# Mock是什么？

[Mock.js](http://mockjs.com) 是一款前端开发中拦截Ajax请求再生成随机数据响应的工具.可以用来模拟服务器响应. 优点是非常简单方便, 无侵入性, 基本覆盖常用的接口数据类型。

# Mock 的应用场景

1. 真实对象具有不可确定的行为（产生不可预测的结果，如股票行情）。
2. 真实对象很难被创建。
3. 真实对象的某些行为很难触发（如网络错误）。
4. 真实对象令程序的运行速度很慢。
5. 等等……

# Mock Quick Start

## mock.js 安装

命令：npm install bower –g 全局安装bower（前端的模块安装工具）

命令：bower install mockjs –save 安装mockjs

或

使用npm安装: npm install mockjs;   
或直接<script src="http://mockjs.com/dist/mock.js"></script>

## 引入 Mock.js

<script src="./bower\_components/mockjs/dist/mock.js"></script>

## 调用 Mock.mock( template ) 生成模拟数据

var data = Mock.mock({

'list|1-10': [{

'id|+1': 1

}]

})

console.log(

JSON.stringify(data, null, 4)

)

**生成的模拟数据**：

{   
"list": [   
{ "id": 1 },   
{ "id": 2 },   
{ "id": 3 },   
{ "id": 4 }   
]   
}

## 调用 Mock.mock( url,template ) 拦截Ajax

Mock.mock('hello.json', {

'list|1-10': [{

'id|+1': 1

}]

})

**发起Ajax请求**：

$.ajax({

url: 'hello.json',

dataType: 'json'

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

console.log(

JSON.stringify(data, null, 4)

)

})

**响应的模拟数据**：

{

"list": [

{ "id": 1 },

{ "id": 2 },

{ "id": 3 },

{ "id": 4 },

{ "id": 5 }

]

}

## 数据模板

每个属性由3部分构成：

//属性名 name

//生成规则 rule

//属性值 value

‘name|rule’ : value

### Mock.mock(template)

1. String：’name|min-max’ : ’value’
2. Number：’name|min-max.dmin-dmax’ : ’value’
3. Boolean：’name|min-max’ : ’value’
4. Object：’name|min-max’ : {}
5. Array：’name|min-max’ : [{},{} …]

### 用于响应数据的函数

1. Mock.mock(rurl,template) 记录数据模板。当拦截到匹配 rurl 的 Ajax 请求时，将根据数据模板 template 生成模拟数据，并作为响应数据返回。
2. Mock.mock(rurl,function(options)) 记录用于生成响应数据的函数。当拦截到匹配 rurl 的 Ajax 请求时，函数 function(options) 将被执行，并把执行结果作为响应数据返回。
3. Mock.mock(rurl,rtype,template) 记录数据模板。当拦截到匹配 rurl 和 rtype 的 Ajax 请求时，将根据数据模板 template 生成模拟数据，并作为响应数据返回。
4. Mock.mock(rurl,rtype,function(options)) 记录用于生成响应数据的函数。当拦截到匹配 rurl 和 rtype 的 Ajax 请求时，函数 function(options) 将被执行，并把执行结果作为响应数据返回。
5. 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'。
6. rtype可选，表示需要拦截的 Ajax 请求类型。例如 GET、POST、PUT、DELETE 等。
7. template 可选，表示数据模板，可以是对象或字符串。例如 { 'data|1-10':[{}] }、'@EMAIL'。
8. function(options)可选，表示用于生成响应数据的函数。options指向本次请求的 Ajax 选项集，含有 url、type 和 body 三个属性

## Mock.setup(settings)

配置拦截 Ajax 请求时的行为。支持的配置项有：timeout。

settings 必选，配置项集合。

timeout 可选。指定被拦截的 Ajax 请求的响应时间，单位是毫秒。值可以是正整数，例如 400，表示 400 毫秒 后才会返回响应内容；也可以是横杠 '-' 风格的字符串，例如 '200-600'，表示响应时间介于 200 和 600 毫秒之间。默认值是'10-100'。

Mock.setup({

timeout: 400

})

## 数据占位符(Mock.Random)

格式：@占位符 或 @占位符(参数[,参数])

与Mock.Random中的方法一一对应

Mock.Random 是一个工具类，用于生成各种随机数据。

Mock.Random的10种随机数据：

### Basic

1. Random.boolean()

Random.boolean( min, max, current )

随机的布尔值

1. Random.natural()

Random.natural( min )

Random.natural( min, max )

随机的自然数（大于等于 0 的整数）

1. Random.integer()

Random.integer( min )

Random.integer( min, max )

随机的整数

1. Random.float()

Random.float( min )

Random.float( min, max )

Random.float( min, max, dmin )

Random.float( min, max, dmin, dmax )

随机的浮点数

1. Random.character()

Random.character( 'lower/upper/number/symbol' )

Random.character( pool )

随机字符

1. Random.string()

Random.string( length )

Random.string( pool, length )

Random.string( min, max )

Random.string( pool, min, max )

随机字符串

1. Random.range( stop )

Random.range( start, stop )

Random.range( start, stop, step )

整型数组

### Date

1. Random.date()

Random.date(format)

随机的日期字符串

1. Random.time()

Random.time( format )

随机的时间字符串

1. Random.datetime()

Random.datetime( format )

随机的日期和时间字符串

1. Ranndom.now( unit, format )

Ranndom.now()

Ranndom.now( format )

Ranndom.now( unit )

当前的日期和时间字符串

### Image

1. Random.image()

Random.image( size )

Random.image( size, background )

Random.image( size, background, text )

Random.image( size, background, foreground, text )

Random.image( size, background, foreground, format, text )

随机的图片地址

1. Random.dataImage()

Random.dataImage( size )

Random.dataImage( size, text )

随机的Base64图片编码

### Color

1. Random.color()

Random.hex()

Random.rgb()

Random.rgba()

Random.hsl()

随机生成一个有吸引力的颜色

### Text

1. Random.paragraph()

Random.paragraph( len )

Random.paragraph( min, max )

随机生成的一段文本

1. Random.cparagraph()

Random.cparagraph( len )

Random.cparagraph( min, max )

随机生成一段中文文本

1. Random.sentence()

Random.sentence( len )

Random.sentence( min, max )

随机生成一个句子，第一个单词的首字母大写

1. Random.csentence()

Random.csentence( len )

Random.csentence( min, max )

随机生成一段中文文本

1. Random.word()

Random.word( len )

Random.word( min, max )

随机生成一个单词

1. Random.cword()

Random.cword( pool )

Random.cword( length )

Random.cword( pool, length )

Random.cword( min, max )

Random.cword( pool, min, max )

随机生成一个汉字

1. Random.title()

Random.title( len )

Random.title( min, max )

随机生成一句标题，其中每个单词的首字母大写

1. Random.ctitle()

Random.ctitle( len )

Random.ctitle( min, max )

随机生成一句中文标题

### Name

1. Random.first()

随机生成一个常见的英文名

1. Random.last()

随机生成一个常见的英文姓

1. Random.name()

Random.name( middle )

随机生成一个常见的英文姓名

1. Random.cfirst()

随机生成一个常见的中文名

1. Random.clast()

随机生成一个常见的中文姓

1. Random.cname()

随机生成一个常见的中文姓名

### Web

1. Random.url()

Random.url( protocol, host )

随机生成一个 URL

1. Random.protocol()

随机生成一个 URL 协议

1. Random.domain()

随机生成一个域名

1. Random.tld()

随机生成一个顶级域名

1. Random.email()

Random.email( domain )

随机生成一个邮件地址

1. Random.ip()

随机生成一个 IP 地址

### Address

1. Random.region()

随机生成一个（中国）大区

1. Random.province()

随机生成一个（中国）省（或直辖市、自治区、特别行政区）

1. Random.city()

Random.city( prefix )

随机生成一个（中国）市

1. Random.county()

Random.county( prefix )

随机生成一个（中国）县

1. Random.zip()

随机生成一个邮政编码（六位数字）

### Helper

1. Random.capitalize(word)

把字符串的第一个字母转换为大写

1. Random.upper( str )

把字符串转换为大写

1. Random.lower( str )

把字符串转换为小写

1. Random.pick( arr )

从数组中随机选取一个元素，并返回

1. Random.shuffle( arr )

打乱数组中元素的顺序，并返回

### Miscellaneous

1. Random.guid()

随机生成一个 GUID

1. Random.id()

随机生成一个 18 位身份证

1. Random.increment()

Random.increment( step )

生成一个全局的自增整数

以上介绍可在<https://github.com/nuysoft/Mock/wiki> 网址上详细查看。

## Mock.valid(template,data)

Mock.valid( template, data ) 校验真实数据 data 是否与数据模板 template 匹配。

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

data 必选。表示真实数据。

例子：

var template = {

name: 'value1'

}

var data = {

name: 'value2'

}

console.log(JSON.stringify(Mock.valid(template, data),null,4))

打印：

[

{

"path": [

"ROOT",

"name"

],

"type": "value",

"actual": "value2",

"expected": "value1",

"action": "equal to",

"message": "[VALUE] Expect ROOT.name'value is equal to value1, but is value2"

}

]

## Mock.toJSONSchema( template )

把 Mock.js 风格的数据模板 template 转换成 [JSON Schema](http://json-schema.org/)。

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

例子：

var template = {

'key|1-10': '★'

}

console.log(JSON.stringify(Mock.toJSONSchema(template),null,4));

效果：

