API（Application Programming interface）主要有两个流派，一派是基于REST，一派是基于RPC。REST是用HTTP封装，而RPC往往用自定义的协议封装。今天REST这一派衍生出了GraphQL，而RPC这一派衍生出了gRPC。但是API的本质不会变。无论是哪种API，它的终极目的就是能让人读数据读的轻松，写数据写的愉快。

**REST API的问题**

既然要写GraphQL，就需要明白它的前辈REST。

经过长期的实践，我们也发现Restful API暴露了很多的问题：

🡪

灵活性差。同一个接口必须尽可能多的返回字段以满足不同客户端的需求，否则就得将一个接口拆分出很多返回不同字段的不同的接口，服务端的代码量接口量增加，客户端的请求量同步增加。这一点随着时间的发展、业务复杂性的提高而越发明显。例如客户端只想请求某一个博客的标题以及作者，而REST API接口将返回该博客的标题，作者，时间，正文，评论等等。

🡪

资源无法嵌套获取。如果要获取一个资源相关联的另一个资源，在大多数情况下，至少得调用两次接口才能完整获取到两个资源的数据，同第一点，不得不获取到两个资源的完整数据，大多数不是我们想要的。

🡪

文档维护困难。每次修改接口都得去重新编写或者生成文档。

🡪

无状态。所有资源通过URI定位，且这个定位与其他资源无关。

🡪

仅支持HTTP协议。

**GraphQL的解决方案**

GraphQL把上面这些问题都解决了，解决的方法就是定了这么几个规矩：

不需要GET、POST、PUT、DELETE这么多动作，一切简化为读和写

Response不会一次给全部数据，根据客户端的查询语言要什么，服务器就返回什么

PostBody可以加入variable

写API之前先在服务器端定好Schema，一切数据都得定义类型，并且数据Dependency确立好，这样Resource结构一目了然

🡪

Graphiql是GraphQL构建于浏览器的图形化调试工具。可以输入Query获取数据进行调试。

🡪

提供GraphQL服务的第三方一般会提供文档描述其GraphQL Schema，我们可以通过其Schema来了解GraphQL所支持的查询。另外，我们可以通过以下查询语句来获取schema:

{

\_\_schema{

queryType{

fields{

name

}

}

}

}

🡪