Forge Data Days

Tokyo – 29 July 2022

Takehiro Kato, Developer Advocate

AUTODESK

GraphQLの概要

GraphQL

Developed: 2012 | Public release: 2015

GraphQL

ユーザ

1stdibs. 20 minutes 99designs adayroi.com aftership airbnb ALEMBIC ALLOCINÉ AlphaSights Amplitude ANTS Appier ArangoDB ART SY Assertible ATLASSIAN attendify

AutoGuru bazinga! BetterPT Braintree Buildkite bynder cheddar cherre CircleHD CLOVERLEAF Club Med colectica commercetools Compara Conduit CONFIGURE ONE coursera craft cms

credit karma curio dailymotion DEVOPS INTERNATIONAL Drift DueDil easy carros ediket eventOne EXPERT360 FAIR Fairfax Media FILEJET FORWARD NETWORKS generation tux GetNinjas GitHub

Goalify GraphCDN graphcms GRAPHCOOL HACKAGES HIJUP Housing Anywhere hudl INSTANA intuit KLM Kyma Loggi lyft Magento Meta Nash Tech. NBC

New Relic. Northflank one medical PayPal PLURALSIGHT Rakuten ResearchGate SALSIFY satispay ScreenCloud serverless shopify stackshare SUPPLY swapcard THE GUILD

The New York Times Travaux.com Vazco wayfair Werkspot World Fuel Services yelp AUTODESK

GraphQL

CRUD

Create
Read
Update
Delete

```
mutation {
 createUser(name: “John Doe”) {
  id
 }
}
```

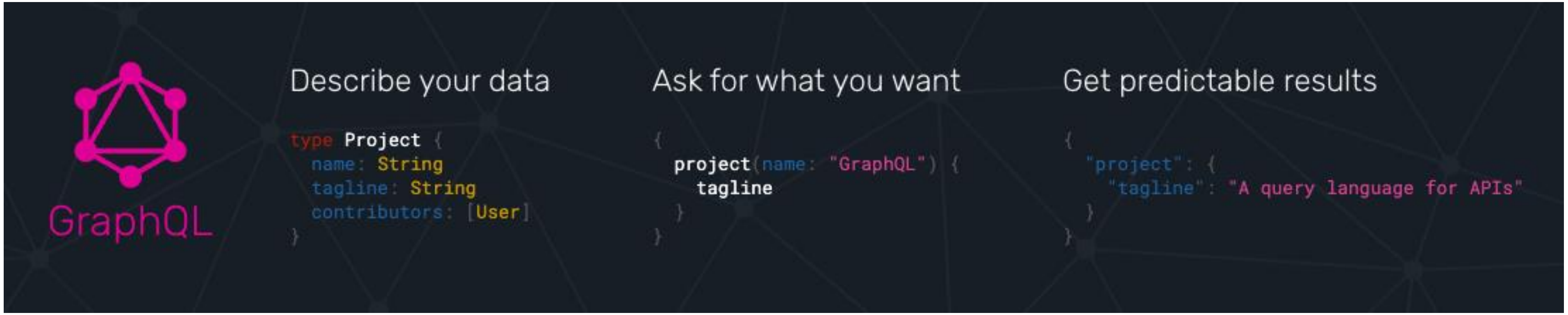
```
mutation {
 updateUser(input : {id: “1234”, address: “New place”}) {
  id
 }
}
```

```
mutation {
 deleteUser(userId : “1234”)
}
```

```
query {
 users() {
  id
  name
 }
}
```

GraphQL
ドキュメント

https://graphql.org/



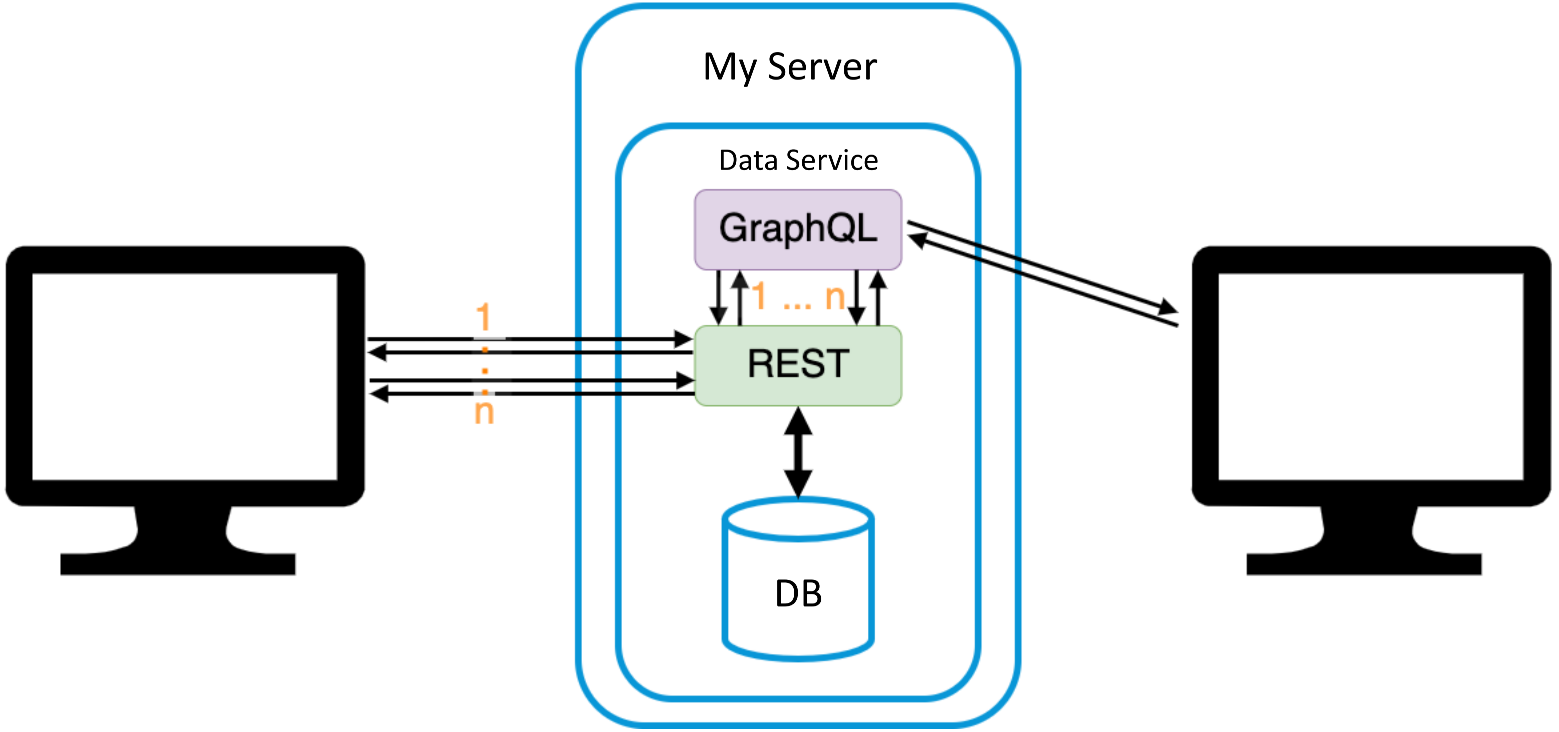
```
type Project {
  name: String
  tagline: String
  contributors: [User]
}
```

```
{
  project(name: "GraphQL") {
    tagline
  }
}
```

```
{
  "project": {
    "tagline": "A query language for APIs"
  }
}
```

GraphQL

GraphQL vs REST



GraphQL

チュートリアル

https://graphql.org/graphql-js/running-an-express-graphql-server/

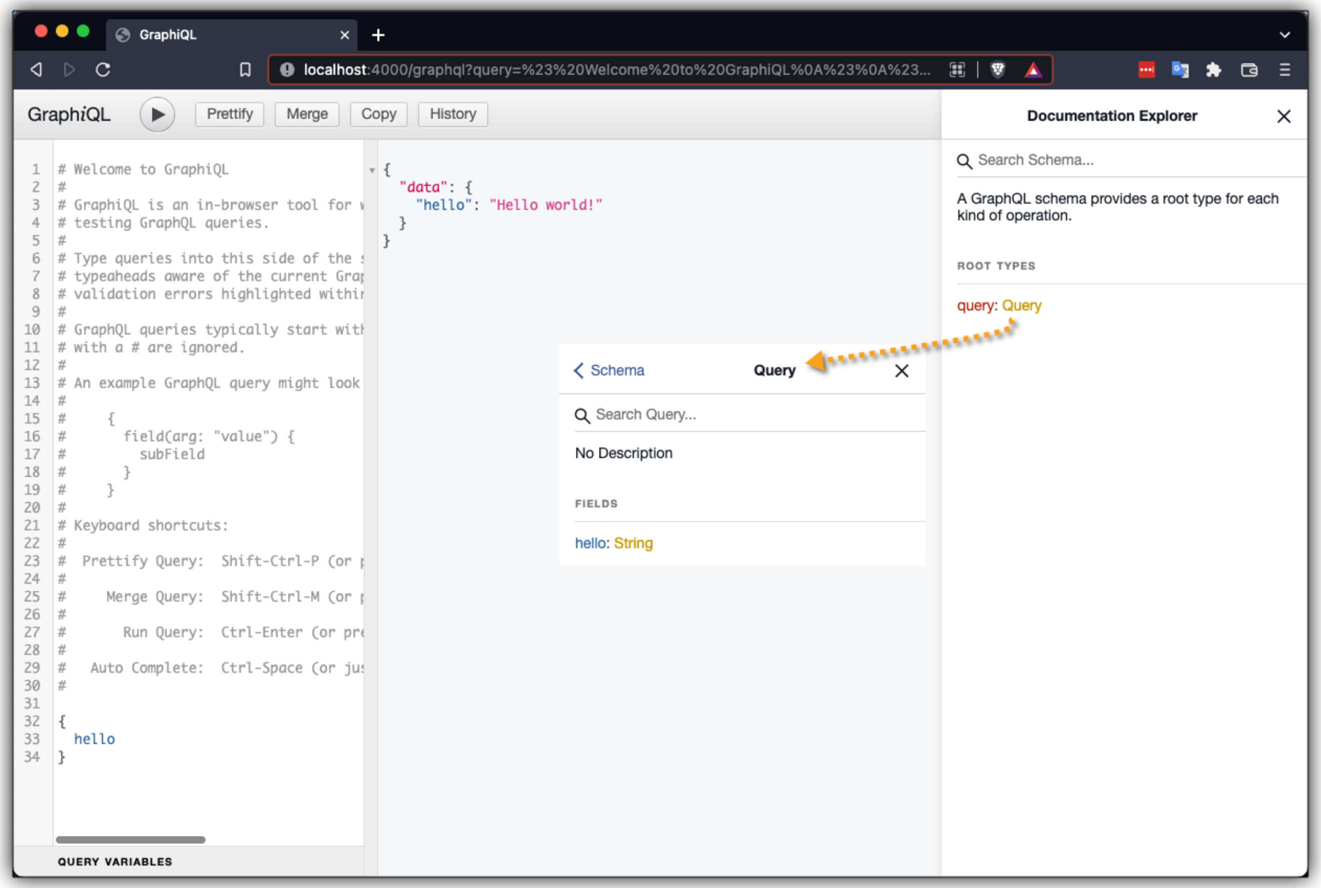
npm init

npm i express express-graphql graphql

GraphQL
チュートリアル

```
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 {
    hello: String
  }
`);
// The root provides a resolver function for each API endpoint
var root = {
  hello: () => {
    return 'Hello world!';
  },
};
var app = express();
app.use('/graphql', graphqlHTTP({
  schema: schema,
  rootValue: root,
  graphiql: true,
}));
app.listen(4000);
console.log('Running a GraphQL API server at http://localhost:4000/graphql');
```

GraphQL
チュートリアル



```
# Welcome to GraphiQL
#
# GraphiQL is an in-browser tool for w
# testing GraphQL queries.
#
# Type queries into this side of the s
# typeaheads aware of the current Grap
# validation errors highlighted withir
#
# GraphQL queries typically start wit
# with a # are ignored.
#
# An example GraphQL query might look
#
#     {
#       field(arg: "value") {
#         subField
#       }
#     }
#
# Keyboard shortcuts:
#
#  Prettify Query:  Shift-Ctrl-P (or p
#
#     Merge Query:  Shift-Ctrl-M (or p
#
#       Run Query:  Ctrl-Enter (or pre
#
#   Auto Complete:  Ctrl-Space (or jus
#

{
  hello
}
```

```
{
  "data": {
    "hello": "Hello world!"
  }
}
```

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

query: Query

No Description

hello: String

GraphQL

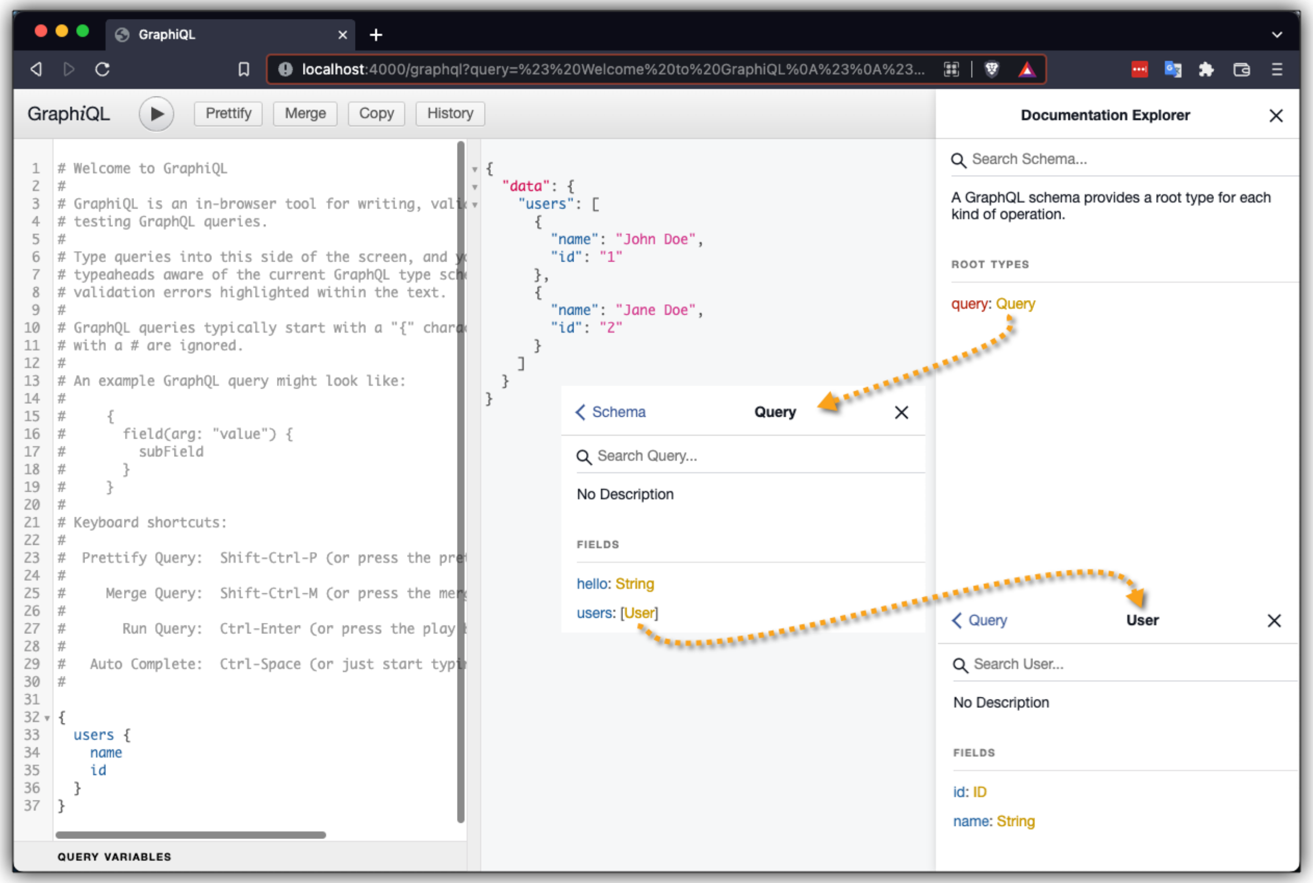
チュートリアル

```
var schema = buildSchema(`
 type Query {
  hello: String
  users: [User]
 },
 type User {
  id: ID
  name: String
 }
`);
```

```
let users = [{
 name: 'John Doe',
 id: '1'
}, {
 name: 'Jane Doe',
 id: '2'
}];

var root = {
 hello: () => {
  return 'Hello world!';
 },
 users: () => {
  return users;
 }
};
```

GraphQL
チュートリアル



```
# Welcome to GraphiQL
#
# GraphiQL is an in-browser tool for writing, vali
# testing GraphQL queries.
#
# Type queries into this side of the screen, and y
# typeaheads aware of the current GraphQL type sch
# validation errors highlighted within the text.
#
# GraphQL queries typically start with a "{" chara
# 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 pre
#
#     Merge Query:  Shift-Ctrl-M (or press the mer
#
#       Run Query:  Ctrl-Enter (or press the play
#
#   Auto Complete:  Ctrl-Space (or just start typi
#


{
  users {
    name
    id
  }
}
```

```
{
  "data": {
    "users": [
      {
        "name": "John Doe",
        "id": "1"
      },
      {
        "name": "Jane Doe",
        "id": "2"
      }
    ]
  }
}
```

Search Schema...

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

ROOT TYPES

query: Query

< Schema　Query

Search Query...

No Description

FIELDS

hello: String
users: [User]

< Query　User

Search User...

No Description

FIELDS

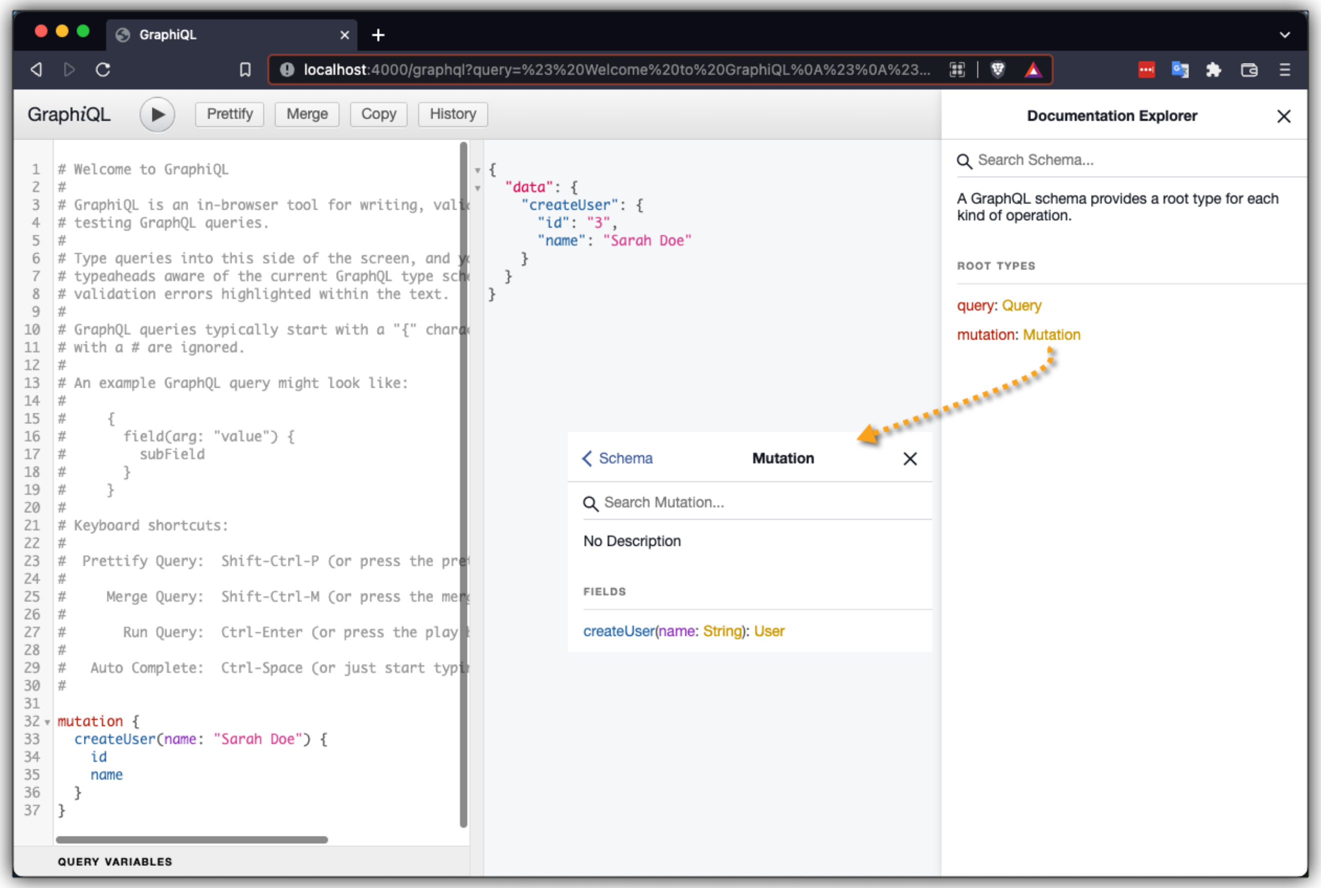
id: ID
name: String

GraphQL
チュートリアル

```
var schema = buildSchema(`
 type Query {
  hello: String
  users: [User]
 },
 type Mutation {
  createUser(name: String): User
 },
 type User {
  id: ID
  name: String
 }
`);
```

```
var root = {
 hello: () => {
  return 'Hello world!';
 },
 users: () => {
  return users
 },
 createUser: (input) => {
  let user = {
   name: input.name,
   id: users.length + 1
  };
  users.push(user);
  return user;
 }
};
```

GraphQL
チュートリアル



GraphiQL
localhost:4000/graphql?query=%23%20Welcome%20to%20GraphiQL%0A%23%0A%23...
Prettify
Merge
Copy
History
QUERY VARIABLES

```
# Welcome to GraphiQL
#
# GraphiQL is an in-browser tool for writing, vali
# testing GraphQL queries.
#
# Type queries into this side of the screen, and y
# typeaheads aware of the current GraphQL type sch
# validation errors highlighted within the text.
#
# GraphQL queries typically start with a "{" chara
# 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 pre
#
#     Merge Query:  Shift-Ctrl-M (or press the mer
#
#       Run Query:  Ctrl-Enter (or press the play
#
#   Auto Complete:  Ctrl-Space (or just start typi
#

mutation {
  createUser(name: "Sarah Doe") {
    id
    name
  }
}
```

```
{
  "data": {
    "createUser": {
      "id": "3",
      "name": "Sarah Doe"
    }
  }
}
```

Documentation Explorer

Search Schema...

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

ROOT TYPES

query: Query

mutation: Mutation

Schema — Mutation

Search Mutation...

No Description

FIELDS

createUser(name: String): User

AUTODESK