GET /facebook-friends
[{
  "name": "matteo",
  "country": "italy",
  "...
  // and 100s more
}, {
  ...
}]

GET /facebook-friends-names ?
GET /facebook-friends?filter=name
Nonstandard
GET /facebook-friends
GET /facebook-likes
GET /facebook-friends-and-likes ?
GraphQL
# execute friends query
friends(name: "mathias") {
  name
}

friends(name: "mathias") { 
  # only return friend.name 
  name 
}

# define your types

type Friend {
    name: String
    country: String
    age: Int
}

type Query {
    friends (name: String): Friend
}
type Friend {
    name: String
    country: String
    age: Int
}

# define your query name/input/output
type Query {
    friends (name: String): Friend
}
friends(name: "mathias") {
  name
}
friends(name: "matteo") {
  name
  age
}
// also send { name: 'mathias' } as variables
// in the same request
query Run ($name: String) {
  friends(name: $name) {
    name
  }
}
GraphQL has a rich ecosystem
A popular community GraphQL framework is Apollo
const { ApolloServer, gql } = require('apollo-server')

const typeDefs = gql`
  type Query {
    hello: String
  }
`

const resolvers = {
  Query: {
    hello: () => 'world'
  }
}

const server = new ApolloServer({ typeDefs, resolvers })
server.listen()
Makes prototyping easy!
GraphQL comes at a cost
In the last few months, several customers felt GraphQL was underperforming compared to REST.
$ clinic flame -- node express.js

--- on another terminal

$ autocannon -c 100 -d 5 \ 'http://localhost:3000/graphql?query={add\(x:2,y:2\)}'
On each and every request the server first has to parse the query to produce an Abstract Syntax Tree and then walk that to execute the query.
Caching!
Experimental implementation
Uses an LRU to cache stable queries
npm install fastify-gql
const fastify = require('fastify')()

const schema = `
  type Query {
    add(x: Int, y: Int): Int
  }
`;

const resolvers = {
  Query: {
    add: async (_, { x, y }) => x + y
  }
}

app.register(require('fastify-cors'))

app.register(require('fastify-gql'), {
  schema,
  resolvers
})

app.listen(3000)
We did additional performance analysis using clinic flame
on the fly code generation
function executor () {
  return execute
  async function _query_(_output) {
    const _res = await friends({ name: 'mathias' })
    _output.friends = _select(_res)
  }
  async function execute (input) {
    const _output = {
      friends: null
    }
    await _query(_output)
    return { data: _output }
  }
  async function execute (input) {
    const _output = {
      friends: null
    }
    await _query(_output)
    return { data: _output }
  }
  function _select (data) {
    if (Array.isArray(data)) {
      const res = []
      for (var i = 0; i < data.length; i++) {
        const item = data[i]
        const val = {
          name: item.name
        }
        res.push(val)
      }
      return res
    }
    const val = {
      name: data.name
    }
    return val
  }
}
Using our improvements, GraphQL becomes very competitive to rolling your own stack.
Come to our booth!

or check us out at www.nearform.com
Check out our workshop: A new way to profile async activity in Node.js.

At 4:40pm today in West Meeting Room 121-122!
Thanks

@mafintosh  @matteocollina