# 常见MOCK方案

1. 代码入侵（直接在代码中写Mock数据，或者请求本地的JSON数据）

优点：无

缺点：和其他方案比Mock效果不好；与真实Server环境的切换非常麻烦，一切需要入侵代码切换环境的行为都是不好的

2.请求拦截

Mock.js 原理是：重写HTMLHttpReaquest的属性，帮助返回想要的数据

例子写法如下图：



优点：与前端代码分离；可生成随机数据

缺点：数据都是动态生成的假数据，无法真实模拟增删改查的实际情况；只支持ajax，不支持fetch

3.接口管理工具

代表：rap, swagger, moco, yapi

优点：配置功能强大，接口管理与Mock一体，后端修改接口Mock也跟着更改，可靠

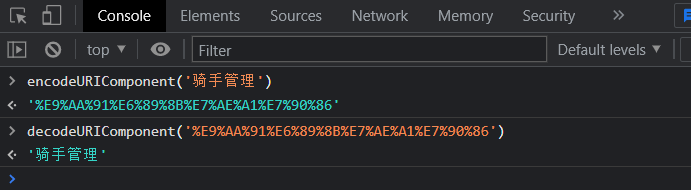
缺点：配置复杂，依赖后端，可能会出现后端不愿意出手，或者等配置完了，接口业开发出来了的情况；一般会作为大团队的基础建设而存在，没有这个条件的话慎重考虑选择

4.本地node服务器

代表：json-server

优点：配置简单，json-server甚至可以0代码 30秒启动一个REST API Server服务；自定义程度高，一切尽在掌握中；增删改查真实模拟

缺点：与接口工具相比，无法随着后端API的修改而自动修改



# qs第三方插件

A querystring parsing and stringifying library with some added security.（来自npm官方解释，详情查阅：<https://www.npmjs.com/package/qs>）

带有一些附加安全性的查询字符串解析和字符串化库，处理接口传参数时，数据量大而造成编写过长，不好书写和维护等问题



# 用Custom Hook 提取并复用组件代码

Custom Hook 是React中最新也是最优秀的组件代码复用方案

项目中最常使用的两个Hook钩子函数即 useState ; useEffect，以下将会写出 useMount 和useDebounce两个自定义Hook钩子函数实现组件逻辑的复用

# 函数编程的point free写法

const login = (form: AuthForm) => auth.login(form).then((user) => setUser(user));

通过point free写法对上述代码进行消参简写：

const login = (form: AuthForm) => auth.login(form).then(setUser)

# 计算单位(rem em)

em 相对于父元素的font-size。

rem 相对于跟元素html的font-size，r就是root的意思，即根元素。

# typescript的类型继承

interface Base {

id: number;

}

interface Person extends Base {

name: string;

}

const p: Person = {name: "123", id: 123};

Person定义的类型接口继承自Base类型接口，所以Person类型接口同时拥有id 和name两个变量的类型定义，此时变量p所定义的name和id子变量都规定其指定的数据类型