

JSON 教程

JSON: **JavaScript Object Notation**(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

```
{
  "sites": [
    { "name": "菜鸟教程" , "url": "www.runoob.com" },
    { "name": "google" , "url": "www.google.com" },
    { "name": "微博" , "url": "www.weibo.com" }
  ]
}
```

这个 sites 对象是包含 3 个站点记录（对象）的数组。

什么是 JSON ？

- JSON 指的是 JavaScript 对象表示法（ **JavaScript Object Notation** ）
- JSON 是轻量级的文本数据交换格式
- 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 具有自我描述性，更易理解

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。

 点我分享笔记

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>
<head>
<meta charset="utf-8">
<title>菜鸟教程(runoob.com)</title>
</head>
<body>
<h2>JavaScript 创建 JSON 对象</h2>
<p>
网站名称: <span id="jname"></span><br />
网站地址: <span id="jurl"></span><br />
网站 slogan: <span id="jslogan"></span><br />
</p>
<script>
var JSONObject= {
"name": "菜鸟教程",
"url": "www.runoob.com",
"slogan": "学的不仅是技术，更是梦想！"
};
document.getElementById("jname").innerHTML=JSONObject.name
document.getElementById("jurl").innerHTML=JSONObject.url
document.getElementById("jslogan").innerHTML=JSONObject.slogan
</script>
</body>
</html>
```

[尝试一下 »](#)

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](#)[JSONP 教程 →](#)[📝 点我分享笔记](#)

JSON 语法

JSON 语法是 JavaScript 语法的子集。

JSON 语法规则

JSON 语法是 JavaScript 对象表示语法的子集。

- 数据在名称/值对中
- 数据由逗号分隔
- 大括号保存对象
- 中括号保存数组

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：

```
"name" : "菜鸟教程"
```

这很容易理解，等价于这条 JavaScript 语句：

```
name = "菜鸟教程"
```

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（在中括号中）
- 对象（在大括号中）
- null

JSON 数字

JSON 数字可以是整型或者浮点型：

```
{ "age": 30 }
```

JSON 对象

JSON 对象在大括号 ({}) 中书写：

对象可以包含多个名称/值对：

```
{ "name": "菜鸟教程" , "url": "www.runoob.com" }
```

这一点也容易理解，与这条 JavaScript 语句等价：

```
name = "菜鸟教程"  
url = "www.runoob.com"
```

JSON 数组

JSON 数组在中括号中书写：

数组可包含多个对象：

```
{  
  "sites": [  
    { "name": "菜鸟教程" , "url": "www.runoob.com" },  
    { "name": "google" , "url": "www.google.com" },  
    { "name": "微博" , "url": "www.weibo.com" }  
  ]  
}
```

在上面的例子中，对象 "sites" 是包含三个对象的数组。每个对象代表一条关于某个网站 (name、url) 的记录。

JSON 布尔值

JSON 布尔值可以是 true 或者 false：

```
{ "flag": true }
```

JSON null

JSON 可以设置 null 值：

```
{ "runoob": null }
```

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 sites = [  
  { "name": "runoob" , "url": "www.runoob.com" },  
  { "name": "google" , "url": "www.google.com" },  
]
```

```
{ "name": "微博" , "url": "www.weibo.com" }  
];
```

可以像这样访问 JavaScript 对象数组中的第一项（索引从 0 开始）：

```
sites[0].name;
```

返回的内容是：

```
runoob
```

可以像这样修改数据：

```
sites[0].name="菜鸟教程";
```

[尝试一下 »](#)

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

JSON 文件

- JSON 文件的文件类型是 ".json"
- JSON 文本的 MIME 类型是 "application/json"

[← JSON 使用](#)

[JSON 简介 →](#)

[✎ 点我分享笔记](#)

JSON 对象

对象语法

实例

```
{ "name":"runoob", "alexas":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。

访问对象值

你可以使用点号 (.) 来访问对象的值：

实例

```
var myObj, x;  
myObj = { "name":"runoob", "alexas":10000, "site":null };  
x = myObj.name;
```

[尝试一下 »](#)

你也可以使用中括号 ([]) 来访问对象的值：

实例

```
var myObj, x;  
myObj = { "name":"runoob", "alexas":10000, "site":null };  
x = myObj["name"];
```

[尝试一下 »](#)

循环对象

你可以使用 for-in 来循环对象的属性：

实例

```
var myObj = { "name":"runoob", "alexas":10000, "site":null };  
for (x in myObj) {  
  document.getElementById("demo").innerHTML += x + "<br>";  
}
```

[尝试一下 »](#)

在 for-in 循环对象的属性时，使用中括号 ([]) 来访问属性的值：

实例

```
var myObj = { "name":"runoob", "alexa":10000, "site":null };
for (x in myObj) {
  document.getElementById("demo").innerHTML += myObj[x] + "<br>";
}
```

[尝试一下 »](#)

嵌套 JSON 对象

JSON 对象中可以包含另外一个 JSON 对象：

实例

```
myObj = {
  "name":"runoob",
  "alexa":10000,
  "sites": {
    "site1":"www.runoob.com",
    "site2":"m.runoob.com",
    "site3":"c.runoob.com"
  }
}
```

你可以使用点号(.)或者中括号([])来访问嵌套的 JSON 对象。

实例

```
x = myObj.sites.site1;
// 或者
x = myObj.sites["site1"];
```

[尝试一下 »](#)

修改值

你可以使用点号(.)来修改 JSON 对象的值：

实例

```
myObj.sites.site1 = "www.google.com";
```

[尝试一下 »](#)

你可以使用中括号([])来修改 JSON 对象的值：

实例


```
myObj.sites["site1"] = "www.google.com";
```

[尝试一下 »](#)

删除对象属性

我们可以使用 **delete** 关键字来删除 JSON 对象的属性：

实例

```
delete myObj.sites.site1;
```

[尝试一下 »](#)

你可以使用中括号([])来删除 JSON 对象的属性：

实例

```
delete myObj.sites["site1"]
```

[尝试一下 »](#)[← JSONP 教程](#)[JSON 数组 →](#)**3 篇笔记** **写笔记**

JSON 数组

数组作为 JSON 对象

实例

```
[ "Google", "Runoob", "Taobao" ]
```

JSON 数组在中括号中书写。

JSON 中数组值必须是合法的 JSON 数据类型（字符串, 数字, 对象, 数组, 布尔值或 null）。

JavaScript 中，数组值可以是以上的 JSON 数据类型，也可以是 JavaScript 的表达式，包括函数，日期，及 *undefined*。

JSON 对象中的数组

对象属性的值可以是一个数组：

实例

```
{  
  "name": "网站",  
  "num": 3,  
  "sites": [ "Google", "Runoob", "Taobao" ]  
}
```

我们可以使用索引值来访问数组：

实例

```
x = myObj.sites[0];
```

[尝试一下 »](#)

循环数组

你可以使用 for-in 来访问数组：

实例

```
for (i in myObj.sites) {  
  x += myObj.sites[i] + "<br>";  
}
```

[尝试一下 »](#)

你也可以使用 for 循环：

实例

```
for (i = 0; i < myObj.sites.length; i++) {  
  x += myObj.sites[i] + "<br>";  
}
```

[尝试一下 »](#)

嵌套 JSON 对象中的数组

JSON 对象中数组可以包含另外一个数组，或者另外一个 JSON 对象：

实例

```
myObj = {  
  "name": "网站",  
  "num": 3,  
  "sites": [  
    { "name": "Google", "info": [ "Android", "Google 搜索", "Google 翻译" ] },  
    { "name": "Runoob", "info": [ "菜鸟教程", "菜鸟工具", "菜鸟微信" ] },  
    { "name": "Taobao", "info": [ "淘宝", "网购" ] }  
  ]  
}
```

我们可以使用 for-in 来循环访问每个数组：

实例

```
for (i in myObj.sites) {  
  x += "<h1>" + myObj.sites[i].name + "</h1>";  
  for (j in myObj.sites[i].info) {  
    x += myObj.sites[i].info[j] + "<br>";  
  }  
}
```

[尝试一下 »](#)

修改数组值

你可以使用索引值来修改数组值：

实例

```
myObj.sites[1] = "Github";
```

[尝试一下 »](#)

删除数组元素

我们可以使用 **delete** 关键字来删除数组元素：

实例

```
delete myObj.sites[1];
```

[尝试一下 »](#)[← JSON 对象](#)[JSON.parse\(\) →](#)

2 篇笔记

[写笔记](#)

json数据格式：主要由对象 { } 和数组 [] 组成:

其中对象包括键值对（属性:属性值）{key : value}，value 可为 str，num，list，obj。取值使用 objcet.key

{key: value, key2:value2，} 键：值用冒号分开，对间用，连接

数组包含元素：num，str，list，objcet 都可以，利用索引访问 [index]，用 . 连接各个值:

e.g :

```
var stu = {"student":          //stu 对象包含student的key,值为一个数组
[                               //数组的每一个值为一个具体的学生对象
{"name": "Tom","Grade":1, "age":11, "gender": "M"},      //学生对象的键为名字,值为对应属性
{"name": "Jerry", "Grade":1, "age":10, "gender": "M"}      //每个属性对应的是一个key,value
],
"classroom": {"class1": "room1", "class2": "room2"}      //对象的值,嵌套对象
};
```

读取数据:

```
document.write(stu.student[1].name);    // 输出第二个学生名
document.write(stu.student[0].age);      // 输出第一个学生年龄
document.write(stu.classroom.class1);    // 输出 classroom 的 class1 值
document.write(stu["classroom"].class2); // 也可用中括号键访问对象值
```

[尝试一下 »](#)

Tom 1年前 (2017-12-01)



删除数值的元素，数组的大小不变，代码如下：

```
var myObj, i, x = "";
myObj = {
  "name": "网站",
  "num": 3,
  "sites": [ "Google", "Runoob", "Taobao" ]
};
delete myObj.sites[1];

x = "sites.length = " + myObj.sites.length + "<br><br>";
```

```
for (i in myObj.sites) {  
    x += i + " " + myObj.sites[i] + "<br>";  
}  
  
document.getElementById("demo").innerHTML = x;
```

[尝试一下 »](#)

delete 运算符并不是彻底删除元素，而是删除它的值，但仍会保留空间。

运算符 delete 只是将该值置为 undefined，而不会影响数组长度，即将其变为稀疏数组（《JS权威指南》7.5节）。

更多内容可参考：[JS 中彻底删除 JSON 对象组成的数组中的元素](#)

喔喔和奶糖 4个月前 (11-05)

JSON.parse()

JSON 通常用于与服务端交换数据。

在接收服务器数据时一般是字符串。

我们可以使用 JSON.parse() 方法将数据转换为 JavaScript 对象。

语法

```
JSON.parse(text[, reviver])
```

参数说明：

- **text**:必需，一个有效的 JSON 字符串。
- **reviver**: 可选，一个转换结果的函数，将为对象的每个成员调用此函数。

JSON 解析实例

例如我们从服务器接收了以下数据：

```
{ "name":"runoob", "alexa":10000, "site":"www.runoob.com" }
```

我们使用 JSON.parse() 方法处理以上数据，将其转换为 JavaScript 对象：

```
var obj = JSON.parse('{ "name":"runoob", "alexa":10000, "site":"www.runoob.com" }');
```

解析前要确保你的数据是标准的 JSON 格式，否则会解析出错。

你可以使用我们的在线工具检测：<https://c.runoob.com/front-end/53>。

解析完成后，我们就可以在网页上使用 JSON 数据了：

实例

```
<p id="demo"></p>
<script>
var obj = JSON.parse('{ "name":"runoob", "alexa":10000, "site":"www.runoob.com" }');
document.getElementById("demo").innerHTML = obj.name + ": " + obj.site;
</script>
```

尝试一下 »

从服务端接收 JSON 数据

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

实例

```
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();
```

[尝试一下 »](#)

查看服务端数据：[json_demo.txt](#)

从服务端接收数组的 JSON 数据

如果从服务端接收的是数组的 JSON 数据，则 JSON.parse 会将其转换为 JavaScript 数组：

实例

```
var xmlhttp = new XMLHttpRequest();
xmlhttp.onreadystatechange = function() {
  if (this.readyState == 4 && this.status == 200) {
    myArr = JSON.parse(this.responseText);
    document.getElementById("demo").innerHTML = myArr[1];
  }
};
xmlhttp.open("GET", "/try/ajax/json_demo_array.txt", true);
xmlhttp.send();
```

[尝试一下 »](#)

查看服务端数据：[json_demo_array.txt](#)

异常

解析数据

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。

实例

```
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;
```

[尝试一下 »](#)

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。

实例

```
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;
```

[尝试一下 »](#)

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。

实例

```
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();
```

[尝试一下 »](#)

不建议在 JSON 中使用函数。

浏览器支持

主流浏览器都支持 JSON.parse() 函数：

- Firefox 3.5
- Internet Explorer 8
- Chrome
- Opera 10
- Safari 4

[← JSON 数组](#)[JSON.stringify\(\) →](#)**4 篇笔记****写笔记**

[← JSON.parse\(\)](#)

JSON.stringify()

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。

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。

JavaScript 对象转换

例如我们向服务器发送以下数据：

```
var obj = { "name": "runoob", "alexa": 10000, "site": "www.runoob.com" };
```

我们使用 JSON.stringify() 方法处理以上数据，将其转换为字符串：

```
var myJSON = JSON.stringify(obj);
```

myJSON 为字符串。

我们可以将 myJSON 发送到服务器：

实例

```
var obj = { "name": "runoob", "alexa": 10000, "site": "www.runoob.com" };  
var myJSON = JSON.stringify(obj);  
document.getElementById("demo").innerHTML = myJSON;
```

[尝试一下 »](#)

JavaScript 数组转换

我们也可以将 JavaScript 数组转换为 JSON 字符串：

实例

```
var arr = [ "Google", "Runoob", "Taobao", "Facebook" ];  
var myJSON = JSON.stringify(arr);
```

myJSON 为字符串。

我们可以将 myJSON 发送到服务器：

实例

```
var arr = [ "Google", "Runoob", "Taobao", "Facebook" ];  
var myJSON = JSON.stringify(arr);  
document.getElementById("demo").innerHTML = myJSON;
```

[尝试一下 »](#)

异常

解析数据

JSON 不能存储 Date 对象。

JSON.stringify() 会将所有日期转换为字符串。

实例

```
var obj = { "name":"Runoob", "initDate":new Date(), "site":"www.runoob.com"};  
var myJSON = JSON.stringify(obj);  
document.getElementById("demo").innerHTML = myJSON;
```

[尝试一下 »](#)

之后你可以再将字符串转换为 Date 对象。

解析函数

JSON 不允许包含函数，JSON.stringify() 会删除 JavaScript 对象的函数，包括 key 和 value。

实例

```
var obj = { "name":"Runoob", "alexa":function () {return 10000;}, "site":"www.runoob.com"};  
var myJSON = JSON.stringify(obj);  
document.getElementById("demo").innerHTML = myJSON;
```

[尝试一下 »](#)

我们可以在执行 JSON.stringify() 函数前将函数转换为字符串来避免以上问题的发生：

实例

```
var obj = { "name": "Runoob", "alex": function () {return 10000;}, "site": "www.runoob.com"};
obj.alex = obj.alex.toString();
var myJSON = JSON.stringify(obj);
document.getElementById("demo").innerHTML = myJSON;
```

尝试一下 »

不建议在 JSON 中使用函数。

浏览器支持

主流浏览器都支持 JSON.stringify() 函数：

- Firefox 3.5
- Internet Explorer 8
- Chrome
- Opera 10
- Safari 4

← JSON.parse()

 点我分享笔记

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 = '{ "sites" : [' +  
'{ "name":"菜鸟教程" , "url":"www.runoob.com" },' +  
'{ "name":"google" , "url":"www.google.com" },' +  
'{ "name":"微博" , "url":"www.weibo.com" } ]}';
```

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

```
var txt = '{ "sites" : [' +  
'{ "name":"菜鸟教程" , "url":"www.runoob.com" },' +  
'{ "name":"google" , "url":"www.google.com" },' +  
'{ "name":"微博" , "url":"www.weibo.com" } ]}';  
var obj = eval ("(" + txt + ")");  
document.getElementById("name").innerHTML=obj.sites[0].name  
document.getElementById("url").innerHTML=obj.sites[0].url
```

[尝试一下 »](#)

JSON 解析器



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

[尝试一下 »](#)

对于较老的浏览器，可使用 JavaScript 库：<https://github.com/douglascrockford/JSON-js>

JSON 格式最初是 [originally specified by Douglas Crockford](#)

[← JSON 教程](#)[JSON 语法 →](#)[✎ 点我分享笔记](#)

JSONP

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_string [, $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)  
}
```

[← PHP 5 常量](#)[PHP MySQL 插入多条数据 →](#)[✎ 点我分享笔记](#)

