The Introduction of GraphQL ecosystem

GraphQL生态介绍

What's GraphQL?
什么是GraphQL？

A query language for your API.
一个用于API的查询语言。

What does it look like?
它什么样子？

What's the advantages?
它有什么优势？

Custom fields, custom nesting structures.
自由的字段，和嵌套结构。

One request, multiple APIs.
一个请求，多个API。

More efficient in development, cost saving.
开发上更有效率，节省成本。

```
seller(id: “30624700”) {
  id
  name
  logo
  rates {
    speedOfDelivery
    customerServiceAttitude
    qualityOfProducts
    average
  }
  numberOfProducts
}
products(filter: { sellerId: “30624700” }) {
  id
  name
  description
  images
  price
  availableQuantity
  comments {
    id
    content
    images
    rate
    createdAt
    commenter {
      id
      name
      avatar
    }
  }
}
```

Now you know GraphQL is a cool stuff
现在你知道了GraphQL是个好东西

But you may not know how to use…
但你还没知道怎么使用

Not knowing it’s ins and outs
还不清楚它的里里外外

Not knowing should you use it or not
还不清楚该不该用它

This presentation is all about answering those questions.
Introducing you the GraphQL stack,
Helping you choosing the right to use.
Improve the joy and efficiency of development.
Saving unnecessary costs and time for your company.
Improve you and your partners working, living quality.

这个分享，为你解答这些疑惑，
介绍给你GraphQL技术栈，
帮助你选择正确的技术栈，
使开发过程更效率，更快乐，
为你的公司节约成本和时间，
使你的项目同事，创业伙伴们工作，生活质量更高。

Demo: Create a simple GraphQL server
演示：创建一个GraphQL服务器

Npm packages that will be used 将会用到的npm包

· graphql, GraphQL itself, GraphQL本身
· apollo-server, the GraphQL ‘server’, 服务器本身
· merge-graphql-schemas, concat separate text graphql schemas into one, 将多个文本GraphQL schema拼接成一个。
· graphql-middleware, graphql middleware structure, used for authentication, logging etc…，graphql中间件结构，用于验证，日志等。

Besides, this presentation uses mongoose, the mongoDB ODM.
另外的，这个演示使用mongoose，这个mongoDB ODM。

Demo

How to use apollo-server?

- const { ApolloServer } = require('apollo-server');
- const server = new ApolloServer({ schema });
- server.listen(PORT);

How to use merge graphql schemas?

- const { mergeTypes } = require('merge-graphql-schemas’);

- const schemaTypes = mergeTypes(map(
- glob.sync(path.join(schemaDir, '**/*.gql')),
- (filename) => fs.readFileSync(filename).toString()
-));

Architecture of a GraphQL app
GraphQL app的结构

- Schema: The declaration of API / API的声明
- Resolver: The implementation of API / API的实现
- Model: The data and ORM / ORM和数据

- Schema是API定义，Resolver根据schema定义的内容去实现，resolver中操作model来修改或获取数据。

Create API with amur

- Amur is a GraphQL scaffolding tool. It creates models, schemas and resolvers with complicated fields, relations and data structure.
- Amur 是一个GraphQL脚手架工具。能够生成带有复杂字段，关联，和数据格式的model，schema，和resolver，也就是生成API。

Create an app with amur

- Create an app with tools above mentioned integrated.
- 创建一个应用，集成着前面提到的工具。

Demo

总结
Summary

- GraphQL以及其生态系统，向我们展示了，开发一个大型应用的后台的核心复杂功能，除了一些情况需要手动写代码，不过几行命令。
- 对于query来说，也就是传统的GET，前端自由组合请求，任意嵌套，去满足前端需要。而后台只需弄好缓存即可。
- 对于mutation来说，也就是传统的POST，跟传统的REST几乎无不同，更好的是，返回数据也可自由决定字段和嵌套。

意义
Significance

- 这样，在开发中，关于GraphQL的好与不好，你已经很清楚了，已经能够根据项目的需求来决定是否使用GraphQL。
- 若是有幸使用GraphQL，希望能够提升您和您的团队的开发效率和快乐程度。希望能够真真切切的节约宝贵的时间和资金成本，让本该成功的人，团队和项目能成功。创业或打工已经很难，让一些为这个世界做贡献的人，有更快乐的家庭和生活。

关于未来应用开发的讨论
Discuss about future app development

- SPA的出现，前端更多的像个应用，从后台中独立出来，承担着主要的业务逻辑。
- 后台应该这样，基本上靠命令，靠脚本，靠生成，来快速开发。
- 前端有着一切和用户交互的逻辑和设备，包括屏幕，各种传感器，话筒，陀螺仪等。而最主要的UI界面，在用户的屏幕中。
- 后台最重要是前端的支持以及唯一数据存储。包括数据库，文件存储，事件系统（尤其是通知），以及高负荷的算法。