Json2.js 手册

JavaScript使用了ECMAScript语言规范第三版进行了标准化。

JSON是JavaScript面向对象语法的一个子集。由于JSON是JavaScript的一个子集，因此它可清晰的运用于此语言中。

```
var myJSONObject = {"bindings": [
 {"ircEvent": "PRIVMSG", "method": "newURI", "regex": "^http://.*"},
 {"ircEvent": "PRIVMSG", "method": "deleteURI", "regex": "^delete.*"},
 {"ircEvent": "PRIVMSG", "method": "randomURI", "regex": "^random.*"}
 ]
};
```

上面的示例，创建了一个包括单独成员”bindings”的对象，此成员包括一个含有三个对象（”ircEvent”, “method”, 与 “regex”）的数组

成员可以通过.或者下标操作符检索。

```
myJSONObject.bindings[0].method // "newURI"
```

1. 使用eval()函数：

为了将**JSON**文本转换为对象，可以使用**eval()**函数。**eval()**函数调用**JavaScript**编辑器。由于JSON是JavaScript的子集，因此编译器将正确的解析文本并产生对象结构。文本必须括在括号中避免产生JavaScript的语法歧义。

```
var myObject = eval('(' + myJSONtext + ')');
```

eval函数非常快速。

它可以编译执行任何JavaScript程序，因此产生了安全性问题。

当使用可信任与完善的源代码时才可以使用eval函 数。

这样可以更安全的使用JSON解析器。

使用XMLHttpRequest的web应用，页面之间的通讯只允许同源，因此是可以信任的。但这却不是完善 的。

如果服务器没有严谨的JSON编码，或者没有严格的输入验证，那么可能传送包括危险脚本的无效JSON文本。eval函数将执行恶意的脚本。使用JSON解析器可以防止此类事件。

2. 使用JSON解析器：

2.1 JSON.parse（）；

JSON parse解析器只能辨识JSON文本，拒绝所有脚本。提供了本地JSON支持的浏览器的JSON解析器将远快于eval函数。预计未来的ECMAScript标准将支持本***地JSON***。

使用格式：

```
var myObject = JSON.parse(myJSONtext, reviver);
```

参数：myJSONtext ,要解析的JSON格式字符串

reviver - function可选参数，做为被最终结果的每一级的键（key）与值（value）调用。 每个值都将被替换函数的值代替。这可以用来将一般的类改变成伪类的实例，或者将日期字符串转变为日期对象。

```
myData = JSON.parse(text, function (key, value) {
 var type;
 if (value && typeof value === 'object') {
 type = value.type;
 if (typeof type === 'string' && typeof window[type] === 'function') {
 return new (window[type])(value);
 }
 }
 return value;
});
```

2.2 JSON.stringify();

JSON stringifier进行反向操作，可以把JavaScript数据结构转换为JSON文本。JSON不支持循环数据结构，因此应小心不要为JSON stringifier提供循环结构。

使用格式：
var myJSONText = JSON.stringify(myObject, replacer);
参数：myObject ,要转为字符串的Javascript object 对象。
replacer,如果stringify函数发现一个带有toJSON方法的对象，它将执行此方法，并且返回产生的值。这样一个对象就可以决定自己的JSON表现。
stringifier方法可以携带一个可选的字符串数组。这些字符串被用于选择包括在JSON文本中的属性。
stringifier方法可以携带一个可选的替代(replacer)函数。它将在结构中每个值的toJSON方法（如果有的话）后面执行。它将每个键与值做为参数传递，当然对象要包含这个键。值将被stringified返回。
如果没有提供数组或替代函数，一个用于忽略被集成的属性的可选替代函数将被提供。如果想要所有被继承的属性，可以提供一个简单的替换函数：

```
var myJSONText = JSON.stringify(myObject, function (key, value) {
 return value;
});
```

值在JSON中不代表任何内容，函数与未定义（undefined）被排除在外。
不能确定的数量将被替换为null。为了替代其它的值，可以像下面一样使用替换（replacer）函数

```
function replacer(key, value) {
 if (typeof value === 'number' && !isFinite(value)) {
 return String(value);
 }
 return value;
}
```

开放源代码的JSON解析器与JSON stringifier可以使用。通过minified可以小于2.5K。