# Mock

1. **mock.js是什么？**

（1）基于 数据模板 生成模拟数据；

（2）基于 HTML模板 生成模拟数据；

（3）拦截并模拟 ajax 请求覆盖（拦截） Ajax 请求，目前内置支持 jQuery、Zepto、KISSY。；

1. **解决的问题？**

（1）开发时，后端还没完成数据输出，前端只好写静态模拟数据；

（2）数据太长了，将数据写在js文件里，完成后挨个改url；  
（3）某些逻辑复杂的代码，加入或去除模拟数据时得小心翼翼；  
（4）想要尽可能还原真实的数据，要么编写更多代码，要么手动修改模拟数据；  
（5）特殊的格式，例如IP,随机数，图片，地址，需要去收集；  
（6）超烂的破网速.....以上都不再是问题；

1. **缺点**

如果是接口式开发的话，经常是前端已经写好了页面，后端的接口还不能给。有时为了让页面正常显示，前端开发人员就会造一堆静态假数据，等接口写好，再调接口，删除静态数据。于是就想能不能自己模拟数据，mock.js可以实现。可以显示模拟数据，but有一个疑问，可以接收参数吗？例如简单的翻页，传入参数页码，显示对应的页码的数据。这是可以做到的嘛，查看了一些博客，好像都没有提到。如果是不能的话，好像不是让人特别满意~

1. **安装**

引入mock.js包(https://github.com/nuysoft/Mock)

1. **语法**

Mock.js 的语法规范包括两部分：

数据模板定义（Data Temaplte Definition，DTD）

数据占位符定义（Data Placeholder Definition，DPD）

数据模板定义 DTD

数据模板中的每个属性由 3 部分构成：属性名、生成规则、属性值：

// 属性名 name// 生成规则 rule// 属性值 value'name|rule': value

注意：

属性名 和 生成规则 之间用 | 分隔。

生成规则 是可选的。

生成规则 有 7 种格式：

'name|min-max': value

'name|count': value

'name|min-max.dmin-dmax': value

'name|min-max.dcount': value

'name|count.dmin-dmax': value

'name|count.dcount': value

'name|+step': value

生成规则 的 含义 需要依赖 属性值 才能确定。

属性值 中可以含有 @占位符 。

属性值 还指定了最终值的初始值和类型。

生成规则和示例：

1. 属性值是字符串 String

'name|min-max': 'value' 通过重复 'value' 生成一个字符串，重复次数大于等于 min，小于等于 max。

'name|count': 'value' 通过重复 'value' 生成一个字符串，重复次数等于 count。

2. 属性值是数字 Number

'name|+1': 100 属性值自动加 1，初始值为 100

'name|1-100': 100 生成一个大于等于 1、小于等于 100 的整数，属性值 100 只用来确定类型。

'name|1-100.1-10': 100 生成一个浮点数，整数部分大于等于 1、小于等于 100，小数部分保留 1 到 10 位。

{

'number1|1-100.1-10': 1,

'number2|123.1-10': 1,

'number3|123.3': 1,

'number4|123.10': 1.123

}

// =>

{

"number1": 12.92,

"number2": 123.51,

"number3": 123.777,

"number4": 123.1231091814

}

3. 属性值是布尔型 Boolean

'name|1': value 随机生成一个布尔值，值为 true 的概率是 1/2，值为 false 的概率是 1/2。

'name|min-max': value 随机生成一个布尔值，值为 value 的概率是 min / (min + max)，值为 !value 的概率是 max / (min + max)。

4. 属性值是对象 Object

'name|min-max': {} 从属性值 {} 中随机选取 min 到 max 个属性。

'name|count': {} 从属性值 {} 中随机选取 count 个属性。

5. 属性值是数组 Array

'name|1': [{}, {} ...] 从属性值 [{}, {} ...] 中随机选取 1 个元素，作为最终值。

'name|min-max': [{}, {} ...] 通过重复属性值 [{}, {} ...] 生成一个新数组，重复次数大于等于 min，小于等于 max。

'name|count': [{}, {} ...] 通过重复属性值 [{}, {} ...] 生成一个新数组，重复次数为 count。

6. 属性值是数组 Function

'name': function(){} 执行函数 function(){}，取其返回值作为最终的属性值，上下文为 'name' 所在的对象。

数据占位符定义 DPD

占位符 只是在属性值字符串中占个位置，并不出现在最终的属性值中。占位符 的格式为：

@占位符@占位符(参数 [, 参数])

注意：

用 @ 来标识其后的字符串是 占位符。

占位符 引用的是 Mock.Random 中的方法。

通过 Mock.Random.extend() 来扩展自定义占位符。

占位符 也可以引用 数据模板 中的属性。

占位符 会优先引用 数据模板 中的属性

{

name: {

first: '@FIRST',

middle: '@FIRST',

last: '@LAST',

full: '@first @middle @last'

}

}

// =>

{

"name": {

"first": "Charles",

"middle": "Brenda",

"last": "Lopez",

"full": "Charles Brenda Lopez"

}

}

常用方法

Mock.mock( rurl?, rtype?, template|function(options) )

根据数据模板生成模拟数据。

参数的含义和默认值如下所示：

参数 rurl：可选。表示需要拦截的 URL，可以是 URL 字符串或 URL 正则。例如 /\/domain\/list.json/、'/domian/list.json'。

参数 rtype：可选。表示需要拦截的 Ajax 请求类型。例如 GET、POST、PUT、DELETE 等。

参数 template：可选。表示数据模板，可以是对象或字符串。例如 { 'data|1-10':[{}] }、'@EMAIL'。

参数 function(options)：可选。表示用于生成响应数据的函数。

参数 options：指向本次请求的 Ajax 选项集。

Mock.mockjax(library)

覆盖（拦截） Ajax 请求，目前内置支持 jQuery、Zepto、KISSY。

Mock.Random

Mock.Random 是一个工具类，用于生成各种随机数据。Mock.Random 的方法在数据模板中称为“占位符”，引用格式为 @占位符(参数 [, 参数]) 。

Mock.tpl(input, options, helpers, partials)

基于 Handlebars、Mustache 的 HTML 模板生成模拟数据。

\*stringify方法顾名思义，就是把JSON序列换，其语法如下：

JSON.stringify(value[, replacer [, space]])

Eg01:

<script>

//demo1

var data = Mock.mock({

'list|1-10': [{

'id|+1': 1

}]

});

console.log(JSON.stringify(data,null, 3));

document.writeln(JSON.stringify(data,null, 3));

</script>

Eg02:

<script>

//demo2

Mock.mock('http://g.cn', {

'name' : '@name',

'age|1-100': 100,

'color' : '@color'

});

$.ajax({

url: 'http://g.cn',

dataType:'json'

}).done(function(data, status, xhr){

document.writeln(JSON.stringify(data, null, 4));

});

</script>

Eg03:

<script>

//demo3

/\*Mock.mock('http://a',{

'number1|1-100.1-10': 1,

'number2|123.1-10': 1,

'number3|123.3': 1,

'number4|123.10': 1.123

})

$.ajax({

url: 'http://a',//url:自己定义，与上面的一致即可

dataType:'json'

}).done(function(data, status, xhr){

document.writeln(JSON.stringify(data, null, 4));

});\*/

/\*Mock.mock('http://a',{

'list|1-10': [{'id|+2': 1 , 'age|20-30': 100}],

'name': '@name',

'color': '@color',

'url': '@url',

'email': '@email',

'friends|3': [{name: '@name'}],

'price|10-20.2-5': 11,

'cost|10-20.3': 11,

'test|3.2-5': 52,

'yourname|2-4': 'alice-'

})

$.ajax({

url:'http://a',

dataType:'json'

}).done(function(data,status,xhr){

document.writeln(JSON.stringify(data,null,4));

console.log(data);

})\*/

</script>