# XML解析使用入门

本篇将主要讨论和学习从原生JS操作XML数据到jQuery方式等等，实现基础入门。

## XML文件

a.xml

```
<?xml version="1.0" encoding="utf-8"?>
<books>
    <book>
        <name>《流浪地球》</name>
        <author>刘慈欣</author>
    </book>
    <book>
        <name>《三体》</name>
        <author>刘慈欣</author>
    </book>
    <book>
        <name>《时间移民》</name>
        <author>刘慈欣</author>
    </book>
</books>
```

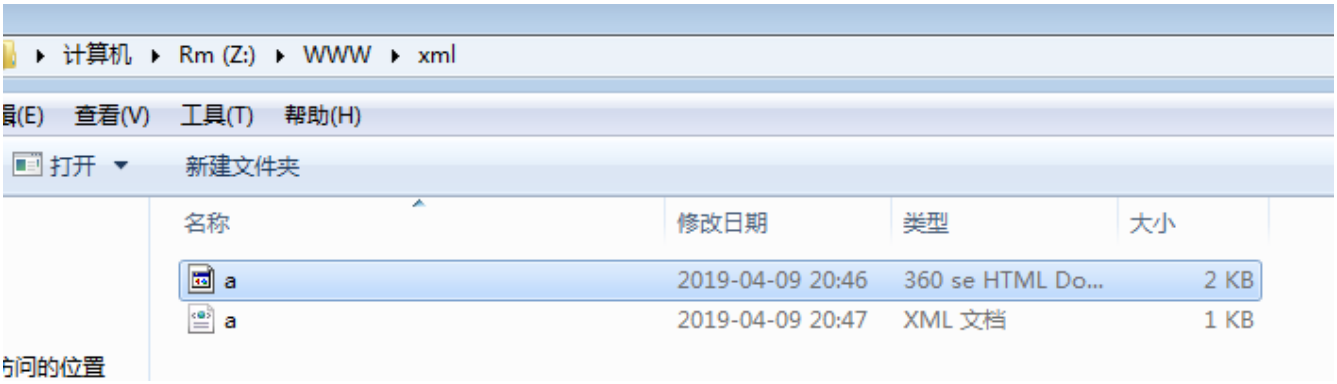
## 套路函数

一个固定套路函数，用于加载外部（同域下）的XML文件。

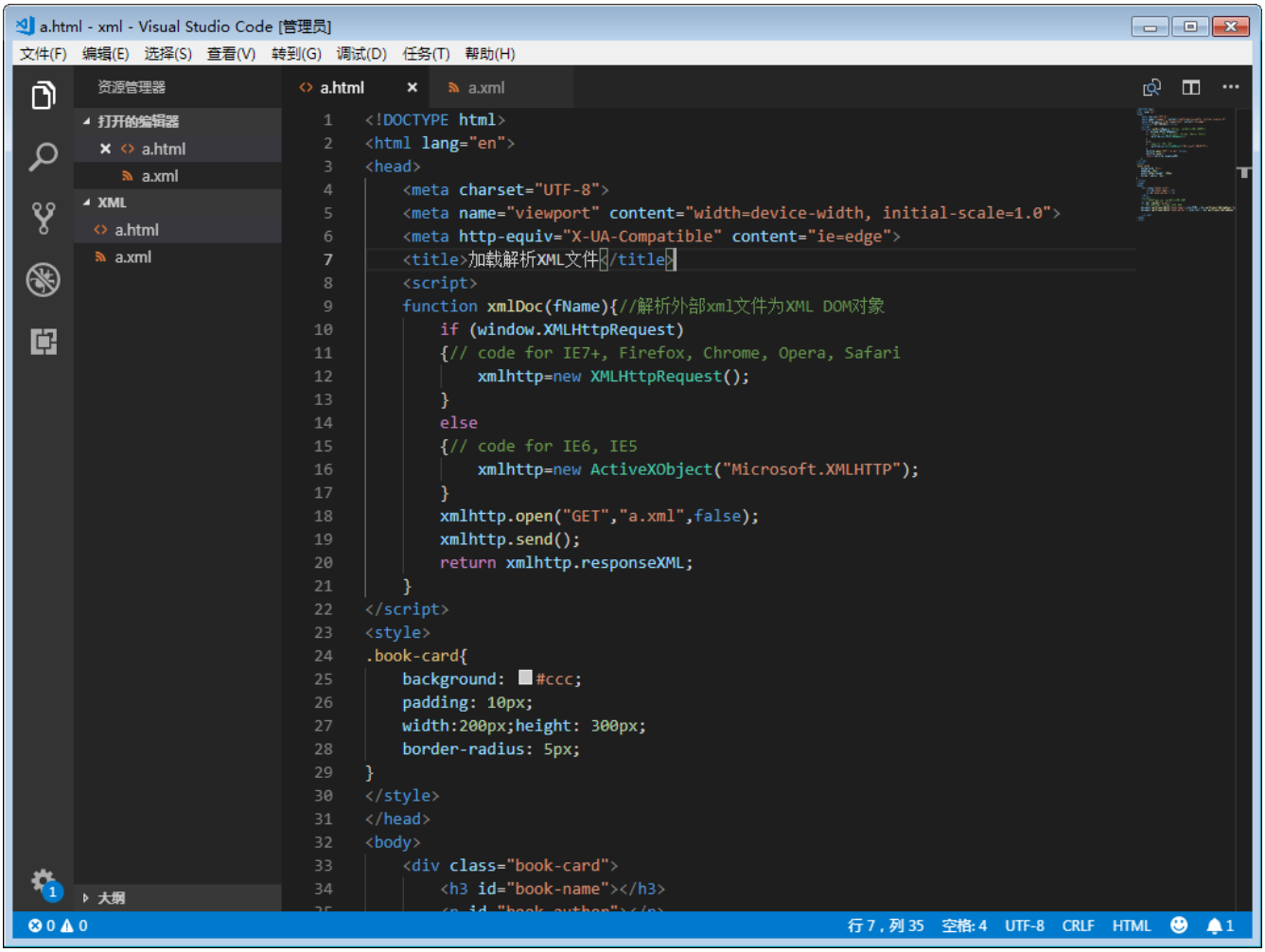
```
function xmlDoc(fName){//解析外部xml文件为XML DOM对象
    if (window.XMLHttpRequest)
    {// code for IE7+, Firefox, Chrome, Opera, Safari
        xmlhttp=new XMLHttpRequest();
    }
    else
    {// code for IE6, IE5
        xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
    }
    xmlhttp.open("GET",fName,false);
    xmlhttp.send();
    return xmlhttp.responseXML;
}
```

## 本地代码测试环境

由于xml文件不能跨域请求，所以本地测试时常常是无效的。首先需要建立一个服务器，此处笔者使用的是phpStudy。

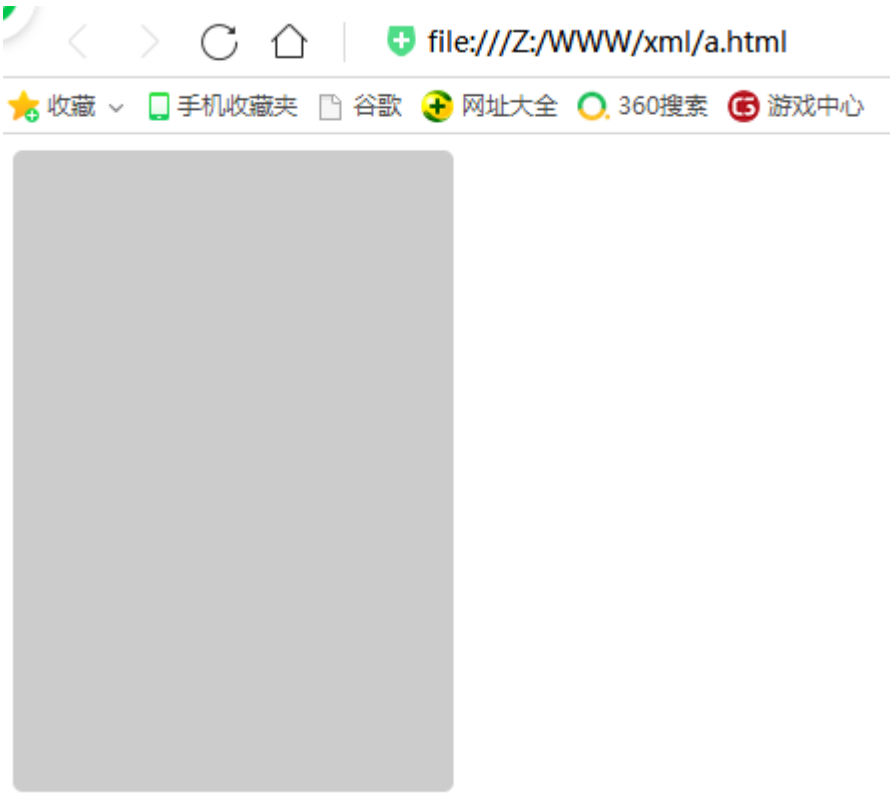



代码编辑器使用VS code:




虚拟服务器下运行效果：


同一html页面，直接本地方式打开将无法正常显示数据。



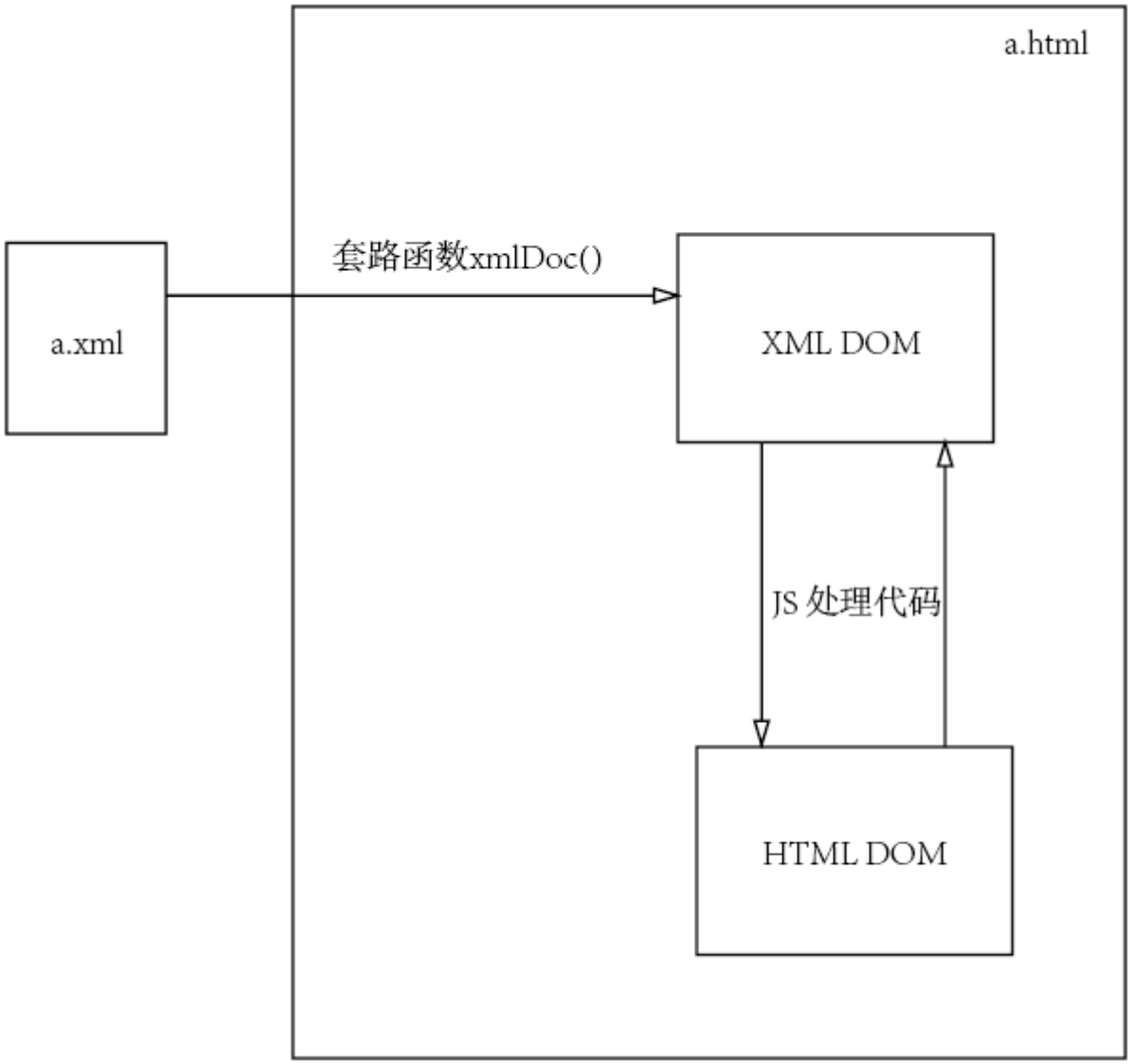

## 剩余代码

```
<!DOCTYPE html>
<html lang="en">
```

```
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>加载解析XML文件</title>
    <script src="xmlDoc.js"></script>
<style>
.book-card{
    background: #ccc;
    padding: 10px;
    width:200px;height: 300px;
    border-radius: 5px;
}
</style>
</head>
<body>
    <div class="book-card">
        <h3 id="book-name"></h3>
        <p id="book-author"></p>
    </div>
    <script>
    //实例化解析一个外部xml文件为XML DOM
    var doc =xmlDoc("a.xml");
    //从XML DOM获取数据显示到HTML DOM
    document.getElementById("book-name").innerText = doc.getElementsByTagName("book")
[0].getElementsByTagName("name")[0].childNodes[0].nodeValue
    document.getElementById("book-author").innerText = doc.getElementsByTagName("book")
[0].getElementsByTagName("author")[0].childNodes[0].nodeValue;
    </script>
</body>
</html>
```

## JS解析使用外部XML文件数据思路

如下图所示，为JS解析使用外部XML文件数据的思路。




## 循环遍历输出XML数据

上面的代码只是输出了一个数据项，而通常一个XML存储的是一个类似列表的多项同类型数据，这就需要使用for循环遍历输出了。

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>加载解析XML文件</title>
    <script src="xmlDoc.js"></script>
<style>
.book-card{
    background: #ccc;
    padding: 10px;
    width:200px;height: 300px;
    border-radius: 5px;
    margin-left: 10px;
```

```
    float: left;
}
.book-card:hover{
    background: #aaa;
}
</style>
</head>
<body>
    <script>
    //实例化解析一个外部xml文件为XML DOM
    var doc =xmlDoc("a.xml");
    //从XML DOM获取数据显示到HTML DOM
    var books=doc.getElementsByTagName("book");
    //循环遍历输出XML数据
    for(i=0;i<books.length;i++){
        document.write('<div class="book-card">');
        document.write('<h3>' + books[i].getElementsByTagName("name")
[0].childNodes[0].nodeValue + "</h3>");
        document.write('<p>' + books[i].getElementsByTagName("author")
[0].childNodes[0].nodeValue + "</p>");
        document.write('</div>');
    }
    </script>
</body>
</html>
```

输出效果如下：


## HTTP请求响应生成的XML

XML设计的初衷是为了数据交换，在服务器程序中，可以将数据库中的查询结果以xml文档格式返回客户端再进行解析和使用。

# 通过 PHP 生成 XML

如需使用 PHP 在服务器上生成 XML 响应，请使用下面的代码：

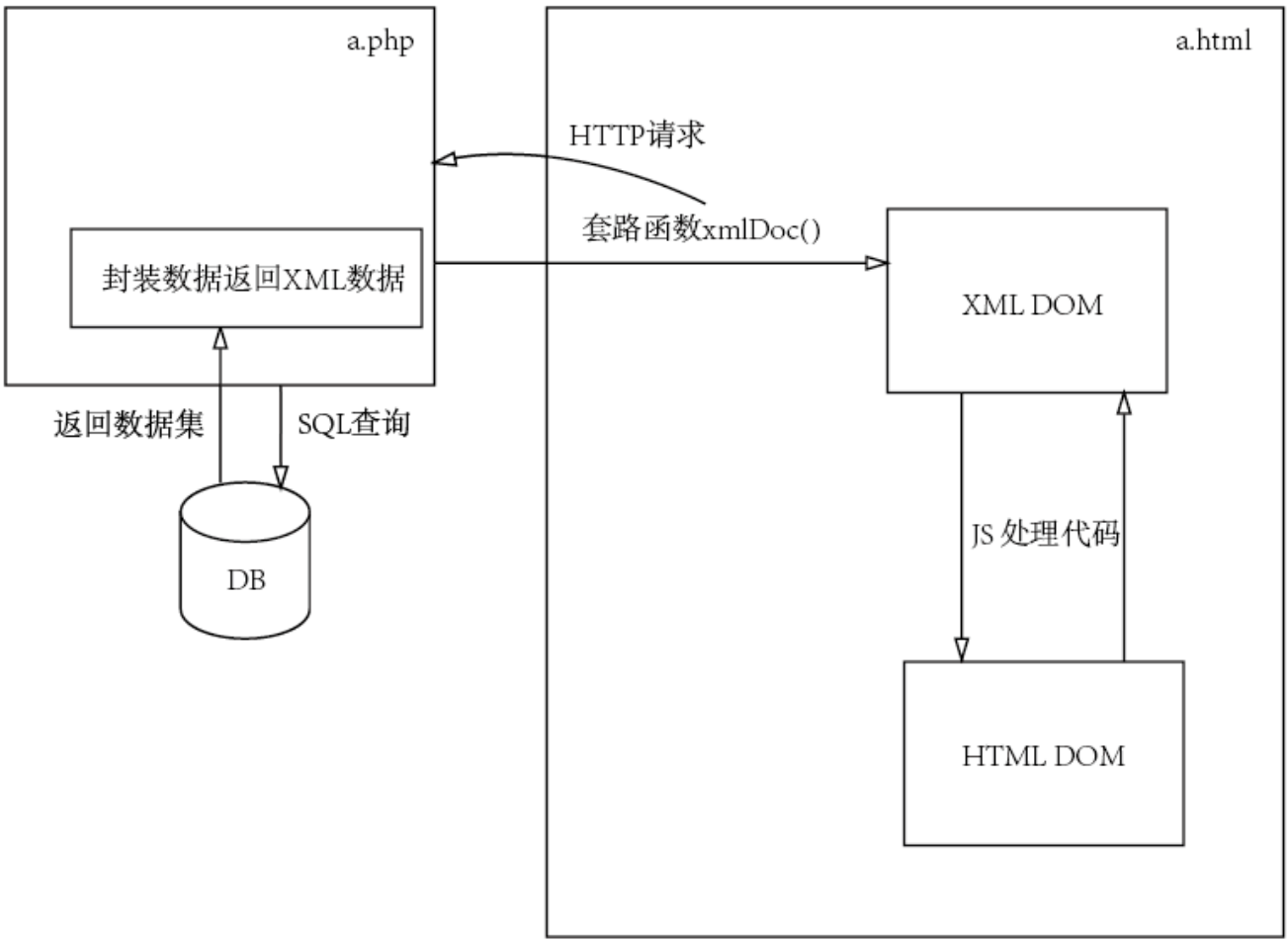
```
<?php
header("Content-type:text/xml");
echo "<?xml version='1.0' encoding='ISO-8859-1'?>";
echo "<note>";
echo "<from>John</from>";
echo "<to>George</to>";
echo "<message>Don't forget the meeting!</message>";
echo "</note>";
?>
```

请注意，响应头部的内容类型必须设置为 "text/xml"。

```php
<?php
header("Content-type:text/xml");
echo "<?xml version='1.0' encoding='ISO-8859-1'?>";
echo "<note>";
echo "<from>John</from>";
echo "<to>George</to>";
echo "<message>Don't forget the meeting!</message>";
echo "</note>";
?>
```

上面的php页面返回的结果是一个可用于解析使用的xml文件。（虽然URI仍然是php）

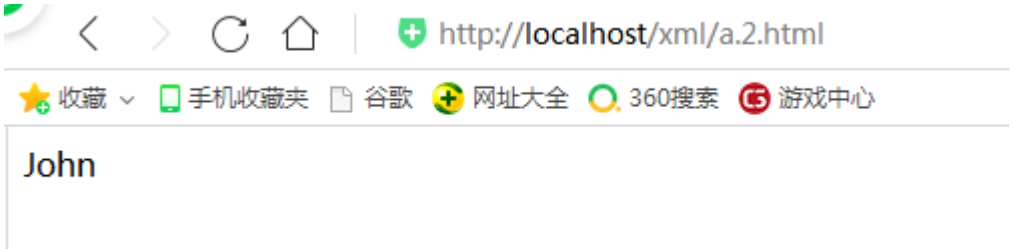

采用Ajax异步加载的话，可以实现数据的一系列交换。同样可以类比同为数据交换格式的JSON在网页前后端之间的数据交换行为和流程。




下面的代码就是用套路函数xmlDoc直接异步请求xml.php，并解析显示返回的XML数据结构中的数据。

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>加载解析XML文件</title>
    <script src="xmlDoc.js"></script>
</head>
<body>
    <script>
    //解析PHP动态生成的xml数据
    var doc =xmlDoc("xml.php");
    //从XML DOM获取数据显示到HTML DOM
    var from=doc.getElementsByTagName("from")[0].childNodes[0].nodeValue;
    document.write(from);
    </script>
</body>
</html>
```

运行结果如下：




## 类化封装

下面封装一个XML类，这样便可以不去理会套路函数xmlDoc()了。

```
/*
xml:封装xml文件解析和一些方法
*/
var XML={
    cNew:function(fName){//返回一个实例
        var obj={};
        //****************私有函数****************
        function xmlDoc(fName){//解析外部xml文件为XML DOM对象
            if (window.XMLHttpRequest)
            {// code for IE7+, Firefox, Chrome, Opera, Safari
                xmlhttp=new XMLHttpRequest();
            }
            else
            {// code for IE6, IE5
                xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
            }
            xmlhttp.open("GET",fName,false);
            xmlhttp.send();
            return xmlhttp.responseXML;
        }
        //****************公有方法****************
        obj.node=function(nodeName){//返回相应TagName的XML节点
            return this.doc.getElementsByTagName("from")[0].childNodes[0].nodeValue;
        }
        //****************公有成员****************
        obj.doc=xmlDoc(fName);
        return obj;
    }
}
```

使用时的代码修改如下：

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>加载解析XML文件</title>
    <script src="xmlDoc.js"></script>
</head>
```

```
<body>
    <script>
    //解析PHP动态生成的xml数据
    var xml = XML.cNew("xml.php");
    var doc =xml.doc;
    //从XML DOM获取数据显示到HTML DOM
    document.write(xml.node("form"));
    </script>
</body>
</html>
```