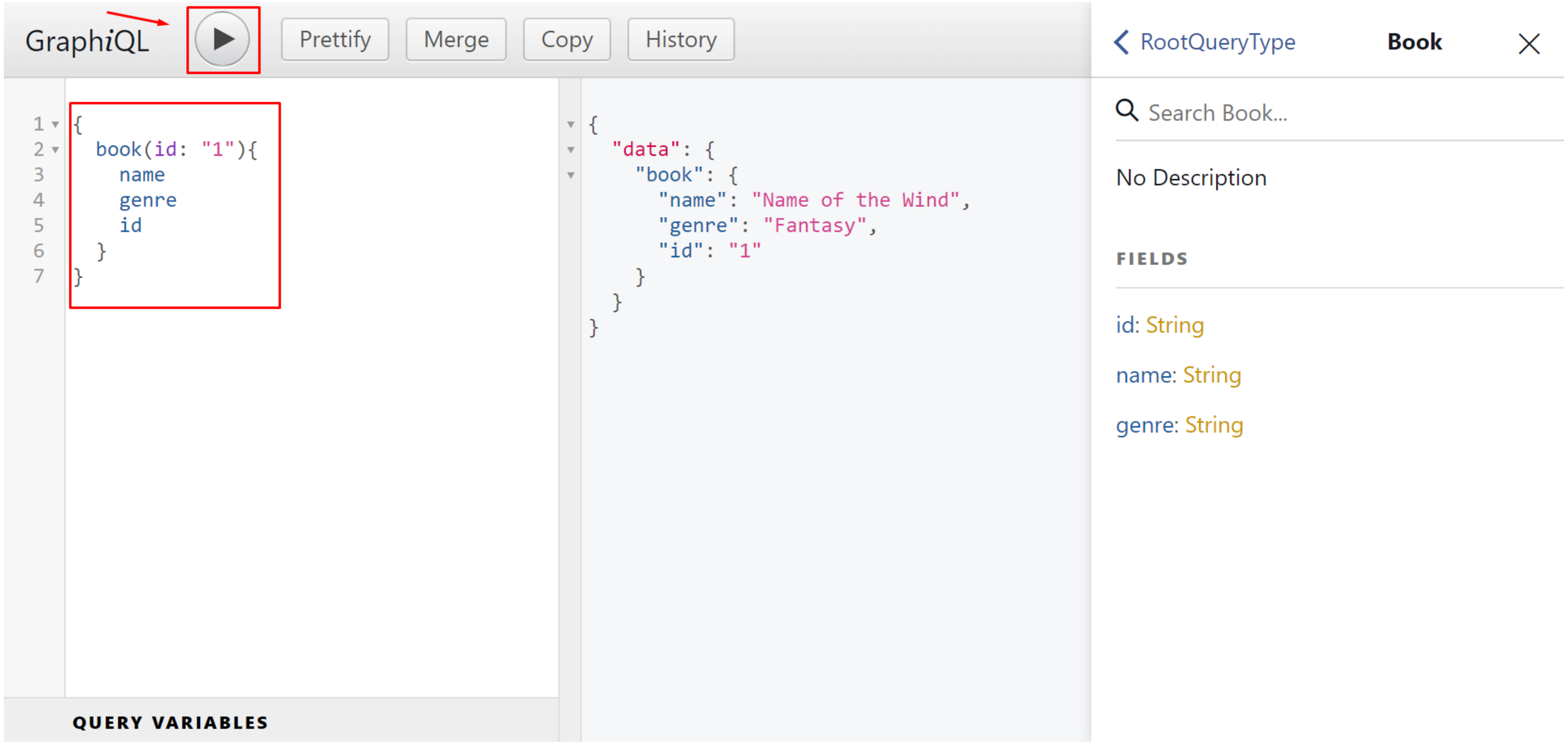
```
# Welcome to GraphiQL
#
# GraphiQL is an in-browser tool for writing, val
# testing GraphQL queries.
#
# Type queries into this side of the screen, and
# typeaheads aware of the current GraphQL type sc
# validation errors highlighted within the text.
#
# GraphQL queries typically start with a "{" char
# with a # are ignored.
#
# An example GraphQL query might look like:
#
#     {
#       field(arg: "value") {
#         subField
#       }
#     }
#
# Keyboard shortcuts:
#
#   Prettify Query:  Shift-Ctrl-P (or press the pr
```

No Description

FIELDS

book(id: String): Book

Go to http://localhost:4000/graphql



```
{
  book(id: "1"){
    name
    genre
    id
  }
}
```

```
{
  "data": {
    "book": {
      "name": "Name of the Wind",
      "genre": "Fantasy",
      "id": "1"
    }
  }
}
```

No Description

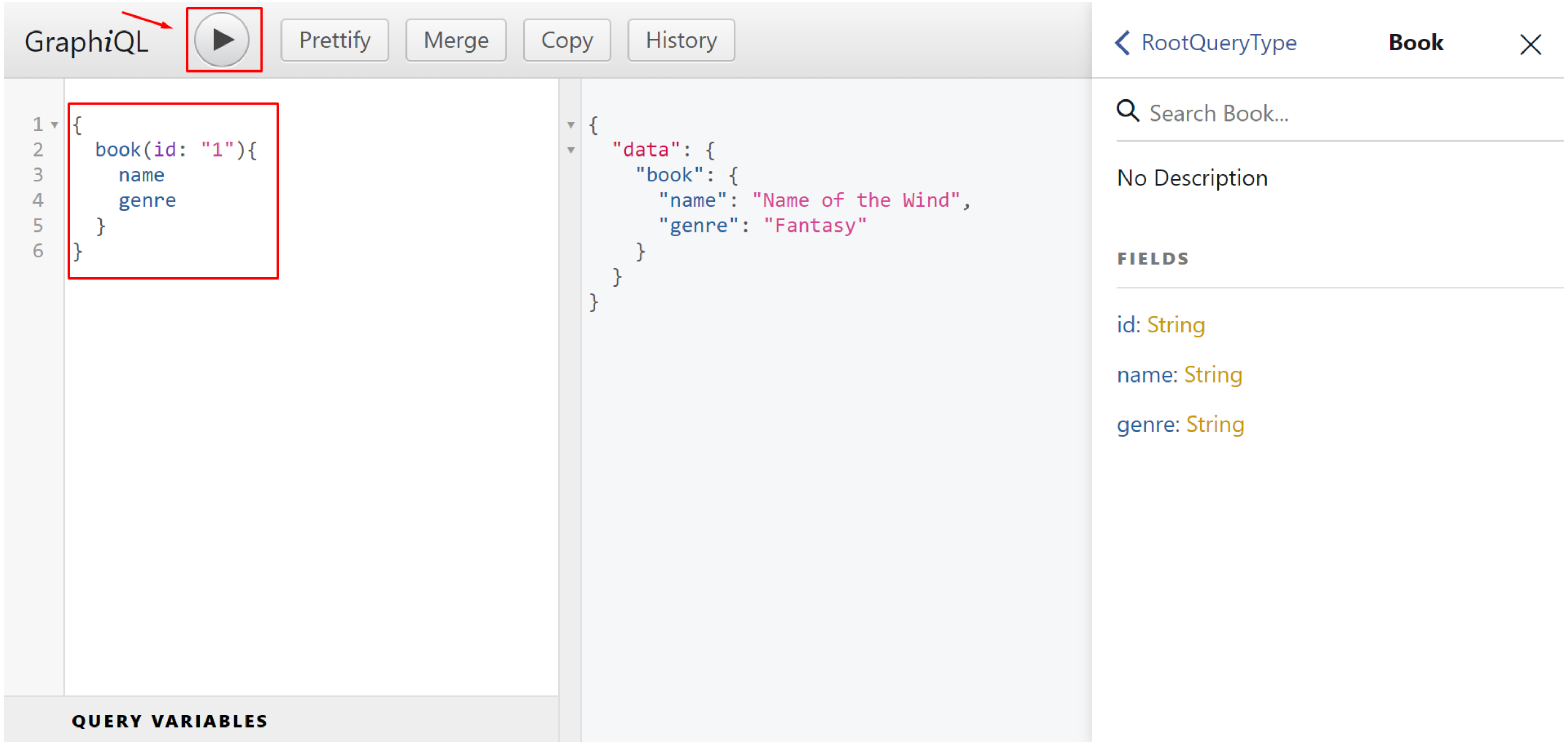
FIELDS

id: String

name: String

genre: String

Go to http://localhost:4000/graphql



```
{
  book(id: "1"){
    name
    genre
  }
}
```

```
{
  "data": {
    "book": {
      "name": "Name of the Wind",
      "genre": "Fantasy"
    }
  }
}
```

No Description

id: String
name: String
genre: String