**JSON**

# JSON概念

JSON：轻量级数据交换格式,全称:JavaScript Object Notation

* JS对象的子集,可以和js无缝对接

JSON 独立于语言：JSON 使用 Javascript语法来描述数据对象，但是 JSON 仍然独立于语言和平台。JSON 解析器和 JSON 库支持许多不同的编程语言。 目前非常多的动态（PHP，JSP，.NET）编程语言都支持JSON。

* JSON 是存储和交换文本信息的语法。类似 XML。
* JSON 比 XML 更小、更快，更易解析。

## 和XML比较

与 XML 相同之处：

* JSON 是纯文本
* JSON 具有"自我描述性"（人类可读）
* JSON 具有层级结构（值中存在值）
* JSON 可通过 JavaScript 进行解析
* JSON 数据可使用 AJAX 进行传输

与 XML 不同之处：

* 没有结束标签
* 更短
* 读写的速度更快
* 能够使用内建的 JavaScript eval() 方法进行解析
* 使用数组
* 不使用保留字

## 为什么使用json传值

对于 AJAX 应用程序来说，JSON 比 XML 更快更易使用：

使用 XML

* 读取 XML 文档
* 使用 XML DOM 来循环遍历文档
* 读取值并存储在变量中

使用 JSON

* 读取 JSON 字符串
* 用 eval() 处理 JSON 字符串

# JSON语法

## 实例

{"sites":[

{"name":"Runoob", "url":"www.runoob.com"},

{"name":"Google", "url":"www.google.com"},

{"name":"Taobao", "url":"www.taobao.com"}

]}

## 语法规则

### 规则

* 数据为 **键/值** 对。
* 数据由逗号分隔。
* 大括号保存对象
* 方括号保存数组

### JSON 值类型

* 数字（整数或浮点数
* 字符串（在双引号中）
* 逻辑值（true 或 false）
* 数组（在中括号中）
* 对象（在大括号中）
* null

### Json对象

#### 创建对象

{ "name":"runoob", "alexa":10000, "site":null }

* JSON 对象使用在大括号({})中书写。
* 对象可以包含多个 key/value（键/值）对。
* key 必须是字符串，value 可以是合法的 JSON 数据类型（字符串, 数字, 对象, 数组, 布尔值或 null）。
* key 和 value 中使用冒号(:)分割。
* 每个 key/value 对使用逗号(,)分割。

#### 嵌套对象

myObj = {

"name":"runoob",

"alexa":10000,

"sites": {

"site1":"www.runoob.com",

"site2":"m.runoob.com",

"site3":"c.runoob.com"

}

}

#### 访问对象



#### 遍历对象

var myObj = { "name":"runoob", "alexa":10000, "site":null };

for (x in myObj) {

document.getElementById("demo").innerHTML += x + "<br>";

}

# JSON数据和JS对象的转换

JSON 格式在语法上与创建 JavaScript 对象代码是相同的。

由于它们很相似，所以 JavaScript 程序可以很容易的将 JSON 数据转换为 JavaScript 对象。

**转换方法：**

* JSON-->JS:JSON.parse(data)
* JS-->JSON:JSON.stringify(JsonObj);

# 从服务器端获取json数据

## 获取Json

我们可以使用 AJAX 从服务器请求 JSON 数据，并解析为 JavaScript 对象。

var xmlhttp = new XMLHttpRequest();

xmlhttp.onreadystatechange = function() {

if (this.readyState == 4 && this.status == 200) {

myObj = JSON.parse(this.responseText);

document.getElementById("demo").innerHTML = myObj.name;

}

};

xmlhttp.open("GET", "/try/ajax/json\_demo.txt", true);

xmlhttp.send();

## 异常

### 解析Date对象

JSON 不能存储 Date 对象。如果你需要存储 Date 对象，需要将其转换为字符串。之后再将字符串转换为 Date 对象。

方法一：

var text = '{ "name":"Runoob", "initDate":"2013-12-14", "site":"www.runoob.com"}';

var obj = JSON.parse(text);

obj.initDate = new Date(obj.initDate);

document.getElementById("demo").innerHTML = obj.name + "创建日期: " + obj.initDate;

方法二：

我们可以启用 JSON.parse 的第二个参数 reviver，一个转换结果的函数，对象的每个成员调用此函数。

var text = '{ "name":"Runoob", "initDate":"2013-12-14", "site":"www.runoob.com"}';

var obj = JSON.parse(text, function (key, value) {

if (key == "initDate") {

return new Date(value);

} else {

return value;

}});

document.getElementById("demo").innerHTML = obj.name + "创建日期：" + obj.initDate;

### 解析函数

JSON 不允许包含函数，但你可以将函数作为字符串存储，之后再将字符串转换为函数。

var text = '{ "name":"Runoob", "alexa":"function () {return 10000;}", "site":"www.runoob.com"}';

var obj = JSON.parse(text);

obj.alexa = eval("(" + obj.alexa + ")");

document.getElementById("demo").innerHTML = obj.name + " Alexa 排名：" + obj.alexa();

# 给服务器端发送数据

JSON 通常用于与服务端交换数据。

在向服务器发送数据时一般是字符串。

我们可以使用 JSON.stringify() 方法将 JavaScript 对象转换为字符串。

## js对象转字符串

语法

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

参数说明：

* value:

必需， 要转换的 JavaScript 值（通常为对象或数组）。

* replacer:
* 可选。用于转换结果的函数或数组。
* 如果 replacer 为函数，则 JSON.stringify 将调用该函数，并传入每个成员的键和值。使用返回值而不是原始值。如果此函数返回 undefined，则排除成员。根对象的键是一个空字符串：""。
* 如果 replacer 是一个数组，则仅转换该数组中具有键值的成员。成员的转换顺序与键在数组中的顺序一样。当 value 参数也为数组时，将忽略 replacer 数组。
* space:

可选，文本添加缩进、空格和换行符，如果 space 是一个数字，则返回值文本在每个级别缩进指定数目的空格，如果 space 大于 10，则文本缩进 10 个空格。space 也可以使用非数字，如：\t。

## 异常

和从服务器端获取数据同理，需要将Date对象和函数进行转换

Date对象

var obj = { "name":"Runoob", "initDate":new Date(), "site":"www.runoob.com"};

var myJSON = JSON.stringify(obj);

document.getElementById("demo").innerHTML = myJSON;

函数

var obj = { "name":"Runoob", "alexa":function () {return 10000;}, "site":"www.runoob.com"};

obj.alexa = obj.alexa.toString();

var myJSON = JSON.stringify(obj);

document.getElementById("demo").innerHTML = myJSON;

# JSONP

Jsonp(JSON with Padding) 是 json 的一种"使用模式"，可以让网页从别的域名（网站）那获取资料，即跨域读取数据。

为什么我们从不同的域（网站）访问数据需要一个特殊的技术( JSONP )呢？这是因为同源策略。

同源策略，它是由 Netscape 提出的一个著名的安全策略，现在所有支持 JavaScript 的浏览器都会使用这个策略。

实例

jsonp.php

<?php

header('Content-type: application/json');

//获取回调函数名

$jsoncallback = htmlspecialchars($\_REQUEST ['jsoncallback']);

//json数据

$json\_data = '["customername1","customername2"]';

//输出jsonp格式的数据

echo $jsoncallback . "(" . $json\_data . ")";

?>

客户端

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<title>JSONP 实例</title>

</head>

<body>

<div id="divCustomers"></div>

<script type="text/javascript">

function callbackFunction(result, methodName)

{

var html = '<ul>';

for(var i = 0; i < result.length; i++)

{

html += '<li>' + result[i] + '</li>';

}

html += '</ul>';

document.getElementById('divCustomers').innerHTML = html;

}

</script>

<script type="text/javascript" src="https://www.runoob.com/try/ajax/jsonp.php?jsoncallback=callbackFunction"></script>

</body>

</html>