Vue.js 混入 Vue.js 响应接口

Vue.js Ajax(vue-resource)

Vue 要实现异步加载需要使用到 vue-resource 库。

```
<script src="https://cdn.staticfile.org/vue-resource/1.5.1/vue-resource.min.js"></script>
```

Get 请求

以下是一个简单的 Get 请求实例，请求地址是一个简单的 txt 文本：

实例

```
window.onload = function(){
var vm = new Vue({
el:'#box',
data:{
msg:'Hello World!',
},
methods:{
get:function(){
//发送get请求
this.$http.get('/try/ajax/ajax_info.txt').then(function(res){
document.write(res.body);
},function(){
console.log('请求失败处理');
});
}
}
});
}
```

尝试一下 »

如果需要传递数据，可以使用 `this.$http.get('get.php',{params : jsonData})` 格式，第二个参数 **jsonData** 就是传到后端的数据。

```
this.$http.get('get.php',{params : {a:1,b:2}}).then(function(res){
    document.write(res.body);
},function(res){
    console.log(res.status);
});
```

post 请求

post 发送数据到后端，需要第三个参数 `{emulateJSON:true}`。

emulateJSON 的作用： 如果Web服务器无法处理编码为 application/json 的请求，你可以启用 emulateJSON 选项。

实例

```
window.onload = function(){
var vm = new Vue({
el:'#box',
data:{
msg:'Hello World!',
},
methods:{
post:function(){
//发送 post 请求
this.$http.post('/try/ajax/demo_test_post.php',{name:"菜鸟教程",url:"http://www.runoob.com"},{em
ulateJSON:true}).then(function(res){
document.write(res.body);
},function(res){
console.log(res.status);
});
}
}
});
}
```

尝试一下 »

demo_test_post.php 代码如下：

```
<?php
$name = isset($_POST['name']) ? htmlspecialchars($_POST['name']) : '';
$city = isset($_POST['url']) ? htmlspecialchars($_POST['url']) : '';
echo '网站名: ' . $name;
echo "\n";
echo 'URL 地址: ' .$city;
?>
```

语法 & API

你可以使用全局对象方式 Vue.http 或者在一个 Vue 实例的内部使用 this.$http来发起 HTTP 请求。

```
// 基于全局Vue对象使用http
Vue.http.get('/someUrl', [options]).then(successCallback, errorCallback);
Vue.http.post('/someUrl', [body], [options]).then(successCallback, errorCallback);

// 在一个Vue实例内使用$http
this.$http.get('/someUrl', [options]).then(successCallback, errorCallback);
this.$http.post('/someUrl', [body], [options]).then(successCallback, errorCallback);
```

vue-resource 提供了 7 种请求 API(REST 风格)：

```
get(url, [options])
head(url, [options])
delete(url, [options])
jsonp(url, [options])
post(url, [body], [options])
put(url, [body], [options])
patch(url, [body], [options])
```

除了 jsonp 以外，另外 6 种的 API 名称是标准的 HTTP 方法。

options 参数说明:

参数	类型	描述
url	`string`	请求的目标URL
body	`Object`, `FormData`, `string`	作为请求体发送的数据
headers	`Object`	作为请求头部发送的头部对象
params	`Object`	作为URL参数的参数对象
method	`string`	HTTP方法 (例如GET，POST，...)
timeout	`number`	请求超时（单位：毫秒）(`0`表示永不超时)
before	`function(request)`	在请求发送之前修改请求的回调函数
progress	`function(event)`	用于处理上传进度的回调函数 ProgressEvent
credentials	`boolean`	是否需要出示用于跨站点请求的凭据
emulateHTTP	`boolean`	是否需要通过设置`X-HTTP-Method-Override`头部并且以传统POST方式发送PUT，PATCH和DELETE请求。
emulateJSON	`boolean`	设置请求体的类型为`application/x-www-form-urlencoded`

通过如下属性和方法处理一个请求获取到的响应对象：

属性	类型	描述
url	`string`	响应的 URL 源
body	`Object`, `Blob`, `string`	响应体数据
headers	`Header`	请求头部对象

属性	类型	描述
ok	`boolean`	当 HTTP 响应码为 200 到 299 之间的数值时该值为 true
status	`number`	HTTP 响应码
statusText	`string`	HTTP 响应状态
方法	**类型**	**描述**
text()	约定值	以字符串方式返回响应体
json()	约定值	以格式化后的 json 对象方式返回响应体
blob()	约定值	以二进制 Blob 对象方式返回响应体

← Vue.js 混入

Vue.js 响应接口 →