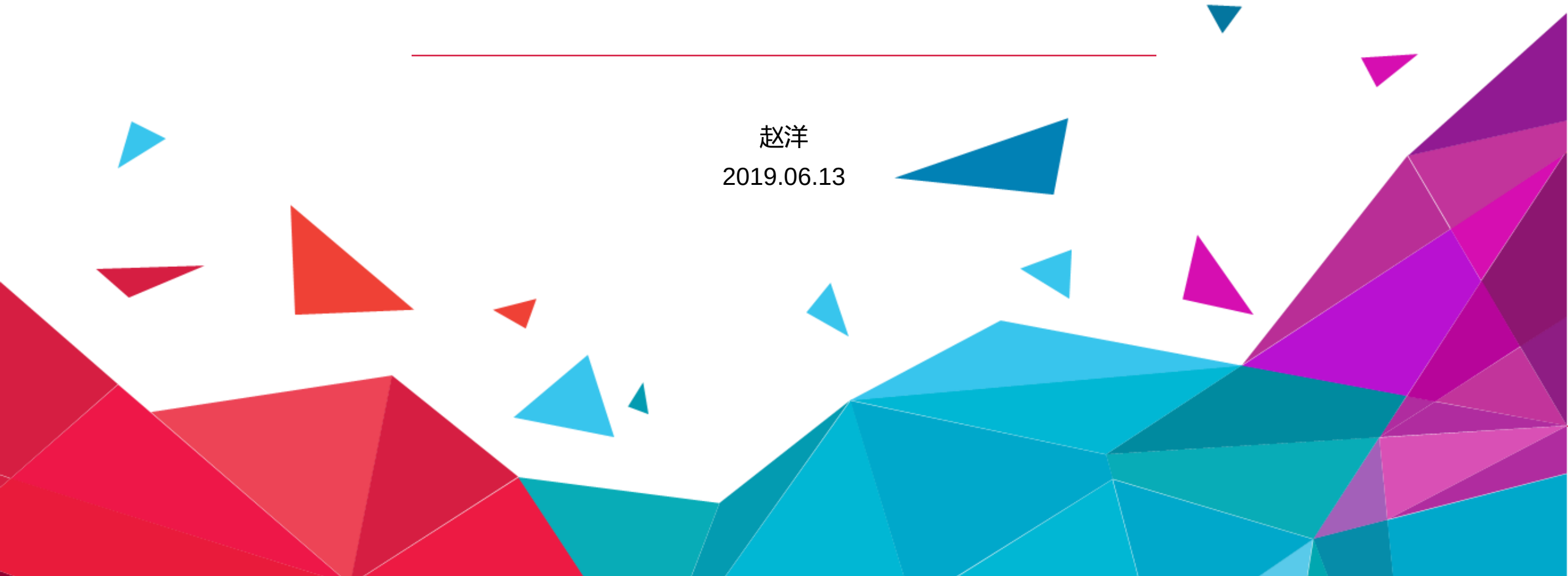
接口规范分享

赵洋

2019.06.13



/01

接口规范

把没有文档的API比作没有文档的第三方库，相信每个人都很讨厌

就像操纵一个黑盒一样，想得到A结果返回的是B，有的时候什么都没有返回

甚至有的时候返回一堆异常，导致程序报错

文档老旧，并且不够全面，相对于没有文档更坑

维护接口文档的工具有很多，如 Swagger、RAP、eoLinker

我们会经常遇到这些问题

- API 的字段更新了
- API 的某个字段被删除了
- API 的入参更新了，导致接口请求失败
- API 返回的数据结构、数据类型改变了
- API 返回了未预期的值
- ……

```
async getInitData() {
  let res = await fordGoodsList(localStorage.getItem('userId'));
  if (!res.data) return;
  let data = res.data;
  if (data.code === 1000) {
    this.addressData = data.data.address || {};
  } else if (data.code === '40001') {
    location.reload();
  } else {
    this.$Toast(data.msg || data.info);
  }
},
```

```
const res = await getLoginVCode(this.mobileNum, true);
if (res && res.data && res.data.code == 1000) {
  this.timingCode();
  this.$Toast("发送成功");
} else {
  this.$Toast("发送失败");
}
```

/02

RESTful API 通用设计规则

URL定位资源，用HTTP动词（GET,POST,DELETE）描述操作

域名

- 应该尽量将API部署在专用域名之下。
- https://api.example.org
- 也可以考虑放在主域名下
- https://example.org/api/

版本（Versioning）

- 应该将API的版本号放入URL。
- https://api.example.org/v1/
- 另一种做法是，将版本号放在HTTP头信息中，但不如放入URL方便和直观

状态码（Status Codes）

- 2XX：请求正常处理并返回
- 3XX：重定向，请求的资源位置发生变化
- 4XX：客户端发送的请求有错误
- 5XX：服务器端错误

使用正确的 Method

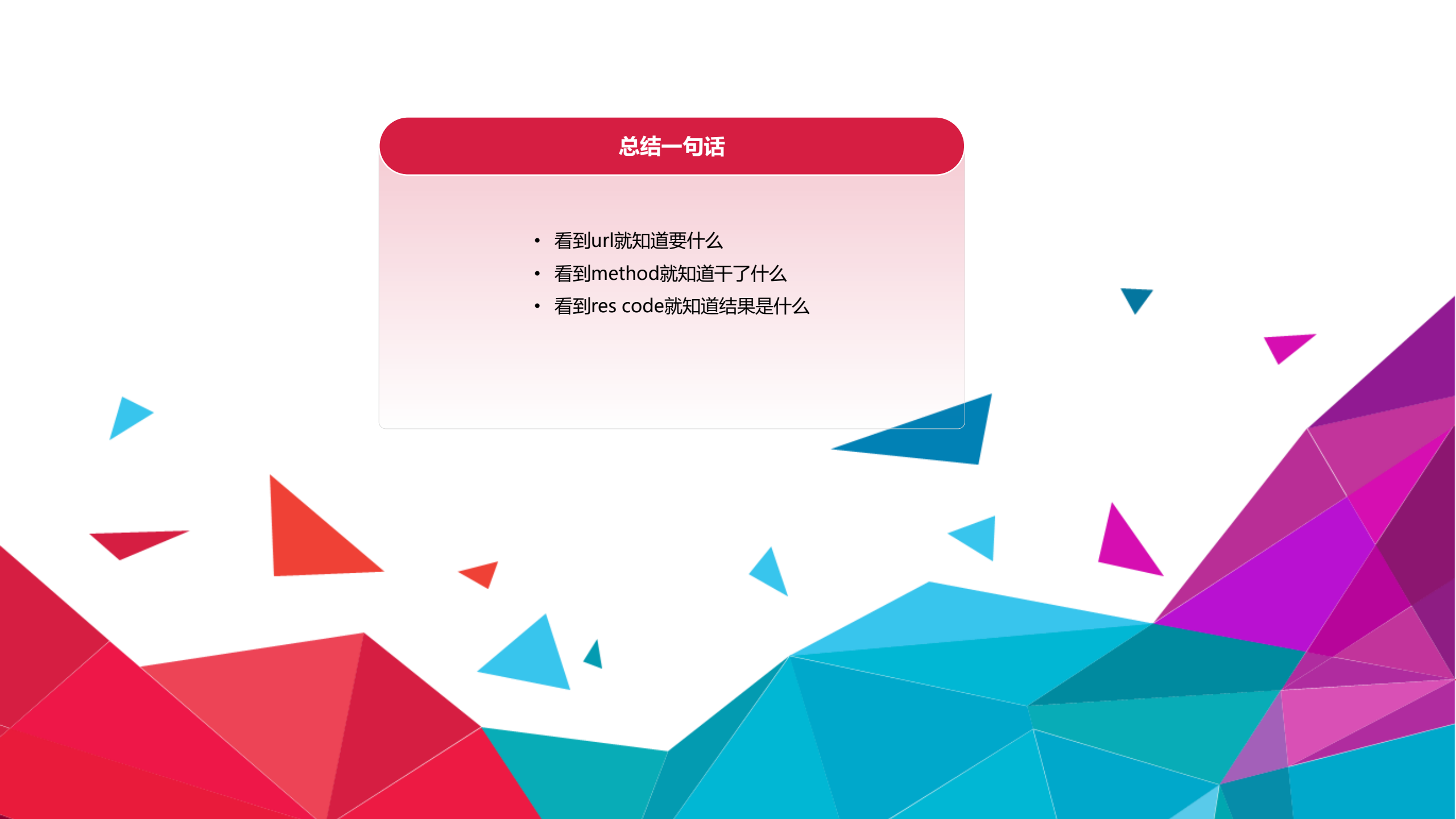
- GET（SELECT）：从服务器取出资源（一项或多项）
- POST（CREATE）：在服务器新建一个资源。
- PUT（UPDATE）：在服务器更新资源
- DELETE（DELETE）：从服务器删除资源。

面向资源的URL设计

- GET /orders 获取所有订单
- POST /orders 创建新订单
- GET /orders/{orderId} 获取指定订单
- PUT /orders/{orderId} 更新指定订单
- DELETE /orders/{orderId} 删除指定订单

总结一句话

- 看到url就知道要什么
- 看到method就知道干了什么
- 看到res code就知道结果是什么



/03

前端理想的接口

前端理想的接口

前端理想的接口

返回的数据结构

- 有序列表前端需要数组，不能是一个对象
- 如果没有值，要返回默认值
- 统一的数据结构

入参

- 标明哪些是必填参数，哪些是非必填参数
- 统一类型的接口要统一入参，如列表分页参数
- 标明参数的数据类型

前端理想的接口

返回的数据类型

- 返回的字段需要确定其数据类型，如果该字段没有值，返回该数据类型的默认值。
- 例如： int类型： 0，string类型：''， 空对象：{}，空数组：[]

```
{
    number: 0, // 0
    object: {}, // null
    array: [], // null
    string: '' // undefined
    boole: false // 'false'
}
```

- 如果某个字段返回null，要保证它不是一个字符串'null'

```
{
    correct: null,
    wrong: 'null'
}
```

THANKS