**JSON 教程**

JSON: **J**ava**S**cript **O**bject **N**otation(JavaScript 对象表示法)

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。

**JSON 实例**

{  
"employees": [  
{ "firstName":"John" , "lastName":"Doe" },   
{ "firstName":"Anna" , "lastName":"Smith" },   
{ "firstName":"Peter" , "lastName":"Jones" }  
]  
}

这个 employee 对象是包含 3 个员工记录（对象）的数组。

**什么是 JSON ？**

* JSON 指的是 JavaScript 对象表示法（**J**ava**S**cript **O**bject **N**otation）
* JSON 是轻量级的文本数据交换格式
* JSON 独立于语言 \*
* JSON 具有自我描述性，更易理解

|  |  |
| --- | --- |
| **\*** | 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 - 转换为 JavaScript 对象**

JSON 文本格式在语法上与创建 JavaScript 对象的代码相同。

由于这种相似性，无需解析器，JavaScript 程序能够使用内建的 eval() 函数，用 JSON 数据来生成原生的 JavaScript 对象。

# JSON - 简介

## 在线实例

通过我们的编辑器，您可以在线编辑 JavaScript 代码，然后通过点击一个按钮来查看结果：

## JSON 实例

<!DOCTYPE html>  
<html>  
<body>  
<h2>JSON Object Creation in JavaScript</h2>

<p>  
Name: <span id="jname"></span><br />   
Age: <span id="jage"></span><br />   
Address: <span id="jstreet"></span><br />   
Phone: <span id="jphone"></span><br />   
</p>

<script>  
var JSONObject= {  
"name":"John Johnson",  
"street":"Oslo West 555",   
"age":33,  
"phone":"555 1234567"};  
document.getElementById("jname").innerHTML=JSONObject.name   
document.getElementById("jage").innerHTML=JSONObject.age   
document.getElementById("jstreet").innerHTML=JSONObject.street   
document.getElementById("jphone").innerHTML=JSONObject.phone   
</script>  
  
</body>  
</html>

[**尝试一下 »**](http://www.w3cschool.cc/try/try.php?filename=tryjson_create)   
点击 "尝试一下" 按钮查看在线实例。

## 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？

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**

JSON 语法是 JavaScript 语法的子集。

**JSON 语法规则**

JSON 语法是 JavaScript 对象表示法语法的子集。

* 数据在名称/值对中
* 数据由逗号分隔
* 花括号保存对象
* 方括号保存数组

**JSON 名称/值对**

JSON 数据的书写格式是：名称/值对。

名称/值对包括字段名称（在双引号中），后面写一个冒号，然后是值：

"firstName" : "John"

这很容易理解，等价于这条 JavaScript 语句：

firstName = "John"

**JSON 值**

JSON 值可以是：

* 数字（整数或浮点数）
* 字符串（在双引号中）
* 逻辑值（true 或 false）
* 数组（在方括号中）
* 对象（在花括号中）
* null

**JSON 对象**

JSON 对象在花括号中书写：

对象可以包含多个名称/值对：

{ "firstName":"John" , "lastName":"Doe" }

这一点也容易理解，与这条 JavaScript 语句等价：

firstName = "John"  
lastName = "Doe"

**JSON 数组**

JSON 数组在方括号中书写：

数组可包含多个对象：

{  
"employees": [  
{ "firstName":"John" , "lastName":"Doe" },   
{ "firstName":"Anna" , "lastName":"Smith" },   
{ "firstName":"Peter" , "lastName":"Jones" }  
]  
}

在上面的例子中，对象 "employees" 是包含三个对象的数组。每个对象代表一条关于某人（有姓和名）的记录。

**JSON 使用 JavaScript 语法**

因为 JSON 使用 JavaScript 语法，所以无需额外的软件就能处理 JavaScript 中的 JSON。

通过 JavaScript，您可以创建一个对象数组，并像这样进行赋值：

**实例**

var employees = [  
{ "firstName":"John" , "lastName":"Doe" },   
{ "firstName":"Anna" , "lastName":"Smith" },   
{ "firstName":"Peter" , "lastName": "Jones" }  
];

可以像这样访问 JavaScript 对象数组中的第一项：

employees[0].lastName;

返回的内容是：

Doe

可以像这样修改数据：

employees[0].firstName = "Jonatan";

[**尝试一下 »**](http://www.w3cschool.cc/try/try.php?filename=tryjson_objectarray)

在下面的章节，您将学到如何把 JSON 文本转换为 JavaScript 对象。

**JSON 使用**

**把 JSON 文本转换为 JavaScript 对象**

JSON 最常见的用法之一，是从 web 服务器上读取 JSON 数据（作为文件或作为 HttpRequest），将 JSON 数据转换为 JavaScript 对象，然后在网页中使用该数据。

为了更简单地为您讲解，我们使用字符串作为输入进行演示（而不是文件）。

**JSON 实例 - 来自字符串的对象**

创建包含 JSON 语法的 JavaScript 字符串：

var txt = '{ "employees" : [' +  
'{ "firstName":"John" , "lastName":"Doe" },' +  
'{ "firstName":"Anna" , "lastName":"Smith" },' +  
'{ "firstName":"Peter" , "lastName":"Jones" } ]}';

由于 JSON 语法是 JavaScript 语法的子集，JavaScript 函数 eval() 可用于将 JSON 文本转换为 JavaScript 对象。

eval() 函数使用的是 JavaScript 编译器，可解析 JSON 文本，然后生成 JavaScript 对象。必须把文本包围在括号中，这样才能避免语法错误：

var obj = eval ("(" + txt + ")");

在网页中使用 JavaScript 对象：

**实例**

<p>  
First Name: <span id="fname"></span><br />   
Last Name: <span id="lname"></span><br />   
</p>   
  
<script>  
document.getElementById("fname").innerHTML = obj.employees[1].firstName   
document.getElementById("lname").innerHTML = obj.employees[1].lastName   
</script>

[**尝试一下 »**](http://www.w3cschool.cc/try/try.php?filename=tryjson_eval)

**JSON 解析器**

lamp  eval() 函数可编译并执行任何 JavaScript 代码。这隐藏了一个潜在的安全问题。

使用 JSON 解析器将 JSON 转换为 JavaScript 对象是更安全的做法。JSON 解析器只能识别 JSON 文本，而不会编译脚本。

在浏览器中，这提供了原生的 JSON 支持，而且 JSON 解析器的速度更快。

较新的浏览器和最新的 ECMAScript (JavaScript) 标准中均包含了原生的对 JSON 的支持。

|  |  |
| --- | --- |
| **Web 浏览器支持** | **Web 软件支持** |
| * Firefox (Mozilla) 3.5 * Internet Explorer 8 * Chrome * Opera 10 * Safari 4 | * jQuery * Yahoo UI * Prototype * Dojo * ECMAScript 1.5 |

# JSONP 教程

本章节我们将向大家介绍 JSONP 的知识。

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。

## JSONP 应用

### 1. 服务端JSONP格式数据

如客户想访问 : http://www.w3cschool.cc/try/ajax/jsonp.php?jsonp=callbackFunction。

假设客户期望返回JSON数据：["customername1","customername2"]。

真正返回到客户端的数据显示为: callbackFunction(["customername1","customername2"])。

服务端文件jsonp.php代码为：

<?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 . ")";

?>

### 2. 客户端实现 callbackFunction 函数

<script type="text/javascript">

function onCustomerLoaded(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>

### 页面展示

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

### 客户端页面完整代码

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >

<head>

<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="http://www.w3cschool.cc/try/ajax/jsonp.php?jsoncallback=callbackFunction"></script>

</body>

</html>

## jQuery 使用 JSONP

以上代码可以使用 jQuery 代码实例：

<!DOCTYPE html>

<html>

<head>

<title>JSONP 实例</title>

<script src="http://apps.bdimg.com/libs/jquery/1.8.3/jquery.js"></script>

</head>

<body>

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

<script>

$.getJSON("http://www.w3cschool.cc/try/ajax/jsonp.php?jsoncallback=?", function(data) {

var html = '<ul>';

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

{

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

}

html += '</ul>';

$('#divCustomers').html(html);

});

</script>

</body>

</html>

# PHP JSON

本章节我们将为大家介绍如何使用 PHP 语言来编码和解码 JSON 对象。

## 环境配置

在 php5.2.0 及以上版本已经内置 JSON 扩展。

## JSON 函数

|  |  |
| --- | --- |
| **函数** | **描述** |
| json\_encode | 对变量进行 JSON 编码 |
| json\_decode | 对 JSON 格式的字符串进行解码，转换为 PHP 变量 |
| json\_last\_error | 返回最后发生的错误 |

## json\_encode

PHP json\_encode() 用于对变量进行 JSON 编码，该函数如果执行成功返回 JSON 数据，否则返回 FALSE 。

### 语法

string json\_encode ( $value [, $options = 0 ] )

### 参数

* **value**: 要编码的值。该函数只对 UTF-8 编码的数据有效。
* **options**:由以下常量组成的二进制掩码：JSON\_HEX\_QUOT, JSON\_HEX\_TAG, JSON\_HEX\_AMP, JSON\_HEX\_APOS, JSON\_NUMERIC\_CHECK,JSON\_PRETTY\_PRINT, JSON\_UNESCAPED\_SLASHES, JSON\_FORCE\_OBJECT

### 实例

以下实例演示了如何将 PHP 数组转换为 JSON 格式数据：

<?php

$arr = array('a' => 1, 'b' => 2, 'c' => 3, 'd' => 4, 'e' => 5);

echo json\_encode($arr);

?>

以上代码执行结果为：

{"a":1,"b":2,"c":3,"d":4,"e":5}

以下实例演示了如何将 PHP 对象转换为 JSON 格式数据：

?php

class Emp {

public $name = "";

public $hobbies = "";

public $birthdate = "";

}

$e = new Emp();

$e->name = "sachin";

$e->hobbies = "sports";

$e->birthdate = date('m/d/Y h:i:s a', "8/5/1974 12:20:03 p");

$e->birthdate = date('m/d/Y h:i:s a', strtotime("8/5/1974 12:20:03"));

echo json\_encode($e);

?>

以上代码执行结果为：

{"name":"sachin","hobbies":"sports","birthdate":"08\/05\/1974 12:20:03 pm"}

## json\_decode

PHP json\_decode() 函数用于对 JSON 格式的字符串进行解码，并转换为 PHP 变量。

### 语法

mixed json\_decode ($json [,$assoc = false [, $depth = 512 [, $options = 0 ]]])

### 参数

* **json\_string**: 待解码的 JSON 字符串，必须是 UTF-8 编码数据
* **assoc**: 当该参数为 TRUE 时，将返回数组，FALSE 时返回对象。
* **depth**: 整数类型的参数，它指定递归深度
* **options**: 二进制掩码，目前只支持 JSON\_BIGINT\_AS\_STRING 。

### 实例

以下实例演示了如何解码 JSON 数据：

<?php

$json = '{"a":1,"b":2,"c":3,"d":4,"e":5}';

var\_dump(json\_decode($json));

var\_dump(json\_decode($json, true));

?>

以上代码执行结果为：

object(stdClass)#1 (5) {

["a"] => int(1)

["b"] => int(2)

["c"] => int(3)

["d"] => int(4)

["e"] => int(5)

}

array(5) {

["a"] => int(1)

["b"] => int(2)

["c"] => int(3)

["d"] => int(4)

["e"] => int(5)

}