# 1. 数据格式提要

1. 在服务器端 AJAX 是一门与语言无关的技术。在业务逻辑层使用何种服务器端语言都可以。
2. 从服务器端接收数据的时候，那些数据必须以浏览器能够理解的格式来发送。服务器端的编程语言只能以如下 3 种格式返回数据：
    - XML
    - JSON
    - HTML

# 2. XML

- 优点：
    - XML 是一种通用的数据格式。
    - 不必把数据强加到已定义好的格式中，而是要为数据自定义合适的标记。
    - 利用 DOM 可以完全掌控文档。
- 缺点：
    - 如果文档来自于服务器，就必须得保证文档含有正确的首部信息。若文档类型不正确，那么 responseXML 的值将是空的。
    - 当浏览器接收到长的 XML 文件后， DOM 解析可能会很复杂
- 示例代码

```
<script type="text/javascript">
    window.onload = function () {
        var aNodes = document.getElementsByTagName("a");
        for(var i = 0; i<aNodes.length; i++) {
            aNodes[i].onclick = function() {

                var xhr = new XMLHttpRequest();
                var method = "GET";
                var url = this.href;

```

```
                xhr.open(method,url);
                xhr.send(null);

                xhr.onreadystatechange = function() {
                    if(xhr.readyState == 4) {
                        if(xhr.status == 200 || xhr.status == 304) {
                            //先把xml文件里面的数据取出来
                            var requestXml = xhr.responseXML;


                            var name = requestXml.getElementsByTagName("name")[0].firstChild.nodeValue;
                            var website = requestXml.getElementsByTagName("website")[0].firstChild.nodeValue;
                            var email = requestXml.getElementsByTagName("email")[0].firstChild.nodeValue;

                            /* alert(name);
                            alert(website);
                            alert(email); */

                            var aNode = document.createElement("a");
                            aNode.appendChild(document.createTextNode(name));
                            aNode.href = "myEmail: "+email;


                            var h2Node = document.createElement("h2");
                            h2Node.appendChild(aNode);

                            var webANode = document.createElement("a");

                            webANode.appendChild(document.createTextNode(website));
                            webANode.href = website;


                            var details = document.getElementById("details");
                            details.innerHTML = "";
                            details.appendChild(h2Node);
                            details.appendChild(webANode);

                        }
                    }
                }

                return false;
            }
```

```
        }
    }
</script>
```

# 3. Json

- JSON（JavaScript Object Notation）一种简单的数据格式，比xml更轻巧。JSON是JavaScript原生格式，这意味着在JavaScript中处理JSON数据不需要任何特殊的API或工具包。
- JSON的规则很简单：对象是一个无序的“‘名称/值’对”集合。一个对象以“{”（左括号）开始，“}”（右括号）结束。每个“名称”后跟一个“:”（冒号）；“‘名称/值’对”之间使用“,”（逗号）分隔。

## 3.1 JSON 示例

```
var user =
{
    "username":"andy",
    "age":20,
    "info": {"tel":"123456","cellphone":"98765"},
    "address":
        [
            {"city":"beijing","postcode":"222333"},
            {"city":"newyork","postcode":"555666"}
        ]
}

alert(user.username);
alert(user.age);
alert(user.info.cellphone);
alert(user.address[0].city);
alert(user.address[0].postcode);
```

- JSON 用冒号(而不是等号)来赋值。每一条赋值语句用逗号分开。整个对象用大括号封装起来。可用大括号分级嵌套数据。
- 对象描述中存储的数据可以是字符串，数字或者布尔值。对象描述也可存储函数，那就是对象的方法。

## 3.2 解析JSON

- JSON 只是一种文本字符串。它被存储在 responseText 属性中
- 为了读取存储在 responseText 属性中的 JSON 数据，需要根据 JavaScript 的 eval 语句。函数 eval 会把一个字符串当作它的参数。然后这个字符串会被当作 JavaScript 代码来执行。因为 JSON 的字符串就是由 JavaScript 代码构成的，所以它本身是可执行的

- 代码实例：

```
var jsonResponse = xhr.responseText;
var personObject = eval("(" + jsonResponse + ")");
var name = personObject.person.name;
var website = personObject.person.website;
var email = personObject.person.email;
```

- JSON提供了json.js包，下载 http://www.json.org/json.js 后，使用parseJSON()方法将字符串解析成 JS 对象

```
var jsonResponse = xhr.responseText;
var personObject = jsonResponse.parseJSON();
var name = personObject.person.name;
var website = personObject.person.website;
var email = personObject.person.email;
```

## 3.3 代码示例

```
<script type="text/javascript">
    window.onload = function () {
        var aNodes = document.getElementsByTagName("a");
        for(var i = 0; i<aNodes.length; i++) {
            aNodes[i].onclick = function() {

                var xhr = new XMLHttpRequest();
                var method = "GET";
                var url = this.href;

                xhr.open(method,url);
                xhr.send(null);

                xhr.onreadystatechange = function() {
                    if(xhr.readyState == 4) {
                        if(xhr.status == 200 || xhr.status == 304) {
                            //先把json文件里面的数据取出来
                            var requestJson = xhr.responseText;

                            var json = eval("("+requestJson+")");

                            var name = json.person.name;
                            var website = json.person.website;
                            var email = json.person.email;
```

```
25
26                              /* alert(name);
27                              alert(website);
28                              alert(email); */
29
30                              var aNode = document.createElement("a");
31                              aNode.appendChild(document.createTextNode(name));
32                              aNode.href = "myEmail: "+email;
33
34
35                              var h2Node = document.createElement("h2");
36                              h2Node.appendChild(aNode);
37
38                              var webANode = document.createElement("a");
39
webANode.appendChild(document.createTextNode(website));
40                              webANode.href = website;
41
42
43                              var details = document.getElementById("details");
44                              details.innerHTML = "";
45                              details.appendChild(h2Node);
46                              details.appendChild(webANode);
47
48                          }
49                      }
50                  }
51
52                  return false;
53              }
54          }
55      }
56    </script>
```

## 3.4 JSON 小结

- 优点：
  - 作为一种数据传输格式，JSON 与 XML 很相似，但是它更加灵巧。
  - JSON 不需要从服务器端发送含有特定内容类型的首部信息。

- 缺点：
  - 语法过于严谨
  - 代码不易读
  - eval 函数存在风险

# 4. HTML

- HTML 由一些普通文本组成。如果服务器通过 XMLHttpRequest 发送 HTML， 文本将存储在 responseText 属性中。
- 不必从 responseText 属性中读取数据。它已经是希望的格式，可以直接将它插入到页面中。
- 插入 HTML 代码最简单的方法是更新这个元素的 innerHTML 属性。

- 优点：
  - 从服务器端发送的 HTML 代码在浏览器端不需要用 JavaScript 进行解析。
  - HTML 的可读性好。
  - HTML 代码块与 innerHTML 属性搭配，效率高。
- 缺点：
  - 若需要通过 AJAX 更新一篇文档的多个部分，HTML 不合适
  - innerHTML 并非 DOM 标准。

- 示例代码

```
<script type="text/javascript">
    window.onload = function() {

        var aNodes = document.getElementsByTagName("a");

        for(var i = 0; i<aNodes.length; i++) {
            aNodes[i].onclick = function() {

                var xhr = new XMLHttpRequest();
                var url = this.href;
                var method = "GET";

                xhr.open(method, url);
```

```
                xhr.send(null);

                xhr.onreadystatechange = function () {
                    if(xhr.readyState==4) {
                        if(xhr.status==200||xhr.status==304) {
                            document.getElementById("details").innerHTML =
xhr.responseText;
                        }
                    }
                }

                return false;
            }
        }
    }
  </script>
```

## 5. 对比小结

- 若应用程序不需要与其他应用程序共享数据的时候, 使用 HTML 片段来返回数据时最简单的
- 如果数据需要重用, JSON 文件是个不错的选择, 其在性能和文件大小方面有优势
- 当远程应用程序未知时, XML 文档是首选, 因为 XML 是 web 服务领域的 “世界语”