



# آموزش AJAX

ارائه شده به : دکتر حسن حقیقی

توسط : سیده زهرا حسینی و سیده الهه جلمبادانی

تیر ۹۰



## فهرست

4	..... Ajax
5	..... هدف AJAX
6	..... XMLHttpRequest شیء
7	..... اولین برنامه AJAX
7	..... سنگ بنای AJAX
10	..... readyState خصوصیت
11	..... ارسال درخواست به Server
12	..... کد ASP سمت Server
17	..... کد AJAX مربوط به فایل HTML
18	..... کد AJAX مربوط به فایل JavaScript
27	..... AJAX و بانک‌های اطلاعاتی
28	..... AJAX JavaScript
31	..... AJAX و تعامل با XML

## Ajax

AJAX مخفف Asynchronous JavaScript And XML است.

AJAX نوعی برنامه‌نویسی است که در سال ۲۰۰۵ توسط Google و به پیشنهاد آن، فراگیر شد.

AJAX یک زبان برنامه‌نویسی جدید نیست، بلکه یک روش جدید برای استفاده از استانداردهای موجود است.

با AJAX می‌توانید برنامه‌های کاربردی تحت Web بهتر، سریعتر و کاربرپسندانه‌تر بسازید.

AJAX برپایه JavaScript و درخواست‌های HTTP بنا شده است.

قبل از اینکه بتوانید به راحتی با AJAX کار کنید، لازم است که آشنایی متوسطی با HTML/XHTML و همچنین JavaScript داشته باشید.

AJAX یک تکنیک برای ساختن صفحات Web است به نحوی که این صفحات، بیشتر با کاربر به صورت فعل و انفعالی کار کنند و مشابه برنامه‌های تحت ویندوز عمل نمایند.

توسط AJAX، کدهای JavaScript شما می‌توانند مستقیماً با Server به وسیله شیء

XMLHttpRequest ارتباط برقرار کنند. توسط این شیء، می‌توانید داده‌ها را با یک Web Server تبادل نمایید، بدون آنکه نیاز به بارگذاری مجدد صفحه داشته باشید.

AJAX از انتقال داده‌های غیرهم‌زمان (درخواست‌های HTTP) بین مرورگر و Server استفاده می‌کند، که به صفحات Web اجازه می‌دهد به جای کل صفحه، تنها درخواست مقادیر کوچک اطلاعات از Server داشته باشند.

تکنیک AJAX برنامه‌های Internet را کوچک‌تر، سریع‌تر و کاربرپسندانه‌تر می‌کند.

AJAX یک تکنولوژی مرورگر است که به نرم‌افزار Web Server وابسته نیست.

AJAX بر پایه استانداردهای Web زیر بنا شده است :

JavaScript

XML

HTML

CSS

این استانداردهای Web که در AJAX مورد استفاده قرار گرفته‌اند، به خوبی در تمامی مرورگرهای مطرح از جمله +۵ Internet Explorer، ۱.۲ Safari، +۸ Opera، ۱.۰/ Firefox/Mozilla، ۷ NetScape و... تعریف و پشتیبانی شده‌اند. همچنین برنامه‌های AJAX وابسته به مرورگر و سیستم عامل نیستند و به خوبی بر روی سیستم‌های عامل مختلف از قبیل Windows، Linux، Linux، MacOS، FreeBSD، UNIX و... کار می‌کنند.

**هدف AJAX**، برنامه‌های کاربردی بهتر در محیط Internet است.

برنامه‌های کاربردی تحت Web مزایای متعددی نسبت به برنامه‌های کاربردی مجزا از شبکه (Desktop) دارند؛ می‌توانند دایره مخاطبین بیشتری را دربر بگیرند، نصب و پشتیبانی آنها ساده‌تر است، و همچنین راحت‌تر توسعه داده می‌شوند.

هرچند، برنامه‌های اینترنت همیشه به اندازه برنامه‌های سنتی رومیزی (Desktop)، کاربرپسندانه و غنی نیستند (به دلیل محدودیت منابع و سرعت).

به وسیله AJAX، برنامه‌های اینترنت را می‌توان غنی‌تر و کاربرپسندانه‌تر نمود.

درواقع هیچ چیز جدیدی برای یادگیری وجود ندارد. AJAX بر مبنای استانداردهای موجود طراحی شده است. این استانداردها توسط اغلب طراحان، سال‌های زیادی مورد استفاده قرار گرفته‌اند.

AJAX از **درخواست‌های HTTP** استفاده می‌کند :

در کدنویسی سنتی JavaScript، اگر می‌خواهید اطلاعات زیادی از یک بانک اطلاعاتی یا یک فایل بر روی Server دریافت کنید، یا اطلاعات کاربر را به یک Server بفرستید، باید یک فرم HTML طراحی کرده و داده‌ها را به روش GET یا POST برای Server ارسال کنید. کاربر باید بر روی دکمه Submit کلیک کند تا اطلاعات را ارسال کند، سپس منتظر پاسخ Server بماند، بعد یک صفحه جدید بارگذاری شده و نتایج را نشان می‌دهد.

این امر بدین دلیل رخ می‌دهد که Server یک صفحه جدید در هر بار ارسال اطلاعات کاربر توسط فشردن دکمه Submit ایجاد می‌کند. لذا اینگونه برنامه‌ها، معمولاً کندتر و کمتر با کاربر دوست هستند.

به کمک AJAX، کد JavaScript شما می‌تواند مستقیماً با Server ارتباط برقرار کند، و یک پاسخ را از Server دریافت نماید - بدون آنکه نیاز به بارگذاری مجدد صفحه باشد. کاربر در همان صفحه باقی می‌ماند، و بدون اینکه متوجه درخواست‌ها و تبادل اطلاعات در پس‌زمینه شود، بخشی از صفحه که اطلاعات آن تغییر کرده است - نه کل صفحه، به‌روزرسانی می‌شود.

## شیء XMLHttpRequest

با استفاده از شیء XMLHttpRequest، یک طراح Web می‌تواند یک صفحه را با داده‌هایی که از Server بعد از بارگذاری صفحه! دریافت کرده است، به‌روزرسانی کند.

AJAX در سال ۲۰۰۵ به پیشنهاد و توسط Google، جهانی شد :

پیشنهاد Google، استفاده از شیء XMLHttpRequest برای ساختن یک رابط تحت Web بسیار پویا بود: وقتی که شما در کادر جستجوی Google شروع به تایپ کردن کنید، یک کد JavaScript، حروف تایپ شده را به یک Server می‌فرستد و Server، فهرستی از موارد پیشنهادی را باز می‌گرداند.

شیء XMLHttpRequest در Internet Explorer از نسخه ۵ به بعد، Safari نسخه ۱٫۲، Mozilla نسخه ۱ و FireFox (نسخه ۱٫۵)، Opera از نسخه ۸ به بعد و NetScape نسخه ۷ پشتیبانی می‌شود.

## اولین برنامه AJAX :

برای درک اینکه AJAX چگونه کار می‌کند، ما یک برنامه AJAX کوچک ایجاد می‌کنیم. ابتدا، یک فرم استاندارد HTML طراحی می‌کنیم که دارای دو کادر متن است: Username و Time. کادر متن Username توسط کاربر و کادر متن Time توسط AJAX پر خواهد شد. نام فایل HTML را testAjax.htm قرار داده و کدهای زیر را در داخل آن می‌نویسیم (دقت کنید که فرم HTML زیر، دکمه Submit ندارد) :

```
<html>
<body>
<form name="myForm">
Name: <input type="text" name="username" />
Time: <input type="text" name="time" />
</form>
</body>
</html>
```

## سنگ بنای AJAX :

کلید اصلی و در واقع، سنگ بنای AJAX، شیء XMLHttpRequest است. مرورگرهای مختلف، روش‌های متفاوتی برای ایجاد شیء XMLHttpRequest دارند. Internet Explorer از یک شیء ActiveX استفاده می‌کند، در حالی که دیگر مرورگرها در شیء JavaScript داخلی (توکار) خود، به نام XMLHttpRequest بهره می‌برند.

برای ایجاد این شیء، و کار کردن با مرورگرهای مختلف، ما از یک دستور try و catch استفاده می‌کنیم. توضیحات بیشتر این دستور، در زبان JAVA موجود است. در اینجا به همین مقدار بسنده می‌کنیم که یک روش را امتحان می‌کند، اگر منجر به خطا شد، روش دیگر و همین‌طور الی آخر.

اجازه‌دهید فایل testAjax.htm خود را با این کد JavaScript که شیء XMLHttpRequest را ایجاد می‌کند، تکمیل کنیم :

```
<script type="text/javascript">
function ajaxFunction()

{
var xmlhttp;

try

{
// Firefox, Opera ۸.۰+, Safari
xmlhttp=new XMLHttpRequest();
}

catch (e)

{
// Internet Explorer
try

{
xmlhttp=new ActiveXObject("Msxml۲.XMLHTTP");
}

catch (e)

{
try

{
xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
}

catch (e)

{
alert("Your browser does not support AJAX!");
return false;
}
}
}
}
}
}
```



```

    }
  }
}
</script>

```

این دستورات را به بعد از خط `<body>` اضافه می‌کنیم.

**تشریح کد :**

خط اول، یک متغیر به نام `xmlHttp` برای نگه‌داری شیء `XMLHttpRequest` ایجاد می‌کند. سپس تلاش می‌کند تا یک شیء با دستور `new XMLHttpRequest()` ایجاد نماید. این دستور در مورد مرورگرهای غیر `Internet Explorer` کار می‌کند. اگر این دستور منجر به شکست شود، از دستور `new XMLHttpRequest("Msxml2.XMLHTTP")` که ویژه مرورگرهای نسخه ۶ به بعد است استفاده می‌کند. اگر مجدداً شکست خورد، از دستور `new XMLHttpRequest("Microsoft.XMLHTTP")` که ویژه مرورگرهای قبل از نسخه ۶ و بعد از نسخه ۵ است، استفاده می‌کند.

اگر هر سه روش فوق شکست خوردند، به معنی استفاده کاربر از یک مرورگر خیلی قدیمی است که از `AJAX` پشتیبانی نمی‌کند. لذا پیغام خطای مناسب برای کاربر تولید می‌شود.

**نکته :** کد مشخص‌کننده مرورگر فوق، طولانی و کمی پیچیده است. هرچند، این کد را می‌توانید هرگاه نیازمند ایجاد یک شیء `XMLHttpRequest` بودید، مورد استفاده قرار دهید. لذا کافی است آنرا `Copy` و در جای مورد نظر، `Paste` نمایید. کد فوق، با تمامی مرورگرهای محبوب از جمله `Internet Explorer`، `Opera`، `Firefox` و `Safari` سازگار است.

**خصوصیت `onreadystatechange` :**

قبل از اینکه داده‌ها را به سمت `Server` ارسال کنیم، باید سه خصوصیت مهم شیء `XMLHttpRequest` را توضیح دهیم. یکی از این خصوصیت‌ها، `onreadystatechange` است. بعد از درخواست از `Server`، به تابعی نیاز داریم که بتواند داده‌هایی را که از سمت `Server` بازگردانده می‌شوند، دریافت کند. خصوصیت

onreadystatechange محل نگهداری تابعی است که پاسخ Server را پردازش می‌کند. کد زیر، یک تابع خالی و همچنین خصوصیت onreadystatechange را به صورت هم‌زمان تعریف می‌کند:

```
xmlHttpRequest.onreadystatechange=function()  
  
{  
  
  // We are going to write some code here  
  
}
```

### خصوصیت readyState :

این خاصیت، وضعیت پاسخ Server را نگهداری می‌کند. هر زمان که readyState تغییر کند، تابع onreadystatechange فراخوانی و اجرا می‌شود. در اینجا، مقادیری که خصوصیت readyState قبول می‌کند را مشاهده می‌کنید:

وضعیت	شرح
۰	درخواست تنظیم نشده است.
۱	درخواست تنظیم شده است.
۲	درخواست ارسال شده است.
۳	درخواست در حال پردازش است.
۴	درخواست تکمیل شده است.

ما می‌خواهیم یک دستور If به تابع `onreadystatechange` اضافه کنیم تا بررسی کند که آیا درخواست ما تکمیل شده است یا نه (این بدان معنی است که می‌توانیم داده‌های خود را دریافت کنیم یا خیر)؟ به دستورات مربوطه دقت کنید :

```
xmlHttp.onreadystatechange=function()
{
if(xmlHttp.readyState==۴)
{
document.myForm.time.value=xmlHttp.responseText;
}
}
```

### ارسال درخواست به Server :

برای ارسال یک درخواست به Server، ما از متدهای `open()` و `send()` استفاده می‌کنیم. متد `open()` سه پارامتر ورودی دارد. اولین پارامتر، متد مورد استفاده برای ارسال درخواست را مشخص می‌کند (GET یا POST). دومین پارامتر بیانگر آدرس Script (کد) مورد نظر بر روی کامپیوتر Server است. سومین پارامتر مشخص کننده این امر است که درخواست باید به صورت غیرهمزمان پردازش شود (`true`) یا همزمان (`false`). متد `send()`، درخواست تنظیم شده را به Server ارسال می‌کند. اگر فرض کنیم که فایل HTML و ASP مورد نظر ما هر دو در یک پوشه مشترک قرار دارند، کد مورد نظر چیزی شبیه به این کد خواهد بود :

```
xmlHttp.open("GET","time.asp",true);
xmlHttp.send(null);
```

حال باید تصمیم بگیریم که تابع AJAX ما چه زمانی باید اجرا شود. در ادامه، ما به تابع اجازه می‌دهیم تا در «پشت صحنه»، وقتی که کاربر کلیدی را در کادر متن Username فشار دهد (یا نگاه دارد)، اجرا شود. لذا کد ایجادکننده کادر متن Username را به صورت زیر تغییر می‌دهیم :

```
Name: <input type="text" onkeydown="ajaxFunction();" name="username" />
```

### کد ASP سمت Server :

حال باید دستوری ایجاد کنیم که زمان جاری Server را نشان دهد. خصوصیت responseText (که قبلاً توضیح دادیم)، داده‌هایی را که از سمت Server بازگردانده می‌شود، ذخیره می‌کند. در اینجا ما زمان جاری را بازمی‌گردانیم. کد زیر را در فایل time.asp می‌نویسیم :

```
<%
```

```
response.expires=-۱
```

```
response.write(time)
```

```
%>
```

**نکته :** خصوصیت Expires مدت زمانی را (برحسب دقیقه) مشخص می‌کند که یک صفحه، بر روی مرورگر باید قبل از اینکه از بین برود، ذخیره گردد. اگر کاربر همان صفحه را قبل از اینکه از بین برود، باز کند، در آن صورت، نسخه ذخیره شده باز خواهد شد و صفحه جدید از Server درخواست نمی‌شود. دستور Response.Expires=-۱ بیانگر این مطلب است که صفحه، هرگز ذخیره نمی‌شود و همیشه باید از Server درخواست گردد (بدیهی است که اگر صفحه ذخیره گردد، در آن صورت، زمان قبلی که در صفحه قبلی ذخیره شده است، نشان داده خواهد شد، نه زمان جدید).

یک مثال عملی تر :

در این مثال، صفحه ای طراحی خواهیم کرد که با فشاردادن هر حرف، فهرستی از اسامی موجود در فایل اسامی را که محتوای کادر متن در آنها وجود دارد، نشان دهد و اگر کادر متن خالی باشد، چیزی نشان ندهد.

```
<html>
<head>
<script type="text/javascript">
function showHint(str)
{
if (str.length==0)
{
document.getElementById("txtHint").innerHTML="";
return;
}
xmlHttp=GetXmlHttpRequest()
if (xmlHttp==null)
{
alert ("Your browser does not support AJAX!");
return;
}
var url="gethint.asp";
url=url+"?q="+str;
url=url+"&sid="+Math.random();
xmlHttp.onreadystatechange=stateChanged;
```

```
xmlHttp.open("GET",url,true);
```

```
xmlHttp.send(null);
```

```
}
```

```
</script>
```

```
</head>
```

```
<body>
```

```
<form>
```

```
First Name: <input type="text" id="txt1" onkeyup="showHint(this.value)">
```

```
</form>
```

```
<p>Suggestions: <span id="txtHint"></span></p>
```

```
</body>
```

```
</html>
```

### کمی توضیح :

همان طور که می بینید، این صرفاً یک فرم HTML ساده با یک کادر متن به نام txt1 است. یک رویداد برای کادر متن تعریف می شود که در زمان رهاکردن یک کلید در داخل کادر متن، فراخوانی و اجرا می شود. دستور بعد، حاوی یک Span (محدوده) به نام txtHint است. Span به عنوان یک نگهدارنده برای داده هایی که از سمت Server دریافت خواهیم کرد، عمل خواهد نمود. وقتی که کاربر داده ها را وارد می کند، یک تابع به نام showHint() اجرا می شود. اجرای این تابع براساس رویداد onkeyup است. به بیان دیگر، هر بار که کاربر انگشت خود را از روی یک کلید از صفحه کلید بردارد، درحالی که کادر متن فعال باشد، تابع showHint فراخوانی می شود.

تابع، هر بار که یک کارکتر وارد کادر متن می شود، اجرا خواهد شد. اگر چیزی در داخل کادر متن باشد (str.length > 0)، تابع کارهای زیر را انجام می دهد :

۱- آدرس نام فایل حاوی اسامی را برای ارسال به Server تنظیم می کند.

۲- عبارت `?q=` را به همراه محتوای کادر متن به آدرس اضافه می کند (شبیه سازی متد GET).

۳- عبارت `&sid=` را به همراه یک عدد تصادفی به آدرس اضافه می کند تا یک آدرس ذخیره شده، بازنشود و هر بار، درخواست جدیدی تولید شود.

۴- شیء `XMLHTTP` را طوری تنظیم می کند که هر زمان خصوصیت `onreadystatechange` تغییر یافت، تابع `stateChanged` فراخوانی و اجرا شود.

۵- شیء `XMLHTTP` را با آدرس تولید شده، باز می کند.

۶- درخواست `HTTP` را برای `Server` ارسال می کند.

اگر کادر متن خالی باشد، تابع مربوطه، به سادگی محتوای `txtHint` را پاک می کند.

تشریح کد - تابع `GetXmlHttpRequest()` :

مثال فوق، تابعی را تحت عنوان `GetXmlHttpRequest()` فراخوانی می کند. هدف این تابع، رفع مشکل

ایجاد اشیاء مختلف `XMLHTTP` برای مرورگرهای مختلف است. در واقع چیز جدیدی نیست. همان

دستورات ایجاد شیء `XMLHttpRequest` است که در اینجا، تحت عنوان یک مجزا، نام گذاری شده است :

```
function GetXmlHttpRequest()
{
    var xmlhttp=null;
    try
    {
        // Firefox, Opera ۸.۰+, Safari
        xmlhttp=new XMLHttpRequest();
```

```

    }
catch (e)
{
    // Internet Explorer
    try
    {
        xmlHttp=new ActiveXObject("Msxml2.XMLHTTP");
    }
catch (e)
{
    xmlHttp=new ActiveXObject("Microsoft.XMLHTTP");
}
}
return xmlHttp;
}

```

تشریح کد - تابع stateChanged() :

تابع stateChanged() حاوی کد زیر است :

```

function stateChanged()
{
    if (xmlHttp.readyState==۴)

```



```

    {
document.getElementById("txtHint").innerHTML+xmlHttp.responseText;
    }
}

```

تابع stateChanged() هر بار که وضعیت شیء XMLHttpRequest تغییر کند، فراخوانی می‌شود. وقتی که وضعیت به حالت ۴ ("تکمیل شده") تغییر کند، محتوای نگهدارنده txtHint با متن پاسخ پر می‌شود.

### کد AJAX مربوط به فایل HTML :

در اینجا، کد فایل HTML را که می‌توانید با نام دلخواه ذخیره کنید، مشاهده می‌نمایید :

```

<html>
<head>
<script src="clienthint.js">
</script>
</head>
<body>
<form>
First Name:<input type="text" id="txt\\" onkeyup="showHint(this.value)">
</form>
<p>Suggestions: <span id="txtHint"></span></p>

```

```
</body>
```

```
</html>
```

## کد AJAX مربوط به فایل JavaScript :

در اینجا، کد فایل clienthint.js را مشاهده می‌کنید که در قسمت Head فایل HTML، فراخوانی می‌شود :

```
var xmlHttp
function showHint(str)
{
  if (str.length==0)
  {
    document.getElementById("txtHint").innerHTML="";
    return;
  }
  xmlHttp=GetXmlHttpRequest()
  if (xmlHttp==null)
  {
    alert ("Your browser does not support AJAX!");
    return;
  }
  var url="gethint.asp";
```

```
url=url+"?q="+str;
url=url+"&sid="+Math.random();
xmlHttp.onreadystatechange=stateChanged;
xmlHttp.open("GET",url,true);
xmlHttp.send(null);
}
function stateChanged()
{
    if (xmlHttp.readyState==4)
    {
        document.getElementById("txtHint").innerHTML=xmlHttp.responseText;
    }
}
```

```
function GetXmlHttpRequestObject()
{
    var xmlHttp=null;
    try
    {
        // Firefox, Opera 9.0+, Safari
```

```

xmlHttp=new XMLHttpRequest();
}
catch (e)
{
// Internet Explorer
try
{
xmlHttp=new ActiveXObject("Msxml2.XMLHTTP");
}
catch (e)
{
xmlHttp=new ActiveXObject("Microsoft.XMLHTTP");
}
}
return xmlHttp;
}

```

صفحه Server تکنیک AJAX - ASP و PHP :

هیچ چیزی در مورد AJAX تحت عنوان Server وجود ندارد. صفحات AJAX را می توان با هر Server اینترنت دلخواه، پیاده سازی نمود.

صفحه Server فراخوانی شده توسط JavaScript در مثال قبل، یک فایل ASP ساده به نام gethint.asp بود.

در زیر، ما دو مثال از کد صفحه Server با زبان های ASP و PHP ارائه می کنیم که می توانید از هر کدام که مایل باشید، به عنوان فایل حاوی اسامی، در مثال فوق استفاده کنید.

## مثال ASP تکنیک AJAX :

در اینجا، کد فایل `gethint.asp` را مشاهده می‌کنید که در فایل `clienthint.js`، فراخوانی می‌شود :

```
'get the q parameter from URL
q=ucase(request.querystring("q"))
'lookup all hints from array if length of q>0
if len(q)>0 then hint=""
    for i=1 to 30
        if q=ucase(mid(a(i),1,len(q))) then
            if hint="" then
                hint=a(i)
            else
                hint=hint & " , " & a(i)
            end if
        end if
    end if
next
end if
'Output "no suggestion" if no hint were found
'or output the correct values
if hint="" then
    response.write("no suggestion")
else
```

```
    response.write(hint)
end if
%>

<%
response.expires=-1

dim a(13) 'Fill up array with names

a(1)="Anna"
a(2)="Brittany"
a(3)="Cinderella"
a(4)="Diana"
a(5)="Eva"
a(6)="Fiona"
a(7)="Gunda"
a(8)="Hege"
a(9)="Inga"
a(10)="Johanna"
a(11)="Kitty"
a(12)="Linda"
a(13)="Nina"
```

a(۱۴)="Ophelia"

a(۱۵)="Petunia"

a(۱۶)="Amanda"

a(۱۷)="Raquel"

a(۱۸)="Cindy"

a(۱۹)="Doris"

a(۲۰)="Eve"

a(۲۱)="Evita"

a(۲۲)="Sunniva"

a(۲۳)="Tove"

a(۲۴)="Unni"

a(۲۵)="Violet"

a(۲۶)="Liza"

a(۲۷)="Elizabeth"

a(۲۸)="Ellen"

a(۲۹)="Wenche"

a(۳۰)="Vicky"

مثال PHP تکنیک AJAX :

در اینجا، کد فایل `gethint.php` را مشاهده می‌کنید که در فایل `clienthint.js`، با تغییر `gethint.asp` به `gethint.php`، فراخوانی می‌شود:

```
//get the q parameter from URL
$q=$_GET["q"];
//lookup all hints from array if length of q>•
if (strlen($q) > •)
{
    $hint="";
    for($i=•; $i<count($a); $i++)
    {
        if (strtolower($q)==strtolower(substr($a[$i],•,strlen($q))))
        {
            if ($hint=="")
            {
                $hint=$a[$i];
            }
            else
            {
                $hint=$hint." , ".$a[$i];
            }
        }
    }
}
```



```
}  
  
// Set output to "no suggestion" if no hint were found  
// or to the correct values  
if ($hint == "")  
{  
$response="no suggestion";  
}  
else  
{  
$response=$hint;  
}  
  
//output the response  
echo $response;  
?>
```

```
<?php  
header("Cache-Control: no-cache, must-revalidate");  
  
// Date in the past  
header("Expires: Mon, 26 Jul 1997 05:00:00 GMT");  
  
// Fill up array with names  
$a[]="Anna";  
$a[]="Brittany";
```

\$a[]="Cinderella";

\$a[]="Diana";

\$a[]="Eva";

\$a[]="Fiona";

\$a[]="Gunda";

\$a[]="Hege";

\$a[]="Inga";

\$a[]="Johanna";

\$a[]="Kitty";

\$a[]="Linda";

\$a[]="Nina";

\$a[]="Ophelia";

\$a[]="Petunia";

\$a[]="Amanda";

\$a[]="Raquel";

\$a[]="Cindy";

\$a[]="Doris";

\$a[]="Eve";

\$a[]="Evita";

\$a[]="Sunniva";

\$a[]="Tove";

\$a[]="Unni";

\$a[]="Violet";

\$a[]="Liza";

```
$a[]="Elizabeth";
```

```
$a[]="Ellen";
```

```
$a[]="Wenche";
```

```
$a[]="Vicky";
```

## AJAX و بانک‌های اطلاعاتی:

در مثال AJAX زیر، به شما نشان می‌دهیم که چگونه یک صفحه Web می‌تواند اطلاعات را از بانک اطلاعاتی با استفاده از تکنولوژی AJAX استخراج کند. یک فایل HTML با نام دلخواه و با کد زیر ایجاد می‌کنیم:

```
<html>
```

```
<head>
```

```
<script src="selectcustomer.js"></script>
```

```
</head>
```

```
<body>
```

```
<form>
```

Select a Customer:

```
<select name="customers" onchange="showCustomer(this.value)">
```

```
<option value="ALFKI">Alfreds Futterkiste
```

```
<option value="NORTS ">North/South
```

```
<option value="WOLZA">Wolski Zajazd
```

```
</select>
```

```
</form>
```

```
<p>
```

```
<div id="txtHint"><b>Customer info will be listed here.</b></div>
</p>
</body>
</html>
```

توضیح :

همان طور که می‌توانید مشاهده کنید، این فایل، یک فرم HTML ساده با یک لیست پایین‌افتادنی به نام customers است. در ادامه، یک تگ div به نام txtHint قرار دارد که برای نگهداری اطلاعات دریافت شده از Web Server به کار می‌رود.

وقتی که کاربر چیزی را انتخاب کند، تابع showCustomer() توسط رویداد onchange فراخوانی می‌شود و محتوای لیست customers برای آن ارسال می‌گردد. به بیان ساده‌تر، هر بار که کاربر مقدار لیست customers را تغییر دهد، تابع showCustomer فراخوانی می‌شود. کد JavaScript مربوطه (selectcustomer.js) را در ادامه می‌بینید.

: [AJAX JavaScript](#)

```
var xmlHttp
function showCustomer(str)
{
xmlHttp=GetXmlHttpRequest();
if (xmlHttp==null)
{
alert ("Your browser does not support AJAX!");
return;
}
```

```

    }
    var url="getcustomer.asp";
    url=url+"?q="+str;
    url=url+"&sid="+Math.random();
    xmlhttp.onreadystatechange=stateChanged;
    xmlhttp.open("GET",url,true);
    xmlhttp.send(null);
}
function stateChanged()
{
if (xmlhttp.readyState==4)
{
document.getElementById("txtHint").innerHTML=xmlhttp.responseText;
}
}

function GetXmlHttpRequestObject()
{
var xmlhttp=null;
try
{
// Firefox, Opera 9.0+, Safari
xmlhttp=new XMLHttpRequest();

```

```

}
catch (e)
{
// Internet Explorer
try
{
xmlHttp=new ActiveXObject("Msxml2.XMLHTTP");
}
catch (e)
{
xmlHttp=new ActiveXObject("Microsoft.XMLHTTP");
}
}
return xmlHttp;
}

```

صفحه Server مربوط به AJAX :

صفحه Server مربوطه که توسط کد JavaScript فراخوانی می شود، یک فایل ASP ساده به نام getcustomer.asp است. مجدداً یادآور می شویم که می توان به راحتی از فایل های PHP و دستورات آن زبان استفاده نمود و AJAX محدود به ASP یا هرگونه زبان Server دیگری نیست (northwind.db) یک بانک اطلاعاتی نمونه است که همراه با نرم افزار Microsoft Access نصب می شود).

کد فایل getcustomer.asp به صورت زیر است :

<%

```

response.expires=-1

sql="SELECT * FROM CUSTOMERS WHERE CUSTOMERID="
sql=sql & "" & request.querystring("q") & ""
set conn=Server.CreateObject("ADODB.Connection")
conn.Provider="Microsoft.Jet.OLEDB.4.0"
conn.Open(Server.MapPath("/db/northwind.mdb"))
set rs = Server.CreateObject("ADODB.recordset")
rs.Open sql, conn
response.write("<table>")
do until rs.EOF
    for each x in rs.Fields
        response.write("<tr><td><b>" & x.name & "</b></td>")
        response.write("<td>" & x.value & "</td></tr>")
    next
    rs.MoveNext
loop
response.write("</table>")
%>

```

## [AJAX و تعامل با XML](#) :

AJAX می تواند برای ارتباط فعل وانفعالی و پویا با یک فایل XML نیز مورد استفاده قرار گیرد. در مثال زیر، نشان می دهیم که چگونه می توان از یک صفحه Web برای استخراج اطلاعات از یک فایل XML به کمک تکنولوژی AJAX استفاده نمود. فایل HTML زیر را با نام دلخواه ایجاد کنید :

```
<html>
<head>
<script src="selected.js"></script>
</head>
<body>
<form>
Select a CD:
<select name="cds" onchange="showCD(this.value)">
<option value="Bob Dylan">Bob Dylan</option>
<option value="Bonnie Tyler">Bonnie Tyler</option>
<option value="Dolly Parton">Dolly Parton</option>
</select>
</form>
<p>
<div id="txtHint"><b>CD info will be listed here.</b></div>
</p>
</body>
</html>
```

توضیح :

همان طور که می‌توانید مشاهده کنید، این فایل، یک فرم HTML ساده با یک لیست پایین افتادنی به نام cds است. در ادامه، یک تگ div به نام txtHint قرار دارد که برای نگهداری اطلاعات دریافت شده از Web Server به کار می‌رود.



وقتی که کاربر چیزی را انتخاب کند، تابع `showCD()` توسط رویداد `onchange` فراخوانی می‌شود و محتوای لیست `cds` برای آن ارسال می‌گردد. به بیان ساده‌تر، هر بار که کاربر مقدار لیست `cds` را تغییر دهد، تابع `showCD` فراخوانی می‌شود.

کد JavaScript مربوطه (`selected.js`) را در ادامه می‌بینید.

: AJAX JavaScript

در اینجا، کد فایل `selected.js` را مشاهده می‌کنید که در قسمت `Head` فایل `HTML`، فراخوانی می‌شود :

```
var xmlHttp
function showCD(str)
{
xmlHttp=GetXmlHttpRequestObject();
if (xmlHttp==null)
{
alert ("Your browser does not support AJAX!");
return;
}
var url="getcd.asp";
url=url+"?q="+str;
url=url+"&sid="+Math.random();
xmlHttp.onreadystatechange=stateChanged;
xmlHttp.open("GET",url,true);
xmlHttp.send(null);
}
```

```
function stateChanged()
{
if (xmlHttp.readyState==4)
{
document.getElementById("txtHint").innerHTML=xmlHttp.responseText;
}
}
```

```
function GetXmlHttpRequestObject()
{
var xmlHttp=null;
try
{
// Firefox, Opera 9.+, Safari
xmlHttp=new XMLHttpRequest();
}
catch (e)
{
// Internet Explorer
try
{
xmlHttp=new ActiveXObject("Msxml2.XMLHTTP");
```

```

    }
catch (e)
{
    xmlHttp=new ActiveXObject("Microsoft.XMLHTTP");
}
}
return xmlHttp;
}

```

صفحه Server مربوط به AJAX :

صفحه Server مربوطه که توسط کد JavaScript فراخوانی می‌شود، یک فایل ASP ساده به نام `getcd.asp` است. باز هم یادآور می‌شویم که می‌توان به راحتی از فایل‌های PHP و دستورات آن زبان استفاده نمود و AJAX محدود به ASP یا هرگونه زبان Server دیگری نیست (یک `cd_catalog.xml` فایل XML نمونه است که می‌توانید از آدرس `http://www.w3schools.com/ajax/cd_catalog.xml` آنرا دریافت کنید).

کد فایل `getcd.asp` به صورت زیر است :

```

<%
response.expires=-1

q=request.querystring("q")
set xmlDoc=Server.CreateObject("Microsoft.XMLDOM")
xmlDoc.async="false"
xmlDoc.load(Server.MapPath("cd_catalog.xml"))
set nodes=xmlDoc.selectNodes("CATALOG/CD[ARTIST='" & q & "'")

```

```

for each x in nodes
  for each y in x.childnodes
    response.write("<b>" & y.nodename & ":</b> ")
    response.write(y.text)
    response.write("<br />")
  next
next
%>

```

## : ResponseXML

در حالی که `responseText`، پاسخ HTTP را به صورت یک رشته متنی بر می گرداند، دستور `responseXML`، پاسخ را به صورت XML باز می گرداند.

خصوصیت `ResponseXML` یک شیء سند XML باز می گرداند که می تواند توسط خاصیت ها و متدهای درخت گره `W3C DOM` مورد بررسی و پردازش قرار گیرد.

در مثال بعدی `AJAX`، از یک صفحه `Web` برای استخراج اطلاعات از یک بانک اطلاعاتی استفاده می کنیم، اما این بار، داده های استخراج شده از بانک اطلاعاتی، به صورت یک سند `XML` تبدیل شده و سپس، از `DOM` برای استخراج مقادیر جهت نمایش استفاده می کنیم.

## : فایل HTML

یک فایل `HTML` با نام دلخواه و با کد زیر ایجاد می کنیم :

```
<html>
```

```

<head>
<script src="selectcustomer_xml.js"></script>
</head>
<body>
<form action="">
Select a Customer:
<select name="customers" onchange="showCustomer(this.value)">
<option value="ALFKI">Alfreds Futterkiste</option>
<option value="NORTS ">North/South</option>
<option value="WOLZA">Wolski Zajazd</option>
</select>
</form>
<b><span id="companyname"></span></b><br/>
<span id="contactname"></span><br/>
<span id="address"></span>
<span id="city"></span><br/>
<span id="country"></span>
</body>
</html>

```

توضیح :

همان طور که می توانید مشاهده کنید، این فایل، یک فرم HTML ساده با یک لیست پایین افتادنی به نام customers است. در ادامه، یک تگ div به نام txtHint قرار دارد که برای نگهداری اطلاعات دریافت شده از Web Server به کار می رود.

وقتی که کاربر چیزی را انتخاب کند، تابع `showCustomer()` توسط رویداد `onchange` فراخوانی می‌شود و محتوای لیست `customers` برای آن ارسال می‌گردد. به بیان ساده‌تر، هر بار که کاربر مقدار لیست `customers` را تغییر دهد، تابع `showCustomer` فراخوانی می‌شود.

کد JavaScript مربوطه (`selectcustomer_xml.js`) را در ادامه می‌بینید.

: AJAX JavaScript

در اینجا، کد فایل `selectcustomer_xml.js` را مشاهده می‌کنید که در قسمت `Head` فایل `HTML` فراخوانی می‌شود:

```
var xmlHttp
function showCustomer(str)
{
xmlHttp=GetXmlHttpRequestObject();
if (xmlHttp==null)
{
alert ("Your browser does not support AJAX!");
return;
}
var url="getcustomer_xml.asp";
url=url+"?q="+str;
url=url+"&sid="+Math.random();
xmlHttp.onreadystatechange=stateChanged;
xmlHttp.open("GET",url,true);
xmlHttp.send(null);
```

```

}
function stateChanged()
{
if (xmlHttp.readyState==4)
{
var xmlDoc=xmlHttp.responseXML.documentElement;
document.getElementById("companyname").innerHTML=
xmlDoc.getElementsByTagName("compname")[0].childNodes[0].nodeValue;
document.getElementById("contactname").innerHTML=
xmlDoc.getElementsByTagName("contname")[0].childNodes[0].nodeValue;
document.getElementById("address").innerHTML=
xmlDoc.getElementsByTagName("address")[0].childNodes[0].nodeValue;
document.getElementById("city").innerHTML=
xmlDoc.getElementsByTagName("city")[0].childNodes[0].nodeValue;
document.getElementById("country").innerHTML=
xmlDoc.getElementsByTagName("country")[0].childNodes[0].nodeValue;
}
}

```

```
function GetXmlHttpRequestObject()
```

```
{
```

```
var xmlhttp=null;

try
{
// Firefox, Opera ۸.۰+, Safari

xmlhttp=new XMLHttpRequest();
}
catch (e)
{
// Internet Explorer

try
{
xmlhttp=new ActiveXObject("Msxml۲.XMLHTTP");
}
catch (e)
{
xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
}
}
return xmlhttp;
}
```

صفحه Server مربوط به AJAX :



توابع showCurtomer() و GetXmlHttpRequest() دقیقاً مشابه مثال‌های قبل هستند. تابع stateChanged() نیز به روشی مشابه در مثال‌های قبل به کار گرفته شده است. لیکن این بار، تعداد Spanها بیشتر شده و برای هر کدام، یک دستور ذکر می‌شود. در ادامه، این بار نتیجه را به عنوان یک سند XML (توسط responseXML) بازگردانده و از DOM برای استخراج مقادیری که می‌خواهیم نشان دهیم، استفاده خواهیم کرد.

```
<%  
response.expires=-1  
  
response.contentType="text/xml"  
sql="SELECT * FROM CUSTOMERS "  
sql=sql & " WHERE CUSTOMERID=" & request.querystring("q") & ""  
on error resume next  
set conn=Server.CreateObject("ADODB.Connection")  
conn.Provider="Microsoft.Jet.OLEDB.۴.۰"  
conn.Open(Server.MapPath("/db/northwind.mdb"))  
set rs=Server.CreateObject("ADODB.recordset")  
rs.Open sql, conn  
if err <> 0 then  
response.write(err.description)  
set rs=nothing  
set conn=nothing  
else  
response.write("<?xml version='۱.۰' encoding='ISO-۸۸۵۹-۱'?>")  
response.write("<company>")  
response.write("<compname>" &rs.fields("companyname")& "</compname>")
```

```
response.write("<contname>" &rs.fields("contactname")& "</contname>")
response.write("<address>" &rs.fields("address")& "</address>")
response.write("<city>" &rs.fields("city")& "</city>")
response.write("<country>" &rs.fields("country")& "</country>")
response.write("</company>")
end if
on error goto •
%>
```

توضیح :

به خط دوم در کد ASP فوق دقت کنید : `response.contentType="text/xml"`  
خصوصیت `ContentType`، نوع محتویات `HTTP` را در مورد شیء پاسخ (`Response`) مشخص می کند.  
نوع پیش فرض این خصوصیت، `"text/html"` است. این بار ما می خواهیم که نوع محتوا `XML` باشد. لذا از `"text/xml"` استفاده می کنیم.  
سپس داده ها را از بانک اطلاعاتی انتخاب کرده، و یک سند `XML` با این داده ها ایجاد می کنیم.