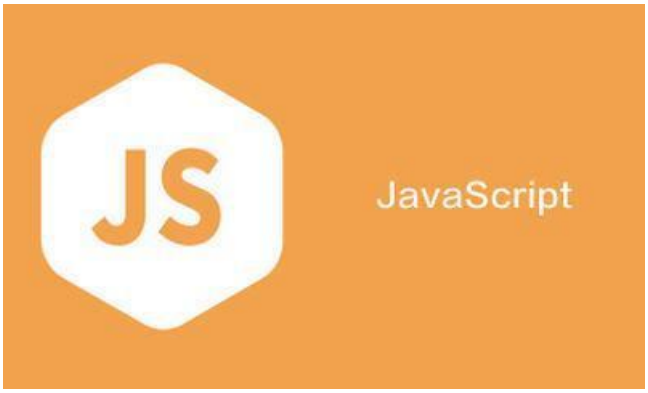
小兔JS教程（五） 简单易懂的JSON入门



上一节的参考答案：

http://xiaotublog.com/demo.html?path=homework/04/index2

本节重点来介绍一下JSON，JSON(JavaScript Object Notation) 是一种轻量级的数据交换格式，我们称之为JavaScript对象表示法。也就是说，JSON是一种格式。首先搞清楚三个概念，即什么是JSON字符串，什么是JavaScript对象，还有什么又叫做JSON对象？先来说一个事，在没有JSON之前，前台页面和Java等语言充当的服务器层，到底是如何传输数据的呢？没错，是通过XML来传输的。比如一个登陆页面。

页面上有用户名和密码两个输入框，当我点击登录按钮，这两个数据就会被传递到服务器层。那么，如何传输呢？如果用XML，也许是这样的：

后台接收到这个数据，然后就可以开始解析，最终拿到zhangsan和123两个字面量。时间线再往前推，在XML还没有出来的时候，怎么办呢？聪明的程序开发人员则会规定几种特殊的格式，拼接一个特殊的字符串，传递到后台中去。比如像这样的："name=zhangsan&password=123"
那么后台的程序员也知道这个规则，如果是Java的话，就可以使用String的splite方法，先通过逗号把这个字符串分割成两份，也就是变成：name=zhangsan还有password=123两个字符串，然后再通过“=”分割，将“name=zhangsan”分割成“name”和“zhangsan”，把“password=123”分割成“password”和“123”两部分。终于，到底还是拿到用户名和密码了。

接下来还是谈JSON，其实JSON就是一种数据格式。诸如：

```
{
    key1 : value1 ,
    key2 : value2
};
```

这样的格式就是JSON格式，它是一系列键值对的集合，不同的键值对之间用逗号分隔，最后一个键值对不需要加逗号。符合这种格式的字符串就是JSON字符串。比如：

```
"{'name' : 'Jack'}"
```

它归根到底还是一个字符串，不是一个对象。而JSON对象，其实就是Javascript对象，我们可以通过字面值的方式直接创造一

个对象，比如：

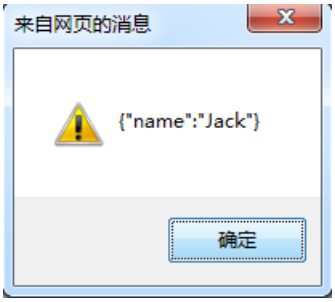
```
var person = {name : 'Jack'}
```

等同于：

```
var person = {'name' : 'Jack'}
```

在上边这个例子中，name可加单引号，也可加双引号，甚至可以什么都不加。而右边的值必须是一个实实在在的东西，比如字符串，或者一个对象，甚至是一个函数。我们不考虑JS内部的对象机制，只是简单地说明一下，是有这么个事情的。这就是所谓的JSON对象，也就是js对象。在JavaScript中，对象是键值对的集合，符合JSON格式。我们可以通过下面的方法，把JS对象转换成JSON格式的字符串。

```
var person = {'name' : 'Jack'}

alert(JSON.stringify(person));
```



同样，一个JSON格式的字符串，可以变成一个JS对象，如：

```
console.log(JSON.parse("{\"name\":\"Jack\"}"));
```

做个小结，JSON字符串就是符合JSON格式的字符串，他还是字符串，JSON对象就是JavaScript对象，我们推荐使用字面值的方式来创建一个JS对象。然后，JS对象和JSON字符串可以互相转换。通过这一个特点，我们能够实现JS对象的拷贝。一般来说，比如我有一个js对象。

```
var person = {'name' : 'Jack'}

var person2 = person;
```

这样做，并不是对象的复制，person2仅仅是一个指针，他和person一样，指向了{'name' :'Jack'}这一片内存空间。当person发生改变，person2必然也跟着改变。

```
var person = {'name' : 'Jack'}

var person2 = person;

person.age = 10; //给person动态地添加一个属性

alert(JSON.stringify(person2)); //person2也跟着变了
```

那有没有什么办法可以实现对象的复制呢？一个好的解决方案就是，先把person转换成JSON字符串，然后再转成JS对象，这个时候就是另外一个JS对象了。比如：

```
var person = {'name' : 'Jack'}

var person2 = JSON.parse(JSON.stringify(person));

person.age = 10; //给person动态地添加一个属性

alert(JSON.stringify(person2)); //person2不变
```

接下来说说js对象内容的访问和操作，我们上面已经说了，JS对象中无非是一些键值对的集合，他更像是一个容器，既然是容器，自然有内容，我们如何访问其中的内容呢？在上面的例子中，我们已经通过“对象.属性名”的方式来访问JS对象的具体内容。比如：

```
var obj = {
     id : 1
};
```

```
var id = obj.id;
alert(id);
```

另外一种方式，就是通过 对象["属性名"] 来操作其内容。比如：

```
var id = obj['id']
```

可以用双引号，也可以用单引号，看个人习惯了。在JS对象中，属性名永远都是字符串，虽然诸如这样的代码：

```
var obj = {
     id : 1
};
```

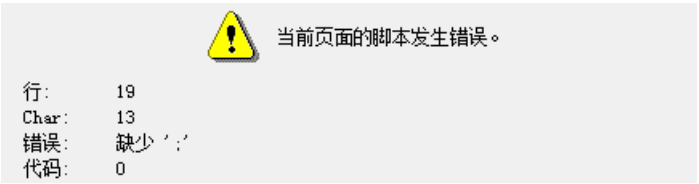
id没有加上引号，但它实际上还是以字符串的形式被保存起来的。再说一遍，如果你要访问和操作JS对象的内容，有两种方式，第一种方式是用点，第二种方式则是用中括号。两种方式如果做一个比较，显然是第二种方式较为灵活，因为它是用字符串去找对应的键值对，而不是用一个标识符。比如刚才的例子，你这样写：

```
var id = obj.id;
```

我问你，obj.id中的id是什么？为了符合规范，id必须是标识符，你不能写 obj.123 吧。这显然是不合法，也无法运行通过的。比如，你能这样写吗？

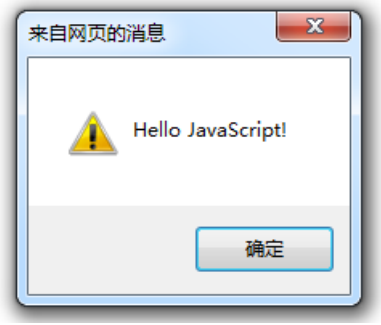
```
var obj = {
     123 : 'Hello JavaScript!'
};

var id = obj.123;
alert(id);
```

肯定不行，会报错的:



但是，如果你用中括号就可以：

```
var obj = {
     123 : 'Hello JavaScript!'
};

var id = obj['123'];
alert(id);
```



具体用那种方式，随你喜好而定。

现在，我们已经对JSON格式和JS对象有了一个比较充分的了解，我要在此抛出一个问题，有没有什么办法能够获取JS对象的属性详情呢？注意我的用词，是属性详情，也就是说，比如有一个JS对象：

```
var obj = {
    message: 'Hello JavaScript!'
};
```

message就是它的属性，关于这个属性，有没有什么详细的描述信息呢？答案是有的，在JS中，有一个内置的Object对象，它给我们提供了一个getOwnPropertyDescriptor方法，可以看到某个对象的某个属性的具体情况。你可以把这个理解为Java中的静态类调用方法。我们可以这样做：

```
var obj = {
    message : 'Hello JavaScript!'
};
```

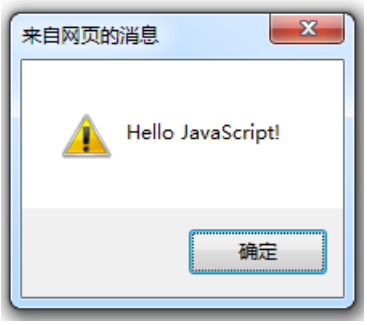
console.log(Object.getOwnPropertyDescriptor(obj,'message'));

可以看到，我们成功挖掘出了四个属性，如果你不明白我在说什么，我就说得更加直白一些，就是说，

```
var obj = {
    message: 'Hello JavaScript!'
};
```

obj里面有一个属性message，而message又有四个描述性的东西，分别是configurable（可配置），enumerable（可枚举），value（值），还有 writable（可写入）。这四样东西，专业术语叫做属性描述符，或者数据描述符。目前我们看到的数据描述符都被赋予了默认值，我们也可以通过defineProperty方法对其进行个性化配置。
比如，我们把message设置为只读：

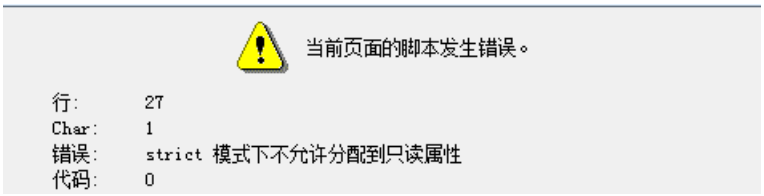
```
var obj = {
    message : 'Hello JavaScript!'
};

console.log(Object.getOwnPropertyDescriptor(obj,'message'));

Object.defineProperty(obj,'message',{
    writable:false
});

obj.message = 'haha';
alert(obj.message);
```



不好意思，修改无效，因为我已经把这个属性设置为只读了。在严格模式下，甚至会报错，啥，你问我什么叫做严格模式？好吧，其实就是一句话的事。

```
"use strict";

var obj = {
    message : 'Hello JavaScript!'
};

console.log(Object.getOwnPropertyDescriptor(obj,'message'));

Object.defineProperty(obj,'message',{
    writable:false
});

obj.message = 'haha';
alert(obj.message);
```

这就是严格模式，你不要问为什么这样就行了，我不会告诉你，因为我也不懂。我只知道，这样写就可以，于是乎，接下来运行就报错了。



本文就介绍到这里，对JSON进行了一个简单的说明。至于深入的学习，还请各位自行去百度吧。 

如果喜欢我的教程，可以关注我的个人博客哦，个人网站保持首更。
网址：www.xiaotublog.com