中南大學
CENTRAL SOUTH UNIVERSITY

第七讲　前后端数据交互

盛 羽

中南大学计算机学院

shengyu@csu. edu. cn

目录

1.数据交互格式

- 数据交互格式
  - 在计算机的不同程序之间，或者不同的编程语言之间进行交换数据，也需要一种大家都能听得懂得‘语言’，这就是数据交换格式
  - JSON、XML……

目录

2.XML

- XML
  - 类似于HTML 的标记语言，但是 XML 没有使用预定义的标记
  - 根据需求定义标记
  - 可用来用来传输和存储数据
  - 基本格式是标准化的，可跨系统或平台共享或传输 XML

- 结构XML文档
  - XML声明
    - 并非是一种标签，其用来传播文档的元数据

```
<?xml version="1.0" encoding="UTF-8"?>
```

    - 版本：当前文档使用的版本；编码：当前文档使用的编码
  - 注释

```
<!-- Comment -->
```

XML语法规则

XML 文档必须有根元素

```
<root>
  <child>
    <subchild>.....</subchild>
  </child>
</root>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
```

- XML语法规则
  - 所有的 XML 元素都必须有一个关闭标签
    - 在 XML 中，省略关闭标签是非法的。
    - 所有元素都必须有关闭标签
  - XML 标签对大小写敏感

```
<Message>这是错误的</message>
<message>这是正确的</message>
```

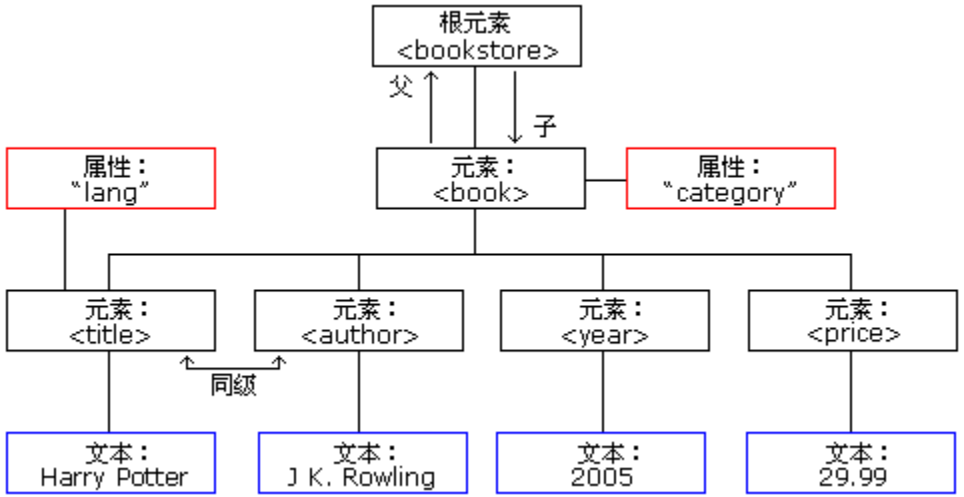
  - XML 必须正确嵌套
  - XML 属性值必须加引号

XML 树结构

```xml
<?xml version="1.0" encoding="UTF-8"?>
<bookstore>
    <book category="COOKING">
        <title lang="en">Everyday Italian</title>
        <author>Giada De Laurentiis</author>
        <year>2005</year>
        <price>30.00</price>
    </book>
    <book category="CHILDREN">
        <title lang="en">Harry Potter</title>
        <author>J K. Rowling</author>
        <year>2005</year>
        <price>29.99</price>
    </book>
    <book category="WEB">
        <title lang="en">Learning XML</title>
        <author>Erik T. Ray</author>
        <year>2003</year>
        <price>39.95</price>
    </book>
</bookstore>
```

所有其他元素的父元素;

，这棵树从根部开始，并扩展到树的最底端;



- XML验证
  - DTD
    - DTD（文档类型定义）的作用是定义 XML 文档的合法构建模块
    - DTD 可被成行地声明于 XML 文档中，也可作为一个外部引用

```
<?xml version="1.0"?>
<!DOCTYPE note [
<!-- 定义此文档是 note 类型的文档 -->
<!ELEMENT note (to,from,heading,body)>
<!-- 定义 note 元素有四个元素："to、from、heading,、body" -->
<!ELEMENT to (#PCDATA)>
<!-- 定义 to 元素为 "#PCDATA" 类型 -->
<!ELEMENT from (#PCDATA)>
<!ELEMENT heading (#PCDATA)>
<!ELEMENT body (#PCDATA)>
]>
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend</body>
</note>
```

```
<?xml version="1.0"?>
<!DOCTYPE note SYSTEM "note.dtd">
<note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
```

目录

3.JSON

- JSON
  - JavaScript对象表示法（JavaScript Object Notation）
  - 按照JavaScript对象语法的数据格式
  - 独立于JavaScript
  - JSON可以作为一个对象或者字符串存在
    - 对象：用于解读 JSON 中的数据
    - 字符串：用于通过网络传输 JSON 数据

3.JSON

JSON结构

- 可以把 JavaScript 对象原原本本的写入 JSON 数据：字符串，数字，数组，布尔还有其它的字面值对象
- 以保存为一个名为 superHeroes 对象为例

```
superHeroes.hometown
superHeroes["active"]
superHeroes["members"][1]["powers"][2]
```

- superHeroes["members"][1]["powers"][2]

1. 首先我们有变量名 superHeroes，储存对象
2. 在对象中我们想访问 members 属性，所以我们使用 ["members"]
3. members 包含有对象数组，我们想要访问第二个元素，所以我们使用[1]
4. 在对象内，我们想访问 powers 属性，所以我们使用 ["powers"]
5. powers 属性是一个包含英雄技能的数组。我们想要第三个，所以我们使用[2]

```
{
  "squadName" : "Super hero squad",
  "homeTown" : "Metro City",
  "formed" : 2016,
  "secretBase" : "Super tower",
  "active" : true,
  "members" : [
    {
      "name" : "Molecule Man",
      "age" : 29,
      "secretIdentity" : "Dan Jukes",
      "powers" : [
        "Radiation resistance",
        "Turning tiny",
        "Radiation blast"
      ]
    },
    {
      "name" : "Eternal Flame",
      "age" : 1000000,
      "secretIdentity" : "Unknown",
      "powers" : [
        "Immortality",
        "Heat Immunity",
        "Inferno",
        "Teleportation",
        "Interdimensional travel"
      ]
    }
  ]
}
```

3.JSON

- JSON结构
  - JSON 对象基于JavaScript 对象
  - 数组对象也是一种合法的 JSON 对象
  - 通过数组索引就可以访问数组元素
    - 如[0]["powers"]

```
[
  {
    "name" : "Molecule Man",
    "age" : 29,
    "secretIdentity" : "Dan Jukes",
    "powers" : [
      "Radiation resistance",
      "Turning tiny",
      "Radiation blast"
    ]
  },
  {
    "name" : "Madame Uppercut",
    "age" : 39,
    "secretIdentity" : "Jane Wilson",
    "powers" : [
      "Million tonne punch",
      "Damage resistance",
      "Superhuman reflexes"
    ]
  }
]
```

- JSON其他注意事项
  - JSON 是一种纯数据格式，它只包含属性，没有方法。
  - JSON要求在字符串和属性名称周围使用双引号,单引号无效。
  - 甚至一个错位的逗号或分号就可以导致 JSON 文件出错。您应该小心的检查您想使用的数据(虽然计算机生成的 JSON 很少出错，只要生成程序正常工作)。您可以通过像 JSONLint 的应用程序来检验 JSON。
  - JSON 可以将任何标准合法的 JSON 数据格式化保存，不只是数组和对象。比如，一个单一的字符串或者数字可以是合法的 JSON 对象。虽然不是特别有用处.....
  - 与 JavaScript 代码中对象属性可以不加引号不同，JSON 中只有带引号的字符串可以用作属性。

- JSON对象和文本间的转换
  - 浏览器拥有一个内建的 JSON
  - 包含以下两个方法
    - parse(): 以文本字符串形式接受JSON对象作为参数，并返回相应的对象。
    - stringify(): 接收一个对象作为参数，返回一个对应的JSON字符串。

```
var myJSONstring='{"name": "Chris", "age" : "38"}';
var myJSON=JSON.parse(myJSONstring);
myJSON
var myString = JSON.stringify(myJSON);
myString
```

  - 一个例子：SupperHero

目录

4.Ajax

- Ajax
  - Asynchronous JavaScript + XML（异步JavaScript和XML）
  - JSON的使用比XML更加普遍
  - 使用XMLHttpRequest与服务端进行交互
  - 可以发送和接受包括JSON, XML, HTML, and text files等类型数据
  - Asynchronous（异步）
    - 使用Ajax技术与服务器通信、交换数据和更新页面都不需要刷新整个页面
  - 两个主要功能：
    - 向服务器发送请求而无需重新加载整个页面
    - 接受和处理服务器发过来的数据

Ajax使用

1. 发送HTTP请求

- 向服务器发送HTTP请求并接收返回

```
httpRequest = new XMLHttpRequest();
```

- Onreadystatechange属性
  - 指明当request状态发生改变时，哪个JS函数来处理这个响应

```
httpRequest.onreadystatechange = nameOfTheFunction;
```

```
httpRequest.onreadystatechange = function(){
    // Process the server response here.
};
```

Ajax使用

1. 发送HTTP请求

- 完成Onreadystatechange定义后，方可进行实际request
- open() and send()

```
httpRequest.open('GET', 'http://www.example.org/some.file', true);
httpRequest.send();
```

- open()
  - 第一个参数：HTTP方法，GET, POST, HEAD等
  - 第二个参数：接收请求的URL。默认不能访问第三方资源。
  - 第三个参数：可选。是否为异步，默认True。异步状态下，在服务端响应返回前，还可以执行其他页面操作。
- send()
  - 使用POST方法时，函数中参数可以为任何想发送到服务器的数据

Ajax使用

2. 处理服务器响应

- 发送请求时定义了处理响应的函数

```
httpRequest.onreadystatechange = nameOfTheFunction;
```

- 处理流程

1. 检查请求状态。如果状态为XMLHttpRequest.DONE（也就是4），则表示已接收到服务端的响应且能够进入下一处理流程。

```
if (httpRequest.readyState === XMLHttpRequest.DONE) {
    // Everything is good, the response was received.
} else {
    // Not ready yet.
}
```

- readyState
  - 0 (uninitialized) or (request not initialized)
  - 1 (loading) or (server connection established)
  - 2 (loaded) or (request received)
  - 3 (interactive) or (processing request)
  - 4 (complete) or (request finished and response is ready)

Ajax使用

2. 处理服务器响应

- 检查HTTP 响应状态代码。（200 OK？）

```
if (httpRequest.status === 200) {
    // Perfect!
} else {
    // There was a problem with the request.
    // For example, the response may have a 404 (Not Found)
    // or 500 (Internal Server Error) response code.
}
```

- 处理和分析服务器返回的数据
  - httpRequest.responseText
  - httpRequest.responseXML

- 注：
  - 以上为异步方式的处理方法，如果为同步方式，无需定义函数。但是用户体验会很差

4.Ajax

- Ajax使用
  - 示例
  - 09-Ajax-Demo

```
<button id="ajaxButton" type="button">Make a request</button>

<script>
  var httpRequest;
  document.getElementById("ajaxButton").addEventListener('click', makeRequest);

  function makeRequest() {
    httpRequest = new XMLHttpRequest();

    if (!httpRequest) {
      alert('Giving up :( Cannot create an XMLHTTP instance');
      return false;
    }
    httpRequest.onreadystatechange = alertContents;
    httpRequest.open('GET', 'test.html');
    httpRequest.send();
  }

  function alertContents() {
    if (httpRequest.readyState === XMLHttpRequest.DONE) {
      if (httpRequest.status === 200) {
        alert(httpRequest.responseText);
      } else {
        alert('There was a problem with the request.');
      }
    }
  }
```

- Ajax使用
  - 示例
  - 09-Ajax-JSON

- 同源策略
  - 域名，协议，端口相同
- 跨域
  - 代理
  - CORS
    - W3C标准
    - "跨域资源共享"（Cross-origin resource sharing）
    - 它允许浏览器向跨源服务器，发出XMLHttpRequest请求，从而克服了AJAX只能同源使用的限制。
  - JSONP：JSON with Padding
    - 通过javascript callback的形式实现跨域访问