تابستان ۱۳۸٦

شروعی بر برنامه نویسی Ajax

تالیف و ترجمه:

محمدهادی قومنجانی

HadiGhomanjani@Gmail.com

محمد توکلی هشجین

M.Tavakoli.h@Gmail.com

فصل اول: مقدمه ای بر Ajax ۱
راه حل ۱
دیدگاه کاربران ۲
دیدگاه برنامه نویسان ۳
نگاهی کلی به مراحل استفاده از Ajax در برنامه های تحت وب ۵
چه کارهای را می توانید با Ajax انجام دهید ۶
مزایای Ajax ۹
چالش های فن آوری Ajax ۱۰

فصل دوم: جاوااسکریپت ۱۱
تگ <script> ۱۲
نوشتن اسکریپت در قسمت head ۱۳
نوشتن اسکریپت در قسمت body ۱۴
فایل خارجی جاوااسکریپت ۱۴
مدیریت رویداد ۱۵
مشاهده خطاهای جاوااسکریپت ۱۷
متغیرها و انواع آن ها ۱۷
تبدیل نوع ۱۹
عملگر ۲۰
دستورات کنترلی ۲۳
تابع ۲۴
شی چیست؟ ۲۷
تعریف DOM ۲۹
شی document ۳۵

فصل سوم : شروع کار با Ajax ۴۱
شروع با HTTP ۴۱
درخواست XMLHttp ۴۵
اولین برنامه ساده Ajax ۵۲
متد GET در XMLHTTP ۵۹
متد POST در XMLHTTP ۶۲

فصل چهارم: کار با Ajax در Asp.Net ۶۹
کنترل ها ۶۹
Ajax در ASP.Net ۶۹
ASP.NET AJAX ۸۵
معماری ASP.NET AJAX ۸۵
فریم ورک سرور ۸۸
مدل توسعه ی کلاینت محوری (Client-centric development model) ۸۹
مدل سرور محوری توسعه ۸۹
اهداف ASP.NET AJAX ۹۰
یک برنامه سمت سرور ۹۱
یک برنامه ساده بر محور کلاینت ۹۶

فصل پنجم: خطایابی برنامه های Ajax ۱۰۱
FireBug ۱۰۱
دیباگ سمت سرور ۱۲۳
استفاده از پنجره Watch ۱۳۱
استفاده از پنجره Command ۱۳۳

فصل ششم: برنامه ChatRoom ۱۳۵
جداول موجود در برنامه ChatRoom ۱۳۵
چگونگی پیاده سازی ChatRoom ۱۳۷
فصل هفتم: برنامه BookStore ۱۵۵
پیوست: طریقه کار با نرم افزار EsayPHP ۱۹۱

فصل اول: مقدمه ای بر Ajax

زمانیکه نرم افزارهای تحت وب (Web Applications) به دنیای نرم افزار وارد شدند، موجب نگرشی جدید در تولید نرم افزار شد. ویژگی های که موجب ایجاد این رویکرد جدید شد عبارتند از:

- نرم افزارهای تحت وب برای اجرا نیاز به سیستم عامل خاصی ندارند.

تا آن روز اجرای یک برنامه بر روی سیستم عامل های متفاوت مساله بزرگی بود ولی نرم افزارهای تحت وب این مشکل را نداشتند.

- نیاز به نصب برنامه های دیگری برای اجرا ندارند، تنها با یک مرورگر صفحات وب می توان از نرم افزار، بر روی هر سیستمی استفاده کرد.
- نیاز به امکانات خاص سخت افزاری ندارند و فضایی از سیستم کاربران را نیز اشغال نمی کنند.
- براحتی بر روی انواع شبکه های محلی و شبکه اینترنت قابل اجرا هستند.

اما با این وجود برنامه های وب(Web Applications)نتوانستند این جایگاه را استحکام بخشند، اما چرا ... ؟

با توجه به اینکه ماهیت اینترنت بصورت synchronous request / response می باشد محتوای صفحات وب دائما در حال تغییر (refresh) می باشد، در واقع محیط کار کاربر در حال تغییر است در نتیجه کاربران تمایلی به این گونه از نرم افزارها ندارند.

معمولا کاربران نرم افزارها، علاقه ی ندارند که با کلیک کردن یک دکمه زمان انتظارشان برای انجام عملیاتی زیاد باشد. این مساله در نرم افزارهای تحت وب نمود بیشتری دارد. زمانیکه از برنامه های وب استفاده می کنید احتمالا این مساله برای شما خسته کننده است که با هر کلیک در صفحه شما باید منتظر نمایش صفحه ی دیگر یا نمایش مجدد همان صفحه با تغییراتی شوید. این مساله مهمترین موضوعی بود که موجب کم فروغی Web Application در برابر Desktop Application می شد. از آن زمان برنامه های تحت وب معروف به word wide wait شدند.

راه حل

یک راه حل ساده برای این مساله این است که افراد در یک صفحه باشند و بتوانند عملیات مورد نظر خود را در همان صفحه انجام دهند. بطور مثال در یک وب سایت خرید بتوان با کلیک کردن بر روی هر یک از کالاهای مورد نظر آن را به سبد خرید خود اضافه کرد، بدون اینکه نیاز به refresh شدن صفحه باشد.

این ایده همان چیزی است که بعنوان Ajax شناخته شده است.

Ajax تکنولوژی است که این امکان را فراهم می آورد که Web Application مانند Desktop Application عمل کند. همان طور که گفته شد، اینکه کاربر، بار شدن (load) صفحه جدید را در

مرورگر خود می بیند مشکل بزرگی در صفحات وب است، Ajax توانایی دریافت اطلاعات از سمت سرور را بدون نیاز به بار شدن مجدد (reload) صفحه یا بارشدن صفحه جدید را داراست.

با Ajax برنامه های تحت وب برای کاربران مانند نرم افزارهای دیگر می باشند. Ajax این توانایی را به برنامه های وب می دهد که عملیات ارسال اطلاعات و دریافت نتایج را در پشت صحنه انجام دهد، اطلاعاتی که نیاز دارد را از سمت سرور دریافت و اطلاعات مد نظر را نمایش دهد.

برای درک بهتری از چگونگی کارکرد Ajax، آن را از دو دیدگاه بررسی می کنیم:

۱. دیدگاه کاربران
۲. دیدگاه برنامه نویسان

دیدگاه کاربران

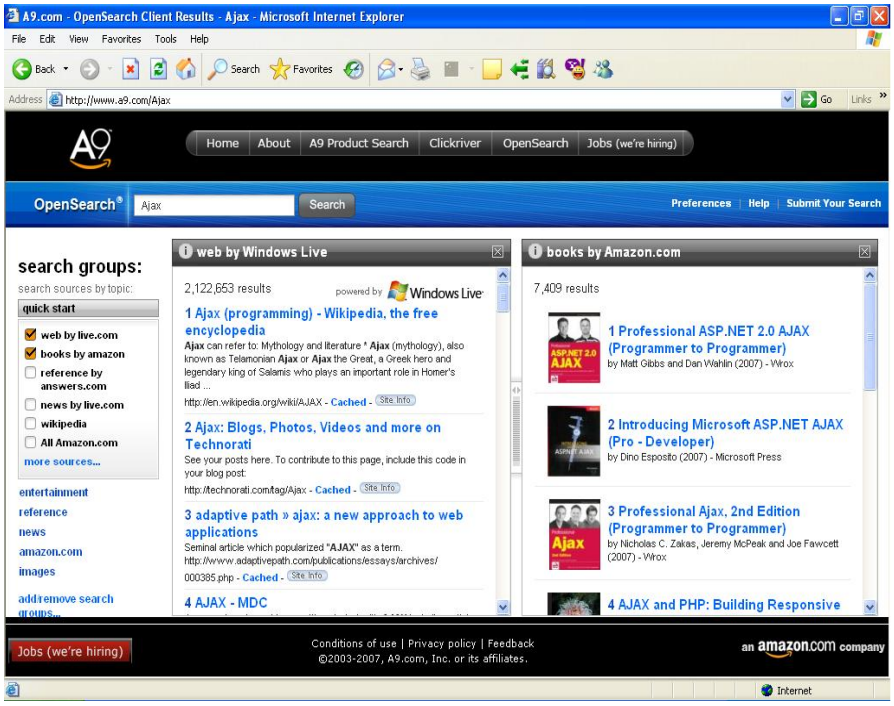
برای اینکه بهتر بدانید چگونه Ajax موجب می شود یک Web Application مانند یک Desktop Application عمل کند، این قسمت را با یک مثال توضیح می دهیم.

یک وب سایت جستجو را در نظر بگیرید. این وب سایت دارای یک textbox برای تایپ کردن عبارت مورد نظر کاربر و یک دکمه برای آغاز عملیات جستجو می باشد. هنگامی که کاربر یک عبارتی را نوشته و دکمه جستجو را کلیک می کند صفحه برای نمایش نتایج جستجو reload می شود، این در حالی است که در برنامه های Desktop Applications نتایج جستجو در همان فرم جستجو، بدون تغییر فرم و با کلیک دکمه جستجو نمایش داده می شود.

حال با تکنولوژی Ajax می توان صفحه جستجوی وب سایت را به گونه ای طراحی کرد تا هنگامی که عبارت مورد نظر نوشته و دکمه جستجو کلیک شود نتایج جستجو در همان صفحه نمایش داده شود. (مانند یک فرم جستجو در Desktop Application)

برای نمونه می توان وب سایت جستجوی http://www.A9.com را مثال زد.

این سایت جستجو متعلق به سایت Amazon.com می باشد. از ویژگی های این سایت می توان، جستجو با تکنولوژی Ajax و قابلیت جستجوی همزمان زمینه های مختلف، را نام برد.



شکل ۱-۱ وب سایت www.a9.com

دیدگاه برنامه نویسان

در مقاله ی که با موضوع Ajax :A New Approch to Web Applications

که در آدرس زیر موجود می باشد

www.adaptivepath.com/publications/essays/archives/000385.php

از jesse james garrett، کسی که اولین بار Ajax را معرفی کرد، آمده است:

" برنامه نویسان صفحات وب فکر می کنند بسیاری از قابلیت های Desktop Application در دسترس آن ها نیست و بین آن ها فاصله زیادی است، ولی Ajax این فاصله را کمتر می کند."

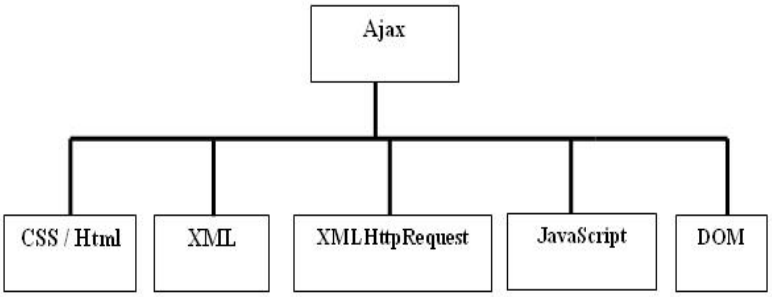
Ajax از **A**synchronous **J**avaScript **A**nd **X**ml گرفته شده است و تکنولوژی است که از قسمت های زیر تشکیل شده است:

- نمایش اطلاعات در مرورگرها در قالب HTML و CSS.
- ذخیره سازی اطلاعات در قالب Text یا XML که از سمت سرور دریافت شده است.
- استفاده از داده های دریافت شده از سمت سرور در پشت صحنه با استفاده از شی XMLHttpRequest در مرورگرها (این شی قابلیت اتصال به سرور را از طریق پروتکل HTTP داراست).
- JavaScript

جاوااسکریپت بخش اصلی Ajax است ولی Ajax مختص به جاوااسکریپت نیست. اخیرا framework های جدیدی برای Ajax ارائه شده است یکی از آن ها Atlas و نسخه جدیدتر آن Asp.Net Ajax است که در .Net کاربرد دارد.

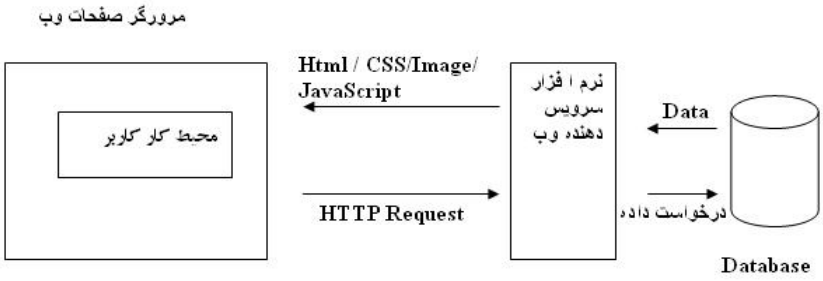
- **D**ocument **O**bject **M**odel (DOM)

DOM یک مدل که نشان دهنده این است که یک صفحه وب مانند یک مجموعه از اشیا به هم مرتبط می باشد که این اشیا بطور پویا قابل تغییر می باشند حتی اگر آن صفحه توسط کاربر دانلود شده باشد.



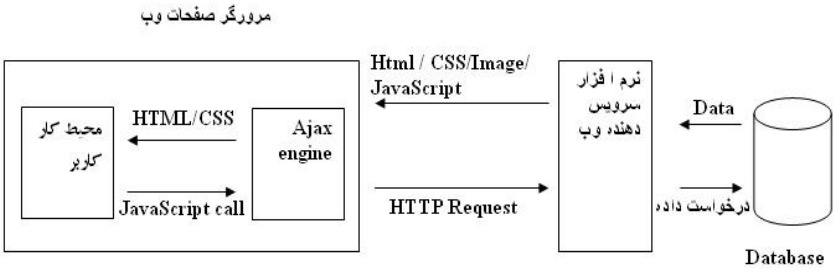
شکل۲-۱ بخش های تشکیل دهنده Ajax

برعکس برنامه های وب(Web Application) معمول، که خود برنامه ها وظیفه ی ارسال درخواست و پردازش نتایج درخواست از وب سرور را بر عهده دارند در Ajax یک لایه ی این عملیات را انجام می دهد که به آن موتور Ajax می گویند. موتور Ajax بطور کلی یک تابعی از جاوااسکریپت است که زمانیکه بخواهیم داده ها را از سمت سرور دریافت کنیم فراخوانی می شود. برعکس صفحات معمول وب که هر لینک یا دکمه یک صفحه جدید را نمایش می داد در اینجا هر لینک یا دکمه فراخوانی کننده موتورAjax می باشد. وقتی که موتور Ajax پاسخی را از سرور دریافت می کند وارد عمل می شود و با استفاده از داده های بازگشتی، محیط و صفحه اجرا شده را تغییر می دهد. شکل ۳-۱ نمای از عملکرد برنامه های معمول وب و شکل ۴-۱ نمای از برنامه های Ajax می باشد شما با مشاهده این شکل ها به تفاوت بین عملکرد این برنامه ها پی می برید.



شکل ۳-۱ نمای از عملکرد برنامه های معمول وب

همان طور که در شکل ۳-۱ مشاهده می نماید هر زمان که درخواستی از سمت کاربر صورت پذیرد اطلاعات درخواست، بصورت مسقیم به سمت سرور ارسال و در آن جا بعد از پردازش و درصورت نیاز، ارتباط با بانک اطلاعاتی در قالب های Html/CSS/Image/JavaScript به سمت کلاینت ارسال می شود.



شکل ۴-۱ نمای از عملکرد برنامه های Ajax

همان طور که در شکل شماره ۴-۱ مشاهده می کنید، با ایجاد یک رویداد در محیط کاربر (می تواند توسط کاربر یا زمان سیستم ایجاد شود) یک تابع از کدهای جاوااسکریپت فراخوانی می شود، که در واقع فراخوان موتور Ajax می باشد. موتور Ajax درخواست را به سمت سرور ارسال می کند. در سمت سرور براساس درخواست دریافت شده داده ها از بانک اطلاعاتی موجود در سرور بازیابی می شوند و به نرم افزار سرویس دهنده وب در سرور تحویل داده می شود این نرم افزار این داده ها را در قالب های Html /CSS /Image/JavaScript گنجانده و برای کلاینت ارسال می کند.

نگاهی کلی به مراحل استفاده از Ajax در برنامه های تحت وب

- نوشتن کد javascript برای دریافت داده های مورد نیاز از سمت سرور.
- استفاده از یک شی به نام XMLHttpRequest برای ارسال درخواست به سمت سرور که از طریق کد جاوااسکریپت انجام می گیرد. جاوااسکریپت دارای ویژگی است که اجرای عملیات را در مرورگر برای رسیدن داده ها از سمت سرور، به تعویق نمی اندازد و در پشت صحنه ی اجرا منتظر پاسخ از سرور می ماند. به این ویژگی asynchronous data retrieval می گویند.
- داده های دریافت شده از سمت سرور در قالب XML ذخیره می شوند.

کد های javascript در مرورگرها قابلیت خواندن و نوشتن داده ها در این قالب را بسرعت دارند.

بطور کلی Ajax از جاوااسکریپت و شی XMLHttpRequest برای ارتباط با سرور بدون Refresh شدن صفحه و از قالب XML برای مدیریت داده های دریافتی از سمت سرور استفاده می کند.

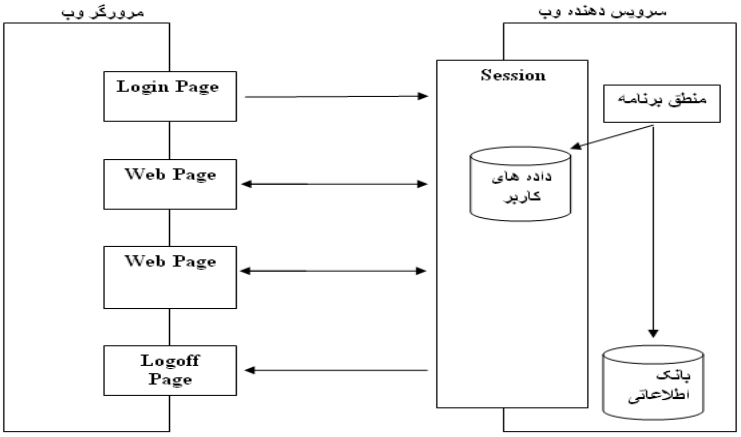
در فصل های آتی به چگونگی کارکرد این قسمت ها با یکدیگر بیشتر می پردازیم.

احتمالا از مطالب بالا نکاتی مفیدی در مورد Ajax فراگرفتید اکنون برای اینکه درک بهتری از تفاوت برنامه های Ajax با برنامه های معمول وب داشته باشید به شکل های ۵-۱ و ۶-۱ دقت فرمایید.

همان طور که در شکل ۵-۱ مشاهده می کنید در هنگام اتصال به سرور در سمت سرور یک session برای کاربر ایجاد شده و تا زمان قطع ارتباط موجود می باشد. زمانیکه کاربر درخواست جدیدی را به سمت سرور ارسال می کند یک وقفه ی در کار کاربر رخ می دهد و کاربر باید منتظر بماند تا صفحه دیگر از سمت سرور ارسال شود.

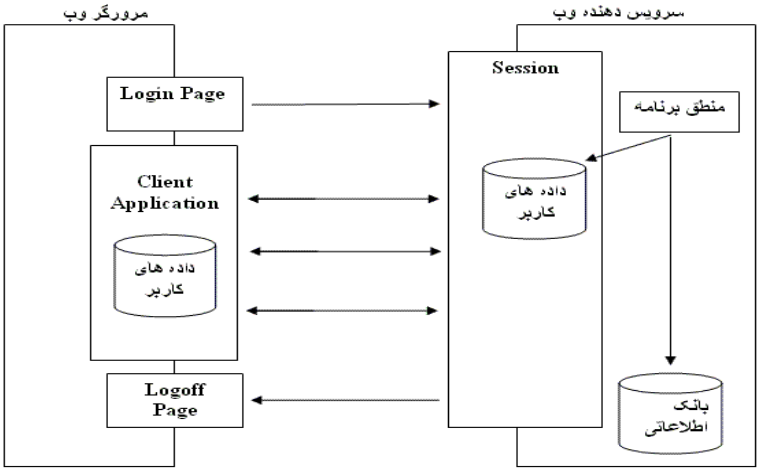
در شکل ۶-۱، هر زمان کاربر درخواستی را به سمت سرور ارسال می کند، در برنامه اجرا شده هیچ خللی ایجاد نمی شود وکاربر می تواند همزمان از برنامه اجرا شده استفاده نماید.

بخش های از Ajax که مطرح شد، قسمت های هستند که در سمت کلاینت قرار دارند اما یک قسمت مهم در Ajax وجود دارد که در سمت سرور می باشد. برای بازیابی داده ها در سرور نیاز به کدهای در سمت سرور می باشد مانند PHP، ASP و ... البته این بخش برای برنامه های معمول وب نیز لازم است.



شکل ۵-۱ نمای از توالی فعالیت ها در برنامه های معمول وب

.



شکل ۶-۱ نمای از توالی فعالیت ها در برنامه های Ajax

چه کارهای را می توانید با Ajax انجام دهید

تکنولوژی Ajax تقریبا از سال ۱۹۹۸ در نرم افزارهای مانند Microsoft Outlook Web Access وجود داشت، اما این تکنولوژی بطور کامل تا سال ۲۰۰۵ که در برنامه های نظیر Google suggest و Google maps استفاده گردید، شناخته نشد.

از جمله مواردی که می توان با تکنولوژی Ajax در صفحات وب پیاده سازی شوند عبارتند از:

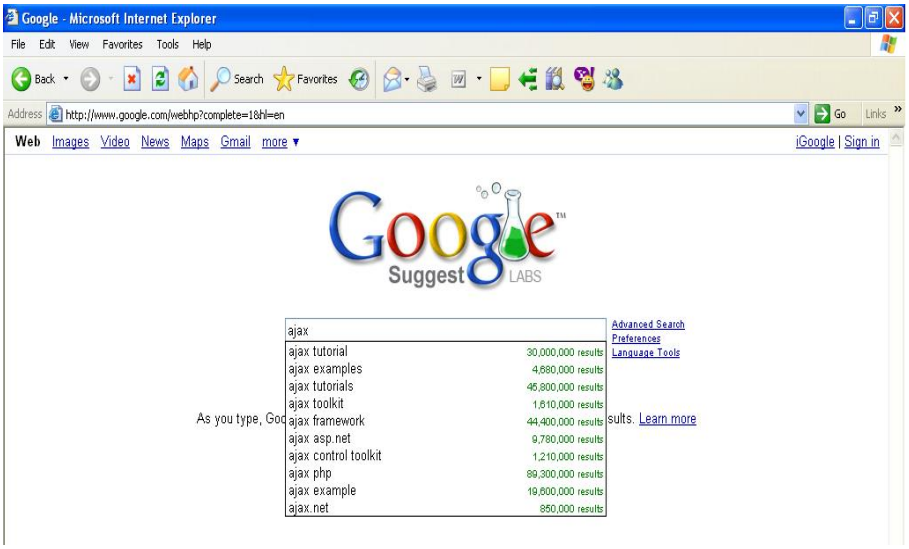
۱- جستجوی همزمان

همزمان با اینکه شما عبارت مورد نظر برای جستجو را تایپ می کنید نتایج مرتبط با آن بخش از عبارت نوشته شده نمایش داده شود.

برای مثال سایت:

http://www.google.com/webhp?complete=1&hl=en

در این سایت زمانیکه شما عبارتی را می نویسید همزمان نتایج جستجو در یک منوی کشویی نمایش داده می شود.



شکل ۷-۱ نمای از سایت Google Suggest

۲- دریافت جواب با autoComplete (یک فرهنگ لغت)

به برنامه های مانند فرهنگ لغات، برنامه های live Serach یا autoComplete گفته می شود. این برنامه ها سعی می کنند آن کلمه ی که شما می نویسید را حدس بزنند. به همین خاطر یک لیستی از کلمات شبیه به آن را از سرور دریافت و نمایش می دهند.

نمونه ی از این برنامه ها در وب سایت زیر می باشد.

www.papermountain.org/demos/live

Chat-۳

به دلیل اینکه Ajax نسبت به صفحات وبی که refresh می شوند برتری دارد، انتخاب مناسبی برای برنامه های پیام رسانی فوری (Chat) وب می باشد.

مثالی از این را در سایت زیر مشاهده می کنید.

www.plasticshore.com/project/chat

در این گونه برنامه ها شما متن خود را تایپ کرده و دکمه ارسال را کلیک می کنید، پیام به سرور ارسال می شود و بعد برای مخاطب ارسال می شود بدون اینکه صفحه شما و مخاطب refresh شود.

مثالی دیگر را می توانید در سایت زیر مشاهده نماید.

http://treehouse.ofb.net/chat/?long=en

همچنین radiusIM یک نمونه دیگر از سایت های chat است که درطراحی آن از google Map نیز استفاده شده است.

شکل ۸-۱ نمای از سایت radiusIM.com

٤- جابه جا کردن اشیا در صفحه با Ajax

بطور مثال در یک وب سایت خرید شما با Drag-Drop کردن کالای مورد نیاز خود به داخل سبد خرید، آن را خریداری می کنید و اطلاعات خرید به سبد خرید شما اضافه می شود.

٥- بازی با Ajax

یک بازی مانند شطرنج را در صفحه وب در نظر بگیرید، اگر هنگامی که شما مهره ی را حرکت می دهید صفحه برای ارسال موقعیت جدید مهره به سرور دوبار بارگذاری شود این بازی خیلی خسته کننده می شود. حال اگر این بازی با Ajax باشد شما می توانید با سرور براحتی و بدون بارگذاری مجدد صفحه، بازی کنید.

۶- دریافت فوری نتیجه (sign-in , sign-up)

یکی از بخش های که اغلب صفحات وب دارند، بخش login می باشد. زمانیکه کاربر اطلاعات خود را وارد کند و دکمه login را کلیک کند در صورتیکه اطلاعات صحیح نباشد صفحه refresh می شود و بعد پیغام نادرست بودن اطلاعات کاربری نمایش داده می شود. حال اینکه با Ajax می توان در همان صفحه در صورت نادرست بودن اطلاعات کاربری، پیغام مربوطه را نمایش داد و دیگر نیازی به بارگذاری مجدد صفحه نیست.

نمونه دیگر، چک کردن تکراری نبودن نام کاربری (UserName) در صفحات sign-up است.

در صفحات معمول وب شما بعد از پرکردن اطلاعات ثبت نام خود و زدن دکمه تایید اطلاعات خود را ارسال می کنید، حجم بالای از اطلاعات ارسال می شود، در سرور نام کاربری چک می شود و در صورت تکراری بودن نام کاربری، اطلاعات کاربری دوباره برگشت داده شود. حجم تبادل داده ها بالاست به همین خاطر عملیات بسیار وقت گیر است در صورتیکه با Ajax تنها کافیست اطلاعات نام کاربری به سمت سرور ارسال و آن جا چک می شود در صورتیکه اطلاعات تکراری باشد یک پیغام کوتاه خطا ارسال می شود.

۷- ایجاد منو با Ajax

در نظر بگیرید یک منوی کشوئی که هر ردیف آن موضوعات خاصی از جدول های موجود در سرور می باشد. مثلا در سرور یک جدول اطلاعات کتاب داریم که دارای موضوعاتی مشخص است از قبیل: کامپیوتر – زبان – روانشناسی و... می خواهیم زمانیکه موس بر روی منو می رود، اطلاعات موضوعات از جدول ها دریافت و بطور مجزا در ردیف های این منو قرار گیرد. شاید این موضوع خیلی برای شما ساده باشد و این طور تصور کنید که چه نیازی به این عملیات است در صورتیکه می توان در همان ابتدای بار شدن سایت اطلاعات بطور معمول خوانده شود و نمایش داده شود. اما توجه به این نکته لازم است که اگر تغییراتی در این جدول ها همزمان اعمال شود، مثلا موضوع جدیدی اضافه شود، شما در منو آن موضوع را نخواهید دید تا اینکه صفحه را دوباره بارگذاری کنید.

مزایای Ajax

طی سال های اخیر جهت صنعت نرم افزار (در زمینه ی وب) به سوی تولید نرم افزارهای است که مستقل از سیستم عامل و مرورگر باشند. نگاهی به سیر تحولات زبان های برنامه نویسی مانند PHP و ASP از طرفی و کاهش اقبال برنامه نویسان به فناوری های همچون ActiveX و Java Applet در سمت کلاینت از طرف دیگر، تایید کننده این مطلب است.

یکی دیگر از دلایل مهم توجه دنیای نرم افزار به Ajax این است که همچون فناوری های مانند Macromedia Flash نیازمند نصب هیچ نرم افزار اضافی بر روی مرورگر نیست و تنها اتکای آن به XML است و این یک مزیت است زیرا XML انعطاف پذیری زیادی دارد و هم اکنون در مقیاس گسترده ای در نرم افزارهای تحت وب مورد استفاده قرار می گیرد.

برنامه های Ajax به علت تبادل حجم داده کم بین سرور و کلاینت ترافیک شبکه را کاهش می دهند و سرعت اجرا را بالا می برند.

نکته مهم دیگر این است که در مورد Ajax شما نیاز به یادگیری زبان برنامه نویسی جدیدی نیستید.

چالش های فن آوری Ajax

برنامه نویسی صفحات به سبک Ajax دارای چالش های متعددی است :

- عناصر موجود در صفحات وب می بایست متناسب با شرایط هر مرورگر برنامه نویسی گردند، چراکه هر مرورگر یک نسخه متفاوت از DOM و DHTML را ارائه می نمایند(هر چند این تفاوت ها اندک باشد) .
- برنامه نویسی سمت کلاینت صرفا با استفاده از جاوااسکریپت انجام می شود. پیاده سازی برخی از بخش های Ajax می تواند برای پیاده کنندگان بسیار پیچیده باشد و نیازمند دانش بالائی در خصوص استفاده از جاوااسکریت باشد.
- جاوا اسکریپت، ویژگی ها و امکانات مورد نیاز پیاده کنندگان برنامه های دات نت را تامین نمی نماید (نظیر یک رویکرد شی گراء کامل) . علاوه بر این، در این فن آوری از کتابخانه ای نظیر آنچه در پلت فرم دات نت ارائه شده است، استفاده نمی گردد و برنامه نویسان می بایست تمامی برنامه را از ابتدا کد نمایند.
- جاوااسکریپت و پیاده سازی سمت کلاینت، عموما بخوبی در محیط های یکپارچه توسعه (IDEs) حمایت نمی گردند.

بعنوان سخن آخر:Ajax تکنولوژی نیست که جدید کشف شده باشد، شی XMLHttpRequest از زمان IE5 عرضه شده بود ولی چون دیگر مرورگرها از آن پشتیبانی نمی کردند، Ajax ناشناخته مانده بود.

فصل دوم: جاوااسکریپت

در اوایل برنامه های وب بیشتر جهت نمایش و ارائه مطالب بودند اما بعد از گسترش وب طراحان می خواستند بتوانند بر روی برنامه های وب مانند برنامه های دیگر کنترل داشته باشند. اما یک مشکل وجود داشت و آن هم این که در زبان Html دستورات کنترلی و حلقه های تکرار وجود نداشت. برای حل این مشکل زبانی به نام JavaScript عرضه شد.

زبان JavaScript محصولی مشترک از دو شرکت Sun Microsystem و Netscape Communications می باشد. جاوااسکریپت یک زبان مفسری است و مفسرهای دستوری آن مرورگرهای وبی هستند که از جاوااسکریپت پشتیبانی می کنند. (مفسر برنامه ی است که کدهای نوشته شده توسط برنامه نویس را براساس گرامر همان زبان برنامه نویسی ترجمه کرده و دستورات را اجرا می کند.) یعنی وقتی کدهای نوشته شده توسط جاوااسکریپت توسط مرورگر خوانده می شود، این کدها توسط مفسر جاوااسکریپتی که در آن تعبیه شده، ترجمه می گردد و حاصل این ترجمه به کاربر ارائه می شود.

جاوااسکریپت همان طورکه از نامش پیداست یک زبان اسکریپتی است. یعنی برنامه هایی که توسط آن نوشته می شوند متن ساده هستند (text only documents) و توسط هر ویرایشگری که بتواند متن ساده ایجاد کند قابل ویرایش و مشاهده هستند. متداول ترین و ساده ترین آن ها، ویرایشگر NotePad است که در تمامی نسخه های ویندوز وجود دارد.

جاوااسکریپت دارای دو بخش اساسی می باشد:

- کد جاوااسکریپت در سمت سرور.
- کد جاوااسکریپت در سمت کلاینت.

کدهای جاوااسکریپت در سمت کلاینت بیشتر استفاده می شوند اما چرا...؟

شما با کد جاوااسکریپت براحتی می توانید بر روی هر شی داخل صفحه وب کنترل داشته باشید. می توانید بسیاری از عملیات اجرایی برنامه خود در سمت کلاینت اجرا کنید در این صورت هم سرعت اجرای برنامه های وب شما بیشتر شده و هم از تبادل اطلاعات بیهوده مابین کلاینت و سرور جلوگیری می کنید.

بطور مثال می توانید بررسی صحت گرامری اطلاعاتی که کاربر می خواهد برای شما ارسال کند را در کلاینت چک کنید. (اگر اطلاعات عددی باید ارسال شود دیگر اطلاعاتی با فرمت رشته ارسال نشود.) قدرت جاوااسکریپت به نگاه رویدادی (Event) این زبان است. این خصیصه موجب تعامل بیشتر بین برنامه وکاربر می شود. به ازای هر عملیاتی که کاربر در برنامه وب انجام دهد (فشردن دکمه، حرکت موس، تایپ متن و ....) برنامه می تواند عکس العمل نشان دهد وعملیات خاصی را اجرا کند. برخی دیگر از خصوصیت های این زبان عبارتند از:

- نسبت به حروف بزرگ و کوچک حساس است.
- مجموعه دستورات یک تابع باید داخل بلوک باشد. یک بلوک با } شروع و به { خاتمه می یابد.

تگ <script>

جاوااسکریپت بعنوان بخشی از زبان HTML می باشد و در HTML برای نوشتن کدهای جاوااسکریپت بخشی مشخص تعیین شده است. زمانیکه می خواهید کدهای جاوااسکریپت را بنویسید باید آن ها را در میان تگ <script> قرار دهید.

```
<script>
  //javaScript Source Code comes here...
</script>
```

تگ <script> به مرورگر اعلام می کند تا مفسر جاوااسکریپت را برای اجرای دستورات فعال کند و تگ </script> اعلام می دارد که دستورات جاوااسکریپت خاتمه یافته است. تگ <script> دارای خاصیت های است که به بررسی آن ها می پردازیم:

خاصیت language

خاصیت language برای مشخص کردن نوع زبان اسکریپتی مورد استفاده در صفحه وب می باشد که مقدار آن را برای جاوااسکریپت باید برابر با javascript قرار دهید.

```
<script language="javascript">
</script>
```

برای مقدار javascript می توانید نسخه آن را هم تعیین کنید. جاوااسکریپت مانند HTML دارای نسخه های مختلفی است که هر کدام آن ها با قابلیتهای جدیدی در وب توسعه یافتند. نسخه های این زبان عبارت از 1.0، 1.1، 1.2، 1.3، 1.4 و 1.5 است. مرورگر IE از نسخه های 1.4 و 1.5 این زبان پشتیبانی نمی کند.

خاصیت type

تگ <script> خاصیت type هم دارد که برای جاوااسکریپت باید آن را برابر با text/javascript قرار دهید. نکته قابل توجه این است که اگر این خاصیت را به تگ<script> اضافه کنید، مرورگر اینترنت اکسپلورر نسخه 1.4 و 1.5 این زبان را اجرا می کند. از یک ویرایشگر متن مانند برنامه Notepad استفاده کرده و کد زیر را در آن وارد کنید:

```
<html>
<head>
   <title> My JavaScript </title>
</head>

<body>
<script language="javascript1.5" type="text/javascript">
     document.write("Hello designers and developers.")
</script>
```

```
</body>
</html>
```

اکنون این فایل را با پسوند html ذخیره کنید و سپس آن را در مرورگر اینترنت اکسپلورر اجرا کنید که موجب نمایش جمله .Hello designers and developers در صفحه می شود. حال خاصیت type را حذف کنید و دوباره فایل را ذخیره و مشاهده کنید، هیچ متنی در صفحه مشاهده نمی شود که با تغییر نسخه 1.5 به 1.3 آن متن دوباره نمایان خواهد شد. نوشتن نسخه جاوااسکریپت اجباری نیست یعنی می توانید آن را ننویسید. همچنین می توانید فقط از خاصیت type استفاده کنید و خاصیت language را ننویسید.

```
<html>
<head>
   <title> My JavaScript </title>
</head>

<body>
<script language="javascript">
     document.write("Hello designers and developers.")
</script>
</body>
</html>
```

خاصیت src

تگ script یک خاصیت دیگر هم دارد که src است. این خاصیت برای آدرس دهی است. کدهای جاوااسکریپت را می توانید در یک فایل جداگانه نوشته و با پسوند js ذخیره کنید سپس با خاصیت src آدرس آن فایل را مشخص کنید. توجه داشته باشید که در آن فایل نباید دیگر تگ script را بنویسید. معمولا در این حالت تگ اسکریپت در قسمت head صفحه گنجانده می شود تا با شروع صفحه فایل جاوااسکریپت فراخوانی شود.

```
<html>
<head>
<script type="text/javascript" src="\example1.js">
</script>
</head>
</html>
```

حال می خواهیم بررسی کنیم کدهای جاوااسکریپت را در کدام قسمت از صفحه وب بنویسیم؟

نوشتن اسکریپت در قسمت head

اگر می خواهید تا یک کد اسکریپت قبل از بارشدن صفحه اجرا شود یا کدها را هر زمان نیاز دارید فراخوانی کنید،آن ها را مابین تگ head بنویسید. بطور مثال شما نیاز به این دارید که به محض ورود کاربر به صفحه در همان ابتدا پیغامی ظاهر شود. به کدهای این مثال توجه کنید:

```
<html>
<head>
<title> My JavaScript </title>
<script type="text/javascript">
```

```
     alert("I told you it was simple!");
</script>
</head>

<body> </body>
</html>
```

قبل از بارشدن کامل صفحه در مرورگر، پیغامی دریک جعبه پیغام نمایش داده می شود.

- alert() یک تابع مفید در جاوااسکریپت است این تابع یک رشته را بعنوان پارامتر دریافت و آن را در یک جعبه پیغام نمایش می دهد.

نوشتن اسکریپت در قسمت body

اگر می خواهید تا کدهای جاوااسکریپت هنگام بار شدن صفحه اجرا شوند آن ها را در قسمت body صفحه وارد کنید. دقت کنید شما برای استفاده از تگ <script> در صفحه محدودیتی ندارید.

```
<html>
<head></head>
<body>
<script type="text/javascript">
     // javascript codes here
</script>
</body>
</html>
```

حتی می توانید توسط کدهای جاوااسکریپت قسمتی از یک صفحه را ایجاد کنید یعنی با ترکیب جاوااسکریپت و تگ های html در body یک عنصر در صفحه ایجاد کنید.

```
<html>
<head></head>
<body>
<script type="text/javascript">
     document.write("<input align='center' type='button'
          value='ok'>");
</script>
</body>
</html>
```

در این مثال هنگام بارشدن صفحه یک دکمه ی ok بر روی صفحه نمایش داده می شود.

فایل خارجی جاوااسکریپت

برای جلوگیری از تکرار یک کد اسکریپت در صفحات یک وب سایت، کدهای جاوااسکریپت را در یک فایل جداگانه نوشته و با پسوند js آن ها را ذخیره کنید سپس در هر قسمت و هر صفحه ای که لازم بود، آن ها را اضافه کنید. خط زیر را نوشته و آن را در فایلی با نام example1.js ذخیره کنید:

```
alert('my external file');
```

حالا در یک صفحه Html کدهای زیر را بنویسید.

```
<html>
<head>
<script type="text/javascript" src="example1.js">
```

```
</script>
</head>

<body></body>
</html>
```

دقت کنید که اگر فایل جاوااسکریپت را در پوشه ای جدا از فایل html قرار می دهید حتماً در خاصیت src آدرس دقیق آن را بنویسید. هنگام مشاهده صفحه جعبه پیغامی با متن مشخص شده نمایش داده می شود.

مدیریت رویداد

کدهای جاوااسکریپت قابلیت قرار گرفتن داخل شی های html، برای مدیریت رویداد را نیز دارا هستند. در این صورت دیگر نیاز به نوشتن تگ <script> نمی باشد. به کدهای زیر توجه کنید:

```
<html>
<head>
</head>
<body>
     <input type='button' value='send'
          onclick='alert("message sent.");'>
</body>
</html>
```

در این مثال هنگام فشردن دکمه ok، پیغامی نمایش داده می شود. به بیان دیگر با عملکردی در صفحه توسط کاربر یک عکس العمل توسط برنامه ایجاد می شود.

در این بخش لیستی از رویدادهای که امکان اتفاق افتادن بر روی یک مرورگر را دارند آورده شده است:

onClick برای کلیک کردن دکمه چپ موس توسط کاربر.

onDblClick رویداد برای دوبار کلیک دکمه چپ موس.

onMouseDown زمانیکه دکمه چپ موس را پایین نگه می داریم.

onMouseUp رها کردن دکمه موس پس از فشردن آن.

onMouseOver قرار گرفتن اشاره گر موس بر روی یک شی از صفحه.

onMouseOut خارج شدن اشاره گر موس از روی یک شی در صفحه.

onKeyPress فشردن یک کلید.

onKeyDown زمانیکه کاربر یک کلید را پایین نگه می دارد.

onKeyUp رها کردن یک کلید پایین نگه داشته شده.

onFocus هنگامیکه توسط موس و یا دکمه Tab بر روی یکی از اجزای صفحه متمرکز شوید.

onBlur هنگامیکه تمرکز از روی یکی از اجزای صفحه خارج شود.

onSelect انتخاب کردن یک متن در صفحه یا در یک textbox.

onChange تغییر در اجزای فرم، مانند تغییر متن در textbox.

onSubmit فشردن دکمه تأیید یک فرم.

onReset فشردن دکمه reset یا همان پاک کردن اطلاعات فرم.

onLoad تکمیل شدن یک صفحه وب یا یک تصویر در مرورگر.

onUnload بستن مرورگر یا خروج از یک صفحه وب.

onResize تغییر اندازه پنجره مرورگر.

onError این رویداد زمانی اجرا می شود که بار گذاری یک صفحه یا یک تصویر به مشکل برخورد کنیم.

به مثال زیر توجه کنید:

```
<html>
<head>
     <title> JavaScript Event Example </title>
</head>
<body onmousedown=”document.bgcolor=’pink’”>
     <h1>
          Click this page to turn it pink!
     </h1>
</body>
</html>
```

در این مثال با فشردن دکمه موس رنگ زمینه صفحه تغییر می کند.

چگونگی استفاده از رویدادهای موس را در مثال زیر می توانید مشاهده کنید.

```
<html>
<head></head>
<body>
<h2>
      Follow the instructions: <br> (Or double-click to get
      help.)</h2>
<ul>
<a href="#" onMouseOver="alert('As you move over the
      spot.');"> onMouseOver </a>
<p>
<a href="#" onMouseUp="alert('You have to press Down to get
      Up');"> onMouseUp </a>
<p>
<a href="#" onMouseDown="alert('You can keep it down.');">
      onMouseDown </a>
<p>
<a href="#" onClick="alert('Any click will do');"> onClick
</a>
<p>
<a href="#" onMouseOut="alert('First move over and then
      off');"> onMouseOut </a>
<p>
<form>

<input type=button value ="onMouseMove"
```

```
        onMouseMove="alert('Press enter or return--don\'t
        move the mouse!')">
<p>
</ul>
</form>
</body>
</html>
```

مشاهده خطاهای جاوااسکریپت

در برنامه نویسی همواره امکان وجود اشتباه از طرف برنامه نویس وجود دارد. بطورکلی در برنامه نویسی دو گونه خطا وجود دارد:۱- خطاهای گرامری (Syntax) ۲- خطاهای منطقی (Logical)

دسته اول خطاهای هستند که از استفاده نادرست از دستورات زبان برنامه نویسی و یا رعایت نکردن قواعد گرامری زبان تولید می شوند به مثال زیر توجه کنید:

```
<html>
<head>
<title>A First Script</title>
<script language="javascript">
     docment.write("You're using JavaScript");
</script>
</head>
<body>
     <h1>A First Script</h1>
</body>
</html>
```

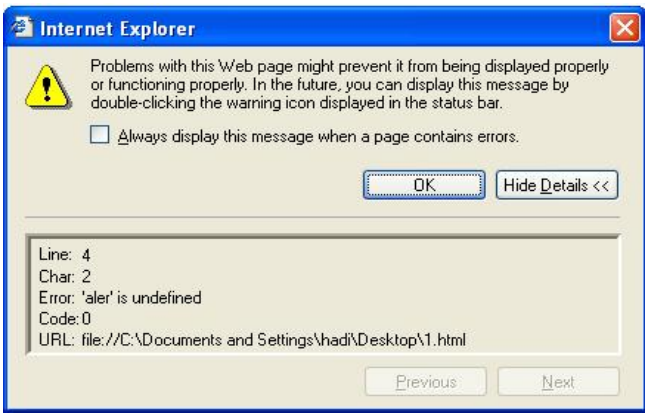
زمانیکه صفحه بارگذاری می شود یک مثلث زرد کوچک با یک علامت تعجب در سمت چپ نوار وضعیت مرورگر IE قرار می گیرد که نشان دهنده وجود خطای گرامری در برنامه است. کافی است برای مشاهده علت خطا بر روی آن مثلث دوبار کلیک کنید، یک پنجره در وسط مرورگر باز می شودکه در آن جزئیات خطای ایجاد شده (توضیح خطا و شماره خط وجود خطا) را می توانید مشاهده کنید. مانند شکل ۱-۲.

دسته دیگر خطاها زمانی است که منطق دستورات غلط است در نتیجه با اجرای برنامه به هدف برنامه دست پیدا نمی کنید و یا منطق داده ها برای عملیات غلط است مانند تقسیم عددی بر صفر، به این گونه، خطاهای منطقی گویند.

متغیرها و انواع آن ها

متغیر، اسمی برای خانه های حافظه RAM است که برای نگهداری اطلاعات به کار می رود و محتویات آن در طول اجرای برنامه ممکن است تغییر کند. برای انتخاب اسامی متغیرها، می توان از حروف بزرگ و کوچک انگلیسی، اعداد و کاراکتر استفاده نمود. اولین کاراکتر اسم انتخاب شده نباید عدد باشد. زبان جاوااسکریپت نسبت به حروف حساس است. مثلا متغیر با اسم counter با متغیر COUNTER فرق داشته و برابر نیستند. در انتخاب اسامی متغیرها حوصله به خرج دهید، از نام هایی بهره گیرید که با مفهوم متغیرتان هماهنگ باشد، که این باعث خواناتر شدن برنامه خواهد شد و در صورت بروز اشکال در برنامه، کار عیب یابی ساده تر خواهد شد.

در جاوااسکریپت سه نوع داده ی موجود می باشد، نوع داده ی عددی برای کار با انواع داده های از نوع عدد، نوع داده ی رشته ا ی برای کار با داده های از نوع کاراکتر، و نوع داده ی منطقی (boolean) برای کار با داده های با مقدارهای منطقی True, False. نوع متغیرها در جاوااسکریپت برابر با نوع داده انتسابی به آن متغیر می باشد.



شکل ۱-۲ پنجره نمایش توضیحات خطا

حالا می پردازیم به جزئیات تعریف یک متغیر در جاوا اسکریپت:

var نام متغیر = مقداری برای متغیر ;

مقدار انتسابی هم برای مقدار دهی به متغیر و هم برای تعیین نوع متغیر بکار می رود. مانند مثال های زیر:

```
1. var item;
2. var price= 33.44;
3. var wholeThing= 86.45 + (20 *7);
4. var name="Willie B. Goode";
5. var address "123 Elm Street";
6. var subTotal=sumItems
7. var mixString= 11.86 + "Toy Cats";
8. var test = (alpha > beta)
```

در شماره ۱ یک تعریف متغیر بدون مقداردهی داریم، شماره ۲ و۳ متغیر از نوع عددی، ۴ و۶ و۷ متغیر از نوع رشته، شماره ۶ متغیر از نوع یک متغیر دیگر و متغیر شماره ۸ از نوع boolean است.

اصطلاح var از کلمه variable می آید که نوشتن آن اختیاری می باشد، یعنی می توانید یک متغیر را بدون نوشتن آن هم تعریف کنید ولی اگر می خواهید کدتان خواناتر باشد بهتر است که از var استفاده کنید. نام یک متغیر نباید از کلمات رزرو شده زبان باشد.

abstract	final	public
boolean	finally	return
break	float	short
byte	for	static
case	function	super

catch	goto	switch
char	if	synchronized
class	implements	this
const	import	throw
continue	in	throws
debugger	instanceof	transient
default	int	true
delete	interface	try
do	long	typeof
double	native	var
else	new	void
enum	null	volatile
export	package	while
extends	private	with
false	protected	

جدول ۱-۲ کلمات رزرو شده ی جاوااسکریپت

تبدیل نوع

در جاوااسکریپت هر متغیر دارای چند متد بطور پیش فرض است که برای تبدیل نوع کاربرد دارند.

تبدیل به رشته

متغیرها با انواع مختلف قابل تغییر به نوع رشته ای با متد toString() می باشند. به مثال زیر دقت کنید:

```
var bfoung = false
alert(bfound.toString());    //خروجی  "false"

var iNum = 10;
var fNum = 10.0;
alert(iNum.toString());      // خروجی  "10"
alert(fNum.toString());     //خروجی  "10"
```

متد toString() بطور پیش فرض اعداد را در مبنای ۱۰ نمایش می دهد ولی می توانید با مشخص کردن مبنای مورد نظر بعنوان پارامتر این متد، اعداد را بصورت رشته ای به مبنای دلخواه تبدیل نماید.

```
var iNum = 10;
alert(iNum.toString(2));      // خروجی  "1010"
alert(iNum.toString(8));      // خروجی  "12"
alert(iNum.toString(16));      // خروجی  "A"
```

تبدیل به عدد

برای تبدیل نوع رشته به نوع عددی از دو متد parseInt() و parseFloat() استفاده می شود.

متد parseInt() از کاراکتر موقعیت صفر رشته شروع می کند و آن را بررسی می کند اگر کاراکتر عددی نباشد مقدار NaN را برمی گرداند و کار خاتمه می یابد. اگر عددی باشد تا زمانیکه با یک کاراکتر غیر عددی

برسد یک کاراکتر، یک کاراکتر می خواند و به عدد تبدیل می کند، و زمانی که به کاراکتر غیر عددی برسد عملیات متوقف می شود و عدد را بر می گرداند.

```
var iNum1 = parseInt('1234blue');  // 1234
var iNum2 = parseInt('1234');      // 1234
var iNum3 = parseInt('0XA');       // 10
var iNum4 = parseInt('22.5');      // 22
var iNum5 = parseInt('blue');      // NaN
```

این متد همچنین می تواند اعداد را به مبنای متفاوت تبدیل کند. کافیست در پارامتر دوم این متد مبنای مورد نظر نوشته شود.

```
var iNum1 = parseInt('AF',16);     // 175
var iNum2 = parseInt('10',2);      // 2
var iNum3 = parseInt('10'8);       // 8
```

متد `parseFloat()` برای تبدیل نوع رشته ی به اعداد اعشاری می باشد.

```
var fNum1 = parseFloat('1234blue');     // 1234.0
var fNum2 = parseFloat('0XA');     // NaN
var fNum3 = parseFloat('22.5');    // 22.5
var fNum4 = parseFloat('blue');    // NaN
```

عملگر

عملگرها (operator) کاراکتری برای نشان دادن یک علامت مشخص هستند که هر کدام از آن ها وظیفه خاصی دارند.

انواع عملگرها

عملگرها به چند دسته تقسیم می شوند:

عملگرهای محاسباتی (Operators Arithmetic)

این نوع عملگرها چهار عمل اصلی محاسبات را انجام می دهند. علاوه بر آن ها یک عملگر وظیفه نمایش باقی مانده یک تقسیم را بعهده دارد و دو عملگر دیگر هر کدام به ترتیب یک واحد به عملوند خود اضافه و یک واحد از عملوند خود کم می کنند که در جدول ۲-۲ مشخص شده اند.

عملگر	توصیف	مثال	نتیجه
+	جمع	x = 2, y = 2 x+y	4
–	تفریق	x = 5 , y = 3 x - y	2
*	ضرب	x = 5 , y = 4 x * y	20
/	تقسیم	x = 20 , y = 4 x / y	5
٪	باقی مانده	x = 19 , y = 4 x % y	3
++	افزایش	x = 5 x++	X=6

X=5	x = 5 x--	کاهش	--

جدول ۲-۲ عملگرهای محاسباتی

عملگرهای دیگر که به نام عملگرهای انتسابی (Operators Assignment) معروف هستند که یک مقداری را به یک متغیر نسبت می دهند. این عملگرها برای خلاصه نویسی در عبارات استفاده می شوند.

برابر است با	مثال	عملگرها
x = y	x = y	=
x = x + y	x + = y	+ =
x = x - y	x - = y	- =
x = x * y	x * = y	* =
x = x / y	x * = y	/=
x = x % y	x %= y	%=

جدول ۳-۲ عملگرهای انتسابی

عملگرهای مقایسه ای

این نوع عملگرها مقدارهای دو متغیر را با یکدیگر مقایسه می کنند و نتیجه این مقایسه یا درست است یا غلط که در دستورات شرطی بسیار کاربرد دارند.

عملگرهای منطقی

در این بخش سه نوع عملگر وجود دارند که در حقیقت یک یا چند نتیجه را بررسی می کنند و جواب را بازمی گردانند. این عملگرها به این شکل می باشند:

```
&& , || , !
```

عملگر && که عملگری برای عملیات and است.

```
x = 5
y = 3
( x < 10 && y > 1)
```

مثال	توصیف	عملگرها
x = 5 , y=5 x == y	مقادیر اگر برابر باشند نتیجه درست خواهد بود.	==
x=5,y = "5" x === y	مقایسه بین مقادیر و جنس متغیر که در مثال روبرو نتیجه غلط بازگردانده می شود	===
5 != 8	اگر مقادیر برابر نباشند نتیجه درست است	!=

>	بزرگتر	8 > 5
<	کوچکتر	8 < 10
>=	بزرگتر یا مساوی که در این مثال نتیجه غلط است	5 >= 8
<=	کوچکتر یا مساوی که در این مثال نتیجه درست است	5 <= 8

جدول ۴-۲ عملگرهای مقایسه ای

در اینجا اگر مقدار متغیر x کوچکتر از عدد ۱۰ باشد و y بزرگتر از ۱ باشد، نتیجه true بازگردانده می شود. چون and آمده است پس باید دو طرف عملگر جواب درست باشد تا در مجموع نتیجه درست باشد.

عملگر || که عملگری برای عملیات or است.

```
x = 6
y = 3
( x == 5 || y == 5 )
```

حداقل یکی از طرفین عملگر باید درست باشد تا نتیجه درست باشد.

عملگر ! در حقیقت یک عملیات و معادله را بررسی می کند اگر جواب آن معادله درست نباشد نتیجه true یا همان درست را برمی گرداند:

```
x = 5
y = 4
! ( x == y )
```

کاربرد عملگرها

عملگر جمع در رشته ها برای الحاق دو رشته بکار می رود.

```
Msg = 'this'+' is text';
```

بر خلاف جمع دو متغیر رشته ای، تفریق دو متغیر رشته ای امکان پذیر نیست و در صورت تفریق، در هر حالت، حاصل برابر با رشته NaN به معنی Not a Number خواهد بود. این خاصیت شامل تفریق یک متغیر رشته ای از عددی و بالعکس می باشد. در تفریق متغیر های «منطقی» True مفهوم 1(یک) و False مفهوم 0 (صفر) خواهد داشت. به مثال زیر توجه کنید:

```
a=true
b=false
c=a-b
d=b-a
```

در نتیجه خواهیم داشت: c=1-0=1 و همچنین d=0-1=-1 در این مورد a-b و b-a با هم متفاوتند.

عملگر ضرب: برای انجام عمل ضرب از '*' استفاده می کنیم. در این حالت می توان به ضرب دو عدد (چه صحیح و چه اعشاری) اشاره نمود. در صورت ضرب دو متغیر رشته ای یا یک متغیر رشته ای در یک متغیر عددی حاصل رشته NaN خواهد بود. پس نمی توان متغیر رشته ای را در هیچ نوع متغیر دیگر ضرب نمود.

در ضرب متغیر های منطقی سه حالت پیش می آید:

۱- در صورت ضرب دو متغیر منطقی True، حاصل 1 خواهد بود.

۲- ضرب دو متغیر منطقی false نیز حاصل صفر خواهد داشت.

۳- در ضرب یک متغیر منطقی True در یک متغیر False، جواب صفر بدست خواهد آمد.

نتیجه اینکه در ضرب متغیر های منطقی فقط دو جواب 0 و1 خواهید داشت و فقط در صورتی جواب برابر 1 خواهد بود که هیچ متغیر False ی در ضرب شرکت نداشته باشد.

عملگر تقسیم: عملگری که می توان با استفاده از آن عمل تقسیم را انجام داد '/' است.

- زبان جاوااسکریپت در حالت اعشاری فقط تا ۱۶ رقم اعشاری را محاسبه می کند.
- در عمل تقسیم هر عددی بر عدد صفر، حاصل برابر با رشته Infinity به معنی بی نهایت خواهد بود.

در تقسیم یک متغیر رشته ای به یک متغیر عددی و بالعکس حاصل برابر با NaN خواهد بود. در تقسیم متغیر های منطقی، حالت های زیر به وجود می آید:

- در تقسیم یک متغیر منطقی True بر True حاصل برابر با 1 خواهد بود
- در تقسیم یک متغیر منطقی True بر False حاصل برابر با رشته Infinityخواهد بود
- در تقسیم یک متغیر منطقی False بر True حاصل برابر با صفر خواهد بود
- در تقسیم یک متغیر منطقی False بر False حاصل برابر با رشته Infinity خواهد بود

دستورات کنترلی

اگر بخواهیم تحت شرایطی، تعدادی از دستورات اجرا شوند و یا تعدادی دیگر از دستورات اجرا نشوند، باید از ساختارهای تصمیم استفاده کنیم. این ساختارها، شرطی را بررسی می کند در صورت درست بودن شرط، مجموعه ای از دستورات را انجام می دهند.

دستور if

ساختار if که نام دیگرش، دستور انتقال کنترل شرطی است، شرطی را تست می کند و در صورتی که آن شرط دارای ارزش درستی باشد، مجموعه ای از دستورات را اجرا می کند.

به کد زیر توجه کنید:

```
<script language="javascript" type="text/javascript">
     ...
     if ( condition )
     {
          //...
     }
     ...
</script>
```

چنانچه condition، یک عبارت با منطق true باشد دستورات مابین بلوک {...} اجرا می شوند.

دستور if...else

در این ساختار در صورت نادرست بودن شرط، مجموعه دستورات داخل بلوک بعد از else اجرا می شوند.

```
if( condition ){
      ...
}
else
{
      ...
}
```

تابع

در برنامه می توان، عملیات برنامه را تقسیم بندی کرد تا هر بخش را یک تابع انجام دهد. برای نوشتن تابع باید اهداف تابع مشخص باشد. تابع چه وظیفه ای برعهده دارد؟ ورودیهای تابع چیست؟ وخروجی تابع چیست؟ با دانستن این موارد نوشتن تابع چندان دشوار نیست.

هر تابع دارای دو جنبه است، جنبه تعریف تابع و فراخوانی تابع. تعریف تابع، مجموعه ای از دستورات است که عملکرد تابع را مشخص می کند و فراخوانی تابع، دستوری است که تابع را فراخوانی می کند. فراخوانی تابع با نام تابع انجام می شود. نام گذاری تابع، از قوانین نام گذاری متغیرها تبعیت می کند.

```
function نام تابع ([ لیست پارامترها])
{
      //بدنه دستورات تابع
}
```

همانطور که مشاهده می کنید برای تعریف یک تابع از کلمه function استفاده می شود، دقت کنید که حروف آن باید حتماً کوچک باشد. بعد از نوشتن دستور function نوبت به تعیین یک نام برای این تابع است که بهتر است این نام متناسب با وظیفه ی که این مجموعه قرار است انجام دهد، باشد.

شما یک تابع را در هر کجای یک صفحه html می توانید قراردهید اما بهتراست در قسمت head نوشته شود. حتی می توانید یک تابع را در فایل خارجی جاوااسکریپت که با پسوند js ذخیره می شود وارد کنید تا در صفحات مختلف آن را بکار ببرید.

فراخوانی یک تابع

توابعی که در یک صفحه html تعریف می شوند به خودی خود اجرا نخواهند شد. اگر شما یک تابع تعریف کنید پس از باز کردن صفحه وب خواهید دید که هیچ اتفاقی نمی افتد. توابع پس از تعریف باید در جای مورد نظر فراخوانی شوند. در حقیقت توابع آماده هستند تا پس از فراخوانی اجرا شوند:

```
<html>
<head>
<title </title>

<script type="text/javascript" >
```

```
     function funAlert()
     {
          alert ( " Hello friends! " )
     }

</script>

</head>
<body>

<form>
   <input type="button" onclick="funAlert()" value="Click me">
</form>

</body>
</html>
```

یکی از راه‌های فراخوانی توابع، استفاده از رویدادهاست. اگر می‌خواهید کدی، به هنگام باز شدن صفحه، اجرا شود، از رویداد onLoad استفاده کنید. اگر می‌خواهید با کلیک کردن بر روی چیزی، تابعی اجرا شود، از onClick استفاده کنید و به همین ترتیب. در کد بالا تابع funAlert به هنگام فشردن دکمه ی اجرا می شود. همچنین اگر بخواهید تابع funAlert، به هنگام بار شدن صفحه اجرا شود. برای این کار باید در تگ Body این کد را اضافه کنید:

```
<Body onLoad="funAlert()">
```

پارامترهای تابع

پارامترها اطلاعاتی هستند که هنگام فراخوانی تابع، از برنامه فراخوان به آن ارسال می شوند. به عبارت دیگر، پارامترها وسیله ی برای تبادل اطلاعات بین بخش فراخوان و تابع فراخوانی شونده هستند. اگر تعداد پارامترها از یکی بیشتر باشد، باید با کاما از هم جدا شوند. بطورکلی پارامترهای یک تابع متغیرهای هستند که تعریف شده اند ولی مقداری ندارند. این متغیر ها با فراخوانی تابع مقدار می گیرند. در فراخوانی تابع به ازای هر پارامتر تابع یک مقدار به تابع ارسال می شود.

```
<html>
<head>
<title>Shed Planner</title>
<script language="JavaScript">
function shedWalls(front,side) {
      var numFront=(front * 12 * 2)/8;
      var numSide=(side * 12 * 2)/8;
      var total=numFront+numSide;
      var message="You will need " + total + " one by eight-
            inch boards for your
      shed walls."
      document.write(message);
}
shedWalls(50,40);    //The function is called
```

```
</script>
</head>
<body bgcolor="palegoldenrod">
</body>
</html>
```

بازگرداندن یک مقدار توسط دستور return

توابع جاوااسکریپت دارای یک دستوری به نام return هستند که وظیفه این دستور برگرداندن مقادیر خروجی از آن تابع است.

```
function test( ) {
    return (number);
}
.......
```

این مجموعه کد را اگر بدون استفاده از return اجرا کنید، نتیجه درست نخواهید گرفت. به مثال زیر توجه کنید:

```
<html>
<head>
<title> return statement </title>

<script type="text/javascript">
function total(a,b )
{
     x = a * b;
     return x;
}
</script>

</head>
<body>
<script type="text/javascript" type="text/javascript">
     price = total( 4, 15 );
     document.write( price );
</script>
</body>
</html>
```

همانطور که مشاهده می کنید دو پارامتر برای تابع total تعریف شده است. حاصلضرب این دو پارامتر در متغیر x ذخیره می شود و توسط دستور return مقدار جواب برگشت داده می شود. در نظر داشته باشید که توابع در جاوااسکریپت بسیار کاربرد دارند و شما هنگام نوشتن اسکریپت خود پی به اهمیت آن ها خواهید برد مخصوصاً زمانی که یک سری کد را بخواهید در کلیه صفحات یک وب سایت بکار ببرید می توانید آن ها را بصورت یک تابع در یک فایل خارجی جاوااسکریپت ذخیره کنید و سپس در هر صفحه آن تابع را فراخوانی کنید.

تابع بعنوان داده

از تابع می توان مانند داده های دیگر (رشته ی، عددی، منطقی) برای مقداردهی به یک متغیر استفاده کرد در این حالت مقدار بازگشتی از تابع به متغیر نسبت داده می شود:

```
<script language="JavaScript">
      var total=function(item,tax) {return item +=tax};
      document.write("Your bill is $" + total(14.95,.06));
</script>
```

متغیر محلی و سراسری

اگر بخواهیم از نظر نوع، به متغیر نگاه کنیم، سه نوع مهم دارد: رشته‌ای، عددی، منطقی. اما از لحاظ موقعیت و مکان استفاده، یک تقسیم بندی دیگر برای متغیر وجود دارد:

۱- متغیر سراسری، عمومی (Global)

اگر متغیر را خارج از تابع بنویسید، عمومی است، چون می‌شود در سرتاسر برنامه، آن را صدا زده و مورد استفاده قرار داد.

۲- متغیر محلی (Local)

اگر متغیری را داخل تابع تعریف کنید، محلی است. یعنی فقط در همان محل و همان تابع، کاربرد دارد، و بیرون از آن، نه می‌شود صدا زد و نه می‌شود استفاده کرد.

خصوصیات متغیر محلی و سراسری:

- از متغیر سراسری می‌شود در توابع مختلف استفاده کرد، اما متغیر محلی، فقط در همان تابع که تعریف شده است، کاربرد دارد.
- متغیر محلی می‌تواند با متغیر سراسری، همنام باشد. اگر این دو متغیر، همنام شدند، تغییر مقدار یکی از آن دو، تاثیری در دیگری ندارد.

شی چیست؟

وظیفه شی ها (Objects) را می توان شبیه به متغیر ها دانست اما شی ها بسیار کامل تر از متغیرها هستند چون آن ها می توانند اطلاعات بیشتری را در خود ذخیره کنند، شی دارای یک مجموعه ی از صفات بعنوان خاصیت و یک مجموعه از رفتارها بعنوان متد هستند و همچنین تفاوت دیگری هم که با متغیرها دارند این است که متغیرها باید تعریف شوند تا جاوااسکریپت آن ها را بشناسد اما بعضی از شی ها در جاوااسکریپت ساخته و تعریف شده اند. این شی های از قبل تعریف شده را درون ساخت یا Built-in می گویند مانند شی Date. البته باید همین اشیا هم در ابتدای کد یکبار دیگر با دستور new تعریف شوند:

```
mydate = new Date()
name = new Array()
```

روش ساخت یک شی

گاهی اوقات نیاز است که یک شی را خودتان تعریف کنید تا بهتر بتوانید کد خود را مدیریت کنید. اگر بخواهید یک شی را در اسکریپت خود بسازید نیاز به دو چیز دارید، ابتدا یک تابع سازنده که به آن Constructor می گویند و در آن نوع شی تعریف می شود، سپس به نمونه ی از شی که به آن Instance می گویند که توسط دستور new تعریف می شود. به مثال زیر توجه کنید.

می خواهیم یک شی برای اتومبیل تعریف کنیم که خاصیت های مانند اسم، رنگ و مدل را داشته باشد. برای این کار ابتدا یک تابع باید تعریف کنیم به نام Car و خاصیت های آن را به عنوان پارامترهایش بنویسیم و سپس این پارامترها را بر اساس خاصیت هایش مقداردهی کنیم:

```
function Car ( pname, pmodel, pcolor ) {

     this.name = pname
     this.model = pmodel
     this.color = pcolor
}
```

به کلمه this دقت کنید، این یک کلمه رزرو شده جاوااسکریپت است و به شی که در حال تعریف است، اشاره دارد و در حقیقت به مفسر جاوااسکریپت می گوید که این خاصیت شی تعریف شده با این مقدار مقداردهی شود.

حالا باید یک نمونه از شی جدید بسازید که این نمونه را در متغیری بنام mycar ذخیره می کنیم:

```
mycar = new Car ( "peugeot", "206", "Red" )
```

این خط کد به جاوااسکریپت می گوید که یک نمونه به نام mycar دارای خاصیت های اسم که peugeot و مدل 206 و رنگ Red می باشد. اکنون می خواهیم از نمونه، اسم ماشین و رنگ را در صفحه چاپ کنیم:

```
document.write ( mycar.name + "<br />" )
document.write ( mycar.color )
```

بعد از اینکه خصوصیاتی برای یک شی تعریف کردید نوبت به متدهای شی ساخته شده، می رسد. در اینجا متدی که تعریف می کنیم برای کلیه ماشین ها می توانیم بکار گیریم. الگوی کلی تعریف یک متد به شرح زیر است:

```
object.methodname()= function_name
```

object نام نمونه ی است که ساختیم، methodname نام متدی است که در نظر می گیرید و function name اسم تابعی است که باید عملیات مورد نظر را انجام دهد. سپس هر کجا که لازم بود از متد استفاده کنید ابتدا نام نمونه و سپس با گذاشتن یک نقطه متد را به همراه پرانتز می نویسید که اگر این متد پارامتری هم داشت باید آن را داخل پرانتز مشخص کنید.

متدی که می خواهید برای شی Car بسازید را باید ابتدا بصورت یک تابع تعریف کنید:

```
function fdisplaycar() {

     var result = " I like " + this.name + " " +  this.model +
          " with " + this.color + " color."
     document.write ( result )
}
```

حالا با استفاده از دستور this این تابع را به عنوان متد شی Car تعریف می کنید:

```
function Car ( name, model, color ) {
     this.name = name
     this.model = model
     this.color = color
     this.displaycar = fdisplaycar
}
```

بعد از تعریف توابع، متد را اجرا می کنید:

```
mycar.displaycar()
```

تعریف DOM

DOM مخفف سه کلمه **M**odel **O**bject **D**ocument می باشد. متنی که در زیر آمده تعریف DOM بر اساس W3C است:

" The Document Object Model is a platform- and language - neutral interface that will allow programs and scripts to dynamically access and update the content, structure and style of documents. The document can be further processed and the results of that processing can be incorporated back into the presented page."

DOM یک واسطی است که به برنامه ها و اسکریپت ها اجازه دسترسی به محتوا و ساختار داخلی پنجره مرورگر را می دهد تا آن ها بتوانند تغییراتی ایجاد کنند که نتیجه این تغییرات در صفحه وب نمایان شود. در همین رابطه زبانی به نام DHTML بوجود آمد که با ترکیب کردن عناصر html با css و اسکریپت ها می تواند محتوای اسناد وب را بصورت متحرک و پویا درآورد. در اینجا پویا بودن فقط به این منظور نیست که متن ها حرکت کنند و از یک طرف وارد بشوند و از طرف دیگر خارج، ممکن است لینک ها هر بار شما را به یک سایتی پیوند دهند و یا تصاویر مرتب تغییر کنند، در مجموع هر تغییری که نیاز به کدنویسی مجدد برای آن صفحه را نداشته باشد، مفهوم پویا بودن را می رساند.

سلسله مراتب اشیا DOM

جاوااسکریپت، شی هایی دارد که می توانند بطور مستقیم با خود مرورگر ارتباط برقرار کنند و همچنین شی هایی که با صفحه وب در ارتباط هستند. در کل می توان این اشیا را به دو دسته اصلی تقسیم کرد: دسته اول که

BOM گفته می شود: (Model Object Browser)شی هایی که با پنجره مرورگر می توانند ارتباط برقرار کنند و دسته دوم DOM که با صفحه html (Document) ارتباط دارند. تمامی این اشیا براساس یک سلسله مراتب این ارتباط را برقرار می کنند که در رأس آن ها شی window می باشد. این سلسله مراتب در شکل ۳-۲ آورده شده است.

همانطور که در شکل۳-۲ مشاهده کردید از شی window تا شی Document جزو دسته BOM می باشند و اشیاء بعد از Document در دسته DOM قرار می گیرند. در حال حاضر در کل این مجموعه را DOM می گویند.

شما باید این سلسله مراتب را برای استفاده اشیا آن، رعایت کنید. برای استفاده از آن ها باید ابتدا اسم پدر و سپس اسم فرزند را نوشت و در آخر هم خاصیت و یا متد آن فرزند که همه این ها با یک نقطه از هم باید جدا شوند. بطور مثال اگر می خواهید یک فرم را به نام form1 صدا بزنید تا با آن ارتباط برقرار کنید باید به شکل زیر عمل شود:

```
window.document.form1
```

مانند همه اشیا در جاوااسکریپت، این شی ها هم دارای خاصیت ها و متدهایی هستند که بعد از رعایت کردن سلسله مراتب با گذاشتن یک نقطه می توانید آن ها را بنویسید. مانند:

```
window.history.go(-1)
```

اگر این شی را در یک لینک قرار دهید با کلیک کردن بر روی آن، کاربر به صفحه قبل بازگشت داده می شود.

در جدول ۵-۲ شی های اصلی این سلسله مراتب لیست شده است.

window	این شی مستقیماً با پنجره مرورگر در ارتباط است که معمولاً خصوصیات و متدهای این شی در تگهای body و frameset بکار می روند.
navigator	این شی شامل اطلاعاتی در رابطه با مرورگر کاربر است که توسط خصوصیات و متدهای آن این اطلاعات ذخیره تا مورد استفاده قرار گیرند.
frame	برای کار کردن بر روی فریمهای html.
document	دسترسی و ارتباط با کلیه عناصر html مانند، متن ها فرم ها و غیره.
location	شامل اطلاعاتی در مورد URL صفحه ای که در حال نمایش است.
history	این شی دارای اطلاعاتی برای رجوع کاربر به صفحاتی که قبل از آن صفحه مشاهده شده و بر عکس.

جدول ۵-۲ شی های اصلی مرورگرها

DOM یک روش استاندارد برای دسترسی به اشیا صفحه وب و تغییرات در صفحه وب می باشد. DOM ساختار صفحه Html را بعنوان یک درخت (Tree) نمایش می دهد همان طور که می دانید یک درخت از نودها (Nodes)تشکیل شده است. براساس منطق DOM هر چیزی در صفحه Html یک گره درخت است.

این گره ها انواع مختلفی به شرح زیر دارند:

۱- گره document (گره صفحه وب)

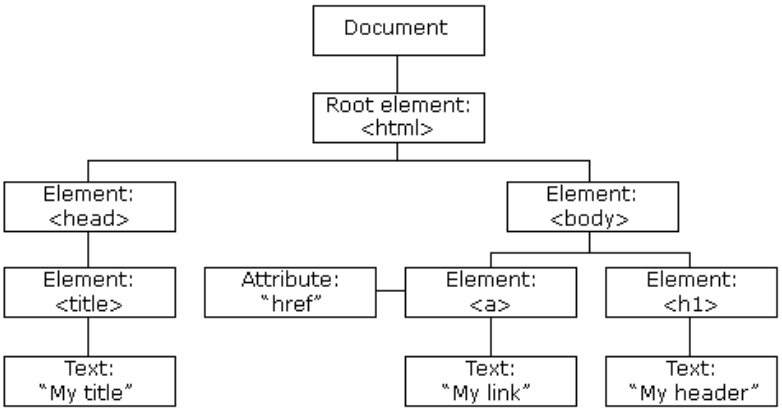
۲- گره text (متن های که در تگ های Html می باشند.)

۳- گره element (تگ های Html)

۴- گره attribute (خاصیت های Html)

۵- گره comment (توضیحات در صفحه وب)

در شکل ۲-۲ ساختار سلسه مراتبی این درخت را مشاهده می کنید.



شکل ۲-۲ نمودار انواع گره های DOM

هر درخت دارای یک گره ریشه ی Document می باشد. گره های نوع text در آخرین سطح درخت و بعنوان گره های برگ می باشند. به کد زیر توجه کنید:

```
<html>
  <head>
    <title>DOM Tutorial</title>
  </head>
  <body>
    <h1>DOM Lesson one</h1>
    <p>Hello world!</p>
  </body>
</html>
```

همه گره ها بجز نود document دارای گره پدر (parent) هستند. در این مثال پدر <head> و <body> گره <html> است ویا پدر متن Hello world! گره <p> است.

بیشتر گره های element دارای فرزند (child) می باشند. گره <head>، فرزندش گره <title> است و <title> یک فرزند دارد که یک گره متنی DOM Tutorial است.

به گره های که دارای یک پدر باشند گره برادر یا همزاد گویند. در این مثال گره <h1> و باهم برادرند برای اینکه پدر هر دو گره <body> است.

با جاوااسکریپت می توانید ساختار یک صفحه را تغییر دهید. برای این کار نیاز به دسترسی به عناصر صفحه دارید، این دسترسی بوسیله متدهای ()createTextNode، ()createElement، ()appendChild برای عناصر Html صفحه از طریق DOM فراهم می شود.

برای مثال کد زیر را در نظر بگیرید، می خواهیم یک عبارتی بصورت Hello World! را در صفحه وب در یک تگ به صفحه اضافه کنیم:

```
<html>
<head>
     <title>createElement() Example</title>
</head>
<body>
</body>
</html>
```

ابتدا یک تگ با دستور ()createElement ایجاد می کنیم:

```
var oP = document.createElement("p");
```

مرحله دوم: یک گره text حاوی متن مورد ایجاد می کنیم:

```
var oText = document.createTextNode("Hello World!");
```

مرحله سوم: باید گره text را بعنوان فرزند گره معرفی کنیم:

```
oP.appendChild(oText);
```

مرحله چهارم: باید گره را بعنوان فرزند گره <body> معرفی کنیم:

```
document.body.appendChild(oP);
```

با دستورات بالا یک خط دستور Html بصورت Hello World! به صفحه اضافه می شود.

دسترسی و یافتن گره ها

DOM امکاناتی را فراهم کرده است تا براحتی بتوانید هر عنصر از صفحه را که برای تغییرات نیاز دارید را جستجو کنید برای این کار چندین روش وجود دارد:

۱- استفاده از متد ()getElementById و متد ()getElementsByTagName.

۲- با استفاده از خاصیت های parentNode، firstChild، lastChild.

روش اول

متدهای ()getElementById و ()getElementsByTagName هر عنصری در صفحه Html را می توانند جستجو کنند. این متدها ساختار درختی صفحه را نادیده می گیرند.

اگر می خواهید عنصر <p> را در صفحه پیدا کنید، متد getElementsByTagName() همه ی آن ها را برمی گرداند صرفنظر از اینکه <p> در کدام سطح درخت قرار دارد همچنین متد getElementById() همواره عنصر موجود در صفحه را برمی گرداند اگرچه آن عنصر بصورت مخفی (hidden) باشد.متد getElementById() یک عنصر Html را با ID مشخص برمی گرداند.

شکل کلی این متد بصورت زیر است:

```
document.getElementById("someID");
```

متد getElementsByTagName() تمام عناصری که دارای نام تگ یکسانی هستند را در یک آرایه بر می گرداند. این متد برای تمامی عناصر موجود بر روی صفحه Html کار می کند.

شکل کلی این متد بصورت زیر است:

```
document.getElementsByTagName("tagname");
```

یا

```
document.getElementById('someID').getElementsByTagName("tagnam
e");
```

به مثال زیر توجه کنید: در این مثال تمام تگ های <p> برگشت داده می شوند.

```
document.getElementsByTagName("p");
```

در کد زیر می خواهیم تمام تگ های <p> که دارای ID="maindiv" می باشند را جستجو کنیم:

```
document.getElementById('maindiv').getElementsByTagName("p");
```

آرایه ی از گره ها

زمانیکه با یک آرایه از گره ها کار می کنید باید از یک متغیر برای نگهداری عناصر آرایه استفاده کنید.

```
var x=document.getElementsByTagName("p");
```

در اینجا متغیر x یک آرایه از تگ های <p> موجود در صفحه می باشد. برای دسترسی به هر کدام از تگ ها باید از اندیس استفاده کرد. اندیس از صفر شروع می شود. به کد زیر توجه کنید:

```
var x=document.getElementsByTagName("p");
for (var i=0;i<x.length;i++)
  {
      // do something with each paragraph
  }
```

مثلا برای دسترسی به سومین تگ <p>

```
var y=x[2];
```

روش دوم

سه خاصیت parentNode، firstChild و lastChild از ساختار درختی صفحه پیروی می کنند و امکان یک حرکت کوتاه را در این درخت فراهم می آورند.

به این کد Html دقت کنید:

```
<table>
  <tr>
    <td>hadi</td>
    <td>mohammad</td>
```

```
    <td>tavakoli</td>
  </tr>
</table>
```

در این کد اولین <td> بعنوان firstChild و آخرین <td> بعنوان lastChild برای گره <tr> می باشد. <tr> پدر همه ی گره های <td> است.

بیشترین کاربرد firstChild برای دسترسی به متن یک عنصر است.

```
var x=[a paragraph];

var text=x.firstChild.nodeValue;
```

از خاصیت parentNode اغلب برای تغییر در ساختار صفحه استفاده می شود. مثلا برای حذف گره ی با ID="maindiv":

```
var x=document.getElementById("maindiv");
x.parentNode.removeChild(x);
```

برای حذف یک گره از متد removeChild() استفاده می شود.

گره ریشه

گره ریشه در ساختار درختی DOM، گره document است این گره دو خاصیت خاص برای دسترسی به تگ ها داراست.

- document.documentElement

- document.body

خاصیت اول گره ریشه صفحه را برمی گرداند و خاصیت دوم برای دسترسی مستقیم به تگ <body> است. در این مثال کاربرد خاصیت body را مشاهده می کنید.

```
<html>

<head>
     <script type="text/javascript">
          function ChangeColor()
          {
             document.body.bgColor="yellow"
          }
     </script>
</head>

<body onclick="ChangeColor()">
       Click on this document!
</body>

</html>
```

اطلاعات گره

خاصیت های nodeName، nodeValue و nodeType شامل اطلاعات گره ها هستند.

هر نود دارای خاصیت های زیر است:

۱-nodeName

۲-nodeValue

۳- nodeType

خاصیت nodeName شامل نام گره است:

- خاصیت nodeName برای گره های element نام تگ است.

- خاصیت nodeName برای گره های attribute نام خاصیت است.

- خاصیت nodeName برای گره های text بصورت #text است.

- خاصیت nodeName برای گره های document بصورت #document است نام گره ها در nodeName با حروف بزرگ نگهداری می شود.

خاصیت nodeValue:

- خاصیت nodeValue برای گره های attribute مقدار خاصیت است.

- خاصیت nodeValue برای گره های text شامل متن است.

برای گره های element و document این خاصیت غیر فعال است.

خاصیت nodeType:

این خاصیت نوع گره را مشخص می کند. مقدارهای این خاصیت برای انواع گره های مختلف درجدول ۶-۲ آورده شده است.

Element type	NodeType
Element	1
Attribute	2
Text	3
Comment	8
Document	9

جدول ۶-۲ مقدار NodeType برای انواع گره ها

شی document

با استفاده از این شی می توان به هر عنصر داخل صفحه وب دسترسی داشت. و نیز با استفاده از خاصیت ها و متدهای این شی تغییراتی در صفحه انجام داد.

نام خاصیت	توضیح
AlinkColor	رنگ لینک زمانیکه کلیک شود
BgColor	رنگ پس زمینه صفحه
Cookie	مقداردهی ویا بازگرداندن تمام کوکی های که در ارتباط با این صفحه می باشند

FgColor	رنگ متن صفحه
Forms[array]	آرایه ی از فرم های موجود در صفحه
Images[array]	آرایه ی از تصویرهای موجود در صفحه
LastModified	یک فرمت رشته ی از آخرین تاریخ تغییر صفحه
LinkColor	رنگ اولیه یک لینک قبل از اینکه ملاقات شود
Links[array]	آرایه ی از لینک های موجود در صفحه
Location	آدرس url صفحه
Title	عنوان صفحه
URL	آدرس url صفحه
VlinkColor	رنگ یک لینک ملاقات شده

جدول ۷-۲ خاصیت های شی document

مهم ترین متدهای شی document عبارتند از:

- write
- writeln

این دو متدآنچه بعنوان پارامتر با آن ها می دهید را بر روی صفحه چاپ می کند.

```
document.write(exp1,exp2,exp3,....)

document.writeln(exp1,exp2,exp3,....)
```

مثال:

```
document.write('test text');

document.write("<h1>Hello World!</h1>");

document.write("Hello World! ","Hello You! ","<p
     style='color:blue;'>Hello World!</p>");
```

برای دسترسی به عنصرهای یک صفحه از طریق نام می توان از متد getElementsByName() و از طریق شناسه از متد getElementById() واز طریق نام تگ از متد getElementsByTagName() استفاده کرد.

مثال برای تغییر متن داخل یک textBox می توان به دو صورت زیر عمل کرد:

```
<input type=text name='txt1' id='tbox1' value='ok'>

document.getElementsByName('txt1').value = 'select name';

document.getElementById('tbox1').value = 'select id';
```

به مثال زیر توجه کنید:

```
<html>
<head>
<script type="text/javascript">
function getValue()
  {
  var x=document.getElementById("myHeader")
  alert(x.innerHTML)
  }
</script>
</head>
<body>
<h1 id="myHeader" onclick="getValue()">This is a header</h1>
<p>Click on the header to alert its value</p>
</body>
</html>
```

در این مثال برای دسترسی به محتوای متن از خاصیت innerHtml استفاده شد تا اطلاعات بـه فرمـت Html نیز دریافت شود.

```
<html>
<head>
<script type="text/javascript">
function getElements()
  {
  var x=document.getElementsByName("myInput");
  alert(x.length);
  }
</script>
</head>
<body>
<input name="myInput" type="text" size="20" /><br />
<input name="myInput" type="text" size="20" /><br />
<input name="myInput" type="text" size="20" /><br />
<br />
<input type="button" onclick="getElements()"
value="How many elements named 'myInput'?" />
</body>
</html>
```

در مثال بالا با متد getElementsByName("myInput") به یک مجموعه از تـگ هـا دسترسـی پیـدا کردیم.

```
<html>
<head>
<script type="text/javascript">
function getElements()
  {
  var x=document.getElementsByTagName("input");
```

```
  alert(x.length);
  }
</script>
</head>
<body>
<input name="myInput" type="text" size="20" /><br />
<input name="myInput" type="text" size="20" /><br />
<input name="myInput" type="text" size="20" /><br />
<br />
<input type="button" onclick="getElements()"
value="How many input elements?" />
</body>
</html>
```

در این مثال با استفاده از متد getElementsByTagName("input") به یک مجموعه از تگ ها براساس نام یکسان دسترسی پیدا کردیم.

برای دسترسی به مقدار انتخاب شده در یک <select> باید یک <form> را بعنوان پدر <select> قرار دارد بصورت زیر:

```
<form name='form1'>
     <select id="select1">
          <option value="0" >نام</option>
          <option value="1" > موضوع</option>
     </select>
</form>
```

برای دسترسی به مقدار از دستور زیر استفاده می کنیم:

```
document.form1.select1.options[document.form1.select1.
     selectedIndex].value;
```

خاصیت innerText و innerHTML

مایکروسافت با عرضه IE4 دو خاصیت جدید برای تمامی عناصر DOM معرفی کرد. کاربرد این دو خاصیت برای تغییر عناصر DOM می باشد ولی خیلی ساده تر از قبل.

خاصیت innerText برای تغییر متن (text) مابین تگ شروع و پایان یک عنصر DOM می باشدبه مثال زیر توجه کنید:

می خواهیم به یک تگ در صفحه، متنی را اضافه کنیم:

```
<html>
<head>
     <title> </title>
</head>
<body>
     <div id="div1" > </div>
</body>
</html>
```

یک راه حل با استفاده از دستورات ساختاری DOM

```
var  dtext = document.createTextNode(“New text for the div.”);

document.getElementById('div1').appendChild(dtext);
```

اما راه حل دوم استفاده از خاصیت innerText

```
document.getElementById('div1').innerText = “New text for the
div.”;
```

خاصیت innerText به تگ های Html حساس نیست و آن ها را جزء متن به حساب می آورد. برای مثال می خواهیم این دستور Html داخل یک بنویسیم:

```
<h1>hadi</h1>
```

اگر این کار را با innerText انجام دهیم متن های تگ <h1> نیز بر روی صفحه نمایش داده می شوند.

```
document.getElementById('div1').innerText ='<h1>hadi</h1>';
```

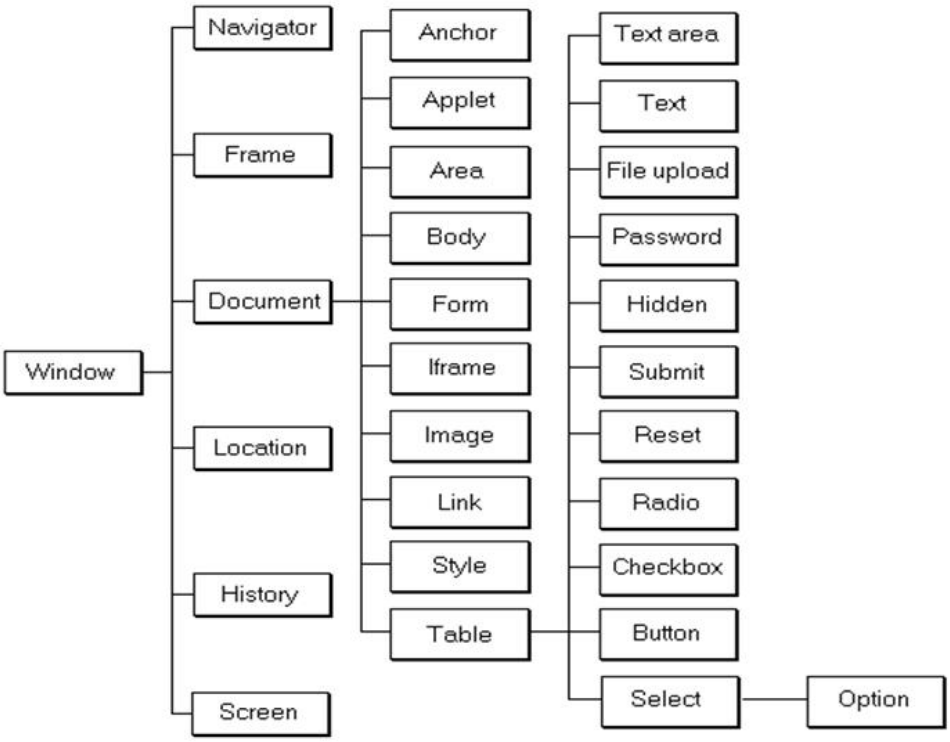
اما خاصیت دیگر innerHTML حساس به تگ های Html می باشد و این مشکل را حل می کند.

```
document.getElementById('div1').innerHTML='<h1>hadi</h1>';
```

اگر می خواستیم این کار را با دستورات DOM بنویسیم بدین گونه می شد:

```
var hText = document.createElement(“h1”);
hText.appendChild(document.createTextNode(“hadi”));

document.getElementById('div1').appendChild(hText);
```

در پایان به این نکته اشاره می کنیم که در تکنولوژی Ajax برای نوشتن برنامه ها از کد جاوااسکریپت سمت سرور استفاده می شود.



شکل ۳-۲ نمودار رابطه ای اشیا مرورگرها (DOM)

فصل سوم : شروع کار با Ajax

نیروی پیشبرنده پشت سر Ajax، ایجاد ارتباط بین کلاینت و سرور است. قبلا این ارتباط محدود به توسعه زبانهای سمت سرور مانند perl و CGI بود، تکنولوژیهای جدید مانند asp.net، PHP و jsp تکنیک های را برای آمیختن برنامه های سمت سرور و کلاینت برای علاقمندان به ایجاد برنامه های وب فراهم آوردند، اما آن ها فاقد تکنیک های سمت کلاینت مانند جاوااسکریپت بودند. اما اکنون آونگ در جهت دیگری تاب می خورد، توسعه دهندگان سمت سرور نیاز به فهمیدن چیزهای زیادی درباره تکنیک های سمت کلاینت دارند تا بتوانند راه حل Ajax را به کار ببرند.

شروع با HTTP

بهترین روش درک Ajax درک HTTP است، HTTP پروتکلی برای ارسال صفحات وب، تصویر و انواع دیگر فایل ها بر روی اینترنت به سمت مرورگر شما و یا برعکس (سمت سرور) است. زمانی که شما یک URL را در مرورگر تایپ می کنید، یک HTTP:// به آدرس شما اضافه می شود که نشان می دهد شما می خواهید از HTTP برای دسترسی به اطلاعات محل ذکر شده استفاده کنید. (اغلب مرورگرها تعداد دیگری از پروتکل ها مانند FTP را نیز پشتیبانی می کنند)

HTTP از دوقسمت تشکیل شده است: درخواست (request) و پاسخ (Response). زمانی که شما در مرورگر وب یک URL را تایپ می کنید، مرورگر از طرف شما یک درخواست ایجاد کرده و آن را ارسال می کند این درخواست شامل Url ی که شما تایپ کرده اید و برخی اطلاعات دیگر که به مرورگر مربوط می شود است. سرور این درخواست را دریافت و به سمت عقب (سمت مرورگر) پاسخی را ارسال می کند. پاسخ شامل اطلاعاتی درباره محل تعیین شده داده در URL است. این وظیفه مرورگر است که پاسخ را تغییر داده و آن را در صفحه وب نمایش دهد.

درخواست های HTTP

فرمت یک درخواست HTTP به صورت زیر است:

```
<request-line>
<headers>
<blank line>
[<request-body>]
```

در یک درخواست HTTP، خط اول بایستی یک خط درخواست باشد تا نوع درخواست، منابع در دسترس، و نسخه HTTP که مورد استفاده قرار می گیرد را نشان دهد. خط بعد، یک قطعه header است که اطلاعات اضافی که توسط سرور مورد استفاده قرار می گیرد را نشان می دهد. بعد از Header یک خط خالی وجود داردکه می تواند با اطلاعات اضافی پر شود.

انواع مختلفی از درخواست وجود دارد که در HTTP تعریف می شود اما دو نوع مورد توجه توسعه دهندگان Ajax درخواست های نوع: GET و POST می باشند. هر زمانی که شما یک URL را در مرورگر وب تایپ کنید، برای آن URL، یک درخواست GET به سمت سرور ارسال می کنید، که بطور کلی به سرور می گوید که منابع را بگیرد و آن را به سمت عقب ارسال کند. اینجا یک مثال از چیزی که درخواست GET برای آدرس www.sample.com می تواند نشان دهد آورده شده است:

```
GET / HTTP/1.1
Host: www.sample.com
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US;
     rv:1.7.6)
Gecko/20050225 Firefox/1.0.1
Connection: Keep-Alive
```

خط اول نشان می دهد که نوع درخواست GET است. قسمت دوم (HTTP) از آن خط نشان می دهد که درخواست برای یک Domain است انتهای خط درخواست معلوم می کند که از نسخه 1.1، HTTP استفاده می شود. (نسخه قبلی آن 1.0 بوده است) اما درخواست به کجا ارسال شده است؟ این موضوع در خط دوم نشان داده شده است. خط دوم اولین Header در درخواست است. هدر Host، مقصد درخواست را نشان می دهد. ترکیب Host با اسلش جلو(/) در خط اول به سرور اعلام می کند که درخواست برای www.sample.com است. (هدر Host در Http1.1 لازم است در نسخه قبل از آن لازم نبود.)
خط سوم شامل User-Agent است، که هم اسکریپتهای سمت سرور و هم کلاینت به آن دسترسی دارند و در برخی مرورگرها برای یافتن خطاهای منطقی لازم است. این اطلاعات توسط مرورگر مورد استفاده شما ایجاد می شوند. (در این مثال Firefox) و به صورت اتوماتیک در هر درخواست برای سرور ارسال می شوند. خط آخر Connection است که برای نمونه بالا با Keep-Alive مقداردهی شده است. (می تواند با مقادیر دیگری نیز ست شود که خارج از بحث کتاب است.) توجه کنید که یک خط خالی بعد از آخرین Header وجود دارد با اینکه هیچ بدنه درخواستی وجود ندارد ولی خط خالی لازم است. اگر یک زیر دمین مانند www.sample.com/books را درخواست کنید درخواست شما به صورت زیر خواهد بود:

```
GET /books/ HTTP/1.1
Host: www.sample.com
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US;
     rv:1.7.6)
Gecko/20050225 Firefox/1.0.1
Connection: Keep-Alive
```

توجه داشته باشید که تنها خط اول تغییر کرده است که شامل قسمتی است که بعد از www.sample.com در URL آمده است.

ارسال پارامتر برای Get نیازمند این است که اطلاعات خارجی را به URL اضافه کنید:

```
URL ? name1=value1&name2=value2&..&nameN=valueN
```

این اطلاعات که رشته query نامیده می شوند، در خط درخواست از یک درخواست HTTP تکرار می شوند:

```
GET /books/?name=root%20Directory HTTP/1.1
Host: www.sample.com
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US;
     rv:1.7.6)
Gecko/20050225 Firefox/1.0.1
Connection: Keep-Alive
```

توجه کنید که متن "root Directory"، هنگامی که به عنوان پارامتر با URL ارسال می شود، بایستی رمزگذاری شود، به همین خاطر جای خالی با 20% جایگزین می شود. به این امر encode کردن URL گفته می شود و در قسمت های مختلف HTTP استفاده می شود. (جاوااسکریپت یک تابع داخلی دارد که عملیات رمزگذاری و رمزگشائی URL را انجام می دهد. درباره این تابع بعداً در این فصل بحث خواهد شد.) مقادیر فیلدهای مختلف با یک آمپرسند "&" از هم جدا می شوند.

بسیاری از تکنولوژیهای سمت سرور به صورت خودکار بدنه درخواست را رمزگشائی می کنند و دسترسی به این مقادیر را به روش های گوناگونی ممکن می سازند. البته، این مربوط به سرور است که با این داده ها چه کار کند. از سوی دیگر، درخواست POST، اطلاعات را به بدنه درخواست اضافه می کند. مثلا، زمانی که شما یک فرم online را پر و آن را submit می کنید، داده ها به صورت درخواست POST ارسال می شوند.

اینجا نمونه ای از درخواست POST آمده است:

```
POST / HTTP/1.1
Host: www.sample.com
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US;
     rv:1.7.6)
Gecko/20050225 Firefox/1.0.1
Content-Type: application/x-www-form-urlencoded
Content-Length: 40
Connection: Keep-Alive

name=root%20Directory&OS=Unix
```

بایستی به چندین تفاوت مهم بین درخواست POST و درخواست GET توجه کنید. اول اینکه خط درخواست بجای GET با POST شروع می شود، که نوع درخواست را نشان می دهد. شما می توانید ببینید که هدرهای Host و User-Agent همراه با دو مورد جدید سرجای خودشان هستند. هدر Content-Type نشان می دهد که بدنه درخواست چگونه کد شده است. مرورگر همیشه داده POST را به صورت application/x-www-form-urlencoded رمزگذاری می کند که رمزگذاری ساده URL از نوع MIME است. هدر Content-Length طول بایت های بدنه درخواست را نشان می دهد. پس از Connection یک خط خالی در بدنه درخواست وجود دارد، که در بیشتر مرورگرها یک جفت <نام-

مقدار> می سازد که در اینجا name برابر با root Directory و OS با Unix برابر است. شما می توانید ببینید که این فرمت شبیه پارامترهای رشته ای Query در URL است. همانطور که قبلا گفتیم، انواع دیگر درخواست HTTP وجود دارد، ولی آن ها نیز از فرمت های شبیه POST و GET تبعیت می کنند. گام بعدی نگاه به چیزی است که سرور به سمت عقب (سمت مرورگر)، بعنوان پاسخ درخواست HTTP ارسال می کند.

پاسخ HTTP

فرمت یک پاسخ HTTP، که خیلی شبیه درخواست آن است، در زیر آمده است:

```
<status-line>
<headers>
<blank line>
[<response-body>]
```

همانطور که می بینید، تنها تفاوت پاسخ با یک درخواست این است که خط اول شامل اطلاعات وضعیت به جای اطلاعات درخواست است. خط وضعیت درباره منابع درخواست شده بوسیله یک کد حالت اطلاع رسانی می کند. اینجا یک پاسخ ساده HTTP آمده است:

```
HTTP/1.1 200 OK
Date: Sat, 31 Dec 2005 23:59:59 GMT
Content-Type: text/html;charset=ISO-8859-1
Content-Length: 122

<html>
<head>
     <title>Wrox Homepage</title>
</head>
<body>
     <!-- body goes here -->
</body>
</html>
```

دراین مثال خط وضعیت یک کد وضعیت HTTP، 200 و یک پیغام ok است. خط وضعیت همیشه شامل کد وضعیت و یک پیغام کوتاه است. مهمترین کدهای حالات در زیر آمده اند:

200 (ok): منبع پیدا شده است و همه چیز خوب می باشد.

300 (NOT MODIFIED): منبع نتوانسته بعد درخواست آخر اصلاح شود. این کد بیشتر اوقات برای مکانیزم های cache مرورگر به کار می رود.

401 (UNAUTHORIZED): کلاینت حق دسترسی به منبع را ندارد. اغلب باعث می شود که مرورگر برای login به سرور نام کاربری و پسورد را درخواست کند.

403 (FORBIDDEN): کلاینت در دسترسی به منابع با خطا مواجعه شده است. این نوع زمانی اتفاق می افتد که شما بعد از 401 نتوانید با یک اسم کاربری صحیح login کنید.

404 (NOT FOUND): منبع در محل مورد نظر وجود ندارد.

در خط وضعیت، هدرهای دیگری نیز موجودند. بطور نمونه، سرور یک هدر Date را برمی گرداند که زمان و تاریخ ایجاد پاسخ را نشان می دهد. (سرور اطلاعاتی در مورد خودش نیز ارسال می کند، اگر چه مورد نیاز نباشند) دو هدر بعدی باید آشنا باشند، آن ها Content-Type و Content-Length که در درخواست POST نیز استفاده می شدند، هستند. در این مثال، هدر Content-Type نوع MIME را برای text/html با یک رمزگذاری از نوع ISO-8859-1 تعیین می کند. (که استاندارد منابع انگلیسی ایالات متحده است.) بدنه پاسخ شامل یک کد HTML ساده است. این همان چیزی است که مرورگر به کاربر نشان خواهد داد. توجه کنید که اشاره ای به نوع درخواست که پاسخ برای آن ارسال می شود نیست؛ اگر چه، این مطلب اثری بر روی سرور ندارد. این مطلب مربوط به کلاینت است که بداند چه نوع داده ای برای هر نوع درخواست بایستی به سمت عقب برگردانده شود تا تصمیم بگیرد چگونه از داده بایستی استفاده کند.

درخواست XMLHttp

XMlHttp ابتدا در IE5 با یک ActiveX معرفی شد. شی XMlHttp توسعه دهندگان را به ایجاد درخواست Http از هر کجای یک برنامه قادر می سازد. برای این درخواست یک XML به عنوان پاسخ برگردانده می شود، به همین خاطرXMlHttp یک روش ساده برای دسترسی به اطلاعات از طریق XML را مهیا می کند.

Mozila نیز بعداً XMlHttp را در مرورگرهای خود (مانند Firefox) قرار داد. بعد از آن مرورگرهای Safari (بعد از نسخه 1,2) و Opera (نسخه 7,6 به بالا) از آن حمایت کردند. امروزه، تقریبا اکثر مرورگرهای مطرح از این نوع درخواست حمایت می کنند. باید توجه داشت که XMlHttp استاندارد W3C نیست، به همین خاطر رفتارهای متفاوتی از آن را می توان در مرورگرهای مختلف مشاهده کرد، ولی اکثر متدها و صفات مهم در همه آن ها به یک صورت حمایت می شوند.

بایستی به این موضوع اشاره کرد که اگر کاربران با مرورگرهای قدیمی تر به وب شما دسترسی دارند بایستی در انتخاب این گزینه تجدید نظر کنید.

ایجاد شی XMlHttp

برای ارسال درخواست و پردازش پاسخ بایستی ابتدا یک شیXMLHttpRequest با استفاده از جاوااسکریپت ایجاد کنید. با توجه به این موضوع که XMLHttpRequest استاندارد W3C نیست، می بایستی از روش های که مرورگرها از آن حمایت می کنند به ایجاد شی مبادرتت کرد. IE، بوسیله یک شی ActiveX، شی XMLHttpRequest را ایجاد می کند در حالی که مرورگرهای دیگر مانند Firefox، Opera،Safari، آن را به صورت یک شی جاوااسکریپتی پیاده سازی می کنند. بخاطر این موضوع، کد

جاوااسکریپت بایستی شامل هر دو روش ایجاد یک نمونه XMLHttpRequest با استفاده از تکنیک ActiveX و شی جاوااسکریپتی باشد.

برای انجام این کار احتیاجی به دانستن نوع مرورگر نیست، تنها چیزی که بایستی انجام بدهید این است که چک کنید آیا مرورگر از ActiveX حمایت می کند یا نه؟ در صورت پشتیبانی کردن، شی XMLHttpRequest را با استفاده از تکنیک ActiveX در غیر اینصورت از شی محلی جاوااسکریپتی برای این منظور استفاده می کنید. برای اینکه در IE با استفاده از ActiveX بخواهید یک شی XMLHttpRequest را نمونه سازی کنید، بایستی به صورت زیر عمل کنید.

```
var xmlHttp = new ActiveXObject(iEVersion[i]);
```

مشکلی که اینجا وجود دارد، وجود چندین نسخه از کتابخانه های MSXML است. چون هر نسخه سرعت و ثبات خود را دارد، باید مطمئن شویدکه از جدیدترین نسخه که ماشین کاربر از آن حمایت می کند استفاده می کنید یا نه؟ بدین منظور باید نسخه های موجود را بررسی کنید.

نسخه های موجود MSXML به قرار زیر می باشد:

```
- Microsoft.XMLHttp
- MSXML2.XMLHttp
- MSXML2.XMLHttp.3.0
- MSXML2.XMLHttp.4.0
- MSXML2.XMLHttp.5.0
```

متاسفانه، تنها راه تشخیص بهترین نسخه برای استفاده، سعی در ایجاد هر یک از این نسخه ها می باشد. بخاطر اینکه این نسخه ها کنترل ActiveX هستند، ناتوانی در ایجاد هر یک از آن ها منجر به ایجاد یک خطا خواهد شد، این مورد به این معنی است که شما بایستی از بلوک try…catch استفاده کنید. در اینصورت با ایجاد یک خطا در یک ActiveX سعی در ایجاد یک نمونهXMLHttpRequest با استفاده از ActiveX بعدی می کند. ایجاد شی XMLHttp در مرورگرهای دیگر بسیار ساده می باشد، تنها کافی است از کد زیر برای این منظور استفاده کنید:

```
var xmlHttp = new XMLHttpRequest();
```

نتیجه کار به صورت زیر خواهد بود:

```
function createXMLHttp() {
     if (typeof XMLHttpRequest !="undefined"){
          return new XMLHttpRequest()
     }else if (window.ActiveXObject){

          var iEVersion = [ "MSXML2.XMLHttp.5.0",
               "MSXML2.XMLHttp.4.0","MSXML2.XMLHttp.3.0",
               "<MSXML2.XMLHttp","Microsoft.XMLHttp"
               ];

          for (var i=0; i< iEVersion.length; i++){
```

```
                try{
                     var xmlHttp = new
                           ActiveXObject(iEVersion[i]);
                     return xmlHttp;

                }catch(oError){
                          هیچ عملی انجام نمی گیرد//
                }

           }
      }
      throw new Error("XMLHttp object could be created.");
}
```

تابع createXMLHttp ابتدا با استفاده از عملگر typeof چک می کند که آیا کلاس XMLHttpRequest تعریف شده است یا نه؟(مرورگر ازآن پشتیبانی می کند یا نه؟)

اگر XMLHttpRequest موجود بود، از آن برای ایجاد شی XMLHttp استفاده می کند، در غیر اینصورت چک می کند که آیا کلاس ActiveXObject موجود است یا نه؟ در صورت وجود، دسته ActiveX ها را در یک آرایه قرار می دهد. (جدیدترین نسخه ابتدای آرایه قرار می گیرد.) با استفاده از یک حلقه تکرار ActivX ها را از آرایه خوانده و سعی می کند یک نوع XMLHttp ایجاد کند اگر این کار امکان نداشت، عبارت catch از ایجاد خطا و توقف عملیات جلوگیری می کند، سپس مورد بعدی آرایه مورد آزمایش قرار می گیرد. زمانی که یک شی ایجاد شد، با برگرداندن آن شی عملیات به پایان می رسد، در غیر اینصورت یک خطا با استفاده از دستور throw ایجاد می کند که نشان می دهد عملیات با شکست مواجه شده است.

متدها و صفات

اکنون توانسته اید یک شی XMLHttp ایجاد کنید. همانطور که دیدید برای ایجاد این شی در مرورگرهای مختلف روشهای مختلفی را اعمال کردیم، به همین خاطر تفاوتهایی نیز در متدها و صفات این شی در مرورگرهای مختلف خواهیم دید. می توانید صفات شی XMLHttpRequest ایجاد شده در IE را در جدول ۱-۳ و متدهای آن را در جدول ۲-۳، مشاهده کنید. صفات این شی در مرورگرهای Mozila،netScape و Firefox در جدول ۳-۳ و متد های آن در جدول ۴-۳ آمده است.

Apple safari هنوز متدها و صفات کامل برای شی XMLHttpRequest را اعلام نکرده است، اما مجموعه ای از صفات عمومی مورد استفاده را اعلام کرده است، که در جدول ۵-۳ آمده است و متدهای عمومی، نیز در جدول ۶-۳ آمده است.

Property	Means	Read/write
onreadystatechange	Holds the name of the event handler that should be called when the value of the readyState property changes	Read/write
readyState	Holds the state of the request	Read-only
responseBody	Holds a response body, which is one way HTTP responses can be returned	Read-only
responseStream	Holds a response stream, a binary stream to the server	Read-only
responseText	Holds the response body as a string	Read-only
responseXML	Holds the response body as XML	Read-only
status	Holds the HTTP status code returned by a request	Read-only
statusText	Holds the HTTP response status text	Read-only

جدول ۱-۳ خاصیت های شی XMLHttpRequest در IE

Method	Means
abort	Aborts the HTTP request
getAllResponseHeaders	Gets all the HTTP headers
getResponseHeader	Gets the value of an HTTP header
open	Opens a request to the server
send	Sends an HTTP request to the server

setRequestHeader	Sets the name and value of an HTTP header

جدول ۲-۳ متدهای شی XMLHttpRequest در IE

Property	Means	Read/write
channel	Holds the channel used to perform the request	Read-only
readyState	Holds the state of the request	Read-only
responseText	Holds the response body as a string	Read-only
responseXML	Holds the response body as XML	Read-only
status	Holds the HTTP status code returned by a request	Read-only
statusText	Holds the HTTP response status text	Read-only

جدول ۳-۳ خاصیت های شی XMLHttpRequest در مرورگرهای نظیر firefox

Method	Means
abort	Aborts the HTTP request
getAllResponseHeaders	Gets all the HTTP headers
getResponseHeader	Gets the value of an HTTP header
openRequest	Native (non-script) method to open a request
overrideMimeType	Overrides the MIME type the server returns

جدول ۴-۳ متدهای شی XMLHttpRequest در مرورگرهای نظیر firefox

Property	Means	Read/write
onreadystatechange	Holds the name of the event handler that should be called when the value of the readyState property changes	Read-only
readyState	Holds the state of the request	Read-only
responseText	Holds the response body as a string	Read-only
responseXML	Holds the response body as XML	Read-only
status	Holds the HTTP status code returned by a request	Read-only
statusText	Holds the HTTP response Read-only status text	Read-only

جدول۵-۳ خاصیت های شی XMLHttpRequest برای safari

Method	Means
abort	Aborts the HTTP request
getAllResponseHeaders	Gets all the HTTP headers
getResponseHeader	Gets the value of an HTTP header
open	Opens a request to the server
send	Sends an HTTP request to the server
setRequestHeader	Sets the name and value of an HTTP header

جدول ۶-۳ متدهای شی XMLHttpRequest برای safari

اجازه بدهید به متدهای مهمی که در اینجا اشاره شد نگاهی داشته باشیم:

متد open(): برای استفاده از شی XMLHttpRequest با استفاده از متد open() بر روی شی پیکربندی اولیه را انجام می دهید. اینجا طریقه استفاده از متد open را نشان می دهیم (توجه کنید که آیتم های بین براکت ها، [و]، موارد اختیاری می باشند)

```
open(string method, string URL [,boolean asyncFlag [, string
username [,string  password]]])
```

این متد دارای ۲ پارامتر اصلی و۳ پارامتر اختیاری می باشد. جدول ۷-۳ این پارامترها را توضیح می دهد:

پارامتر	معنی آن
method	متد HTTP که برای ایجاد ارتباط در open استفاده می شود، مانند GET, POST, PUT, HEAD, or PROPFIND
URL	آدرسی که از آنجا درخواست صورت می گیرد
asyncFlag	مقدار بولی که مشخص غیر همزمان بودن درخواست را مشخص می کند
username	نام کاربری
password	کلمه عبور

جدول ۷-۳ پارامترهای متد open

توجه کنید که asyncFlag به صورت پیش فرض true می باشد. True بودن این پارامتر لازمه روش ajax می باشد، در صورت False بودن آن هدف استفاده از XMLHttpRequest زیر سوال می رود. البته ممکن است این مقدار را در برخی مواقع مانند اعتبارسنجی ورودی های کاربر مفید باشد.

متد send(): در حقیقت این متد درخواست را ارسال می کند. اگر درخواست به صورت asynchronous تعریف شده باشد، این متد فوراً به اتمام میرسد، در غیر اینصورت برای رسیدن پاسخ منتظر می ماند. این متد به صورت زیر تعریف می شود:

```
send([content])
```

پارامتر اختیاری می تواند یک نمونه از شی DOM باشد و یا یک رشته یا یک استریم ورودی باشد، در غیر اینصورت مقدار null می گیرد. این مقدار به عنوان قسمتی از درخواست ارسال می شود.

متد setRequestHeader : این متد مقدار header درخواست را تنظیم می کند. این متد به صورت زیر تعریف می شود:

```
setRequestHeader(string header, string value)
```

یک رشته برای تنظیم header و یک رشته که نوع محتوا را نشان می دهد را به عنوان پارامتر دریافت می کند. این متد پس از متد open مقدار دهی می شود.

مثلا اگر بخواهید داده هایتان را به صورت Xml ارسال کنید Header درخواستتان را با نوع محتوای "text/xml" تنظیم می کنید:

```
xMLHttp.setRequestHeader(“Content-Type”, “text/xml”);
```

سپس می توانید مستقیما xml تان را به صورت مستقیم با استفاده از متد send ارسال کنید، که می تواند چیزی شبیه این باشد:

```
xMLHttp.send(“<doc><name>limit</name><data>5</data></doc>”);
```

متد abort(): از ارسال درخواست جلوگیری می کند.

متد getAllResponseHeaders(): این متد یک رشته محتوی هدرهای یک پاسخ برای یک درخواست Http را برمی گرداند. هدرها شامل Content-Length, Date و URL هستند.

اولین برنامه ساده Ajax

اکنون که با متد های XMLHttp آشنا شدید اجازه بدهید با یک مثال ساده، هم یک برنامه با روش Ajax ایجاد کنیم و هم اینکه صفات شی XMLHttp را توضیح دهیم.

در این مثال از کاربر خواسته می شود که کلید روی صفحه وب را کلیک کند، پس از فشردن کلید، برنامه بوسیله تکنیک Ajax اطلاعات را از سرور خوانده و آن ها را در صفحه وب نمایش می دهد. این عملیات بدون بارگذاری مجدد صفحه کاربر انجام خواهد شد.

سعی ما این است که طریقه ایجاد برنامه را توضیح دهیم. در ابتدای کار یک صفحه وب که شامل یک شی Button و شی Divمی باشد را ایجاد می کنیم.

```
<html>
 <head>
      <title>Untitled Document</title>
 </head>
 <body>
      <input type="button" value="Display Message" >
      <div id="targetDiv">
           <p>Display Message Hear.</p>
      </div>

 </body>
</html>
```

این کد، صفحه با یک کلید و یک متن بر روی آن را ایجاد خواهد کرد. اما کلیک بر روی کلید منجر به هیچ عملی نخواهد شد. در onClick دکمه قسمت کدی را اضافه می کنیم که با فشردن کلید عملیات گفته شده را انجام دهد.کد زیر کد تصحیح شده قسمت قبل می باشد:

```
<html>
 <head>
      <title>Untitled Document</title>
```

```
 </head>
 <body>
<input type="button" value="Display Message"  onclick =
           "getData('Data.txt','targetDiv')">
     <div id="targetDiv">
          <p>Display Message Hear.</p>
     </div>

 </body>
</html>
```

همانطور که می بینید تابع getData دو پارامتر رشته ای می گیرد: نام فایل رشته ی Data.txt، و محلی که محتوای این فایل نمایش داده می شودtargetDiv. فشردن کلید روی صفحه وب جز یک پیغام خطا عکس العمل دیگری به همراه نخواهد داشت.

همانطور که قبلا اشاره شد برای کار با برنامه های Ajax احتیاج به یک شی XMLHttp می باشد، قبلا طریقه ی ایجاد شی XMLHttp را توضیح دادیم، اکنون بدون توضیح اضافه ی تابع createXMLHttp که این وظیفه را بر عهده داشت به برنامه اضافه می کنیم.

```
<head>
     <title>First Project of Ajax</title>
     <script language = "javascript">
          function createXMLHttp() {
             if (typeof XMLHttpRequest !="undefined"){
               return new XMLHttpRequest()
             }else if (window.ActiveXObject){

               var iEVersion = [ "MSXML2.XMLHttp.5.0",
                    "MSXML2.XMLHttp.4.0","MSXML2.XMLHttp.3.0",
                         "<MSXML2.XMLHttp","Microsoft.XMLHttp"
                    ];
               for (var i=0; i< iEVersion.length; i++){
                  try{
                    var xmlHttp = new
                            ActiveXObject(iEVersion[i]);
                    return xmlHttp;
                  }catch(oError){
                            //هیچ عملی انجام نمی گیرد
                  }

               }
             }
             throw new Error("XMLHttp object could be
                            created.");
          }
     </script>
 </head>
```

اکنون نیازمند این هستید که از طریق تابع بالا یک نمونه شی XMLHttp ایجاد کنید. و از این نمونه شی برای انجام عملیات مورد نظر استفاده کنید.

اکنون سوالی که مطرح می شود این است که چگونه از این شی برای خواندن و نمایش اطلاعات استفاده کنیم؟ همه این عملیات در تابع getData پیاده سازی می شود.

```
<script language = "javascript">
          ..
          ..
          ..
          ..

          function getData(dataSource, divID){
               ..
               ..
               ..
          }
</script>
```

در این تابع ابتدا یک متغیر از نوع شی XMLHttp ایجاد می کنیم.

```
<script language = "javascript">
          ..
          ..
          ..
          ..

          function getData(dataSource, divID){

               xmlHttp = createXMLHttp();
               ..
               ..
               ..
          }
</script>
```

اکنون یک نمونه از شی XMLHttp در متغیر xmlHttp دارید، اکنون با متد open همانطور که قبلا توضیح داده شد شی xmlHttp را برای استفاده از URL مورد نظر پیکربندی می کنیم. اغلب از متد GET زمانی که می خواهید اطلاعاتی را بازیابی کنید استفاده می شود و از متد POST زمانی استفاده می شود که بخواهید تعدادی داده برای سرور ارسال کنید. در این مثال با استفاده از GET برای خواندن فایل Data.txt به صورت زیر عمل می کنیم:

```
<script language = "javascript">
          ..
          ..
```

```
        ..
        ..

        function getData(dataSource, divID){

            xmlHttp = createXMLHttp();
            xmlHttp.open("GET", dataSource);
            ..
            ..
            ..
        }
</script>
```

این کد شی xmlHttp را با استفاده از URl ای که مشخص نموده اید (در اینجا Data.txt) پیکر بندی می کند، اما هنوز به صورت واقعی به فایل مورد نظر وصل نشده است. (اگر از یک دامین خاص استفاده می کنید آدرس URL مورد نظر بایستی شامل آدرس دامین نیز باشد)

به صورت پیش فرض این ارتباط به صورت asynchronous می باشد، که بدین معنی است که برنامه منتظر خواندن فایل و دانلود آن نمی ماند. (شما می توانید از گزینه سوم asyncFlag، برای اجرای synchronous برنامه استفاده کنید، که به این معنی خواهد بود که برنامه همه فعالیت های خود را تا زمان دانلود فایل متوقف کند.)

حال فرض کنیم شما اطلاعاتتان را برای سرور ارسال کرده اید و سرور به درخواست شما پاسخ داده و فایل Data.txt را برایتان ارسال کرده است.اکنون چگونه بایستی از آماده بودن اطلاعاتتان برای دانلود آگاه شوید تا بتوانید از آن ها استفاده کنید؟

شی XMLHttp یک property (صفت) با نام onreadystatechange دارد که برای این منظور تدارک دیده است. این property اجازه مدیریت عملیات بارشدن asynchronous را به شما می دهد. اگر شما نام یک تابع را برای این property مشخص کنید، هر زمان که تغییری در اسکریپت شما ایجاد شد این property فراخوانی می شود، و آن نیز تابع شما را فراخوانی می کند. البته چون این تابع در موارد دیگر کاربردی ندارد می توانید به صورت زیر آن را داخل تابع getData ایجاد کنید:

```
<script language = "javascript">
        ..
        ..
        ..
        ..

        function getData(dataSource, divID){

            xmlHttp = createXMLHttp();
            xmlHttp.open("GET", dataSource);

            xmlHttp.Onreadystatechange = function(){
                ..
```

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

این تابع جدید هر زمان که در شی xmlHttp تغییری ایجاد شود فراخوانی خواهد شد. شما بایستی دو property از این شی را بررسی کنید: صفات readyState و status. readyState که وضعیت لود شدن داده ها را اعلام می کند، اینجا مقادیر ممکنی که readyState می تواند داشته باشد را می آوریم:

- 0 uninitialized (مقدار دهی نشده)
- 1 loading (در حال لود)
- 2 loaded (لود شده)
- 3 interactive (در حال تبادل داده)
- 4 complete (کامل)

توجه کنید که 4 به معنی این است که داده های شما کاملا دانلود شده اند.

صفت status: در اینجا برخی مقادیر ممکن برای status آمده است (توجه کنید که 200 بدین معنی است که همه چیز خوب پیش می رود.)

- 200 OK
- 201 Created
- 204 No Content
- 205 Reset Content
- 206 Partial Content
- 400 Bad Request
- 401 Unauthorized
- 403 Forbidden
- 404 Not Found
- 405 Method Not Allowed
- 406 Not Acceptable
- 407 Proxy Authentication Required
- 408 Request Timeout
- 411 Length Required
- 413 Requested Entity Too Large
- 414 Requested URL Too Long
- 415 Unsupported Media Type
- 500 Internal Server Error
- 501 Not Implemented
- 502 Bad Gateway
- 503 Service Unavailable
- 504 Gateway Timeout
- 505 HTTP Version Not Supported

برای اطمینان از دانلود کامل اطلاعاتتان و خوب بودن همه چیز، باید مطمئن شوید که صفت readyState برابر 4 و صفت status با 200 برابر است. اینجا چگونگی پیاده سازی آن را نمایش می دهیم:

```
<script language = "javascript">
          ..
          ..
          ..
          ..

          function getData(dataSource, divID){

               xmlHttp = createXMLHttp();
               xmlHttp.open("GET", dataSource);

               xmlHttp.Onreadystatechange = function(){
               if(xmlHttp.readyState == 4 &&
                    xmlHttp.status == 200){

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

حالا مرورگر اطلاعات را بدست آورده است، ولی شما چگونه می توانید به این اطلاعات دسترسی داشته باشید.

برای گرفتن داده ها از شی XMLHttp دو روش وجود دارد:

- اگر شما با داده های دریافتی می خواهید همانند متن استاندارد رفتار کنید، شما می توانید از صفت responseText شی XMLHttp استفاده کنید
- اگر داده شما بصورت XML می باشد، می توانید از صفت responseXML استفاده کنید.

در این مثال فایل Data.text محتوی متن می باشد، به همین خاطر شما بایستی از صفت responseText استفاده کنید. برای اینکه داده دانلود شده را در صفحه وبتان در محلی که می خواهید ببینید، می توانید آن را در یک Div که با ID،targetDiv که در صفحه موجود می باشد و به عنوان پارامتر به تابع getData ارسال شده، نمایش دهید:

```
<script language = "javascript">
          ..
          ..
          ..
          ..

          function getData(dataSource, divID){
```

```
                xmlHttp = createXMLHttp();
                xmlHttp.open("GET", dataSource);

                xmlHttp.Onreadystatechange = function(){
                if(xmlHttp.readyState == 4 &&
                     xmlHttp.status == 200){

                     obj.innerHTML = xmlHttp.responseText;
                }
                }
                ..
                ..
           }
</script>
```

اکنون شما کدتان را برای مدیریت پاسخ سرور، زمانی که پاسخ را دریافت کردید آماده کرده اید. اما اکنون چگونه به صورت واقعی برای دریافت این پاسخ به سرور وصل شوید؟ برای این کار از متد send استفاده می کنید. این متد قبلا توضیح داده شد، زمانی که از متد GET استفاده می کنید مقدار null برای سرور ارسال می کنید:

```
<script language = "javascript">
           ..
           ..
           ..
           ..

           function getData(dataSource, divID){

                xmlHttp = createXMLHttp();
                xmlHttp.open("GET", dataSource);

                xmlHttp.Onreadystatechange = function(){
                if(xmlHttp.readyState == 4 &&
                     xmlHttp.status == 200){

                     obj.innerHTML = xmlHttp.responseText;
                }
                }
              xmlHttp.send (null);
           }
</script>
```

اکنون برنامه شما اطلاعات درخواستی را به سرور ارسال می کند و قسمت مدیریت، اطلاعات دریافتی را به شما نشان می دهد. این اولین برنامه واقعی شما تحت برنامه Ajax بود. برنامه در پشت صحنه اطلاعاتی را از سرور دریافت و به شما نمایش می دهد، بدون اینکه وب شما همه فعالیت های خود را تا رسیدن اطلاعات متوقف

کرده و برای نمایش داده ها صفحه را refresh کند. کدهای کامل این مثال در CD کتاب در مسیر example\xmlhttp\1\ با نام firstAjax.html موجود می باشد.

متد GET در XMLHTTP

حال که با چگونگی کار XMLHttp آشنا شدید، اکنون با دو مثال طریقه استفاده از متد های GET و POST را خواهیم دید.در این مثال یک کاربر شماره مشتری را وارد می کند و اطلاعات مربوط به کاربر را دریافت می کند.

در این مثال از PHP که یک زبان متن باز سمت سرور بسیار خوب است و MySql به عنوان بانک اطلاعاتی استفاده شده است. برای شروع کار ابتدا باید جدول مربوط به کاربر را در mySql ایجاد کنید.می توانید این جدول را به صورت زیر ایجاد کنید:

```
CREATE TABLE `user` (
     `ID` INT NOT NULL AUTO_INCREMENT ,
     `username` VARCHAR( 50 ) NOT NULL ,
     `firstname` VARCHAR( 50 ) NOT NULL ,
     `lastname` VARCHAR( 50 ) NOT NULL ,
     `password` VARCHAR( 50 ) NOT NULL ,
     `age` INT NOT NULL ,
     `gender` CHAR( 1 ) NOT NULL ,
     `status` TINYINT NOT NULL ,
     PRIMARY KEY ( `ID` )
);
```

مهمترین فیلد در این جدول CustomerId می باشد که برای گرفتن اطلاعات کاربر از آن استفاده می کنید. ابتدا فایل PHP را ایجاد می کنیم. اولین کار، تعریف همه داده های است که با آن ها سروکار دارید. در این مثال، این داده ها عبارتند از کد مشتری برای جستجو، متغیر$sInfo که اطلاعات را برمی گرداند و اطلاعات مربوط به اتصال به بانک اطلاعاتی. (سرور بانک اطلاعاتی، نام بانک اطلاعاتی، نام کاربر، پسورد و یک رشته دستوری Sql)

```
<?php
     $sID = $_GET["id"];
     $sInfo = "";
     $sDBServer = "your.databaser.server";
     $sDBName = "your_db_name";
     $sDBUsername = "your_db_username";
     $sDBPassword = "your_db_password";
     $sQuery = "Select * from user where ID=".$sID;
     ..
     ..
     ..
?>
```

این کد با گرفتن id شروع می شود. PHP برای سادگی اطلاعات ارسالی از سمت کاربر را در یک آرایه $_GET قرار می دهد. این id در $sID ذخیره می شود، که در ایجاد Query، Sql ذخیره شده در $sQuery مورد استفاده قرار می گیرد. متغیر $sInfo نیز اینجا ایجاد می شود و با یک رشته خالی مقدار دهی می شود.کدهای دیگر مربوط به تنظیمات بانک اطلاعاتی می باشد که شما بایستی با اطلاعات صحیح آن را تنظیم کنید.کار بعدی اتصال به بانک اطلاعاتی و اجرای Query و برگرداندن نتیجه می باشد. اگر کاربری با ID داده شده موجود باشد، $sInfo با کد Html که حاوی اطلاعات مشتری است پر می شود، اگر ID نامعتبر باشد، $sInfo با یک پیغام خطا برای نشان دادن در صفحه وب مقدار دهی می شود.

```
<?php
     $sID = $_GET["id"];
     $sInfo = "";

     $sDBServer = "your.databaser.server";
     $sDBName = "your_db_name";
     $sDBUsername = "your_db_username";
     $sDBPassword = "your_db_password";
     $sQuery = "Select * from user where ID=".$sID;

     $oLink =
     mysql_connect($sDBServer,$sDBUsername,$sDBPassword);
     @mysql_select_db($sDBName) or $sInfo=" Unable to open
     database";

          if($oResult = mysql_query($sQuery) and
            mysql_num_rows($oResult) > 0)
          {
           $aValues = mysql_fetch_array($oResult,MYSQL_ASSOC);
           $sInfo = $aValues['firstname']."<br
                />".$aValues['lastname'].
                "<br />".$aValues['age']."<br
                />".$aValues['gender']."<br />";
          }
          else {
            $sInfo = "Customer with ID $sID doesn't exist.";
          }
     mysql_close($oLink);
?>
```

دو خط اول در کدهای پررنگ شده، شامل اتصال به بانک اطلاعاتی از طریق دستورات PHP می باشد. تابع mysql_query() برای اجرای Query مورد نظر فراخوانی می شود. اگر تابع مقداری را برگرداند که تنها شامل یک سطر بود آنگاه برنامه برای گرفتن اطلاعات وذخیره آن ها در $sInfo ادامه پیدا می کند، در غیر اینصورت $sInfo با یک پیغام خطا مقدار دهی می شود. کد صفحه آخر برای پاک کردن ارتباط بانک اطلاعاتی به کار می رود.

حال بایستی اطلاعات را برای استفاده در XMLHttp به آن پاس دهیم.

```
<?php
     header("Content-Type: text/plain");

     $sID = $_GET["id"];
     $sInfo = "";

     $sDBServer = "your.databaser.server";
     $sDBName = "your_db_name";
     $sDBUsername = "your_db_username";
     $sDBPassword = "your_db_password";
     $sQuery = "Select * from user where CustomerId=".$sID;

     $oLink =
     mysql_connect($sDBServer,$sDBUsername,$sDBPassword);
     @mysql_select_db($sDBName) or $sInfo=" Unable to open
     database";

     if($oResult = mysql_query($sQuery) and
        mysql_num_rows($oResult) > 0)
     {
       $aValues = mysql_fetch_array($oResult,MYSQL_ASSOC);
       $sInfo = $aValues['firstname']."<br
          />".$aValues['lastname']."
          <br />".$aValues['age']."<br />".$aValues['gender'].
          "<br />";
     } else {
       $sInfo = "Customer with ID $sID doesn't exist.";
     }
     mysql_close($oLink);


     echo $sInfo;
?>
```

دو تغییر را می توانید ببینید، اولی در ابتدای فایل انجام شده است، جایی که تابع header() برای تنظیم نوع محتوای صفحه مورد استفاده قرار می گیرد. با اینکه اغلب صفحه یک تکه کد Html بر می گرداند، اما بهتر است که با text/plain تنظیم شود. بخاطر اینکه یک صفحه کامل HTML نیست. شما بایستی Header همه صفحات غیر html ای که به مرورگر می فرستید را تنظیم کنید. تغییر دوم در زیر برنامه قرار دارد، جای که $sInfo به عنوان یک استریم با دستور echo به خروجی داده می شود. حال به سمت کلاینت می رویم و به تنظیم برنامه وب می پردازیم. ابتدا صفحه وبی را به صورت زیر ایجاد می کنیم:

```
<p>Enter customer ID number to retrieve information:</p>
<p>Customer ID: <input type="text" id="txtCustomerId" value=""
/></p>
<p><input type="button" value="Get Customer Info"
     onclick="requestCustomerInfo()" /></p>
<div id="divCustomerInfo"></div>
```

تابع requestCustomerInfo() با استفاده از XMLHttp اطلاعات ارسالی را از فایل PHP گرفته و آن ها را نمایش می دهد.

```
function requestCustomerInfo() {
   var sId = document.getElementById("txtCustomerId").value;
   var oXmlHttp = createXMLHttp();
   oXmlHttp.open("get", "secondAjax.php?id=" + sId, true);
   oXmlHttp.onreadystatechange = function () {
     if (oXmlHttp.readyState == 4) {
     if (oXmlHttp.status == 200) {
       displayCustomerInfo(oXmlHttp.responseText);
     } else {
       displayCustomerInfo("An error occurred: "+
          oXmlHttp.statusText);
     }
     }
   };
   oXmlHttp.send(null);
}
```

تابع requestCustomerInfo مقدار موجود در txtCustomerId را به عنوان ID مشتری دریافت و آن را در متغیر sId ذخیره می کند سپس با استفاده از تابع createXMLHttp یک شی XMLHttp ایجاد می کند. متد open() برای مشخص کردن یک درخواست GET غیرهمزمان، برای فراخوانی صفحه secondAjax.php می باشد. (که ID گفته شده به عنوان اضافه شده است.) سپس مقدار readyState را بررسی می کند که اگر برابر با 4 بود بعد وضعیت status را بررسی می کند اگر status خوب بود(برابر با 200) آنگاه تابع displayCustomerInfo() برای نمایش اطلاعات کاربر فراخوانی می شود. اگر عملیات با خطا روبرو شود اطلاعات خطا به این تابع ارسال می شود.

```
function displayCustomerInfo(sText) {
     var divCustomerInfo =
          document.getElementById("divCustomerInfo");
     divCustomerInfo.innerHTML = sText;
}
```

در این تابع ابتدا، خط اول یک ارجاع به المنت Div برای نمایش اطلاعات صورت می گیرد. در خط دوم، اطلاعات کاربر(sText)، خاصیت innerHTML المنت Div را مقدار دهی می کند. استفاده از innerHTML این امکان را می دهد که در Div بتوانیم مقادیر html قرار دهید. کدهای کامل این مثال در CD کتاب در مسیر 'example\ xmlhttp\2\'موجود می باشد

متد POST در XMLHTTP

اکنون که دیدید XMLHttp چگونه کار با GET را آسان می کند، زمان آن است که به درخواست POST نیز نگاهی بیاندازیم.

ابتدا با فایل SaveCustomer.php که مدیریت درخواستهای POST را به عهده دارد شروع می کنیم. کد PHP، وظیفه استفاده از اطلاعات موجود در درخواست و ذخیره آن ها را بر عهده دارد. با توجه به اینکه این یک درخواست POST است، آرایه $_POST شامل تمام تمامی اطلاعاتی است که ارسال می شود:

```
<?php
     $sName = $_POST["txtName"];
     $sAddress = $_POST["txtAddress"];
     $sCity = $_POST["txtCity"];
     $sState = $_POST["txtState"];
     $sZipCode = $_POST["txtZipCode"];
     $sPhone = $_POST["txtPhone"];
     $sEmail = $_POST["txtEmail"];

     $sStatus = "";

     $sDBServer = "your.database.server";
     $sDBName = "your_db_name";
     $sDBUsername = "your_db_username";
     $sDBPassword = "your_db_password";

     $sSQL = "Insert into
          Customers(Name,Address,City,State,Zip,Phone,'E-
          mail')"."values('$sName','$sAddress','$sCity',
          '$sState','$sZipCode'". ", '$sPhone', '$sEmail')";

     //more here
?>
```

این تکه کد همه اطلاعات را از طریق POST دریافت می کند، علاوه بر این، یک پیغام وضعیت ($sStatus)، و اطلاعات مربوط به بانک را ایجاد می کند، عبارت SQL در این مثال INSERT است، که همه اطلاعات بازیابی شده را ذخیره می کند.

```
<?php
     $sName = $_POST["txtName"];
     $sAddress = $_POST["txtAddress"];
     $sCity = $_POST["txtCity"];
     $sState = $_POST["txtState"];
     $sZipCode = $_POST["txtZipCode"];
     $sPhone = $_POST["txtPhone"];
     $sEmail = $_POST["txtEmail"];

     $sStatus = "";

     $sDBServer = "your.database.server";
     $sDBName = "your_db_name";
     $sDBUsername = "your_db_username";
     $sDBPassword = "your_db_password";

     $sSQL = "Insert into Customers(Name,Address,City,State,
          Zip,Phone,'E-mail') ". " values
```

```
        ('$sName','$sAddress','$sCity','$sState',
         '$sZipCode'". ", '$sPhone', '$sEmail')";
    $oLink =
    mysql_connect($sDBServer,$sDBUsername,$sDBPassword);
    @mysql_select_db($sDBName) or $sStatus = "Unable to open
     database";

    if($oResult = mysql_query($sSQL)) {
        $sStatus = "Added customer; customer ID is
                ".mysql_insert_id();
    } else {
        $sStatus = "An error occurred while inserting;
        customer not saved.";
    }

    mysql_close($oLink);

?>
```

اینجا، نتیجه تابع mysql_query() یک شاخص برای تائید درستی عملیات است. در صورت درستی، متغیر $sStatus با پیغامی که موفقیت عملیات ذخیره وID مشتری، را نشان می دهد، مقداردهی می شود. تابع mysql_insert_id() همیشه جدیدترین مقدار auto-incremented را بر می گرداند. در صورتیکه، عملیات درج به درستی انجام نشده باشد $sStatus با یک پیغام خطا مقدار دهی می شود.

حال اطلاعات نوع محتوا و عملیات ارسال متن را اضافه می کنیم.

```
<?php
    header("Content-Type: text/plain");

    $sName = $_POST["txtName"];
    $sAddress = $_POST["txtAddress"];
    $sCity = $_POST["txtCity"];
    $sState = $_POST["txtState"];
    $sZipCode = $_POST["txtZipCode"];
    $sPhone = $_POST["txtPhone"];
    $sEmail = $_POST["txtEmail"];

    $sStatus = "";

    $sDBServer = "your.database.server";
    $sDBName = "your_db_name";
    $sDBUsername = "your_db_username";
    $sDBPassword = "your_db_password";

    $sSQL = "Insert into Customers(Name,Address,City,
        State,Zip,Phone,'E-mail') ". " values
        ('$sName','$sAddress','$sCity','$sState',
        '$sZipCode'". ", '$sPhone', '$sEmail')";
```

```
     $oLink =
     mysql_connect($sDBServer,$sDBUsername,$sDBPassword);
     @mysql_select_db($sDBName) or $sStatus = "Unable to open
     database";

     if($oResult = mysql_query($sSQL)) {
          $sStatus = "Added customer; customer ID is
            ".mysql_insert_id();
     } else {
           $sStatus = "An error occurred while inserting;
                customer not saved.";
     }

     mysql_close($oLink);
     echo $sStatus;
?>
```

تابع header برای تنظیم نوع محتوا و تابع echo برای ارسال $sStatus استفاده می شود.

در صفحه اصلی یک تگ form برای دریافت اطلاعات کاربر ایجاد می کنیم:

```
<form method="post" action="SaveCustomer.php"
     onsubmit="sendRequest(); return false">

     <p>Enter customer information to be saved:</p>
     <p>Customer Name: <input type="text" name="txtName"
     value="" /><br />
     Address: <input type="text" name="txtAddress" value=""
     /><br />
     City: <input type="text" name="txtCity" value="" /><br />
     State: <input type="text" name="txtState" value="" />
     <br />
     Zip Code: <input type=" text" name="txtZipCode" value=""
     /><br />
     Phone: <input type="text" name="txtPhone" value="" />
     <br />
     E-mail: <input type="text" name="txtEmail" value=""/></p>
     <p><input type="submit" value="Save Customer Info" /></p>
</form>
<div id="divStatus"></div>
```

توجه کنید که رخدادonsubmit نوع فراخوانی تابع sendRequest() را تغییر داده است. (همچنین مقدار false را برمی گرداند تا از هرگونه submit واقعی صفحه جلوگیری کند.) این متد ابتدا داده ها را برای درخواست POST آماده می کند سپس شی XMLHttp را برای ارسال آن آماده می سازد. اطلاعات بایستی با فرمت رشته ای زیر ارسال شوند:

```
name1=value1&name2=value2&name3=value3
```

name و value ها برای هر پارامتر بایستی به صورت URL رمزگذاری شوند تا در مدت انتقال از، از بین رفتن داده ها ممانعت شود. جاوااسکرپت یک تابع داخلی با نام encodeURIComponent() برای این منظور

تدارک دیده است. برای ایجاد این رشته بایستی استخراج و رمزگذاری name و value را روی فیلدهای فرم تکرار کنید. تابع getRequestBody() این کار را انجام می دهد:

```
function getRequestBody(oForm) {
   var aParams = new Array();
   for (var i=0 ; i < oForm.elements.length; i++) {
      var sParam = encodeURIComponent(oForm.elements[i].name);
      sParam += "=";
      sParam += encodeURIComponent(oForm.elements[i].value);
      aParams.push(sParam);
}
return aParams.join("&");
}
```

این تابع فرض را بر این می گذارد که شما یک ارجاع به فرم از طریق یک آرگومان خواهید داشت(oForm). یک آرایه (aParams)برای ذخیره جفت مقدار name-value ایجاد می شود. سپس المنتهای Form در یک حلقه قرار گرفته، یک رشته ایجاد و درsParam ذخیره می شوند، که سپس در یک آرایه قرار می گیرند. انجام این عملیات، در برخی مرورگرها ممکن است کند باشد. آخرین مرحله فراخوانی join() در آرایه است برای برگرداندن داده ها است. این دستور بین جفت مقادیرname-value یک امپرسند "&" قرار داده و یک رشته با فرمت صحیح ایجاد می کند.

تابع sendRequest() تابع getRequestBody() را فراخوانی می کند و درخواست را تنظیم می کند:

```
function sendRequest() {
     var oForm = document.forms[0];
     var sBody = getRequestBody(oForm);

     var oXmlHttp = createXMLHttp();
     oXmlHttp.open("post", oForm.action, true);
     oXmlHttp.setRequestHeader("Content-Type", "application/x-
          www-form-urlencoded");

     oXmlHttp.onreadystatechange = function () {
          if (oXmlHttp.readyState == 4) {
               if (oXmlHttp.status == 200) {
                    saveResult(oXmlHttp.responseText);
               } else {
                    saveResult("An error occurred: "+
                    oXmlHttp.statusText);
               }
          }
     };
     oXmlHttp.send(sBody);
}
```

همانند مثال قبلی، مرحله اول دراین تابع ایجاد یک فرم و ذخیره آن در متغیر (oForm) است که پس از ایجاد بدنه درخواست، در sBody ذخیره می شود. مرحله بعد ایجاد یک شی XMLHTTP است. توجه کنید که در متد open آرگومان اولی به جای GET، POST می باشد، هچنین آرگومان دوم باoForm.action مقداردهی می شود. هچنین Header درخواست نیز تنظیم شده است. هنگامی که یک فرم از مرورگر به سرور ارسال می شود، نوع محتوای درخواست آن با application/x-www-form-urlencoded مقداردهی می شود. اکثر زبانهای تحت سرور در درخواست های POST به این کد توجه می کند، بنابراین این دستور از اهمیت خاصی برخوردار است.رخداد onreadystatechange تابع saveResult() را فراخوانی می کند:

```
function saveResult(sMessage) {
     var divStatus = document.getElementById("divStatus");
     divStatus.innerHTML = "Request completed: "+ sMessage;
}
```

این تابع وظیفه نمایش اطلاعات مشتری و یا پیغام خطا را دارد خط آخر بسیار مهم است، sBody به تابع send پاس داده می شود که بعد جزئی از درخواست می شود.

فصل چهارم: کار با Ajax در Asp.Net

ASP.Net فریم ورک تقریباً جدیدی از شرکت مایکروسافت می باشد که با معرفی آن برنامه نویسی Web پا در عرصه جدیدی گذاشت. با اینکه آن را نمودی از JSP جاوا می دانند، اما نمی توان تفاوت بین این دو، روش آسانتر Asp.Net نسبت به روش جاوا و البته کمی سخت نسبت به PHP را کتمان کرد. این فصل به هیچ عنوان قصد مقایسه روشهای ایجاد وب و یا آموزش آنها را ندارد؛ و تنها به ایجاد برنامه های Ajaxی بر روی ASP.Net خواهد پرداخت.

کنترل ها

در ASP.Net علاوه بر عناصر وب، شاهد وجود عناصر ASP.Net ی با نام کنترل هستیم که برنامه نویسی را برای توسعه دهندگان به مراتب آسان تر از برنامه نویسی در پلتفرم های دیگر کرده است. این کنترل ها به صورت عمومی در قالب <asp:controlName> تعریف می شوند و عملیات پویا را در واسط کاربری ایجاد می کنند . با ایجاد رویدادی بر روی کنترل ها (فشردن دکمه و...)، کنترل ها اطلاعاتی را به سمت سرور ارسال می کنند ومنتظر اتمام کار سرور و پاسخ از سمت سرور می شوند ؛ با دریافت اطلاعات از سمت سرور صفحه وب مجددا بارگذاری شده و احتمال از دست دادن اطلاعات قبلی وجود دارد.

اما عناصر معمولی html این خاصیت را ندارند وحالت پیش فرض آن ها اجرای رویداد در سمت کلاینت می باشد. به علت سر بار زیادی که کنترل های ASP.NET دارند در اینجا از عناصر ساده html استفاده می کنیم.

Ajax در ASP.Net

دراین بخش قصد داریم چگونگی کار با Ajax را با یک مثال توضیح دهیم: قرار است برنامه ای ایجاد کنید که کاربر با انتخاب یک کلید اطلاعات یک فایل XML را در صفحه وب مشاهده کند. این کار را از طریق ajax انجام خواهید داد.

ابتدا یک پروژه وب در Microsoft Visual Studio با نام AspAjaxSample ایجاد کنید. یک پوشه با نام javaScript ایجاد کرده و با استفاده از گزینه Add New Item یک فایل جاوا اسکریپتی با نام xmlHttp.js در آن ایجاد می کنید. باید توجه داشت که این پوشه تنها برای دسترسی آسان به داده است و لزومی برای وجود آن نیست. حال دو متد که برای ایجاد ajax ساخته بودیم را در این فایل قرار می دهیم.

حال یک web Form با نام SimpleResponseText.aspx را به پروژه اضافه می کنیم.

تابع getData() (که در فصل سوم با آن آشنا شدید) را به صورت زیر اصلاح کنید. این تابع را به همراه تابع createXMLHttp() به فایل xmlHttp.js اضافه نماید:

```
function getData(dataSource, divID)
{
    xmlHttp = createXMLHttp();
```

```
    if(xmlHttp) {
        var obj = document.getElementById(divID);
        xmlHttp.open("GET", dataSource);
        xmlHttp.onreadystatechange = function()
        {
            if (xmlHttp.readyState == 4 && xmlHttp.status ==
               200) {
                obj.childNodes[0].nodeValue =
                        xmlHttp.responseText;
            }
        }
        xmlHttp.send(null);
    }
}
```

صفحه وب (SimpleResponseText.aspx) بایستی متضمن فایل xmlHttp.js شود تا بتواند یک درخواست Ajaxی داشته باشد. برای اینکار مانند زیر باید عمل کنید:

```
<head runat="server">
  <title> Ajaxian Simple ResponseText</title>
  <script type="text/javascript"
     src="javaScript/xmlHttp.js"></script>
</head>
```

حال کد اصلی را به صورت زیر تغییر می دهیم تا اطلاعات XML را نمایش دهد.

```
<body>

    <form id="form1" runat="server">

        <input type="button" onclick="getData('http://' +
               location.host +
               '/AspAjaxSample/resource/DataFile.xml','divResu
          lts');" value="Test XMLHTTP Call"/>
        <br />
        <br />

        <div id="divResults">{no results}</div>

    </form>

</body>
```

حال اصلی ترین عنصر این مجموعه یعنی فایل DataFile.xml که یک فایل xml است را در پوشه resource ایجاد می کنیم. محتویات فایل DataFile.xml به صورت زیر می باشد.

```
<?xml version="1.0" encoding="utf-8" ?>
<Customers>
     <Customer>
          <Firstname>Joe</Firstname>
          <Lastname>Bloggs</Lastname>
          <email>joe@bloggs.com</email>
```

```
    </Customer>

    <Customer>
        <Firstname>Alan</Firstname>
        <Lastname>Anonymous</Lastname>
        <email>anon@ymous.com</email>
    </Customer>

    <Customer>
        <Firstname>Marvin</Firstname>
        <Lastname>Martian</Lastname>
        <email>marvin@mars.com</email>
    </Customer>

</Customers>
```

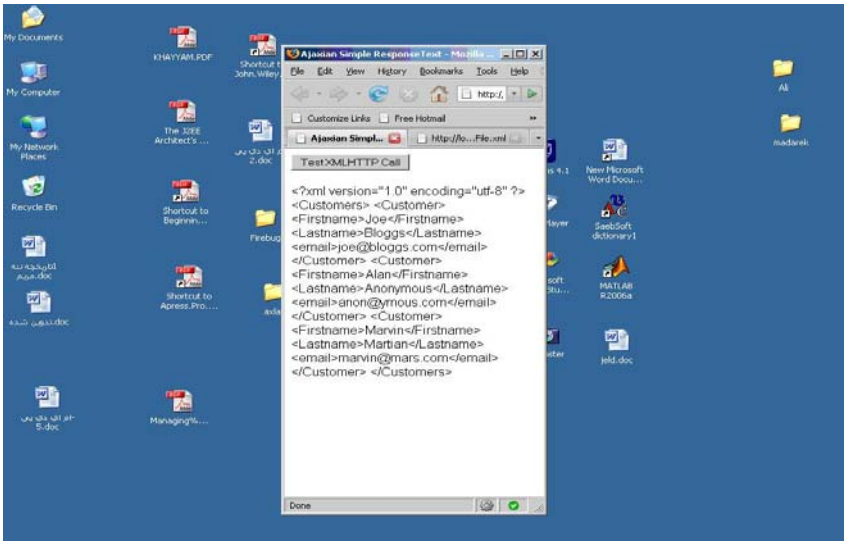
زمانی که کلید را انتخاب کنید، نتیجه به صورت شکل ۱-۵ نمایش داده می شود.

همانطور که می بینید، اطلاعات فایل XML به همان صورتی که در آن ذخیره شده اند نمایش داده می شوند.

استفاده از خاصیت responseXML

در سناریوی اصلی شما می خواهید که چندین آیتم نمایش دهید، که می تواند به صورت انتخاب نام افراد از لیست باشد. همانطور که در فصل سوم نیز اشاره شد برای خواندن فرمت XML ، شی XMLHTTP یک خاصیت با نام responseXML را فراهم کرده است. این خاصیت یک شی سندی XML استاندارد می باشد که می تواند به بررسی و تجزیه اسناد XML یا استفاده متدها و خاصیت های گره های DOM بپردازد.

ابتدا یک Web Form با نام SimpleResponseXML.aspx ایجاد کنید. و فایل xmlHttp.js را به آن الحاق کنید.



شکل ۱-۵ نمایش فایل DataFile.xml

قبل از شروع به نوشتن برنامه جدید، اصلاحی در فایل جاوا اسکریپتی انجام دهید. برای اینکه تابع gatData را یک متد قابل انعطاف تری بسازید آن را به گونه ای تغییر بدهید که بتوانید تقریبا در هر برنامه ای آن را به کار

ببرید. توجه دارید که برای هر برنامه بایستی عملیاتی که رویداد onreadystatechange انجام می دهد را تغییر دهید، تا از برنامه جواب بگیرید. جاوااسکریپت یک ویژگی ساده و جالبی دارد که شما را قادر می سازد کد یک تابع را به تابع دیگری به عنوان پارامتر پاس دهید. از این خاصیت استفاده کرده و تابع جدیدی به صورت زیر ایجاد کنید:

```
function getDataCallBack(callBack,dataSource)
{
    xmlHttp = createXMLHttp();
    if(xmlHttp) {

        xmlHttp.open("GET", dataSource);
        xmlHttp.onreadystatechange = function()
        {
            if (xmlHttp.readyState == 4 && xmlHttp.status ==
               200) {
                callBack(xmlHttp.responseText,xmlHttp.
                    responseXML);
            }
        }
        xmlHttp.send(null);
    }
}
```

callBack همان تابعی است که به صورت پارامتر به تابع جدید getDataCallBack پاس داده شده است. در رویداد onreadystatechange نیز پس از عملیات اعتبار سنجی به جای انجام مستقیم عملیات، یک تابع پاس داده شده با دو پارامتر فراخوانی می شود. با این کار کد خواناتری خواهید داشت.

ابتدا یک تابع جدیدی به فرم اضافه کنید، این تابع به صورت زیر خواهد بود:

```
<head runat="server">
    <script type="text/javascript"
          src="javaScript/xmlHttp.js"></script>
    <script type="text/javascript" >
        function getDataCallBack(responseText,responseXML){
           var obj = document.getElementById("divResults");
           var node = responseXML.selectSingleNode(
               "//Customers/Customer/Firstname/text()");
           obj.childNodes[0].nodeValue = node.nodeValue;
        }
    </script>
</head>
```

به جای اینکه داده خروجی را به صورت مجموعه کامل نمایش دهید، اولین نمونه <Firstname>، را نمایش می دهید. شما این کار را با استفاده متد selectSingleNode انجام می دهید. این متد در صورت پیدا کردن جواب یک گره XMLی را برمی گرداند در غیر اینصورت مقدار خروجی آن null خواهد بود. مقدار شی گره در nodeValueی اولین عنصر بدست آمده توسط خاصیت childNodes از عنصر

divResults قرار داده می شود. اگر از Firefox استفاده می کنید باید توجه داشته باشید که Firefox از متدهای selectSingleNode() و selectNodes() حمایت نمی کند.

اکنون بدنه ی مثال جدید را همانند مثال قبلی نوشته و تنها برای استفاده از متد جدید در آن تغییر کوچکی بدهید:

```
<form id="form1" runat="server">
     <input type="button"
      onclick="getDataCallBack(parsingData,'http://' +
     location.host + '/AspAjaxSample/resource/DataFile.xml');"
      value="Test XMLHTTP Call"/>
     <br />
     <br />
     <div id="divResults">{no results}</div>
</form>
```

این مثال فعلا یک مثال ساده می باشد، حالا با ایجاد یک صفحه وب ساده می خواهیم که به شما اجازه انتخاب یک فرد از لیست افراد و نمایش نام کامل واطلاعات mail با انجام یک جستجوی سمت سرور از فایل XML به صورت غیر همزمان داده شود.

با استفاده از یک عنصر drop-down، لیستی از نام افراد موجود در فایل XML را ایجاد می کنیم. زمانی که کاربر نامی را از لیست انتخاب کرد ، یک درخواست سمت سرور برای بازیابی آن داده و بازیابی email اجرا می شود. بدون استفاده از XMLHTTP کاربر بایستی تا زمان ارسال اطلاعات به سرور و پردازش آن و بازیابی اطلاعات منتظر بماند.

بدنه فرم را به صورت زیر تغییر بدهید:

```
<body onload="getDataCallBack(LoadCustomers,'http://' +
     location.host + '/AspAjaxSample/resource/
     DataFile.xml');">
    <form id="form1" runat="server">


 <div>
         <select id="ddlCustomers"
         onchange="getDataCallBack(DisplayCustomerDetails,
         'http://' + location.host +
         '/AspAjaxSample/resource/DataFile.xml');">
               <option value="">- Select a Customer -
              </option>
         </select>

           <hr />

           <div>
               <p>Details:</p>
               <span id="spnDetailDisplay">
```

```
                (You have not made a selection yet.)
              </span>
            </div>

        </div>

    </form>
</body>
```

داخل عنصر فرم یک عنصر <select> قرار دارد که نام کارمندان را نمایش می دهد و یک عنصر که در آن می توانید مشخصات کارمندان را نمایش دهید. توجه کنید که رویداد onload <body> تابع getDataCallBack که متد LoadCustomers() به آن پاس داده شده را فراخوانی می کند که برای مقدار دهی اولیه لیست با نام کارمندان می باشد، همچنین رویداد onchange عنصر <select> ، تابع DisplayCustomerDetails() را فراخوانی می کند که اطلاعات کارمند انتخابی را از انتخاب کرده و نمایش می دهد.

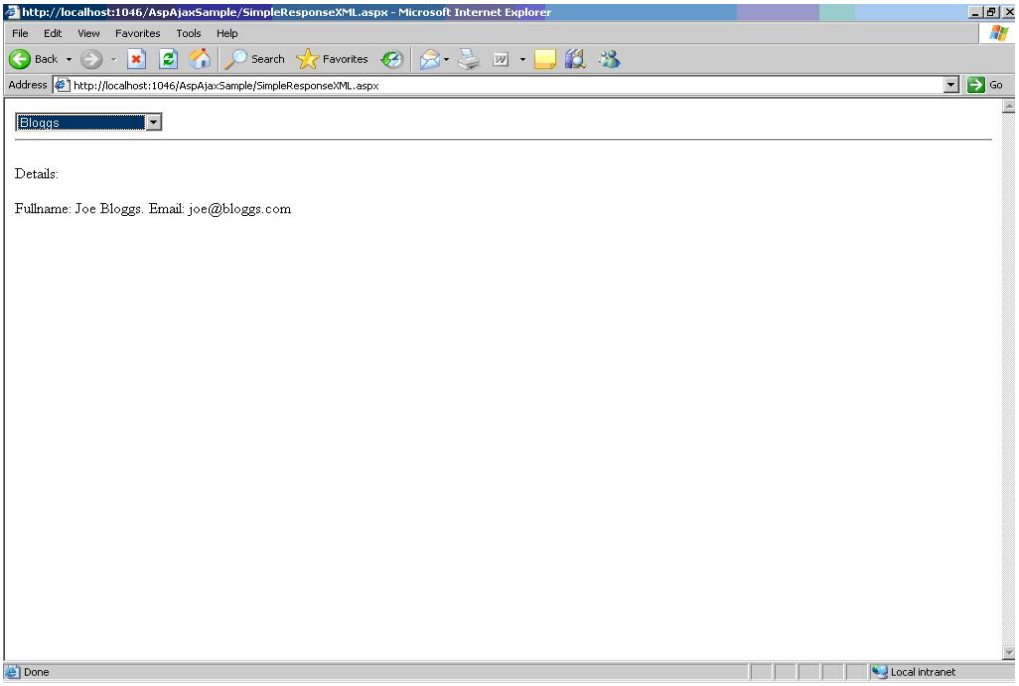
متد LoadCustomers() که نام کارمندان را نمایش می دهد به صورت زیر خواهد بود:

```
function LoadCustomers(responseText,responseXML){

            var xmlDoc = responseXML;
            var nodes = xmlDoc.selectNodes
               ("//Customers/Customer/Lastname/text()");
            var ctrl =
               document.getElementById("ddlCustomers");
            for (var i=0; i < nodes.length; i++)
            {

                var lastName = nodes[i].nodeValue;
                var htmlCode=
                    document.createElement('option');
                ctrl.options.add(htmlCode);
                htmlCode.text = lastName;
                htmlCode.value = lastName;
            }
}
```

حال برای نمایش اطلاعات کارمند باید متد DisplayCustomerDetails() را پیاده سازی کنید:

```
function DisplayCustomerDetails(responseText,responseXML){

            var ctrl =
               document.getElementById("ddlCustomers");
            var doc = responseXML;
            var lastName=ctrl.options[ctrl.selectedIndex]
               .value;

            var node = doc.selectSingleNode
```

```
            ("//Customers/Customer[Lastname='" +
              lastName + "']");
         var details = 'Fullname: ' + node
            .selectSingleNode('Firstname/text()')
            .nodeValue + ' ' +lastName + '. Email: ' +
            node.selectSingleNode('email/text()')
            .nodeValue;


      document.getElementById("spnDetailDisplay")
      .childNodes[0].nodeValue = details;
  }
```



شکل۲-۵ نمایش مشخصات فرد انتخابی از لیست

ارسال پارامتر به سرور

موارد زیادی وجود دارد که شما نیاز به ارسال داده به سمت سرور دارید. روش سنتی برای انجام این کار همانطور که در فصلهای قبل گفته شد ارسال از طریق رشته آرگومانهای به عنوان قسمتی از درخواست XMLHTTP می باشد. یک مثال می تواند به صورت زیر باشد:

```
xmlHttp.open("GET","http://" + location.host +
     "/XmlHttpExample1/WebForm1.aspx?arg=123", true);
```

صفحه ای که این درخواست را گرفته و پردازش می کند (در این مثال WebForm1.aspx) می تواند رشته پارامترها موجود در URL که از آن ها query string نام برده می شود را برداشته و از آن ها استفاده کند. نمونه از به گرفتن رشته پارامترها در زیر آمده است:

```
private void Page_Load(object sender, System.EventArgs e)
{
```

```
    if (Request.QueryString.Count > 0)
    {
        string queryArg = Request.QueryString["arg"];
        switch (queryArg)
        {
        case "123":
            Server.Transfer("DataFile1.xml");
            break;
        case "456":
            Server.Transfer("DataFile2.xml");
            break;
        default:
            Server.Transfer("DataFile1.xml");
            break;
        }
    }
}
```

استفاده از روش سنتی گزینه مناسبی برای استفاده از آرگومانها در سمت سرور نمی باشد،

HTTP Handeler ها

یک روش برای مدیریت این نوع درخواست ها استفاده یکHTTP handeler می باشد. یک handeler توانائی پاسخ دادن به درخواست های با کنترل مستقیم بر روی داده ی پاسخ را دارد. HTTP handeler می تواند مستقیما با درخواست ها سروکار داشته باشد بدون اینکه نگران داده ی HTML تولید شده توسط صفحه، باشد.

برای نشان دادن استفاده از HTTP handeler در دریافت آرگومانهای ارسالی از شی XMLHTTP ، بایستی ابتدا واسط کاربری صفحه وب را طراحی کنید، که در کد زیر نشان داده شده است. این مثال یک لیست drop-down که با نام سه کارمند مقدار دهی شده است را نشان می دهد . زمانی که یکی از کارمندان انتخاب می شوند، ID کارمند به عنوان قسمتی از درخواست Ajax بمنظور دریافت مجموعه خاصی از داده ها که به مشخصات کارمند انتخابی مربوط می شوند استفاده می شود.

```
<form id="form1" runat="server">
        <div>
            <select id="ddlCustomers"
               onchange="LoadCustomers();">
                <option value="">- Select a Customer -
                </option>
                <option value="1">Customer 1</option>
                <option value="2">Customer 2</option>
                <option value="3">Customer 3</option>
            </select>
            <hr />
            <div>
                <p>Details:</p>
                <span id="spnDetailDisplay">
               (You have not made a selection yet)
```

```
            </span>
          </div>
      </div>
  </form>
```

در این مثال onchange عنصر <select> متد LoadCustomer را فراخوانی می کند. دراین متد یک درخواست Ajax داریم که متد ShowCustomer را به تابع getDataCallBack پاس می دهد، همچنین از یک handeler برای ایجاد درخواست asynchronous استفاده می کند و arg را با گزینه انتخابی در عنصر <select> مقدار دهی می کند.

متد LoadCustomer به صورت زیر پیاده سازی می شود:

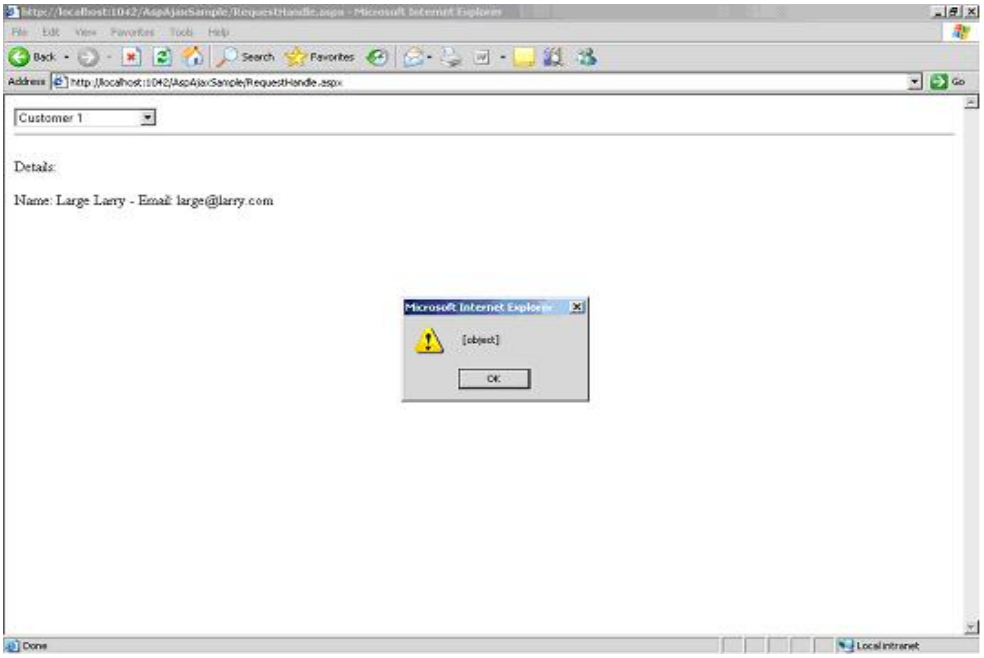
```
<head runat="server">
    <script type="text/javascript"
          src="javaScript/xmlHttp.js"></script>
    <script type="text/javascript" >

        function LoadCustomers(){

            var ddlCtrl=document.
               getElementById("ddlCustomers");

            var custNumber = ddlCtrl.value;
            getDataCallBack(ShowCustomer,'http://' +
               location.host + '/AspAjaxSample/
               AsyncRequestHandler.ashx?arg='+custNumber);
        }

    </script>
</head>
```

تابع ShowCustomer نیز به صورت زیر پیاده سازی می شود:

```
function ShowCustomer(responseText,responseXML){

            var disp = document.
               getElementById("spnDetailDisplay");
            var xmlDoc = responseXML;
            var name =xmlDoc.
              selectSingleNode("//root/Customer/name/text()");
            var email =xmlDoc.selectSingleNode
               ("//root/Customer/email/text()");
            alert(name);
            disp.childNodes[0].nodeValue = "Name: " +
               name.nodeValue + " - Email: " +
               email.nodeValue;
        }
```

پیاده سازی یک HTTP handeler را در زیر می بینید. این handeler توسط شی XMLHTTP فراخوانی می شود، و ID کاربر را گرفته و از آن برای برگرداندن یک سند XML، شامل نام و email کارمند، به شی XMLHTTP استفاده می کند. این کد در فایل AsyncRequestHandler.ashx قرار می گیرد.

```
<%@ WebHandler Language="C#" Class="AsyncRequestHandler" %>

using System;
using System.Web;

public class AsyncRequestHandler : IHttpHandler {

    public void ProcessRequest (HttpContext context) {

        string param = context.Request.QueryString["arg"];
        const string xmlData = @"<?xml version=""1.0""
          encoding=""utf-8"" ?><root><Customer><name>{0}
         </name><email>{1}</email></Customer></root>";
        string returnXML = null;
        switch (param)
        {
            case "1":
                returnXML = string.Format(xmlData, "Big Bob",
                     "big@bob.com");
                break;
            case "2":
                returnXML = string.Format(xmlData,
                     "Small Sammy","small@sammy.com");
                break;
            case "3":
                returnXML = string.Format(xmlData,
                     "Large Larry","large@larry.com");
                break;
        }
        context.Response.ContentType = "application/xml";
        context.Response.Write(returnXML);
    }

    public bool IsReusable {
        get {
            return false;
        }
    }

}
```



شکل۳-۵ نمایش مشخصات کارمند با HTTP handeler

Web Service ها چیستند؟

.NET به صورت گسترده از Web Service ها حمایت می کند. Web Service مکانیسم برتر برای ارائه عملیات سمت سرور و یا نقطه ای برای ورود به پردازش های سمت سرور می باشد. دسترسی به این سرویس ها در روش asynchronous یک مرحله ی ساده و روشی مناسب در ایجاد هماهنگی بین برنامه های سمت کلاینت و سرور است که توسط .NET در اختیار توسعه دهندگان قرار گرفته است.

مثال زیر شامل یک تابع Web Service با نام Adder است که وظیفه آن محاسبه حاصل جمع دو آرگومان ارسالی از سمت کلاینت به سرویس است:

```
<form id="form1" runat="server">
        <div>
            <input type="text" id="val1" />
            <input type="text" id="val2" />
            <input type="button" value="Calculate"
               onclick="ExecWebService();" />
            <hr />
            <div>
                <p>Details:</p>
                <span id="spnDetailDisplay">
               (You have not made a selection yet)
              </span>
            </div>
        </div>
    </form>
```

این صفحه ساده HTML از دو فیلد text که مقادیر عددی می گیرند و به عنوان آرگومانهای عملیات جمع استفاده می شود و یک کلید که برای محاسبه حاصل جمع دو آرگومان است تشکیل شده است. در

onclick کلید تابع ()ExecWebService که WebService شما را به صورت ()ExecWebService فراخوانی می کند، نوشته شده است.

حال باید کد WebService، که یک کد ساده می باشد را پیاده سازی کنید. Web Service در فایل AsyncService.asmx ایجاد می شود و شامل تنها یک متد با نام Adder می باشد:

```
[WebService(Namespace = "http://tempuri.org/")]
[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
public class AsyncService : System.Web.Services.WebService {

    public AsyncService () {
    }

    [WebMethod]
    public int AdderAdder(int arg1, int arg2) {
        return arg1 + arg2;
    }

}
```

حال به پیاده سازی متد ()ExecWebService می پردازیم. این متد مقادیر دو فیلد را گرفته و یک درخواست ajax-ی با پاس دادن متد ShowResoult به تابع getDataCallBack، ایجاد می کند:

```
function ExecWebService(){

           var ctlVal1 = document.getElementById("val1");
           var ctlVal2 = document.getElementById("val2");

           getDataCallBack(ShowResoult,'http://' +
            location.host +'/AspAjaxSample/AsyncService.asmx
            /Adder?arg1=' + ctlVal1.value + '&arg2=' +
           ctlVal2.value);
 }
```

متد ShowResoult برای نمایش مقدار برگردانده شده از Web Server به کار می رود:

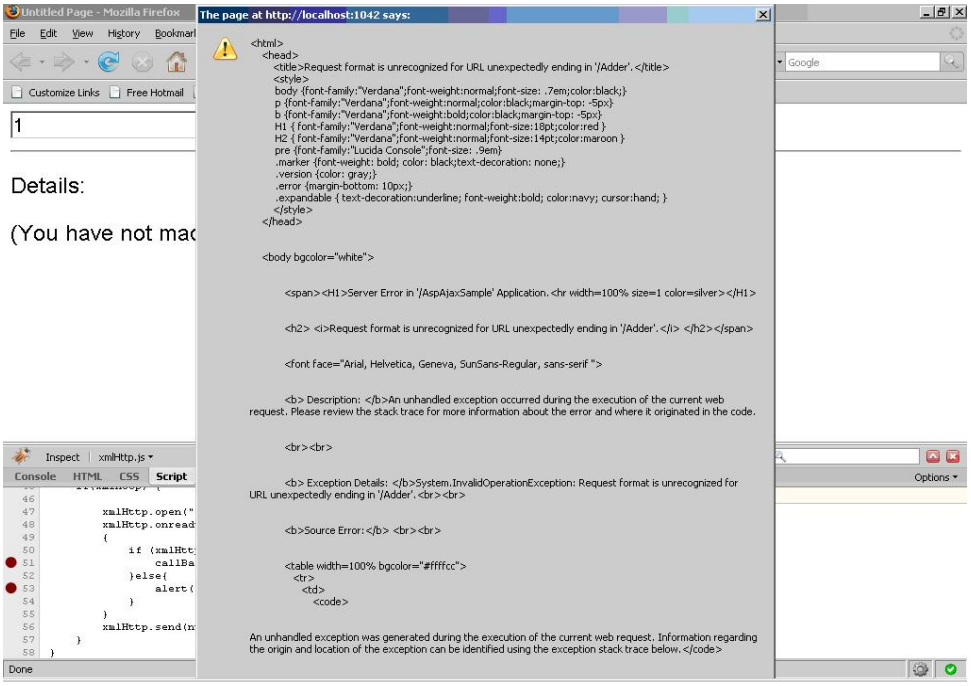
```
function ShowResoult(responseText,responseXML){
            var disp = document
               .getElementById("spnDetailDisplay");
            var result = responseXML.lastChild
               .childNodes[0].nodeValue;
            disp.childNodes[0].nodeValue = "Result: " +
                    result;
      }
```

با اجرای برنامه، هیچ جوابی از سمت سرور دریافت نخواهید کرد. این موضوع نشان از ایجاد یک مشکل در کد شما می باشد. برای پی بردن به مشکل می توانید از نرم افزارهای Debug مانند Firebug که بر روی مرورگر Firefox کار می کنند استفاده کنید. در اینجا ما از تغییر در کد برنامه اشکال موجود را بررسی می کنیم. در کد جاوااسکریپت تابع getDataCallBack را به صورت زیر تغییر دهید:

```
function getDataCallBack(callBack,dataSource)
```

```
{     .
      .
      .

        if (xmlHttp.readyState == 4 && xmlHttp.status == 200) {
            callBack(xmlHttp.responseText,xmlHttp.responseXML);
        }else{
            alert(xmlHttp.responseText);
        }

    }
}
```

با اجرای دوباره برنامه یک پیغام خطا در برنامه شما ظاهر می شود، که مشکل در ارسال درخواست به سمت سرور را بیان می کند.



شکل ۴-۵ وجود خطا در ارسال درخواست به سرور

برای حل مشکل بایستی در web.config برای کار با Web Service ها تغییراتی را اعمال کنید:

```
<configuration>
    <system.web>
        .
        .
        .
        <webServices>
            <protocols>
                <add name="HttpGet"/>
```

```
                </protocols>

            </webServices>
        </system.web>
</configuration>
```

با اضافه کردن کد بالا در web.config، برنامه اجرا اجرا خواهد شد.

برای استفاده از رشته آرگومانها، که در مثال بالا نشان داده شد، می توانید از متد send() شی XMLHTTP برای الحاق داده ها با بدنه درخواست استفاده کنید. در این صورت احتیاجی به اضافه کردن کد در web.config نخواهید داشت.

برای نوشتن این برنامه از پروتکل POST استفاده می کنیم، به همین دلیل بایستی getDataCallBack() را برای استفاده از این پروتکل آماده کنید. روش خوبی که برای این کار می توان پیشنهاد داد استفاده از یک آرگومان برای مشخص کردن نوع پروتکل می باشد که در هر بار فراخوانی بایستی پروتکل را به آن پاس داد. در اینجا یک متد جدید برای استفاده از این پروتکل اضافی پیاده سازی می شود.

```
function getDataCallBackWithPost(callBack,dataSource,sendData)
{
    xmlHttp = createXMLHttp();
    if(xmlHttp) {

        xmlHttp.open("POST", dataSource);
        xmlHttp.onreadystatechange = function()
        {
            if (xmlHttp.readyState == 4 && xmlHttp.status ==
               200) {

               callBack(xmlHttp.responseText,xmlHttp.
                    responseXML);
            }else{
                alert(xmlHttp.responseText);
            }
        }
        xmlHttp.setRequestHeader("Content-
Type","application/x-
          www-form-urlencoded");
        xmlHttp.send(sendData);
    }
}
```

تغییرات تابع getDataCallBackWithPost() نسبت به تابع getDataCallBack() به صورت پر رنگ نشان داده شده است. آرگومان sendData ، رشته آرگومانی می باشد که به بدنه درخواست الحاق شده و به سرور ارسال می شود. حال به بدنه صفحه HTML دکمه ی برای انجام عملیات جمع با متد POST اضافه کنید:

```
<form id="form1" runat="server">
```

```
      .
      .
      .
<input type="button" value="CalculateWithPost"
     onclick="ExecWebServiceWithPost();" />
      .
      .
      .
</form>
```

حال متد را به صورت زیر پیاده سازی کنید:

```
function ExecWebServiceWithPost(){

           var ctlVal1 = document.getElementById("val1");
           var ctlVal2 = document.getElementById("val2");

           var sendData =  'arg1=' + ctlVal1.value + '&arg2='
               + ctlVal2.value ;
           getDataCallBackWithPost(ShowResoult,'http://' +
          location.host +'/AspAjaxSample/AsyncService.
          asmx/Adder', sendData);
 }
```

به کد بالا نگاه بیاندازید با مثالهای قبلی چند تفاوتهای را مشاهد خواهید کرد.

با استفاده از Web Service، URL مورد استفاده کمی متفاوت از درخواست های قبلی می باشد:

```
xmlHttp.open("POST","http://" + location.host + "/AspAjaxSample
     /AsyncService.asmx/Adder");
```

می بینید که URL شامل آدرس Web Server و فایل asmx می باشد، بعلاوه متدی از Web Server که برای انجام عملیات فراخوانی شده است (/Adder) به URL اضافه می شود. قبل از ارسال درخواست به سرور، هدر درخواست با نام "Content-Type" را با "application/x-www-form-urlencoded" مقدار دهی می کنیم. صفت "Content-Type" نوع رمزگشائی فرم را تعیین می کند. مقدار پیش فرض برای "Content-Type"، "application/x-www-form-urlencoded" می باشد. البته این مقدار پیش فرض XMLHTTP نمی باشد و بایستی به صورت دستی نشان داده شود.

برای اجرای متد send() در شی XMLHTTP ، باید لیستی از آرگومان ها را همانند روشی که برای رشته آرگومان های URL استفاده کردید، ایجاد کنید.

```
xmlHttp.send("arg1=" + ctlVal1.value + "&arg2=" +
      ctlVal2.value);
```

توجه کنید که پارامترهای arg1 و arg2 با آمپرسند (&) همانند روشی URL از هم جدا شده اند. نتیجه این رشته به صورت زیر خواهد بود:

```
arg1=1&arg2=2
```

که مقدار arg1 برابر ۱ و arg2 برابر با ۲ می باشد. این رشته به صورت خودکاربه عنوان قسمتی از بدنه درخواست در آمده و با توسط web Service مدیریت خواهد شد. این روش مانند همان روشی است که درخواست GET انجام می شد. درخواست مشابه با استفاده از رشته آرگومانها در URL می تواند شبیه کد زیر باشد:

```
http://www.somesite.com/SomePage.aspx?arg1=1&arg2=2
```

در آخر، حاصل جمع به صورت زیر نمایش داده می شود:

```
var result = responseXML.lastChild.text;
```

این دستور، مقدار رشته ی آخرین گره فرزند را از پاسخ WebServer به دست می آورد. پاسخی که به آرگومانهایی با مقادیر گفته شده به درخواست webServer، داده می شود، به صورت زیر خواهد بود:

```
<?xml version="1.0" encoding="utf-8"?>
<int xmlns="http://tempuri.org/">3</int>
```

در این مثال، امکان روبرو شدن با دونوع خطا وجود داردکه باید به روشهای مختلفی مدیریت شود:

- اولین خطا زمانی اتفاق می افتد که متد تعیین شده در url (در این مثال Adder/) از آن حذف شود و یا آرگومان نامعتبر با درخواست ارسال شود.
- نوع بعدی از خطا زمانی اتفاق می افتد که تعین "Content-Type" برای هدر درخواست فراموش شود.

برای هر دو مورد خطا، شما مقدار ۵۰۰ را برای status خواهید داشت. در نمونه اول، بدنه پاسخ مانند مثال زیر خواهید داشت:

```
System.InvalidOperationException: Request format is invalid.:
at System.Web.Services.Protocols.HttpServerProtocol.ReadParameters()
at
System.Web.Services.Protocols.WebServiceHandler.CoreProcessRequest()
```

در نمونه دوم، پاسخ زیر را خواهید داشت:

```
<?xml version="1.0" encoding="utf-8"?><soap:Envelope
xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"><soap:Body><soap:Fault>
<soap:Code><soap:Value>soap:Receiver</soap:Value></soap:Code>
<soap:Reason><soap:Textxml:lang="en">
System.Web.Services.Protocols.SoapException: Server was unable to
process request. ---&gt; System.Xml.XmlException: Data at the root
level is
invalid. Line 1, position 1.
at System.Xml.XmlTextReaderImpl.Throw(Exception e)
at System.Xml.XmlTextReaderImpl.Throw(String res, String arg)
at System.Xml.XmlTextReaderImpl.ParseRootLevelWhitespace()
at System.Xml.XmlTextReaderImpl.ParseDocumentContent()
at System.Xml.XmlTextReaderImpl.Read()
at System.Xml.XmlTextReader.Read()
```

```
at
System.Web.Services.Protocols.SoapServerProtocol.SoapEnvelopeReader.
Read()
at System.Xml.XmlReader.MoveToContent()
at
System.Web.Services.Protocols.SoapServerProtocol.SoapEnvelopeReader.
MoveToContent()
at
System.Web.Services.Protocols.SoapServerProtocolHelper.GetRequestEle
ment()
at
System.Web.Services.Protocols.Soap12ServerProtocolHelper.RouteReques
t()
at
System.Web.Services.Protocols.SoapServerProtocol.RouteRequest(SoapSe
rverMessage
message)
at System.Web.Services.Protocols.SoapServerProtocol.Initialize()
at   System.Web.Services.Protocols.ServerProtocolFactory.Create(Type
type,
HttpContext  context,  HttpRequest  request,  HttpResponse  response,
Boolean&amp;
abortProcessing)
--- End of inner exception stack trace ---
</soap:Text></soap:Reason><soap:Detail
/></soap:Fault></soap:Body></soap:Envelope>
```

ASP.NET AJAX

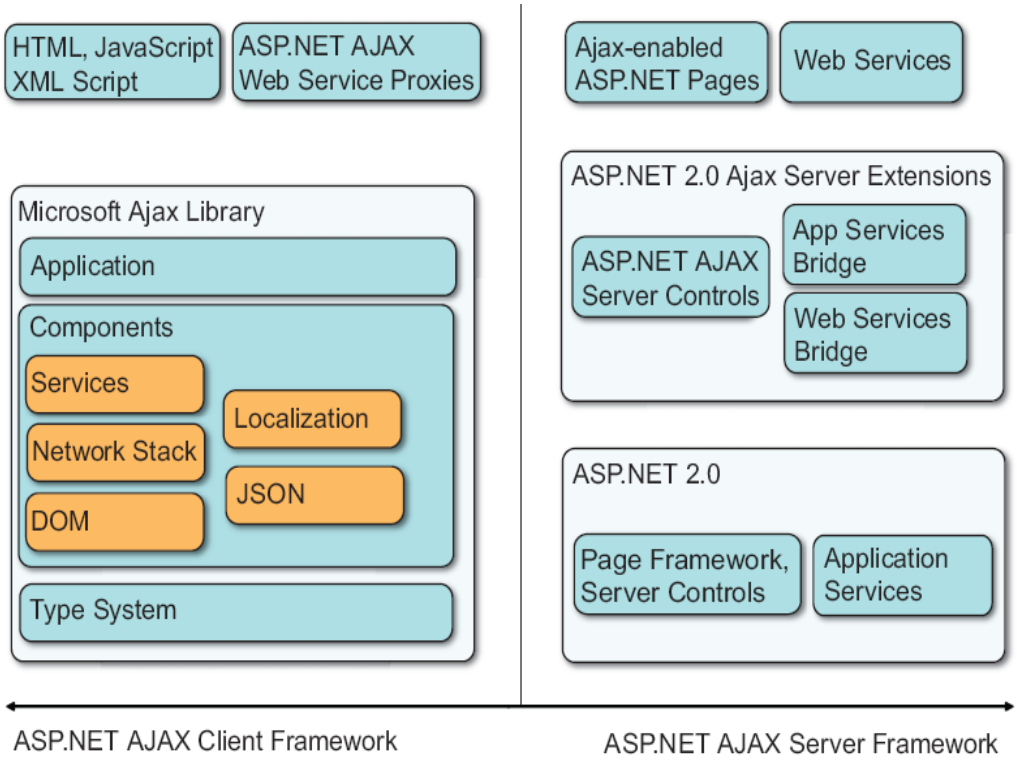
MS Ajaxیا به بیان کامل تر Microsoft Ajax کتابخانه ای از توابع می باشد که توسط شرکت مایکروسافت به منظور بهره گیری از تکنولوژی Ajax در برنامه های تحت وب در VS 2005 ایجاد شده است.

لازم به ذکر است که نام سابق آن Atlas بوده و هم اکنون به صورت منبع باز (Open Source) در دسترس می باشد.

معماری ASP.NET AJAX

فریم ورک ASP.NET AJAX توسعه دهندگان را به ایجاد برنامه های وب قدرتمند، با قابلیت فعل و انفعال، و با قابلیت شخصی سازی بالا قادر می سازد. در نگاه اول، شاید فکر کنید که این فریم ورک مانند روش های دیگر می باشد که در آن ها فریم ورک Ajax را تنها کتابخانه Ajax می دانند. در حقیقت، این فریم ورک نیز یک کتابخانه Ajax می باشد، ولی به خصوصیات دیگری اشاره می کند که می توانند کیفیت و سودمندی برنامه وب شما را افزایش دهند. این مطلب ما را به بررسی معماری ASP.NET AJAX که در شکل ۵-۵ آمده است مشتاق می کند.

اولین چیزی که شاید در مورد معماری فریم ورک ASP.NET AJAX به آن توجه کنید این است که کلاینت و سرور را پوشش می دهد. به غیر از یک مجموعه از کامپوننت ها و کتابخانه ها موارد بسیاری وجود دارد که در سمت سرور با کنترلهای سمت سرور و سرویس های ASP.NET حمایت می شوند.



شکل ۵-۵ معماری فریم ورک ASP.NET AJAX

فریم ورک کلاینت

مورد جالبی که درباره فریم ورک کلاینت وجود دارد این است که کتابخانه هسته ی این قسمت به کامپوننت های سرور متکی نیست. کتابخانه هسته می تواند در توسعه برنامه های ایجاد شده در Cold Fusion، PHP و زبانها و پلتفرمهای دیگر استفاده شود. با توجه به این انعطاف پذیری، معماری از نظر منطقی می تواند به دو قسمت تقسیم شود: فریم ورک کلاینت و فریم ورک سرور. درک چگونگی انجام عملیات در فریم ورک کلاینت حتی برای توسعه دهندگان سمت سرور نیز ضروری می باشد، بخاطر اینکه این بخش باعث موجودیت صفحات وب می شود. در هسته، کتابخانه Ajax مایکروسافت (Microsoft Ajax Library) قرار دارد.

Microsoft Ajax Library

همانطور که قبلا توضیح دادیم قلب فریم ورک Microsoft Ajax Library می باشد، همچنین به نام کتابخانه هسته (core) نیز شناخته می شود. کتابخانه ترکیبی از مجموعه فایلهای جاوا اسکریپت می باشد که می تواند مستقل از خصوصیات سرور استفاده شود. با کتابخانه هسته از طریق توضیح مفهوم هر یک از قطعه ها و لایه های آن، آشنا خواهید شد، ابتدا با زیر بنای آن شروع می کنیم: سیستم نوع(type system).

هدف سیستم نوع معرفی مفهومی شبیه برنامه نویسی شی گرا در جاوا اسکریپت می باشد؛ مانند کلاس ها، ارث بری، اینترفیس ها و مدیریت رویداد. این لایه همچنین نوع های موجود در جاوااسکریپت را بسط می دهد. برای مثال، نوع های String و Array در جاوااسکریپت هر دو برای آماده سازی توابع اضافی و آشنا بودن برای توسعه دهندگان ASP.NET بسط داده شده اند. سیستم نوع زمینه ی کاری را برای اتکای (راحتی) کتابخانه هسته ای Ajax ایجاد می کند.

در بالای سیستم نوع لایه کامپوننت ها (components layer) قرار دارند، این لایه بخش عمده ی کتابخانه هسته می باشد. این لایه از سری کردن (سریال سازی) JSON، ارتباطات شبکه، محلی سازی، فعالیتهای DOM و سرویسهای برنامه ASP.NET مانند اعتبار سنجی و پروفایل، حمایت می کند. همچنین تصویری از ساختمان مدل های قابل استفاده را معرفی می کند که می توانند به عنوان کنترل و رفتارها در صفحه دسته بندی شوند.

این لایه ما را به لایه بالاتر در کتابخانه متصل می کند. لایه برنامه (Application layer) یک بخش توضیحی برای Application model است. همانند چرخه حیات صفحه در ASP.NET این لایه یک مدل برنامه نویسی مدیریت رویداد را ایجاد می کند که می توانید برای کار با عناصر DOM، کامپوننت ها و چرخه حیات هر برنامه در مرورگر از آن استفاده کنید.

HTML، JavaScript و XML Script

صفحات وب با توانایی ASP.NET AJAX در HTML، جاوااسکریپت و یک تعریف کننده گرامر جدید بر پایه XML که XML Script نامیده می شود، نوشته می شوند. این موضوع بیش از یک گزینه برای نوشتن کد در سمت کلاینت را برای شما آماده می کند. عناصر تعریف شده در XML Script داخل تگ جدید اسکریپتی قرار می گیرند:

```
<script type="text/xml-script">
```

مرورگر می تواند تگ اسکریپت را تشخیص بدهد اما مکانیسمی برای پردازش نوع XML Script ندارد. در عوض، جاوااسکریپت موجود درفریم ورک ASP.NET AJAX می تواند اسکریپتها را تجزیه کرده، یک نمونه کامپوننت و کنترل را در صفحه ایجاد کند. درادامه تکه کدی را مشاهده می کنید که نشان می دهد XML Script چگونه برای نمایش یک پیغام بعد از اینکه صفحه لود شد مورد استفاده قرار می گیرد:

```
<script type="text/xml-script">
<page xmlns="http://schemas.microsoft.com/xml-script/2005">
<components>
     <application load="page_load" />
</components>
     </page>
</script>

<script type="text/javascript">
function page_load(sender, e){
     alert("Hello from XML-Script!");
```

```
}
</script>
```

دراین مثال، در تگ XML Scriptیک تابع جاوااسکریپتی به نام page_load را به رویداد load صفحه نسبت می دهد. اجرای این صفحه تابع page_load را بعد از فعال شدن رویداد load برای نمایش جعبه پیغام در کلاینت فراخوانی می کند.

XMl Script و جنبه های دیگر موجود در فریم ورک در یک مجموعه جداگانه ای از منابع به نام ASP.NET Future ارائه می شوند. این ویژگی ها در وضعیت پیش نمایش تکنولوژی اجتماع (CTP) موجود می باشند. CTP (Community Technology Preview)وضعیت فعلی محصولاتی را که هنوز امکان تغییر دارند را منعکس می کند. دراین صورت، ASP.NET Future CTP بسطی از هسته فریم ورک ASP.NET AJAX است که سرانجام در بسته هسته ادغام خواهد شد.

فریم ورک سرور

ASP.NET 2.0 از یک مجموعه باارزشی از کنترل ها و سرویس ها که قابل توسعه برای پشتیبانی Ajax می باشد، ایجاد شده است. این لایه از فریم ورک سرور ASP.NET AJAX server extensions نامیده می شود. server extensions به سه قسمت تقسیم می شوند: کنترل های سروری، پل سرویس های وب (web service) و پل سرویس های برنامه(Application services). هر یک از این قسمت ها در ارتباط نزدیک با مدل برنامه در کلاینت برای بهبود فعل و انفعال صفحات ASP.NET بر یکدیگر، هستند.

کنترل های سرور ASP.NET AJAX

یک مجموعه جدید از کنترل های سرور به نوار ابزار ASP.NET اضافه می شود که عمدتاً با دو کنترل مدیریت می شود. اولین نوع از این کنترل ها ScriptManager می باشد، که مفهوم یک صفحه با قابلیت Ajax را طرح ریزی می کند. یکی از وظایف عمده ی ScriptManager تنظیم بخشی ازصفحه است که هنگام دریافت جواب بصورت غیرهمان به روز رسانی می شود. کنترل دوم، UpdatePanel نام دارد، و برای مشخص کردن آن بخشی از صفحه که نیاز دارد به روز رسانی شود استفاده می شود. این دو کنترل با هم برای بالا بردن تجربه کاربر با جایگزینی ارسال و دریافت اطلاعات غیر همزمان بجای روش قبلی به کار می روند. این کنترل ها به جای به روز رسانی (refresh) کل صفحه در بخشی از صفحه که باید به روز رسانی شود مورد استفاده قرار می گیرند.

پل Web services

بطورنمونه، برنامه های وب به منابع بر روی سرورهای محلی شان محدود می شوند. گذشته از یک سری منابع خارجی، مانند تصاویر و فایلهای CSS، برنامه ها اجازه دسترسی به منابعی که در محدوده کلاینت هستند را ندارند. برای غلبه بر این مانع، server extensions در فریم ورک ASP.NET AJAX شامل یک

web services bridge است که دروازه ای برای شما ایجاد می کند تا از طریق آن سرویس های وب خارجی را از طریق اسکریپت سمت کلاینت فراخوانی کنید.

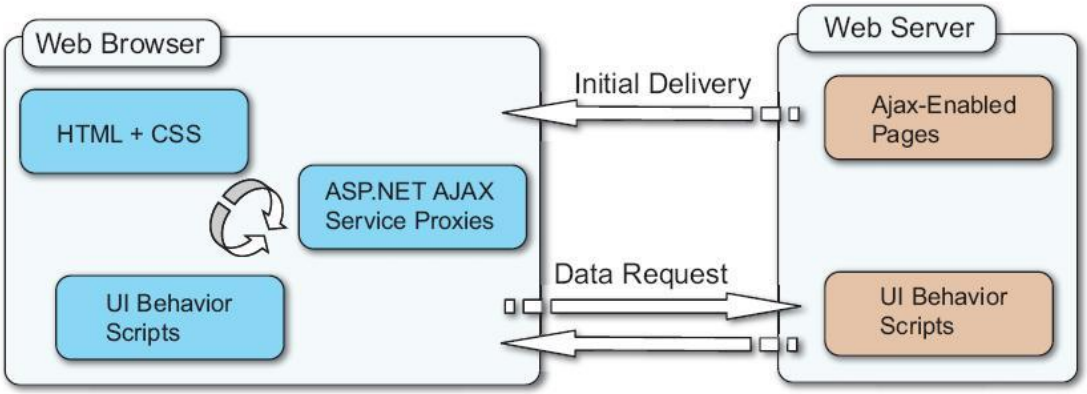
پل Application services

به علت اینکه ASP.NET AJAX با ASP.NET عمیقا ترکیب شده است دسترسی به برخی از سرویس های برنامه مانند اعتبار سنجی (authentication) و سابقه (profile) که به برنامه اضافه شده باشند خیلی بدون دردسر است. این ویژگی قادر می سازد عملیاتی مانند تصدیق (verifying) اعتبارنامه یک کاربر و دسترسی به اطلاعات سابقه کاربر از طریق اسکریپت های کلاینتی سازماندهی شوند.

اکنون که یک ایده کلی از بخش های سازنده فریم ورک دارید بهتر است با سوالی که چگونه این بخش ها را بطور موثر بکار ببریم؟ شروع کنیم. جواب این سوال منجر به بررسی دو سناریوی توسعه می شود.

مدل توسعه ی کلاینت محوری (Client-centric development model)

طراحی انعطاف پذیر از معماری برنامه های وب دو سناریو را فراهم می آورد. اولین سناریو در سمت کلاینت پیاده سازی شده و به عنوان ***Client-centric development model*** شناخته می شود. ارزش دارد زمانی را برای درک اینکه این مدل ها چگونه کار می کنند و چه زمانی از هر کدام استفاده می شود صرف کنید. در مدل کلاینت محوری لایه نمایش از اسکریپت های کلاینتی که از DHTML و جاوااسکریپت استفاده می کند، تشکیل شده است. این بدین معنی است که یک برنامه فعال تر و هوشمندتر، مرورگر را از تحویل درخواست به سرور زمانیکه صفحه برای دفعه اول بار گذاری می شود، مطلع می سازد. بعد از این عملیات ارتباط بین برنامه سمت کلاینت با سرور محدود می شود تا زمانیکه نیاز به داده برای به روز رسانی صفحه باشد. این مدل موجب ارتباط بهتر کاربر و برنامه وب می شود. شکل ۶-۵ مدل توسعه کلاینت محور را نشان می دهد.



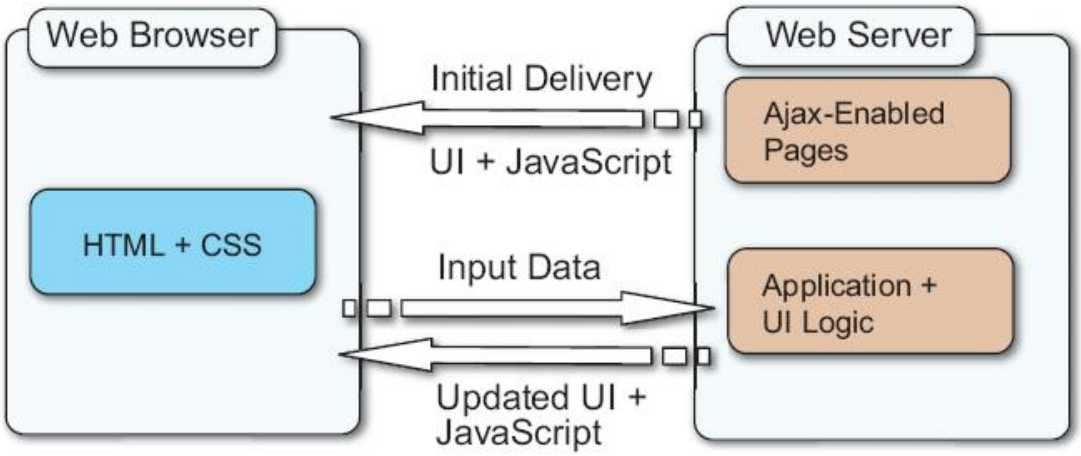
شکل ۶-۵ مدل کلاینت محور توسعه

مدل سرور محوری توسعه

در مدل سرور محور منطق برنامه و بیشتر واسط کاربری در سمت سرور باقی می مانند. بیشتر تغییرات در واسط کاربری به برنامه مرورگری ارسال می شود بجای اینکه در سمت کلاینت اجرا شود. این روش مانند مدل قبلی

صفحات ASP.NET است مدلی که در آن واسط کاربری را با ارسال داده به مرورگر به عنوان خروجی تولید می کند. تفاوت بین این مدل و مدل قبلی وب در Asp.net این است که آن قسمت از واسط کاربری که نیازدارد بعنوان خروجی نمایش داده شود به برنامه مرورگری پاس داده می شود نه مانند مدل قبلی که تمام صفحه ارسال می شد.

شکل ۷-۵ مدل سرور محوری توسعه را نشان می دهد.



شکل ۷-۵ مدل توسعه سرور محور

این روش توسعه دهندگان وب را قادر می سازد که هسته واسط کاربری و منطق برنامه شان را در سمت سرور نگه دارند. همچنین قسمت جذاب مطلب این است برنامه ها می توانند در صورت غیرفعال کردن جاوااسکریپت در مرورگر توسط کاربر مانند برنامه های معمولی وب رفتار کنند.

اهداف ASP.NET AJAX

۱-استفاده آسان، وجود یک فریم ورک تولید کننده قوی

اضافه کردن قابلیت Ajax به برنامه های وب توسط این فریم ورک بسیار ساده می باشد.

این کار توسط یک کتابخانه کلاینتی و یک مجموعه از کنترل های سروری که نحوه کارکرد ساده ی دارند، فراهم می شود.

۲- ترکیب مدل برنامه نویسی سرور

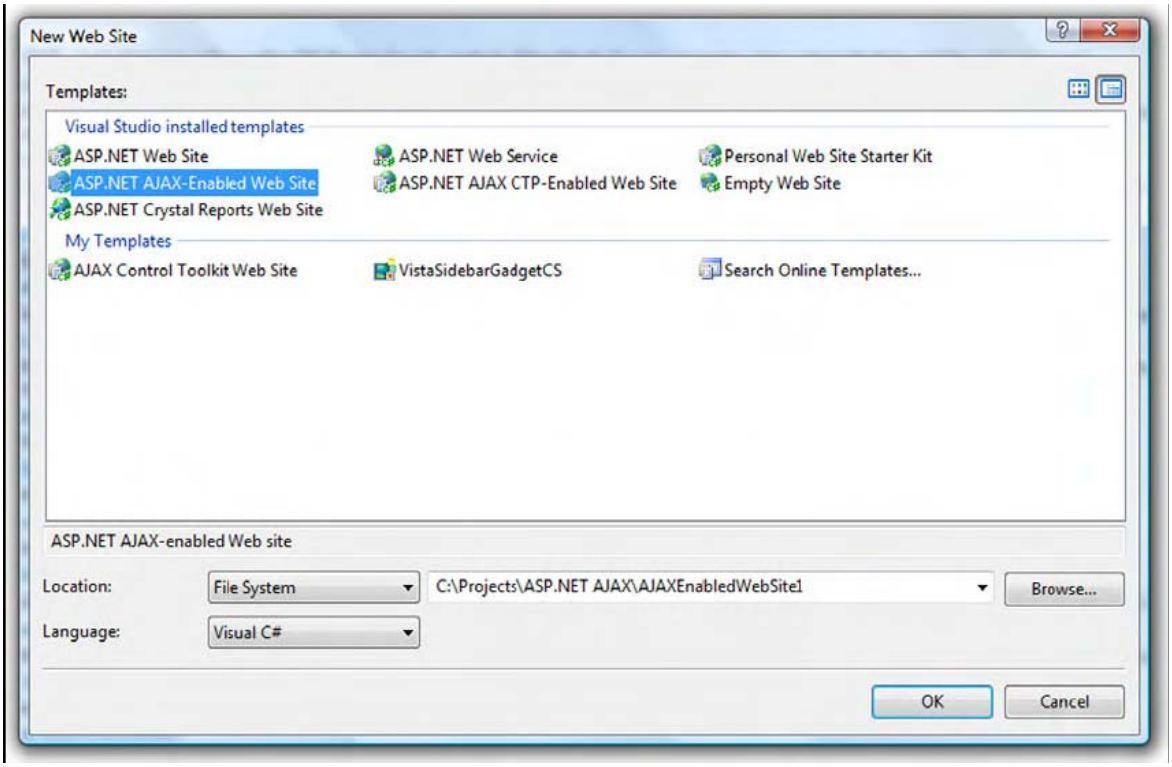
کنترل های سروری برای توسعه دهندگان Asp.net یک محیط راحت برای توسعه برنامه های وب فراهم می کند. این کنترل ها نیاز به جاوااسکریپت برای فعال کردن Ajax در صفحه را حذف می کند و نیاز به دانش زیادی در مورد شی XHL را برطرف می سازند.

۳- ابزار world-class و کامپوننت ها

کامپوننت ها و ابزارهای ساخته شده بر روی فریم ورک فقط برای توسعه فریم ورک نیست این ابزارها شامل بخش های برای debug- trace-profiling می باشند.

یک برنامه سمت سرور

ویژوال استودیوی ۲۰۰۵ را اجرا کرده و از قسمت New Web Site گزینه ASP.NET AJAX-Enabled Web Site را انتخاب کنید. با انتخاب این گزینه سایتی را ایجاد می کند که به اسمبلی ASP.NET AJAX یعنی System.Web.Extensions.dll که متعلق به کش اسمبلی عمومی (GAC) است اشاره می کند. همچنین یک فایل web.config پیچیده که شامل تنضیمات اضافی برای ترکیب با فریم ورک ASP.NET AJAX می باشد را ایجاد می کند.



شکل ۸-۵ گزینه ساخت برنامه ASP.NET AJAX-Enabled Web Site

یک کلاس به نام HumanResources.cs به منظور جستجوی منطقی کارمندان ایجاد کرده و کد زیر را در آن کپی کنید:

```
using System;
public static class HumanResources
{
    public static int GetEmployeeCount(string department)
    {
        int count = 0;
        switch (department)
        {
            case "Sales":
                count = 10;
                break;

            case "Engineering":
                count = 28;
```

```
                break;

            case "Marketing":
                count = 44;
                break;

            case "HR":
                count = 7;
                break;

            default:
                break;
        }

        return count;
    }
}
```

این کلاس شامل یک متد بنام GetEmployeeCount می باشد، که نام دپارتمان را به عنوان پارامتر ورودی دریافت می کند. این متد از یک switch برای مشخص کردن تعداد کارمندان هر بخش استفاده می کند.

زمانیکه یک وب سایت جدید ایجاد می کنید، یک صفحه پیش فرض بنام Default.aspx در آن ایجاد می شود که محتوای آن مانند کد زیر خواهد بود:

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeFile="Default.aspx.cs" Inherits="_Default" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Untitled Page</title>
</head>
<body>
    <form id="form1" runat="server">
        <asp:ScriptManager ID="ScriptManager1" runat="server"
               />
        <div>
        </div>
    </form>
</body>
</html>
```

تفاوت بین این صفحه با صفحه پیش فرض ایجاد شده با ویژوال استودیو در این است که یک کنترل ScriptManager در آن اضافه شده است. ScriptManager مسئول تشخیص دریافت اطلاعات در سرور برای مرورگر سمت کلاینت و مدیریت به روز رسانی جزئی در صفحه می باشد. اگر هنوز با این مفهوم آشنا نیستید نگران نباشید چرا که اگر با ویژوال استودیو کار کنید مفهوم آن برای شما ملموس تر خواهد بود.

حال کد زیر را به داخل تگِ DIV پایین تگ ScriptManager اضافه کنید:

```
<div>
            <asp:ListBox AutoPostBack="true" runat="server"
                    ID="Departments"
OnSelectedIndexChanged="Departments_SelectedIndexChanged">
                <asp:ListItem Text="Engineering"
Value="Engineering" />
                <asp:ListItem Text="Human Resources"
Value="HR" />
                <asp:ListItem Text="Sales" Value="Sales" />
                <asp:ListItem Text="Marketing"
Value="Marketing" />
            </asp:ListBox>
        </div>
        <br />
        <div>
            <asp:Label ID="EmployeeResults" runat="server" />
</div>
```

از یک لیست برای نمایش نام دپارتمان ها جهت انتخاب استفاده می شود. خاصیت AutoPostBack کنترل، به منظور درخواست یک پاسخ از سرور با مقدار true تنظیم می شود. این عمل یعنی انتخاب نام یک دپارتمان از لیست که باعث فعال شدن رویداد SelectedIndexChanged و فراخوانی متد Departments_SelectedIndexChanged در کد می شود. در انتهای فرم یک کنترل Label قرار دارد که نتیجه عملیات را نمایش می دهد. برای تکمیل برنامه یک واسط کاربری را پیاده سازی کنید که برای جستجوی تعداد کارمندان موجود در دپارتمان انتخابی باشد:

```
protected void Departments_SelectedIndexChanged(object
sender,EventArgs e)
{
     EmployeeResults.Text = string.Format("Employee count:
     {0}",HumanResources.GetEmployeeCountDepartments.SelectedV
alue));
}
```

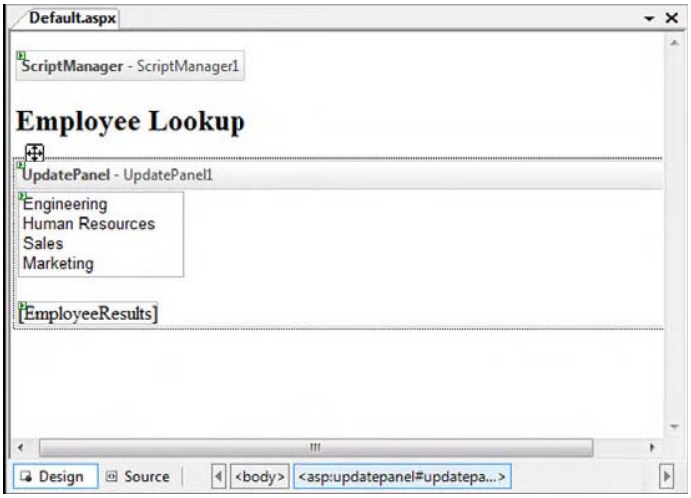
زمانیکه برنامه اجرا شود و یکی از دپارتمانها انتخاب گردد، نتیجه ی مانند شکل زیر را می توان مشاهده کرد:

شکل ۹-۵ نمای از اجرای مثال Employee

برنامه همان کاری که انتظارش را داشتیم انجام می دهد: انتخاب یک دپارتمان از صفحه، یک عدد را برگردانده و آن را در صفحه نمایش می دهد. تنها موضوع این است که هر زمان که یک دپارتمان انتخاب می شود صفحه refresh می شود. حل این مشکل آسان می باشد؛ محتویات صفحه را داخل یک کنترل UpdatePanel قرار دهید:

```
<asp:UpdatePanel ID="UpdatePanel1" runat="server">
        <ContentTemplate>
            <div>
                <asp:ListBox AutoPostBack="true"
runat="server" ID="Departments"


OnSelectedIndexChanged="Departments_SelectedIndexChanged">
                    <asp:ListItem Text="Engineering"
                        Value="Engineering" />
                    <asp:ListItem Text="Human Resources"
                        Value="HR" />
                    <asp:ListItem Text="Sales" Value="Sales"
                        />
                    <asp:ListItem Text="Marketing"
                        Value="Marketing" />
                </asp:ListBox>
            </div>
        </ContentTemplate>
</asp:UpdatePanel>
```

نتیجه کار در قسمت Design ویژوال استودیو به صورت زیر خواهد بود:



شکل ۱۰-۵ طراحی برنامه با UpdatePanel

بصورت پیش فرض محتوای داخل تگ ContentTemplate از کنترل UpdatePanel زمانی که یک درخواست غیر همزمان انجام می شود بصورت خودکار به روز می شود.

کنترل UpdateProgress

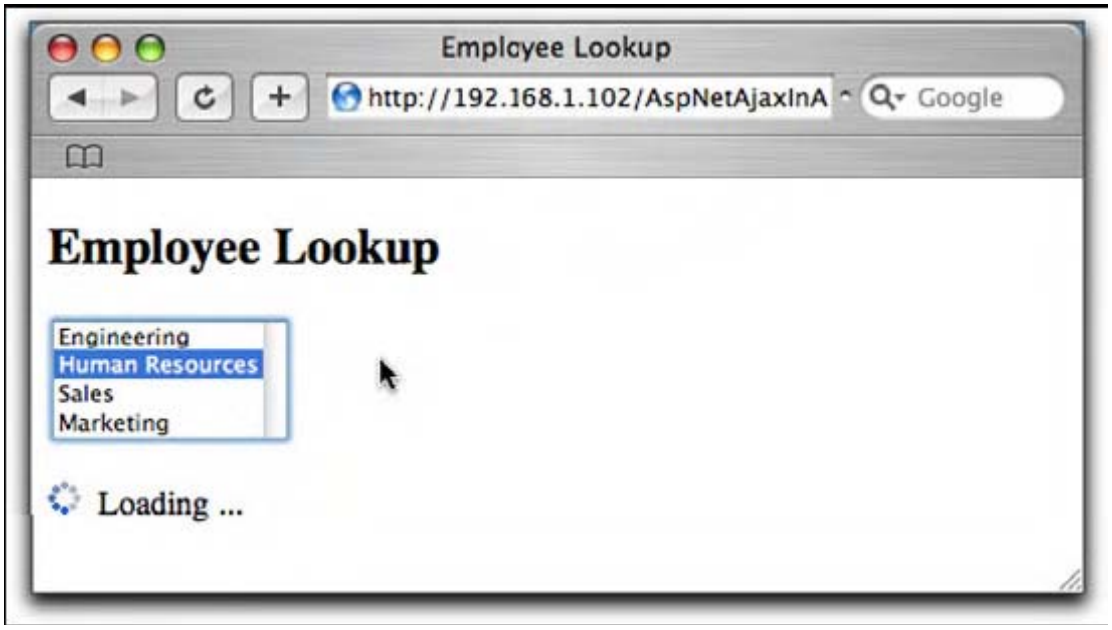
معمولا یک زمان تاخیری همیشه بین ارسال درخواست و دریافت جواب در برنامه ها وجود دارد. این زمان در برنامه های وب به عوامل زیادی از جمله ترافیک شبکه، حجم اطلاعات ارسالی و دریافتی بستگی دارد.

قبل از ارائه Ajax، refresh شدن یک صفحه برای اغلب کاربران دلیلی بر پردازش چیزی و یا قبول عملیاتشان بود. حال، با جلوگیری از درخواست و پاسخ معمولی توسط Ajax، کاربر دلیلی بر اینکه تا کامل شدن عملیات چیزی در پشت صحنه انجام می شود، ندارند. آن ها به یک نوع بازخور بصری برای نشان دادن اینکه کارشان در حال پیشرفت می باشد نیاز دارند.

کنترل UpdateProgress برای حل این مشکل ارائه شده است. هدف آن آماده سازی یک راهنمای بصری برای کاربر هنگام ایجاد یک درخواست asynchronous می باشد. برای استفاده از آن کافی است کد زیر را اضافه کرده و از یک تصویر برای نمایش این زمان انتظار استفاده کنید:

```
<asp:UpdateProgress ID="UpdateProgress1" runat="server">
   <ProgressTemplate>
       <img src="images/indicator.gif" />  Loading
...
   </ProgressTemplate>
</asp:UpdateProgress>
```

با اجرای دوباره برنامه هنگام انتخاب دپارتمان از لیست تا زمان لود شدن اطلاعات یک تصویر نمایش داده می شود.(همانند شکل زیر)



شکل ۱۱-۵ اجرای برنامه با نوار UpdateProgress

البته در صورت اجرا بر روی سرور محلی ایجاد شده بر روی سیستم خودتان، به علت سرعت انجام عملیات شانس کمی برای مشاهده این تصویر دارید. برای اینکه بتوانید این تصویر را مشاهده کنید یک دستور sleep به کدتان اضافه کنید:

```
protected void Departments_SelectedIndexChanged(object
sender,EventArgs e)
    {
        EmployeeResults.Text = string.Format("Employee count:
                    {0}",HumanResources.GetEmployeeCount
                        (Departments.SelectedValue));
        System.Threading.Thread.Sleep(2000);
    }
```

یک برنامه ساده بر محور کلاینت

نوشتن برنامه ها بر محور سرور بسیار ساده بودند ولی دارای نقطه ضعف می باشند. هنوز بیشتر عملیات در سمت سرور اجرا می شود. ولی می خواهیم عملیات سمت سرور محدود شوند، و تنها زمانی که به داده ای نیاز داشته باشیم آن را از سرور دریافت کنیم.

برای نوشتن برنامه بر محور کلاینت در پنجره Add new Item گزینه Place Code in Separate File را غیر فعال کنید. و یک وب سرویس بنام HRService.asmx را به برنامه اضافه کنید.

برای کامل شدن راه حل کد زیر را در برنامه کپی کنید:

```
using System;
using System.Web;
using System.Web.Services;
using System.Web.Services.Protocols;
using System.Web.Script.Services;
```

```
[ScriptService]
[WebService(Namespace = "http://tempuri.org/")]
[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
public class HRService : System.Web.Services.WebService {

    [ScriptMethod]
    [WebMethod]
    public int GetEmployeeCount(string department)
    {
        return HumanResources.GetEmployeeCount(department);
    }
}
```

ابتدا به فضای نامی System.Web.Script.Services توجه کنید. این فضای نامی قسمتی از هسته فریم ورک ASP.NET AJAX می باشد که برخی از ارتباطات شبکه ای و عملیات اسکریپتی را کپسوله می کند. سپس به خاصیت های جدیدی که کلاس ([ScriptService]) و تعریف متد ([ScriptMethod]) را در web service در بر می گیرند توجه کنید. این خاصیت ها توسط ASP.NET Ajax تجزیه شده و برای تعیین اینکه چه قسمتهایی از سرویس در اختیار پراکسی های جاوااسکریپت قرار بگیرند استفاده می شوند. البته خاصیت ScriptMethod ضروری نمی باشد، اما می توانید از آن برای دستکاری تنظیمات برخی از متدها استفاده کنید.

اگر در مرورگر تان فایل ASMX را با یک js/ در انتهای URL آن مشاهده کنید، یک مجموعه پراکسی جاوااسکریپت برای این سرویس ایجاد شده را نشان می دهد. شکل ۱۲-۵ این مورد را نشان می دهد.

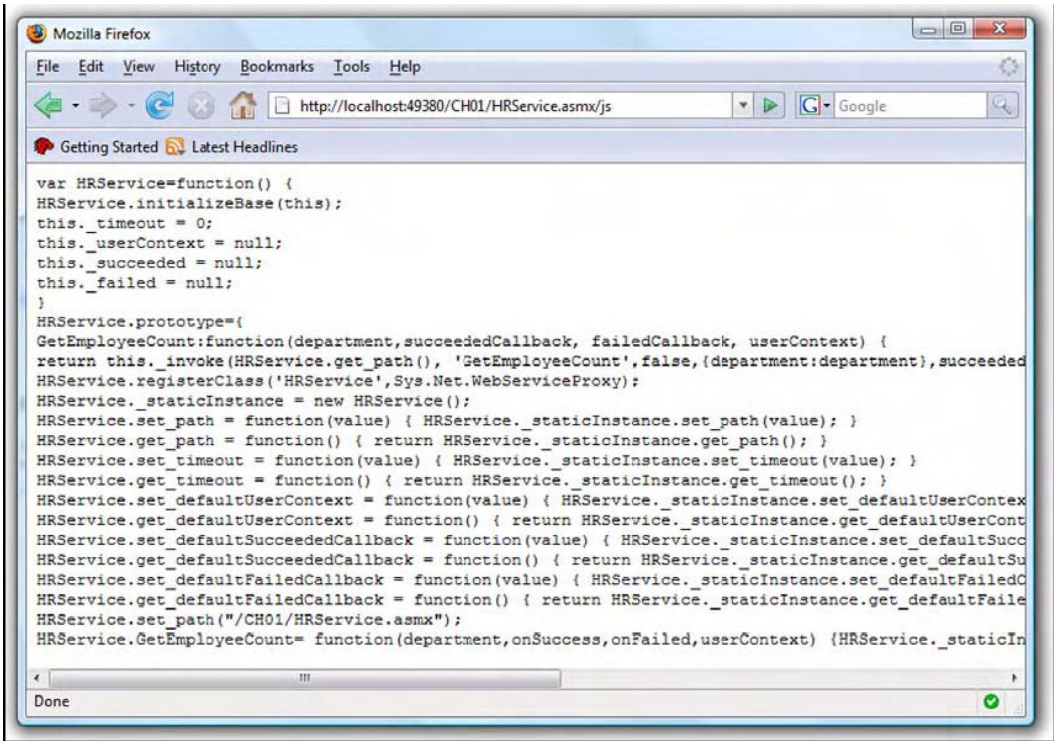
با استفاده از web service آماده، می توانید یک صفحه جدید برای این راه حل ایجاد کنید. برای شروع یک فرم وب به نام EmployeeLookupClient.aspx به سایت اضافه کنید. ابتدا احتیاج به اضافه کردن کنترل ScriptManager به صفحه برای پشتیبانی از ajax می باشد. حال، یک ارجاع به سرویس وب محلی تعریف می کنید، برای ایجاد پروکسی جاوااسکریپت برای سرویس می توانید مانند کد زیر آن را فراخوانی کنید:

```
<asp:ScriptManager ID="ScriptManager1" runat="server">
     <Services>
          <asp:ServiceReference Path="HRService.asmx" />
     </Services>
</asp:ScriptManager>
```

برای کامل شدن این قسمت کد زیر را در برنامه بنویسید:

```
<div>
     <select id="Departments" size="5">
          <option value="Engineering">Engineering</option>
          <option value="HR">Human Resources</option>
```

```
            <option value="Sales">Sales</option>
            <option value="Marketing">Marketing</option>
        </select>
</div>
<br />
<div>
        <span id="employeeResults"></span>
            <span id="loading" style="display:none;">
            <img src="images/indicator.gif" alt="" />
              Loading ...
        </span>
</div>
```



شکل ۱۲-۵ پروکسی جاوااسکریپت برای فرخوانی سرویس وب

به این علت که نمی توان به منطق واسط کاربری و یا عملیاتی که از سمت سرور می آید اعتماد کنید، می توانید از عناصر Html بجای کنترل های سنگین سروری استفاده کنید. در اینجا عنصر select برای لیست دپارتمان و عنصر span برای نمایش نتیجه اطلاعات دریافتی از سمت سرور بکار می روند. برای ایجاد صفحه مورد نظر کدهای جاوااسکریپت زیر را به آن اضافه کنید:

```
<script type="text/javascript">
    <!--
    var departments = null;
    Sys.Application.add_load(page_load);
    Sys.Application.add_unload(page_unload);

    function page_load(sender, e){
        departments = $get("Departments");
```

```
            $addHandler(departments, "change",
                 departments_onchange);
        }
        function page_unload(sender, e){
            $removeHandler(departments, "change",
                 departments_onchange);
        }
        function departments_onchange(sender, e){
            $get("employeeResults").innerHTML = "";
            $get("loading").style.display = "block";
            var selectedValue = departments.value;
            HRService.GetEmployeeCount(selectedValue,
            onSuccess);
        }
        function onSuccess(result){
            $get("loading").style.display = "none";
            $get("employeeResults").innerHTML = "Employee count:
            " + result;
        }
        //-->
</script>
```

همان طور که می بینید متدهای load و unload در مرورگر با مدل برنامه تنظیم شده اند. اگر به کتابخانه هسته که قبلا گفته شد نگاه بیاندازید خواهید دید که فریم ورک کلاینت یک چرخه حیات صفحه مانند آنچه در Asp.net است فراهم می کند. در این صورت از رویداد load برای ثبت تغییراتی که در لیست دپارتمان صورت می گیرد، استفاده کرد. همچنین از متد unload برای حذف تغییراتی که ثبت شده است استفاده می شود.

در این کد برخی موارد جدید می باشند: دستوراتی که با $ شروع می شوند. این دستورات میانبر یا نام مستعار معادل دستور جاوااسکریپت شان می باشند که در آخر به دستور معادل خود ترجمه می شوند. برای مثال $get معادل دستور document.getElementById می باشد.

زمانیکه یک دپارتمان در صفحه انتخاب می شود متد زیر برای بازیابی تعداد کارمندان سرویس وب را فراخوانی می کند.

```
HRService.GetEmployeeCount(selectedValue, onSuccess);
```

پارامتر اول در این متد دپارتمان انتخاب شده از لیست می باشد. پارامتر دوم نام تابعی می باشد که هنگام اتمام موفق متد فراخوانی می شود. هنگامی که تابع onSuccess اجرا شود واسط کاربری به روز (update)می شود.

فصل پنجم: خطایابی برنامه های Ajax

یکی از مراحل مهم چرخه توسعه نرم افزار، تست نرم افزار می باشد. هر نرم افزار نوشته شده بدون خطا نیست به همین علت نرم افزارهای دیباگ به ابزارهای با ارزشی تبدیل شدند.

به صورت سنتی دیباگ کدهای جاوااسکریپت مشکل می باشد. تا چندی پیش مرورگرها به صورت محدودی از دیباگ جاوااسکریپت حمایت می کردند. اولین نسخه Safari کنسول جاوااسکریپت نداشت. Opera بعد از نسخه ۸ این کنسول را اضافه کرد. Firefox از کنسول جاوااسکریپت در نسخه ۱,۰ خود حمایت کرد. IE در نسخه ۷,۰ هم هنوزکنسولی برای جاوااسکریپت ندارد، و تنها هنگام ایجاد خطا پنجره ای برای اعلام خطا نشان می دهد.

اینجا یکی از ابزارهای مفید دیباگ را معرفی خواهیم کرد.

FireBug

Firefox داری قابلیتی به نام add-on می باشد که می توان نرم افزارهایی ایجاد کرد که از توانایی کار با Firefox برخوردار می باشند و این نرم افزارها را به Firefox اضافه کرد. در سال ۲۰۰۶، Joe Hewitt، نرم افزار Firebug را به عنوان ابزار جدیدی برای کمک به توسعه دهندگان در ایجاد و دیباگ برنامه های وب معرفی کرد. Firebug به صورت توکار اجازه بررسی و بازرسی DOM صفحه ای که اکنون لود شده است را می دهد، اطلاعاتی در مورد قالب عناصر ارائه می کند، و قسمتی که برای توسعه دهندگان Ajax جالب می باشد، نظارت (monitoring) بر همه ارسال ها و دریافت های که از طریق شی XHR انجام می گیرد.

نصب و آماده سازی

Firebug را می توانید به صورت رایگان از سایت www.getfirebug.com دانلود کنید. Firebug به صورت یک XPI دانلود می شود، به این معنی که خود Firefox می داند که چگونه آن را نصب کند. در مرورگر Firfox با کلیک بر روی لینک دانلود، به بسته اجازه نصب بدهید. روی کلید OK پیغام نصب Firebug کلیک کنید. (بعد از آن نیاز به Restart مرورگر دارید.)

Firebug به دو روش به پنجره Firefox اضافه می شود:

۱- ابتدا، در سمت راست نوار وضعیت، پانل کوچکی را اضافه می کند که خطاهای مربوط به صفحه را نمایش می دهد.(دایره توپر سبز در صورتی که بدون خطا باشد، و علامت "x" قرمز رنگ اگر خطائی داشته باشد.)

۲- روش دوم، پانل اصلی که اطلاعاتی درباره صفحه لود شده فعلی را نمایش می دهد و در پائین پنجره نمایش داده می شود. این پانل می تواند به صورت پیش فرض مخفی یا آشکار باشد و می تواند با کلیک بر روی پانل وضعیت از یک حالت به حالت دیگر برود.

واسط

واسط فایرباگ شامل شش نوار در پانل اصلی می باشد:

- نوار اولی Console نام دارد، و شامل خط فرمان پایه برای جاوااسکریپت می باشد. در این نوار پیغام های خطا نمایش داده می شوند، همچنین می توان با نوشتن دستورات جاوااسکریپت با صفحه تعامل داشت.
- نوار دومی HTML نامیده می شود، و شامل بازرس DOM پایه می باشد. در این نوار، می توانید گره های DOM،Style و Layout را به صورت سلسله مراتبی ببینید.
- نوار سوم CSS می باشد، و بعنوان بازرس CSS، اجازه مشاهده همه ی شیوه نامه ها (CSS) و ویرایش قالب ها (Style) را به شما می دهد.
- نوار چهارم Script می باشد، و شامل دیباگر کدهای جاوااسکریپت می باشد. می توانید در فایل جاوااسکریپت نقاط توقف اجرا (breakpoint) ایجاد کنید.
- نوار پنجم DOM نامیده می شود، که تمامی اشیا موجود در صفحه را نمایش می دهد.
- نوار ششم Net می باشد که شامل نشانهای گرافیکی برای تمامی منابعی می باشد که توسط صفحه لود شده اند. زمان لازم برای لود این منابع را نیز نمایش می دهد.

در ادامه همه نوارهای موجود در فایرباگ مفصلا شرح داده خواهند شد.

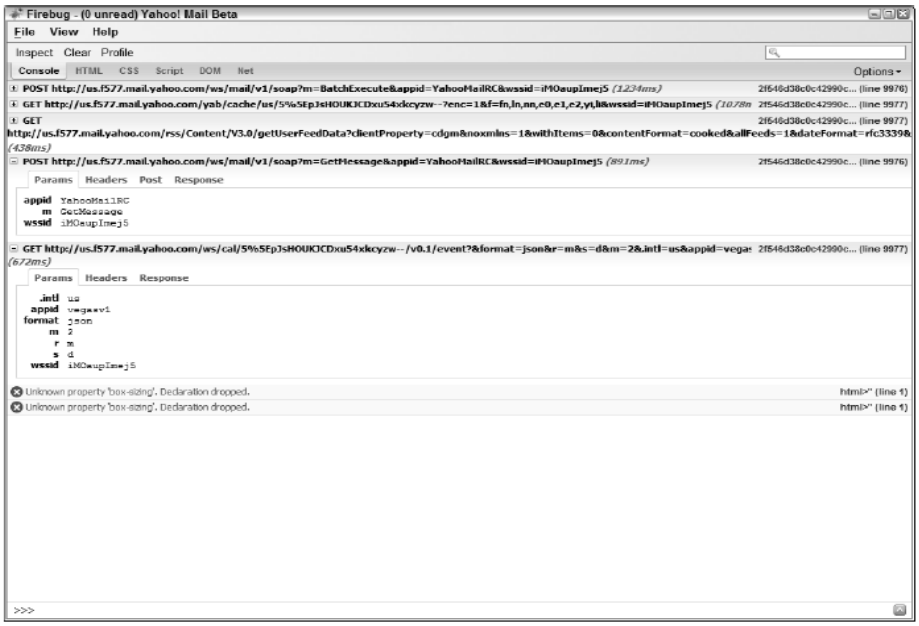
واقعه نگاری (Logging) XHR

اغلب اوقات شی XHR برای ایجاد درخواست HTTP استفاده می شود، این اطلاعات در نوار کنسول به صورت یک آیتم یک خطی نوشته می شوند. این خط شامل متد درخواست (عموماً GET و یا POST) و همچنین به دنبال آن URL استفاده شده برای درخواست می باشد. این خط می تواند با کلیک بر روی URL بسط داده شده و نوارهای بیشتری را نمایش دهد.

اگر درخواست GET ارسال شده باشد، سه نوار موجود خواهد بود: Params،Headers و Respons .

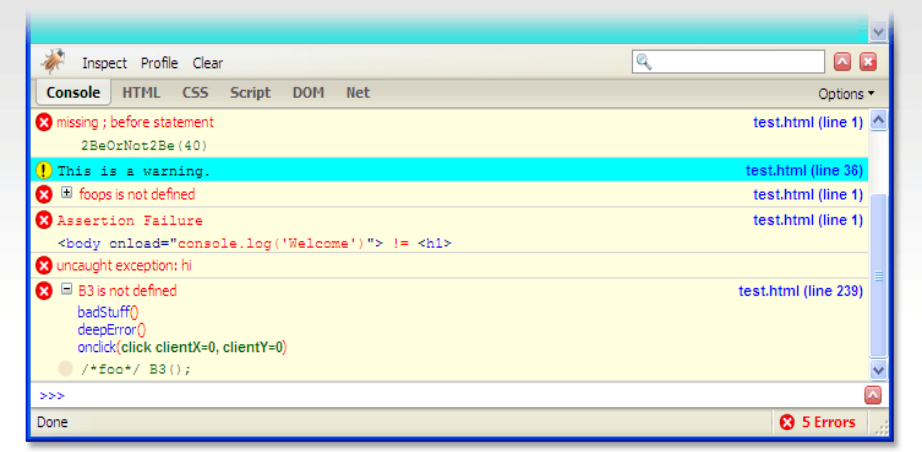
- نوار Params پارامترهای رشته ارسالی درخواست که به صورت نام-مقدار می باشد را در بردارد.
- نوار Header شامل هدرهای پاسخ HTTP که همراه با داده های درخواست ارسال می شوند، می باشد. این نوار شامل اطلاعات مفیدی مانند کوکی ها، نوع محتوا، و زمان ارسال پاسخ می باشد.
- نوار Response شامل اطلاعات ارسالی از سمت سرور به کلاینت می باشد. ممکن است که این اطلاعات فرمت مناسبی نداشته باشند و مجبور باشید برای مطالعه، آن ها را در یک وایشگر کپی کنید.

در صورتی که نوع درخواست POST باشد، تعداد نوارها چهار تا خواهد بود. علاوه بر موارد فوق نوار Post نیز داده ارسالی به سرور را نمایش خواهد داد. داده ها برای درخواست POST به عنوان بدنه درخواست ارسال می شوند.



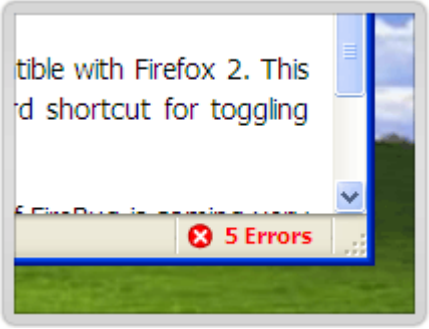
خطایابی

زمانی که در برنامه خطا وجود داشته باشد، فایرباگ فورا به شما خبر داده و اطلاعات جامع و مفیدی درباره خطاهای موجود در جاوااسکریپت، CSS و XML را به شما می دهد.



نمایش خطا در نوار وضعیت

در سمت راست نوار وضعیت مرورگر Firefox یک آیکون سبز رنگ را می توانید ببینید. (بعد از نصب Firebug) این آیکون روش فایرباگ برای اعلام صحت همه چیز می باشد. زمانی که این آیکون به ضربدر قرمز تبدیل شود، نشان دهنده ایجاد مشکل خواهد بود. با کلیک بر روی "x" فایرباگ کنسول خطا را باز می کند که تمامی خطاهای موجود در صفحه را به شما نشان خواهد داد.

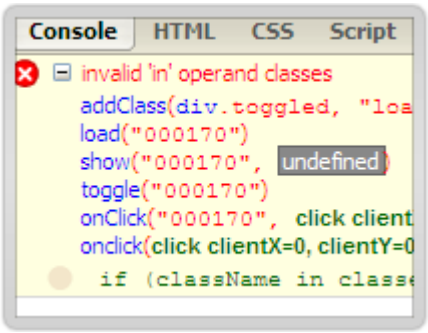


تفکیک خطا

اغلب مرورگرها خطاها را در یک پنجره به صورت یک لیستی شامل خطاهای تمام صفحاتی که تا به حال دیده اید نمایش می دهند. فایرباگ به گونه دیگری این کار را انجام می دهد، فایرباگ تنها خطاهای موجود در صفحه ای را که اکنون در حال تماشا هستید را نمایش می دهد.

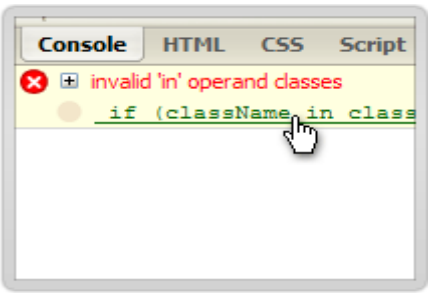
اطلاعات یک خطا

در مورد خطاهای جاوااسکریپت اطلاعات کاملی درباره خطای پیش آمده را نشان می دهد. در مورد خطا توضیح داده و فایل و شماره خطی که خطا در آن ایجاد شده را نیز اعلام می کند. اغلب با کلیک بر روی پیکان می توانید توضیحات را بسط داده و اطلاعات کامل پشته ی زمان اجرا، شامل مقادیر آرگومان های پشته را ببینید.



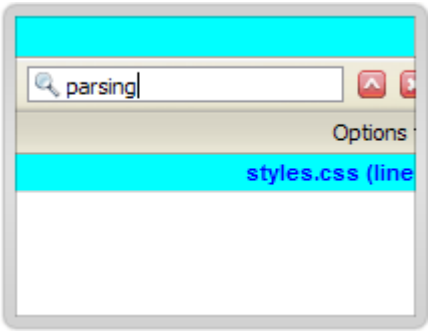
پرش به موقعیت خطا

خطاهای بوجود آمده یک لینک به فایل و خطی که خطا در آن ایجاد شده است دارند. با کلیک بر روی این لینک به دیباگر جاوااسکریپت فایرباگ و یا CSS inspector می روید و بعد از آن شروع به حل مشکل پیش آمده می کنید. برخی از خطاها شامل تکه کدی از کد خطا دار می باشند، که آن نیز لینکی به فایل اصلی می باشد.



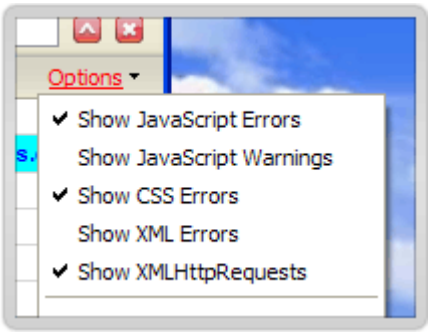
جستجوی سریع

با استفاده از جعبه جستجوی سریع، می توانید کنسول فایرباگ را برای نمایش خطاهایی که با متنی که شما جستجو می کنید یکسانند فیلتر کنید. سطرهای جدیدی که به کنسول اضافه می شوند، تنها زمانی ظاهر می شوند که با متن موجود در جعبه جستجوی سریع مطابقت داشته باشند.



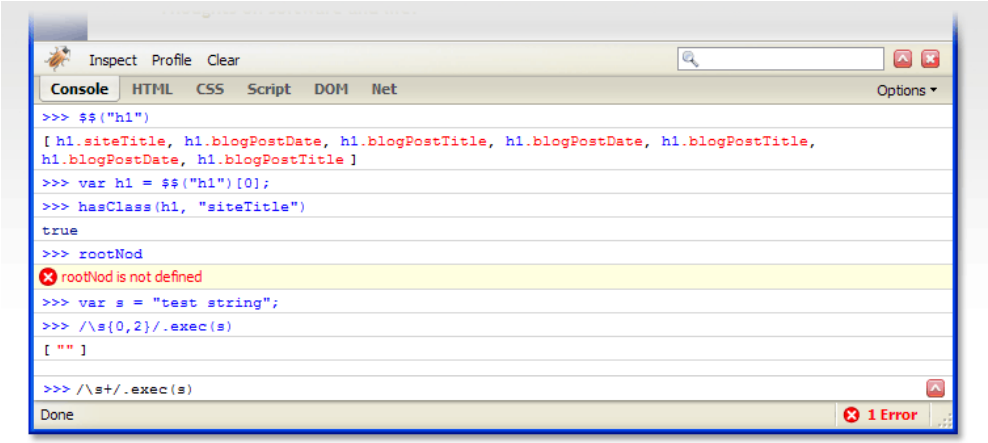
فیلترینگ خطاها

فایرباگ توانایی گزارش دهی درباره ی خطاهای موجود در فایل های جاوااسکریپت، CSS، یا XML را دارد. اگر به خطاهای یکی از این زبان ها علاقمند نباشید، با انتخاب نکردن هر کدام که نمی خواهید ببینید در منوی Options کنسول می توانید به این نتیجه برسید.



خط فرمان جاوااسکریپت

خط فرمان یکی از قدیمی ترین ابزار در toolbox برنامه نویسی می باشد. فایرباگ یک خط فرمان قدرتمند برای جاوااسکریت را در اختیار شما قرار می دهد.



احتمالا با این مساله برخورد کرده اید که بررسی کدهای DOM و مشاهده نتیجه آن خیلی آسان نیست اما با خط فرمان براحتی می توانید دستور خود را نوشته و نتیجه اجرای آن را مشاهده کنید.

```
>>> var number = "(408) 328-
>>> /\(\d{1,3}\)/.exec(numbe
[ "(408)" ]
>>> /\(\d{1,3}\)\s+\d+/.exec
[ "(408) 328" ]
>>> /\(\d{1,3}\)\s+(\d+)/.ex
[ "(408) 328", "328" ]
>>>
```

کامل شدن خودکار عبارات (Autocomplete)

با استفاده از کلید tab می توانید به صورت خودکار نام متغیرها و خاصیت اشیا را کامل کنید. مجموعه ممکن به صورت چرخشی نشان داده می شود، و می توانید با shift-tab به عقب برگردید.

```
>>> document.get
Done
```

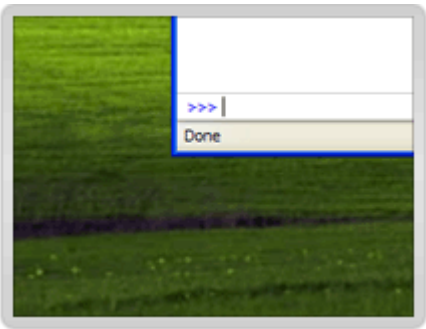
ویرایشگر راحت

اگر هنگام نوشتن خطی احساس می کنید که محدوده کارتان کوچک می باشد، نگران نباشید. خط فرمان فایرباگ می تواند به یک ویرایشگر بزرگ تبدیل شود پس از آن می توانید کدتان را در یک خط وارد کنید.

```
Options *
function findNode(node, criteria) {
    while (node) {
        if (criteria(node)) {
            return node;
        }
        node = findNext(node);
    }
    return null;
}
Run Clear Copy
```

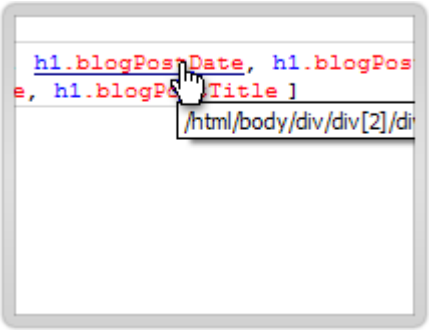
هوشمندی خط فرمان

یک دسته کد جاوااسکریپت را نوشته و می خواهید صحت آنها را تست کنید. تنها کاری که باید انجام دهید کپی آنها به خط فرمان می باشد. اگر بیش از یک خط باشد، به صورت خودکار توسط فایرباگ یک ویرایشگر بزرگ برای اسکریپت شما آماده خواهد شد.



کاربرد کلید چپ موس

برخلاف خط فرمانهای سنتی، خروجی هر فرمان یک رشته ایستا نمی باشد، بلکه به صورت hypertext می باشد. با کلیک بر روی خروجی می توانید بررسی کنید هر یک از اشیا در کدام تب فایرباگ مناسب تر است.



کاربرد کلید میانی موس

اگر تا به حال جادو کلیک-میانی و tab های مرورگر را نیاموخته اید، اکنون زمان خوبی برای این کار است. زمانی که بر روی یک لینک در Firefox کلیک-میانی را انتخاب می کنید، یا کلیک-میانی بر روی لینک یک فایل یا آدرس وب در کنسول فایرباگ ، آن را در tab جدیدی باز می کند. برای انجام این کار بدون کلیک- میانی موس، می توانید کلید Ctrl را گرفته و برای گرفتن نتیجه مورد نظر بر روی آن کلیک کنید.

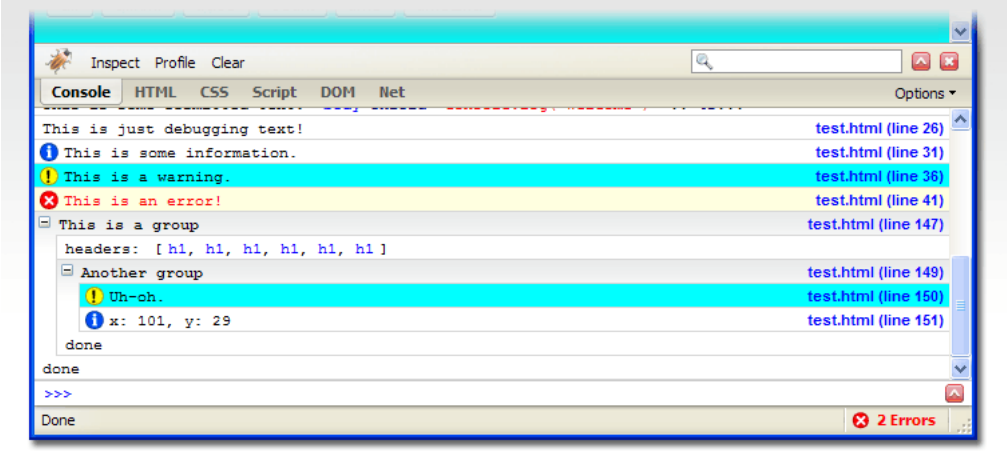
Inspect and command

پس از یافتن یک عنصر HTML با استفاده از ابزار Inspect، می خواهید از خط فرمان برای کار بر روی آن استفاده کنید. فایرباگ انجام این کار را آسان کرده است؛ تنها با استفاده از متغیر "$1" برای اشاره به آخرین عنصری که بررسی کرده اید استفاده کنید، یا از "$2" برای یکی قبل از آن استفاده کنید.

همچنین، می توانید از "$$1" و"$$2" برای اشاره به اشیایی که در نوار DOM انتخاب کرده اید استفاده کنید.

واقعه نگاری جاوااسکریپت (JavaScript Logging)

داشتن یک دیباگر جاوااسکریپت خیلی جالب است، اما بعضی مواقع روش سریع برای یافتن خطاها دیدن اطلاعات زیادی در کنسول می باشد. فایرباگ مجموعه ای قدرتمند برای واقعه نگاری توابع را به شما می دهد که می توانید آن ها را برای صفحات وب شخصی تان فراخوانی کنید.



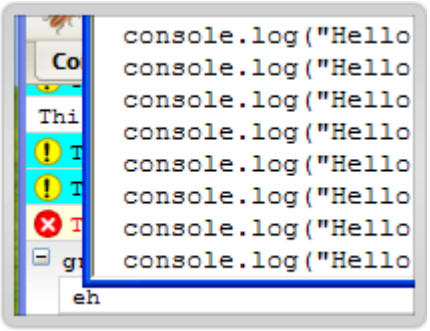
تابع console.log()

روش آسان برای نوشتن در کنسول فایرباگ مانند این می باشد:

```
console.log("hello world")
```

می توانید آرگومانهای زیادی را که می خواهید با هم کاری انجام دهند را وارد کنید، مانند:

```
console.log(2,4,6,8,"foo",bar).
```

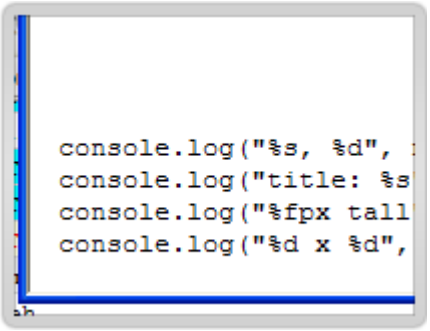


هایپرلینک اشیا

console.log و توابع مربوط به آن می توانند کاری بیش از نوشتن متن در کنسول انجام دهند. می توانید هر نوع شی را به console.log ارسال کنید و آن را به صورت هایپرلینک مشاهده کنید. عناصر، توابع و آرایه ها مانند شی خواهند بود، و شما باید نام آن ها را وارد کنید. با کلیک بر روی این لینک ها می توانید بر روی نواری که به آن ها اختصاص دارند به بررسی مقادیر آن ها بپردازید.

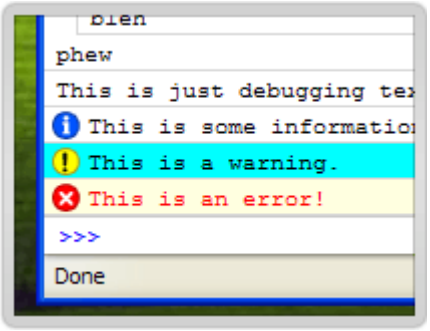
رشته فرمت

می توانید به سبک دستور printf زبان c دستور console.log را نیز فرمت بندی کنید. برای مثال، می توانید به صورت console.log("%s is %d years old.", "Bob", 42) بنویسید.



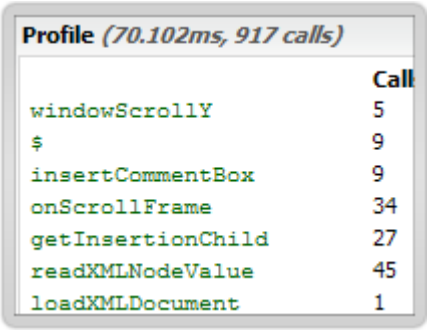
توابعی برای چاپ رنگی

علاوه بر console.log، چندین تابع دیگری وجود دارند که می توانید برای چاپ پیغام به صورت رنگی از آن ها استفاده کنید. این توابع شامل console.debug، console.info، console.warn و console.error می باشند.



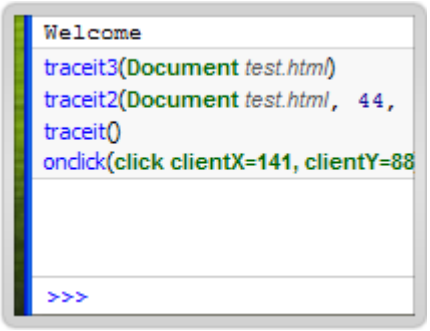
زمان اجرای کد

فایرباگ دو روش ساده برای سنجش بهینه گی کد جاوااسکریپت در اختیار شما قرار می دهد. روش کم هزینه اول فراخوانی console.time("timing foo") قبل از کدی که می خواهید مورد سنجش قرار دهید می باشد، و پس از پایان کد console.timeEnd("timing foo") را فراخوانی کنید. فایرباگ زمانی که در این بین صرف شده را گزارش خواهد داد. روش پر هزینه استفاده از plofiler جاوااسکریپت می باشد. تنها کافی است که console.profile() را قبل از کدی که می خواهید مورد سنجش قرار دهید، و console.profileEnd() در پایان آن فراخوانی کنید. فایرباگ گزارش مفصلی درباره زمان سپری شده برای همه توابع خواهد داد.



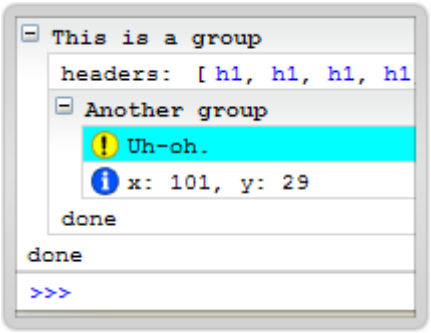
بررسی پشته

فایرباگ با فراخوانی console.trace() اطلاعات کاملی درباره stack trace در کنسول خواهد داد. نه تنها در مورد توابع موجود در پشته گزارش خواهد داد، بلکه در مورد آرگومان های که به توابع پاس داده شده اند نیز اطلاعاتی خواهد داد. برای بررسی توابع و یا اشیا می توانید بر روی آن ها کلیک کنید.



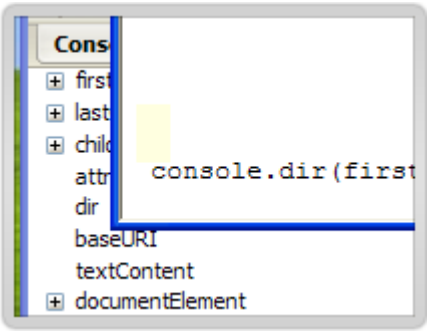
گروه بندی تودرتو

بعضی اوقات خواندن لیست مسطحی از پیغام ها سخت می باشد، فایرباگ راه حلی برای دندانه دار کردن پیغام ها در کنسول را برای شما ارائه می دهد. تنها با فراخوانی console.group("a title") در شروع یک بلوک دندانه دار جدید، و console.groupEnd() برای بستن آن به این هدف خواهید رسید. می توانید در سطوح مختلفی دندانه دار بودن را ایجاد کنید.



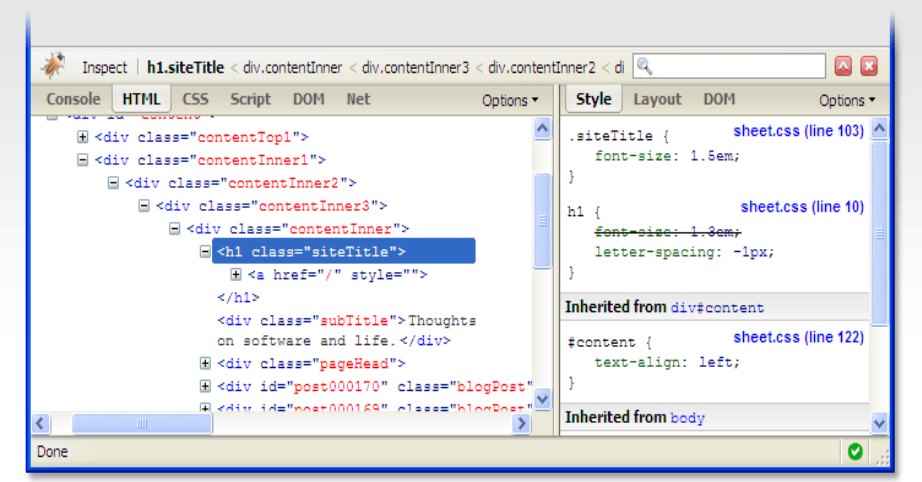
بررسی شی

چندین بار تابحال برای یک عنصر مجبور به نوشتن تمام خاصیت هایش شده اید؟ با فایرباگ نیازی به نوشتن تمام خاصیت ها نیست. با فراخوانی console.dir(object) لیستی از خاصیت های یک شی را خواهید داشت.



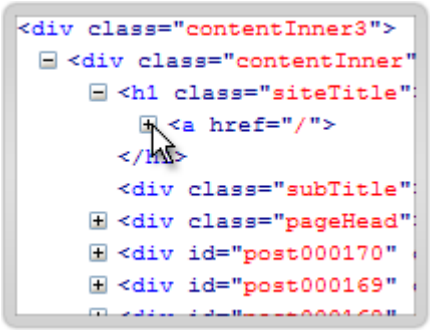
توسعه HTML

فایرباگ یافتن عناصر HTML که داخل صفحات پنهان شده اند را آسان کرده است. زمانیکه عنصری را که به دنبال آن می گردید پیدا کنید، فایرباگ اطلاعات کافی از آن را در اختیارتان قرار خواهد داد، و به شما اجازه می دهد که HTML را ویرایش کنید.



نمایش کد صفحه

Firefox داری پنجره "View Source" می باشد که در آن می توانید کد صفحه را ببینید، اما نحوه تغییر کد توسط جاوااسکریپت آن را به شما نشان نخواهد داد. نوار HTML موجود در فایرباگ چیزی را که اکنون در صفحه وبتان مشاهده می کنید را به شما نشان می دهد. بعلاوه، نوارهای سمت راست اجازه مشاهده خاصیتهای اختصاصی عنصر، شامل قوانین CSS که سبک عنصر را نشان می دهد، پیکسل ها که موقعیت و سایز عناصر را نشان می دهند، و خاصیت های DOM که توسط جاوااسکریپت می توانید به آنها دسترسی داشته باشید را می دهد.

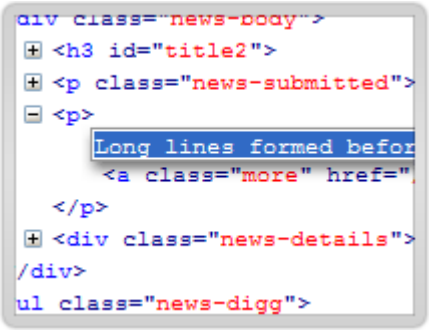


نمایش تغییرات صفحه

در هر وب سایتی که جاوااسکریپت در آن به کار رفته است، عناصر HTML ایجاد، حذف و ویرایش می شوند. جالب نخواهد بود اگر شما ندانید که واقعاً کی، چگونه، و کجا این تغییرات انجام می گیرند. فایرباگ به محض اینکه این تغییرات در HTML انجام می شوند با رنگ زرد آن ها را مشخص می کند.

ویرایش کد Html

فایرباگ روش جالبی را برای ایجاد تغییرات آزمایشی در HTML و دیدن فوری تاثیر آن ها در اختیارتان قرار می دهد. تنها با کلیک آن ها و رفتن از یک صفت به دیگری با استفاده از tab می توانید صفات و متن HTML را ایجاد، حذف و یا ویرایش کنید. تغییرات به همان صورتی که شما نوشته اید اعمال خواهند شد. اگر هدفتان انجام تغییراتی بیش از یک تغییر کوچک می باشد، فایرباگ اجازه ویرایش تمام کد HTML برای هر عنصری را می دهد. تنها کافی است که بر روی عنصر مورد نظر کلیک- راست کرده و از منو "Edit HTML..." را انتخاب کنید.

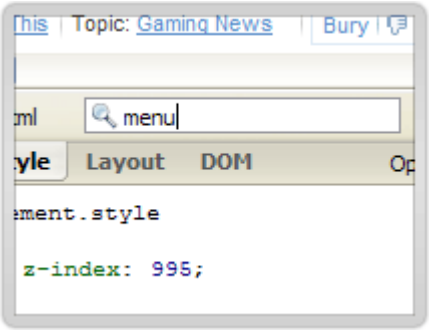


یافتن عناصر با استفاده از موس

برخی اشیا در صفحه وبتان زیاد درست به نظر نمی رسند و می خواهید دلیل آن را بدانید. راه سریعتری نسبت به کلیک بر روی دکمه "Inspect" در نوار ابزار فایرباگ و پس از آن آماده شدن برای جستجو در صفحه وب برای گرفتن جواب وجود دارد. اگر شما در اطراف صفحه وب حرکت کنید، هر چیزی که زیر موس شما باشد فوراً داخل Firebug معلوم می شود، و به شما HTML و CSS پشت آن را نشان می دهد.

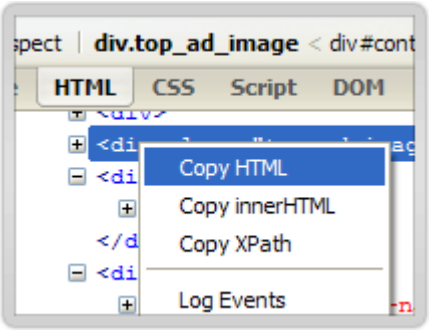
جستجو در کد

اغلب اوقات می دانید که دنبال چه چیزی می گردید، اما نمی توانید آن را در صفحه ببینید. با استفاده از جعبه جستجوی سریع در toolbar موجود در فایرباگ می توانید در کد HTML جستجو کرده و نتیجه را به صورت پررنگ ببینید.



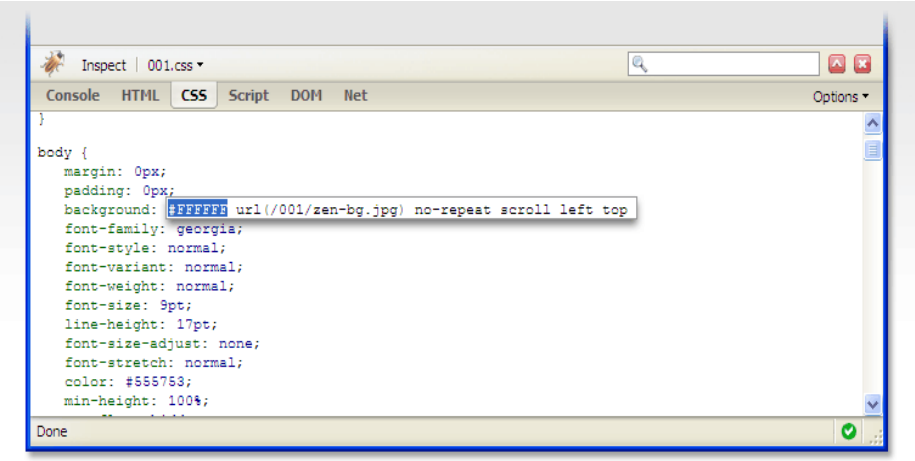
کپی کد Html

با کلیک راست بر روی هر عنصر، چندین گزینه برای کپی عنصر از جنبه های مختلفی خواهید داشت، مانند یک کد HTML، مقدار موجود در خاصیت innerHTML آن، و یا به صورت عبارت Xpath.



توسعه CSS

قسمت Firebug CSS هر چیزی که در مورد قالب وب سایتتان را بخواهید به شما می گوید، و اگر چیزی را دوست نداشته باشید، می توانید آن را تغییر داده و نتیجه کار را فوراً ببینید.

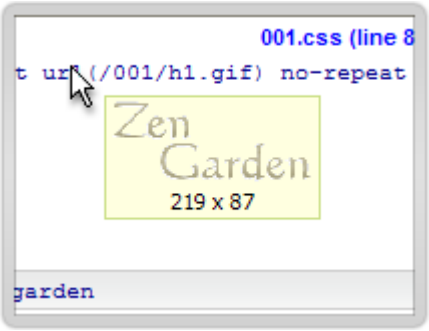


بررسی آبشاری (cascade)

بدون فایرباگ احتمالا زمانی که بایستی سرتیتر صفحه وب تان سیاه باشد و آن را آبی می بینید، بسیار عصبانی خواهید شد. اما با فایرباگ بررسی قسمت مشکل دار بسیار آسان می شود. فایرباگ قوانینی که به صورت آبشاری باهم بر قالب یک عنصر تاثیر می گذارند را نشان می دهد.این قوانین به ترتیب اولویت ذخیره شده اند. هر قانون یک لینک به فایلی که در آن قرار دارد را داراست که می توانید با کلیک کردن به آن خط فایل بروید.

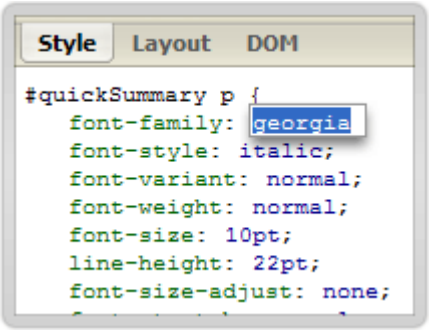
پیش نمایش رنگ و تصویر

اگر داخل CSSTab موس را روی رنگ و یا تصویر ببرید، یک tooltip کوچک پیش نمایشی از رنگ و یا تصویر را نشان می دهد. Tooltip تصویر ابعاد فایل تصویری را نیز نشان می دهد، مخصوصا وقتی که احتیاج به نوشتن CSS برای ایجاد یک عنصر دارید که حتما با سایز تصویر یکسان باشد.



ویرایش آسان کدهای CSS

نوشتن CSS کار ساده ای نبوده است. با کلیک روی هر خاصیت CSS یک textbox کوچک ظاهر می شود. با تایپ شما، تغییرات فورا اعمال می شود. همچنین در حین نوشتن فایرباگ مقادیر ممکن را به شما پیشنهاد خواهد داد (autocompletes). با کلید Esc می توانید تغییراتی را که ایجاد کرده اید را لغو کنید، یا با کلید Tab از خاصیتی به خاصیت دیگر بروید.



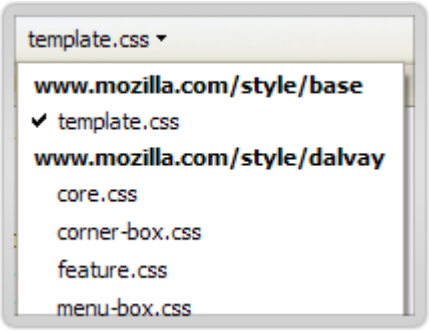
فعال و غیر فعال کردن خاصیت ها

اغلب راه حل یک مشکل را می توان با غیر فعال کردن چندین خاصیت CSS پیدا کرده و دید که بدون آن ها چگونه به نظر می رسند. با بردن موس روی هر خاصیت، یک آیکون مدور کوچک در سمت چپ آن خواهید دید. کلیک بر روی آن، خاصیت را غیر فعال می کند، و کلیک دوباره بر روی آن باعث فعال شدنش می شود.

```
    border: medium none;
}

#firefox .features h3 a {
    padding: 2px 0pt 6px 35px
    background: transparent u
    display: block;
    text-decoration: underlin
}

#firefox .features p {
```

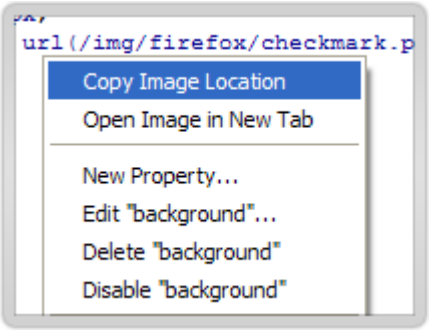
بررسی کل صفحه CSS

در حالی که نوار HTML به شما اجازه بازرسی (inspect) از CSS یک عنصر را می دهد، نوار CSS اصلی اجازه دیدن کل فایل CSS را می دهد. با کلیک بر روی لیست فایل ها درtoolbar همه CSSهای که به صفحه اضافه (imported) شده اند را خواهید دید.



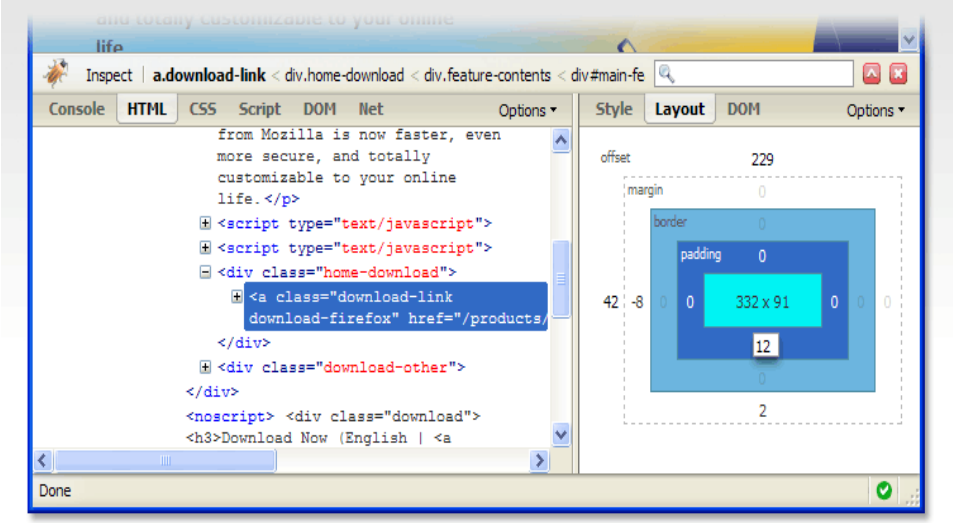
توسعه راحت برنامه وب

برای بسیاری از ما، Copy و Paste مهمترین قسمت هر توسعه است. فایرباگ، کپی دو عنصر مهم را آسان کرده است، URL تصویر و رنگ. فقط با کلیک راست منویی با امکانات clipboardی خواهید دید.



طرح بندی قالب (CSS Layout)

زمانی که جعبه CSS شما به درستی طرح بندی نشده باشد فهمیدن دلیل مشکل کمی سخت است. اجازه بدهید فایرباگ چشمان شما باشد و offset، margin، padding و size را برای شما توضیح دهد.

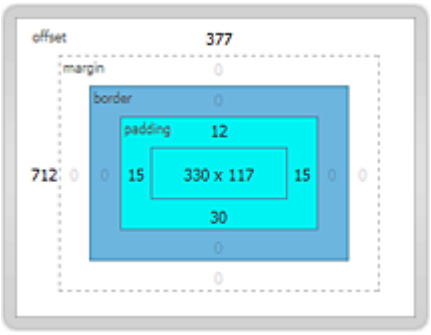


جعبه نمایشی CSS

با CSS، همه عناصر به صورت جعبه های تودرتو چیده می شوند. با حرکت موس بر روی یک عنصر HTML در هر یک از نوارهای فایرباگ، جعبه های عنصر را که با رنگ متفاوتی مشخص می شوند خواهید دید. روش سریعتر برای تمایز بصری بین Padding و marginها وجود ندارد.

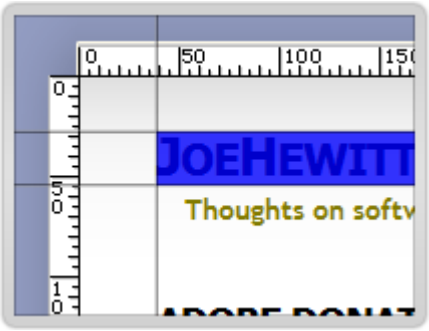
موقعیت لبه های جعبه ها

Layout یک مدل جعبه ی برای نمایش لبه های محیط های Padding و margin ارائه می کند بعلاوه، به شما طول و عرض جعبه داخلی تر را می دهد، و آفست x و y عنصر مورد نظر از عنصر پدر را می دهد.



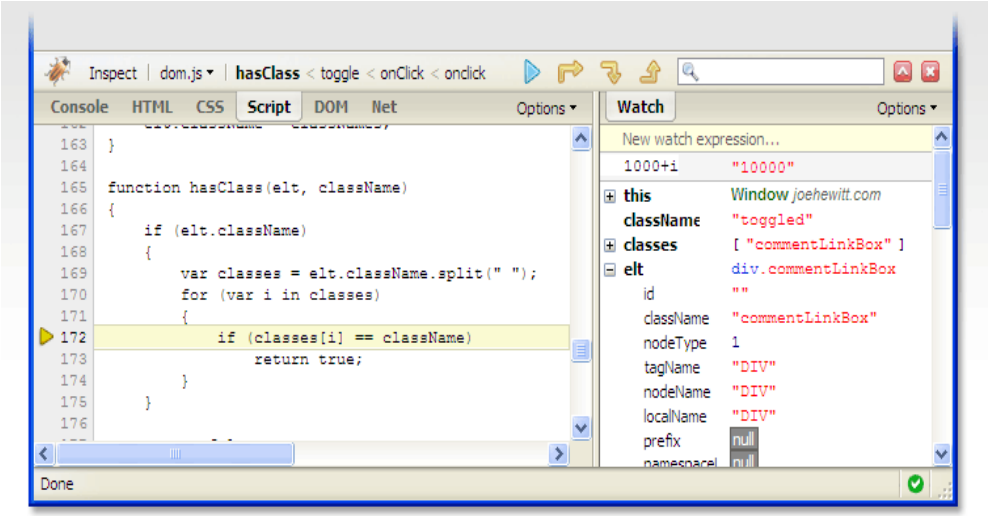
خط کش راهنما

موستان را در نوار Layout اطراف جعبه ها حرکت دهید، با انجام این کار، خط کش ها و راهنمایی ها در صفحه ظاهر می شوند. با این کار موقعیت عناصر را خواهید دید.



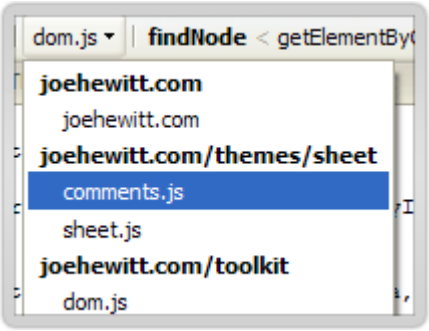
اشکال زدائی جاوااسکریپت

فایرباگ شامل یک دیباگر قوی جاوااسکریپت می باشد که به شما اجازه توقف اجرا را در هر زمان و تماشای اینکه هر متغیر چه مقداری دارد را می دهد.



یافتن آسان اسکریپتها

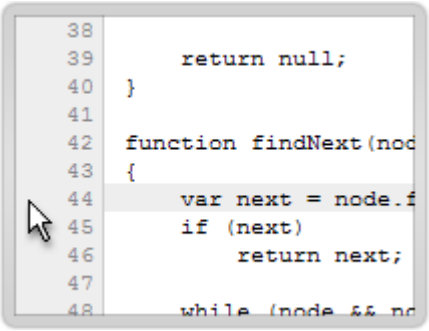
صفحات وب ترکیبی از چندین فایل می باشند، یافتن هر یک از آن ها برای دیباگ کار سختی می باشد. انتخابگر فایل اسکریپتی فایرباگ، فایل ها را داخل یک لیست مرتب و سازماندهی می کند که برای دسترسی سریع به هر یک از فایلها کمک می کند.



متوقف کردن اجرا در هر زمانی

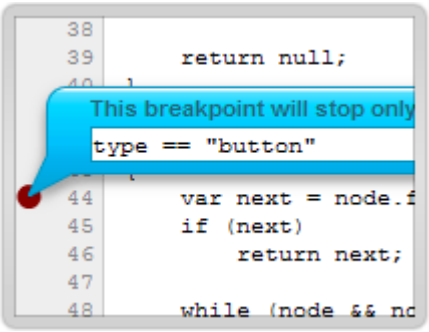
فایرباگ به شما اجازه می دهد نقاط توقف اجرا (breakpoints) را تنظیم کنید، هنگام رسیدن دیباگر به آن نقطه خاص اجرای کد، متوقف می شود. تا زمانی که اجرا متوقف می باشد، می توانید مقدار هر یک از متغیر ها را مشاهده کرده و تا زمانی که اشیا بدون تغییر می باشند می توان آن ها را بررسی کرد. برای تنظیم نقاط توقف

اجرا، فقط بر روی شماره خط کلیک کنید، یک نقطه قرمز برای نمایش اینکه نقطه توقف تنظیم شده ظاهر می شود. برای حذف نقطه توقف دوباره بر روی نقطه قرمز کلیک کنید.



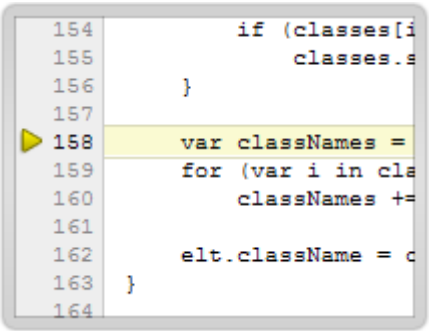
توقف شرطی اجرا

اگر نقاط توقف را هر بار ببینید، آزار دهنده خواهند بود. بعضی اوقات شما در حالت خاصی می خواهید که توقف داشته باشید. فایرباگ به شما اجازه می دهد که برای نقاط توقف شرط قرار دهید، که بایستی عبارتی را برای توقف چک کنند. برای تنظیم شرط نقاط توقف تنها کافی است که بر روی هر شماره خط راست- کلیک کنید. حبابی ظاهر خواهد شد که می توانید داخل آن عبارت جاوااسکریپتتان را وارد کنید. برای تغییر عبارت شرطی می توانید دوباره راست-کلیک کنید، و یا برای حذف عبارات از کلیک چپ استفاده کنید.



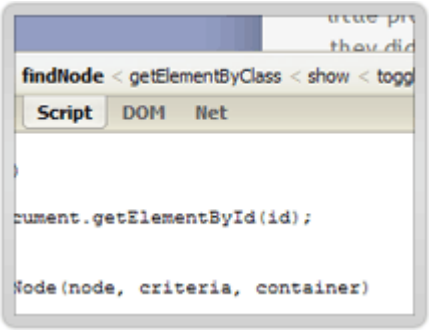
یک گام در یک زمان (trace)

زمانی که دیباگر توقف می کند، می توانید اجرا را به صورت یک خط در یک زمان ادامه دهید. این موضوع به شما اجازه می دهد که از وضعیت متغیرها و اشیا، زمانی که خط خاصی اجرا می شود اطلاع داشته باشید. همچنین می توانید اجرای گام به گام بیش از یک خط را انتخاب کنید. از منوی context، گزینه "Run to this Line" برای ادامه اجرای تا زمانی که به خط مورد نظر شما برسد را انتخاب کنید.



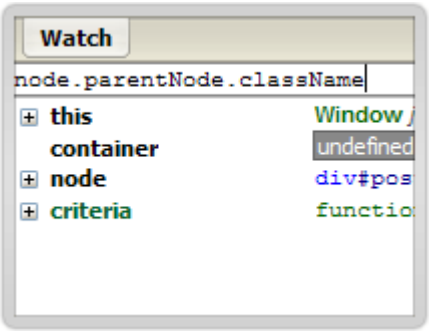
پشته زمان اجرا

زمانی که دیباگر متوقف است، فایرباگ به شما پشته فراخوانی را نشان می هد، که مجموعه ای از توابع تودرتو فراخوانی شده که اکنون در حال اجرا و منتظر اتمام می باشند را نشان می دهد. پشته فراخوانی به صورت ترکیبی از دکمه ها که هر یک نام یک تابع در پشته را بر خود دارند در toolbar نشان داده می شود. می توانید برای رفتن به محلی که تابعی در آن جا توقف کرده بر روی دکمه با نام آن تابع کلیک کنید، و توابع محلی آن تابع را ببینید.



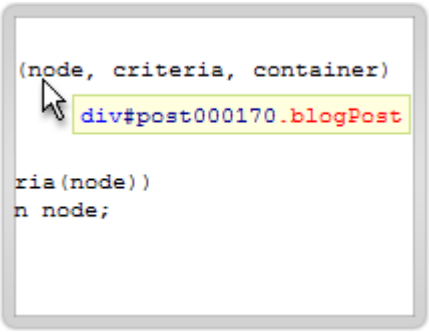
پنجره Watch

در طول دیباگ، می خواهید مقدار عبارات پیچیده و یا اشیائی که داخل DOM می باشند را ببینید. فایرباگ به شما اجازه می دهد که عبارات جاوااسکریپتی را که هنگام دیباگ مقادیرشان بروز می شود را بنویسید. هنگام نوشتن این عبارات، فراموش نکنید که می توانید از کلید tab برای انتخاب خاصیت شی استفاده کنید.



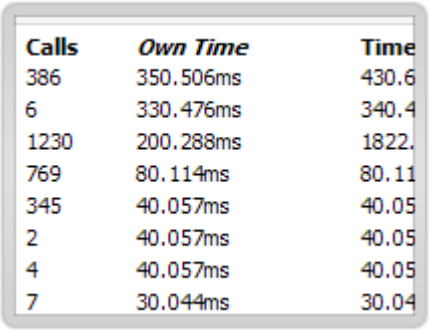
Tooltip متغیرها

زمانی که دیباگر متوقف می باشد، می توانید موس را بر روی هر یک از متغیرها تابع در حال اجرا برده تا مقدار آن ها را به صورت tooltip ببینید.



کارآیی کد های جاوااسکریپت

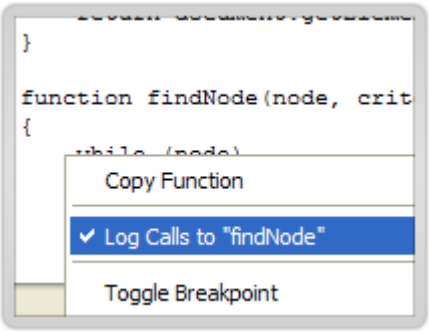
با استفاده از profiler برنامه فایرباگ می توانید کدهای کند را از کدهای سریع تشخیص دهید. برای استفاده از profiler، بایستی به نوار Console رفته و بر روی کلید profiler کلیک کنید. سپس برنامه خود را اجرا کنید و یا صفحه را دوباره لود کنید و سپس دوباره بر روی کلید profiler کلیک کنید. پس از آن گزارش مفصلی از توابع فراخوانی شده و مدت زمان صرف شده برای اجرایشان را خواهید دید.



Calls	Own Time	Time
386	350.506ms	430.6
6	330.476ms	340.4
1230	200.288ms	1822.
769	80.114ms	80.11
345	40.057ms	40.05
2	40.057ms	40.05
4	40.057ms	40.05
7	30.044ms	30.04

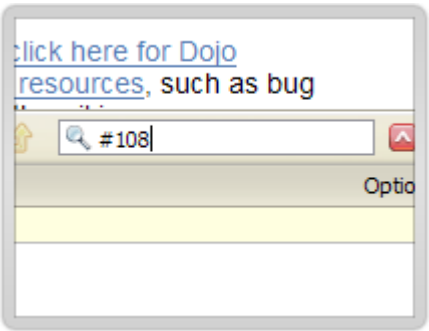
بررسی فراخوانی های یک تابع

برخی مواقع تابع مشکل داری فراخوانی می شود که اغلب اوقات نمی توانید برای حل مشکل به دیباگر بروید. شما تنها می خواهید بدانید که این توابع چه زمانی فراخوانی شده و کدام پارامترها به آن ها پاس داده می شوند. برای پیگیری همه ی فراخوانی های یک تابع، به نوار Script رفته و "Log calls to 'function name'" را انتخاب کنید. سپس به نوار Console رفته و جریان فراخوانی های تابع را مشاهده کنید.



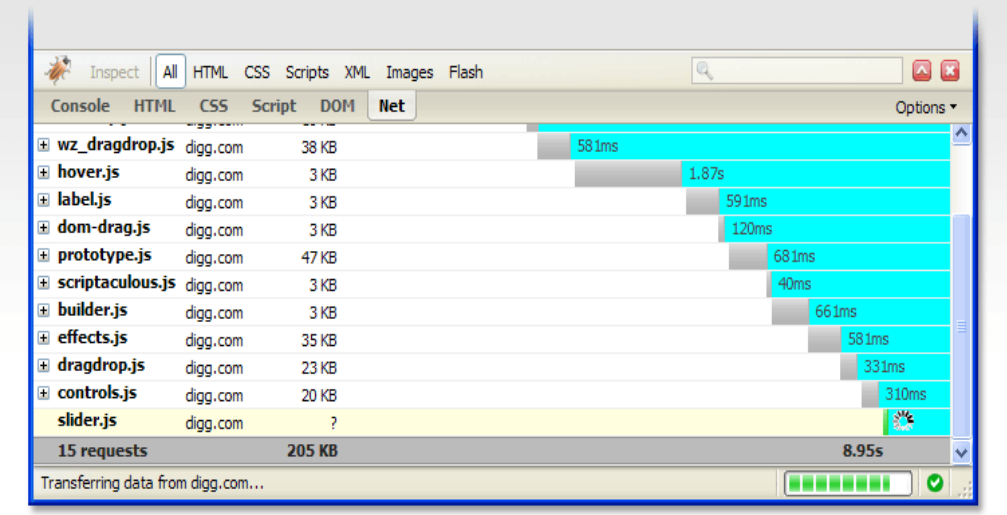
مراجعه مستقیم به خط مشخص

گاهی اوقات می خواهید به صورت مستقیم به خط خاصی از اسکریپت تان بروید. انجام این کار زیاد آسان نخواهد بود. برای این کار تنها شماره خط که قبل از آن علامت # قرار دارد را در جعبه جستجوی سریع وارد کنید.



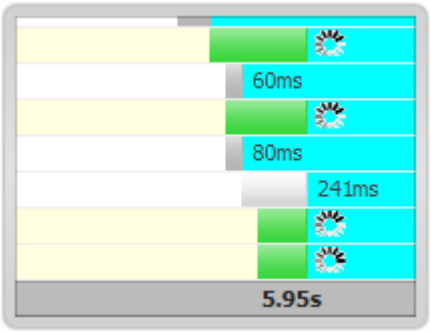
نظارت بر شبکه (Network Monitoring)

بعضی از صفحات زمان زیادی برای لود شدن احتیاج دارند، اما به چه دلیل؟ آیا کدهای جاوااسکریپت زیادی نوشته اید؟ آیا فراموش کرده اید که تصاویرتان را فشرده کنید؟ آیا مشکل از سرور شما می باشد؟ فایرباگ یک به یک فایل ها را برایتان بررسی می کند.



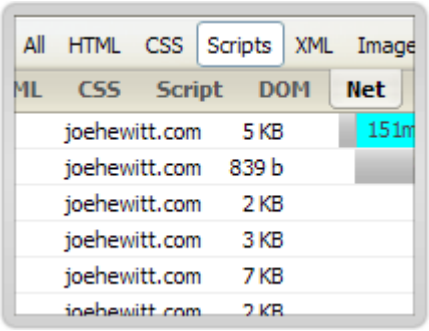
مشاهده زمان بار شدن صفحات

هر فایلی در بخش Net داری یک میله می باشد که نشان دهنده مدت زمان شروع و پایان لود شدن آن فایل نسبت به فایل های دیگر می باشد. برای مثال، شما می دانید که فایل های جاوااسکریپتی تنها یک بار لود می شوند. این قسمت به شما کمک می کند تا ترتیب لود شدن فایل ها در صفحه را مشاهده نماید.



تقسیم بندی براساس نوع

برخی اوقات درباره بعضی از انواع فایل ها نگرانی دارید، مانند جاوااسکریپت و یا تصاویر. برای فیلتر کردن نوع مورد نظرتان در نوار ابزار Net بر روی آن نوع کلیک کنید. این مورد روش مناسبی برای پی بردن به زمان دانلود برای هر کدام از انواع می باشد.



حافظه کش (cache) سایت

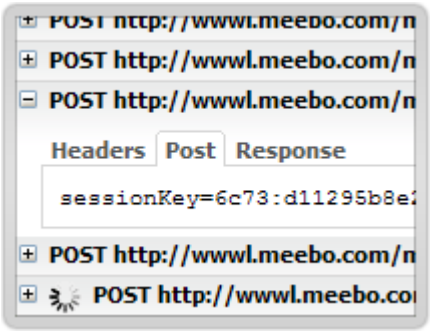
همه درخواست های شبکه ای یکسان نمی باشند- برخی از آنها به جای شبکه از قسمت کش مرورگر لود می شوند. فایرباگ درخواستهای را که از کش لود می شوند را خاکستری روشن نمایش می دهد بنابراین به راحتی می توانید تاثیر کش را بر روی سایتتان ببینید.

مشاهده هدر های HTTP

هدر HTTP شامل اطلاعات جالبی مانند نوع mime فایل، نوع وب سرور، کش مستقیم، کوکی ها و موارد دیگر می باشد. برای دیدن هدر HTTP بر روی پیکان سمت چپ هر درخواست کلیک کنید تا نمایش داده شوند.

نظارت بر شی XMLHttpRequest

پدیده Ajax حول محور شی کوچکی به نام XMLHttpRequest می چرخد. جالب نخواهد بود اگر این درخواست را ارسال کرده و هیچ بازخورد بصری از آن دریافت نشود. فایرباگ همه XMLHttpRequest را در نوارهای Console و Net به همراه نوشته ای که ارسال شده و نوشته ای که دریافت شده به شما نشان می دهد.



دیباگ هنر تشخیص و برطرف سازی مشکلات کد برنامه تان می باشد. همه برنامه نویسان بعضی اوقات در حین توسعه برنامه احتیاج به یک درجه از دیباگ برنامه هایشان خواهند داشت.

Asp.Net به تکنولوژی سمت سرور مسلح بوده و به صورت وسیع از دیباگ برنامه های ASP.Net حمایت می کند. برنامه های Ajaxی جنبه های جدیدی را مطرح کردند که دیباگ برنامه ها را بسیار پیچیده کرده است. استفاده وسیع از جاوااسکریپت و این حقیقت که داده سفارشی با استفاده از انتقال غیر همزمان ارسال می شود به این معنی خواهد بود که با چالش جدیدی هنگام دیباگ برنامه های Ajaxی در ASP.NET روبرو خواهید شد.

در این بخش به روش دیباگ جاوااسکریپت از طریق ASP.NET سمت سرور خواهیم پرداخت.

دیباگ سمت سرور

ASP.Net محیط توسعه سمت سرور می باشد، موتور runtime در ASP.NET به صورت مجازی همه صفحات و کدهای اسمبلی .NET را تجزیه و کامپایل می کند.

زمانیکه صفحه وب ASP.NET درخواست می شود (مانند www.SomeSite.com/SomePage.aspx)، موتور runtime موجود در ASP.NET صفحه وب و کدهای که با آن در ارتباط می باشند را تجزیه می کند. این کدها معمولا در یک فایل کد در دایرکتوری App_Code وب سایت قرار دارند، یا اینکه می توانند در خود صفحه وب (ASPX) جاگذاری شوند. صفحه وب و کد ها در یک اسمبلی .NET کامپایل شده و برای اجرا داخل کش (cache) اسمبلی لود می شوند.

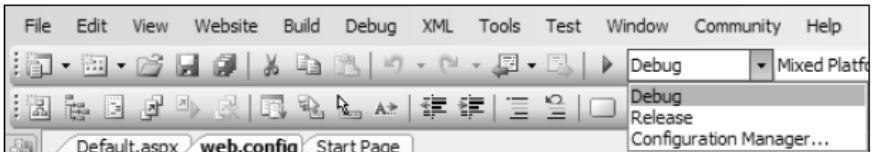
فعال سازی برا ی پشتیبانی از دیباگ

قبل از اینکه بتوان از دیباگ استفاده کرد پشتیبانی از دیباگ نیاز به فعال سازی دارد. در برنامه های وب ASP.NET، برای رسیدن به این منظور بایستی به صورت زیر در فایل پیکربندی برنامه وب (web.config) تنظیمات زیر را اعمال کرد:

```
<configuration>
     <system.web>
          <compilation debug="true">
          </compilation>
     </system.web>
</configuration>
```

در صورتیکه بخواهید برنامه وب را با استفاده از Visual Studio .NET 2005 در حالت دیباگ اجرا کنید، و پیکربندی <compilation debug="true"> تنظیم نشده باشد، شما باید آن را فعال با مقدار true تنظیم کنید:

برای برنامه های دیگر مانند کتابخانه های کلاسی، بایستی مانند شکل زیر در فایل پیکربندی debug انتخاب شود.



در هریک از این حالات، زمانی که فایل های برنامه کامپایل می شوند، اطلاعات خاص دیباگ تولید می شوند که دیباگر Visual Studio.NET را برای ردیابی فعال ساخته و دقیقا نشان می دهند که کدام خط در حال اجرا می باشد. می توانید این موضوع را با نگاه به دایرکتوری خروجی برنامه ای که کامپایل می کنید ببینید. اگر debug build انتخاب شده باشد، یا دیباگ از طریق web.config همان طور که قبلا گفته شد تنظیم شده باشد، آن جا فایل های مربوط به دیباگ ظاهر خواهد شد، که پسوند فایلی pdb. را دارند. برای مثال، اگر برنامه شما MyApp.exe را تولید کند، یک فایل متناظر با آن به نام MyApp.pdb ایجاد خواهد شد.

تنظیم نقاط توقف اجرا

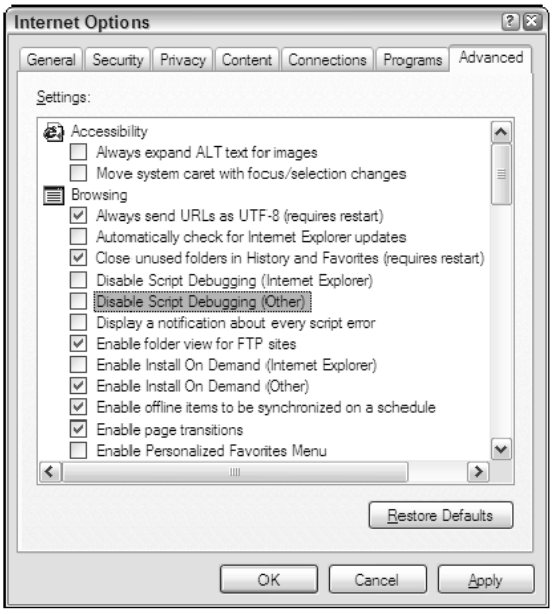
حال که دیباگ فعال شده است، اجرای برنامه ها در سرور می توانند ردیابی(track) شوند. آسان ترین روش برای این کار تنظیم نقاط توقف اجرا می باشد. نقاط توقف اجرا یک خط نشاندار در کد می باشد که به دیباگر اعلان توقف اجرا در خطی که با نقطه توقف اجرا نشاندار شده است را می دهد. یک نقطه توقف اجرا که با یک نقطه قرمز در سمت چپ خط نشاندار شده است، با ورود به خط آن نیز به صورت قرمز رنگ در خواهد آمد. شکل زیر یک نقط توقف اجرا را نشان می دهد.

```
namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("My application is about to start...");
            InitialiseApp();
            Console.WriteLine("Application Initialised successfully.");
            PerformWork();
            Console.WriteLine("Application is terminating.");

        }
```

دیباگ اسکریپت در Visual Studio

تهیه یک محیط دیباگ قدرتمند برای جاوااسکریپت، مانند محیط سمت سرور، در ویژوال استودیو ممکن می باشد. اما این امر نیازمند برخی تعاملات بین مرورگر و ویژوال استودیو می باشد؛ به همین خاطر مقداری پیکربندی سریع برای فعال سازی این عملیات که به صورت پیش فرض غیر فعال می باشد لازم است.

IE احتیاج به پیکربندی برای اجازه به دیباگ را دارد. به صورت پیش فرض عمل دیباگ در IE غیر فعال می باشد. برای فعال سازی این خصوصیت در IE گزینه Tools ⇦ Internet Options... را انتخاب کنید. یک جعبه گفتگو با تعدادی نوار انتخابی ظاهر می شود. انتخاب نوار Advanced موجب نمایش تعدادی گزینه خواهد شد.

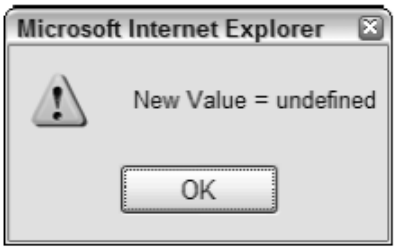


مطمین باشید که هر دو گزینه Disable Script Debugging(Internrt Explorer) و Disable Script Debugging(Other) انتخاب نشده باشند. البته برای اینکه دیباگ کار کند تنها لازم است که Disable Script Debugging(Internrt Explorer) انتخاب نشده باشد. انتخاب نکردن Disable Script Debugging(Other) بدین معنی می باشد که در ابزارهای به غیر از IE مانند Microsoft Outlook دیباگ نیز امکان پذیر می باشد. عملیات بالا همه ی پیکربندی مورد نیاز برای فعال سازی دیباگ اسکریپت می باشد.

برای امتحان این مورد یک پروژه جدید در ویژوال استودیو ایجاد کنید. یک فرم جدید ASPX ایجاد کنید و یا صفحه Default.aspx، داخل پروژه را ویرایش کنید. اعلام <html></html> موجود در صفحه و محتویات آن را پاک کرده و کد زیر را جایگزین آن کنید:

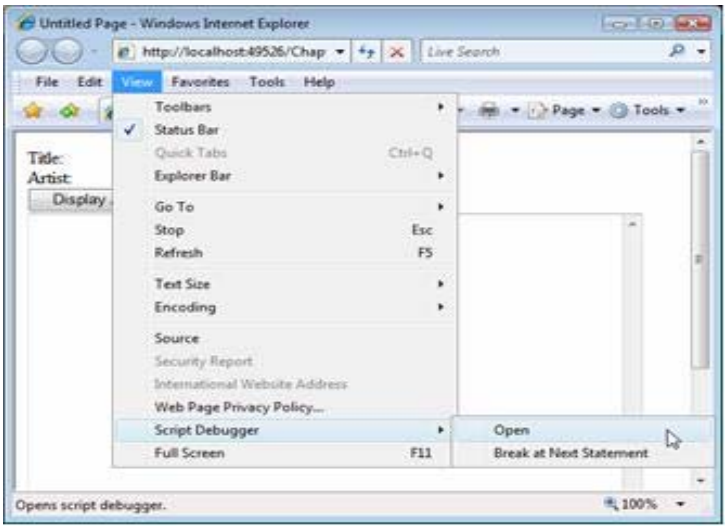
```
<html xmlns="http://www.w3.org/1999/xhtml" >
    <head runat="server">
        <title>Test Script Debugging</title>

        <script type="text/javascript">

            function DoSomeWork()
            {
                var cntrl =
                    document.getElementById("txtInput");
                var number = cntrl.value;
                number++;
                var newValue = DoSomeMoreWork(number);
                alert("New Value = " + newValue);
            }

            function DoSomeMoreWork(arg)
            {
                // Do some calculations
                arg = arg * 2 + 32;
                arg = arg + 18 / 2;
            }
        </script>
    </head>

    <body>
        <form id="form1" runat="server">
            <div>
                <input type="text"id="txtInput" value="test"/>
                <input type="button" value="Submit value"
                    onclick="DoSomeWork();" />
            </div>
        </form>
    </body>
</html>
```

مطمئن باشید که صفحه ی جدیدی که ایجاد کرده اید به عنوان صفحه شروع تنظیم شده است. (با کلیک راست در ویژوال استودیو قسمت Solution Explorer بر روی صفحه و انتخاب گزینه Set as Start page). برای اجرای برنامه در حالت دیباگ بر روی کلید شروع کلیک کنید و یا F5 را در صفحه کلید بزنید. (مطمئن باشید که web.config برای دیباگ تنظیم شده است.) کلیک برروی کلید Submit Valure بایستی خطایی مانند شکل زیر را تولید کند.

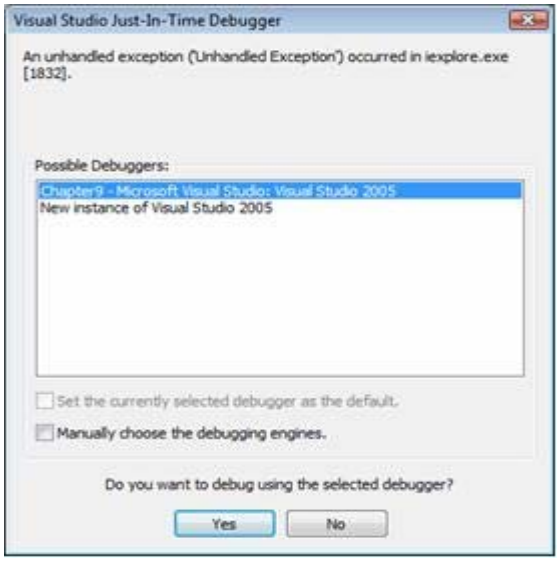


این نتیجه آن چیزی نیست که شما می خواستید، و احتمالا شما انتظار چیزی بیش از undefined داشتید. برای اصلاح این موضوع، شما بایستی به دیباگ در سمت کلاینت برای این برنامه بپردازید. برای این کار باید از قابلیت دیباگ اسکریپت موجود در IE استفاده کنید.

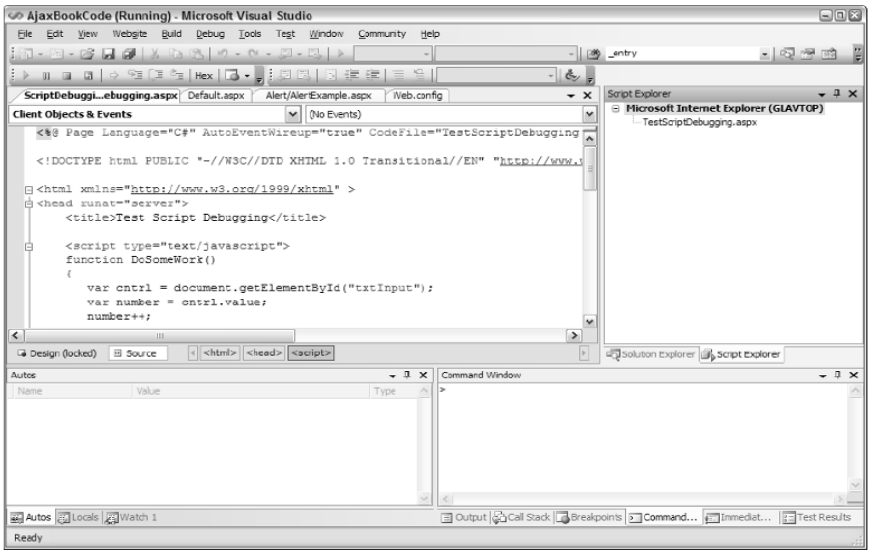
۱- برای ایجاد یک جلسه دیباگ برای یک صفحه ASP.NET، به صفحه ی برنامه مورد نظر رفته و طبق شکل زیر گزینه View ⇦ Script Debugger را در IE انتخاب کنید.



در اینجا با دو گزینه روبرو می شوید اگر Open را انتخاب کنید، فوراً شروع به استفاده از Visual Studio.NET به منظور دیباگ خواهید کرد همانطور که در شکل زیر می بینید. اگر Visual Studio.NET باز شود می توانید شروع به دیباگ کنید یا اینکه یک نمونه جدید از Visual Studio.NET را برای شروع دیباگ انتخاب کنید. با انتخاب Break at Next Statement، در حین اجرای اسکریپت صفحه هیچ اتفاقی رخ نخواهد داد.

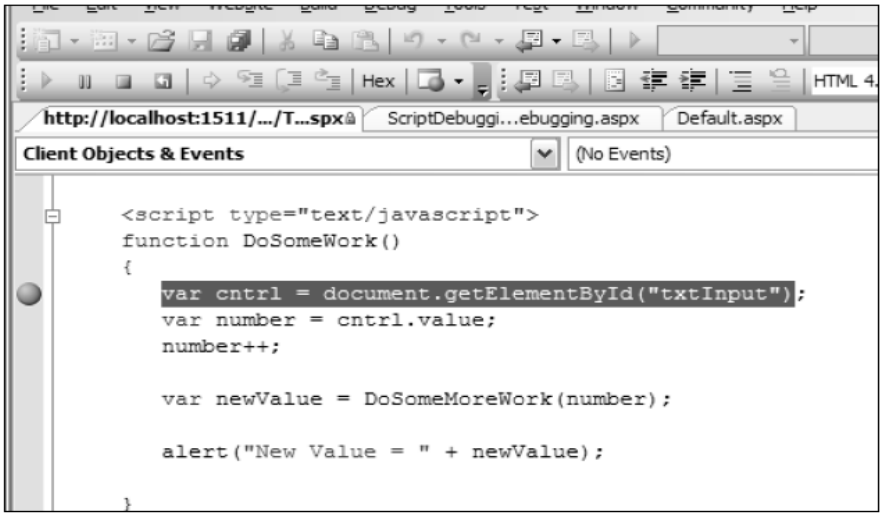


۲- در ویژوال استودیو پنجره Scrit Explorer نمایش داده می شود. اگر طرح بندی پنجره شما به صورت تنظیمات پیش فرض باشد، در این صورت پنجره Scrit Explorer بایستی در سمت راست صفحه نمایش داده شود همانند شکل زیر.

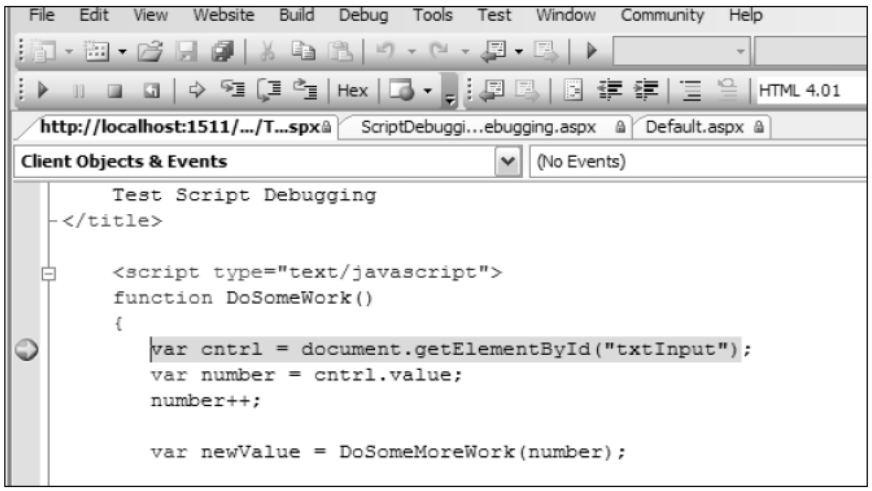


۳- در پنجره Scrit Explorer، اگر به تنها صفحه ASPX موجود در لیست توجه کنید خواهید دید که صفحه ی در حال اجرای فعلی می باشد. با انتخاب صفحه مورد نظر کد صفحه را خواهید دید. می بینید که بسیار شبیه به صفحه زمان طراحی و توسعه می باشد. بزرگترین تفاوت این است که اکنون می توانید نقاط توقف اجرا را در آن تنظیم کنید و مقدار متغیرهای موجود را به روش کد های سمت سرور امتحان کنید.

۴- در داخل پنجره ویرایش، به ابتدای تابع DoSomeWork رفته و کلید F9 را فشار دهید و یا بر روی نوار خاکستری در سمت چپ کلیک کنید تا در آن خط یک نقطه توقف اجرا ایجاد شود. پنجره ویرایش بایستی مانند شکل زیر باشد:



۵- حال به نمونه IE در حال اجرا، رفته و بر روی کلید Submit Value کلیک کنید. ویژوال استودیو بایستی به صورت خودکار فعال شده و زمانی که به نقطه توقف اجرا رسید اجرای برنامه را متوقف کند. این خط همانند شکل زیر به صورت رنگی مشخص خواهد شد.



۶- برای رفتن به خط بعدی کلید F10 را انتخاب کنید. حالا خط بعدی رنگی خواهد شد. با قرار دادن اشاره گر موس بر روی متغیر cntrl که در خطی که در حال خوانده شدن است:

```
var cntrl = document.getElementById("txtInput");
```

یک جعبه محاوره کوچک نمایش داده خواهد شد، این جعبه، متغیر cntrl را با یک گزینه برای بسط مقدار متغیر بوسیله کلیک بر روی نماد + نشان می دهد. (شکل زیر را مشاهده کنید)

کلیک بر روی نماد + به توسعه دهنده اجازه می دهد تا مقدار متغیر را بررسی کرده و خصوصیات آن را در وضعیت فعلی تماشا کند. این روش بررسی متغیرها، بسیار شبیه دیباگ کدهای سمت سرور می باشد.

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head><title>
    Test Script Debugging
</title>

    <script type="text/javascript">
    function DoSomeWork()
    {
        var cntrl = document.getElementById("txtInput");
        var num cntrl|{...} value;
        number++;

        var newValue = DoSomeMoreWork(number);

        alert("New Value = " + newValue);
```

با کلیک راست بر روی متغیر cntrl، منویی ظاهر می شود که شبیه به منوی ارائه شده در کدهای سمت سرور می باشد. مکانیسم دیباگ های سنتی مانند Add watch برای اضافه کردن متغیر cntrl به پنجره Watch، در قسمت پایین صفحه در دسترس می باشد. این موضوع را در ادامه فصل بیشتر توضیح خواهیم داد. همینطور، توسعه دهنده می تواند پنجره Quick watch را باز کند که اجازه باز کردن یک جعبه محاوره که برای بررسی آسانتر محتویات متغیرها تهیه شده است را می دهد. این پنجره شبیه به پنجره watch می باشد اما یک روش مستقیم و برتری برای کار بر روی متغیر خاصی را می دهد. تنها هدف این پنجره نمایش محتویات متغیر انتخاب شده می باشد وگرنه عملیات آن همانند پنجره Watch می باشد. تنها تفاوت واقعی آماده سازی کلید Reevaluate است که اجازه وارد کردن متغیر جدید داخل فیلد text را می دهد.

۱- دوباره کلید F10 را برای پیشروی در کد به خط بعدی انتخاب کنید. اجرای برنامه بایستی در خطی با محتویات زیر توقف کرده باشد:

```
number++;
```

۲- موس را بر روی متغیر number قرار دهید. دیباگر بایستی همانند شکل زیر مقدار متغیر number را نشان دهد. معمولاً مقداری که انتظار می رود در متغیر number باشد برابر با محتوای کنترل textbox موجود در صفحه می باشد.

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head><title>
    Test Script Debugging
</title>

    <script type="text/javascript">
    function DoSomeWork()
    {
        var cntrl = document.getElementById("txtInput");
        var number = cntrl.value;
        number++;
            number|"test"
        var newValue = DoSomeMoreWork(number);
```

۳- دوباره کلید F10 را برای رفتن اجرای برنامه به خط بعدی انتخاب کنید. اکنون دیباگر بایستی در خط زیر متوقف شده باشد:

```
var newValue = DoSomeMoreWork(number);
```

برای نمایش مقدار فعلی، موس را بر روی متغییر number ببرید. احتمالا چیزی شبیه شکل زیر نمایش داده شود:

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head><title>
    Test Script Debugging
</title>

    <script type="text/javascript">
    function DoSomeWork()
    {
        var cntrl = document.getElementById("txtInput")
        var number = cntrl.value;
        number++;

        var newValue = DoSomeMoreWork(number);
                 newValue | undefined
        alert("New Value = " + newValue);
```

اکنون می توانید ببینید که مقدار کدام متغیر نامعتبر می باشد. معلوم می شود که مقدار number یک مقدار متنی با مقدار test بوده است. مرحله بعدی اجرا انجام عملیات ریاضی بر روی مقدار رشته ای می باشد، که در حقیقت نامعتبر است. جاوااسکریپت همانند زبانهای VB و #C نیست. این عملیات ظاهراً نامعتبر، هیچ استثنایی را ایجاد نمی کند، اما مقدار متغیر number با undefined مقدار دهی می شود، که دقیقاً همان چیزی می باشد که شما در خروجی مرورگرتان مشاهده می کنید.

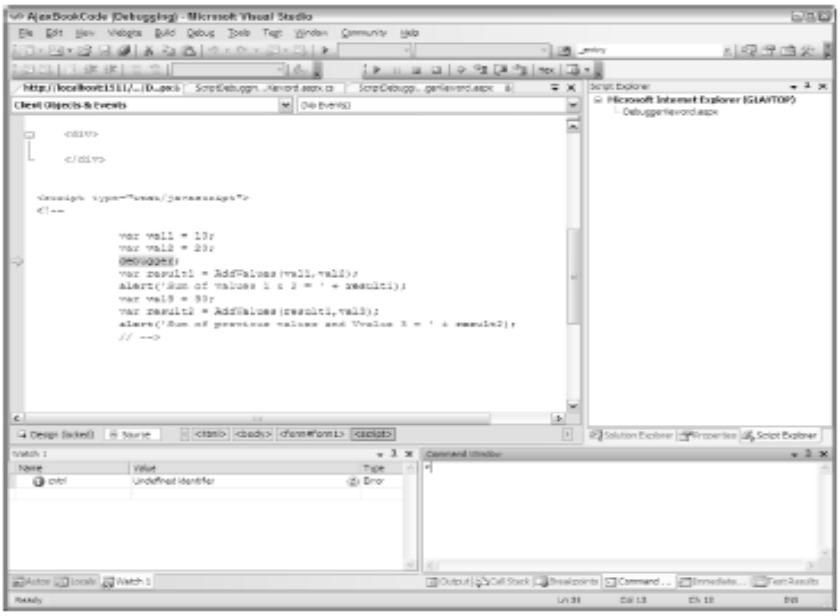
۴- کلید F5 را انتخاب کنید یا اینکه دکمه Play را برای اجرای معمولی برنامه کلیک کنید، همان نتیجه ای را که قبلاً دیدید ظاهر خواهد شد.

شما با موفقیت دیباگ را به اتمام رساندید و متوجه چرایی رفتار مرورگر شدید. دیباگ به این روش بسیار مفید می باشد و برنامه ها می توانند با جزئیات بیشتری مورد بررسی قرار گیرند.

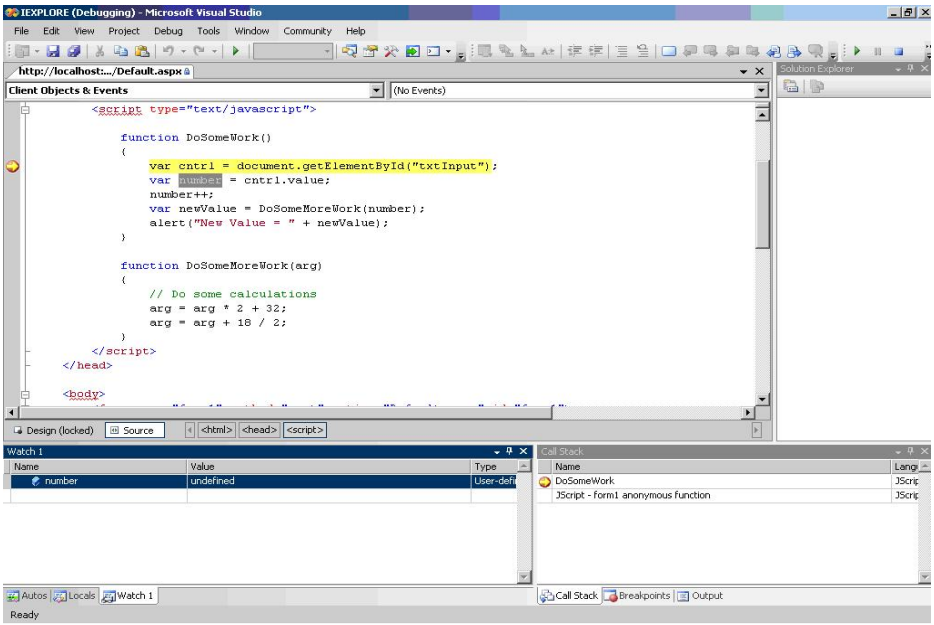
استفاده از پنجره Watch

همان طوری که قبلا اشاره شد، در زمان دیباگ، توسعه دهنده می تواند به راحتی موس را بر روی متغیر قرار داده و مقدار آن را مشاهده کند. اما انجام این کار برای برخی از متغیرها و ثابتها در هر خط که کد اجرا می شود می تواند سنگین باشد. اما شما گزینه های دیگری نیز دارید.

همانند تکنیک دیباگ سمت سرور، می توانید از Watch برای مشاهده مقدار متغیرها و فعل و انفعال های آن ها با برنامه تان استفاده کنید.



با استفاده از مثال قبلی، در حالت دیباگ بر روی number کلیک راست کنید. با انجام این کار منویی ظاهر می شود. از منو گزینه Add Watch را انتخاب کنید که دراینصورت متغیر number را به پنجره watch اضافه می کند. پنجره watch به صورت پیش فرض درپایین، قسمت چپ محیط ویژوال استودیو قرار دارد. که در شکل زیر می توانید مشاهده کنید.



مشاهده می کنید که متغیر number مقدار undefined را نشان می دهد. با دوبار فشردن F10 مقدار متغیر number تغییر پیدا می کند و متغیر number داخل پنجره Watch نیز به روز می شود. چندین آیتم می توانند به پنجره watch اضافه گردند که توسعه دهنده را قادر به پیگیری مقادیر متغیرها می کند.

استفاده از پنجره Command

توسعه دهنده ممکن است بخواهد وضعیت قسمتی از برنامه اش که در حال اجرا می باشد را ارزیابی کند. برای انجام این کار، توسعه دهنده می تواند از پنجره Command استفاده کند. برای انجام این کار، در حالت دیباگ، منوی View ⇦ Other Windows ⇦ Command Window را انتخاب کنید. (و یا Ctrl+Alt+A را انتخاب کنید.)

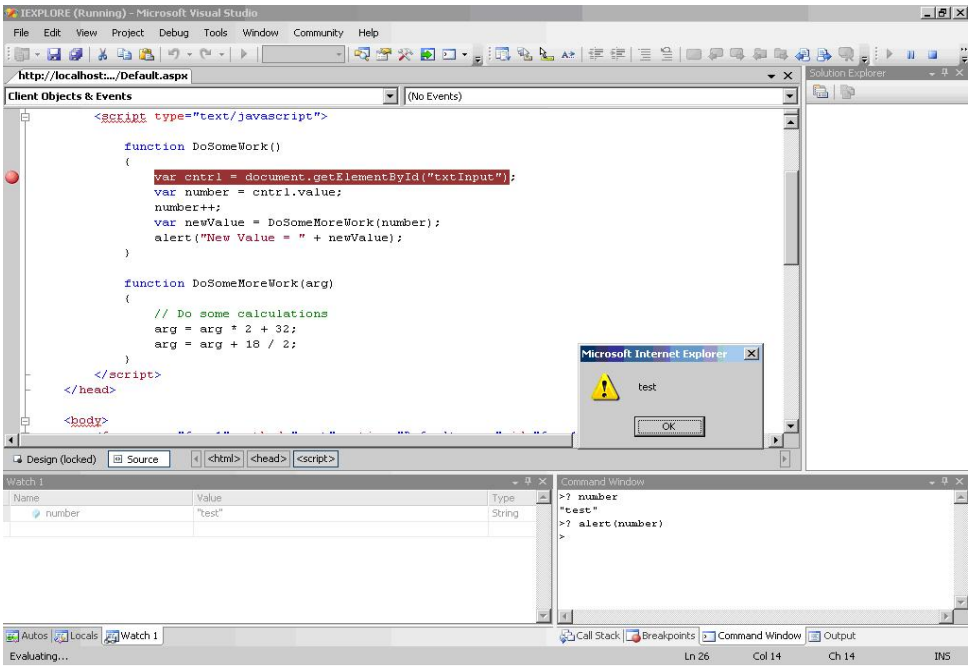
پنجره Command داری نماد > خواهد بود. در پنجره Command دستور زیر را وارد کرده و Enter را بزنید:

```
? number
```

احتمالا مقدار test نشان داده شود. حال سعی کنید که دستور زیر را وارد کنید:

```
? alert(number)
```

در اینصورت پیغامی که مقدار test را نشان می دهد ظاهر می شود. بعد از اجرای این دستورات ویژوال استودیو همانند شکل زیر خواهید بود.



علامت سوال اختصار print و یا display می باشد. این تکنیک یک روش بسیار مفید برای ارزیابی وضعیت های مختلف داخل برنامه تان می باشد.

فصل ششم: برنامه ChatRoom

در این فصل ایجاد یک chatroom را با کمک Ajax بررسی خواهیم کرد.کدهای کامل این برنامه در CD کتاب در مسیر project\chat و جدول های آن در مسیر project\database\chatroom موجود می باشد.

این برنامه یک chatroom عمومی است، یعنی هرکاربر با داشتن یک شناسه کاربری وارد محیط ارسال پیام می شود و می تواند پیام های خود را ارسال کند. پیام ارسال شده برای تمامی کاربرانی که همزمان در chatroom حضور داشته باشند قابل رویت است. بطور کلی در این برنامه امکان ارسال پیام به یک کاربر خاص به گونه ای که دیگر کاربران نتوانند آن پیام را مشاهده کنند وجود ندارد. کاربران در صورت نداشتن شناسه کاربری می توانند یک شناسه برای خود ایجاد کنند.

در سایت لیستی از کاربران فعال در chatroom نمایش داده می شود، هرکاربر با ورود به chatroom می تواند این لیست کاربران را مشاهده نماید. با کلیک کردن بر روی آیکون هر کاربر از لیست نمایش داده شده، اطلاعات شخصی آن کاربر نمایش داده می شود.

فرض کنید بخواهید این chatroom را بطور معمول در وب برنامه نویسی کنید، هر زمان که پیامی از سمت یک کاربر به کاربر دیگری ارسال می شود برای نمایش پیام جدید باید صفحه ی گیرنده پیام دوباره بارگذاری شود یا زمانیکه یک کاربر جدید به سایت وارد می شود باید لیست کاربران فعال دوبار بارگذاری شود. این عملیات در چنین برنامه ی خیلی خسته کننده است در نتیجه نوشتن این برنامه بصورت برنامه نویسی معمول وب یک تصمیم نادرست است.

جداول موجود در برنامه ChatRoom

تا اینجا تقریبا با شکل کلی برنامه آشنا شدید اما مرحله بعد ایجاد جدول های مورد نیاز برای این برنامه است. با نگاهی به مطالب بالا می توان متوجه شد که نیاز به جدولی برای نگهداری اطلاعات کاربری از جمله شناسه کاربری، می باشد. اطلاعات کاربر باید هنگام ایجاد یک شناسه برای کاربر از او درخواست شود و در جدول ثبت شود. اطلاعات مورد نیاز از یک کاربر را می توان بعنوان فیلد های جدول در نظر گرفت. فیلدهای داده ای که برای یک کاربر بطور معمول نیاز است در جدول ۱-۷ آورده شده است. ما یک جدولی به نام user با ۸ فیلد داده ای برای نگهداری اطلاعات کاربران در نظرگرفته ایم.

- فیلد ID شماره منحصر بفردی است که به کاربر در جدول تخصیص داده می شود. این فیلد بصورت خوردکار مقدار دهی می شود(بعنوان کلید اصلی).
- فیلد username و password شناسه کاربری فرد و رمزعبور در سیستم chat می باشد که به هنگام ورود به سایت به آن ها نیاز می باشد.

- فیلد firstname, lastname, age, gender مشخصات شخصی فرد است که هنگام ثبت نام کردن، از او درخواست می شود.

- فیلد status وضعیت کاربر را مشخص می کند. در این برنامه ما قرارداد کردیم که در صورت 0 بودن، کاربر در سیستم وارد نشده یا خارج شده (offline) و در صورت 1 بودن کاربر به سیستم login کرده است (online). این فیلد برای این است که با یک شناسه کاربری امکان ورود مجدد به سایت را نداشته باشیم و همچنین کاربران فعال را بتوانیم مشخص کنیم.

شما می توانید با اجرای این کد جدول را براحتی ایجاد کنید.

```
CREATE TABLE `user` (
     `ID`         INT NOT NULL AUTO_INCREMENT ,
     `username`   VARCHAR(50) NOT NULL ,
     `firstname`  VARCHAR(50) NOT NULL ,
     `firstname`  VARCHAR(50) NOT NULL ,
     `lastname`   VARCHAR(50) NOT NULL ,
     `password`   VARCHAR(50) NOT NULL ,
     `age`        INT NOT NULL ,
     `gender`     CHAR(1) NOT NULL ,
     `status`     TINYINT NOT NULL ,
     PRIMARY KEY (`ID`)
);
```

auto_increment	int(10)	ID*
	varchar(50)	username
	varchar(50)	firstname
	varchar(50)	lastname
	varchar(50)	password
	int(10)	age
	Char(1)	gender
	tinyint(1)	status

جدول ۱-۷ فیلدهای جدول user

اما نکته دیگری که در این برنامه وجود دارد، پیام است. پیام باید از سمت یک کاربر به سرور ارسال و از آن جا برای تمامی کاربران فعال ارسال شود در نتیجه ما نیاز به جدولی برای نگهداری اطلاعات پیام داریم تا مشخص شود: ارسال کننده پیام چه کسی است؟ در چه زمانی این پیام ارسال شده است؟ متن پیام چیست؟

در این قسمت به بررسی جدول پیام و فیلدهای داده ای آن می پردازیم. این جدول حاوی ۴ فیلد داده ای می باشد.

- فیلد id برای اختصاص یک شماره منحصربفرد به پیام است(بعنوان کلید اصلی).
- فیلد Message حاوی متن پیام است.

- فیلد senderId شناسه کاربری (username)فرد ارسال کننده است.
- فیلد Date تاریخ و زمان ارسال پیام است.

```
CREATE TABLE `message` (
     `senderId` VARCHAR(50) NOT NULL ,
     `message`  VARCHAR(50) NOT NULL ,
     `date`     DATETIME NOT NULL ,
     `Id`       INT NOT NULL AUTO_INCREMENT ,
     PRIMARY KEY (`Id`)
);
```

	varchar(50)	senderId
	varchar(50)	message
	datetime	date
auto_increment	int(10)	Id*

جدول ۲-۷ فیلدهای جدول پیام

چگو نگی پیاده سازی ChatRoom

در این مرحله از کار باید به طراحی برنامه بپردازید. در مرحله اول باید عملیات اجرایی در برنامه را شناسایی کنید. ایجاد یک شناسه کاربری، ورود به سایت، ارسال پیام، مشاهده لیستی از کاربران فعال، مشاهده اطلاعات کاربران، دریافت پیام و خروج از سایت عملیاتی هستند که در این برنامه انجام می گیرند.

در مرحله بعد باید عملیات را دسته بندی کنید تا بدانید در هر صفحه ای که طراحی می کنید چه عملیاتی اجرا شود و تعداد صفحات مورد نیاز برنامه چه تعداد است. این دسته بندی، به مجموعه عملیات مشابه و یا مرتبط با یکدیگر، نظر و تفکر طراح سایت بستگی دارد. دسته بندی که در این برنامه صورت گرفته بدین شکل است:

۱- ورود به سایت و ایجاد یک شناسه کاربری.

برای اینکه کاربر جدید هنگام مراجعه به برنامه دچار سردرگمی نشود این دو بخش در کنار هم ذکر شده است تا کاربر بلافاصله بعد از ایجاد یک شناسه کاربری بتواند به سایت وارد شود.

۲- مجموعه عملیات ارسال پیام، مشاهده لیستی از کاربران فعال، مشاهده اطلاعات کاربران، دریافت پیام و خروج از سایت را می توان در یک دسته قرار داد زیرا به نظر می رسد همه ی این عملیات باید در یک صفحه قرار بگیرند اما برخی از این عملیات مانند نمایش لیست کاربران فعال مرتبط با نمایش پیام ها نیست. در این مورد هنگام پیاده سازی بیشتر بحث می کنیم.

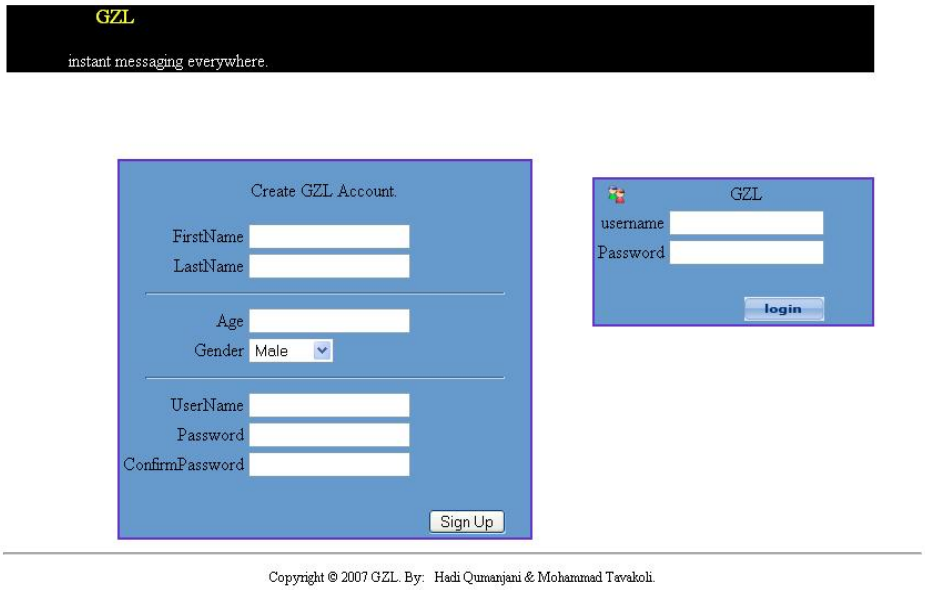
صفحه Login.html

همان طور که بیان شد هر دسته از عملیات منجر به طراحی یک صفحه وب می شوند در نتیجه برای دسته اول یک صفحه ورود به سایت با نام login.html ایجاد می کنیم. این صفحه دارای دو بخش است:

- signup بخش ایجاد شناسه کاربری برای کسانی که username ندارند.
- signon بخش ورود به برنامه برای کاربرانی که دارای username می باشند.

در شکل ۱-۷، نمای از طراحی این صفحه را مشاهده می نماید. این صفحه می تواند با طراحی شما متفاوت پیاده سازی شود.

در این قسمت به نوشتن کدهای مورد نیاز این بخش ها می پردازیم. کاربر در صورت ندا شتن username با وارد کردن اطلاعات شخصی خود و signup کردن می تواند یک username برای خود ایجاد کند. با فشرده شدن دکمه signup باید اطلاعات کاربر به جدول user اضافه شود. پس با فشردن دکمه signup باید عملیاتی انجام گیرد که ما اجرای آن عملیات را با اجرای تابع signUp() محقق می سازیم اما مرحله بعد باید تصمیم بگیریم این تابع چه عملیاتی را برای ما انجام دهد. در اینجا ما قصد داریم اطلاعات کاربران بصورت Ajax به سرور ارسال شود تا آن جا این اطلاعات به جدول کاربران اضافه گردد. برای ارسال اطلاعات به سرور در تمام طول این برنامه از متد GET استفاده می شود. ابتدا مقادیر داده ای درون فرم ثبت نام را به متغیرهای جاوااسکریپت انتساب می دهیم. به هر عنصر بر روی صفحه یک Id نسبت می دهیم تا بتوانیم با استفاده از دستور document.getElementById(id).value به مقدار آن عنصر دسترسی داشته باشیم.



شکل ۱-۷ نمای از طراحی صفحه login.html

```
function signUp(){
     var textSignupUserName = document.
     getElementById('textSignupUserName').value;
     var textName=document.getElementById('textName').value;
     var textLastName=document.getElementById('textLastName')
          .value;
     var signupPassword = document
          .getElementById('signupPassword').value;
     var textAge=document.getElementById('textAge').value;
     var selectGender =document.getElementById('selectGender')
          .options[document.getElementById('selectGender')
          .selectedIndex].value;
```

همان طور که گفته شد در این برنامه از متد GET برای ارسال اطلاعات به سمت سرور استفاده می کنیم به همین خاطر باید به نام صفحه ای که می خواهیم در سمت سرور اجرا شود این اطلاعات را اضافه کنیم در نتیجه یک رشته به نام snd با مقادیر متغیرها ایجاد می کنیم.

```
var snd = '?textSignupUserName=' + textSignupUserName
     +'&textName=' + textName + '&textLastName=' +
     textLastName +'&signupPassword=' + signupPassword
     +'&textAge=' + textAge + '&selectGender='
     + selectGender +'&textAge=' + textAge;
```

در نظر بگیرید که برای ارسال اطلاعات به سرور و دریافت جواب از سرور با استفاده از روش Ajax یک کلاس به نام Asynchronous نوشته ایم. از کلاس Asynchronous یک شی ایجاد می کنیم و صفحه مورد نظر در سرور را به همراه آن رشته تولیدی در قسمت قبل فراخوانی می کنیم.

```
Asynchronous = new Asynchronous();
```

متدی در این کلاس برای فراخوانی صفحه از سمت سرور به نام call نوشته شده است. متد call برای اشیا کلاس دارای سه پارامتر است که پارامتر اول آن نام صفحه ی است که می خواهیم در سمت سرور اجرا شود پارامتر دوم یک مقدار تهی است و پارامتر سوم نام روش ارسال اطلاعات به سرور می باشد.

```
Asynchronous.call('SignUp.php'+snd,'','GET');

}
```

در این جا بهتر است کلاس Asynchronous را بررسی کنیم. کلاس Asynchronous کلاسی است که در جاوااسکریپت تعریف می شود و توابع لازم برای کار با شی XmlHttpRequest را درکنار هم قرار می دهد این کلاس را در یک فایل دیگر به نام Asynchronous.js تعریف می کنیم. برای تعریف کلاس در جاوااسکریپت ابتدا یک تابع با نام کلاس تعریف می کنیم. این تابع در واقع سازنده کلاس است.

```
function Asynchronous( ) {
     //...
}
```

برای تعریف متد ها ی کلاس از کلمه prototype استفاده می کنیم بدین صورت:

```
نام کلاس.prototype.نام متد = نام یک تابع;
```

متد کلاس را به نام یک تابع نسبت می دهیم که درنتیجه فراخوانی متد موجب فراخوانی آن تابع می شود.

این تعریف ها خارج از تابع سازنده و مجزا نوشته می شوند.

می توان متدها را بصورت دیگر نیز تعریف کرد، متدها را بدون کلمه prototype در داخل سازنده کلاس به نام یک تابع نسبت داد اما تفاوتی بین این دو روش است. در روش اول متدهای کلاس بصورت مشترک برای تمام نمونه های کلاس استفاده می شوند اما در روش دوم هر نمونه کلاس از هر تابع مرتبط به این متدها یک کپی می گیرد. در این کلاس ما ۵ تا متد بدین شکل تعریف می کنیم.

```
Asynchronous.prototype.loading = Asynchronous_loading;
Asynchronous.prototype.loaded = Asynchronous_loaded;
Asynchronous.prototype.interactive = Asynchronous_interactive;
Asynchronous.prototype.complete = Asynchronous_complete;
Asynchronous.prototype.call = Asynchronous_call;
```

تابع های مرتبط با این متدها را نیز تعریف می کنیم.

```
function Asynchronous_loading()
{
}
function Asynchronous_loaded()
{
}
function Asynchronous_interactive()
{
}
function Asynchronous_complete(status, statusText,
          responseText, responseHTML)
{
}
```

متد complete را با یک تابعی مرتبط می سازیم که دارای چهار تا پارامتر است. این متد زمانی اجرا می شود و موجب فراخوانی تابع مرتبط به خود می شود که درخواست در سرور اجرا شده باشد و اطلاعات برای کلاینت ارسال شود. در سازنده کلاس یک عضو داده ای به نام _xmlhttp تعریف می کنیم که به یک نمونه از شی XMLHttpRequest نسبت داده می شود. لازم بذکر است که برای دسترسی به اعضای داده ای در یک کلاس باید از کلمه کلیدی this استفاده شود.

```
function Asynchronous( )
{
    this._xmlhttp = new XmlHttpRequest();
}
```

ایجاد نمونه از شی XmlHttpRequest موجب اجرای تابع سازنده XmlHttpRequest می شود. در این تابع، نوع شی XmlHttpRequest متناسب با مرورگر انتخاب می شود. این دستورات قبلا توضیح داده شده بودند. این تابع یک کلاس XmlHttpRequest ایجاد می کند و خود تابع، سازنده کلاس می باشد. در دستور if اول بررسی می کنیم که شی XMLHttpRequest موجود است؟ اگر موجود باشد از آن یک نمونه می گیریم. دستور window.XMLHttpRequest برای تمام مرورگرها بجز Internet Explorer کار می کند. در دستور if دوم با دستور window.ActiveXObject مرورگر IE را بررسی می کنیم در حلقه for تک تک نسخه های مرورگر IE بررسی می شود و از شی مورد نظر نمونه می گیریم. در صورتی که مرورگر شناخته نشود پیغام خطای را ایجاد می کنیم.

```
function XmlHttpRequest()
{
    if(window.XMLHttpRequest)
     {
        return new XMLHttpRequest();
    }
    else if(window.ActiveXObject)
     {
        var msxmls = new Array(
            'Msxml2.XMLHTTP.5.0',
            'Msxml2.XMLHTTP.4.0',
            'Msxml2.XMLHTTP.3.0',
            'Msxml2.XMLHTTP',
            'Microsoft.XMLHTTP');
        for (var i = 0; i < msxmls.length; i++)
          {
            try
               {
                return new ActiveXObject(msxmls[i]);
            } catch (e)
               {
            }
        }
    }
    throw new Error("Could not instantiate XMLHttpRequest");
}
```

متد call برای ایجاد یک درخواست در سمت کلاینت می باشد. بعد از فراخوانی این متد تابع Asynchronous_call فراخوانی می شود. برای اینکه نمونه های متعدد از کلاس Asynchronous بتوانند با یکدیگر کار کنند یک متغیر به نام instance تعریف کردیم که به شی جاری مرتبط می شود در این صورت برای درخواست های متفاوت، تابع Asynchronous_call جداگانه فراخوانی می شود.

```
function Asynchronous_call(url,send,method)
{
    var instance = this;
    this._xmlhttp.open(method, url, true);
    this._xmlhttp.onreadystatechange = function()
     {
        switch(instance._xmlhttp.readyState)
          {
            case 1:
                instance.loading();
                break;
            case 2:
                instance.loaded();
```

```
                break;
            case 3:
                instance.interactive();
                break;
            case 4:
            try{

                instance.complete(instance._xmlhttp.status,
                    instance._xmlhttp.statusText,
                    instance._xmlhttp.responseText,
                 instance._xmlhttp.responseXML);

                }catch(e){
                    instance.complete('','','Error','' );
                }break;
        }
    }
    this._xmlhttp.send(send);
}
```

برای استفاده از کلاس Asynchronous باید این کلاس به صفحه وب اضافه گردد این عملیات را با دستور زیر انجام می دهیم.

```
<script type="text/javascript"
     src="Java_script/Asynchronous.js"></script>
```

رشته snd به نام صفحه مورد نظر الحاق می شود. بعد از فراخوانی صفحه signUp.php توسط دستور asynchronous.call('SignUp.php'+snd,'','GET') اطلاعات کاربر به سرور و این صفحه ارسال می شود. در این صفحه ما باید اطلاعات کاربر را دریافت و بعد از اینکه شناسه کاربری چک شد که تکراری نباشد به جدول user اضافه شود. در این جا به بررسی کد صفحه signup.php می پردازیم.

```
<?php
```

در این قسمت اطلاعات کاربر که از سمت کلاینت ارسال شده، دریافت می شود و در متغیرهای مرتبط نگهداری می شود.

```
$UserName=$_GET['textSignupUserName'];
$FirstName =$_GET['textName'];
$LastName =$_GET['textLastName'];
$Password =$_GET['signupPassword'];
$Age =$_GET['textAge'];
$Gender =$_GET['selectGender'];
```

مرحله بعد اتصال به بانک اطلاعاتی است که با دستور mysql_connect('localhost','root','') انجام می شود. پارامتر اول این دستور آدرس سرویس دهنده وب است پارامتر دوم نام کاربری برای اتصال به بانک اطلاعاتی و پارامتر سوم این دستور رمز عبور برای اتصال می باشد.

```
$link = mysql_connect('localhost' ,'root','');
```

بعد از اتصال به سرور باید بانک اطلاعاتی مورد نظر را انتخاب کنیم.

```
mysql_select_db('chatroom',$link);
```

در این مرحله یک دستور پرس و جو برای بررسی تکراری نبودن شناسه کاربری می نویسیم.

```
// search for Duplicate userName.
$cmd = "Select id from User where UserName='$UserName'";
```

پرس و جو مورد نظر را با دستور زیر اجرا می کنیم.

```
$res = mysql_query($cmd,$link);
```

برای اینکه بدانیم آیا رکودی پیدا شد یا نه؟ از دستور `mysql_num_rows($res)` استفاده می کنیم. پارامتر ارسال شده به این تابع مجموعه رکورد حاصل از پرس وجو می باشد. اگر تعداد رکوردهای حاصل از این جستجو برابر 1 باشد پس شناسه کاربری در جدول کاربران موجود است.

```
if(mysql_num_rows($res)==1)
{
     echo "Duplicate '$UserName'";
     exit();
}
```

اما اگر حاصل این جستجو یک نباشد(منظور صفر باشد) در نتیجه شناسه کاربری تکراری نیست و اطلاعات کاربری باید به جدول user اضافه شود.

```
// Insert new account to user table.
$q = "insert into User
     (UserName,FirstName,LastName,Password,Age,Gender)
      values";

$q.="'$UserName','$FirstName','$LastName','$Password'
     ,$Age,'$Gender')";
$res = mysql_query($q,$link);

if (!$res)
```

اگر خطایی حین اجرای دستور insert رخ دهد دستور `echo mysql_error()` توضیحات خطا را بر روی صفحه وب نمایش می دهد.

```
     echo mysql_error();
else
     echo 200;
mysql_close($link);

?>
```

با استفاده از دستورات قبل اطلاعات کاربر به جدول کاربران اضافه شد اما مدیریتی برای صحت اطلاعات نداشتیم مثلا اینکه فیلد email بصورت یک آدرس email باشد و یا اینکه فیلد age باید عدد باشد.پس نیاز به قسمت کدی برای مدیریت بر روی این فیلدها، احساس می شود. در قسمت دیگر برنامه مکانی برای ورود به سایت قرار دارد. کاربر با تایپ کردن شناسه کاربری و رمز عبور خود می خواهد به برنامه chatroom وارد شود. در نتیجه نیاز به تابعی برای ارسال اطلاعات کاربر به سرور برای بررسی صحت اطلاعات داریم. این تابع اطلاعات وارد شده توسط کاربر را به صفحه SignOn.php ارسال می کند.

```
 function signOn(e){


     var textUserName = document
          .getElementById('textUserName').value;
     var signOnPassword = document
          .getElementById('signOnPassword').value;
     var snd = "?textUserName=" + textUserName +
          "&signOnPassword=" + signOnPassword ;

     asynchronous = new Asynchronous();
     asynchronous.complete = signOncomplete;
     asynchronous.call('SignOn.php'+snd,'','GET');
}
function signOncomplete(status, statusText, responseText,
      responseXML){

     if (responseText == '200'){
          window.location.href = "./HomePage.html";
     }
     else{

          Var textSignupUserName = document
               .getElementById('buttonSignon');

          var errorMsg = document.createElement( 'span' );
          errorMsg.className = 'error';
          errorMsg.appendChild( document.createTextNode
               (    responseText ) );

          textSignupUserName.parentNode.insertBefore(
               errorMsg, textSignupUserName );

          var errorMsg = document.createElement( 'br' );
          errorMsg.className = 'error';

          textSignupUserName.parentNode.insertBefore(
               errorMsg, textSignupUserName );
      }

  }
```

با کلیک کردن دکمه login صفحه signon.php فراخوانی می شود. در این صفحه ابتدا جستجو می شود که کاربری با username و password وارد شده داریم، که به سایت وارد نشده است. در اینصورت session را ایجاد می کنیم و متغیر آن را به Username انتساب می دهیم:

```
<?php

     $UserName=$_GET['textUserName'];
     $Password =$_GET['signOnPassword'];
     $link = mysql_connect('localhost' ,'root','');
     mysql_select_db('chatroom',$link);
```

یک پرس و جو برای اینکه بدانیم کاربری با مشخصات وارد شده وجود دارد در حالیکه فعال نباشد یعنی Status=0 باشد.

```
     // search for Valid userName & PassWord.
     $cmd = " Select id,userName from user where
          UserName='$UserName' and Password='$Password'
          and Status=0";
     $res = mysql_query($cmd,$link);
     $r=mysql_fetch_array($res);

     if( mysql_num_rows($res)==0 )
     {
          echo "Invalid Username & Password";
          exit();
     }
```

Session را باز می کنیم و متغیر Session را با شناسه کاربری مقدار می دهیم.

```
     Session_start();
     $_SESSION['UserName'] = $r[1];
```

وضعیت کاربر هنگام ورود به سایت از حالت غیر فعال به فعال تغییر می کند یعنی فیلد status را 1 می کنیم.

```
     $q = "update  User set status=1 where
          UserName='$UserName' ";
     $updateRes = mysql_query($q,$link);

     echo '200';
     mysql_close($link);
?>
```

در این جا طراحی و پیاده سازی دسته اول به پایان رسید اما همان طور که قبلا اشاره کردیم دسته دوم شامل مجموعه عملیات ارسال پیام، مشاهده لیستی از کاربران فعال، مشاهده اطلاعات کاربران، دریافت پیام و خروج از سایت می باشد. این عملیات بطور مجزای باید اجرا شوند ولی مجموعه این عملیات باید قابل دسترس به آسانی برای کاربران باشد. برای اینکه بتوانیم هر عملیات را بطور جداگانه ی اجرا کنیم و تمامی مجموعه عملیات در

یک صفحه قابل دسترس باشد باید بتوانیم صفحه را به چند بخش تقسیم کنیم که هر بخش یک صفحه وب جداگانه ای باشد.

برای عملی کردن این ایده در html از **<iframe>** استفاده می کنیم. **<iframe>** برای نمایش یک صفحه بعنوان بخشی در یک صفحه دیگر استفاده می شود. از **<iframe>** زمانی استفاده می شود که می خواهیم طرح کلی سایت ثابت باشد و تغییراتی در بخشی از سایت هر زمان که نیاز باشد روی دهد. با **<iframe>** می توان عملیاتی که مجزا باشند با Ajax را در صفحات متفاوت قرار داد. تگ **iframe** دارای خاصیتی به نام src است که در آن مسیر صفحه ای که می خواهیم در **iframe** نمایش داده شود را مشخص می کنیم.

صفحه homepage.html

در قسمت title این صفحه شناسه کاربری، کاربر وارد شده به سایت نمایش داده می شود. تابع setTitle بعد از فراخوانی صفحه GetInfo.php عنوان صفحه homepage.html را تغییر می دهد.

```
 function setTitle(){
     asynchronous = new Asynchronous();
```

متد complete زمانی اجرا می شود که اطلاعات از سرور دریافت می شود. اطلاعات دریافت شده از سرور در پارامتر responseText تابع مرتبط به آن قرار می گیرد که حاوی username کاربر می باشد که ما آن را به رشته ای برای نمایش در عنوان سایت اضافه می کنیم.

```
Asynchronous.complete = function(status, statusText,
          responseText, responseXML){
     if (responseText!=0)
          document.title= "Welcome    "+responseText
     else
           location.href("login.html");
};
   asynchronous.call('GetInfo.php','','GET');
}
```

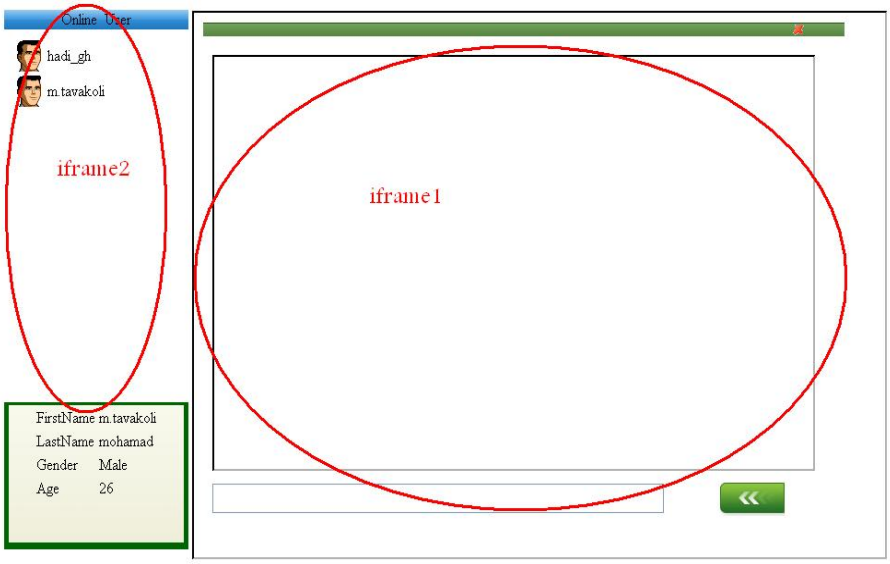
در صفحه GetInfo.php بعد از بررسی موجود بودن شناسه کاربری (username) در session، شناسه کاربری برای کلاینت با استفاده از دستور echo ارسال می شود.

```
<?php
     session_start();
     if(isset($_SESSION['UserName']))
           echo $_SESSION['UserName'];
     else
          echo '0' ;
?>
```

صفحه homepage.html شامل دو قسمت است که هر قسمت یک <iframe> می باشد.

```
<iframe  src ="messageBox.html" id='List' style=" margin-
```

```
top:10px; width:750px;height:555px" ></iframe>
```

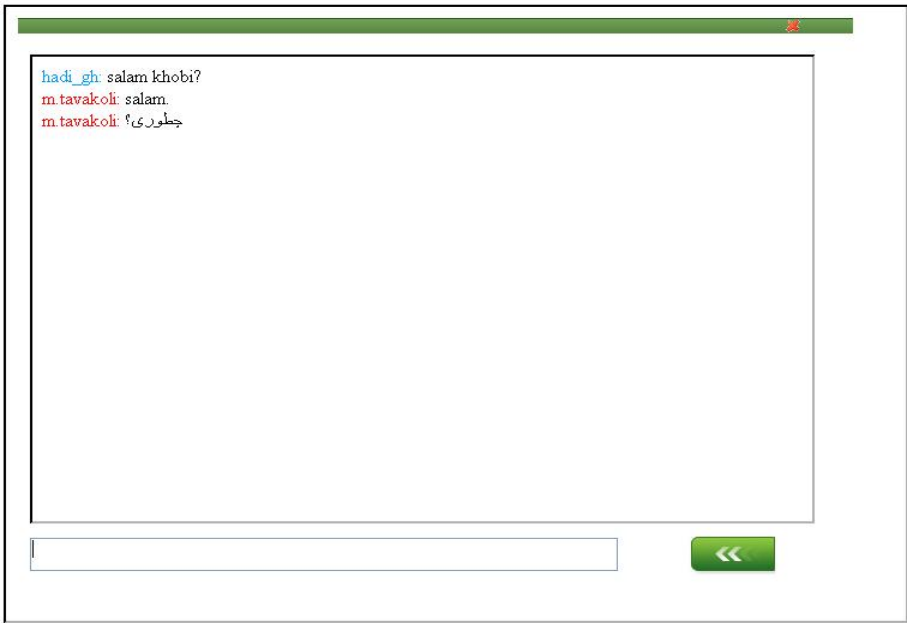


شکل ۲-۷ طراحی صفحه homepage.html

صفحه messagebox.html

یک صفحه حاوی یک قسمت برای نوشتن پیام ها و دکمه ی برای ارسال پیام ها و یک تصویر ضربدر برای خروج از سایت می باشد. می خواهیم بعد از فشرده شدن دکمه ارسال، پیام داخل TextBox ارسال شود. بدین منظور تابعی بدین صورت نوشته ایم تا زمانیکه دکمه فشرده شد فراخوانی شود. در این تابع متن پیام به صفحه sendMessage.php ارسال می شود.

```
function SendMessage(){
     asynchronous = new Asynchronous();
     asynchronous.complete = function(status, statusText,
          responseText, responseXML){
       document.getElementById("message").value = "";
       document.getElementById("message").focus();
     };

     var mssg = document.getElementById("message").value;
     asynchronous.call('sendMessage.php?msg='+mssg,'','GET');

}
```



شکل ۳-۷ طراحی صفحه messagebox.html

برای سهولت کار در برنامه با فشرده شدن دکمه Enter نیز پیام ارسال می شود که موجب فراخوانی تابع SendMessagekey می شود.

```
function SendMessagekey(e){
     if (e.keyCode == '13'){
          asynchronous = new Asynchronous();
          asynchronous.complete = function(status,
                statusText, responseText, responseXML){

          document.getElementById("message").value = "";
          document.getElementById("message").focus();
          }
     };
     var mssg = document.getElementById("message").value;
     asynchronous.call('sendMessage.php?msg='+mssg,'','GET');
}
```

در صفحه sendMessage.php ابتدا از session شناسه کاربر خوانده می شود سپس متن پیام دریافت می شود و تاریخ دریافت پیام نیز از سیستم دریافت می شود. در صورتی که عملیات با موفقیت صورت پذیرفت مقدار 200 را برای کلاینت جهت اطلاع از انجام شدن موفق عملیات ارسال می کنیم.

```
<?php
     session_start();
     $usrName = $_SESSION["UserName"];

     $msg = $_GET["msg"];
     $date = date("Y-m-d H:i:s");

     $link = mysql_connect('localhost','root','');
     mysql_select_db("chatroom",$link);
```

با این دستور اطلاعات پیام به جدول message اضافه می شود.

```
$q="insert into message (SenderId, Message ,Date) values
      ('$usrName','$msg','$date')";

$res=mysql_query($q,$link);
if ($res)
     echo '200';
mysql_close($link);
?>
```

در این صفحه با زدن علامت ضربدر با فراخوانی تابع logOut، صفحه logout.php اجرا می شود.

```
function logOut(){
          asynchronous = new Asynchronous();
          asynchronous.complete = function(status,
                statusText, responseText, responseXML){
```

زما نیکه کاربر از برنامه خارج شد صفحه ورود به سایت بار می شود.

```
          if(responseText == '200')
               parent.location.href("login.html");

          };
          asynchronous.call('LogOut.php','','GET');
}
```

در logout.php ابتدا کاربر از حالت فعال خارج می شود(فیلد status=0 می شود). دستور پاک کردن session نیز اجرا می شود و جهت اطلاع کلاینت از انجام موفق خروج از سایت 200 به کلاینت ارسال می شود.

```
<?php
     session_start();
     $u = $_SESSION['UserName'];
     $link = mysql_connect('localhost','root','');
     mysql_select_db("chatroom",$link);
     $q=" update User set status=0  where  UserName='$u'";
     $res = mysql_query($q,$link);
     mysql_close($link);
     session_destroy();
     if($res)
          echo '200';
?>
```

همچنین messagebox.html دارای یک قسمتی برای نمایش پیام ها می باشد که در واقع یک صفحه دیگر است.

```
<iframe  src="message.html" id='List' style=" margin-top:10px;
     width:650px;height:420px" ></iframe>
```

صفحه message.html

صفحه message.html صفحه ی است که در داخل messagebox.html نمایش داده می شود.

در این صفحه بصورت ajax هر ثانیه صفحه showmessage.php برای نمایش پیام ها فراخوانی می شود. در صفحه showmessage.php پیام های انتخاب می شوند که دقیقا در آن لحظه دریافت می شوند و سپس نمایش داده می شوند. شناسه کاربری ارسال کننده پیام برای خودش به رنگ قرمز و برای دیگر کاربران به رنگ آبی نمایش داده می شود. متن پیام جدید به متن پیام های دیگر که در داخل یک با id='show' قرار دارند، اضافه می شود.

```
function showMessage(){
          asynchronous = new Asynchronous();
          asynchronous.complete = function(status,
                statusText, responseText, responseXML){
          if(responseText !=null)
          document.getElementById('show').innerHTML +=
                    responseText ;
          };
          asynchronous.call('showMessage.php','','GET');
          setTimeout(showMessage,1000);
}
```

در یک برنامه Chatroom هر زمان امکان تولید و ارسال یک پیام جدید می باشد به همین منظور باید عملیات نمایش پیام ها متناوبا در یک بازه زمانی خاص اجرا گردد. با استفاده از تابع setTimeout می توان مشخص کرد یک عملیات مشخص در چه بازه های زمانی اجرا شود. پارامتر اول این تابع نام تابعی است که می خواهیم اجرا شود و پارامتر دوم زمان اجرا آن در بازه های زمانی(برحسب هزارم ثانیه)

```
setTimeout(showMessage,1000);
</script>
</head>
<body >
     <span id='show' style="width:100% ;height:320px"></span>
</body>
</html>
```

در تابع ShowMessage صفحه ShowMessage.php فراخوانی می شود.

```
<?php
     session_start();
     if(isset($_SESSION["UserName"]))
               $user = $_SESSION["UserName"];
     else
     {
          exit(0);
     }

     $link = mysql_connect('localhost','root','');
```

```
mysql_select_db("chatroom",$link);
```

در این جا پرس و جوی می نویسیم که پیام های که هم اکنون دریافت شدند انتخاب شوند تا برای تمامی کاربران ارسال شوند.ارسال پیام با دستور echo $str انجام می گیرد.

```
$q="select * from message Where( day(now())-day(date)=0
      and month(now())-month(date)=0 and year(now())-
     year(date)=0 and hour( now()) -hour(date)=0 ) and
     (( MINUTE( now()) -MINUTE(date)=0  and  second(
     now()) - second(date) = 1) or
     ( minute(now())- minute(date)=1  and   60
     +second(now())- second(date)  = 1) )";

$res=mysql_query($q,$link);
$row=mysql_fetch_array($res);

$str = NULL;
while ($row){
     if ($user == $row[0])
     {
     $str.="<font style=\"color:#FF0000;\">$row[0]:
          </font> $row[1]<br>" ;
     }
     else
     {
     $str.="<font style=\"color:#0099FF;\">$row[0]:
          </font> $row[1]<br>";
     }
     $row = mysql_fetch_array($res);
}

echo $str;
mysql_close($link);
?>
```

در قسمت دیگر صفحه homepage.html کاربران فعال (online) نمایش داده می شوند.

```
<iframe  src="userList.html" id='List' style=" background-
     color:#FFFFFF;margin-top:0px;width:200px;
     border:thin;height:380px" >
</iframe>
```

در این قسمت، صفحهuserList.html نمایش داده می شود. این صفحه برای نمایش لیستی از کاربران فعال در سیستم می باشد و این عمل را با فراخوانی صفحه showlist.php انجام می دهد. اما دلیل استفاده از تابع setTimeout در این جا برای آن است که هر لحظه امکان ورود یک کاربر جدید به سایت و یا

خروج یک کاربر وجود دارد در نتیجه این لیست کاربران فعال باید هر چند ثانیه دوباره چک شود. هنگامی که لیست کاربران از سرور ارسال شود متد complete اجرا می شود و لیست کاربران را که در پارامتر responseText قرار دارد برای نمایش به محتوای یک با id='divList' انتساب می دهد.

```
function setLIst(){

     asynchronous = new Asynchronous();
     asynchronous.complete = function(status, statusText,
               responseText, responseXML){
     document.getElementById('divList').innerHTML =
               responseText;
     };
     asynchronous.call('ShowList.php','','GET');
     setTimeout(setLIst, 5000);
}
setTimeout(setLIst, 1000);
function showInfo(){
     alert(parent.value);
     parent.handleResponse;
}
```

در لیست، ما می خواهیم کاربر با انتخاب آیکون یکی از کاربران بتواند اطلاعات فردی آن کاربر را مشاهده نماید. در صفحه 'ShowList.php' بعد از جستجوی کاربران فعال، اطلاعات آن ها را در سطر های یک جدول قرار می دهیم و در خاصیت onclick خانه های جدول دستور فراخوانی تابع showInfo را قرار می دهیم. لیست کاربران فعال که در رشته $str قرار دارد با دستور echo به سمت کلاینت ارسال می شود.

```
<?php
     $link=mysql_connect('localhost','root','');
     mysql_select_db('chatroom',$link);
     $q = "select * from User where status=1";
     $res = mysql_query($q,$link);
     $r = mysql_fetch_array($res);
     $str= NULL;
     $str = "<table>";

               while($r != NULL)
               {
                    if ($r[6]=='M')
                         $str.= "<tr > <td> <img
                              src='Image\m2.ico'></td><td
                              style=\"cursor:pointer\"
                              onclick='parent.showInfo
                              (this)'>$r[1]</td></tr>";
                    else
                         $str.= "<tr > <td> <img
                              src='Image\f2.ico'></td>
                              <td  style=\"cursor:pointer
                              \"onclick='parent.showInfo
                              (this)'>$r[1]</td></tr>";
```

```
                    $r = mysql_fetch_array($res);
                }
     $str.="</table>";
     echo $str;
     mysql_close($link);
?>
```

با کلیک کردن بر روی خانه های جدول تولیدی صفحه showlist.php، جهت نمایش مشخصات شخصی کاربران در قسمت پائین صفحه، تابع showInfo اجرا می شود که موجب فراخوانی صفحه userinfo.php می شود. همچنین می خواهیم لیست ارسالی از سمت سرور در یک با id='UserInfo' قرار گیرد.

```
function showInfo(e){
    asynchronous = new Asynchronous();
    asynchronous.complete = function(status, statusText,
                responseText, responseXML){
         document.getElementById("UserInfo").innerHTML =
                    responseText;
    };
    asynchronous.call('UserInfo.php?UserName='+e.innerHTML,'',
          'GET');
}
```

در صفحه userinfo.php اطلاعات کاربر مورد نظر در جدولی قرار می گیرد و برای نمایش با دستور echo $str به کلاینت ارسال می شود.

```
<?php
     $userName=$_GET['UserName'];

     $link=mysql_connect('localhost','root','');
     mysql_select_db('chatroom',$link);

     $q="select * from user where UserName='$userName'";
     $res = mysql_query($q,$link);
     $r = mysql_fetch_array($res);

     $str = "<table>";
     $str .="<tr><td>FirstName</td> <td> $r[1] </td></tr>"  ;
     $str .= "<tr><td>LastName</td> <td> $r[2] </td></tr>";
     if($r['Gender']=='M')
          $str .= "<tr><td>Gender</td> <td>Male</td></tr>";
     else
          $str  .=  "<tr><td>Gender</td>  <td>Female</td></tr>";

     $str .= "<tr><td>Age</td> <td>$r[5]</td></tr>";
     $str .="</table>";
     echo $str;
     mysql_close($link);
?>
```

این برنامه در حین سادگی بسیار کارآمد و زیباست.

فصل هفتم: برنامه BookStore

BookStore یک وب سایت نمایشگاه و فروشگاه اینترنتی کتاب می باشد. این برنامه بیشتر برای آشنایی خواننده با چگونگی استفاده از Ajax در برنامه های تجاری می باشد. در این برنامه از زبان PHP برای کدنویسی سمت سرور استفاده شده است. کدهای کامل این برنامه در CD کتاب در مسیر project\bookstore و جدول های آن در مسیر project\database\bookstore موجود می باشد. ابتدا بهتر است با کلیات و عملیات اجرایی برنامه بیشتر آشنا شوید.

در این پروژه ۳ نوع کاربر تعریف شده است:

۱- کاربر مدیر سایت

۲- کاربر عضو سایت

۳- کاربر میهمان

عملیات اجرایی برنامه: کاربران هنگام ورود به سایت و مشاهده صفحه اصلی سایت می توانند اطلاعات مورد نیاز خود را در مورد کتاب جستجو نمایند. امکان دانلود کردن فایل از وب سایت برای کاربران فراهم است. این فایل ها می توانند نسخه الکترونیکی کل کتاب یا چند فصل برای نمونه باشند. در صورتیکه کاربران برای خرید کتابی تمایل داشته باشند باید به سایت login کنند تا بتوانند کتاب مورد نظر خود را خریداری نمایند. بطور کلی خرید بدون عضویت در سایت امکان پذیر نیست. برای ورود به سایت نیاز به کدکاربری و رمز عبور می باشد. کاربرانی که کدکاربری ندارند می توانند با رفتن به صفحه ثبت نام و پر کردن فرم ثبت نام، یک کدکاربری برای خود ایجاد و به سایت وارد شوند. در هنگام ثبت نام بطور پیش فرض مقدار ۵۰۰۰۰ بعنوان موجودی اولیه به کاربران داده می شود. در زمان ورود به سایت اگر اطلاعات کد کاربری یا کلمه عبور اشتباهی وارد شود بدون اینکه صفحه تغییری کند پیغام هشداری نمایش داده می شود نمایش این پیغام توسط Ajax پیاده سازی شده است.

زمانیکه کاربر به سایت وارد می شود، اطلاعات شخصی کاربر به همراه میزان موجودیش، نمایش داده می شود. کاربر می تواند کتاب مورد نظر خود را در قسمت مشخص، جستجو کند و با وارد کردن تعداد خرید و فشردن دکمه خرید، آن کتاب را به سبد خرید خود اضافه نماید. سبد خرید در سایت قابل رویت است و کاربر می تواند هر زمان نسبت به حذف خرید از سبد خرید اقدام نماید. در پایان عملیات خرید، کاربر باید خرید را تایید کند در این صورت مبلغ کتاب های موجود در سبد خرید از موجودی کاربر کم می شود. انجام این عملیات بصورت Ajax می باشد تا صفحه دوباره بارگذاری نشود. همچنین هر کتاب می تواند دارای یک فایل به منظور ارائه سرفصل های کتاب به مشتریان، باشد.کاربران امکان دانلود این فایل را دارند. کاربران می توانند در سایت لیست کتاب های که قبلا توسط آن ها خریداری شده است به همراه تاریخ خرید کتاب ها، را مشاهده نمایند.

کاربران نوع دیگر، کاربران میهمان هستند که بدون داشتن کد کاربری می خواهند اطلاعاتی از کتاب ها را جستجو و مشاهده یا فایل مربوط به کتاب را دانلود نمایند. کاربر نوع میهمان در صفحه اصلی سایت شروع به جستجوی کتاب می نماید و اطلاعات کتاب درخواستی او در صفحه دیگر نمایش داده می شود.

کاربر نوع دیگر مدیر سایت می باشد. مدیر سایت می تواند لیستی از کاربران عضو سایت را مشاهده نماید. همچنین می تواند اطلاعات خرید یک کاربر خاص را مشاهده، برای یک کاربر خاص اقدام به شارژ موجودیش نماید و همچنین لیستی از تاریخ ها و مقادیر شارژ حساب های مشتریان را مشاهده نماید. همچنین امکان انجام عملیات اضافه کردن، حذف، ویرایش را بر روی کتاب ها و امکان Upload کردن تصویر و فایل الکترونیکی کتاب ها را داراست.

جداول موجود در سیستم

در این برنامه ما نیاز به جدول های برای نگهداری اطلاعات کتاب ها، اطلاعات کاربران، اطلاعات خرید و اطلاعات شارژ موجودی کاربران داریم. در اینجا به بررسی جدول های لازم در این برنامه می پردازیم.

جدول کتاب

جدول کتاب، جدولی برای نگهداری مشخصات کتاب ها می باشد.

Column Name	Data Type	Length	Allow Nulls
code	int	4	
Isbn	varchar	20	✓
name	varchar	20	✓
subject	varchar	20	✓
author	varchar	20	✓
translater	varchar	20	✓
press	varchar	20	✓
price	float	8	✓
Image	varchar	150	✓
[language]	varchar	10	✓
path	varchar	150	✓

جدول ۱-۸ جدول کتاب

- Code یک شماره منحصر بفرد برای هر کتاب
- Isbn شابک کتاب
- Name نام کتاب
- Subject موضوع کتاب
- Author نام نویسنده کتاب
- Translater نام مترجم کتاب
- Press نام انتشارات کتاب
- Price قیمت کتاب
- Image مسیر تصویر upload شده کتاب
- Language زبان کتاب
- Path مسیر فایل الکترونیکی کتاب برای دانلود کردن.

جدول سبد خرید

جدول سبد خرید حاوی اطلاعات اقلام خرید کاربران می باشد. اطلاعات این جدول موقتی است، تا زمانی که کاربر خرید خود را تایید نماید. به ازای هر خرید کاربر یک رکورد به این جدول اضافه می شود. علت استفاده از جدول سبد خرید بعنوان یک جدول موقتی این است که این امکان در سایت فراهم است که کاربر بدون تایید خرید از انجام کل خرید منصرف شود و از سایت خارج شود.

Column Name	Data Type	Length	Allow Nulls
BookCode	int	4	
price	float	8	
number	int	4	
total_price	float	8	
customerid	int	4	
[date]	datetime	8	

جدول ۲-۸ جدول سبد خرید

- bookCode کد کتاب
- Price قیمت کتاب
- Number تعداد خرید
- total_price قیمت کل
- customerid شماره مشتری
- date تاریخ خرید.

جدول آرشیو خرید

پس از تا ئید خرید توسط خریدار اطلاعات سبد خرید به این جدول وارد می شود. ساختار این جدول مشابه جدول سبد خرید می باشد.

جدول کاربران

مشخصات کاربران در این جدول نگهداری می شود.

Column Name	Data Type	Length	Allow Nulls
customerid	int	4	
customername	varchar	20	✓
customerfamily	varchar	20	✓
address	varchar	50	✓
tel	varchar	15	✓
mail	varchar	50	✓
account	float	8	✓
username	varchar	50	✓
pass	varchar	50	✓
status	tinyint	1	✓
type	tinyint	1	✓

جدول ۳-۸ جدول اطلاعات کاربران

- Customerid شماره کاربر
- Customername نام کاربر
- Customerfamily نام خانوادگی کاربر
- Address آدرس کاربر

- Tel شماره تلفن کاربر
- Mail آدرس پست الکترونیکی کاربر
- Account موجودی کاربر
- Username کد کاربری
- Pass رمز عبور
- Status حالت کاربر 1=online, 0=offline
- Type نوع کاربر، ۱- مدیر، ۰= کاربر معمولی

جدول شارژ

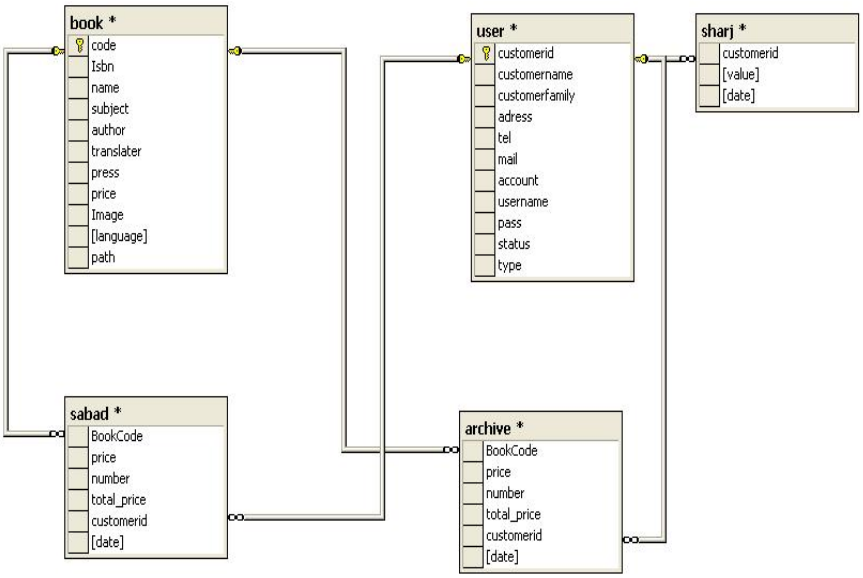
در این جدول تاریخ شارژ، میزان شارژ موجودی و شماره مشتری نگهداری می شود. این جدول بیشتر به داشتن آمارهای دقیق فروشگاه کمک می کند.

Column Name	Data Type	Length	Allow Nulls
customerid	int	4	
[value]	float	8	
[date]	datetime	8	

جدول ۴-۸ جدول اطلاعات شارژ

- Customerid شماره کاربر
- Value میزان شارژ
- Date تاریخ شارژ

نمودار رابطه ی جدول ها



بعد از مروری بر برنامه بهتر است به طراحی سایت و نحوی کدنویسی آن بپردازیم.

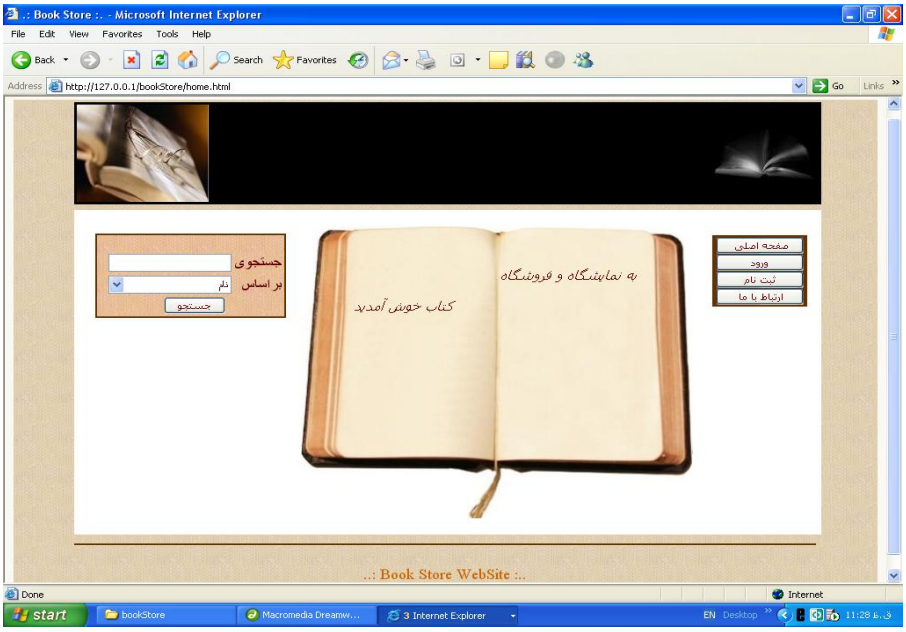
بهتر است صفحه اصلی سایت یک صفحه کم حجم باشد تا این صفحه به سرعت load شود و کاربر برای مشاهده سایت منتظر نشود و عملیات اصلی سایت را به صفحات دیگر منتقل کنیم همچنین برای آنکه صفحه اصلی سایت یک قالب مشخص داشته باشد، به مانند شکل ۱-۸ طراحی شده است. اما بهتر است در اول کار یک دسته بندی از عملیات موجود در این برنامه داشته باشیم تا بتوانیم بر روی طراحی صفحات بهتر عمل کنیم.

عملیات را نسبت به نوع کاربران تقسیم بندی می کنیم:

۱. عملیات مربوط به کاربر نوع میهمان
۲. عملیات مشترک برای کاربران
۳. عملیات کاربر عضو سایت
۴. عملیات مربوط به مدیر سایت

عملیات مربوط به کاربر نوع میهمان

این کاربر در صفحه اصلی سایت تنها می تواند کتابی را جستجو نماید تا اطلاعاتی از آن کتاب را مشاهده نماید و یا فایل الکترونیکی کتاب را دانلود نماید در نتیجه در صفحه اصلی سایت نیاز به قسمتی برای جستجوی کتاب داریم اما نتایج جستجو را در صفحه ی دیگر نمایش می دهیم.



شکل ۱-۸ نمای از طراحی صفحه home.html

صفحه home.html، صفحه اصلی سایت می باشد. کاربر می تواند در این صفحه مستقیما و بدون Login کردن به سایت به جستجوی کتاب مورد نظر خود بپردازد ولی امکان خرید کتاب را ندارد. در این صفحه امکان فرستادن پیام به مدیر سایت، ورود به صفحه ی شخصی و ثبت نام کردن کاربر نیز فراهم است.

بررسی کدهای صفحه home.html

زمانیکه که کاربر در صفحه اصلی سایت بر روی دکمه جستجو کلیک کند. تابع senddata() فراخوانی می شود و برای ارسال نوع جستجو (براساس نام یا موضوع) مقدار (value) خانه ی انتخاب شده را به یک input از نوع hidden با id='var1' انتساب می دهد از این input به عنوان یک متغیر کمکی استفاده می شود.

```
function senddata(){

     var t = document.form1.select1.options[document.form1
          .select1.selectedIndex].value;

     document.getElementById('var1').value=t;
}
```

صفحه home.html اطلاعات را از طریق <form> و بصورت متد GET به صفحه ViewBook.php برای جستجوی کتاب ارسال می کند. چون نوع دکمه جستجو Submit می باشد اطلاعات با کلیک شدن این دکمه ارسال می گردد. در این صفحه متغیر کمکی دیگر به نام h2 با id='var2' موجود می باشد و برای آن است که با مقدار دهی صفر به آن بتوانیم در صفحه ViewBook.php تابع newsearch را با پارامتر صفر به منظور شروع جستجو فراخوانی کرد.

```
<form method=GET action="ViewBook.php" name="form1">
```

بررسی کدهای صفحه ViewBook.php

این صفحه طراحی و کد آن تقریبا مانند صفحه Home.html می باشد با این تفاوت که برای جستجوی کتاب و نمایش نتایج کدهای php در آن نوشته شده است. با کلیک شدن دکمه جستجو تابع senddata فراخوانی می شود. بدنه این تابع همانند تابع senddata در صفحه home.html می باشد.

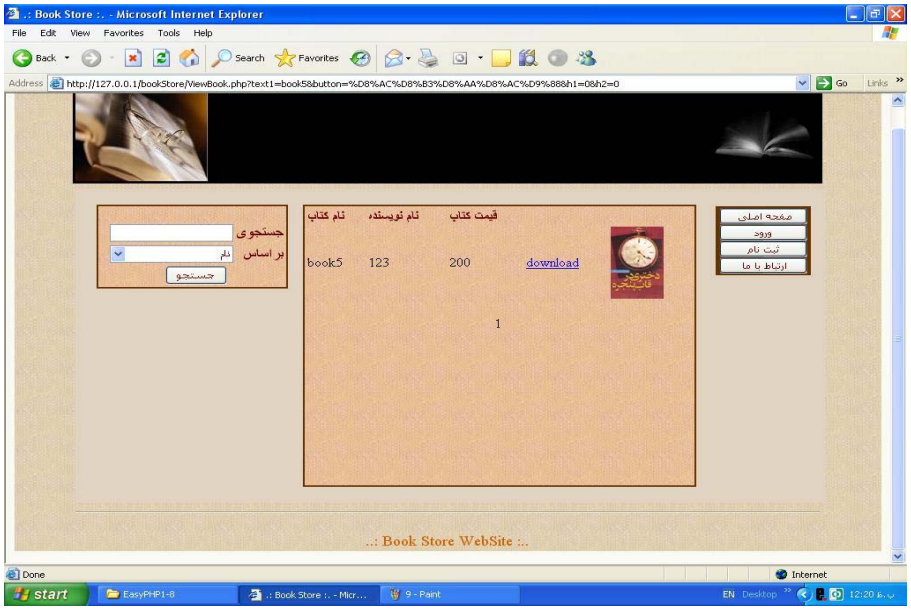
```
function senddata(){

     var t = document.form1.select1.options[document.form1.
          select1.selectedIndex].value;
     document.getElementById('var1').value=t;
}
```

نتایج جستجو در قالبی خاص تقسیم بندی (صفحه بندی) می شوند برای آن که بتوانیم با کلیک بر روی شماره صفحه اطلاعات آن صفحه نمایش داده شود، تابع newsearch(e) را با پارامتر شماره رکوردی که از آن رکورد به بعد می خواهیم یک تعداد رکورد را از مجموعه جواب بازیابی کنیم فراخوانی می کنیم. با مقداردهی e=0، از ابتدای مجموعه جواب رکوردها را بازیابی می کنیم.

```
function newsearch(e){
     document.getElementById('var2').value = e;
}
```

صفحه ViewBook.php خودش را به هنگام کلیک کردن دکمه جستجو فراخوانی می کند. برای ارسال اطلاعات از متد GET استفاده شده است.

```
<form method=GET action="ViewBook.php" name='form1'>
```



شکل۲-۸ نمای از طراحی صفحه viewBook.php

در صفحه ViewBook.php قسمت کدی به زبان php برای جستجوی کتاب قرار دارد.

```
<?php
```

این دستور برای چک کردن تهی (خالی) نبودن فیلد (این فیلد همان متنی است که کاربر می نویسد) جستجو است.

```
if (isset($_GET['text1']))
{
```

در این قسمت اطلاعاتی که توسط متد GET ارسال شده اند، دریافت می شوند.

```
$index = $_GET['h1'];
$name = $_GET['text1'];
```

$count شماره رکوردی است که باید اطلاعات از آن جا بازیابی شود. این شماره توسط یک متغیر با نام h2 و یا id='var2' در تابع newsearch() مقداردهی شده است.

```
$count =$_GET['h2'];
```

در این جا دستور اتصال به وب سرور محلی را می نویسیم.

```
$link = mysql_connect('localhost','root','');
```

مر حله بعد انتخاب بانک اطلاعاتی BookStore با دستور زیر می باشد.

```
mysql_select_db('BookStore');
```

بعد نوبت نوشتن دستور SQL برای جستجوی کتاب مورد نظر است اگر $index=0 باشد کاربر جستجو براساس نام را انتخاب کرده و اگر برابر یک جستجو براساس موضوع را انتخاب کرده است.

```
$q = "select * from book where ";
if ($index == '0')
```

```
     $q.= "'$name' = name ";
else if ($index == '1')
     $q.= "'$name' = subject ";
```

مرحله بعد اجرای دستور sql است.

```
$res = mysql_query($q,$link);
```

دستور زیر تعداد رکوردهای جواب دستور Sql را برمی گرداند.

```
$num = mysql_num_rows($res);
```

برای اینکه اگر نتایج جستجو زیاد باشد صفحه دارای نوار مرور عمودی نشود، از تکنیک صفحه بندی برای نمایش نتایج جستجو استفاده می کنیم. در این روش بعد از اینکه تعداد رکوردهای که می خواهید در یک صفحه نمایش دهید را مشخص کردید،(دستور mysql_num_rows($res) تعداد رکودهای جواب را به ما می دهد.) تعداد کل رکودهای نتایج جستجو را بر آن تقسیم کرده و تعداد صفحات را بدست می آورید آنگاه دستور پرس جوی را به آن تعداد رکورد محدود می کنید. $num تعداد کل رکوردهای جواب، $k تعداد صفحاتی که جستجو توانسته برگرداند.

```
$k = $num/3;
```

در این برنامه ما قصد داریم که هر صفحه را به تعداد ۳ رکورد جواب محدود کنیم. برای این کار لازم است در انتهای دستور select از عبارت limit $count,3 استفاده کنیم. این عبارت مشخص می کند از رکوردی به شماره $count در مجموعه جواب سه تا رکورد بازیابی شود. اگر در این عبارت پارامتر دوم ذکر نشود بدین معنی است که رکوردهای که بعد از رکورد به شماره $count قرار دارند (در مجموعه جواب) را نمایش دهد.

```
$q.=" limit $count,3";

$res = mysql_query($q,$link);
```

برای نمایش جواب ها یک جدول html ایجاد می کنیم. در ابتدا قسمت هدر جدول را می سازیم.

```
$row = mysql_fetch_array($res);
$str = '<table width=100% ><tr><th align=left
     style="color:#660000">نام کتاب</th><th align=left
     style="color:#660000">نام نویسنده</th><th align=left
     style="color:#660000">قیمت کتاب</th></tr>';
```

حال باید از این مجموعه جواب سه عنصری تک تک رکوردها برای نمایش، بازیابی شوند.

رکوردهای حاصل از جستجو را در سطر های یک جدول html قرار می دهیم تا آن جدول را نمایش دهیم.

```
while( $row != NULL)
{
     $p='number'.$row[0];
     $str.="<tr><td>$row[2]</td><td>$row[4]</td>
          <td>$row[7]</td><td><a href='$row[10]'>
          download</a></td><td><img src='$row[8]'></td></tr>";
```

```
    $row = mysql_fetch_array($res);
}
$str.='</table><br><p align=center>';
```

با استفاده از یک حلقه تکرار از صفر تا تعداد صفحات جواب ها ($k) یک شماره که همان شماره صفحه است ($m) را بعنوان یک لینک در نظر می گیریم و برای رویداد onclick لینک تابع newsearch را با پارامتر $i*3 فراخوانی می کنیم (برای صفحه اول باید رکوردهای از ۰ تا ۳ ولی $m برای صفحه اول یک است پس از $i استفاده می کنیم. برای صفحه اول $i=0 است پس $count=0 یعنی دستور بصورت limit 0,3 برای صفحه دوم $i=1 یعنی limit 3,3. چون هر صفحه دارای سه رکورد است شماره صفحه در عدد سه ضرب می شود تا آدرس شروع رکوردی که می خواهیم از آن آدرس رکوردهای جواب را بازیابی کنیم بدست آوریم).

```
for($i=0;$i < $k; $i++){
    $m = $i+1;
    $str.= "<a onclick='newsearch(($i)*3);'
        style='cursor:pointer'>$m</a> ";
}
$str.='</p>';
echo $str;
```

در این قسمت رشته حاوی جدول نتایج جستجو از سمت سرور به کلاینت با دستور echo $str برای نمایش، ارسال می شود.

```
mysql_close($link);
}
?>
```

اگر کاربر مهیمان بخواهد عضو سایت شود باید ثبت نام کند. عملیات ثبت نام را می توان در یک صفحه مجزا قرار داد. کاربران در صفحه ثبت نام، فرم ثبت نام را با اطلاعات شخصی خود پر می کنند که این اطلاعات در جدول customers ذخیره می شوند. هنگام ثبت اطلاعات کاربر به جدول، یک کد منحصر بفرد به کاربر تعلق می گیرد. این صفحه نیازی ندارد بصورت Ajax پیاده سازی شود.

بررسی کدهای صفحه RegisterUser.php

در این صفحه بعد از دریافت اطلاعات کاربری با فشردن دکمه ثبت صفحه register.php فراخوانی می شود. تابع check در این صفحه برای بررسی یکسان بودن رمز عبور وتکرار آن می باشد.

```
function check(){

  if(document.getElementById('pas1').value!=document
     .getElementById('pas2').value)
     document.getElementById('error').innerHTML = ' رمز و تکرار آن را
          یکسان وارد نماید';
  else
      document.getElementById('error').innerHTML = '';
}
```

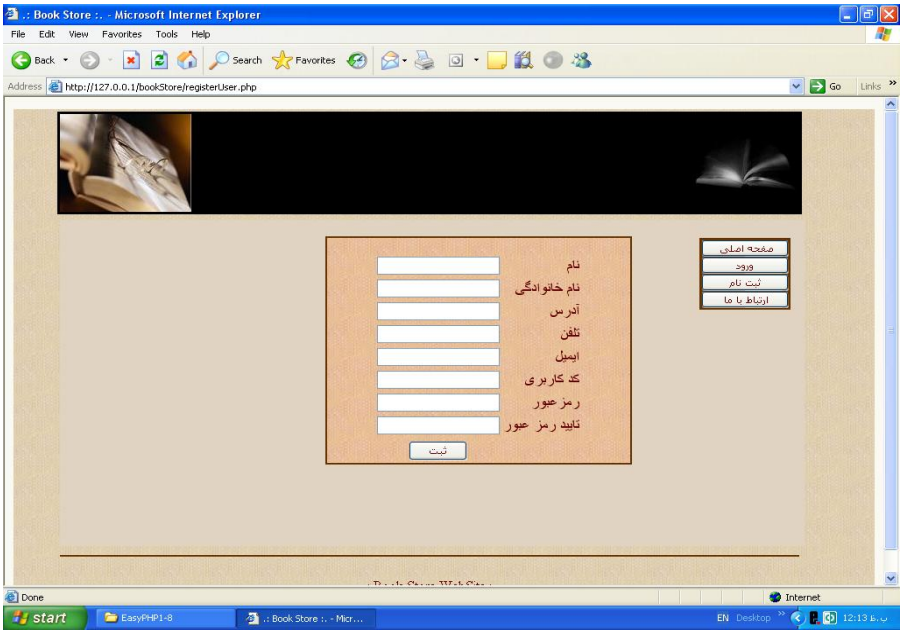
همان طور که مشخص است هنگام ثبت اطلاعات صفحه register.php با متد POST فراخوانی می شود.

```
<form action="register.php" method=POST >
```

بررسی کدهای صفحه Register.php

در این صفحه ابتدا مقادیر post شده (ارسال شده توسط متد POST) به این صفحه دریافت می شوند.

```
<?php
     $firstname = $_POST['firstname'];
     $lastname = $_POST['lastname'];
     $address = $_POST['address'];
     $tel = $_POST['tel'];
     $mail = $_POST['mail'];
     $username = $_POST['username'];
     $pass1 = $_POST['pass1'];

     $link = mysql_connect('localhost','root','');
     mysql_select_db('BookStore');
```



شکل ۳-۸ نمای از طراحی صفحه RegisterUser.php

یک دستور sql بررسی می کند که کد کاربری وارد شده در جدول موجود نباشد. (کد کاربری غیر تکراری)

```
$q = "select * from customers where Username = '$username'";
$r = mysql_query($q);
$t = mysql_num_rows($r);
```

اگر کد کاربری تکراری نباشد اطلاعات کاربر جدید به جدول اضافه می شود.

```
if ($t==0){
  $q = " insert into customers(customername ,customerfamily
      ,address ,tel,mail,account ,Username ,pass ,status
     ,type)values('$firstname','$lastname','$address','$tel'
     ,'$mail',50000,'$username','$pass1',0,0)";

  $res = mysql_query($q);
```

```
}
else
  $res=NULL;
```

نهایتا اگر کد کاربری تکراری بود پیغام unsuccessful را برمی گردانیم و در صورت صحت اطلاعات ورودی بعد از ثبت شدن کاربر، به صفحه login.html می رویم.

```
if($res )
     echo "<script> window.location='login.html'</script>";
else
     echo "<script> window.location='registerUser.php?msg
          =unsuccessful'</script>";
mysql_close($link);
?>
```

عملیات مشترک برای کاربران

یکی از این عملیات ها، login است می توانیم برای این قسمت یک صفحه مجزا طراحی کنیم و یا آن را بعنوان بخشی در صفحه اصلی سایت قرار دهیم. در این برنامه یک صفحه مجزا برای login در نظرگرفته شده است. شاید در نظر گرفتن یک صفحه مجزا کمی بهتر باشد زیرا با توجه به داشتن یک صفحه تقریبا خالی می توانید خطاهای زمان ورود به سایت را در همان صفحه نمایش دهید و نیاز نباشد صفحه ی دیگری را بارگذاری کنید. یک مزیت دیگر که صفحه ورود جداگانه طراحی شود این است که در صفحه اصلی و صفحه ورود، فضای خالی بیشتری برای قرار گرفتن گزینه های دیگر (مانند تبلیغات) موجود می باشد.

بعد از ثبت اطلاعات، کاربر می تواند به صفحه شخصی خود در سایت وارد شود. در صفحه ورود به سایت کد کاربری و رمز عبور را وارد می کند. چک شدن صحت اطلاعات کاربری با فراخوانی صفحه login.php انجام می گیرد. ارسال اطلاعات کاربری به سمت سرور با Ajax پیاده سازی شده است.

بررسی کدهای صفحه login.html

در این برنامه نیز از کلاسی که برای Ajax در فصل قبل نوشته ایم استفاده می کنیم.

```
<script type="text/javascript"
src="Java_script/Asynchronous.js"></script>
```

این صفحه برای دریافت اطلاعات کدکاربری و رمز عبور برای ورود به سایت می باشد. بعد از فشردن دکمه ' ورود ' تابع login فراخوانی می شود و اطلاعات را از textBox های مر بوطه گرفته و به صفحه login.php بصورت GET ارسال می کند.

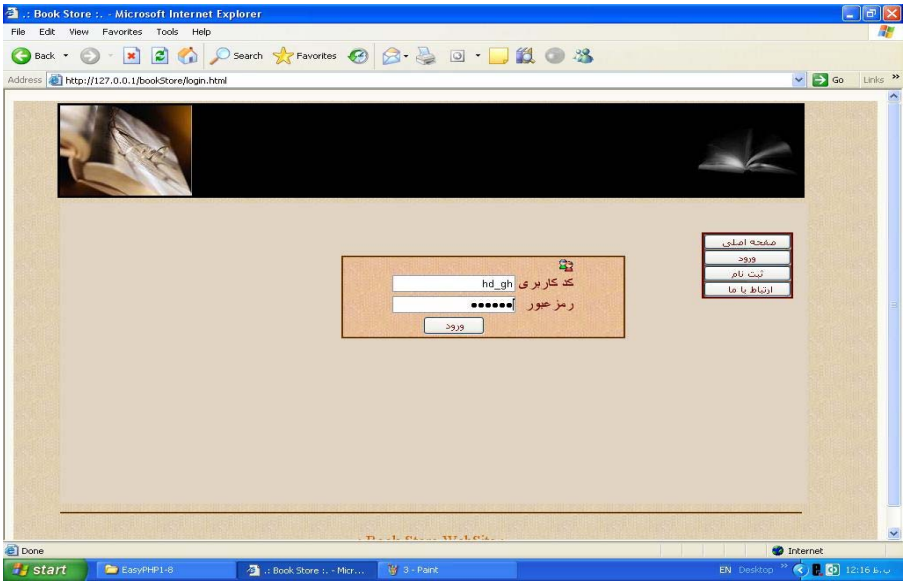
در تابع login() اگر مشخصات کاربر صحیح باشد و کاربر عضو سایت باشد از سرور مقدار 200 بعنوان پاسخ ارسال می شودکه این پاسخ در responseText قرار می گیرد، اگر کاربر، مدیر سایت باشد مقدار 500 ارسال می شود و در غیر اینصورت یک پیغام خطا در صفحه نمایش می دهیم.

```
function login(){
asynchronous = new Asynchronous();
asynchronous.complete = function(status, statusText
     ,responseText,responseXML)
{
```

```
   if (responseText=='200')

      location.href("personal.html");
else if(responseText=='500')

      location.href("admin.html");
else
            document.getElementById('div1').innerHTML=
            'لطفا کد کاربری ورمز عبور را صحیح وارد نمایید';
      };
var txt = 'user='+document.getElementById('user').value
      +'&pass='+document.getElementById('pass').value+'';
```

به این علت که متد ارسال داده ها به سرور بصورت GET است باید داده های ارسالی را به رشته فراخوانی صفحه اضافه شود.

```
   asynchronous.call('login.php?'+txt,'','GET');
}
```



شکل ۴-۸ نمای از طراحی صفحه login.html

بررسی کدهای صفحه Login.php

در این صفحه بعد از جستجوی کاربر و اطمینان از صحت اطلاعات کاربری، تغییر وضعیت کاربر از حالت offline به online صورت می گیرد و نوع کاربر (مدیر- عضو سایت) با دستور echo به کلاینت ارسال می شود.

```
<?php
      $user = $_GET['user'];
      $pass = $_GET['pass'];
      $link = mysql_connect('localhost','root','');
      mysql_select_db('BookStore');
```

با این دستور sql کاربرانی را جستجو می کنیم که offline هستند یعنیstatus=0. در صورتی که status=1 باشد کاربر قبلا وارد سایت شده و خارج نشده و دسترسی با نام کاربریی که هم اکنون در سایت است غیرمجاز است.

```
$q = "select * from customers where '$user'=Username and
     '$pass'=pass  and status=0 ";
$res = mysql_query($q,$link);
$row = mysql_fetch_array($res);
//
if(mysql_num_rows($res) == 1){

     session_start();
```

شماره کاربر را در session برای مراجعات بعدی ذخیره می کنیم.

```
     $_SESSION['id'] = $row['customerid'];
```

زمانی که کاربر به سایت وارد می شود فیلد status آن یک می شود.

```
     $q = " update  customers set status=1 where customerid =
           $row[0] ";
     $r =  mysql_query($q,$link);

     if($row['type']=='1'){
          echo '500';
     }
```

هنگام ورود به سایت تمام رکوردهای هم شماره با این کاربر در صورت وجود از سبد خرید حذف می شود. جدول سبد خرید یک جدول موقتی برای نگهداری اطلاعات خرید است. اما علت حذف رکوردها در هنگام ورود، برای آن است که امکان دارد کاربری بدون قطعی کردن خرید از سایت خارج شود در نتیجه در مراجعه بعدی به سایت نیازی به اقلام خرید قبلی خود ندارد.

```
     else{
      $cmd = "delete  from sabad where customerid = $row[0] ";
      $r = mysql_query($cmd,$link);
      echo '200';
     }
     }
     else echo '50';
     mysql_close($link);
?>
```

عملیات کاربر عضو سایت

دسته دیگر عملیات مر بوط به کاربر عضو سایت می باشد. این کاربر می خواهد بعد از جستجوی کتاب و مشاهده اطلاعات کتاب آن را خریداری کند، بتواند کتابهای خریداری شده را در سبد خرید مشاهده نماید، از میزان موجودی خود آگاه شود و لیست خرید های قبلی خود را مشاهده کند. ما این عملیات را در یک صفحه قرار دادیم. طراحی این صفحه را می توانید در شکل ۵-۸ مشاهده نماید.

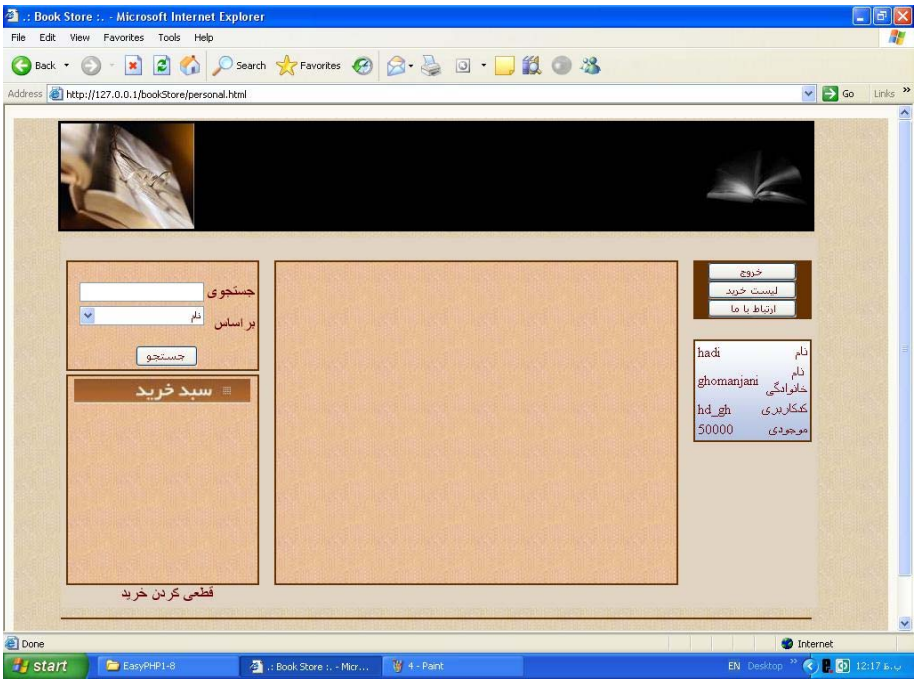
این صفحه، صفحه ی شخصی کاربر است که بعد از ورود به سایت مشاهده می شود در سمت راست صفحه اطلاعات شخصی کاربر بهمراه میزان موجودیش نشان داده می شود و این صفحه امکاناتی از قبیل جستجوی کتاب، خرید کتاب، مشاهده لیست خرید قبلی، مشاهده سبد خرید، حذف از سبد خرید، ارسال پیام به مدیر سایت را فراهم می کند. هنگامی که صحت اطلاعات در صفحه login چک شود این صفحه load می شود.

بررسی قسمت اول کدهای صفحه Personal.HTML

تابع show برای نمایش اطلاعات کاربر در کادری در سمت راست سایت هنگام ورود به سایت فراخوانی می شود. برای اینکه بتوانیم این تابع در زمان بار شدن صفحه اجرا شود فراخوانی آن را در خاصیت onload تگ <body> قرار می دهیم.

```
<body bgcolor="#fbf9f3"  onLoad="show();">
```

تابع show صفحه showInfo.php را فراخوانی می کند و اطلاعات دریافتی از این فراخوانی را که در responseText قرار دارد در یک div با id='div1' نمایش می دهد.



شکل ۵-۸ نمای از طراحی صفحه personal.html

```
function show(){
      asynchronous = new Asynchronous();
      asynchronous.complete = function(status, statusText,
                 responseText, responseXML){

     document.getElementById('div1').innerHTML=responseText;

      };
      asynchronous.call('showInfo.php','','GET');
 }
```

بررسی کدهای صفحهShowInfo.php

بوسیله کدهای این صفحه هنگام ورود به سایت اطلاعات شخصی فرد از جدول خوانده شده و برای کلاینت ارسال می شود.

```
<?php
     session_start();
     if (isset($_SESSION['id']))
     {
          $u=$_SESSION['id'];
          $link = mysql_connect('localhost','root','');
          mysql_select_db('BookStore');
          $q ="select * from customers where $u=customerid ";
          $res = mysql_query($q);
          $row = mysql_fetch_array($res);
          if($row)
          {
```

ابتدا رشته ای برای ارسال اطلاعات فردی کاربر در نظر می گیریم، این رشته را با یک جدول htmlای پر کرده و ارسال می کنیم.

```
            $t = "<table><tr><td style='color:#660000'>
               $row[1] </td><td style='color:#660000'
               align=right> نام </td></tr>";
            $t.="<tr><td style='color:#660000'>$row[2]</td>
               <td style='color:#660000' align=right>
               نام خانوادگی </td></tr><tr><td
               style='color:#660000'>$row[7]</td><td
               style='color:#660000'  align=right>
               کدکاربری </td></tr><tr><td
               style='color:#660000'>$row[6]</td>
               <td style='color:#660000'  align=right>موجودی
               </td></tr></table>";
             echo $t;
```

ارسال اطلاعات فردی کاربر با دستور echo.

```
          }
          mysql_close($link);
     }
     else
          echo '-50';
?>
```

بررسی قسمت دوم کدهای صفحه Personal.HTML

زمانیکه کاربر دکمه جستجو را کلیک کند تابع searchBook اجرا می شود. این تابع صفحه searchUserBook.php را فراخوانی می کند و اطلاعات مورد نیاز برای جستجو را از طریق متد GET به آن صفحه ارسال می کند. این اطلاعات با یک رشته (در این کد رشته msg) به URL صفحه فراخوانی شونده

الحاق می شود. پارامتر Counter از تابع searchBook شماره رکوردی است که بعنوان آدرس شروع رکوردهای جواب، برای بازیابی استفاده می شود. نتایج جستجو را برای مشاهده بهتر صفحه بندی می کنیم. رکوردهای جواب برای نمایش در قسمت مرکزی صفحه در یک div با id='div2' نمایش داده می شوند.

```
 function searchBook(counter){
    asynchronous = new Asynchronous();
    asynchronous.complete = function(status, statusText,
           responseText, responseXML){

     document.getElementById('div2').innerHTML=responseText;

 };

 var t=document.form1.select1.options[document.form1.select1
          .selectedIndex].value;
 var searchTxt = document.getElementById('txt1').value;
 var msg = 'number=' +t+'&txt1=' +searchTxt+'&count=
          '+counter+'';

 asynchronous.call( 'searchBookUser.php?'+msg,'','GET');
}
```

بررسی کدهای صفحه SearchBookUser.php

این صفحه عملیات جستجوی کتاب در صفحه شخصی کاربر را انجام می دهد.

```
<?php
     if (isset($_GET['txt1']))
     {
          $index = $_GET['number'];
          $name = $_GET['txt1'];
          $count =$_GET['count'];
          $link = mysql_connect('localhost','root','');
          mysql_select_db('BookStore');
          $q = "select * from book where ";
```

اطلاعات فیلدهای جستجو دریافت می شود و در دستور sql به کار می روند.

```
          if ($index == '0')
               $q.= "'$name' = name ";
          else if ($index == '1')
               $q.= "'$name' = subject ";

          $res = mysql_query($q,$link);
          $num = mysql_num_rows($res);
          $k = $num/3;
```

در هر صفحه ۳ رکورد برای نمایش انتخاب می شوند.

```
          $q.=" limit $count,3";
          $res = mysql_query($q,$link);
```

```
        $row = mysql_fetch_array($res);

        $str = '<table width=100% ><tr><th align=left
              style="color:#660000">نام کتاب</th><th
              align=left style="color:#660000">
              نام نویسنده</th><th align=left
              style="color:#660000">قیمت کتاب</th></tr>';

        while( $row != NULL)
        {
          $p='number'.$row[0];
          $str.="<tr><td> $row[2]</td><td>$row[4]</td>
             <td>$row[7]</td><td><a href='$row[10]'>
              download</a></td><td><img src='$row[8]'>
             </td></tr><tr><td><img src='picture/buy.gif'
              onclick='buy($row[0]);'
              style='cursor:pointer' ></td><td>
             <input  style='width:20px' type=text id='$p'
              value=1></td><td>تعداد خرید</td></tr>";
          $row = mysql_fetch_array($res);
        }
        $str.='</table><br><p align=center>';
        for($i=0;$i < $k; $i++){
           $m = $i+1;
           $str.= "<a onclick='searchBook(($i)*3);'
                  style='cursor:pointer'>$m</a> ";
        }
        $str.='</p>';
        echo $str;
        mysql_close($link);
    }
?>
```

بررسی قسمت سوم کدهای صفحه Personal.HTML

تابع showSabad محتوای سبد خرید را با استفاده از فراخوانی صفحه sabad.php بصورت یک جدول در یک خانه از جدولی با id='row2'، در صفحه نمایش می دهد. پارامتر e شماره شروع رکورد برای بازیابی رکوردها از مجموعه جواب، برای نمایش در صفحات متوالی سبد خرید است. در صفحه اول e=0 است (بازیابی رکوردها از اول مجموعه جواب). این پارامتر به صفحه sabad.php ارسال می شود.

```
function showSabad(e){

      asynchronous = new Asynchronous();
      asynchronous.complete = function(status, statusText,
                responseText, responseXML){

     document.getElementById('row2').innerHTML=responseText;
     };
```

```
    var msg = 'num=' +e+'';

    asynchronous.call( 'sabad.php?'+msg,'','GET');
}
```

بررسی کدهای صفحه Sabad.php

این صفحه برای بازیابی اطلاعات خرید از سبد خرید برای یک کاربر خاص است. اطلاعات کاربر از session دریافت می شود.

```
<?php
    $count  = $_GET['num'];
    session_start();
    $u = $_SESSION['id'];
    $link = mysql_connect('localhost','root','');
    mysql_select_db('BookStore');
    //
```

تمامی رکوردهای که برای این کاربر می باشند از جدول سبد خرید بازیابی می شوند.

```
    $q = "select * from sabad where customerid = $u";
    $rs = mysql_query($q,$link);
    $num = mysql_num_rows($rs);
    $k = $num/5;
```

این رکوردها بصورت ۵ تا ۵تا صفحه بندی شده و نمایش داده می شوند.

```
    $q = "select * from sabad where customerid = $u
          limit $count,5";
    $res = mysql_query($q,$link);
    $row = mysql_fetch_array($res);
    //
    $str = "<table width=100%><tr><th ></th>
         <th style='color:#660000 'align=left>قیمت</th>

         <th style='color:#660000 'align=left>نام کتاب</th>

         </tr>";
    while($row){
```

اطلاعات کتاب، مانند قیمت کتاب که لازم است از جدول کتاب خوانده می شود.

```
         $q = "select * from book where code= $row[0]";
         $rt = mysql_query($q,$link);
         $r = mysql_fetch_array($rt);
         //
         $str.="<tr><td onclick='del($row[0]);'
              style='cursor:pointer;color:#0000ff'>حذف</td>

              <td>$row[3]</td><td>$r[2]</td></tr>";
         $row = mysql_fetch_array($res);
    }
    $str .= '</table><br><p align=center>';
    for($i=0;$i < $k; $i++){
         $m = $i+1;
         $str.= "<a onclick='showSabad(($i)*5);'
              style='cursor:pointer'>$m</a> ";
```

```
    }
    $str.='</p>';
    echo $str;
    mysql_close($link);
?>
```

بررسی قسمت چهارم کدهای صفحه Personal.HTML

هنگامی که دکمه خرید کتاب کلیک شود تابع buy فراخوانی می شود و اطلاعات کد کتاب را برای اضافه شدن به سبد خرید به صفحه BuyBook.php ارسال می کند. پارامتر e در این تابع، کد کتابی است که قصد خرید آن را داریم. این پارامتر و تعداد خرید بوسیله num به صفحه BuyBook.php ارسال می شود. برای نمایش اطلاعات خرید در سبد خرید بعد از خرید کتاب تابع showSabad(0) فراخوانی می شود.

```
function buy(e){

    asynchronous = new Asynchronous();
    asynchronous.complete = function(status, statusText,
              responseText, responseXML){
          showSabad(0);
    };
    var id='number'+e;
    var num = document.getElementById(id).value;
    var msg = 'code=' +e+'&num=' + num+'';
    asynchronous.call( 'buyBook.php?'+msg,'','GET');
}
```

بررسی کدهای صفحه BuyBook.php

```
<?php
```

اطلاعات کد کتاب و تعداد خرید از صفحه personal.html را دریافت می شود.

```
    $code = $_GET['code'];
    $num  = $_GET['num'];
    //
    session_start();
```

شماره مشتری از داخل متغیر SESSION خوانده می شود.

```
    $u = $_SESSION['id'];
    $link = mysql_connect('localhost','root','');
    mysql_select_db('BookStore');
    //
```

تاریخ خرید از سیستم خوانده می شود.

```
    $date = date("Y-m-d H:i:s");
```

اطلاعات کتاب، مانند قیمت که لازم است از جدول کتاب خوانده می شود.

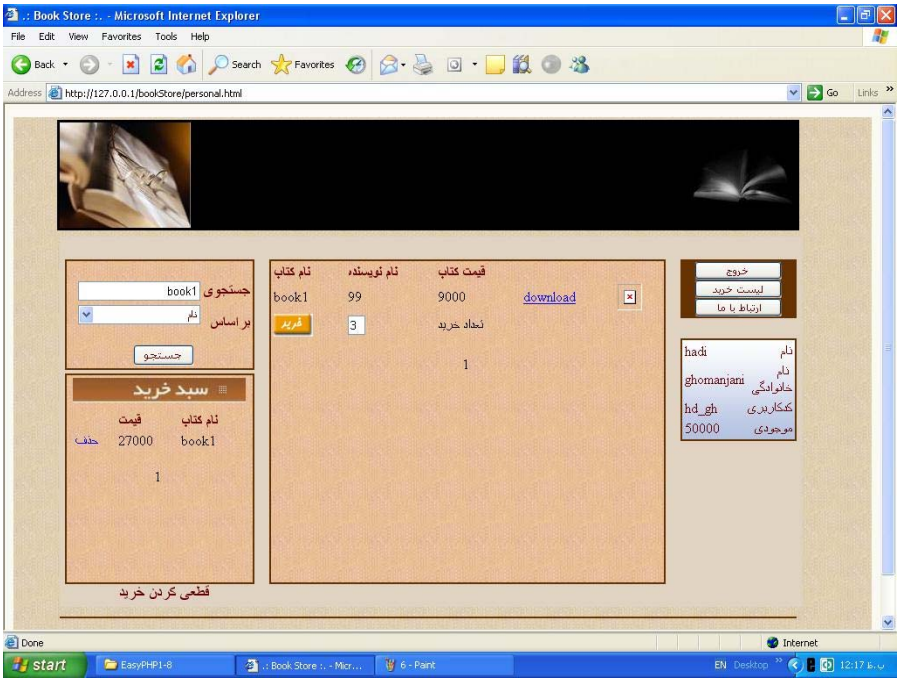
```
    $q = "select * from book where code= $code";
    //
    $res = mysql_query($q,$link);
    $row = mysql_fetch_array($res);
```

قیمت کتاب را در تعداد خرید ضرب می کنیم تا جمع کل حاصل شود.

```
$totalp = $row[7]*$num;
//
```

به سبد خرید رکوردی را حاوی کد کتاب، قیمت کتاب، تعداد خرید، جمع کل، و تاریخ وساعت خرید اضافه می کنیم.

```
$q = "insert into sabad values
      ($row[0],$row[7],$num,$totalp,$u,'$date')";
$r = mysql_query($q,$link);
echo '200';
mysql_close($link);
?>
```



شکل ۶-۸ نمای از سبد خرید بعد از خرید

بررسی قسمت پنجم کدهای صفحه Personal.HTML

هنگامی که یک قلم خرید را از سبد خرید حذف کنیم تابع del فراخوانی می شود. این تابع کد کتاب را برای حذف به صفحه ی DelSabad.php ارسال می کند. پارامتر e کد کتابی است که قصد حذف آن را از سبد داریم. برای نمایش دوباره سبد بعد حذف تابع showSabad(0) فراخوانی می شود.

```
function del(e){

      asynchronous = new Asynchronous();
      asynchronous.complete = function(status, statusText,
                responseText, responseXML){

            showSabad(0);
       };
      var msg = 'code=' +e+'';
      asynchronous.call( 'DelSabad.php?'+msg,'','GET');
}
```

بررسی کدهای صفحه DelSabad.php

```
<?php
     session_start();
```

در این قسمت کد کتابی که می خواهد از سبد خرید حذف شود دریافت می شود و شماره کاربر را نیز از داخل متغیر SESSION خوانده می شود.

```
     $code = $_GET['code'];
     $u = $_SESSION['id'];
     //
     $link = mysql_connect('localhost','root','');
     mysql_select_db('BookStore');
```

از جدول سبد خرید، رکورد مذکور حذف می شود.

```
     $q = "delete  from sabad where customerid = $u and
           bookcode = $code ";
     $rs = mysql_query($q,$link);
     //
     echo '200';
     mysql_close($link);
?>
```

بررسی قسمت ششم کدهای صفحه Personal.HTML

چون جدول سبد خرید یک جدول موقتی است هنگام قطعی کردن خرید باید اطلاعات سبد خرید به جدول آرشیو خرید منتقل شود. این عمل با فراخوانی تابع sabt که منجر به اجرای صفحه finalBuy.php در سرور می شود، انجام می گیرد.

```
function sabt(){

   asynchronous = new Asynchronous();
   asynchronous.complete = function(status, statusText,
                  responseText, responseXML){
  if (responseText == '200'){
     show();
     document.getElementById('row2').innerHTML='';
     document.getElementById('link1').innerHTML='قطعی کردن خرید';
   }
   else
      document.getElementById('link1').innerHTML=
           'موجودی به اندازه کافی موجود نیست<br>قطعی کردن خرید';
    };
    asynchronous.call( 'finalBuy.php','','GET');
    show();
}
```

بررسی کدهای صفحه finalBuy.php

هنگام قطعی کردن خرید این صفحه اجرا می شود و رکوردهای سبد خرید برای یک کاربر خوانده شده و به جدول Archive اضافه می گردد. نهایتا رکوردها از جدول سبد خرید حذف می شوند. همچنین به میزان مبلغ کل خرید از حساب مشتری کاسته می شود.

```
<?php
     session_start();
     $u = $_SESSION['id'];
     $link = mysql_connect('localhost','root','');
     mysql_select_db('BookStore');
     $q = "select * from customers where customerid = $u ";
     $res = mysql_query($q,$link);
     $user = mysql_fetch_array($res);
```

با این دستور sql قیمت کل خرید محاسبه می شود.

```
     $q = "select sum(total_price) from sabad where
           customerid = $u";
     $t = mysql_query($q,$link);
     $row = mysql_fetch_array($t);
     $account = $user['account'];
     //
     if ($user['account'] >= $row[0])
     {
          $q = "insert into archive select * from sabad
                where customerid = $u";
          $res = mysql_query($q,$link);
          //
```

از موجودی کاربر به میزان قیمت کل خرید کاسته می شود.

```
          $account -=$row[0];
          $q = "update customers set account = $account
                where customerid = $u";
          $res = mysql_query($q,$link);
          //
```

با یک دستور اطلاعات خرید از سبد خرید حذف می شود.

```
          $q = "delete from sabad where customerid = $u";
          $res = mysql_query($q,$link);
          echo '200';
     }
     else
          echo'-50';
     mysql_close($link);
?>
```

بررسی قسمت هفتم کدهای صفحه Personal.HTML

اگر کاربر دکمه لیست خرید را کلیک کند تابع list اجرا می شود. این تابع لیست تمام خرید های قبلی یک کاربر را با فراخوانی صفحه buyList.php نشان می دهد. پارامتر e شماره شروع رکوردی برای بازیابی از مجموعه جواب لیست خرید ها می باشد.

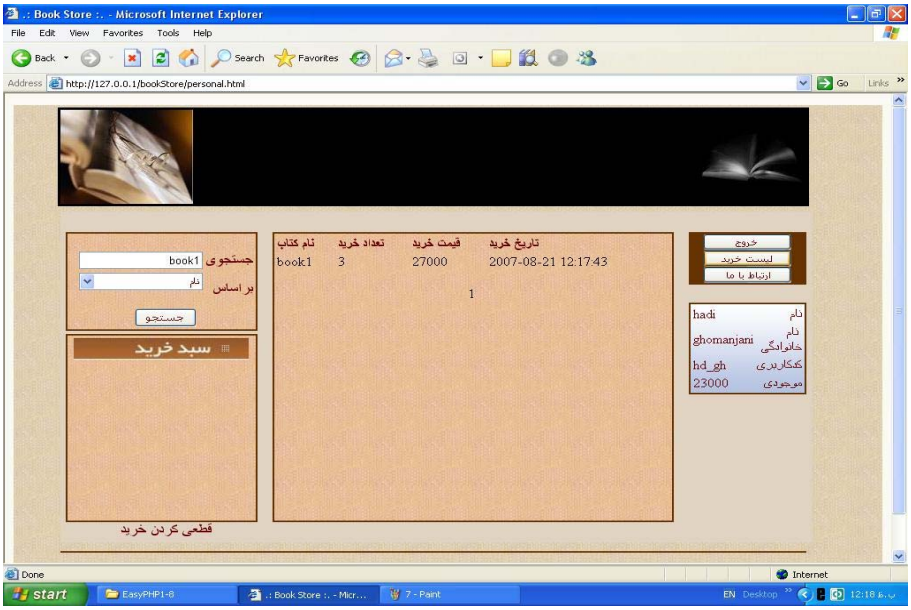
```
function list(e){

   asynchronous = new Asynchronous();
   asynchronous.complete = function(status, statusText,
              responseText, responseXML){

     document.getElementById('div2').innerHTML=responseText;
   };

   var msg = 'count=' +e+'';
   asynchronous.call( 'buyList.php?'+msg,'','GET');
}
```

بررسی کدهای صفحه BuyList.php

زمانیکه فرد می خواهد اطلاعات خریدهای قبلی را مشاهده کند این صفحه اجرا می شود. شماره کاربر از داخل متغیر SESSION خوانده می شود و رکوردهای متناظر با آن از جدول ARCHIVE نمایش داده می شود.

```
<?php
     $count = $_GET['count'];
     session_start();
     $u = $_SESSION['id'];
     $link = mysql_connect('localhost','root','');
     mysql_select_db('BookStore');
     //
     $q = "select * from archive where customerid=$u ";
     $r = mysql_query($q,$link);
     $num = mysql_num_rows($r);
     $k = $num/5;

     $q.="limit $count,5";
     $res = mysql_query($q,$link);
     $row = mysql_fetch_array($res);

     $str = '<table width=100% ><tr >
          <th align=left style="color:#660000">نام کتاب</th>
          <th align=left style="color:#660000">تعداد خرید</th>
          <th align=left style="color:#660000">قیمت خرید</th>
          <th align=left style="color:#660000">تاریخ خرید</th>
          </tr>';

     while( $row != NULL)
```

```
        {
            $q="select * from book where code =$row[0]";
            $rs = mysql_query($q,$link);
            $r = mysql_fetch_array($rs);
            $str.="<tr><td> $r[2]</td><td>$row[2]</td>
                <td>$row[3]</td><td>$row[5]</td></tr>";
            $row = mysql_fetch_array($res);
        }
        $str.='</table><br><p align=center>';
        for($i=0;$i < $k; $i++){
            $m = $i+1;
            $str.= "<a onclick='list(($i)*5);'
                  style='cursor:pointer'>$m</a> ";
        }
        $str.='</p>';
        echo $str;
        mysql_close($link);
?>
```



شکل ۷-۸ لیست خرید قبلی کاربران

بررسی کدهای صفحه logout.php

در این صفحه ابتدا با خواندن اطلاعات کاربر از متغیر Session حالت کاربر را در جدول به offline تبدیل می کند.

```
<?php
        session_start();
        $u=$_SESSION['id'];
        $link = mysql_connect('localhost','root','');
        mysql_select_db('BookStore');
        $q =  " update  customers set status=0
            where customerid = $u ";
        $res = mysql_query($q,$link);
```

```
     session_destroy();
     echo "<script> window.location='login.html'</script>";
     mysql_close($link);
?>
```

عملیات مربوط به مدیر سایت

شکل ۸-۸ صفحه ی مدیر سایت می باشد که امکاناتی از قبیل اضافه کردن کتاب، حذف کتاب، ویرایش کتاب، upload کردن فایل، مشاهده لیست خرید کاربران، مشاهده تمامی کاربران، شارژ موجودی کاربران، مشاهده لیست شارژ و جستجوی کتاب را داراست.

بررسی کدهای صفحه Admin.html

```
<script type="text/javascript"
src="Java_script/Asynchronous.js"></script>
```

تابع searchBook جستجوی کتاب را در صفحه Admin با فراخوانی صفحه searchBookAd.php بصورت Ajax بر عهده دارد.

```
function searchBook(counter){
     asynchronous = new Asynchronous();
     asynchronous.complete = function(status, statusText,
               responseText, responseXML){

     document.getElementById('div2').innerHTML=responseText;
     };
     var t=document.form1.select1.options[document.form1.
          select1.selectedIndex].value;
     var searchTxt = document.getElementById('txt1').value;
     var msg = 'number=' +t+'&txt1=' +
          searchTxt+'&count='+counter+'';
     asynchronous.call( 'searchBookAd.php?'+msg,'','GET');
}
```

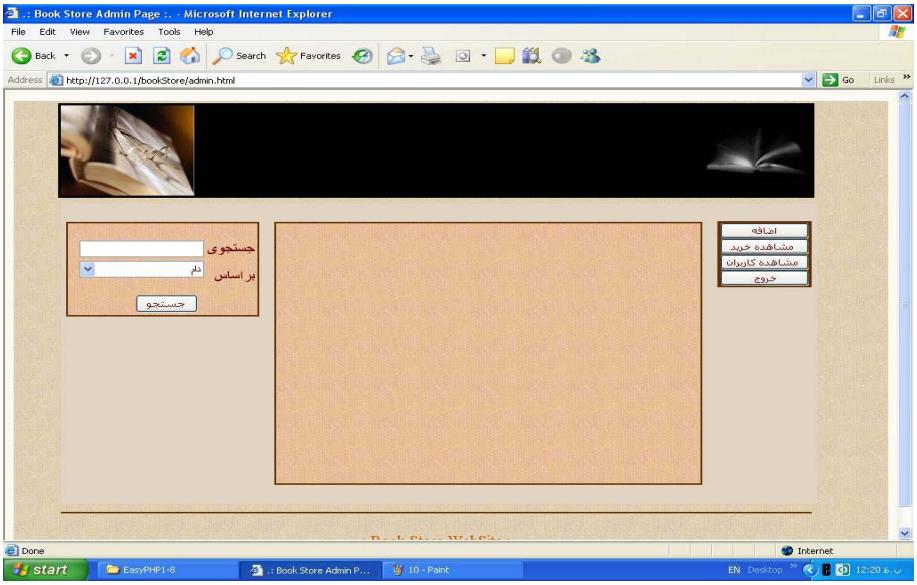
این تابع برای مشاهده اطلاعات خرید یک کاربر صفحه searchUser.php را فراخوانی می کند.

```
function InfoCustomer(e){
     ajax = new Asynchronous();
     ajax.complete = function(status, statusText,
          responseText, responseXML){

     document.getElementById('div2').innerHTML=responseText;
     };
     var user=document.getElementById('txt2').value;
     var msg ='user='+user+'&count='+e;
     ajax.call('searchuser.php?'+msg,'','GET');
};
```

تابع Del زمانی فراخوانی می شود که یک کتاب را می خواهیم حذف کنیم. این تابع صفحه deleteBook.php را فراخوانی می کند.

```
function Del(e){
     ajax = new Asynchronous();
```

```
     ajax.complete = function(status, statusText,
               responseText, responseXML){
      searchBook(0);
     };

     var msg ='code='+e;
     ajax.call('deleteBook.php?'+msg,'','GET');
};
```



شکل ۸–۸ نمای از طراحی صفحه Admin.html

تابع show برای نمایش قسمت شارژ موجودی برای کاربر می باشد.

```
function show(){
if (document.getElementById('div4').style.display=='block' ){
     document.getElementById('div2').innerHTML='';
     document.getElementById('div4').style.display='none';
}
else
     document.getElementById('div4').style.display='block';
}


function sharj(i){

     if(document.getElementById(i).style.display=='none'){
          document.getElementById(i).style.display='block';
     }
     else
          document.getElementById(i).style.display='none';
}
```

تابع getData مقادیر برای شارژ را دریافت و به صفحه sharj.php ارسال می کند.

```
function getData(te){
     ajax = new Asynchronous();
```

```
     ajax.complete = function(status, statusText,
               responseText, responseXML){
       showUser(0);
     };
     var f=te+'a';
     var acc=document.getElementById(f).value * 1;
     var msg ='code='+te+'&account='+acc;
     ajax.call('sharj.php?'+msg,'','GET');
}
```

تابع showUser برای نمایش کلیه کاربران عضو سایت می باشدکه صفحه ShowUser.php را فراخوانی می کند.

```
function showUser(e){
     ajax = new Asynchronous();
     ajax.complete = function(status, statusText,
               responseText, responseXML){

     document.getElementById('div2').innerHTML=responseText;
     };
     var msg ='u='+e;
     ajax.call('showUser.php?'+msg,'','GET');
}
```

تابع listSharj لیست شارژ ها و تاریخ آن ها را برای هر کاربر نمایش می دهد.

```
function listSharj(e){
     if(document.getElementById('div8').style.display=='none')
        document.getElementById('div8').style.display='block';
     else
         document.getElementById('div8').style.display='none';
     ajax = new Asynchronous();
     ajax.complete = function(status, statusText,
               responseText, responseXML){
       document.getElementById('div8').innerHTML=responseText;
     };
     var msg ='u='+e;
     ajax.call('listSharj.php?'+msg,'','GET');

}
```

بررسی کدهای صفحه searchUser.php

جستجوی یک کاربر و نمایش اطلاعات آن کاربر توسط این صفحه انجام می گیرد.

```
<?php
     $u=$_GET['user'];
     if($u){
          $count=$_GET['count'];
          session_start();
          $link=mysql_connect('localhost','root','');
          mysql_select_db('bookstore');
          $q="select * from customers where username='$u'";
          $rs = mysql_query($q,$link);
```

```
$t=mysql_fetch_array($rs);
if($t){
     $id=$t[0];
     //
     $str="<p align=center ><table
             width=100%><tr>
             <td onclick='listSharj($id);'
             style='color=#660000;font-size:18px;
             cursor:pointer'>لیست شارژها</td>
             <td >$t[6]</td><td style='color
             =#660000;font-size:18px'> موجودی
             </td><td >$t[7]</td><td style=
             'color=#660000;font-size:18px'>
             کدکاربری </td><td >$t[1]</td>
             <td style='color=#660000;
             font-size:18px'>نام</td></tr>
             </table></p><div id='div8'
             style='display:none'></div>";
          //
```

دستورsql زیر برای بازیابی اطلاعات کل خرید یک کاربر با کد $id از جدول archive می باشد.

```
     $q="select * from archive where ustomerid=$id";
     $res=mysql_query($q,$link);
     $num = mysql_num_rows($res);
     $k = $num/5;
     $row=mysql_fetch_array($res);
     $q=" select * from archive where
          customerid=$id limit $count,5";
     $r=mysql_query($q);
     $rs=mysql_fetch_array($r);
     //
     $str.="<table width=100%><tr><th
               style='color:#660000 '
               align=left>نام کتاب</th><th
               style='color:#660000 '
               align=left>تعداد کتاب</th><th
               style='color:#660000 '
               align=left>قیمت کتاب</th><th
               style='color:#660000 '
               align=left>تاریخ خرید</th></tr>";
     while ($rs!=NULL)
     {
```

دستورsql زیر برای بازیابی اطلاعات کتاب های خریداری شده است.

```
          $q1= "select * from book where
                    code=$rs[0]";
```

```
                    $t=mysql_query($q1);
                    $rt=mysql_fetch_array($t);
                    $str.="<tr><td>$rt[2]</td><td>
                         $rs[2]</td><td>$rs[3]</td>
                         <td>$rs[5]</td></tr>";
                    $rs=mysql_fetch_array($r);
               }
               $str .= '</table><br><p align=center>';
               for($i=0;$i < $k; $i++){
                 $m = $i+1;
                 $str.="<a
                     onclick='InfoCustomer(($i)*5);'
                    style='cursor:pointer'>$m</a> ";
               }
               $str.='</p>';
               echo $str;
          }
          mysql_close($link);
     }
?>
```

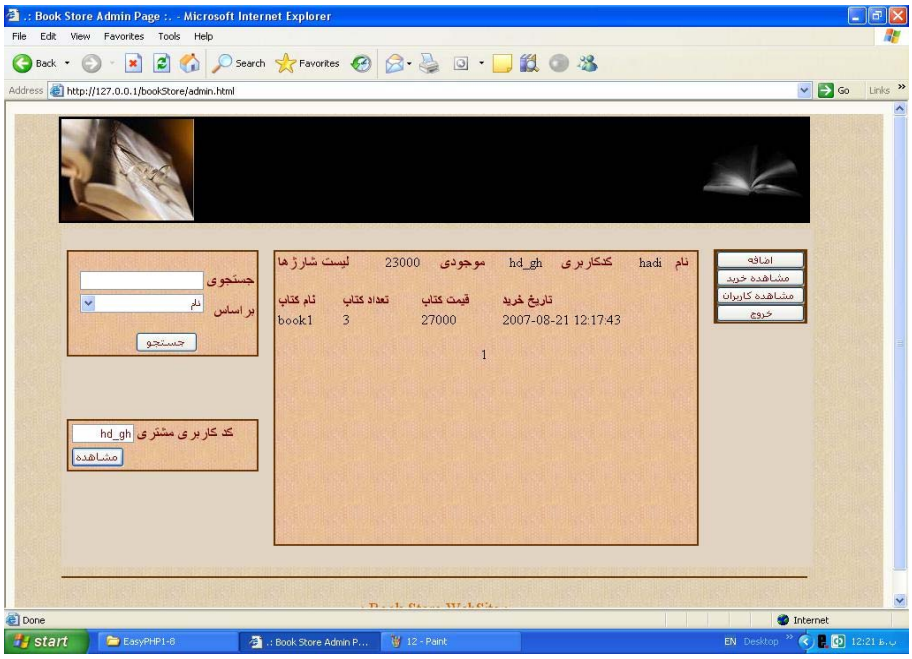
در شکل ۹-۸ نمو نه ی از اطلاعات کاربران بهمراه اطلاعات خرید کاربران و اطلاعات لیست شارژ را مشاهده می کنید.

بررسی کدهای صفحه ShowUser.Php

جستجوی تمامی کاربران عضو سایت بر عهده این صفحه است.

```
<?php
     $count=$_GET['u'];
     session_start();
     $link=mysql_connect('localhost','root','');
     mysql_select_db('bookstore');
     $q="select *from customers where type=0";
     $res=mysql_query($q,$link);
     $num=mysql_num_rows($res);
     $k=$num/5;
     $q="select *from customers where type=0 limit $count,5";
     $res=mysql_query($q,$link);
     $row=mysql_fetch_array($res);
     $id = $row[0];
$str='<table width=100% ><tr><th></th><th align=left
          style="color:#660000">موجودی</th><th align=left
          style="color:#660000">تلفن</th><th align=left
          style="color:#660000">کدکاربری</th><th align=left
          style="color:#660000">نام</th></tr>';
while ($row!=NULL){
          $id = $row[0];
          $tid = $id.'a';
```

```
        $str.="<tr><td onclick='sharj($id);'
             style='cursor:pointer;color:#FF3300;
             font-size:18px'>شارژ موجودی</td><td>$row[6]</td>
             <td>$row[4]</td><td>$row[7]</td><td>$row[1]
             </td></tr><tr><td><div id='$id'
              style='display:none'> <input type=button
             value='ثبت' style='font-family:Tahoma;
             font-size:12px;color:#660000;width:50px'
             onclick='getData($id);'> <input type=text
             id='$tid' style='width=70px'>
             </div>
             <hr style='color:#660000'></hr></td></tr>";
        $row=mysql_fetch_array($res);
    }
    $str.='</table><p align=center >';
    for ($i=0;$i<$k;$i++){
      $m=$i+1;
      $str.="<a onclick='showUser($i*5)'
         style='cursor:pointer'>$m</a>";
    }
    $str.='</p>';
    echo $str;
?>
```



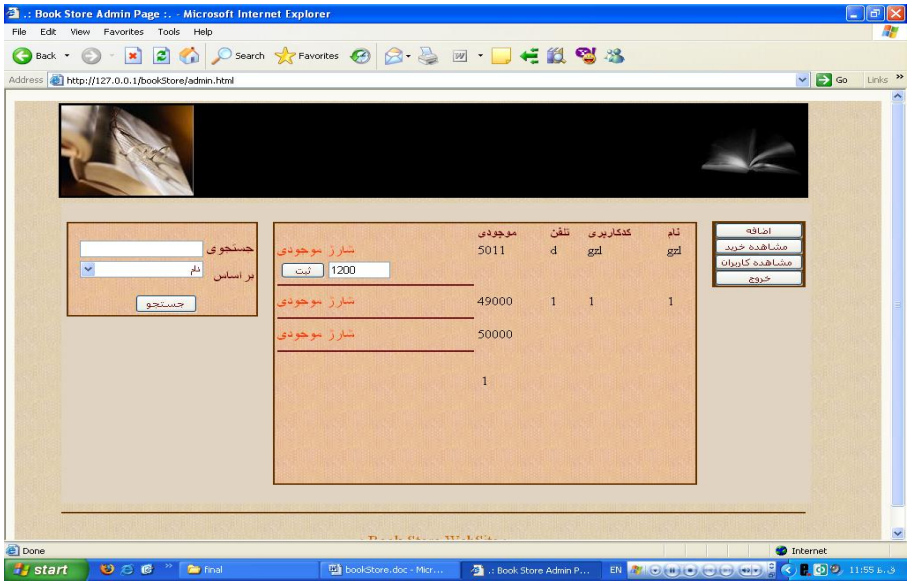
شکل ۹–۸ نمایش اطلاعات یک کاربر

برای شارژ موجودی کاربر کافی است میزان شارژ را نوشته و ثبت نماید (شکل ۱۰–۸).

بررسی کدهای صفحه sharj.php

عملیات این صفحه تغییر موجودی کاربر و ذخیره کردن اطلاعات شارژ در جدول Sharj است.

```
<?php
     session_start();
     $u = $_GET['code'];
     $t = $_GET['account'];
     $link = mysql_connect('localhost','root','');
     mysql_select_db('BookStore');
     $cmd = "select * from customers where customerid=$u";
     $r = mysql_query($cmd);
     $row = mysql_fetch_array($r);
     $g = $t + $row['account'];
     $q = " update customers set account = $g
           where customerid=$u ";
     $res = mysql_query($q,$link);
     $date = date("Y-m-d H:i:s");
     $q1 = " insert into sharj(customerid,value,date)
           values($u,$t,'$date')";
     mysql_query($q1,$link);
     echo '50';
     mysql_close($link);
 ?>
```



شکل ۱۰-۸ مشاهده لیست تمام کاربران

بررسی کدهای صفحه sharjList.php

عملیات این صفحه نمایش لیست شارژ های انجام شده برای یک کاربر است.

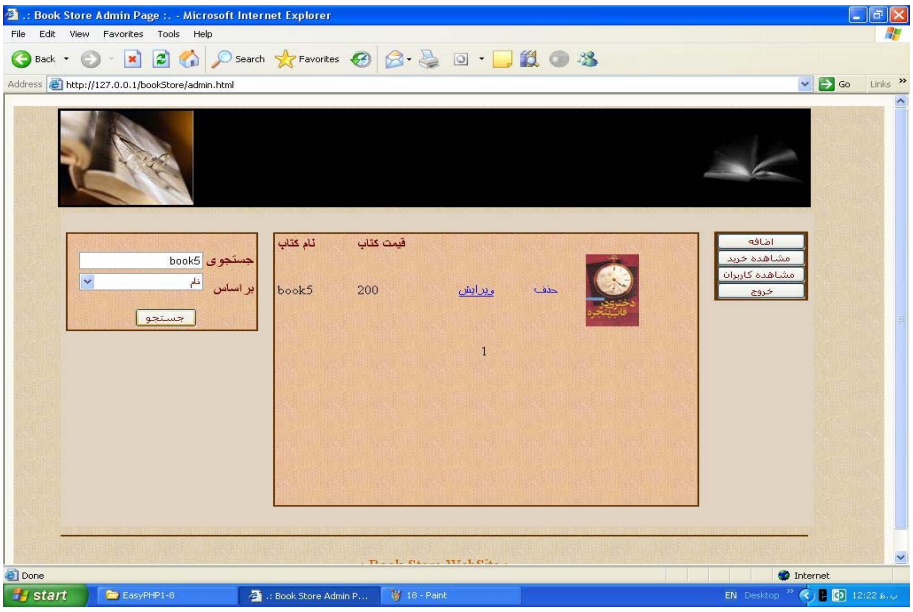
```
<?php
     $u = $_GET['u'];
     session_start();
     $link = mysql_connect('localhost','root','');
     mysql_select_db('BookStore');
     $q = "select * from sharj where customerid=$u";
     $res = mysql_query($q);
```

```
    $row = mysql_fetch_array($res);
    $str='<table width=100%>';
    while($row){
        $str.="<tr><td>$row[2]</td><td>$row[1]</td></tr>";
        $row = mysql_fetch_array($res);
    }
    $str.='</table>';
    echo $str;
?>
```

شکل ۱۱-۸ صفحه ی است که مدیر می تواند با کلیک کردن بر روی هر کدام یک از گزینه های حذف و ویرایش عملیات را انتخاب نماید.

بررسی کدهای صفحه serachBookad.php

عملیات این بخش جستجوی اطلاعات کتاب برای صفحه مدیر است.



شکل ۱۱-۸ اطلاعات نمایش داده شده بعد از جستجو

```
<?php
    session_start();
    if (isset($_GET['txt1']))
    {
        $index = $_GET['number'];
        $name = $_GET['txt1'];
        $count =$_GET['count'];
        $link = mysql_connect('localhost','root','');
        mysql_select_db('BookStore');
        //
        $q = "select * from book where ";
        if ($index == '0')
            $q.= "'$name' = name ";
        else if ($index == '1')
            $q.= "'$name' = subject ";
        //
        if(isset($_SESSION['code'])){
```

```
            $u=$_SESSION['code'];
            $q.=" and code != $u ";
        }
        $res = mysql_query($q,$link);
        $num = mysql_num_rows($res);
        $k = $num/3;
        //
        $q.="limit $count,3";
        $res = mysql_query($q,$link);

        $row = mysql_fetch_array($res);

        $str = '<table width=100% ><tr><th align=left
              style="color:#660000">نام کتاب</th>
            <th align=left style="color:#660000">
            قیمت کتاب</th></tr>';
        while( $row != NULL)
        {
          $str.="<tr><td> $row[2]</td><td>$row[7]</td>
            <td><a href='edit.php?code=$row[0]'> ویرایش
            </a></td><td><a onclick='Del($row[0]);'
              style='cursor:pointer;color:#0000ff' >
            حذف</a></td><td><img src='$row[8]'></td>
            </tr>";
          $row = mysql_fetch_array($res);
        }
        $str.='</table><br><p align=center>';
        for($i=0;$i < $k; $i++){
          $m = $i+1;
          $str.= "<a onclick='searchBook(($i)*3);'
            style='cursor:pointer'>$m</a> ";
        }
        $str.='</p>';
        echo $str;
        mysql_close($link);
    }
?>
```

شکل ۱۲-۸: هنگامی که عملیات ویرایش کتاب انتخاب شود، اطلاعات کتاب به صفحه edit.php ارسال می شود.

بررسی کدهای صفحه edit.php

این صفحه همانند صفحه اضافه کردن کتاب می باشد با این تفاوت که در فیلد های فرم، اطلاعات کتاب مورد نظر قرار دارد.

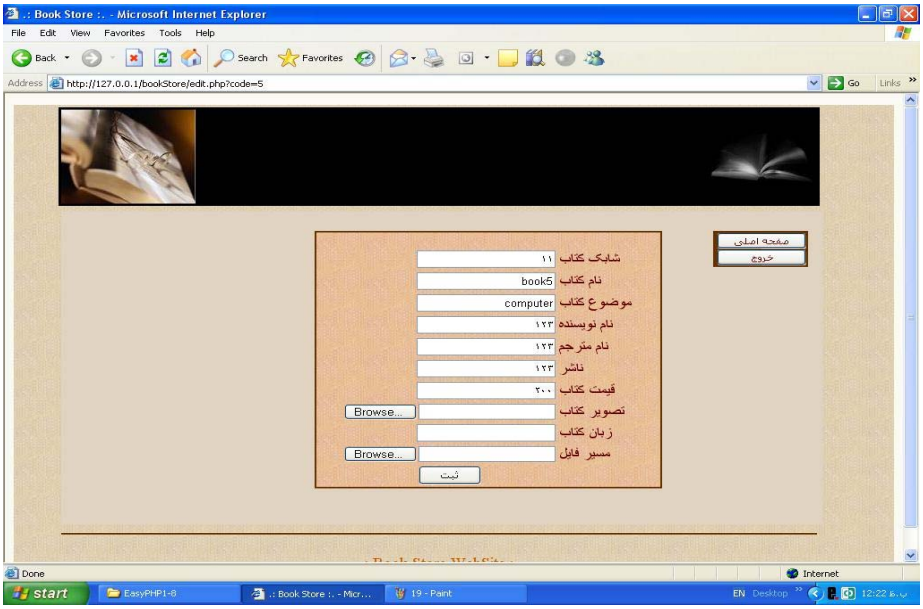
```
<?php
    $code=$_GET['code'];
```

```
    $link = mysql_connect('localhost','root','');
    mysql_select_db('BookStore');
    $q="select * from book where code=$code";
    $r=mysql_query($q,$link);
    $row=mysql_fetch_array($r);
?>
```

بررسی کدهای صفحه DeleteBook.php

عملیات این بخش مربوط به حذف کتاب از جدول کتاب ها است.

```
<?php
    session_start();
    $u = $_GET['code'];
    $_SESSION['code']=$u;
    $link = mysql_connect('localhost','root','');
    mysql_select_db('BookStore');
    $q="delete from book where code=$u";
    mysql_query($q,$link);
    mysql_close($link);
    echo '200';
?>
```



شکل ۱۲-۸ نمای از طراحی صفحه edit.php

بررسی کدهای صفحه update.php

این صفحه تغییرات اعمال شده بر روی اطلاعات کتاب، که در edit.php صورت گرفته را ذخیره می کند.

```
<?php
    $code = $_POST['code'];
    $isbn=$_POST['isbn'];
    $name = $_POST['name'];
    $subject = $_POST['subject'];
    $author = $_POST['author'];
```

```
    $translater = $_POST['translater'];
    $press = $_POST['press'];
    $price = $_POST['price'];
    $language = $_POST['language'];
    $image = $_FILES['image'];
    $path = $_FILES['path'];

    $dir = 'mybookpic/'.$image['name'];
    copy($image['tmp_name'],$dir);

    $dir1 = 'mydownload/'.$path['name'];
    copy($path['tmp_name'],$dir1);

    $link = mysql_connect('localhost','root','');
    mysql_select_db('BookStore');

    $q="update book set isbn='$isbn',name='$name',
        subject='$subject',author='$author',
        translater='$translater'
        ,press='$press',price=$price,language='$language'
        ,image='$dir',path='$dir1' where code=$code";
    mysql_query($q,$link);
    echo "<script >location.href('admin.html');</script>";
    mysql_close($link);
?>
```

بررسی کدهای صفحه insertBook.php

این صفحه فرمی را برای اضافه کردن کتاب به جدول کتاب، فراهم می کند. همچنین امکان upload کردن تصویر و فایل برای دانلود در این قسمت فراهم است. اطلاعات فرم پرشده به صفحه insert.php ارسال می شود. برای اینکه در صفحه بتوانیم عملیات upload را انجام دهیم باید Enctype تگ فرم مقدار multipart/form-data را دارا باشد.

```
<form action="insert.php" method=POST enctype="multipart/form-
data">
<?php
    if(isset($_GET['msg']))
    if($_GET['msg']==1)
        echo 'موفقیت آمیز بود';
    else
        echo 'موفقیت آمیز نبود';
?>
```

بررسی کدهای صفحه insert.php

در این قسمت اطلاعات کتاب را به جدول کتاب اضافه می کنیم.

```
<?php
    $code = $_POST['code'];
    $name = $_POST['name'];
```

```
$subject = $_POST['subject'];
$author = $_POST['author'];
$translater = $_POST['translater'];
$press = $_POST['press'];
$price = $_POST['price'];
```

اطلاعات فایل upload شده را باید با آرایه $_FILE بگیرید.

```
$image = $_FILES['image'];
$language = $_POST['language'];
$path = $_FILES['path'];
```

دستور copy برای upload کردن فایل به پوشه ی درون سرور است. پارامتر اول این دستور مسیر موقتی فایل در وب سرور و پارامتر دوم مسیر اصلی برای کپی شدن فایل می باشد.

```
$dir = 'mybookpic/'.$image['name'];
copy($image['tmp_name'],$dir);

$dir1 = 'mydownload/'.$path['name'];
copy($path['tmp_name'],$dir1);

$link = mysql_connect('localhost','root','');
mysql_select_db('BookStore');

$q="insert into book(Isbn,name,subject,author,Translater
     ,press,price,image,language,path)values(
     '$code','$name','$subject','$author','$translater'
     ,'$press',$price,'$dir','$language','$dir1')";
$res = mysql_query($q,$link);

if($res )
  echo "<script>
     window.location='insertBook.php?msg=1'</script>";
else
  echo "<script>
     window.location='insertBook.php?msg=0'</script>";

mysql_close($link);
?>
```

در انتها یادآور می شویم که این فصل فقط یک مثالی برای کاربردهای تجاری تر Ajax در برنامه های وب بود هر چند که این برنامه در مقیاس کوچکی از برنامه های تجاری واقعی است ولی مفاهیم اصلی آن برنامه ها را داراست.

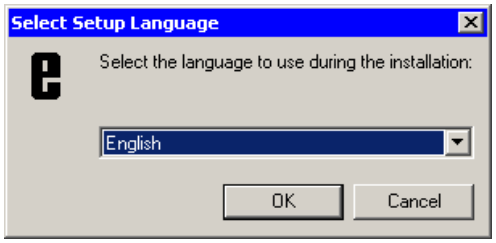
پیوست: طریقه کار با نرم افزار EsayPHP

این نرم افزار یک وب سرور محلی بر روی سیستم شما ایجاد می کند. این وب سرور برای اجرای کدهای PHP و سرویس دهی صفحات وب به شما کمک می کند. فایل لازم برای نصب EsayPHP در CD کتاب موجود می باشد.

طریقه نصب نرم افزار

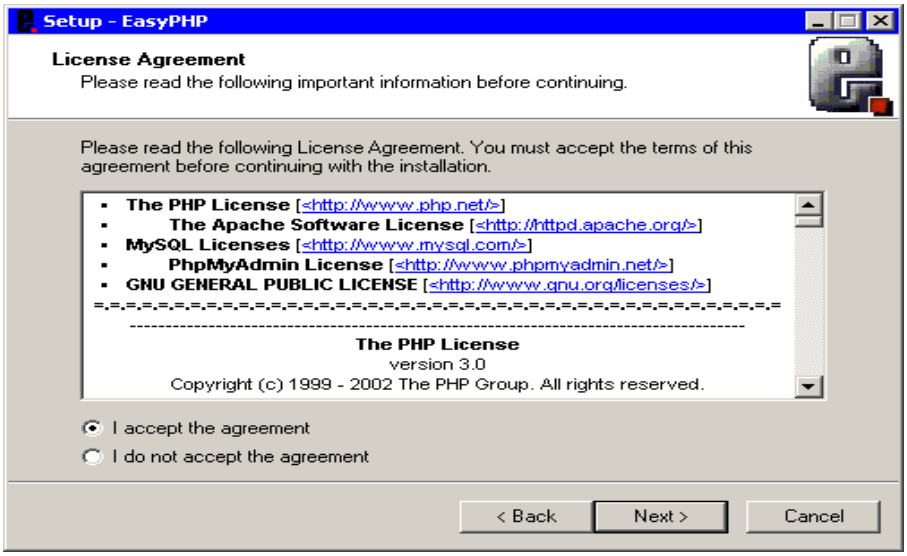
۱- فایل easyphp1-8_setup.exe را اجرا کنید.

۲- زبان مورد نظر خود را انتخاب کنید.

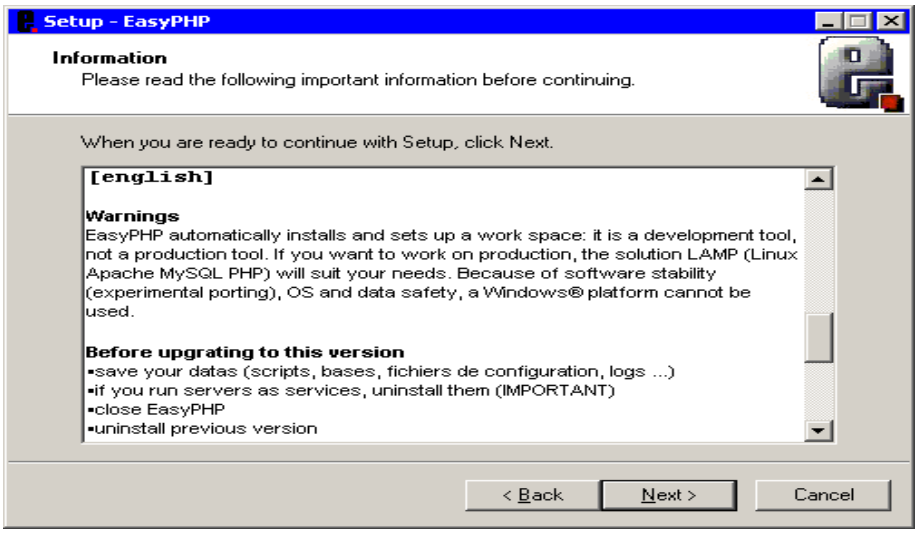


۳- صفحه مربوط به خوش آمدگویی است که دکمه Next را کلیک می کنید.

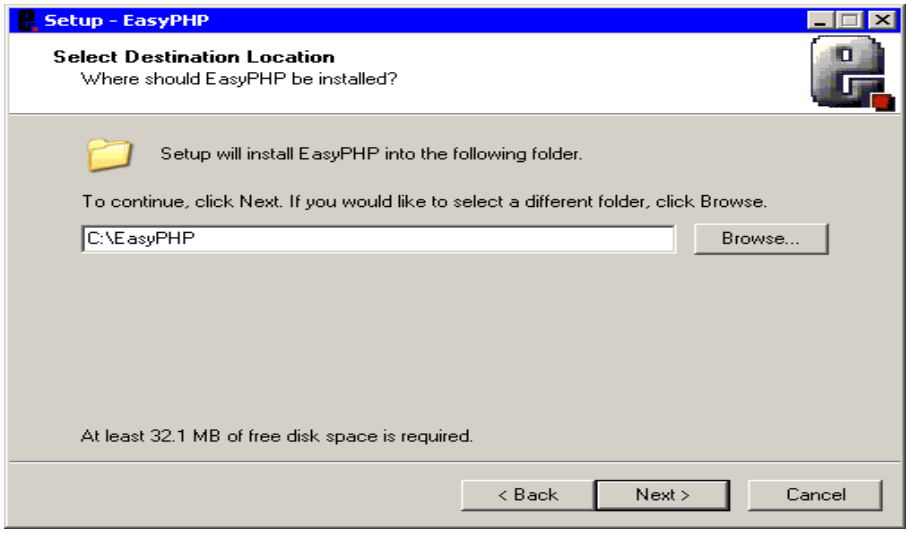
۴- موافقت با قوانین نرم افزار.



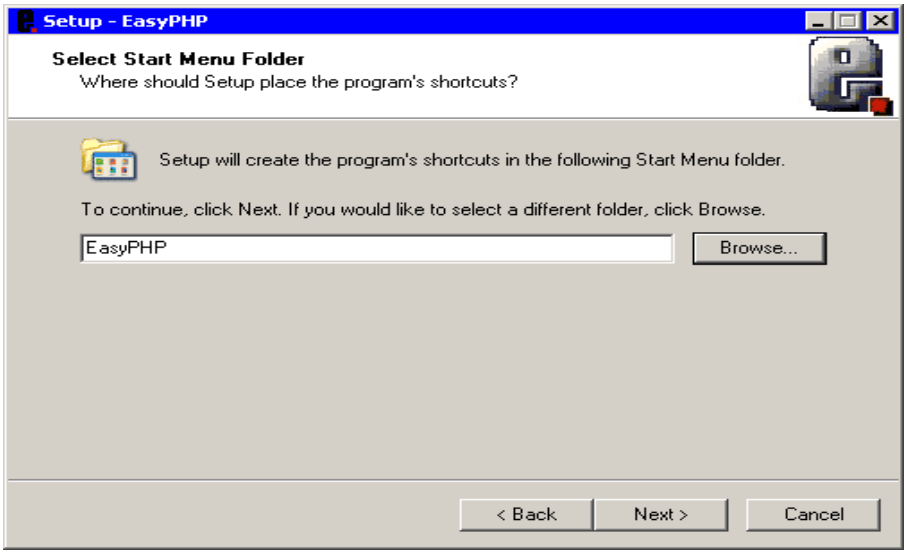
۵- مایل بودن به نصب نرم افزار با کلیک دکمه Next.



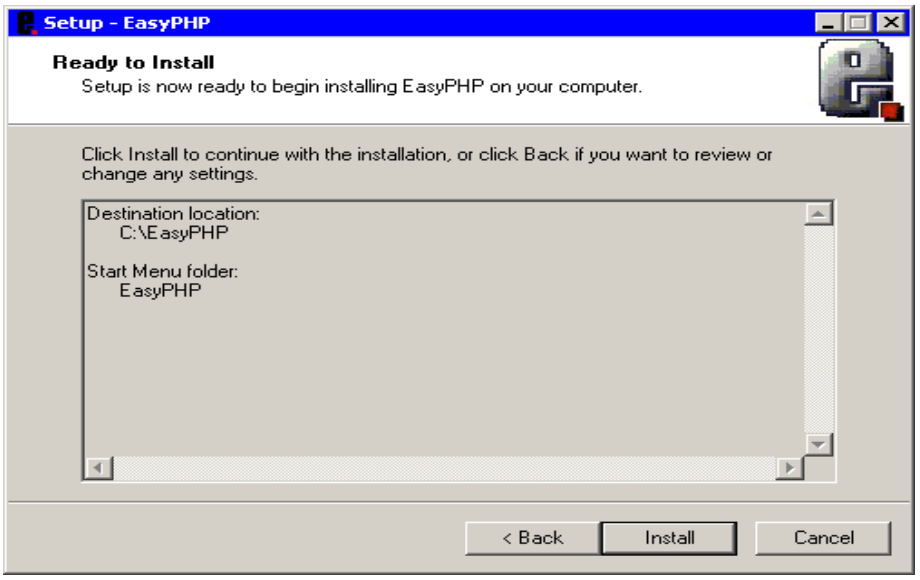
۶- انتخاب مسیر برای نصب نرم افزار.

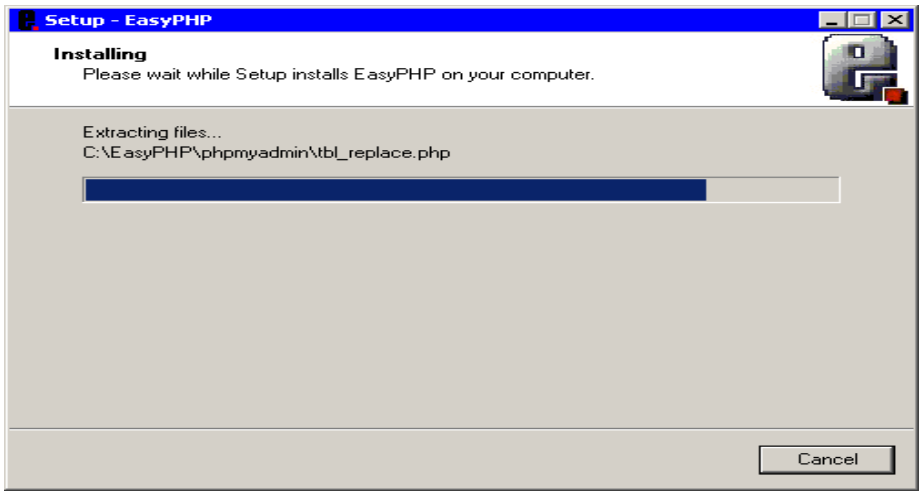


۷- عملیات نصب نرم افزار.



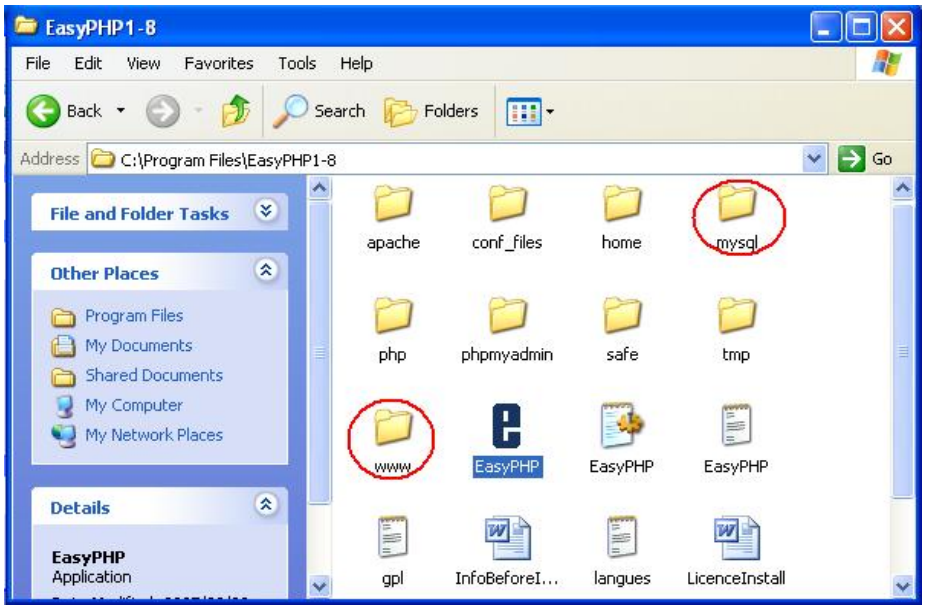
ادامه ی عملیات مربوط به نصب.





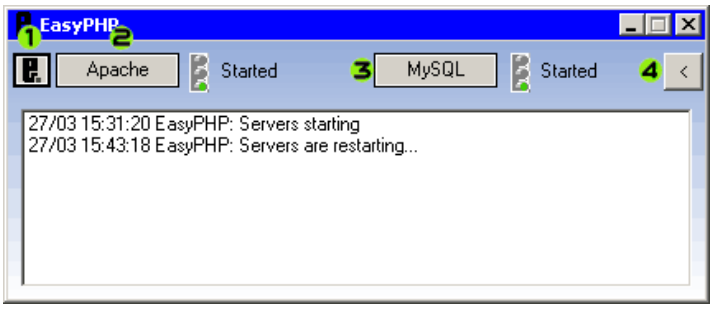
۸- پایان عملیات نصب.

بعد از نصب نرم افزار، به محل نصب نرم افزار بروید و فایل EsayPHP.exe را اجرا کنید.



در این مسیر پوشه ی به نام www وجود دارد که محلی است برای این که شما صفحات وب خود را در آنجا قرار دهید. این نرم افزار از mysql برای مدیریت جدول ها استفاده می کند. در مسیر نصب برنامه، پوشه ای به نام mysql وجود دارد که در آن، پوشه ای به نام Data است که محلی است که در آن جدول های برنامه قرار می گیرند.

این نرم افزار وب سروری را به آدرس http://localhost.com یا 127.0.0.1 بر روی سیستم شما ایجاد می کند. بعد از اجرای EsayPHP پنجره ی به شکل زیر نمایش داده می شود.



۱- دکمه برای مشاهده منوی نرم افزار.

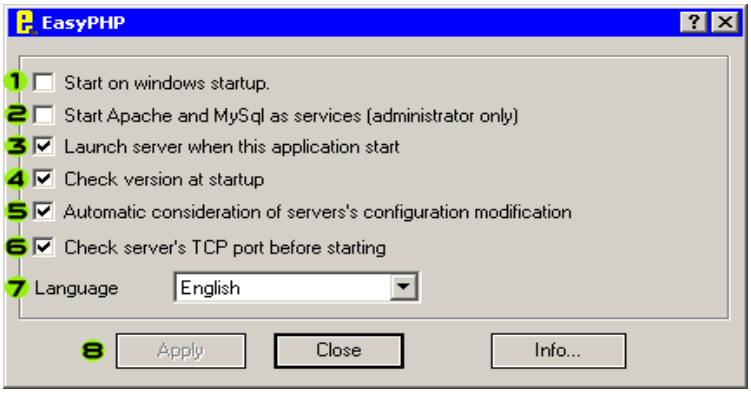
۲- این دکمه برای start,stop,restart سرویس دهنده Apache است.

۳- این دکمه برای start,stop,restart سرویس دهنده MySql است.

۴- این دکمه برای نمایش پیام های خطا و یا پیام های صحت اجرای Apache و MySql است.

پیکربندی EsayPHP

برای پیکربندی EsayPHP، ابتدا با کلیک راست بر روی آیکون EsayPHP در نوار وظیفه انتخاب Configuration/ EsayPHP فرمی به شکل صفحه بعد نمایش داده می شود.



۱- این گزینه برای این است که EsayPHP هنگام بار شدن ویندوز اجرا شود.

۲- اگر می خواهید نرم افزار همیشه در حال اجرا باشد این گزینه را انتخاب کنید.

۳- این گزینه برای آن است که هنگام اجرای نرم افزار، سرویس های MySql و Apache هم اجرا شوند.

۴- این گزینه مشخص می کند که هنگام اجرای EsayPHP، وب سایت آن برای نسخه جدیدتر نرم افزار بررسی شود.

۵- این گزینه برای بررسی خودکار فایل های پیکربندیMySql و Apache می باشد.

۶- انتخاب این گزینه موجب بررسی پورت های MySql و Apache جهت فعال یا غیر فعال بودن می شود.

۷- انتخاب زبان برای برنامه پیکربندی.

۸- اعمال تغییرات انجام شده.

در اینجا لیست فایل های پیکربندی نرم افزار آورده شده است.

- **C:\Program Files\EasyPHP\EasyPHP.ini**
فایل پیکربندی EsayPHP (این فایل نیازی به تغییر ندارد)

- **C:\Program Files\EasyPHP\conf_files\httpd.conf**
فایل اصلی پیکربندی Apache

- **C:\Program Files\EasyPHP\conf_files\php.ini**
فایل اصلی پیکربندی PHP

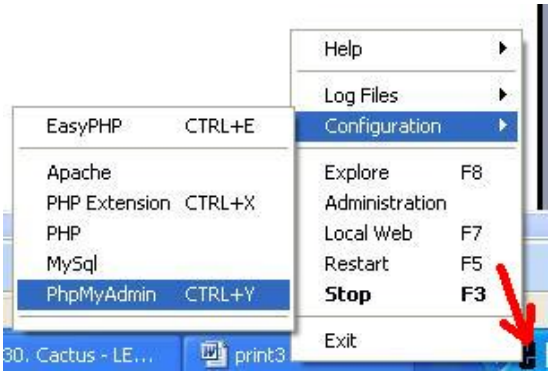
- **C:\Program Files\EasyPHP\conf_files\my.ini**
فایل پیکربندی MySql

- **C:\Program Files\EasyPHP\phpmyadmin\config.inc.php**
فایل پیکربندی PHP MyAdmin

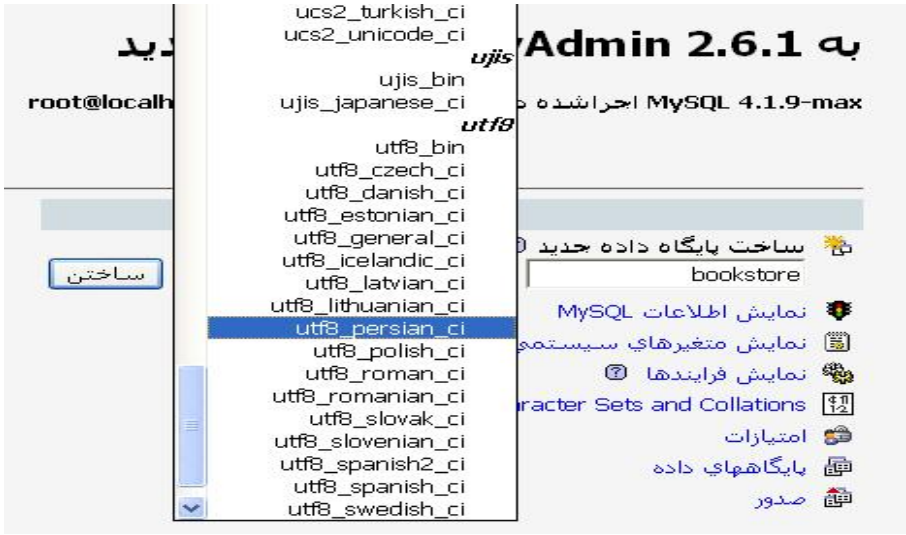
همچنین EsayPHP یک کپی از این فایل ها را در مسیر C:\ProgramFiles\EasyPHP\safe نگهداری می کند.

ایجاد جدول

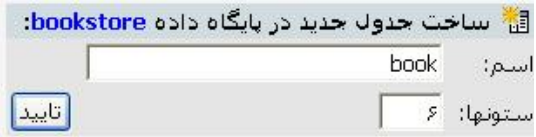
در شکل زیر طریقه ایجاد جدول در EsayPHP را مشاهده می نماید. با کلیک راست بر روی آیکون نرم افزار در نوار وظیفه انتخاب Configuration/PhpMyAdmin. در مرحله اول صفحه اصلی phpMyAdmin را باز می کنید.



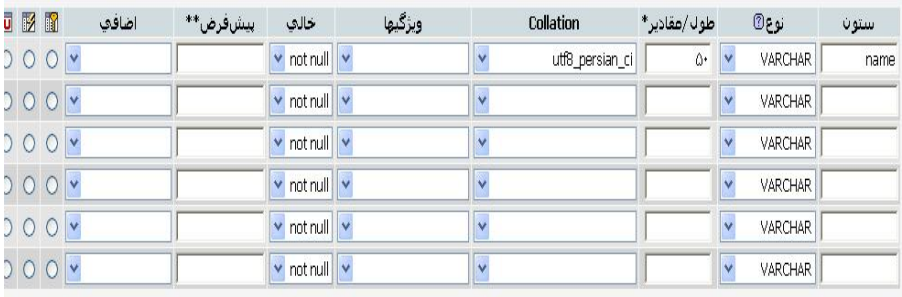
مرحله دوم: در این صفحه می توانید یک بانک اطلاعاتی برای برنامه خودتان ایجاد کنید. برای اینکه جدول های شما بتوانند کاراکترهای فارسی را پشتیبانی کنند Collation بانک را Utf_persian_ci انتخاب کنید.



مرحله سوم انتخاب نام برای جدول و مشخص نمودن تعداد فیلد های اطلاعاتی (ستون های جدول).



مرحله چهارم پر کردن اطلاعات فیلد های جدول.



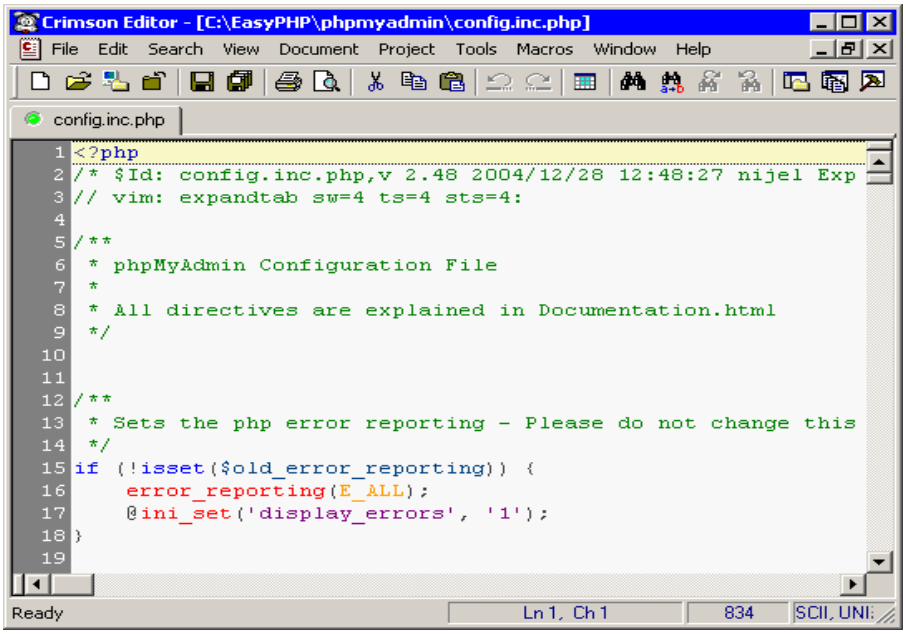
بعد از ذخیره کردن جدول شما می توانید براحتی با دستورات sql با این جدول ها کار کنید.

تنظیمات MySql

PhpMyAdmin برای دسترسی به MySql یک کد کاربری 'root' بدون رمز عبور ایجاد می کند

برای تغییر رمز عبور مراحل زیر را دنبال کنید:

فایل پیکربندی PhpMyAdmin را باز کنید. **(config.inc.php)**

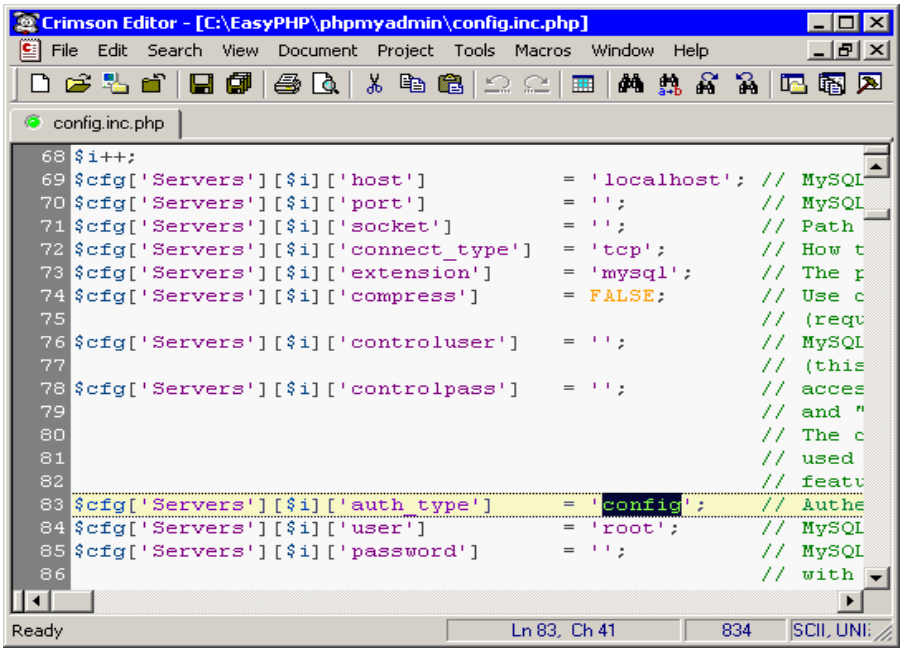


خط کد زیر را پیدا کنید.

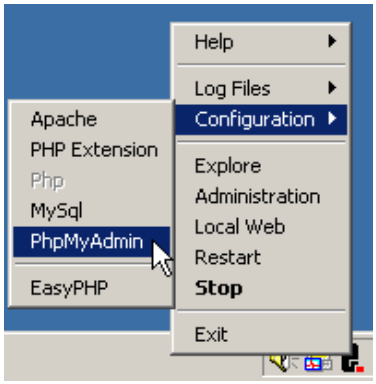
```
$cfg['Servers'][$i]['auth_type'] = 'config';
```

بجای کلمه 'config' کلمه 'http' قرار دهید.

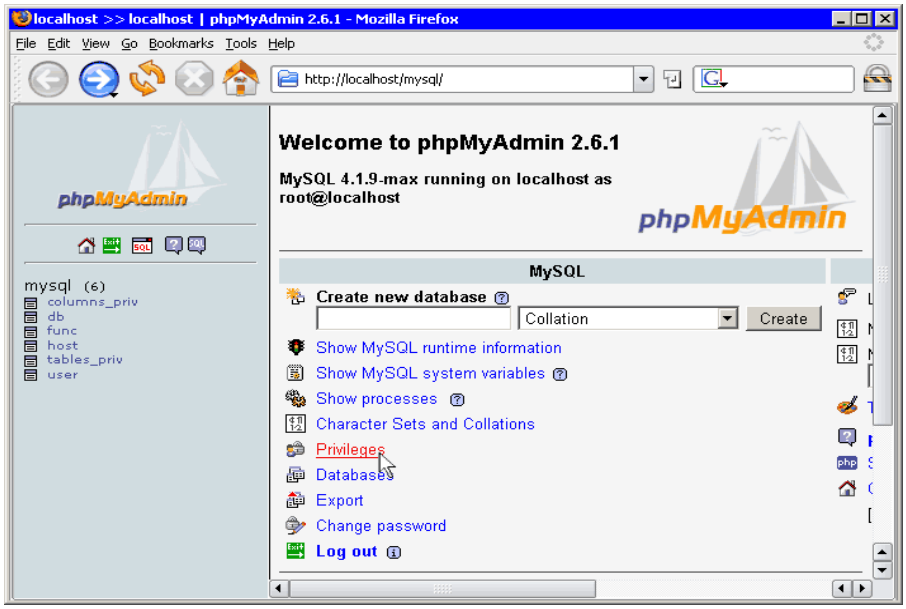
```
$cfg['Servers'][$i]['auth_type'] = 'http';
```



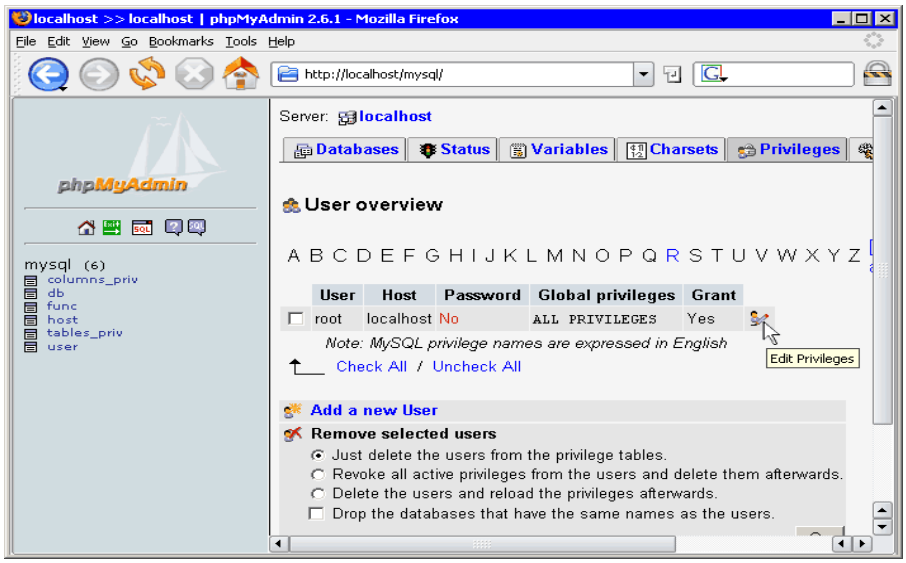
فایل را ذخیره کنید. این تغییر موجب می شود PhpMyAdmin هر زمان که اجرا شود از شما رمز عبور را درخواست کند. در مرحله بعد از آیکون EsayPhp، PhpMyAdmin را اجرا کنید.



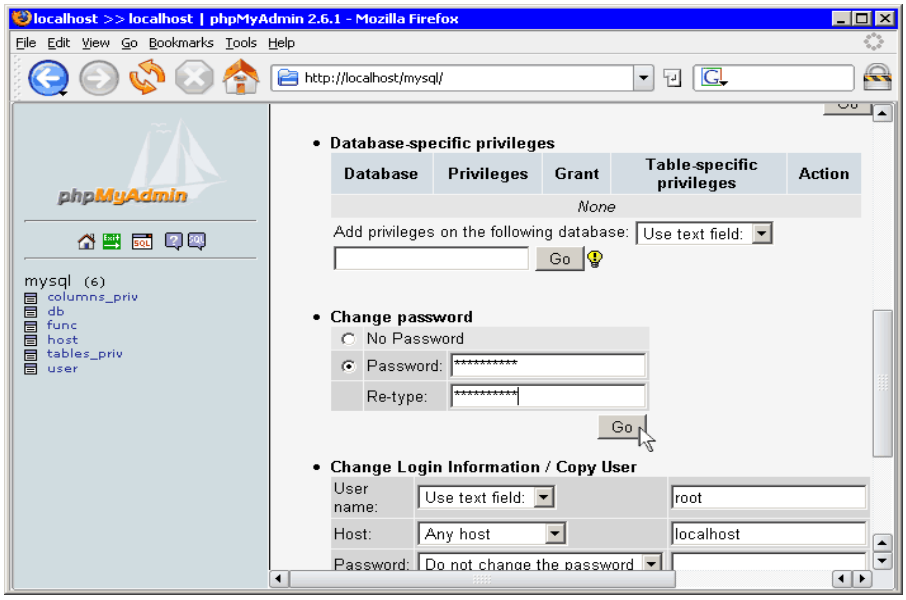
در صفحه نمایش داده شده گزینه 'Privileges' را انتخاب کنید.



در این صفحه شما می توانید کاربری را اضافه، حذف یا تغییری در مشخصات آن ایجاد کنید. برای تغییر رمز عبور آیکون Edit Privileges را انتخاب نماید.



رمز عبور جدید خود را تایپ کرده و دکمه ok را کلیک کنید.



این نرم افزار بسیار ساده و کارآمد می باشد شما براحتی می توانید برنامه های خود را نوشته و تست کنید.

منابع:

1-ASP.NET AJAX in Action
Copyright ©2008 by Manning Publications Co.
by: ALESSANDRO GALLO, DAVID BARKOL,RAMA KRISHNA VAVILALA

2-Ajax in Action
Copyright ©2006 by Manning Publications Co.
by: DAVE CRANE, ERIC PASCARELLO, WITH DARREN JAMES

3-Professional Ajax 2nd Edition
Copyright © 2007 by Wiley Publishing,
by: Nicholas C. Zakas, Jeremy McPeak, Joe Fawcett

4-Beginning Ajax with ASP.NET
Copyright © 2006 by Wiley Publishing,Inc.
by: Wallace B. McClure, Scott Cate, Paul Glavich, Craig Shoemaker

5-Ajax For Dummies®
Copyright © 2006 by Wiley Publishing, Inc.
by: Steve Holzner,PhD

6-Ajax Patterns and Best Practices
Copyright © 2006 by Apress Publishing, Inc.
by: Christian Gross

7-http://ajax.asp.net/

8-www.getfirebug.com