javaScript和JSON

> 本文输出和JSON有关的以下内容
> ❒ JSON和javaScript
> ❒ JSON的语法介绍
> ❒ JSON的数据类型
> ❒ JSON和XMLHTTPRequest
> ❒ JSON的序列化和反序列化处理

1.1 JSON和javaScript

JSON是一种数据交换格式。

JSON的全称是JavaScript Object Notation，翻译为JavaScript对象表示法。JSON的这个全称，无疑让很多人既兴奋又困惑，兴奋的人直接认为这就是JavaScript中的对象，困惑的人觉察出JSON数据和JavaScript对象好像有些不一样。接下来我们先谈一谈JSON数据和JavaScript的关系。

诚然，从JSON的全称可以看出JSON和JavaScript语言必定有种某种神秘关联，至少能够确定的是JSON的命名确实来源于JavaScript这门语言。

JSON基于JavaScript对象字面量，但JSON本身是一种数据交换格式，因此它是独立于语言的。JSON全称为JavaScript对象表示法，在理解的时候可以认为JSON ==> JavaScript && 对象 && 表示法

JavaScript我们知道是一门动态脚本语言，那么 对象表示法 是什么?

对象是面向对象编程语言中一种常见的数据类型，表示键值对的集合，那么 表示法 是什么?

表示法:是指一个可以表示诸如数字或单词等数据的字符系统。

JSON起源于JavaScript（灵感来源于JavaScript的对象语法），但真正重要的是具体的表示法本身。JSON不仅独立于语言，而且使用了一种在许多编程语言中能够找到共同元素的表达方式。基于这种简洁的表达方式，JSON迅速成为一种流行的数据交换格式。目前，客户端和服务器端在进行数据通信的时候，常见的数据格式就是JSON和XML。

1.2 JSON的语法介绍

1.2.1 JSON的语法

JSON因为基于JavaScript的字面量，所以我们先来看下JavaScript字面量的样子，下面给出简单的代码示例，描述了一个书对象。

```
var  book = {
```

```
        name:"声名狼藉者的生活",
        price:42.00,
        author:"福柯",
        press:"北京大学出版社",
        read:function () {
            console.log("我的书名为:声名狼藉者的的生活,作者为福柯....");
        }
    };
```

顺便贴出一个简短的JSON数据

```JSON
{
        "name":"声名狼藉者的生活",
        "price":42.00,
        "author":"福柯",
        "press":"北京大学出版社",
        "content":["a","b","c",123]
    }
```

我们可以对比下上面的JavaScript对象和JSON数据，会发现它们的结构和语法形式很像，都是键值对的集合，接下来我们做更详细的说明。JSON数据在表达上和对象保持一致，但因为数据交换格式的核心是数据，所以 JSON并不会保存函数等信息 。JSON数据所基于的JavaScript对象字面量单纯指对象字面量以及其属性的语法表示。

JSON的主要语法特点

> ① 以键值对的方式来保存数据
> ② 标准的JSON数据的key必须要使用双引号包裹
> ③ {} 用于表示和存放对象，[] 用于表示和存放数组数据

JSON数据的读取，在读取JSON的时候

```
{ 表示开始读取对象，} 表示对象读取结束
[ 表示开始读取数组，] 表示数组读取结束
： 用于分隔键值对中的key和value
， 用于分隔对象中的多个键值对或者是数组中的多个元素
```

> ! JavaScript对象字面量中的key可以使用单引号，可以使用双引号，可以不必加上引号包裹，但是在JSON中，所有的key必须要加上双引号。

1.2.2 JSON的验证和格式化工具

下面列出一些能够对JSON数据进行校验和格式化的在线地址
https://jsonlint.com/

http://tool.oschina.net/codeformat/json
https://jsonformatter.curiousconcept.com/

1.2.3 JSON文件和MIME类型

在开发中我们经常需要处理大量的JSON数据，JSON这种数据交换格式可以作为独立的文件存在于文件系统中，文件扩展名为 `.json`

JSON的MIME类型是 `application/json` ，详细信息请参考IANA官网维护的所有媒体类型列表。

1.3 JSON的数据类型

JSON中（作为value值）的数据类型包括 对象、字符串、数字、布尔值、null和数组六种 。

① 字符串

JSON中的字符串可以由任何的Unicode字符构成，字符串的两边必须被双引号包裹。需要注意的是：虽然在JavaScript语言中字符串可以使用单引号来包裹，但是在JSON中的字符串必须使用双引号包裹。

如果字符串中存在以下特殊字符，那么需要在它们的前面加上一个反斜线（\）来进行转义。

- " 双引号
- \ 反斜线
- \/ 正斜线
- \b 退格符
- \f 换页符
- \t 制表符
- \n 换行符
- \r 回车符
- \u 后面跟16进制字符

② 数字

JSON中的数字可以是整数、小数、负数或者是指数。

③ 布尔类型

JSON数据仅仅支持小写形式的布尔类型值：true 和 false。

④ null类型

JSON中没有undefined这种数据类型，它使用null表示空，并且必须小写。
在JavaScript语言中，var obj = null 表示把obj这个对象清空，它和undefined不太一样，null表示什么都没有，undefined表示未定义。

⑤ 对象类型

对象类型是使用逗号分隔的键值对的集合，使用大括号(`{}`)裹。

⑥ 数组类型

数组类型是元素的集合，每个元素都可以是字符串、数字、布尔值、对象或者数组中的任何一种。元素与元素之间使用逗号隔开，所有的元素被方括号（ `[]` ）包裹，建议数组中所有的元素都应该是相同数据类型的。

1.4 JSON和XMLHTTPRequest

在前端开发中有一种发送网络请求的技术Ajax，它可以实现异步处理网络通信而不刷新页面。

Ajax的全称为Asynchronous JavaScript and XML，即异步的JavaScript和XML。我们知道JSON的定位是轻量级的数据交互格式，客户端在和服务器端进行网络通信的时候，服务器端返回给我们的数据大多数是JSON或者是XML。也就是说JSON数据在Ajax网络通信中可能扮演重要的角色， 那什么Ajax不叫异步的JSON而叫做异步的XML呢? 答案是：因为刚提出这种网络请求技术的时候，XML相比JSON更流行。

在Ajax网络请求中用到的核心对象XMLHTTPRequest也是如此，其实这个对象命名中包含XML也仅仅是因为对于当时而言，XML是网络请求中最常用的数据交换格式。如果放在今天，那么它们的名字应该叫做AjaJ(Asynchronous JavaScript and JSON)和JSONHTTPRequest更合适一些。

1.5 JavaScript中JSON数据的序列化和反序列化处理

在网络请求中，如果服务器返回给我们的数据是JSON数据，那么为了方便对数据的操作，通常我们在网络请求成功拿到JSON数据之后会先对JSON数据进行反序列化操作。
在前端开发中，早期的JSON解析基本上由eval函数来完成，ECMAScript5对解析JSON的行为进行了规范，定义了全局对象JSON。目前IE8+、FireFox 3.5+、Opera 10.5、Safari 4+和Chrome等浏览器均支持原生的JSON全局对象。

JSON数据的处理主要涉及到两方面：序列化处理和反序列化处理

1.5.1 使用eavl函数来处理JSON数据

eavl函数说明

JavaScript语言中eavl函数可以把字符串转换为js的代码并且马上执行，使用情况和Function构造函数用法类型。

```
eval("var a = 123;");
console.log(a + 1);  //输出结果为124
```

因为从某种程度上来讲，json其实是JavaScript语言的严格子集，所以我们可以直接通过eval函数来对json数据进行解析。需要注意的是，使用eavl函数来对json数据结构求值存在风险，因为可能会执行一些恶意代码。

eavl函数解析JSON

服务器返回给前端的json数据可能是 `{...}` 形式的，也可能是 `[...]` 形式的，分别对应js中的对象和数组。如果是 `{...}` 形式的，那么在解析的时候，如果直接以eval(json)的方式处理会报错，因为js中不允许直接写{name:”zs”}类似的语句。遇到这种结构的json数据，通常我们有两种方式进行处理：① 包装成表达式 ② 赋值给变量。

```
//001 [...] 格式的json数据
```

```
var arrJson= '[{"name":"zs","age":18},{"name":"lisi","age":28}]';
var jsonArr = eval(arrJson);

//002 {...} 格式的json数据
var objJson = `{"name":"wendingding","age":18,"contentAbout":["JavaScript","CSS","HTML"],"car

//eval(json);  错误的演示：报错
//处理方式(1)：以拼接的方式赋值给变量
eval("var jsonObj1 = " + objJson);
//处理方式(2)：包装成表达式
var jsonObj2 = eval("(" + objJson +")");

//打印转换后得到的数组|对象
console.log(jsonArr);
console.log(jsonObj1);
console.log(jsonObj2);
```

```
▼ (2) [{…}, {…}]
  ▶ 0: {name: "zs", age: 18}
  ▶ 1: {name: "lisi", age: 28}
    length: 2
  ▶ __proto__: Array(0)
▼ {name: "wendingding", age: 18, contentAbout: Array(3), car: {…}}
    age: 18
  ▶ car: {number: "粤A6666", color: "red"}
  ▶ contentAbout: (3) ["JavaScript", "CSS", "HTML"]
    name: "wendingding"
  ▶ __proto__: Object
▶ {name: "wendingding", age: 18, contentAbout: Array(3), car: {…}}
```

1.5.2 使用JSON全局对象来处理JSON数据

JSON全局对象拥有两个方法：stringify()和parse()，其中parse方法用于把json数据反序列化为原生的js，stringify方法用于把js对象序列化为json字符串。

parse方法的使用

语法：JSON.parse(jsonString,[fn])

参数说明

> 第一个参数：jsonString为要解析的json字符串
> 第二个参数：fn是一个可选参数，该参数为函数类型，接收两个参数，分别是每个键值对的key和value。

```
//json字符串
var objJson = `{"name":"wendingding","age":18,"contentAbout":["JavaScript","CSS","HTML"],"car

//把json字符串转换为js数组
var arrJson= '[{"name":"zs","age":18},{"name":"lisi","age":28}]';
```

```
var arrJson= "[{"name":"zs","age":18},{"name":"lisi","age":28}]";

//把json字符串转换为js对象
var jsonObj = JSON.parse(objJson);
var jsonArr = JSON.parse(arrJson);
console.log(jsonObj);
console.log(jsonArr);

//演示parse方法中函数参数的使用
function fn(key, value) {
    if (key === "name") {
        return value + "++"          //在原有value值的基础上拼接++字符串
    } else if (key === "age") {
        return undefined             //如果返回undefined，则表示删除对应的键值对
    } else {
        return value                 //正常返回对应的value值
    }
}
console.log(JSON.parse(objJson, fn));
```

stringify方法使用说明

语法：JSON.stringify(Obj,[fn|arr],[space])

参数说明

> 第一个参数：Obj为要进行序列化操作的JavaScript对象
> 第二个参数：过滤器，可以是函数或者是一个数组
> 第三个参数：是否在生成的json字符串中保留缩进，用于控制缩进的字符

```
//js中的普通对象
var obj = {
    name:"zs",
    age:18,
    friends:["小霸王","花仙子","奥特曼"],
    other:undefined,
    showName:function () {
        console.log(this.name);
    }
```

```
};


//把js中的对象转换为json字符串
//注意：
//001 如果键值对中存在value值为undefined的数据，那么会被跳过
//002 对象中的方法以及该对象的原型成员数据在进行转换的时候，会被有意忽略
console.log(JSON.stringify(obj));

//控制缩进，该参数的值可以是数字也可以是字符串，自动换行
//001 如果是字符串那么会把对应的字符拼接在键值对前面，超过10个字符的省略
//002 如果是数字那么会设置对应的缩进，最多为10，超过则默认为10
console.log(JSON.stringify(obj, null, 4));
console.log(JSON.stringify(obj, null, "@@"));

//过滤器（数组）：表示只处理key为name和age这两个键值对
console.log(JSON.stringify(obj, ["name","age"]));

//过滤器（函数）：
function fn(key,value) {
    if (key === "age")
    {
        return value + 20;
    }else if (key === "name")
    {
        return undefined;        //过滤掉key为name这个键值对
    }else
    {
        return value;
    }
}
console.log(JSON.stringify(obj,fn));
```

```
{"name":"zs","age":18,"friends":["小霸王","花仙子","奥特曼"]}
{
    "name": "zs",
    "age": 18,
    "friends": [
        "小霸王",
        "花仙子",
        "奥特曼"
    ]
}
{
@@"name": "zs",
@@"age": 18,
@@"friends": [
@@@@"小霸王",
@@@@"花仙子",
@@@@"奥特曼"
@@]
}
{"name":"zs","age":18}
{"age":38,"friends":["小霸王","花仙子","奥特曼"]}
```

JSON数据总结

> ❒ JSON全称是JavaScript Object Notation基于JavaScript，是JavaScript的子集。
> ❒ JSON虽然是JavaScript的子集，但并不从属于JavaScript，它独立于语言。

> ❒ JSON是用来表示和传输数据的格式，比XML更轻量级，现已成为web数据交换的事实标准。
> ❒ JSON的优势在于其可以方便的把JSON字符串数据转换为对应的对象，比XML更方便且数据更小。
> ❒ JSON语法可以表示：字符串、数值、布尔值、null、对象和数组6种类型的值，不支持undefined。
> ❒ JSON中的”键”区别于JavaScript，必须要加上双引号。
> ❒ JSON解析可以使用传统的eval函数，或ECMAScript5推出的全局对象来处理。

参考资料

JSON官网：http://json.org/
JSON维基百科：https://en.wikipedia.org/wiki/JSON
JSON作者简介：https://en.wikipedia.org/wiki/Douglas_Crockford
JSON必知必会:https://book.douban.com/subject/26789960/
JavaScript高级程序设计：https://book.douban.com/subject/10546125/