Before getting started, you should have Node v6 installed, although the examples should mostly work in previous versions of Node as well. For this guide, we won't use any language features that require transpilation, but we will use some ES6 features like [Promises](http://www.html5rocks.com/en/tutorials/es6/promises/), [classes](http://javascriptplayground.com/blog/2014/07/introduction-to-es6-classes-tutorial/), and [fat arrow functions](https://strongloop.com/strongblog/an-introduction-to-javascript-es6-arrow-functions/), so if you aren't familiar with them you might want to read up on them first.

To create a new project and install GraphQL.js in your current directory:

npm init

npm install graphql --save

Writing Code

To handle GraphQL queries, we need a schema that defines the Query type, and we need an API root with a function called a “resolver” for each API endpoint. For an API that just returns “Hello world!”, we can put this code in a file named server.js:

var { graphql, buildSchema } = require("graphql")

// Construct a schema, using GraphQL schema language

var schema = buildSchema(`

type Query {

hello: String

}

`)

// The rootValue provides a resolver function for each API endpoint

var rootValue = {

hello: () => {

return "Hello world!"

},

}

// Run the GraphQL query '{ hello }' and print out the response

graphql({

schema,

source: "{ hello }",

rootValue,

}).then(response => {

console.log(response)

})

If you run this with:

node server.js

You should see the GraphQL response printed out:

{

data: {

hello: "Hello world!"

}

}

Congratulations - you just executed a GraphQL query!

For practical applications, you'll probably want to run GraphQL queries from an API server, rather than executing GraphQL with a command line tool. To use GraphQL for an API server over HTTP, check out [Running an Express GraphQL Server](https://graphql.org/graphql-js/running-an-express-graphql-server/).