Basic Types

In most situations, all you need to do is to specify the types for your API using the GraphQL schema language, taken as an argument to the `buildSchema` function.

The GraphQL schema language supports the scalar types of `String`, `Int`, `Float`, `Boolean`, and `ID`, so you can use these directly in the schema you pass to `buildSchema`.

By default, every type is nullable - it's legitimate to return `null` as any of the scalar types. Use an exclamation point to indicate a type cannot be nullable, so `String!` is a non-nullable string.

To use a list type, surround the type in square brackets, so `[Int]` is a list of integers.

Each of these types maps straightforwardly to JavaScript, so you can just return plain old JavaScript objects in APIs that return these types. Here's an example that shows how to use some of these basic types:

```
var express = require('express');
var { graphqlHTTP } = require('express-graphql');
var { buildSchema } = require('graphql');

// Construct a schema, using GraphQL schema language
var schema = buildSchema(`
  type Query {
    quoteOfTheDay: String
    random: Float!
    rollThreeDice: [Int]
  }
`);

// The root provides a resolver function for each API endpoint
var root = {
  quoteOfTheDay: () => {
    return Math.random() < 0.5 ? 'Take it easy' : 'Salvation lies within';
  },
  random: () => {
    return Math.random();
  },
  rollThreeDice: () => {
    return [1, 2, 3].map(_ => 1 + Math.floor(Math.random() * 6));
  },
};

var app = express();
app.use('/graphql', graphqlHTTP({
  schema: schema,
  rootValue: root,
  graphiql: true,
}));
app.listen(4000);
console.log('Running a GraphQL API server at localhost:4000/graphql');
```

If you run this code with `node server.js` and browse to http://localhost:4000/graphql you can try out these APIs.

These examples show you how to call APIs that return different types. To send different types of data into an API, you will also need to learn about passing arguments to a GraphQL API.

Continue Reading →

Passing Arguments