مهندسی اینترنت

برنامه‌نویسی کاربردی روی سرویس‌دهندهٔ وب

گردآوری و تألیف : بهزاد اکبری

شرکت ناقوس اندیشه

(سهامی خاص)

تقدیم به

همسرم که در همهٔ مشکلات همدرد و مشوق من بود.

فهرست

مقدمه

سپاس خدایی را که قدرت داد تا قلم در دست گرفته و با تحمل سختی‌های بسیار تدوین و جمع‌آوری این کتاب را به اتمام رسانم.

همان‌گونه که در بخش‌های مختلف این کتاب خواهید آموخت هدف این کتاب آموزش ساختار و پروتکل-های اینترنت، طرق پیاده‌سازی سایت وب، ISP همچنین روش‌های مختلف برنامه‌نویسی تحت وب همانند HTML, Cgi و بررسی روش‌هایی چون Applet های جاوا و مقایسهٔ این روش‌ها با ISAPI و Activex و Active Server Page می‌باشد.

این کتاب ترجمه و برداشت چندین کتاب می‌باشد که گروهی از آنها عبارتند از:

Active Server Page Unleashed

Cgi Programming Unleashed

Web Programming Unleashed

Internet Intranet Engineering

HTML &Cgi Unleashed Professional Refrence Edition

با کمال احترامات

بهزاد اکبری

فصل اول

معماری اینترنت

۱-۱ مقدمه

ارتباط در عصر حاضر و در شروع قرن بیست و یکم، الفبای زندگی صنعتی، مدرن و متمدن جوامع مترقی این کرهٔ خاکی را تشکیل می‌دهد. اهمیت ارتباطات و در معنایی ساده‌تر، تبادل اطلاعات بین جوامع بشری و انسان‌ها که تشکیل دهندهٔ جوامع بشری می‌باشند، در عصر امروز آنقدر مهم و حائز اهمیت است، که به زعم بزرگان علم، در جهان امروز اگر کسی خود را بی‌نیاز از تبادل اطلاعاتی بداند، در حقیقت دچار توهمی بزرگ از یک محیط پررمز و واقعیت گشته و در حقیقت از آمادگی لازم برای ورود به قرن بیست و یکم برخوردار نیست و از قافلهٔ علم بشری به شدت عقب مانده است.

نظریات ارتجاعی در محدود کردن جوامع بشری، در دستیابی به اطلاعات، محکوم به شکست هستند. پیشرفت علم وفن درجهان امروز آنقدر سریع و شتابان است، که هرگز هیچ کشور مقتدری در دنیا خود را از دریافت اخبار و اطلاعات مربوط به مراکز دیگر یا کشور دیگربی نیاز نمی‌داند. حتی کشورهای در حال توسعه، یا حتی کشورهای فقیر، به تناسب خود سود می‌برند. در جهان متنوع و رنگارنگ ما، قومیت‌ها و مذاهب و فرهنگ‌های جوامع مختلف به از بین رفتن منازعات قومی و فرهنگی و مخاصمات دینی منجر می‌شود و افکار ملل دنیا به طرز بی‌سابقه‌ای به یکدیگر نزدیک می‌گردند. علم انحصاری و تکنولوژی که سابق بر این منحصراً در اختیار کشورهای پیشرفتهٔ صنعتی قرار داشت، از انحصار آنها خارج می‌گردد و در سطح جهان گسترده و ملل مختلف از نعمات آن بهره‌مند می‌شوند.

اگر حادثه‌ای در نقطه‌ای از جهان روی دهد، با مخابرهٔ خبر آن، در عرض کمتر از یک صدم ثانیه در سراسر جهان، همهٔ مردم دنیا به کمک این قسمت از کرهٔ خاکی می‌شتابند و همهٔ اینها ممکن نیست، مگر به‌وسیلهٔ تکنولوژی ارتباط و اطلاعات. گسترش سیستم‌های اطلاعاتی در سرتاسر جهان از قبیل ماهواره‌ها، سیستم‌های مایکروویو، سیستم‌های اطلاعات کامپیوتری و غیره جهان بزرگ ما را به یک دهکدهٔ کوچک تبدیل کرده است، به‌طوری که هر فرد از هر ملیتی در دورترین فاصلهٔ کرهٔ خاکی می‌تواند در آن واحد با دیگری ارتباط برقرار کند و هر اتفاقی هر چقدر کوچک و بی‌اهمیت توسط سیستم‌های پیشرفتهٔ تبادل اطلاعات در عرض صدم ثانیه به دورترین فاصله از آن نقطه مخابره می‌شود، گو اینکه فاصله‌ها در جهان ما از بین رفته و بعد جغرافیایی کرهٔ زمین و چه بسا فضای کیهانی تبدیل به مسافتی کوتاه شده است.

۱-۲ تاریخچهٔ اینترنت

برای درک اساسی و بنیادین یک علم و جهت‌گیری به سمت جنبه‌های علمی آن، دانستن تاریخ و علل به‌وجود آمدن آن ضروری می‌نماید، لذا ابتدا به تشریح تاریخ این علم می‌پردازیم. تولد ارتباطات کامپیوتری تاریخ جالبی دارد و آن به رقابت بین دو ابر قدرت قرن بیستم یعنی اتحاد جماهیر شوروی سابق و ایالات متحدهٔ آمریکا مربوط می‌شود. همان‌طور، که می‌دانید اولین ماهوارهٔ مصنوعی ساخت دست بشر در سال ۱۹۵۷ به نام اسپوت نیک توسط شوروی به فضا پرتاب گردید، درست از همین سال علم تبادل اطلاعات کامپیوتری به‌وجود آمد.

ماهواره‌ها در ارتباط مخابراتی و جاسوسی و به‌طورکلی جذب اخبار و ارقام توانایی‌های فراوانی دارند. این توانائی و قابلیت کاملاً مورد توجه دانشمندان آمریکایی و به‌طورکلی نظام آمریکایی بود. نظامی که در این زمینه در آن سال‌ها دارای عقب ماندگی محسوسی از شوروی بود، لذا مراکز تحقیقاتی به‌طور اعم و مراکز نظامی به‌طور اخص در آمریکا مأمور شدند تا با توجه به قابلیت و توانایی موشک‌های شوروی در پرتاب ماهواره‌ها به فضا در نتیجهٔ حمل بمب اتمی توسط موشک و پرتاب آن به سمت شهرهای آمریکا و همچنین توانایی گسترده مخابراتی شوروی سیستمی را طراحی کنند، که اگر به فرض یکی از شهرهای آمریکا توسط بمب‌های اتمی نابود شد، سیستمی موجود باشد و اطلاعات موجود در کامپیوترهای این شهر را قبل از نابودی به شهر دیگر منتقل کند. دانشمندان و محققان در پنتاگون (وزارت دفاع آمریکا) موفق به طراحی سیستمی شدند، که قابلیت انتقال اطلاعات مثلاً از طبقهٔ دوم پنتاگون اتاق ٤٠٢ را به طبقهٔ چهارم اتاق ٩٤٤ و سایر طبقات و اتاق‌های این وزارتخانه، داشت. یعنی دو کاربر یا چند نقطهٔ مختلف این سازمان توانایی تبادل اطلاعات بین یکدیگر و بین یک کامپیوتر مرکزی را داشتند و همچنین می‌توانستند توسط این سیستم به تبادل نامه بپردازند، که این سیستم انتقال نامه هم اکنون پست الکترونیکی[1] نامیده می‌شود، اما یادآوری این نکته ضروری می‌نماید که در سیستم‌های مدرن امروزی و سیستم‌هایی که در آینده طراحی خواهند شد، انتقال نامه، به یک موضوع پیش پا افتاده و بسیار ساده تبدیل خواهند شد. در سیستم‌های آینده، انسان خود انتخابگر می‌باشد و آنچه او اراده کند که انجام دهد، فقط با یک کامپیوتر و یک مودم و یک خط تلفن در منزل یا محل کار او امکان‌پذیر می‌شود.

در آینده می‌توان فیلم‌های سینمایی را در خانه دید، به خرید اجناس مورد نیاز روزانه از طریق سرویس Remote Shopping پرداخت و کارهای اداره را از طریق کامپیوتر در منزل انجام داد. در آن زمان دیگر رادیو و تلویزیون و سینماها نیستند که برای ما تعیین و تکلیف کنند که چه برنامه‌ای ببینیم، بلکه آنچه خود اراده کنیم که ببینیم، برروی صفحهٔ نمایشگر کامپیوتر ظاهر می‌شود. دیگر در کتابخانه-های عریض و طویل به دنبال این کتاب و آن کتاب با کتابدار چانه نمی‌زنیم، بلکه هر کتاب که مورد علاقهٔ ما باشد تنها با فشار دکمه‌ای برروی صفحهٔ مانیتور ظاهر می‌شود و می‌توانیم صفحات آن را ورق بزنیم و مطالعه کنیم. عبور و مرور در شهرهای بزرگ کاهش می‌یابد، آلودگی زیست محیطی شهرهای

1. E-mail

بزرگ کم می‌شود و انسان‌ها عمر خود را در خیابان‌های پررفت و آمد و پشت راهبندان‌های سنگین تلف نمی‌کنند. وقت برای تفریح، تحقیق، مطالعه و کارهای فرهنگی و علمی و غیره فراهم می‌شود. بازارهای بورس توسعه می‌یابد و در نتیجهٔ گسترش و پیشرفت اقتصاد، جوامع بشری به رفاه می‌رسند.

آیا این زمان سطح بهداشت در کرهٔ زمین به صورت امروز باقی می‌ماند؟ یا ریشهٔ همهٔ بیماری‌ها به علت همفکری و تبادل اطلاعات در سرتا سر دنیا و در نتیجه پیشرفت علوم بهداشتی از بین می‌رود؟

آیاجایی برای سانسور اخبار از طریق دیکتاتور در دنیا باقی می‌ماند؟ یا دموکراسی به طرز چشمگیری در تمام کشورها فراگیر می‌شود؟ پاسخ با شما است.

اما برگردیم به تاریخچهٔ اینترنت : سیستمی که در این قسمت شرح گردید، در ابتدا به نام آرپا[1] مشهور شد. مأموریت اصلی و نهایی آرپا تحقیق و اتصال کامپیوترهای دانشگاه و مراکز نظامی از طریق بستر مخابراتی به نحوی بود که چندین کاربر بتوانند در یک محیط ارتباطی یا خط ارتباطی با هم شریک شوند. هدف، ایجاد شبکه‌هایی بود که در آن اطلاعات که همان داده‌های کامپیوتری می‌باشند، بتوانند از نقطه‌ای به نقطهٔ دیگر بروند و تمام شبکه‌های محلی در نقاط مختلف به یکدیگر متصل شوند.

البته در ابتدا هدف آرپا، ایجاد شبکه‌ای مانند اینترنت نبود و فقط یک اقدام احتیاطی در مقابل حملهٔ احتمالی موشک‌های اتمی دوربرد اتحاد جماهیر شوروی بود. در اوایل ۱۹۷۳ یعنی زمانی که سیستم-های کامپیوتری بزرگ[2] در بازار بودند و هنوز خبری از کامپیوترهای شخصی نبود، آرپا که با افزوده شدن انگلیس (DEFENCE) به آژانس پروژه‌های پژوهشی پیشرفته دفاعی، به DARPA تغییر نام داده بود، شروع به کار بر روی پروژهٔ جدیدی برای به هم مرتبط‌سازی سیستم‌ها کرد. هدف از این پروژه یافتن راهی برای متصل ساختن شبکه‌ها به یکدیگر بود، البته باید توجه داشت که هر یک از این شبکه‌ها برای جابه‌جایی اطلاعات خود از روش‌های متفاوتی استفاده می‌کردند. وقتی روش مرتبط ساختن کامپیوترهای شخصی پیش می‌آمد، صاحبان شبکه‌ها می‌توانستند از طریق تجهیزات خاصی موسوم به دروازه‌ها[3] شبکه‌های خود را به هم وصل کنند، که البته ارتباط بین شبکه‌ها احتیاج به پروتکل‌های مناسب داشت.

در سال ۱۹۶۲ پاول باران در مقاله‌ای تحت عنوان روی شبکه‌های ارتباطی به تشریح شبکه‌های PACKET SWITCH پرداخت، در این روش، داده‌ها به قطعات و بسته‌های کوچک‌تری خرد می‌شوند و هر بسته شبیه یک نامهٔ پستی شامل آدرس فرستنده و گیرنده است و می‌تواند از هر مسیری به مقصد برسد. در مقصد، بسته‌ها مجدداً یک پارچه می‌شوند و به فرم کامل، تحویل مقصد می‌شوند.

درسال ۱۹۶۹، ایالات متحده چهار کامپیوتر را با استفاده از تکنولوژی PACKET SWITCH در ایالت-های کالیفرنیا و یوتا به هم متصل کرد. این شبکه خوب کار کرد، کاربران این کامپیوترها توانستند تقریباً همزمان به دیگر کاربرها پیام بفرستند و فایل به اشتراک بگذارند، این پروژه همان آرپا نام گرفت، اما یک کلمهٔ جدید به انتهای آن اضافه شد و به صورت (ARPANET) درآمد. با گذشت زمان، کامپیوترها و

1. Arpa
2. Main Frame
3. Gateways

کاربران جدید در سایت‌های دولتی و دانشگاهی به آن اضافه شدند. درسال ۱۹۷۰، کامپیوترهای میزبان، استفاده از پروتکل کنترل شبکهٔ NCP را شروع کردند و یک سال بعد تعداد گره‌های این پروژه به ۱۵ و تعداد میزبان‌های آن به۳۲ عدد رسید. در همین سال شخصی به نام تایلون، سیستم نامه‌رسان الکترونیکی را برای یک شبکهٔ توزیع شده ابداع نمود. در ۱۹۷۳ کشورهای بریتانیا و نروژ به ARPANET متصل شدند. در سال ۱۹۷٤ دکتر رابرت متکالف نیز نظریهٔ خود را در مورد اینترنت ارائه داد. در همین سال سیرف و باب کان جزئیات پروتکل TCP را ارائه دادند. کمپانی BBN نیز نسخهٔ تجاری آرپانت به اسم رتل بنت را ارائه کرد. از اواسط دههٔ ۱۹۷۰ تا ۱۹۸۰ شبکه‌های کوچکی از تکنولوژی آرپانت استفاده می‌کردند و تصمیم گرفتند تا به صورت شبکه‌ای باهم کار کنند. آنها آرپانت را به عنوان هسته انتخاب کردند و شروع به ارتباط از طریق خطوط استیجاری نمودند. در سال ۱۹۸٦ سرعت این شبکه ٥٦ کیلو بیت در ثانیه بود. سرانجام در سال ۱۹۹۰ آرپانت تغییر نام یافت و نام اینترنت برروی آن گذاشته شد. در آن زمان از سیستم‌های یونیکس در یک محیط خط فرمانی[1] برای استفاده از امکانات اینترنت استفاده می‌شد. با دستوراتی مثل Telnet و Ftpبرای اتصال و استفاده از سایر امکانات اینترنت استفاده می‌شد، که لازم بود هر بار کاربر، شناسهٔ کاربری و رمز عبور[2] وارد نماید. (شکل ۱-۱)

```
$
$ftp cnn.com
  username:david
  password:*****

ftp> dir
Not connected.
ftp> help
Commands may be abbreviated.  Commands are:

!               delete          literal         prompt          send
?               debug           ls              put             status
append          dir             mdelete         pwd             trace
ascii           disconnect      mdir            quit            type
bell            get             mget            quote           user
binary          glob            mkdir           recv            verbose
bye             hash            mls             remotehelp
cd              help            mput            rename
close           lcd             open            rmdir
ftp> _
```

شکل ۱-۱ اینترنت متنی روی سیستم‌های UNIX

1. Command prompt
2. Account

۳-۱ وب

ویژگی اصلی اینترنت این است، که هر نوع کامپیوتری صرف‌نظر از مشخصات سخت‌افزاری و سیستم عامل، با رعایت یک مجموعه استاندارد می‌تواند به کامپیوترهای دیگری که آن استانداردها را رعایت می‌کنند، وصل شود.

جنبه‌های مختلف اینترنت در طول سالیان، تکامل بسیاری یافته است، اما مسئلهٔ اساسی که تا این اواخر وجود داشت، دشواری استفاده از اینترنت برای افراد غیرحرفه‌ای بود. راه حلی در اوایل سال ۱۹۹۳ پیدا شد. هنگامی که شیوهٔ موسوم به تارجهان گستر یا WWW[1]، استفاده از شبکه را برای هر کسی ممکن ساخت.

در سال ۱۹۸٤ دو متخصص فیزیک در دو نقطه از اروپا GENEVA و CERN روی یک پروژهٔ مشترک فیزیک کار می‌کردند و لازم بود به جدیدترین و به روزترین اطلاعات در رابطه با تحقیقات یکدیگر دسترسی داشته باشند. آنها روشی را ابداع کردند، که از این طریق می‌توانستند مستندات خود را روی اینترنت به اشتراک بگذارند، این روش را وب نامیدند. این مستندات که بعداً Page یا صفحه نامیده شد، می‌توانست توسط یک نرم‌افزار به نام مرورگر[2] نمایش داده شود و در این مستندات، کلمات کلیدی وجود داشتند که کاربر با انتخاب آنها می‌توانست به یک صفحهٔ دیگرروی اینترنت متصل شود، بدون اینکه هیچ‌گونه رمز عبور و شناسه کاربری وارد کند.

به سیستمی که روی اینترنت امکان به اشتراک گذاشتن صفحات را می‌داد WWW یا، تارجهان گستر می‌گویند.

تعریف وب

به مجموعه‌ای از روش‌ها و پروتکل‌ها که بین سرویس‌گیرنده و سرویس‌دهنده وب[3] مطرح می‌شوند، تا مستندات قابل اشتراک روی شبکهٔ اینترنت باشند، وب می‌گویند.

وب گرافیکی

در سال ۱۹۹۳ یکی از دانشجویان دانشگاه illinois به نام Andreessen، ایدهٔ رابط کاربر گرافیکی را روی وب مطرح کرد، که به نام مرورگرهای گرافیکی وب معرفی شدند و برای آن یک مرورگر به نام Mosaic (شکل ۲-۱) نوشت. این نرم‌افزار اولین مرورگر گرافیکی روی اینترنت بود.

Andreessen بعداً شرکت Netscape Navigator را تأسیس کرد، که بسیاری از کاربران اینترنت امروزه از این مرورگر استفاده می‌کنند. در این مرورگر امکان استفاده از تصاویر، تصاویر متحرک و

1. World Wide Web
2. Browser
3. Web Server

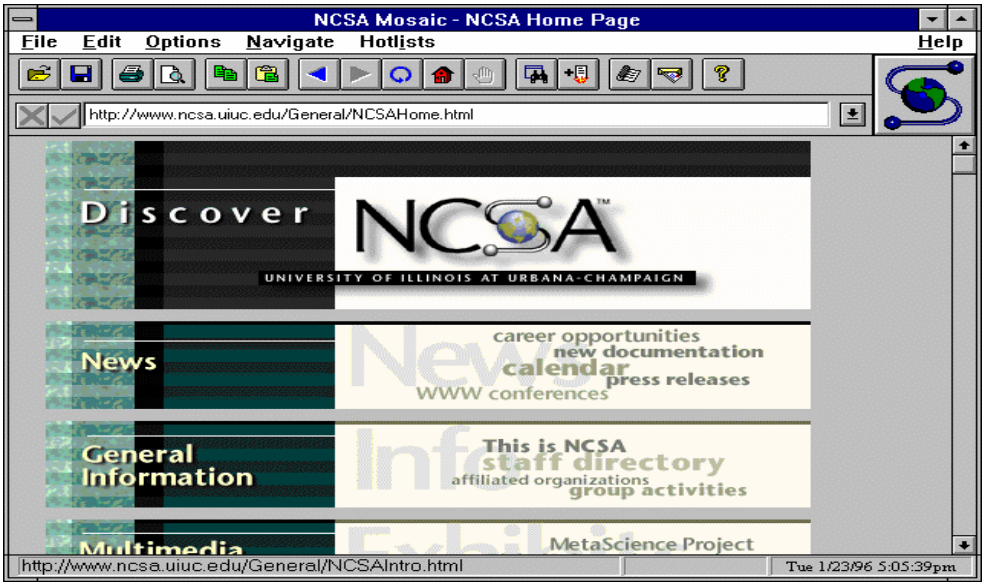
متن وجود داشت. به این مستندات که می‌توانستند علاوه بر متن و پیوند، دارای صدا و تصاویر متحرک نیز باشند، فرامتن[1] گویند.

مزایای وب

۱. بدون وارد کردن رمز عبور و شناسهٔ کاربری، کاربران می‌توانند مستندات خود را به اشتراک بگذارند.

۲. لزومی به دانستن اینکه سند در کجای اینترنت قرار دارد، وجود نداشت (یعنی محل قرار گرفتن صفحه[2] روی اینترنت از دید کاربر پنهان است)

۳. امکان استفاده از فرا متن به جای متن‌های معمولی وجود داشت.



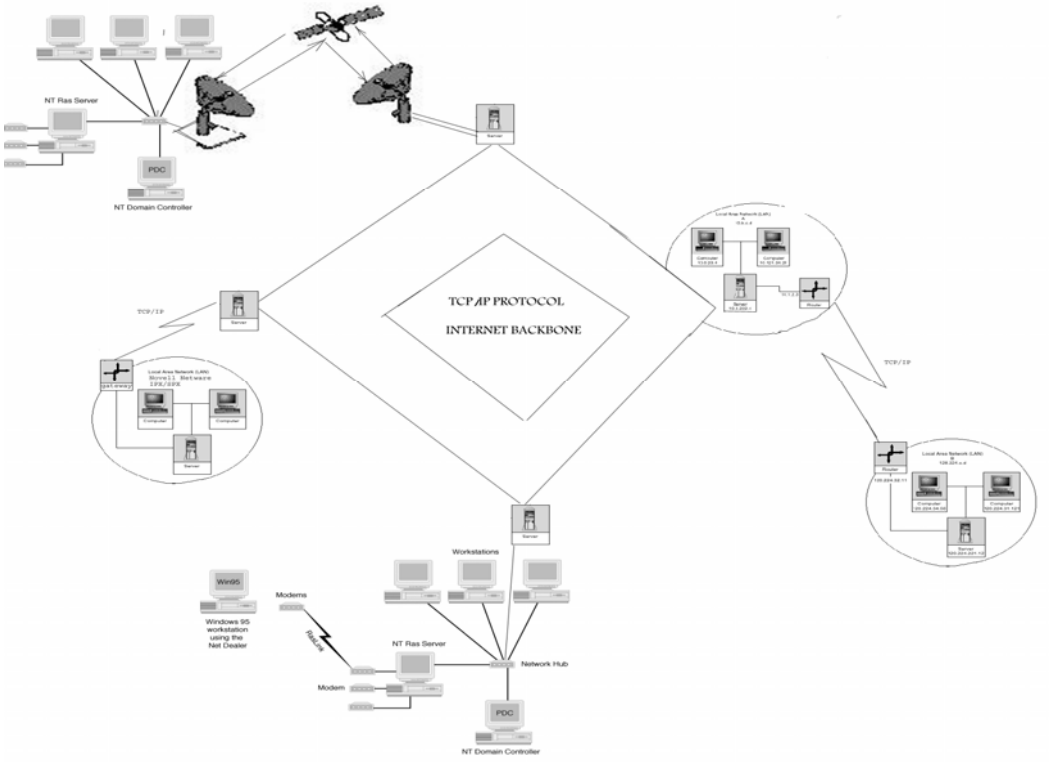
شکل۲-۱ شمایی از Mosaic

۱-۴ ساختار داخلی اینترنت و زیرشبکه‌های آن

اینترنت، شبکه‌ای در حد WAN، متشکل از تعدادی شبکهٔ کوچکتر LAN می‌باشد، که به این شبکه‌های کوچکتر زیرشبکه‌های اینترنت یا Subnet می‌گوییم. اینترنت شبکه‌ای از زیرشبکه‌ها می‌باشد. شبکهٔ اینترنت در تمام جهان پراکنده است و تمام کامپیوترها، مستقل از سکوی نرم‌افزاری و سخت‌افزاری[3]، یعنی مستقل از معماری و سیستم عامل، از طریق یک زبان مشترک یا پروتکل مشترک که همان TCP/IP می‌باشد، با هم مرتبط بوده و می‌توانند تبادل اطلاعات داشته باشند. در اینترنت چه معماری کامپیوتر

1. Hyper text
2. Page
3. Plateform

شما Risc باشد، یا Cisc یا پردازندهٔ شما INTELL، HP، Alpha یا Motorola باشد و چه سیستم عامل شما مایکروسافت باشد، یا یونیکس یا اپل، می‌توانند از طریق یک زبان مشترک با هم ارتباط داشته باشند. البته بعضی از زیر شبکه‌های اینترنت که دارای پروتکل TCP/IP نیستند، می‌توانند از طریق تجهیزات بین شبکه‌ای مثل Gateway، به اینترنت متصل شوند. شمایی از ساختار اینترنت را می‌توان در شکل زیر مشاهده کرد.



شکل ۳-۱ ساختار اینترنت شبکه‌های مختلف

ستون فقرات[1] اینترنت

به خطوط پرسرعتی که بیشتر فیبرنوری یا خطوط ماهواره‌ای با پهنای باند بالا می‌باشند و کامپیوترهای میزبان اینترنت را که عملیات مدیریت کاربران روی اینترنت را برعهده دارند به هم متصل می‌کنند، ستون فقرات اینترنت گویند. ستون فقرات اینترنت در مکان‌هایی که امکان‌پذیر بوده است، فیبرنوری که دارای سرعت بسیار بالایی است، می‌باشد و در مناطقی که امکان استفاده از آن وجود نداشته است از خطوط ماهواره‌ای با پهنای باند بالا استفاده می‌شود.

1. Back bone

روش‌های ارتباط ماهواره‌ای با اینترنت

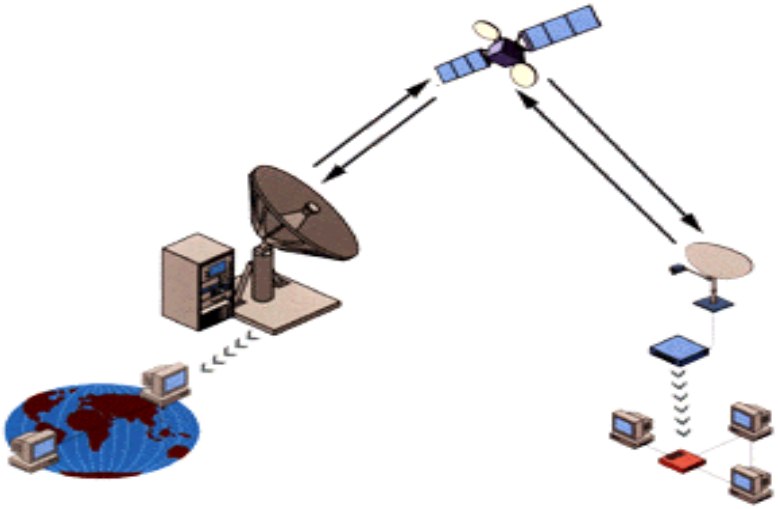
جهت برقراری ارتباط از طریق ماهواره به اینترنت، به دو صورت می‌توان عمل کرد، این دو روش عبارتند از:

۱. متقارن

دراین روش هم ارسال اطلاعات وهم دریافت اطلاعات از طریق خط ماهواره‌ای انجام می‌گیرد. در این حالت لازم است، که از دو Dish جهت این منظور استفاده کرد. هزینهٔ این روش بالا می‌باشد و در بعضی کشورها نیاز به مجوزهای خاصی دارد. (شکل ٤-١)

۲. نامتقارن

در این روش ارسال که بیشتر اوقات کمتر از دریافت می‌باشد، با خط تلفن یا خطوط اجاره‌ای[1] انجام می‌شود و دریافت اطلاعات که بیشتر می‌باشد، از طریق ماهواره انجام خواهد شد.



شکل ٤-١ ارتباط متقارن با اینترنت از طریق ماهواره

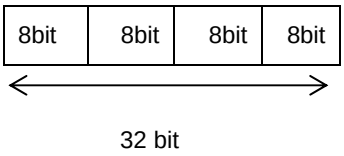
۵-۱ آدرس‌های IP

شبکهٔ اینترنت یک شبکهٔ TCP/IP می‌باشد و در یک شبکهٔ TCP/IP هر کامپیوتر شبکه چه سرویس گیرنده و سرویس‌دهنده و چه تجهیزات بین شبکه[2] مثل RAC,Router و غیره، دارای یک شناسهٔ منحصربه‌فرد می‌باشند، که به آن آدرس IP گویند.

آدرس IP یک عدد۳۲ بیتی می‌باشد، که از ٤ عدد ٨ بیتی تشکیل می‌شود و با نقطه ازهم جدا می‌شوند.

1. Leasedline
2. Inter network device

مثال : 194.224.1.2



هر آدرس IP دو بخش دارد:

۱. Net ID (مشخص کنندهٔ یک زیرشبکهٔ اینترنت)

۲. Host ID (مشخص کنندهٔ یک کامپیوتر درزیرشبکه)

در مثال بالا 194.224.1 مشخص کنندهٔ یک زیرشبکه، مثلاً www.avand.net می‌باشد و عدد 2 مشخص کنندهٔ یکی از Host های این زیر شبکه، مثلاً سرویس‌دهندهٔ وب این زیر شبکه یا یکی از کاربران این زیر شبکه می‌باشد.

کلاس‌های IP

به مجموعه IP هایی که به هر ISP یا زیرشبکهٔ اینترنت اختصاص می‌یابد، کلاس IP آن زیر شبکه گویند.

مثلاً مجموعهٔ 194.224.1.32 تا 194.224.1.63، که مشخص کنندهٔ ۳۲ تا IP از کلاس C می‌باشند (البته ۳۲ تای دوم).

اصلی‌ترین کلاس‌های IP عبارتند از:

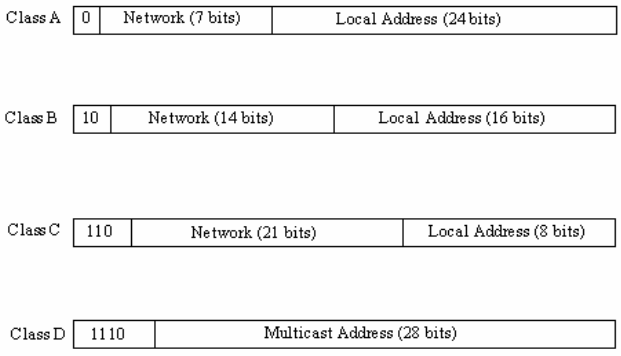
۱. کلاس A

۲. کلاس B

۳. کلاس C

البته کلاس‌های D, E نیز وجود دارند، که کلاس‌های اصلی نیستند.

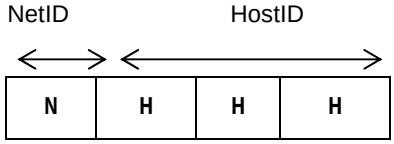
* محدودهٔ عدد اول در آدرس IP مشخص کنندهٔ نوع کلاس است.



شکل ۵-۱ کلاس‌های IP

کلاس A

بزرگ‌ترین کلاس ازنظر تعداد سرویس‌گیرنده‌ها در اینترنت می‌باشد. قالب آدرس‌ها در کلاس A به‌صورت زیر است. عدد اول در آدرس IP، NetID بوده و سه عدد آخر HostID است. با ارزش‌ترین بیت در این کلاس (MSB) 0 است.



مثال:

34.23.2.76

تعداد زیرشبکه کلاس A

دراینترنت تنها می‌توان تعداد ۱۲٦ زیرشبکه کلاس کامل A داشت، البته از آنجا که کلاس A بسیار بزرگ می‌باشد و امکان راه‌اندازی شبکه‌ای برای سرویس‌دهی (2^{24}-2)، تقریباً امکان‌پذیر نیست، کلاس‌های A به توانی از ۲ تقسیم می‌گردند مثلاً $1/2^{16}$ کلاس A و غیره

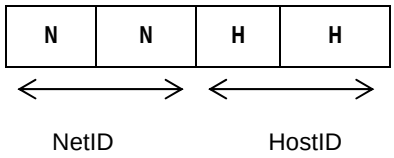
تعداد Host ها در کلاس A

به جهت غیرقابل استفاده بودن بعضی از IP های خاص برای عملیات Loopback , Multicast تعداد Host کلاس A عبارت خواهند بود از:

تعدادHost= 2^{24} -2

کلاس B

قالب آدرس‌ها در کلاس B به‌صورت زیر است. در این کلاس دو بیت با ارزش (10) می‌باشد. عدد اول همواره بین 128 تا 191 می‌باشد.



مثال :

140 . 2 . 0 . 3

NetID HostID

توجه: *باید توجه کرد که در کل، شبکهٔ NetID ثابت است، بنابراین همهٔ کامپیوترهای یک زیرشبکه، دارای Net ID ثابتی هستند و HostID تنها می‌تواند متفاوت باشد.*

جدول ۱-۱ کلاس‌ها در شبکه TCP/IP

Class	IP Address
A	1.y.z.w to
	126.y.z.w
B	128.y.z.w to
	191.y.z.w
C	192.y.z.w to
	223.y.z.w

تعداد زیرشبکه‌های کلاس B

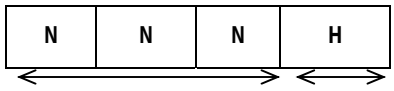
$(192\text{-}128)\ 2^{8} = 2^{14}$

تعداد Host در کلاس B

در این کلاس، از آنجا که HostID دارای ۱۶ بیت می‌باشد، بنابراین تعداد Host ها 2^{16}-2 می‌باشد.

کلاس C

این کلاس کوچک‌ترین کلاس از نظر تعداد کاربر می‌باشد. قالب آدرس‌ها به‌صورت زیر است:



NetID HostID

در این کلاس، سه بیت با ارزش عبارت است از (110) . عدد اول در آدرس بین 192 تا 223 می‌تواند باشد.
مثال :

194 . 224 . 1 . 2

NetID HostID

جدول ۲-۱ کلاس‌ها در شبکهٔ TCP/IP

IP Address Class	First Octal Minimum Maximum	Start in Binary	Number of Networks Hsots
Class A	1 - 126	1	128 - 16777214
Class B	128 - 191	10	16384 - 65534
Class C	192 - 223	110	2097152 - 254

تعداد زیرشبکه‌ها در کلاس C

تعداد زیرشبکه‌ها در اینترنت برای این کلاس بسیار زیاد می‌باشد و برابر است با:

$(224\text{-}192)2^{16} = 2^{21}$

تعداد Host ها در کلاس C

در این کلاس تعداد Host ها از همه کمتر است و برابر (2-256) می‌باشد.

تقسیم کلاس‌ها

بسته به تعداد Host های یک زیرشبکه، می‌توان یک کلاس را به‌صورت کامل استفاده کرد، یا تقسیم نمود. این تقسیم می‌تواند توانی از 2 باشد. (مثلاً $\frac{1}{2}$، $\frac{1}{4}$، .... $\frac{1}{8}$) طوری که تعداد Host بسته به کلاس به 2، 4 ،... 8 تقسیم می‌شود. می‌توان نتیجه گرفت، که همواره تعداد IP های یک زیر شبکه توانی از ۲ خواهد بود.

مثال: فرض کنید یک زیرشبکهٔ اینترنت۱/٤ کلاس C می‌باشد (۱/٤ اول)، در این صورت محدودهٔ آدرس بین 194.224. 1.0 تا 194.224.1.63 خواهد بود.
یعنی ٦٤ تا Host می‌تواند داشته باشد.

Subnet mask

تعدادی Bit می‌باشد، که با آدرس IP ،AND شده تا آدرس Subnet یا NetID به‌دست آید. باید توجه کرد این آدرس هنگام نصب سرویس‌دهنده، باید وارد گردد. بسته به نوع و تقسیم کلاس، این آدرس فرق می‌کند.

برای کلاس کامل A : 255.0.0.0
برای کلاس کامل B : 255.255.0.0
برای کلاس کامل C : 255.255.255.0

مثال : اگر آدرس 194.224.1.2 را با 0 .255.255.255 AND کنیم، آدرس 194.224.1 به‌دست می‌آید، که آدرس Subent است.

باید توجه کرد، که اگر کلاس‌های ما تقسیم شوند، Subnet mask تغییر می‌کند. مثلاً در حالتی که کلاس ما ¼کلاس C باشد Subnetmask عبارت است از 255.255.255.192 . برای محاسبهٔ Subnetmask ابتدا باید محدودهٔ IP یا Scope را تعیین کنیم، سپس از روی آن، بخش ثابت آدرس‌ها یا NetID را محاسبه نماییم. حال کافی است به جای بیت‌های NetID، یک و به جای بیت‌های HostID صفر قرار می‌دهیم، تا Subnetmask به‌دست آید، مثلاً برای محدودهٔ IP برابر:

200.200.200.194

200.200.200.255

باشد، NetID برابر 200.200.200.11xxxxxx است (عدد آخر به‌صورت باینری نمایش داده شده است) بنابراین Subnetmask برابر 255.255.255.191 خواهد بود.

IP های خاص

این‌گونه IP ها، که گروهی آنها را جزو کلاس‌های D و E معرفی می‌کنند، شامل IP های رزرو شده، Broadcast، Multicast، Loopback وغیره می‌باشند. هیچ کامپیوتری حق استفاده ازاین IPها را جهت ارتباط با اینترنت ندارد.

Broadcast

اگریک بسته به آدرس255.255.255.255 ارسال شود، به کل کاربران اینترنت خواهد رسید (البته Routerها اجازهٔ عبور این بسته را به سادگی نمی‌دهند) به این آدرس آدرس پخشی یا Broadcast می‌گوییم.

Multicast

اگر بخواهید یک بسته را به تمام کاربران یک زیرشبکه ارسال کنید، باید آن را Multicast کنید. مثلاً برای زیرشبکه‌ای با آدرس Subnet 194.224.1 مثلاً مربوط به زیرشبکهٔ دانشگاه آزاد باشد، آدرس Multicast برابر 194.224.1.255خواهد بود، بنابراین تمام کاربران زیرشبکه‌ای با آدرس NetID (194.224.1) این بسته یا پیغام را دریافت خواهند کرد.

توجه: *هنگام تقسیم کلاس‌های IP، باید توجه داشت که IPهایی را که HostID آنها به 0 یا 1 ختم شوند، می‌توانید استفاده کنید، زیرا آنها IP ها، Multicast و Loopback هستند.*

۱-۶ سرویس‌دهندهٔ خدمات اینترنتISP[1]

بعضی از زیرشبکه‌های اینترنت، عملیات و خدمات اتصال به اینترنت را برای کاربران فراهم می‌کنند، به این شرکت‌ها یا زیرشبکه‌ها، ISP گویند. البته اکثر ISP ها، خودشان امکانات وب و صفحات وب نیز دارند

1. Internet service provider

و دارای آدرس اینترنتی می‌باشند. همچنین بعضی از ISPهای اینترنت بسیاری ازصفحات ایستای[1] اینترنت را Cach می‌کنند، که این امر باعث بالا رفتن سرعت کار با آنها خواهد شد.

یک زیرشبکهٔ TCP/ IP، که طریقهٔ ارتباط و تمامی پروتکل‌های آن همانند اینترنت باشد، ولی به اینترنت متصل نباشد را اینترانت گویند و از اتصال چندین اینترانت یک شبکهٔ بزرگتر ایجاد می‌شود، که به آن Extranet گویند. حال هرکدام از آنها که به اینترنت وصل شوند، دیگر زیرشبکه‌ای از اینترنت خواهند شد.

امکانات مورد نیاز جهت راه‌اندازی یک ISP

جهت راه‌اندازی یک ISP، نیاز به یک سری امکانات نرم‌افزاری و سخت‌افزاری به‌صورت زیر است.

۱. راه اندازی یک شبکهٔ TCP / IP از طریق سیستم عامل WinNT یا Unix یا غیره

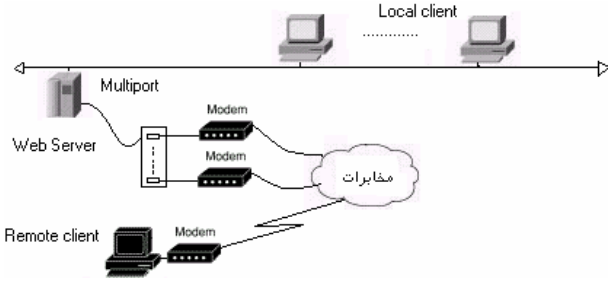
۲. فراهم کردن تجهیزاتی برای کاربران جهت ارتباط راه‌دور.

- استفاده از Multiport
- استفاده از RAC (Remote Access Controller)

الف) Multiport

Multiport یک کارت خاص می‌باشد، که روی یکی از Slot های سرویس‌دهنده نصب شده و بسته به نوع Multiport، امکان ایجاد تعداد زیادی پورت سریال را فراهم می‌کند، که از طریق یک کابل به تعدادی Box متصل می‌گردد. به هر پورت یک مودم خارجی متصل خواهد شد. (شکل ٦-١)

کارت Multiport، نیاز به یک نرم‌افزار خاص جهت سرویس‌دهی مودم‌ها و کاربران متصل شده از طریق این مودم‌ها به سایت دارد، که این نرم‌افزار یا سرویس، سرویس‌دهندهٔ راه‌دور یا RAS[2] نامیده می‌شود. هر کاربر که به سایت متصل می‌شود. ابتدا RAS از آن شناسهٔ کاربری یا Account سؤال می‌کند، اگر شناسهٔ کاربری که روی سرویس‌دهنده تعریف شده است، معتبر بود، به کاربر اجازهٔ اتصال به سایت را خواهد داد. البته باید توجه کرد که RAS در این حالت یک گلوگاه محسوب می‌شود و ایجاد اشکال در این نرم‌افزار یا نفوذ و Hack کردن آن باعث از کار افتادن کاربران راه‌دور خواهد شد.

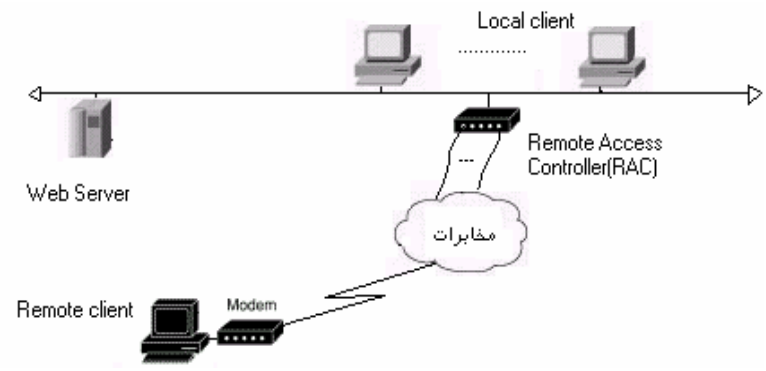


شکل ٦-١ ساختار یک شبکه Intranet با Multiport

ب) استفاده از RAC (Remote Access Controller)

1. Static
2. Remote Access Server

RAC یک سیستم مجزا است، که خود به تنهایی دارای خیلی از امکانات یک سیستم کامل می‌باشد که امکان برقراری ارتباط را فراهم می‌کند. جهت ارتباط تنها کافی است کاربران روی RAC تعریف شده باشند. خود RAC هنگام برقراری ارتباط تلفنی با آن ارتباط را برقرار کرده و از کاربر، شناسهٔ کاربری و رمز عبور سؤال می‌کند. این شناسه از طریق یک نرم‌افزار مخصوص توسط مدیر سیستم روی خود RAC تعریف شده است. باید توجه کرد که بعضی از این مسیریاب‌ها نیز امکان ارتباط راه‌دور را فراهم می‌آورند، یعنی نقش RAC را بازی می‌کنند.



شکل ۷-۱ساختار یک شبکهٔ Intranet با RAC

توجه: *امنیت و سرعت در RAC، بسیار بیشتر از Multiport است. امکان از کار افتادن نرم‌افزار کنترل کنندهٔ Multiport (RAS) و قطع شدن ارتباط همهٔ کاربران در آن وجود دارد، ولی درRAC به جهت انجام سخت‌افزاری بسیاری از عملیات امنیت و سرعت آن خصوصاً تصدیق شناسهٔ کاربری بسیار بالاتر است، البته هزینه و قیمت آن نیز بیشتر از Multiport است.*

۳. گرفتن خط اینترنت

در این حالت می‌توان از طریق خطوط اجاره‌ای یا ماهواره‌ای، سایت خود را از طریق یک مسیریاب به اینترنت متصل کرد. باید توجه کرد، که حتماً جهت اتصال دو زیر شبکه نیاز به مسیر یاب واجب است.

٤. گرفتن کلاس IP

به مجموعه IP های منحصربه‌فردی که سایت شما می‌تواند استفاده کند کلاس IP آن سایت گویند.

مثلاً IP های بین 194.224.1.31 تا 194.224.1.0

این مجموعه IP بخشی از یک کلاس C می‌باشد این مجموعه IP باید روی اینترنت معتبر[1] و منحصربه‌فرد[2] باشند. باید توجه کرد، که به ازای هر IP باید هزینه پرداخت شود. روش‌هایی وجود دارند که از طریق این روش‌ها می‌توان IP های مجازی روی سایت استفاده کرد، یکی از این روش‌ها استفاده

1. Valid
2. Unic

از نرم‌افزار Proxy است. این نرم‌افزار سه وظیفهٔ اصلی دارد: یکی ایجاد IPهای مجازی، دیگری ایجاد یک سطح امنیت و Caching.

امکانات نرم‌افزاری

بعد از فراهم کردن امکانات سخت‌افزاری، نیاز به سرویس‌هایی می‌باشد، که کاربران بتوانند به سادگی از اینترنت استفاده کنند. در مورد این سرویس‌ها بعداً صحبت خواهد شد.

۱. نصب حوزه (Domain)

۲. بسته به نیاز نصب سرویس‌های DNS و DHCP و RAS و .....

۱-۷ حوزه[1] و شرکت‌های ثبت حوزه

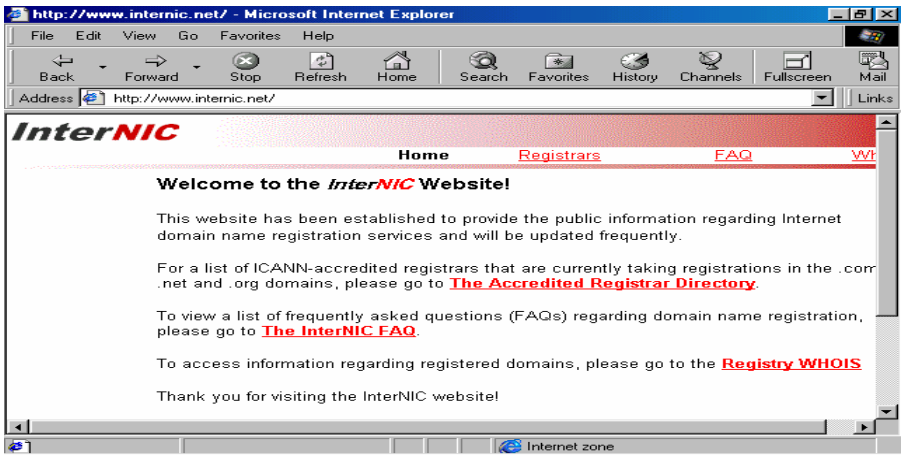
هر ISP روی اینترنت، نیاز به یک نام اینترنتی دارد، که توسط آن در اینترنت شناخته می‌شود، مثلاً Cnn.Com یا Yahoo.Com و.... به این اسامی اینترنتی حوزه گوییم.

هر سایت ISP جهت ثبت حوزه، باید با یکی از شرکت‌های ثبت حوزه ارتباط برقرار کند. سه تا از مهم‌ترین این شرکت‌ها که درواقع Host های اصلی اینترنت را دارا می‌باشند، عبارتند از: Internic، Apnic، Rips (شکل ۸-۱) که برای ارتباط با آنها روی اینترنت، می‌توان آدرس آنها را به‌صورت زیر وارد کرد.

www.internic.net

www.Apnic.net

www.Rips.net



شکل ۸-۱ سایت Internic در اینترنت

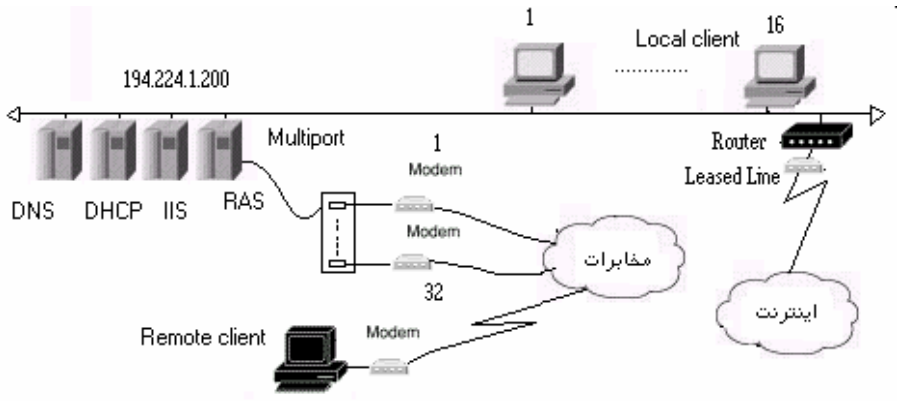
1. Domain

این سه شرکت دارای سه DNS[1] اصلی اینترنت می‌باشند، که عملیات تبدیل اسامی اینترنتی به آدرس IP، توسط آنها انجام می‌شود.

هنگامی که شما یک آدرس اینترنتی مثلاً www.altavista.com را می‌نویسید، ابتدا یک درخواست به DNS ISP شما ارسال می‌شود. DNS یک بانک اطلاعاتی دارد، اگر در آنجا معادل آدرس IP یافت شود، به کاربر (مرورگر) ارسال می‌شود، اگر نباشد با یکی از شرکت‌ها (DNS های اصلی اینترنت) تماس اینترنتی برقرار شده و معادل IP آن به‌دست آمده و سپس بسته روی شبکه ارسال خواهد شد.

مثال ۱ :

فرض کنید یک زیر شبکهٔ اینترنت، دارای چهار سرویس‌دهنده (IIS, DNS, RAS, DHCP) می‌باشد، آدرس سرویس‌دهندهٔ وب برابر 194.224.1.200 است. در این زیر شبکه، تعداد ۱۶ Local client و ۳۲ تا Remote client وجود دارد. ساختار شبکه به‌صورت زیر (شکل ۹-۱) خواهد بود.



شکل ۹ -۱

حال برای محاسبهٔ کلاس IP، ابتدا باید تعداد Host را محاسبه کرد. تعداد Host، برابر تعداد سرویس-دهنده‌ها به‌علاوه تعداد سرویس‌گیرنده‌های محلی و راه‌دور به‌علاوه تعداد تجهیزات بین شبکه‌ای مثل Router و RAC.

n=4+16+32+1=53

حال از آنجا که تعداد IP های یک سایت، همواره باید توانی از دو باشد، بنابراین اولین توان ۲ بزرگتر از n را باید حساب کرد.

K=64

حال با یک تناسب می‌توان گفت، چون کلاس ما C است (194.224.1.200) و یک کلاس C کامل، ۲۵۶ کامپیوتر می‌تواند داشته باشد، پس داریم:

1. Domain Name Server

C	256
(1/x)C	64

x=4

بنابراین کلاس IP برابر ۱/٤ کلاس C خواهد بود.

برای محاسبهٔ Submetmask نیز کافی است، ابتدا محدودهٔ IP را حساب کرده و بخش ثابت آدرس‌ها را به‌دست آوریم (NetID) سپس به جای بخش ثابت 1 و بخش متغیر (HostID) 0 قرار دهیم:

محدودهٔ IP:

194.224.1.192

194.224.1.255

توجه: *البته باید توجه کرد، که IP های اول و آخر را نمی‌توانید استفاده کنید. این IPها برای عملیات Multicast و Loopback رزرو شده‌اند.*

بخش ثابت آدرس‌ها:

194.224.1.11XXXXXX

توجه: *عدد آخر در آدرس به‌صورت باینری نوشته شده است و ٦ بیت آخر، مشخص کنندهٔ HostID می‌باشد.*

Subnet mask :

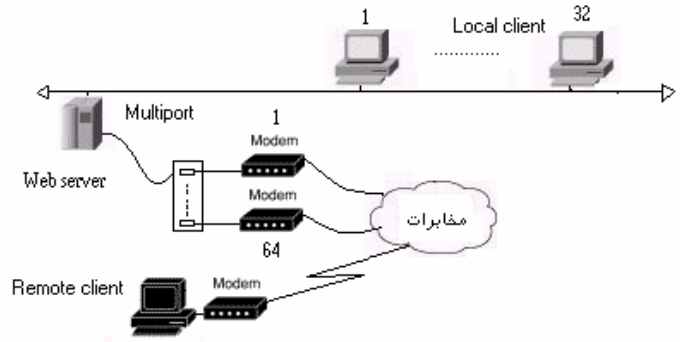
11111111.11111111.11111111.11000000

که معادل دهدهی آن عبارت است از:

255.255.255.192

مثال۲:

فرض کنید یک زیر شبکهٔ اینترانت، دارای یک سرویس‌دهندهٔ وب با آدرس 130.224.1.200 است. در این زیرشبکه، تعداد ۳۲ Local client و ٦٤ تا Remote client وجود دارد. ساختار شبکه به‌صورت زیر خواهد بود.



شکل ۱۰ -۱

حال برای محاسبهٔ کلاس IP، ابتدا باید تعداد Host را محاسبه کرد. تعداد Host برابر تعداد سرویس-دهنده‌ها به‌علاوه تعداد سرویس‌گیرنده‌های محلی و راه‌دور، به‌علاوه تعداد تجهیزات بین شبکه‌ای مثل Router و RAC می‌باشد، که البته در این زیرشبکه به جهت متصل نبودن به اینترنت وجود ندارد.

n=1+64+32=97

حال همان‌طور که گفته شد، از آنجا که تعداد IPهای یک سایت همواره باید توانی از دو باشد، بنابراین اولین توان ۲ بزرگتر از n را باید حساب کرد.

K=128

حال با یک تناسب می‌توان گفت چون کلاس ما B است (130.224.1.200) و یک کلاس B کامل 2^{16} کامپیوتر می‌تواند داشته باشد پس داریم:

B	2^{16}
$(\frac{1}{x})B$	128

$x=2^9$

بنابراین کلاس IP برابر $\frac{1}{2^9}B$ خواهد بود.

برای محاسبهٔ Submetmask نیز کافی است، ابتدا محدودهٔ IP را حساب کرده و بخش ثابت آدرس‌ها را به‌دست آوریم (NetID)، سپس به جای بخش ثابت 1 و بخش متغیر (HostID) 0 قرار دهیم:

محدودهٔ IP:

130.224.1.128

130.224.1.255

توجه: *البته باید توجه کرد که IP های اول و آخر را نمی‌توانید استفاده کنید. این IP ها برای عملیات Multicast و Loopback رزرو شده‌اند.*

بخش ثابت آدرس‌ها:

130.224.1.1XXXXXXX

توجه: *عدد آخر در آدرس به صورت باینری نوشته شده است و۷ بیت آخر مشخص کنندهٔ HostID می‌باشد.*

Subnet mask :

11111111.11111111.11111111.10000000

که معادل دهدهی آن عبارت است از:

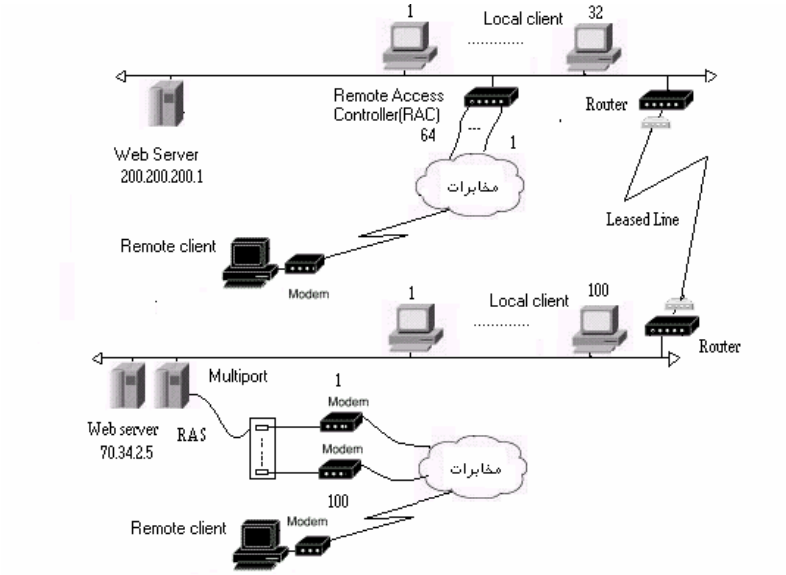
255.255.255.128

مثال۳:

فرض کنید یک Extranet، دارای دو زیر شبکه با مشخصات زیر باشد:

زیرشبکهٔ اول، دارای یک سرویس‌دهندهٔ وب، با آدرس 200.200.200.1 است. در این زیرشبکه، تعداد ۳۲ Local client و ٦٤ تا Remote client وجود دارد. کاربران راه‌دور، از طریق RAC سرویس می‌گیرند. زیرشبکهٔ دوم دارای دو سرویس‌دهندهٔ (IIS,RAS) با آدرس سرویس‌دهندهٔ وب 70.34.2.5 و ۱۰۰ تا سرویس‌گیرندهٔ محلی و ۱۰۰ تا راه‌دور می‌باشد.

ساختار شبکه به‌صورت زیر خواهد بود. (شکل ۱۱-۱)



شکل ۱۱ -۱

حال برای محاسبهٔ کلاس IP هر کدام مانند مثال‌های قبل، ابتدا باید تعداد Host را محاسبه کرد. تعداد Host، برابر تعداد سرویس‌دهنده‌ها به‌علاوه تعداد سرویس‌گیرنده‌های محلی و راه‌دور، به‌علاوه تعداد تجهیزات بین شبکه‌ای مثل Router و RAC می‌باشد. برای زیرشبکه اول داریم:

n=1+64+32+1+1=99

حال همان‌طورکه گفته شد، از آنجا که تعداد IP های یک سایت همواره باید توانی از دو باشد، بنابراین اولین توان ۲، بزرگتر از n را باید حساب کرد.

K=128

حال با یک تناسب می‌توان گفت چون کلاس ما C است (200.200.200.1) و یک کلاس C کامل 256 کامپیوتر می‌تواند داشته باشد پس داریم:

C	256
(C)	128

x=2

بنابراین کلاس IP برابر ۱/۲ کلاس C خواهد بود.

برای محاسبهٔ Submetmask نیز کافی است، ابتدا محدودهٔ IP را حساب کرده و بخش ثابت آدرس‌ها را به‌دست آوریم (NetID)، سپس به جای بخش ثابت 1 و بخش متغیر (HostID) 0 قرار دهیم:

محدودهٔ IP:

200.200.200.0

200.200.200.127

توجه: البته باید توجه کرد، که IPهای اول و آخر را نمی‌توانید استفاده کنید. این IPها برای عملیات Multicast و Loopback رزرو شده‌اند.

بخش ثابت آدرس‌ها:

200.200.200.1XXXXXXX

توجه: *عدد آخر در آدرس، به‌صورت باینری نوشته شده است و ۷ بیت آخر، مشخص کنندهٔ HostID می‌باشد.*

: Subnet mask

11111111.11111111.11111111.10000000

که معادل دهدهی آن عبارت است از:

255.255.255.128

برای زیرشبکه دوم داریم :

n=2+100+100+1=203

حال همان‌طور که گفته شد، از آنجا که تعداد IP های یک سایت، همواره باید توانی از دو باشد، بنابراین اولین توان ۲ بزرگتر از n را باید حساب کرد.

K=256

حال با یک تناسب می‌توان گفت چون کلاس ما A است (70.34.2.5) و یک کلاس A کامل 2^{24} کامپیوتر می‌تواند داشته باشد، پس داریم:

A	2^{24}
$(\frac{1}{x})$A	256

$x=2^{16}$

بنابراین کلاس IP برابر $\frac{1}{2^{16}}$A خواهد بود.

برای محاسبهٔ Submetmask نیز کافی است، ابتدا محدودهٔ IP را حساب کرده و بخش ثابت آدرس‌ها را به‌دست آوریم (NetID)، سپس به‌جای بخش ثابت 1 و بخش متغیر (HostID) 0 قرار دهیم:

محدودهٔ IP:

70.34.2.0

70.34.2.255

توجه: *البته باید توجه کرد که IP های اول و آخر را نمی‌توانید استفاده کنید. این IPها برای عملیات Multicast و Loopback رزرو شده‌اند.*

بخش ثابت آدرس‌ها:

70.34.2.XXXXXXXX

توجه: *عدد آخر در آدرس به‌صورت باینری نوشته شده است و8 بیت آخر مشخص کننده HostID می‌باشد.*

: Subnet mask

11111111.11111111.11111111.00000000

که معادل دهدهی آن عبارت است از:

255.255.255.0

توجه:*جهت اتصال این شبکه به اینترنت کافی است، یکی از زیرشبکه‌ها را به اینترنت متصل کرد. در این حالت نیاز به یک Router و خط اینترنت می‌باشد.*

۸-۱ تعریف سایت وب

یک سایت وب از یک یا چند صفحه تشکیل می‌شود، که به هم پیوند شده‌اند. طوری که ممکن است این صفحات همه در یک سایت بوده و یا در هر کجای اینترنت باشند. تنها کافی است صفحهٔ اول در آن سایت باشد.

Home page , Welcome page

اولین صفحه یا صفحهٔ آغازین هر سایت وب را Welcome page گویند، که با نوشتن نام حوزه ظاهر می-شود. مثلاً با نوشتن آدرس www.cnn.com صفحهٔ اول این سایت ظاهر خواهد شد.

Home page مجموعهٔ صفحات مربوط به یک شخص، شرکت یا مؤسسهٔ خاص از یک سایت وب را گویند. مثلاً Home page مربوط به یک دانشگاه یا یک شخص خاص در سایت وب Neda . البته باید توجه کرد که در خیلی از حالات Welcome page و homepage یکی هستند، مثلاً آدرس www.yahoo.com هم Welcome pageو همhomepage سایت Yahoo می‌باشد، ولی مثلاً آدرس www.yahoo.com/mehdi آدرس homepage شخصی به نام Mehdi در Yahoo است و Welcome page نیست.

اطلاعات موجود در یک Welcome page

Welcome page از آنجا که اولین صفحه می‌باشد که برای کاربر ظاهر می‌گردد از این جهت باید بسیار جذاب و گیرا باشد، طوری که کاربر را برای دیدن سایر صفحات آن سایت تشویق کند.

بهتر است صفحهٔ اول یک سایت خیلی سنگین نباشد، تا سریع بار گردد. اطلاعات موجود در یک Welcome page بهتر است مطالب زیر باشد.

۱. فهرست یا جدولی از عناوین و مطالب موجود در سایت.

۲. دارای آرم و یا شناسهٔ آن مؤسسه باشد (هم می‌تواند به‌صورت زمینهٔ کمرنگ زیر متن باشد و هم می‌تواند دربالای صفحه قرار گیرد).
۳. دارای تعدادی پیوند به صفحات دیگر مربوط به آن صفحهٔ خاص باشد.
*بهتر است در هر صفحه یک مسیر بازگشت به Welcome page وجود داشته باشد.

عملیات لازم قبل از ایجاد سایت وب

بعد از فراهم کردن تجهیزات سخت‌افزاری قبل از شروع به برنامه‌نویسی، بد نیست که عملیات زیر را انجام دهید:
۱. صفحهٔ وب خود را روی چندین سکوهای سخت‌افزاری[1] تست کنید.
۲. وقتی شروع به برنامه‌نویسی می‌کنید، مطمئن شوید که سرویس‌دهنده در امن‌ترین حالت خود قرار دارد.
۳. به‌طور غیرمجاز به سایت خود وارد شوید. این کار را با یک Pc روی شبکه و از طریق Telnet و Ftp یا ... انجام دهید. از طریق یک شناسهٔ کاربری Dial-up از راه‌دور نیز این کار را تکرار کنید.
٤. شروع به جستجو روی گزارش اشکالات و خطاها روی سرویس‌دهنده‌ها و مرورگرهای اینترنت نمایید.

(Uniform Resource Locator) URL

همان‌طور که گفته شد، هر کامپیوتر سرویس‌دهندهٔ وب دارای یک نام اینترنتی منحصربه‌فرد روی اینترنت می‌تواند باشد، که به آن حوزه گوییم. هر صفحه روی اینترنت نیز دارای یک نام اینترنتی منحصربه‌فرد می‌باشد، که به آن URL گوییم. به عنوان مثال:

http://www. cnn.com /sample / main.htm

باید توجه کرد که آدرس www.cnn.com به یک آدرس زیرشاخهٔ خاص روی سرویس‌دهنده اشاره دارد، که به آن زیرشاخهٔ ریشه گویند. در سیستم‌های مختلف، زیرشاخهٔ ریشه متفاوت می‌باشد. در سیستم Windows NT این زیرشاخه به‌صورت پیش‌فرض عبارت است از:

C:\Inetpub\wwwRoot\

حال اگر URL به‌کار رفته توسط کاربر، دارای چندین فهرست نیز باشد، حتماً در فهرست wwwroot این فهرست‌ها باید موجود باشند.

هر URL کامل می‌تواند اجزای زیر را داشته باشد.
۱. پروتکل مورد استفاده جهت انتقال سند می‌تواند (http://, ftp:// , file:// ,gopher://.....) باشد.
۲. نام سرویس‌دهندهٔ وب یا حوزه مثلاً www.yahoo.com (ممکن است آدرس IP سرویس‌دهنده وب باشد).
۳. فهرست روی سرویس‌دهندهٔ وب (ممکن است چندین فهرست تودرتو به‌کار رود).

1. Plateform

٤. نام صفحه.

http : // www. sample.com / samples / sampele.html

↓ ↓ ↓ ↓

transfer protocol type Specific host computer Directory on hast Web page filename

باید توجه کرد، که اگر نام فایل ذکر نگردد، سرویس‌دهندهٔ وب به‌صورت پیش‌فرض به دنبال فایل‌های index.htm و یا Default.htm بسته به سیستم عامل خواهد گشت و اگر پیدا کند، آنها را برای سرویس‌گیرنده ارسال می‌نماید.

پسوندهای[1] حوزه

این پسوندها هم می‌تواند مشخص کنندهٔ کشور مربوطه و یا نوع سازمان مربوطه باشند، مثلاً :

پسوند	معنی
Com	Commercial
Edu	Education
Gov	Government
Org	Organization
Net	Network provider
AC	Academic
Ca	Canada
Au	Australia
Ir	Iran
.	.
.	.

باید توجه کرد که ممکن است این پسوندها با هم ترکیب نیز گردند، مثلاً www.iau.edu.ir

۹-۱ زبان‌های نشانه‌ای[2]

زبان‌های نشانه‌ای، پایه و اساس تولید صفحات وب می‌باشند. این زبان‌ها دارای تعدادی دستورالعمل، یا برچسب[3] می‌باشند که به سند اضافه شده و به مرورگر طریقهٔ نمایش سند را از نظر فونت، رنگ و اندازه و سایر موارد نشان می‌دهند.

معروف‌ترین زبان نشانه‌ای، زبان HTML (Hyper Text Markup Language) یا زبان نشانه‌ای فرامتن می‌باشد. زبان‌های VRML (Virtual Reality Modeling Language)و XML (Extensibl Markup Language)، نیز امروزه برای تولید صفحات وب استفاده می‌شوند. زبان HTML به‌طور مفصل شرح داده خواهد شد.

VRML، نیز یک زبان مدل‌سازی برای وب می‌باشد، که برای شبیه‌سازی محیط‌ها و اجسام می‌تواند استفاده شود. ازطریق آن می‌توان محیط‌هایی مثل یک اتاق یا یک هواپیما و خانه و... را شبیه‌سازی کرد،

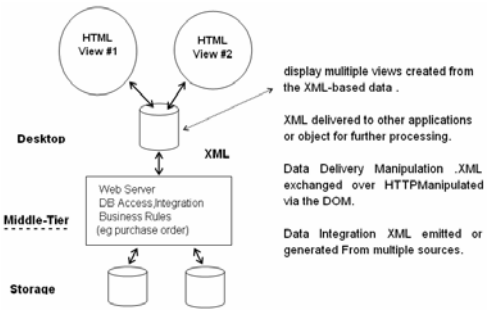
1. Suffix
2. Markup
3. Tag

که کاربر بتواند در آن محیط حرکت کرده و ارتباطی متقابل[1] با محیط داشته باشد. باید توجه کرد، که در این حالت کامپیوتر کاربر (سرویس‌گیرنده) باید قدرتمند باشد، زیرا به‌طور مرتب در حال Render کردن تصاویر سه بعدی می‌باشد، تا محیطی کاملاً واقعی را برای کاربر شبیه‌سازی نماید. فایل‌های VRML دارای پسوند WRL می‌باشند، که توسط مرورگرهای معمولی قابل نمایش نیستند. برای اینکه یک مرورگر بتواند این فایل را نمایش دهد، نیاز به نصب یک برنامهٔ اتصالی[2] خاص می‌باشد، که امکان نمایش این فایل‌ها را فراهم می‌کند. یکی از این برنامه‌های اتصالی، Cosmo Player می‌باشد. برنامه‌نویسی با VRML بسیار دشوار است، به همین جهت بیشتر از تولید کننده[3]های این زبان برای ایجاد این صفحات استفاده می‌شود.

XML، نیز یک زبان مبتنی بر Tag براساس (SGML) است، که برای نقل و انتقالات دنیای وب، بهینه-سازی شده است. XML، در واقع دید شیءگرا به HTML داده است و قرار است جایگزینی برای HTML باشد.XML دارای ویژگی‌های بسیار زیادی می‌باشد. از جمله اینکه در آن، امکانات برنامه‌نویسی و ایجاد TAG وجود دارد. مثلاً کاربر می‌تواند یک برچسب به نام <Test> ایجاد نماید و گرامر آن را تعریف نماید و از این به بعد مرورگر می‌تواند این TAGرا بشناسد.

یکی از مهم‌ترین برتری‌های XML، در آن است که به نحو بسیار مؤثری محتوای یک صفحه وب را از چگونگی نمایش آن جدا نموده است. به همین دلیل هنگامی که محتوا نیاز به روزرسانی داشته باشد، اما نیاز به ذخیرهٔ طولانی مدت نداشته باشد، استفاده از مدل ارائه شدهٔ XML ایده‌آل است.

ایدهٔ اصلی که XML بر آن بنا شده است، چندان جدید نیست: داده باید به فرم استاندارد مبادله شود. چنین فرآیندی در قراردادها و تقاضاهای خرید در عمل به وفور رخ می‌دهد.



شکل ۱۲-۱ معماری XML

به وسیلهٔ XML، یک برنامه‌نویس می‌تواند داده را از طریق یک ساختار داده‌ای منطقی[4] که مستقل از تمام پیاده‌سازی‌های بخش پیشین[5] می‌باشد، مورد استفاده قرار دهد، یا آنها را خلق نموده و تغییر دهد.

1. Intractive
2. Plugins
3. Generator
4. Logical data strucure
5. Back-end

منابع داده‌ای چندگانه[1] می‌توانند داده را به یک ساختار XML واحد تغذیه نمایند. چنین فرایندی امکان تجمع سیستم‌های متفرق را تحت یک سقف فراهم می‌آورد. از سوی دیگر، از آنجا که XML در قالب متنی ارائه می‌شود، تجمع سیستم‌های متفرق می‌تواند برروی وب از طریق HTTP منتقل گردد. (شکل ۱۲-۱)

به عبارتی شاید مهم‌ترین دلیل پدید آمدن XML، اعمال نوعی ساختار شیءگرا به صفحات HTML بوده باشد.

مثال زیر، چگونگی اختلاف دیدگاه‌ها در مواجهه با یک سند HTML و XML را نشان می‌دهد:

مثال ۱ :

```
<HEAD>
<TITLE> Printer</TITLE>
</HEAD>
<BODY>
<H1> Computer source is wizbang 3000 dot matrix printer</H1>
Features:
40 Pages per Minute
$200.00
10 1bs.
</BODY>
```

حال اگر این برنامه را بخواهیم به‌صورت XML بنویسیم، داریم:

```
<?XML Version=”1.0”?>
<MANUFACTURER>Computer source
   <PRODUCT>
      <CLASS>Printer
      <TYPE> dot Matrix</TYPE>
      </CLASS>
   <NAME>Wizbang3000</NAME>
      <EEATURES>
      <SPEED Units=”ppm”>40</SPEED>
      <QUALITY Units=”dpi”>60</QUALITY>
      </EEATURES>
      <PRICE Units=”USD”>
      <RETAIL>200</RETAIL>
      <WHOLESALE>110</ WHOLESALE>
       <PRICE>
   <WEIGHT  Units=”1bs”>10</WEIGHT>
   </PRODUCT>
```

1. Multiple data sources

```
</MANUFACTURER>
```

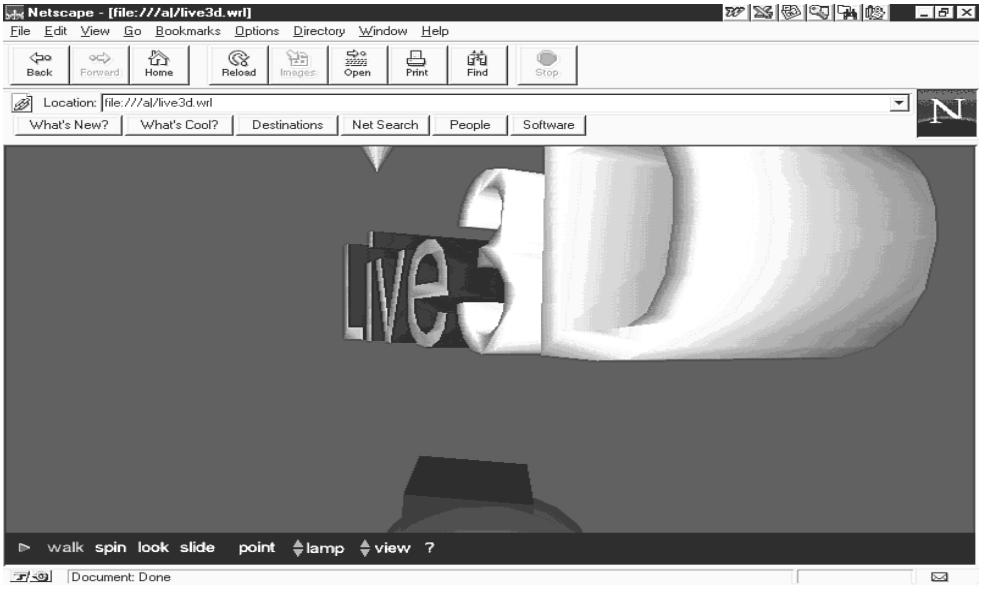
در مثال اول، در ارتباط با محصولی مانند چاپگر، توضیحاتی با زبان HTML ارائه شده است. تشریح همان محصول از XML نیز در مثال دوم آمده است.

همان‌گونه که در این مثال‌ها آشکار است، XML بر تشریح محتوای سند استوار است، در حالی که HTML، ساختار سند و Layout را مدنظر قرار می‌دهد، درعین حال هر دو زبان از روشی مبتنی بر Tagها برای تشریح نام‌ها و ویژگی‌ها استفاده می‌نمایند.

مثال فوق به خوبی ماهیت شیءگرای XML را آشکار می سازد. در این مثال، <Product> فرزند <Manufacturer> است. چنین تقسیم‌بندی ازنظر ساختاری، کاملاً عقلانی است، به جهت آنکه اگر کارخانه-ای دارای محصولات متعدد بود، می‌توان حجم کدهای نوشته شده را به نحو در خور توجهی کاهش داده و علاوه بر آن خوانایی برنامه را افزایش داد.

از دیگر ویژگی‌های منحصربه‌فرد XML آن است، که می‌توان دیدگاه‌های داده‌ای[1] متفاوتی را بسته به نیاز مشتریان تعریف نمود و آنگاه برای هر یک از مشتریان، کاتالوگ‌های متفاوتی ارائه نمود.

XML، برتری‌های فراوانی (نسبت به HTML) دارد به گونه‌ای که در آیندهٔ نزدیک تا حدود بسیاری جای HTML را خواهد گرفت، اما نمی‌توان ازنظردور داشت، که درحال‌حاضر XML، توسط مرورگرهای بسیار کمی حمایت می‌شود. (از جمله Internet Explorer5) لذا XML، در واقع بیشتر طرحی برای آینده است. (شکل ۱۳-۱)



شکل ۱۳-۱ نمایش یک فایل VRML

1. Data Views

۱۰-۱ پروتکل‌های اینترنت

پروتکل TCP/IP، اصلی‌ترین پروتکل شبکهٔ اینترنت است. TCP/IP برروی LANهایی، که براساس توپولوژی اترنت و Tokenring بسته شده‌اند، کار می‌کند و همچنین قابلیت ایجاد سرویس برای مشترکین Dial–Up را نیز دارد، اما قبل از اینکه به ادامهٔ بحث بپردازیم، لازم است کمی در مورد چگونگی کار کرد پروتکل TCP/IP توضیح داده شود. پروتکل TCP/IP دارای پنج لایه (در بعضی کتاب‌ها دو لایهٔ پایین را ترکیب کرده و به ٤ لایه کاهش می‌دهند) می‌باشد، که وظایف هر لایه مشخص است. سیستم حوزه در اینترنت، به کاربران امکان دسترسی ساده‌تر به اینترنت را می‌دهد، گرچه آدرس‌های اینترنت برای یادآوری آسان نیستند، اما استفاده از آنها بسیار ساده‌تر از روش کامپیوترها برای اشاره به همان محل است، کامپیوترها در اینترنت ازیک سیستم مبتنی براعداد برای آدرس‌دهی استفاده می‌کنند. سیستم آدرس‌های اسمی، که در بالا مورد بحث قرار گرفت به آدرس عددی تبدیل می‌شوند و سپس دسترسی صورت می‌گیرد. این کار اندکی شبیه ذخیره کردن شماره تلفن‌ها در یک حافظهٔ تلفن با استفاده از اسم صاحبشان می‌باشد. در این حالت برای گرفتن شماره تلفن یک شخص خاص، کافیست اسم وی را یافته و دکمهٔ مربوطه را فشار دهید و نیازی نیست که شمارهٔ وی را بدانید. این همان راهی است که پروتکل اینترنت طبق آن عمل می‌کند.

سیستم‌های کامپیوتری در اینترنت، یک جدول تبدیل اسم به عدد دارند و از آن برای فرستادن اطلاعات به مقصد مناسب استفاده می‌کنند. در اینترنت، اجزای متفاوتی وجود دارند؛ شبکه‌های شرکتی، آموزشی، شبکه‌های منطقه‌ای و ایستگاه‌های کار اختصاصی، تمام این اجزاء از طریق ستون فقرات با خطوط تلفنی یا اختصاصی به یکدیگر متصل شده‌اند. می‌بینید که بدون یک سیستم آگاه از اسامی و آدرس‌ها غیرممکن خواهد بود، که اطلاعات را به محل دلخواه فرستاد. مدیریت این کار توسط مسیریاب-های کامپیوتری، که در اینترنت مشغول به کار هستند و ضمن نظارت بر نقل و انتقال به مطالعهٔ آدرس-های بسته‌ها می‌پردازند، انجام می‌شود. برای اتصال هر دو زیرشبکه، مسیریاب لازم است. پروتکل اینترنت، داده‌ها را با گروه‌بندی آنها به‌صورت یک بسته، از نقطه‌ای به نقطه‌ای دیگر می‌فرستد. یک بسته عبارت است از یک مجموعه از کاراکترها که کمتر از 1500 بیت است، که همهٔ آنها به یک مقصد می‌روند به عبارت دیگر همهٔ آنها دارای آدرس اینترنت مشابه هستند. بسته‌ها به روش نردبانی از یک سری کامپیوتر می‌گذرند، تا در نهایت به مقصد برسند. در روش آدرس‌دهی عددی، هر آدرس شامل حداکثر چهار عدد است، که با نقطه از یکدیگرجدا شده‌اند و همهٔ آنها از 256 کمترند. مثلاً سرویس‌دهندهٔ Archie (Archie.Unl.Edu) دارای آدرس IP به صورت 129.93.1.14 است. برخی از اعداد در این آدرس مشخص می‌کند، که کدام شبکه در اینترنت باید بسته را دریافت کند. وقتی این شبکه بسته را گرفت، یک مسیریاب دیگر، بسته را تا مقصد بعدی می‌فرستد و الی آخر تا کامپیوتر مشخص در انتها بسته را بگیرد. در آنجا بسته شکسته شده، یا در یک فایل ذخیره می‌شود، یا روی مانیتور نمایش داده می‌گردد.

جدول ۳-۱ لایه‌های مختلف مدل TCP/IP در مقایسه با OSI

OSI Layers	TCP/IP Layels	TCP/IP Protocols & Components	
Application	Application	Application Protocols	
Presentation			
Session			
Transport	Transport	TCP	UDP
Network	Internet	IP	ARP
		ICMP	RARP
Data Link	Network Interface	Your Network Sofware & Network Card	
Physical	Hardware		

این سیستم مبتنی بر اعداد برای صحبت کردن کامپیوترها با یکدیگر بدون اشتباه کردن آدرس و بدون سر در گم شدن در سیستم‌های شبکه، ضروری است. اما برای ما سیستم اسمی راحت‌تر است و برای دسترسی‌ها به منابع اینترنت کافی است. با این حال یادگرفتن مبانی پروتکل اینترنت برای فهمیدن اینکه اطلاعات چطور از این سوی اینترنت به آن سو می‌روند، خوب است.

پروتکل اصلی اینترنت، TCP/IP است، ولی در هر کدام از لایه‌های آن، پروتکل‌های مختلفی مطرح می‌شوند اصلی‌ترین پروتکل‌ها در لایهٔ کاربرد عبارتند از FTP، Gopher، Http. که البته HTTP امروزه عملیات مربوط به FTP و Gopher را نیز تحت پوشش قرار داده است.

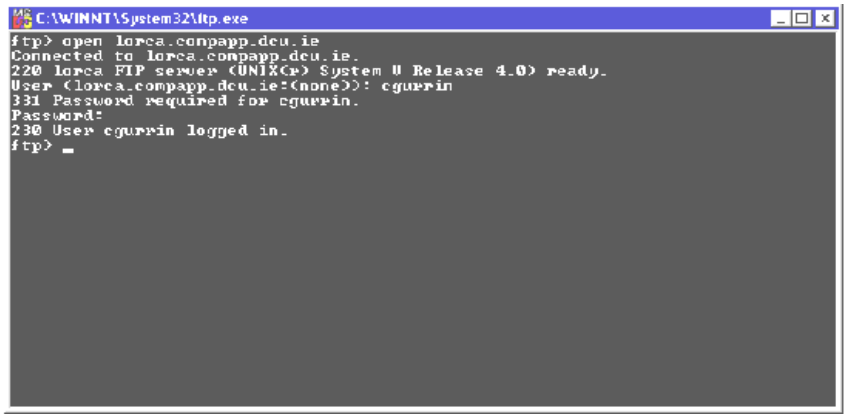
پروتکل انتقال فایل[1] FTP

این پروتکل جهت انتقال فایل به‌کار می‌رود. از همان ابتدای ایجاد اینترنت در آن وجود داشت و بعضی سایت‌ها این خدمات را ارائه می‌دادند. دستوری در آنها وجود داشت به‌نام ftp، که همانند شکل می-توانستیم ازطریق آن به سایت‌های ftp متصل شده و خدمات انتقال (Download و upload) را انجام داد.

وظیفهٔ اصلی و خصوصیت اصلی FTP این است، که از طریق آن می‌توان به سایت‌هایی که دارای شناسهٔ کاربری می‌باشند، از طریق شناسهٔ کاربری anonymous و رمز عبور آدرس پست الکترونیکی، متصل شد و از امکانات آنها (با حد اقل دسترسی) استفاده کرد. در مرورگرهای گرافیکی نیز می‌توانید در بخش آدرس از آدرس‌های ftp مثل زیر استفاده کنید.

ftp :// ftp.vax.orgیاftp ://ftp.dec.com

1. File Transfer protocol



شکل ١٤-١ صفحهٔ مربوط به دسترسی به یک سایت FTP

در این صورت، خود مرورگر به‌صورت یک میانجی عمل کرده و به‌صورت anonymous با کمترین دسترسی ارتباط شما را با سایت برقرار می‌کند. در این حالت، مرور گر دستورات FTP را شبیه‌سازی می‌کند و نیازی به وارد کردن User ID نیست و فهرست ریشهٔ سایت ftp را نشان می‌دهد. در این حالت به سادگی می‌توانید فایل‌ها را برای خود کپی کرده و یا فایل‌هایی را روی سایت (در صورت اجازه) کپی (Upload) کنید.

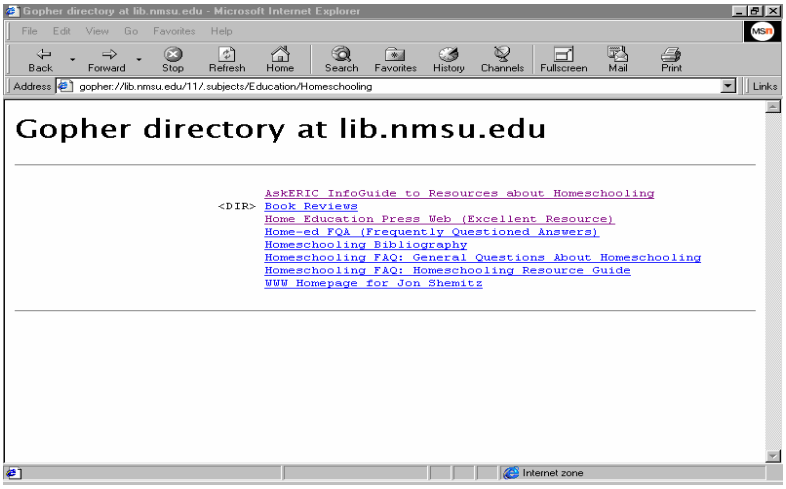
گوفر (Gopher)

گوفر به معنی سنجاب می‌باشد. این پروتکل نیز همانند ftp، یک پروتکل بسیار قدیمی روی اینترنت می‌باشد، که از همان ابتدا روی اینترنت متنی (Unix) وجود داشت و مانند ftp جهت استفاده از آن نیاز به ورود شناسهٔ کاربری می‌باشد.

ازطریق سایت‌های گوفر، امکان جستجو روی اینترنت (فقط در همان سایت) وجود دارد. طریقهٔ جستجو در سایت گوفر بدین صورت است، که در سایت تعدادی عناوین اصلی وجود دارد که بسته به موضوع مورد جستجو، یکی را انتخاب کرده و به‌صورت سلسله مراتبی آنقدر پیش می‌رویم، تا به مستندات موردنظر خود برسیم، مثلاً برای یافتن مقالات در مورد مدیریت حافظه باید به‌صورت زیر به موضوع موردنظر دسترسی پیدا کنیم.

مثلاً:

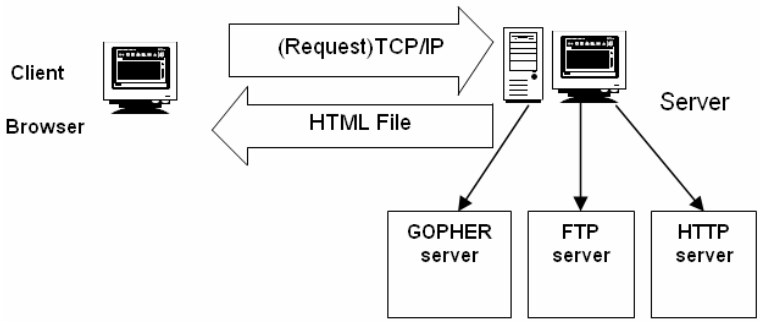
Engineering
↓
Computer
↓
Software
↓
Operating system
↓
Memory management



شکل۱۵-۱ سایت Gopher

پروتکل فرامتن[1] HTTP

HTTP یک پروتکل انتقال فرامتن می‌باشد، پروتکلی است که پایه و اساس وب را شامل می‌شود. HTTP یک پروتکل غیروابسته به حالت[2] است. به این معنی که هیچ‌گونه اطلاعاتی از چگونگی انتقال و مسیریابی و... روی سیستم سرویس‌گیرنده نگهداری نمی‌کند.



شکل۱۶-۱ عملکرد پروتکل TCP/IP

مراحل و نحوهٔ عملکرد پروتکل HTTP

هر سرویس‌دهندهٔ وب دارای تعدادی پورت می‌باشد، که آنها واسط بین لایهٔ TCP/IP و برنامهٔ سرویس-دهندهٔ آن پروتکل مثل سرویس‌دهنده Http ، Ftp یا Gopher می‌باشد. باید توجه کرد که همین پورت‌ها راه نفوذ برنامه‌های مخرب، مثل Hacker ها یا Craker ها به سیستم نیز هستند.

1. Hyper Text Transfer Protacol
2. State less

مراحلی که هنگام نوشتن یک آدرس اینترنتی یا زدن کلید موس روی یک پیوند ایجاد می‌شود به‌صورت زیر است.

۱. اگر روی پیوند کلیک شود، دراین صورت اطلاعات مربوط به آدرس صفحه و حوزه در یک بسته قرار گرفته و به برنامهٔ مدیریت پروتکل TCP / IP، که روی سیستم نصب است، ارسال می‌گردد.
۲. در لایهٔ TCP، برنامه مدیریت پروتکل TCP / IP یک فیلد نوع پروتکل به بسته اضافه می‌کند، که نام پروتکل در آنجا نوشته شده است. مثلاً http , ftp و غیره.
۳. در لایهٔ IP، بسته به‌صورت استاندارد با ساختار TCP / IP در آمده و روی خط فیزیکی انتقال می‌یابد.
٤. ابتدا روی ISP عملیات تبدیل آدرس اینترنتی به آدرس IP، توسط DNS صورت می‌گیرد. اگر آدرس در آنجا موجود نبود با DNS های اصلی اینترنت مثل Internic و غیره ارتباط برقرار می‌شود. آدرس حتماً باید در آنجا موجود باشد (اگر موجود نبود آدرس نادرست می‌باشد)، سپس یک بستهٔ کامل آماده ارسال روی اینترنت، ساخته می‌شود.
٥. اگر خط شما IP باشد، بسته از مسیریاب‌ها عبور کرده تا به مقصد برود و اگر خط IP نباشد حتماً از یک Gateway عبور کرده، تا به پروتکل مربوطه تبدیل شود.
٦. در مقصد در لایهٔ TCP، سرویس‌دهنده پروتکل بیرون کشیده شده و Packet براساس نوع پروتکل درخواستی به یکی از پورت‌ها سرویس‌دهندهٔ وب ارسال می‌شود. برای هر پورت روی سرویس‌دهندهٔ وب، یک برنامهٔ مقیم در حافظه وجود دارد، که سرویس‌دهندهٔ آن پروتکل می‌باشد، مثلاً سرویس‌دهندهٔ HTTP.
۷. ابتدا سرویس‌دهندهٔ HTTP آدرس URL به‌دست آمده را به آدرس فیزیکی تبدیل کرده و به دنبال فایل مربوطه روی سرویس‌دهنده می‌گردد.
۸. اگر فایل یافت نشود، یک سند خطا، سرویس‌دهندهٔ HTTP ساخته و ارسال می‌کند.
۹. اگر فایل یافت شود، ابتدا سرویس‌دهندهٔ HTTP نوع آن را تشخیص داده (از روی پسوند آن) و یک سرآیند[1] با نام (Content-Type) ساخته و به مرورگر ارسال می‌کند (مرورگر از روی این سرآیند می‌فهمد که چه اطلاعاتی در حال ارسال به آن می‌باشد و خود را آمادهٔ گرفتن اطلاعات می‌کند).
۱۰. پاسخ که ممکن است یک سند HTML یا هر برنامهٔ دیگری باشد، به لایهٔ TCP برگردانده می‌شود. لایهٔ TCP روی سرویس‌دهنده، وظیفهٔ بسته‌بندی[2] کردن اطلاعات و چسباندن سرآیند پروتکل برای هر سند را برعهده دارد.
۱۱. در لایهٔ IP، هر بسته به قالب استاندارد TCP / IP در آمده و روی خط فیزیکی ارسال می‌گردد.
۱۲. در سرویس‌گیرنده، عمل جمع‌آوری اطلاعات[3] در لایهٔ TCP انجام شده و یک صفحهٔ کامل به مرورگر جهت نمایش ارسال خواهد شد.

1. Header
2. Packetize
3. Assemble

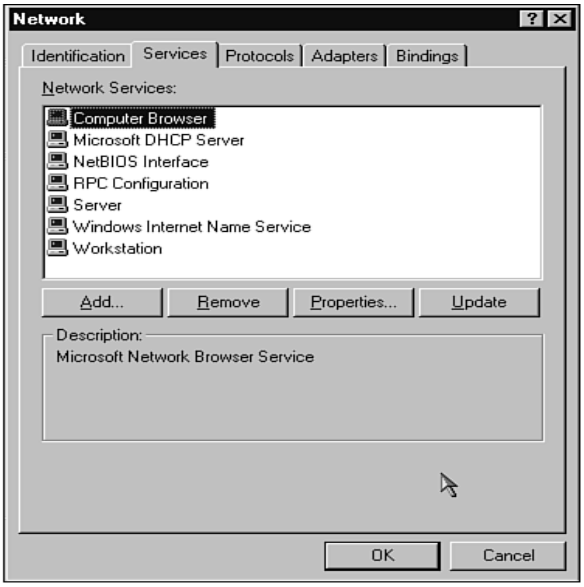
۱۳. اگر سرویس‌گیرنده بعد از دریافت فایل قادر به نمایش آن نباشد، آن را روی سیستم ذخیره و یا از یک برنامهٔ کمکی دیگر برای نمایش (PlugIns) استفاده می‌کند.
۱٤. اگر یک تصویر یا صفحهٔ دیگری (Frame ها) همراه صفحهٔ ما باشد، دوباره یک درخواست جدید ساخته شده و همهٔ این مراحل تکرار خواهند شد (State less).

۱۱-۱ سرویس‌های مورد نیاز جهت راه‌اندازی یک سایت وب

در این بخش، طریقهٔ ایجاد سرویس‌دهندهٔ وب در Win NT را شرح خواهیم داد، به‌صورت پیش‌فرض، هنگامی که Win NT را نصب می‌کنیم، این سیستم سرویس‌دهندهٔ وب نیست، ولی به کمک تعدادی از سرویس‌ها در Control panel و Network می‌توان آن‌را به یک سرویس‌دهندهٔ وب تبدیل کرد، طوری که اگر کاربر از روی همان سرویس‌دهنده و یا یک سرویس‌گیرندهٔ محلی روی Bus، آدرس IP یا حوزه را بنویسد، بتواند صفحات موجود در سرویس‌دهندهٔ وب را ببیند.

مهم‌ترین سرویس‌های مورد نیاز جهت ایجاد یک سایت وب عبارتند از:

- IIS
- DNS
- DHCP
- RAS



شکل ۱۷ -۱ سرویس‌های Win 2000

(Internet Information service) IIS

سرویسی جهت ایجاد سرویس‌دهندهٔ وب می‌باشد. IIS در Windows NT Option Pack، نیز قرار داده شده است. در کل IIS، یک محیط برنامه‌نویسی و ارائهٔ خدمات را برای نوشتن کاربردهای وب اینترنتی ارائه می‌نماید.

از آنجا که در بین ویرایش‌های مختلف IIS ، از ویرایش 4 به بعد، تغییرات محسوسی در فرایند کاری آن پدید می‌آید، توضیحات ما در حقیقت بر مبنای نسخه‌های 4 و 5, IIS می‌باشد. موارد زیر به همراه IIS4 ارائه شده‌اند:

١. سرویس‌دهندهٔ WWW
٢. سرویس‌دهندهٔ FTP
٣. سرویس‌دهندهٔ تراکنش Microsoft (MTS)
٤. سرویس‌دهندهٔ Microsoft SMTP
٥. سرویس‌دهندهٔ Microsoft MNTP
٦. سرویس‌دهندهٔ Microsoft NNTP
٧. سرویس‌دهندهٔ Index Microsoft
٨. سرویس‌دهندهٔ Microsoft Certificate
٩. Microsoft Server Express

شکل١٨-١ نصب IIS و مشخص کردن فهرست‌های ریشه

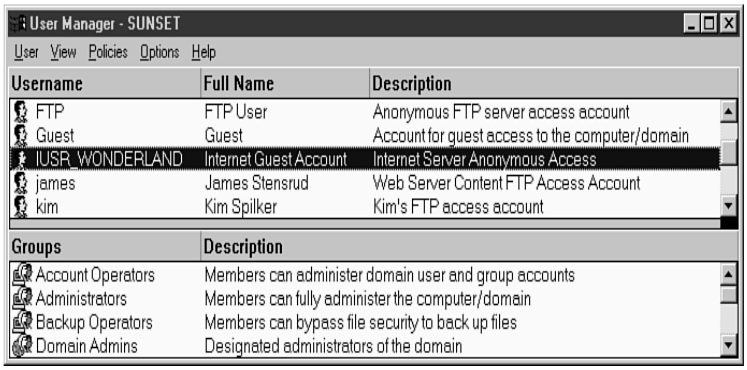
با استفاده از سرویس فوق، می‌توان سایت‌های وب، سایت‌های Ftp، گروه‌های خبری، سرویس‌دهندهٔ خدمات پستی، دسترسی امن کاربر و ویژگی‌های مفیدی در محیط‌های برنامه‌نویسی را ایجاد نمود. (شکل ١٨-١)

در IIS موارد زیر نیز وجود دارند:

۱. سرویس ارتباط با اینترنت برای RAS (Internet Connection Service for RAS)
۲. سرویس‌دهندهٔ صف پیغام Microsoft (Microsoft Message Queue Server)
۳. میزبان اسکریپت‌نویسی Windows (Windows Scripting Host)

جهت نصب IIS به‌صورت زیر عمل می‌کنیم:

۱. نصب IIS از طریق Controlpanel و Network و Services است.
۲. با برنامهٔ Internet service Manager در منوی Microsoft Internet ، Program می‌توان آن را اجرا کرد. (شکل ۱۹-۱)



شکل ۱۹-۱ شناسهٔ کاربردی اینترنت در IIS

هنگامی که IIS نصب شود، یک شناسهٔ کاربری به نام IUSR-SERVER_ NAME ساخته می‌شود.

نام سرویس‌دهندهٔ NT

برای اینکه همهٔ افراد بتوانند صفحات موجود در فهرست ریشهٔ سرویس‌دهندهٔ وب wwwRoot را ببینند، باید به این شناسه کاربری حقوق دستیابی به این فهرست را بدهید. با نصب IIS از زیر شاخه‌های مربوط به ریشهٔ وب ، Ftp ، Gopher را سؤال می‌کند، که به‌صورت پیش‌فرض عبارتند از :

c:\Inet pub\wwwroot\

c:\Inet pub\ ftproot\

c:\Inet pub\GopherRoot\

البته می‌توان آنها را تغییر داد (در شکل قبل این فهرست‌ها تغییر یافته‌اند).

(Domain Name Service) DNS

در اکثر شبکه‌های بزرگ، یک سرویس‌دهنده به نام Name server داریم، که وظیفهٔ انجام عملیات Namming (یعنی تبدیل اسم به آدرس) را برعهده دارد.

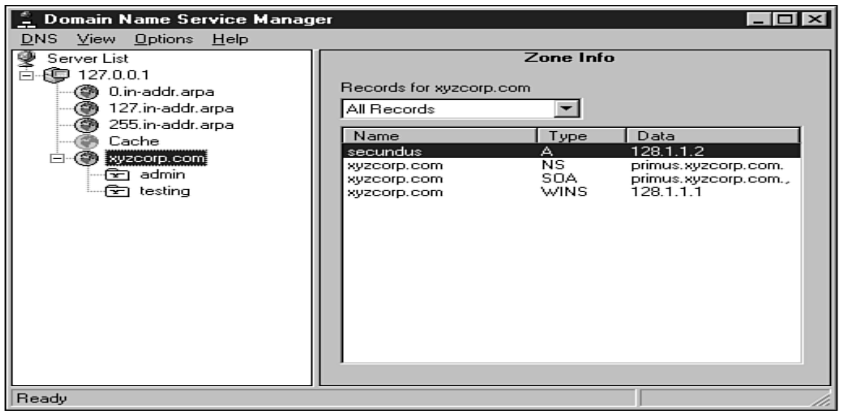
DNS در یک شبکهٔ TCP/IP، وظیفهٔ تبدیل نام‌های اینترنتی مثل www.cnn.com به آدرس‌های IP مثل107.2. 31.150 را برعهده دارد. بستهٔ انتقالی جهت رسیدن به مقصد، نیاز به عدد IP دارد و از طریق

نام حوزه، مسیریابی ممکن نیست. با نصب DNS در ویندوز NT، آن سیستم به سرویس‌دهندهٔ نام حوزه[1] تبدیل خواهد شد. برای اجرای آن بعد از نصب سرویس، به‌صورت زیر عمل می‌کنیم:

DNS Manager → Administrative → Programs

عملیات لازم جهت راه‌اندازی DNS

۱. از منوی DNS...NEW Server، برای اضافه کردن آدرس IP، سرویس‌دهندهٔ NT به فهرست سرویس‌دهندهٔ DNS استفاده می‌کنیم.
۲. از DNS....NEW Zone برای مشخص کردن Zone و نوع آن استفاده می‌شود.
۳. در نهایت با به‌کاربردن DNS....NEW Domain، حوزهٔ اولیه یا ثانویهٔ سرویس‌دهندهٔ DNS را مشخص می‌کنیم. شکل (۲۰-۱)



شکل ۲۰-۱ نصب DNS روی Win NT

حال بعد از این که بانک اطلاعاتی ایجاد گردید، می‌توان رکوردهای مجزایی به آن اضافه کرد. این بانک اطلاعاتی یک فایل متنی در زیر شاخهٔ زیر می‌باشد. (شکل ۲۱-۱)

Server Root \ System32 \ Dns \ Name.Dns

```
@   IN  SOA     randall.telemark.net.
postmaster.randall.telemark.net. (
                              1995101001       ; serial number
                              10800    ; refresh [3h]
                              3600     ; retry   [1h]
                              691200   ; expire  [8d]
                              86400 ) ; minimum [1d]
    IN  NS  randall.telemark.net.
    IN  NS  www2.canada-stockwatch.com.
    IN  A   204.191.227.65
    IN  MX  10  randall
```

1. Domain Name Server

```
localhost       IN      A       127.0.0.1


randall     IN  A   204.191.227.65
        IN  MX  10  randall

randallg    IN  A   204.191.227.66
        IN  MX  10  randall

ras     IN  A   204.191.227.130

mail    IN  CNAME   randall
smtp    IN  CNAME   randall
pop IN  CNAME   randall
www IN  CNAME   randall.telemark.net.

pam IN  CNAME   randallg
```

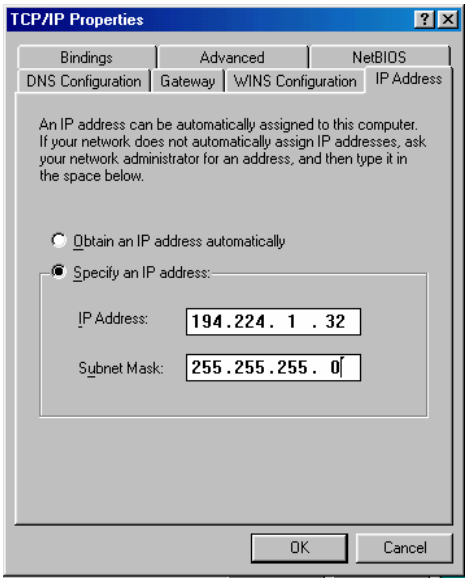
شکل ۲۱-۱ محتویات فایل DNS روی Win NT

(Dynamic Host Configurate Protocol) DHCP

طریقهٔ انتساب IP به کاربران به دو صورت ایستا[1]، پویا[2] انجام‌پذیر است. در حالت ایستا کاربر یک آدرس IP دارد، که مخصوص خود او می‌باشد و باید آن را در سیستم خود نصب نماید. شکل (۲۲-۱)

IP Address → Properties → Tcp / IP → Network → Control panel

1. Static
2. Dynamic



شکل ۲۲-۱ اختصاص ایستای IP به سرویس گیرنده در Win98

با نصب DHCP، امکان انتساب پویای IP از سوی سرویس‌دهنده به سرویس‌گیرنده اضافه می‌شود. با این کار دیگر کامپیوترهای سرویس‌گیرنده به سادگی به شبکه متصل خواهند شد و سربار[1] سرپرستی کاهش خواهد یافت.

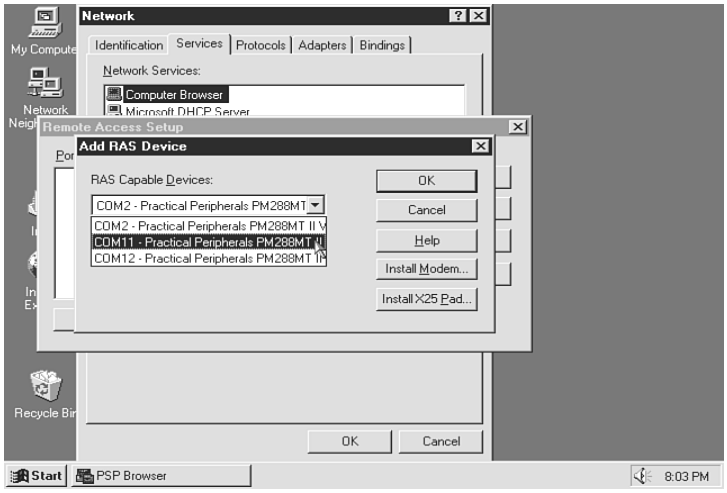
توجه: *اگر نیاز به تغییری در شماره‌های IP وجود داشته باشد، تنها DHCP تغییر می‌کند و نیازی به تغییر در سرویس‌گیرند ها نیست.*

بانصب DHCP، سیستم به‌یک سرویس‌دهنده DHCP تبدیل خواهد شد. درنرم‌افزارDHCP Management در منوی Scope، می‌توان محدودهٔ IP که باید به‌صورت پویا به کاربران داده شود (طبق کلاس IP) را مشخص کرد.

(Remote Access Service) RAS

RAS نیز یک سرویس می‌باشد، که با نصب آن سیستم مربوطه به یک سرویس‌دهندهٔ راه دور تبدیل خواهد شد، وظیفهٔ RAS مدیریت و کنترل پورت‌ها و مودم‌های متصل به آنها، همچنین کاربران راه‌دوری که از طریق خط تلفن با سرویس‌دهنده ارتباط برقرار نموده‌اند، می‌باشد. این نرم‌افزار هنگامی به‌کار می‌رود، که بخواهیم از طریق کارت Mutiport امکان دسترسی از راه دور را فراهم کنیم. بعد از نصب RAS تمامی پورت‌ها و مودم‌ها تشخیص داده شده و مدیریت می شوند. در صورت تماس تلفنی کاربر، RAS خودش گوشی را برداشته و شناسهٔ کاربر و رمز عبور را سؤال می‌کند و ارتباط ppp را برقرار می‌نماید. (شکل ۲۳-۱)

1. Overhead



شکل ۲۳-۱ تنظیمات RAS روی Win Nt

۱۲-۱ تمرین‌ها

۱. روش‌های مختلف ارتباط ماهواره‌ای را شرح داده و معایب و مزایای هر کدام را بیان کنید.
۲. دو روش ارتباط کاربران راه‌دور به یک سایت ISP را شرح داده و معایب و مزایای هر کدام را بیان کنید. (با شکل نشان دهید)
۳. جهت اتصال یک کاربر به اینترنت از طریق یک ISP خاص :
الف – کاربرچه تجهیزات نرم‌افزاری و سخت‌افزاری را باید فراهم نماید.
ب – ISP چه تجهیزات نرم‌افزاری و سخت‌افزاری را باید داشته باشد، به‌طور کامل شرح دهید.
٤. جهت ایجاد یک Domain به‌صورت www.sample.ir چه باید کرد، شرح دهید.
۵. اینترنت ، اینترانت و Exteranet را با هم مقایسه کنید.
٦. کاربردهای DNS در یک زیر شبکهٔ اینترنت و اینترانت را با هم مقایسه کنید.
۷. سرویس DHCP چه مزایایی در یک زیر شبکهٔ اینترنت دارد، شرح دهید.
۸. تفاوت XML،VRML،HTML را بیان نموده و چرا به این زبان‌ها Markup می‌گوییم.
۹. هنگامی که کاربر روی آدرس http://www.cnn.com/image/test.jpg کلیک کند، چه فایلی به‌صورت پیش‌فرض روی سرویس‌دهنده (سرویس‌دهنده WinNT است) مورد دسترسی قرار خواهد گرفت.
۱۰. اگرکاربردرمحیط‌اینترنت درBrowser خود روی آدرس اینترنتی(http://www.sample.com/Test.htm) کلیک کند، چه مراحلی طی می‌شود تا صفحهٔ مربوطه روی Browser نمایش داده شود.
الف– تمامی مراحل مربوط به پروتکل TCP/IP چه در Client و چه در Server را شرح دهید.
ب– اگر به جای Test.htm از Test.pdf استفاده شود، چه تفاوتی خواهد کرد.

۱۱. یک اینترانت دارای٤ سرویس‌دهنده (DHCP – IIS – RAS – DNS) می‌باشد، که آدرس IP سرویس-دهندهٔ وب 194.224.1.200 است، این زیر شبکه دارای ٤٠ تا Localclient و ٥٠ تا Remoteclient می‌باشد، که ١٦ تا از این کاربران از طریق RAC و بقیه از طریق Multiport به شبکه متصل هستند.

الف- ساختار شبکه را رسم کرده و طریقهٔ اتصال client ها را نشان دهید.

ب- کلاس IP شبکه و subnetmask و NetIP را به‌دست آورید.

ج - جهت اتصال این زیر شبکه به اینترنت چه عملیات نرم‌افزاری و سخت‌افزاری مورد نیاز است.

۱۲. دو زیرشبکه با آدرس‌های IP سرویس‌دهنده‌های وب 200.120.23.78 و 190.2.3.20 داریم، با این شرایط که هرکدام ٦ سرویس‌گیرنده محلی و ٨ سرویس‌گیرنده راه دور دارند: به سؤالات زیر پاسخ دهید.

الف- بلاک دیاگرام شبکه را رسم کرده و جهت اتصال این دو زیر شبکه و ایجاد Exteranet چه تجهیزاتی مورد نیاز است؟ در شکل نشان دهید.

ب- کلاس IP وsubnetmask و NetID وhostID را برای هر کدام مشخص کرده و به اجزای شبکهٔ IP اختصاص دهید.

ج- اگر امکان اتصال کاربران از راه‌دور را در هر زیرشبکه بخواهیم ایجاد کنیم، چه تجهیزات سخت‌افزاری و نرم‌افزاری هم در سرویس‌دهنده و هم در سرویس‌گیرنده، طوری که تجهیزات یکسان نباشد، موردنیاز است؟ با شکل نشان دهید.

د- جهت اتصال این Exteranet به اینترنت، طوری که بتوان با یک Domain خاص به هر کدام متصل شد، چه عملیاتی لازم است؟ شرح دهید.

۱۳. فرض کنید بخواهیم یک ISP با کلاس c 1/4 با سرویس‌دهندهٔ NT ایجاد نماییم:

الف- چه عملیاتی قبل از تهیهٔ تجهیزات باید انجام داد؟ (در رابطه با مخابرات)

ب- امکانات سخت‌افزاری موردنیاز جهت ارتباط حداکثر ٨ نفر در آن واحد به سیستم چیست؟

ج - امکانات نرم‌افزاری موردنیاز جهت برقراری ارتباط از راه دور و دسترسی به سایت با آدرس www.iau.edu.ir چیست؟

د- با یک نمودار، طریقهٔ ارتباط یک کاربر از طریق modem به سایت و ارتباط از آن طریق به اینترنت را نشان دهید؟

فصل دوم

زبان HTML (Hyper Text Markup Language)

۲-۱ مقدمه

HTML، مهم‌ترین زبان نشانه‌ای می‌باشد، که بیشتر صفحات وب با این زبان نوشته شده، یا از آن استفاده می‌کنند. HTML بر مبنای SGML[1] که یک استاندارد مدیریت اطلاعات است، ایجاد گشته است. این استاندارد توسط سازمان بین المللی استاندارد ISO[2] در سال ۱۹۸٦ به قصد مهیا نمودن اسنادی که اطلاعات قالب بندی شده و دارای پیوند را مستقل از کاربرد و سکوی سخت‌افزار ارائه نماید، معرفی شد.

HTML، دارای برچسب‌هایی می‌باشد که این برچسب‌ها به سند اضافه می‌شوند و طریقهٔ نمایش سند را از طریق مرورگر مشخص می‌کنند. در این بخش در مورد زبان HTML چگونگی برنامه‌نویسی و برچسب‌های آن بحث خواهیم کرد. فهرست مطالبی که در این بخش خواهیم گفت عبارتند از:

۱. ساختار برنامه‌های HTML
۲. برچسب‌های Text
۳. تصاویر و صدا . (Picture & Sound)
٤. جداول[3] در HTML
٥. پیوندها (link)
٦. نقشه (Map)
۷. Frame ها در HTML
۸. Form ها و عناصر آنها مثل : Key, Check box , Radio botton ,Editor, Inputline ....
۹. Script ها در HTML (Client Side)

می‌توان برنامه‌های HTML را در هر ویرایشگر متنی، مثل Edit در Dos و یا Notepad در Windows بنویسید و آنها را توسط مرورگرهایی چون Internet explorer یا Netscape اجرا و مشاهده نمائید. البته HTML دارای Generator هایی، مثل Frontpage نیز می‌باشد، که می‌توان از طریق آنها صفحهٔ HTML را تولید کرد.

---

1. Standard Generalized Markup Language
2. International Organization for Standardization
3. Table

۲-۲ ساختار برنامه‌های HTML

اکثر Tag های HTML، به‌صورت دوتایی یا دوگان به‌کار می‌روند، طوری که با یک برچسب همانند <Name> شروع و با </Name> خاتمه پیدا می کند. ممکن است یک برچسب دارای آرگمان‌هایی باشد، که این آرگمان‌ها عملکرد معمولی برچسب را تغییر می‌دهند. اگر یک Tag دارای آرگمان باشد، این آرگمان‌ها در Tag اول به‌کار می‌روند. (شکل ۱-۲)

```
<HTML>
<HEAD>
<TITLE >  Test  </TITLE>
</HEAD>
<BODY>
 Test 1
</BODY>
</HTML>
```

شکل۱-۲خروجی برنامهٔ HTML

ساختار Body

Body دارای آرگمان‌هایی می‌باشد، که به‌صورت زیر استفاده می‌شود:

```
"                "   <Body     BGColor =
   "                  " Text =
"                     " LINK =
"                    " VLINK =
>   "        "  Background =
                  ________
                  _____
                </Body>
```

BGColor، رنگ زمینه را مشخص می‌کند، که هم می‌تواند به‌صورت نام رنگ (مثلاً RED) و یا کد رنگ (مثلاً #FF0000) به‌کار رود. Text و Link نیز رنگ متن و پیوندها را در صفحه مشخص می‌کند. VLINK رنگ پیوندها بعد از دیده‌شدن (Visit) می‌باشد. Background نیز آدرس یا URL عکسی است، که می‌خواهید به عنوان زمینهٔ صفحه قرار داده شود، که می‌تواند هر فایل Gif یا Jpg باشد.

۲-۳ رنگ‌ها در HTML

در HTML رنگ‌ها را هم می‌توان به‌صورت کد و هم نام به‌کار برد مثلاً:

```
Black        # 000000
# FFFFFF      White
ff0000        Red  #
#   00ff00        Green
# 0000ff          Blue
 #   ffff00          Yellow
```

هر رنگ، ترکیبی از RGB می‌باشد، که به‌صورت زیر ساخته می‌شود : (طوری‌که هر X یک عدد Hex بین 0 تا F می‌باشد.)

بدین صورت می‌توان 2^{24} رنگ ایجاد کرد:

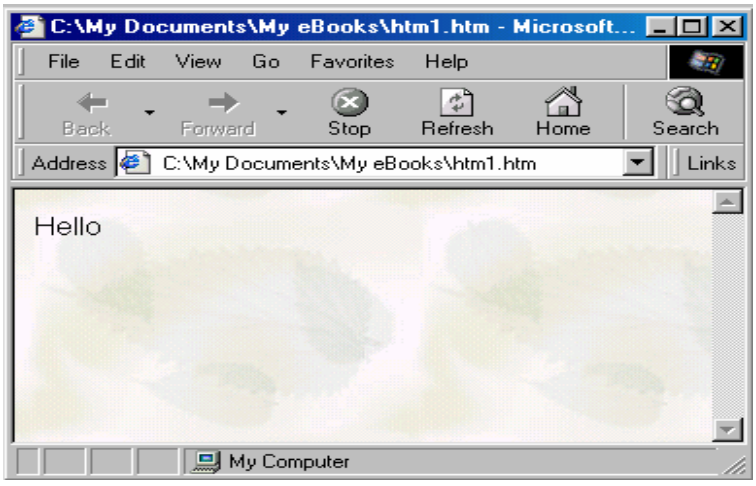
R G B

XX XX XX

مثال :

برنامهٔ زیر، یک صفحه ایجاد خواهد کرد که در آن متن‌ها به‌صورت سیاه رنگ و پیوندها به رنگ آبی و پیوندها بعد از دیده‌شدن زرد خواهند شد، زیر صفحه نیز با تصویر Map.gif پر خواهند شد. (شکل ۲-۲)

```
<HTML>
     <Body      BGCOLOR    = "White"
                TEXT       = "Black"
                LINK       = "#0000ff"
                VLINK      = "#ffff00"
                Background = "d:\image\Nabkgnd.jpg" >
        Hello
     </Body>
  </HTML>
```



شکل۲-۲ خروجی برنامهٔ HTML

۲-۴ Tag های متن

این برچسب‌ها درداخل بدنهٔ برنامهٔ HTML قرار می‌گیرند و تأثیر خود را روی متن داخل بدنه می‌گذارند.

<B> _____ </B> (Bold)

<I> _____ </I> (Italic)

<U> _____ </U> (Under line)

توجه: *باید توجه کرد، که در HTML بعضی برچسب‌ها دوتایی و بعضی تنها به‌کار می‌روند.*

پاراگراف‌ها در متن

پاراگراف‌ها با برچسب <P> مشخص می‌شوند. از پاراگراف‌ها جهت راست چین یا چپ چین کردن متن می‌توان استفاده کرد.

<P Align = " ">

_____

_____

_____

</P>

طوری که در آن Align می‌تواند Center– Left –Right قرار گیرد. همچنین می‌توان از برچسب < Center > برای به‌کاربردن متن در وسط صفحه استفاده کرد.

<Center> _____</Center>

توجه *: در حالت معمولی، اگر متن را زیر هم بنویسیم، مرورگر آنها را به هم می‌چسباند. به همین جهت برای فرستادن متن به خط بعد از
 استفاده می‌شود. (البته می‌توان از برچسب <P> به تنهایی نیز برای این موضوع استفاده کرد.)*

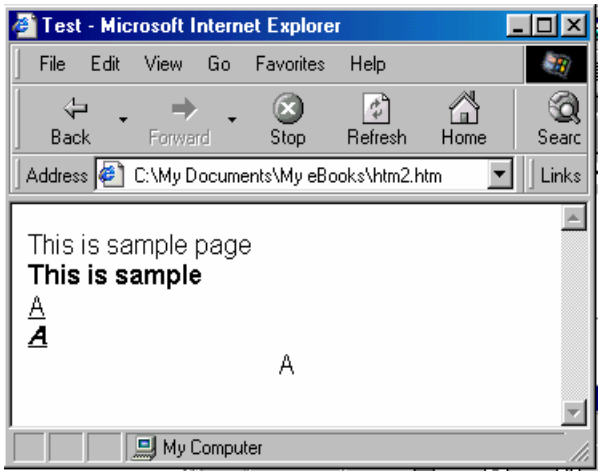
توضیحات در HTML

از طریق برچسب زیر، توضیحات یا Comment ها می‌توانند به‌کار روند: (شکل ۳-۲)

< ! _____ >

مثال :

```
<HTML>
          <Head >
               <Title> Test  </Title>
          </ Head>
          <Body>
             This is sample page <Br>
             <B> This is sample </B> <Br>
             <U> A </U> <Br>
             <U> <B><I> A </B> </U></I> <Br>
             <Center> A  </Center>
          </Body>
 </HTML>
```



شکل ۳- ۲ اجرای برنامه HTML بالا

چشمک زن شدن متن

جهت اینکار از برچسب <Blink> استفاده می‌شود، البته این برچسب درIE کار نمی‌کند، ولی در Netscape می‌توان از آن استفاده کرد.

<Blink> _____ </Blink>

۲-۵ Header در متن

برای این کار از برچسب‌های زیر استفاده می‌شود.

<H1> _ _ _ </H1>
<H2> _ _ _ </H2>
</H6> _ _ _ <H6>

اندازهٔ فونت‌ها افزایش می‌یابد

۲-۶ استفاده از اندیس و توان

جهت ایجاد چند جملهٔ ریاضی می‌توان از آنها استفاده کرد. جهت این کار از برچسب‌های <SUB> و <SUP> استفاده می‌شود.

_{_ _ _}اندیس
^{_ _ _}توان

مثال : $2x^{3}_{2}$

2 × ₂ ³

۷-۲مشخص کردن Font متن

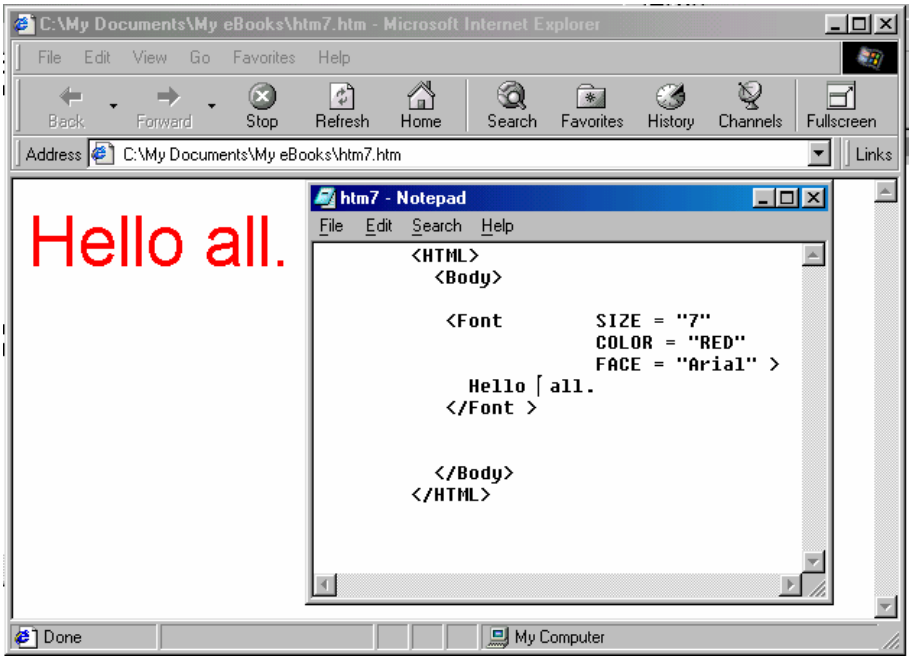
جهت این کار از برچسب <Font> استفاده می‌شود. در این برچسب، آرگمان Size اندازه را مشخص می‌کند، که می‌تواند اعداد 1 تا 7 باشد. هرچه این عدد بزرگ‌تر باشد، Font بزرگ‌تر است، Color نیز رنگ و FACE نام فونت را مشخص می‌کند، مثلاً Arial یا Sepehr برای فونت فارسی.

<Font SIZE = “ ”
COLOR = " “
FACE = " ” >
_____
_____
_____
</Font >

*برای تعمیم دادن Font اصلی، از نشانهٔ زیر استفاده می‌شود: (شکل ٤-۲)

<BASE FONT SIZE = “ “ >

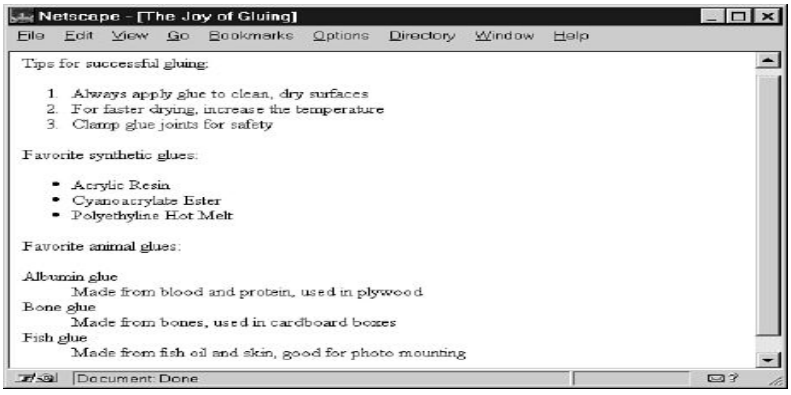
مثال :



شکل ٤-۲ مثالی از Font

۲-۸ Bulleted list , Order list

برای ایجاد یک لیست ترتیبی و یا یک لیست گلوله دار، از برچسب‌های زیر استفاده می‌کنیم. در این صورت، لیست را خودش شماره‌گذاری می‌کند و یا با علامت گلوله، آن را مشخص می‌کند.



شکل ۲-۵ Bulleted list , Order list

```
            <OL>
</LI>   _____________ <LI>
</LI>  _______________  <LI>
</OL>

  <UL>
</LI>  _____________  <LI>
</LI>  _____________  <LI>
</UL>
```

۲-۹ استفاده از تصویر در HTML

برای نمایش تصاویر با قالب Gif و Jpg، می‌توان از برچسب زیر استفاده کرد.

```
 <Img  SRC = "   "    ALT = "      "          WIDTH = "    "    HIGHT =
"      "
                                  BORDER = "   "   ALIGN =  "      " >
```

این برچسب که به‌صورت تنها به‌کار می‌رود، دارای آرگمان‌های SRC، جهت مشخص کردن آدرس عکس وAlt مشخص‌کنندهٔ متنی است، که به‌جای عکس نمایش داده می‌شود، به آن Alternative Text گویند، (این متن در مرورگرهایی که قادر به نمایش تصویر نیستند، به جای عکس قرار می‌گیرد، همچنین در مرورگرهای معمولی وقتی با mouse روی تصویر حرکت می‌کنیم بلافاصله ظاهر می‌گردند). Width و Hight نیز ابعاد عکس و Border ضخامت کادر عکس را مشخص می‌کند Align نیز می‌تواند (Top یا Middle یا Buttom باشد) وضعیت عکس نسبت به خط را نشان می‌دهد. باید توجه کرد که از برچسب

IMG می‌توان جهت فراخوانی برنامه‌های دروازه‌ای روی سرویس‌دهنده نیز استفاده کرد. این موضوع در بخش cgi مورد بررسی قرار خواهد گرفت.

۲-۱۰ جدول (Table)

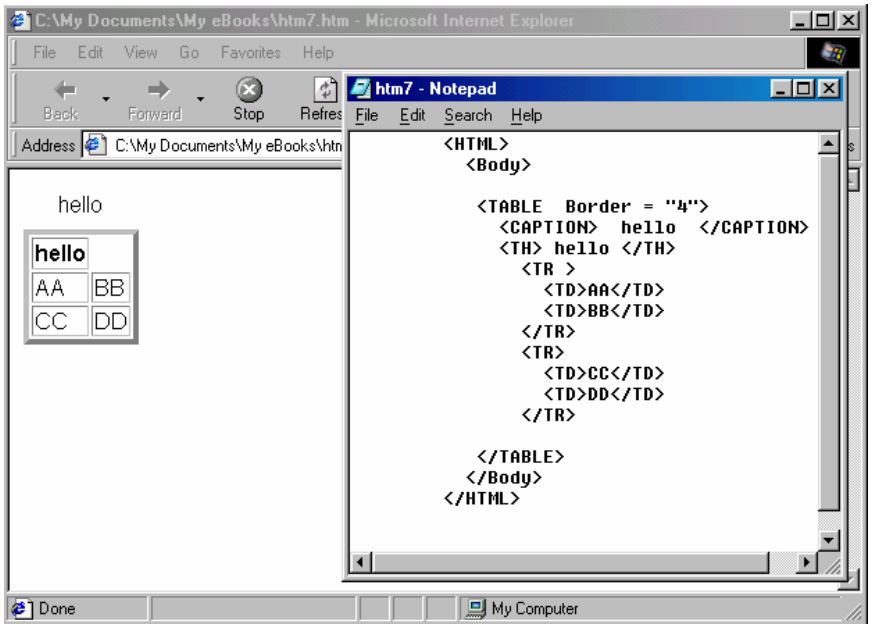
جداول در HTML دو کاربرد دارند:

۱. جدول کشیدن

۲. تقسیم کردن صفحهٔ مرورگر

برچسب آن <Table> می‌باشد، که به‌صورت زیر استفاده می‌شود. (شکل ۵-۲)

```
<TABLE  Border =  “”    >
<CAPTION>  ----- </CAPTION>
     <TH>-----  </TH>
    <TR >
          <TD>  _____________  </TD>
         <TD>  _____________  </TD>
      < /TR>
      <TR>
          <TD>  _____________  </TD>
          <TD>  _____________  </TD>
       < /TR>
       </ TABLE>
```



شکل ۵-۲ مثالی از یک جدول در HTML

استفاده از جدول برای تقسیم‌بندی صفحه در مثال زیر نشان داده شده است:

```
<HTML>
          <Body>
             <Table Border = "0">
             <TR>
                <TD>
                    Buy ID and Password <BR>
                    Buy Books and Other Things<BR>
                </TD>
                <TD>
                    <img          src="C:\WIN98\Temporary     Internet
Files\VITZNXOW\freeship_0402.gif">
                </TD>
             </TR>
             </Table>
          </Body>
 </HTML>
```

شکل ۷-۲ شکل اجرای برنامهٔ بالا

خط کشی در صفحه

جهت کشیدن خط در صفحه، از برچسب <HR> استفاده می‌شود و توسط آن می‌توان اندازه و رنگ خط را نیز مشخص کرد:

```
<HR Width="       "  Color = "    "
Size="     ">
```

۲-۱۱ صدا در HTML

برای این کار از برچسب BGSOUND استفاده می‌شود. به کمک این برچسب، می‌توان فایل‌های WAV , MIDI را در صفحه پخش کرد.

```
<BGSOUND  SRC= " "LOOP = " ">
```

Loop تعداد دفعات پخش صدا در صفحه می‌باشد. می‌توان به جای " "=Loop، که تعداد دفعات تکرار موزیک است، از Infinite استفاده کرد. در این صورت آهنگ به‌طور دائم نواخته خواهد شد.

۲-۱۲ پیوندها در HTML

برچسب مربوطه <A> (anchors به معنی لنگر) است، که دو کاربرد می‌تواند داشته باشد.

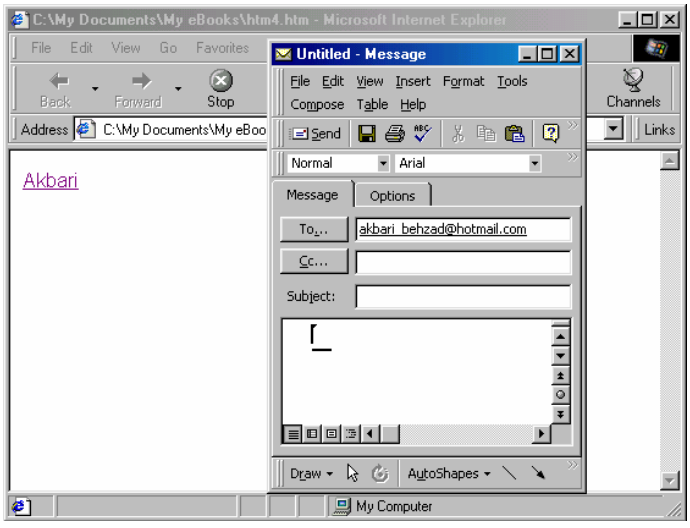
۱. برای email، حالت کلی آن به‌صورت زیر است:

```
< A HREF = " Mailto : your email Address" >
 ______
 ______
</A>
```

در این حالت، بعد از کلیک کردن روی جمله نوشته شده بین تگ <A> ، برنامه mailto باز می‌شود، که آدرس پست‌الکترونیکی درآن ذکر شده است. تنها کافی است اطلاعات پرشده و ارسال گردد. (شکل۸-۲)

مثال :

```
<HTML>
        <Body>
           <A  HREF  =  "mailto:Akbari_behzad@hotmail.Com  ">  Akbari
</A>
        </Body>
</HTML>
```



شکل ۸-۲ مثال پیوند

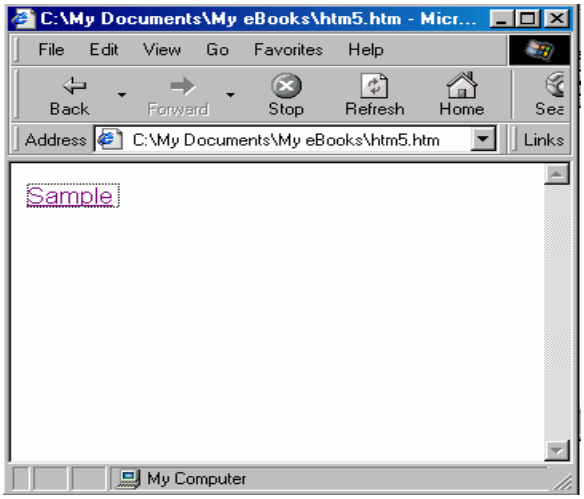
۲. برای ایجاد پیوندهای وب

```
<A HREF = "   " Name = "    " >
_____________

_____
< /A>
```

در این حالت با کلیک کردن روی جمله نوشته شده بین تگ <A> ، یک پیوند به آدرس صفحهٔ نوشته شده در آرگمان HREF برقرار می‌شود. (شکل ۹-۲)

مثال :

```
<HTML>
        <Body>
              <A HREF = "http://www.sample.com/default.htm"> Sample  </A>
        </Body>
</HTML>
```



شکل ۹-۲ شکل اجرای برنامهٔ بالا

۲-۱۳ Map در HTML

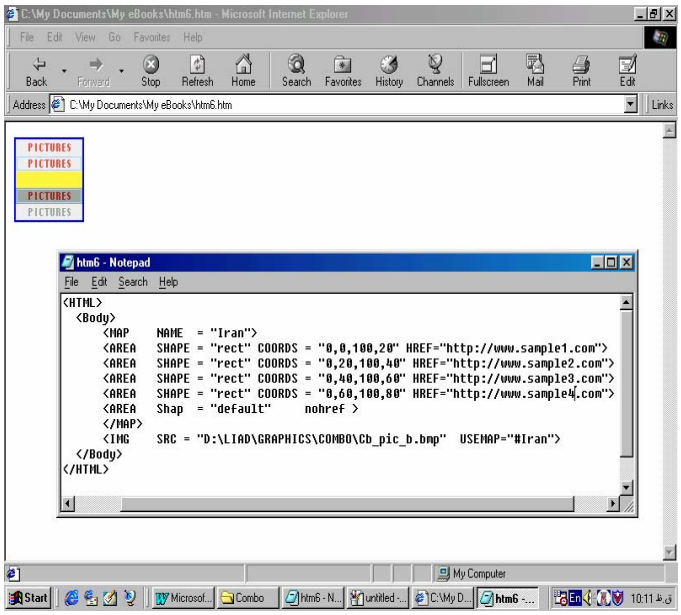
در HTML، این امکان وجود دارد که یک عکس را (gif یا jpa) به چندین ناحیه تقسیم نمود، طوری که کاربر با کلیک کردن در آن ناحیه بتواند به صفحهٔ دیگری متصل گردد.

برچسب مربوط به آن <MAP> می‌باشد.

هر Map به یک تصویر مربوط می‌شود، که این کار از طریق برچسب <IMG> و آرگمان USEMAP انجام می‌شود. (شکل ۱۱-۲)

```
                <MAP     NAME   = "Iran">
               <AREA    SHAPE = "rect" COORDS = "0,0,100,20"
HREF="http://www.sample1.com">
               <AREA    SHAPE = "rect" COORDS = "0,20,100,40"
HREF="http://www.sample2.com">
```

```
            <AREA    SHAPE = "rect" COORDS = "0,40,100,60"
HREF="http://www.sample3.com">
            <AREA    SHAPE = "rect" COORDS = "0,60,100,80"
HREF="http://www.sample4.com">
            <AREA    Shap  = "default"      nohref >
            </MAP>
            <IMG     SRC = "D:\LIAD\GRAPHICS\COMBO\Cb_pic_b.bmp"
USEMAP="#Iran">
```

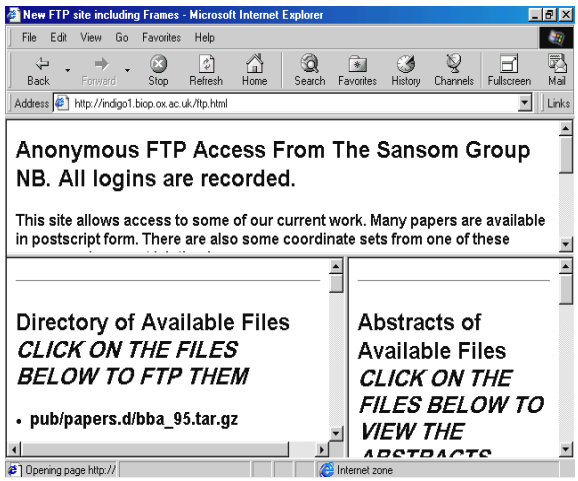


شکل ۱۱-۲ مثالی از یک map

۲-۱۴ Frame ها در HTML

از طریق فرم‌ها می‌توان چندین HTML را در یک صفحهٔ مرورگر نمایش داد. در این حالت از برچسب frameset استفاده می‌کنیم، مانند زیر:

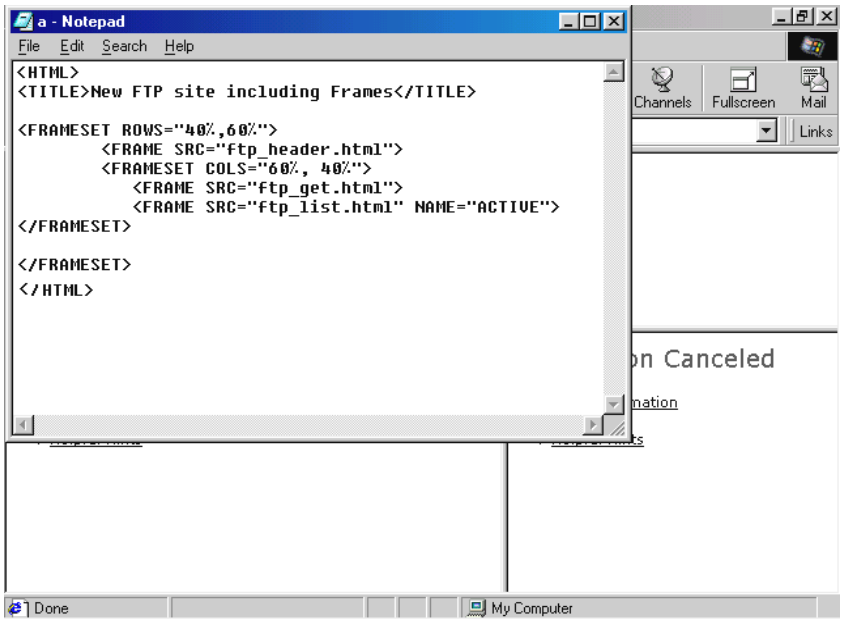
هر frameset می‌تواند عمودی یا افقی باشد، که خودش می‌تواند تعدادی frame یا frameset داشته باشد. (شکل ۱۲-۲)



شکل ۱۲-۲مثالی از یک frame در اینترنت

برای استفاده از frame ها همواره باید یک برنامهٔ main.htmتعریف کرد و در آن بدنهٔ frame را تعریف نمود، سپس بسته به تعداد frame ها برای هر کدام یک html جدا نوشت، مثلاً برای صفحهٔ زیر نیاز به چهار برنامه به نام‌های زیر می‌باشد: (شکل ۱۳-۲)

```
main.htm
title.htm
menu.htm
about.htm
```



شکل ۱۳-۲ متن برنامهٔ HTML برای شکل قبل

فرمت کلی

هر مجموعهٔ Frame، یا افقی است، یا عمودی، به همین جهت باید در Frameset نوع را مشخص کرد. اگر مجموعهٔ Frame افق باشد، از آرگمان Rows در Frameset استفاده می‌شود و اگر عمودی باشد از Cols استفاده می‌کنیم. همچنین آرگمان‌های NORESIZE و " "=Scrolling در Frame، می‌توانند استفاده شوند. آرگمان اول باعث می‌شود، که نتوان Frame را با Mouse تغییر اندازه داد و دومی Scroll bar را فعال یا غیرفعال می‌کند.

مثال :

```
<HTML>
<FRAMESET        ROWS = " 10% ; * ">
<FRAME        SRC = " title.htm "    Scrolling = No   NORESIZE>
<FRAMESET    Cols=  "20%; *">
      <FRAME            SRC = " menu.htm"   Scrolling =NO
NORESIZE>
     <FRAME             SRC = " About.htm" Resize>
 </FRAMESET>
</FRAMESET>
 </HTML>
```

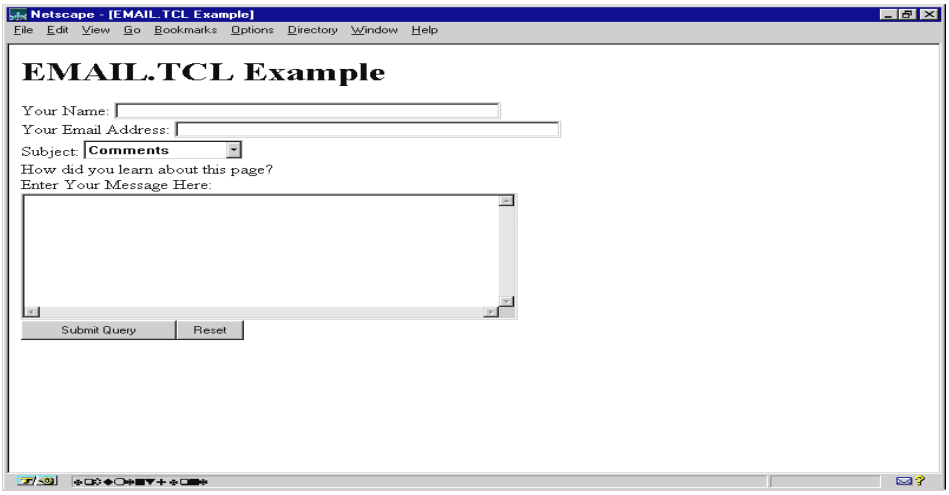
توجه: *باید توجه کرد، که Frame ها را نباید در Body تعریف کرد. به‌طورکلی برنامهٔ Main.htm دارای Body نباید باشد.*

۲-۱۵ FORM ها در HTML

همانند سایر زبان‌های برنامه‌نویسی در HTML، نیز می‌توان فرم ورود اطلاعات ایجاد نمود. به کمک این فرم می‌توان از کاربر اطلاعاتی مثل مشخصات کاربر را سؤال کرد. برای این کار از برچسب form استفاده می‌شود. Form می‌تواند اشیای زیادی را در خود داشته باشد. بعضی از این اشیاء عبارتند از:

```
TEXT (Inputline)
Editor
Radiobotton
Check box
Key
```

هر کدام از شیءهای بالا را می‌توان در فرم به‌کار برد، مثلاً فرم زیر را داریم. (شکل ١٤-٢)



شکل ١٤-٢ یک فرم ورود اطلاعات در HTML

هر فرم، نیاز به یک برنامه جهت اداره کردن آن فرم دارد. بیشتر اوقات این برنامه که به آن برنامهٔ دروازه‌ای می‌گوییم، یک برنامهٔ Script یا یک برنامهٔ exe برروی سرویس‌دهنده است. برچسب form، خودش دارای تعدادی آرگمان می‌باشد، که به‌صورت زیر به‌کار می‌رود.

```
 < form      Method = "          "      Action= "    " >
_____________________</form>
```

Method، روش ارسال اطلاعات به برنامهٔ دروازه‌ای[1] روی سرویس‌دهنده و Action آدرس برنامهٔ دروازه‌ای است. در مورد این آرگمان‌ها، در بخش برنامه‌نویسی Cgi توضیح خواهیم داد.

حالا هر کدام از این اشیاء را که بخواهیم، باید در داخل form به‌کار ببریم.

اکثر این اشیاء را از طریق برچسب Input، که داخل form به‌کار می‌رود، می‌توان ساخت. قالب آن به‌صورت زیر است: (شکل ١٥-٢)

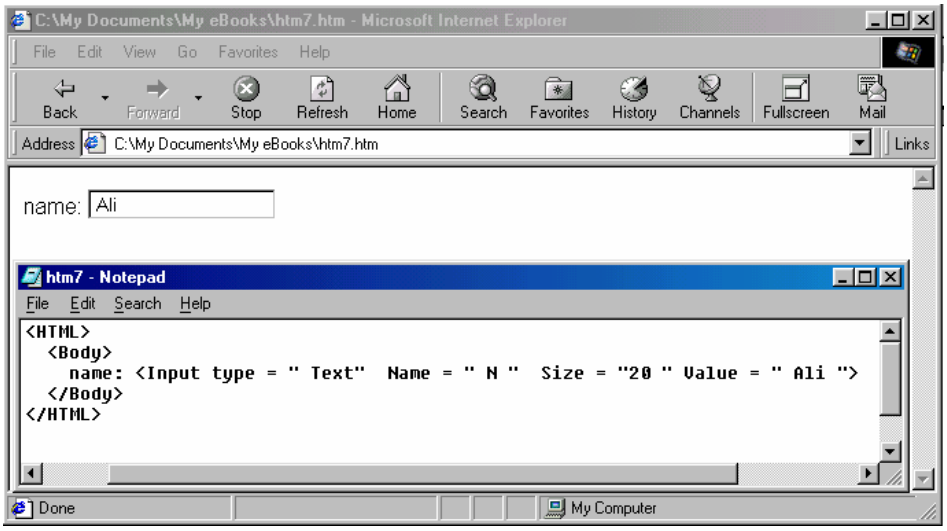
توجه: *باید توجه کرد نوع پیش‌فرض برای Input ، Text می‌باشد.*

```
  <Input     type = " Text"      Name = "    "     Size = "    "
Value = "        ">
```

مثال :

```
<Input     type = " Text"      Name = " N "     Size = "20 "     Value
= " Ali ">
```

1. Gateway pragram



شکل ۱۵-۲ مثالی از یک input line

اندازه اگر گفته نشود، به‌طور پیش‌فرض ۵۰ کاراکتر است و Name نام شیء است، که در برنامهٔ دروازه‌ای یا برنامه‌نویسی Script به‌کار می‌رود.

این برچسب هم می‌تواند تنها به‌کار رود و هم دوتایی، که در این صورت با </ Input> خاتمه می‌یابد.

کلیدها در HTML

سه نوع کلید در HTML می‌توان ساخت:

۱. Submit کنترل و مدیریت آن توسط مرورگر است.

۲. Reset کنترل و مدیریت آن توسط مرورگر است.

۳. کلیدهای خاص (Button) کنترل و مدیریت آن توسط کاربر است.

کلید Submit کلیدی است، که با انتخاب آن اطلاعات موجود در form به برنامهٔ دروازه‌ای که آدرس آن در Action مشخص شده است، ارسال می‌گردند. باید توجه کرد که این اطلاعات ابتدا کد شده و بعد بسته به متد به برنامهٔ دروازه‌ای ارسال شوند.

مثال :

```
<Input    type = " Submit "Name = "OK" Value = " OK">
```

کلید Reset، باعث خالی شدن اطلاعات داخل form خواهند شد.

```
<Input    type = " Reset"    Name =" Res"    Value = " Rest">
```

کلید Button، برای برنامه‌نویسی خصوصاً اسکریپت‌ها به‌کار می‌رود.

```
< Input    type = " Button"    Name =" Bu1"    Value = " OK"    On click
="Win open ( )">
```

با فشار کلید OK زیر روال Win open ( )، که قبلاً توسط اسکریپت تعریف شده است، فراخوانی می‌شود.

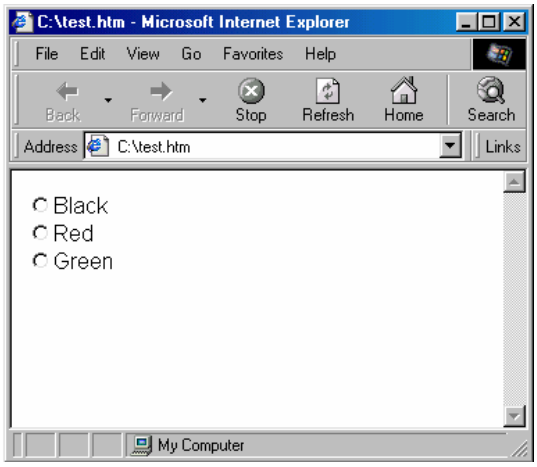
Radio Button

برای ایجاد لیستی از گزینه‌ها، که بتوان تنها یکی از هر کدام را انتخاب کرد از Radio استفاده می‌شود. مثل لیست زیر: (شکل ۱۶- ۲)

شکل ۱۶-۲ Radio Button و Check Box در HTML

مثال : (شکل ۱۷- ۲)

```
<Input    Type = "Radio"    Name ="P1"    value="B">Black <Br>
<Input    Type = "Radio"    Name ="P1"    value="B" >Red <Br>
<Input    Type = "Radio"    Name ="P1"    value="B">Green <Br>
```

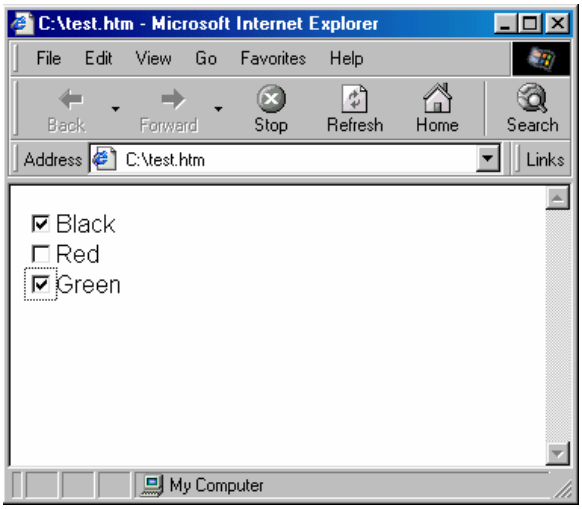


شکل ۱۷-۲ مثال Radio Button

CHECKBOX

برای ایجاد لیستی، که بتوان هر کدام از گزینه‌ها را انتخاب نمود، طوری که بتوان بیشتر از یکی را نیز انتخاب نمود: (شکل ۱۸- ۲)

```
<Input   Type = "CheckBox"   Name ="P1"   value="B">Black <Br>
<Input   Type = "CheckBox"   Name ="P2"   value="B" >Red <Br>
<Input   Type = "CheckBox"   Name ="P3"   value="B">Green <Br>
```



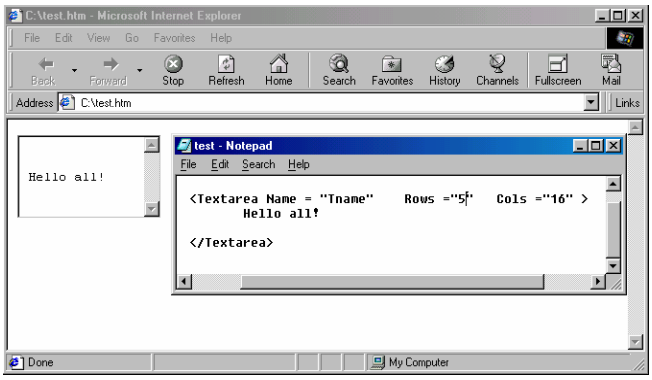
شکل ۱۸-۲ مثال CheckBox

ویرایشگر در HTML

برای ایجاد ویرایشگر از برچسب Textarea استفاده می‌شود. قالب آن به‌صورت زیر است:

```
<Textarea Name = "Tname"    Rows ="5"   Cols ="6" >
    _____________________
    _____________________
    _____________________
</Textarea>
```

در این Tag، ابتدا باید سطر و ستون ویرایشگر و همچنین نام آن را وارد کنیم. اطلاعاتی که بین <Textarea> نوشته شود، داخل Editor قرار خواهند گرفت: (شکل ۱۹- ۲)



شکل ۱۹-۲مثال ویرایشگر

مثال: برنامهٔ زیر، مثالی از یک فرم کامل می‌باشد.

```
<HTML>
<HEAD>
       <TITLE>My Guestbook</TITLE>
</HEAD>
<BODY>
     <H1>My Guestbook: Please Sign In</H1>
     <FORM METHOD="post"     ACTION=http://www.cgi_free.com/cgi/genevic.exe”>
        Your name: <INPUT TYPE="text" NAME="name" SIZE=20><P>
         Your sex: <INPUT TYPE="radio" NAME="sex" VALUE="male"> male
        <INPUT TYPE="radio" NAME="sex" VALUE="female"> female<P>
         Your age: <INPUT TYPE="text" NAME="age" SIZE=4><P>
        Your e-mail address: <INPUT TYPE="text" NAME="email" SIZE=30><P>
         <INPUT TYPE="submit" VALUE="Sign In">
        <INPUT TYPE="reset" VALUE="Erase">
     </FORM>
</BODY>
</HTML>
```

توجه کنید، که در این مثال به جای
 از <B> ، استفاده شده است. (شکل ۲۰– ۲)

شکل۲۰–۲ مثالی از یک form در HTML

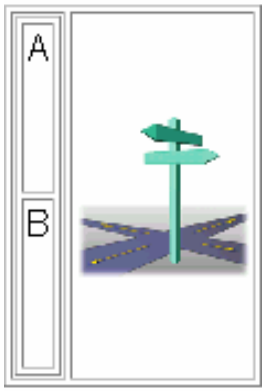
۲–۱۶ تمرین‌ها

۱. برنامه‌ای با HTML بنویسید، که نام و نام خانوادگی خود را به‌صورت Bold و Underline Italic در سه خط جدا زیر هم در وسط صفحه با رنگ سبز و رنگ زمینه سیاه نمایش دهد.

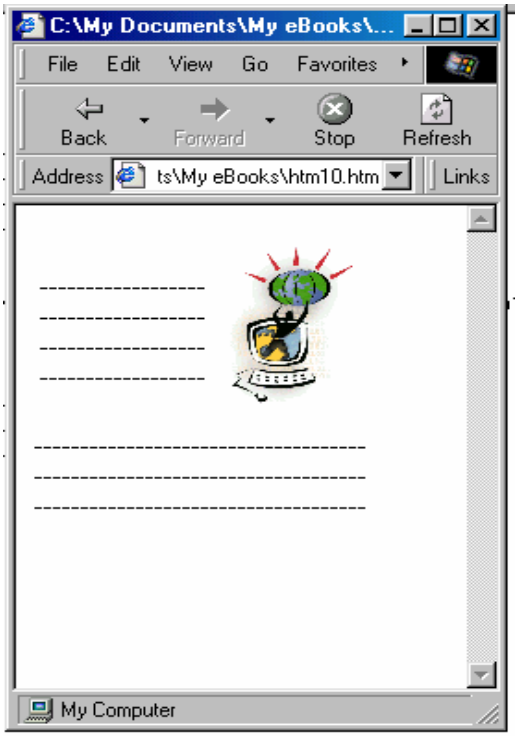
۲. برنامه‌ای با HTML بنویسید، که چند جملهٔ زیر را در صفحه نمایش دهد.

$$4x_1^5 + 3x_2^3 + 3x_1^2$$

۳. برنامه‌ای با HTML بنویسید، که جدولی به‌صورت زیر نمایش دهد. (در صورتی که روی تصویر کلیک کنیم، به آدرس www.image.com وصل می‌شویم.)

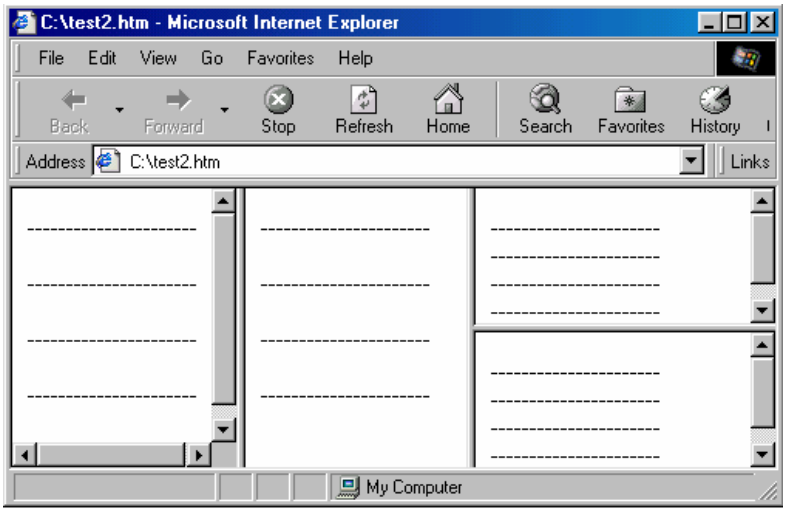


٤. برنامه‌ای با HTML بنویسید، که خروجی زیر را تولید کند و اگر روی تصویر click شود، بتوان به آدرس Test@yahoo.com پست الکترونیکی فرستاد.



۵. برنامه‌ای با HTML بنویسید، که نقشهٔ ایران را به استان‌های مختلف تقسیم کند و روی هر استان که کلیک شود، به اطلاعات آن استان بتوان رجوع کرد.

٦. برنامه‌ای با HTML بنویسید، که از طریق frame، صفحهٔ زیر را ایجاد کند.



برنامه‌ای بنویسید، که از کاربر، رمز عبور و شناسهٔ کاربر را سؤال کند.

۷. فرمی با HTML طراحی کنید، که اطلاعات کاربر از جمله نام و نام خانوادگی و جنسیت و رنگ‌های مورد علاقهٔ کاربر (بین سه رنگ آبی، قرمز و سبز) و توضیحات اضافی را گرفته و کلیدهای Ok و reset داشته باشد.

فصل سوم

HTML پویا[1]

۳-۱ مقدمه

HTML پویا، DHTML ویژگی است، که امکان خلق صفحات وبی تعاملی[2] با قابلیت‌های چند رسانه‌ای را ایجاد می‌کند، برای خلق HTML پویا می‌توان از زبان‌های اسکریپت مختلفی استفاده نمود، که با استفاده از قابلیت‌های آنها می‌توان عناصر صفحهٔ وب اعم از Tag ها، تصاویر، اشیاء و متون را تغییر داد.

HTML پویا همچنین از رویدادهای صفحهٔ کلید، ماوس و غیره حمایت می‌کند. HTML پویا دستاورد نسبتاً جدیدی در عرصهٔ وب است.

بر مطالب فوق سنتی و دیرینه، دو غول عرصهٔ مرورگرهای اینترنت، یعنی Microsoft و Netscape را نیز باید افزود. رقابت بین این دو بازیگر اصلی، عملاً چنان است، که این دو شرکت عملاً روش‌های متفاوتی را در تعریف HTML پویا پیش گرفته‌اند. این اختلاف به حدی شدید است، که شرکت Microsoft از نماد DHTML و شرکت Netscape از نماد (dHTML) برای معرفی HTML پویا استفاده می‌کنند. دو شرکت فوق با افزودن قابلیت‌های خاصی بر مرورگرهای خود، بر پیچیدگی مسئله می‌افزایند.

از آنجا که در عرصهٔ جهانی تقریباً 60% از استفاده کنندگان اینترنت از Internet Explorer و 40% از Netscape Navigator استفاده می‌نمایند (البته این درصد تقریبی است و تعداد استفاده کنندگان از IE در حال افزایش است). لذا اگر کاربردی برای عرضه برروی اینترنت ایجاد شود، به هیچ عنوان نمی‌توان از مسئلهٔ سازگار بودن چشم‌پوشی نمود. کوشش‌هایی که برای دستیابی به یک استاندارد معین صورت گرفته است، به مدلی به نام DOM منجر گشته است.

DOM مخفف (Document Object Model) است، که می توان از آن با عنوان (مدل شیئی سند) یاد کرد. در واقع نسخهٔ HTML4 مدل DOM و زبان‌های اسکریپتی را به عنوان راهکاری برای بهبود عملکرد پویای صفحه ارائه نموده است. DOM در واقع طریقه‌ای است، که به کمک آن می‌توان به تک تک اعضای صفحه به مثابه یک شیء نگریست. سپس می‌توان به اجزای صفحه با کمک زبان‌های اسکریپتی دست یافت و در نهایت با انجام کدنویس‌های کوتاه در چنین زبان‌هایی، پویایی را به صفحه، اعمال نمود.

در این بخش، صرفاً به این مطلب اشاره می‌گردد، که اختلاف دیدگاه بین دو شرکت بزرگ عرصهٔ مرورگر، معمولاً به این صورت است، که شرکت Netscape با افزودن Tag هایی، قابلیت HTML استاندارد

---

1. Dynamic HTML
2. Interacetive

را افزایش می‌دهد، در حالی است که Microsoft با افزودن قابلیت‌های جدید به Tag های موجود، امکانات مرورگر را توسعه می‌دهد.

۳-۲ برنامه‌نویسی سمت- سرویس‌گیرنده و سمت- سرویس‌دهنده[1]

برنامه‌نویسی سمت- سرویس‌گیرنده، اصطلاحاً به برنامه‌هایی گفته می‌شود، که می‌تواند توسط مرورگر تفسیر و اجرا گردد، وقتی یک برنامه توسط یکی از زبان‌های سمت سرویس‌گیرنده نوشته می‌شود، هنگامی که این برنامه‌ها به‌درون مرورگر بار می‌شوند، مرورگر به‌صورت خودکار برنامه را اجرا می‌کند.

همچنین برنامه‌های سمت- سرویس‌دهنده به برنامه‌هایی گفته می‌شود، که برروی سرویس‌دهنده اجرا می گردد. یک زبان سمت- سرویس‌دهنده تمامی اعمال خود را برروی سرویس‌دهنده صورت می‌دهد. به عنوان مثال Vbscript، Javascript هر دو می‌توانند نقش زبان‌های سمت سرویس‌دهنده و سرویس‌گیرنده را بازی نمایند، اما از آنجا که هر دو شرکت Microsoft و Netscape ملزم به حمایت از Java در مرورگرهای خود هستند، لذا از Javascript ، Jscript (ویرایش ارائه شده توسط Microsoft از زبان Javascript) غالباً برای برنامه‌نویسی سمت سرویس‌گیرنده استفاده می‌گردد و Vbscript برای برنامه‌نویسی سمت- سرویس‌دهنده به‌کار می‌رود.

این امر به دلیل آن است که اسکریپت‌های نوشته شده توسط زبان‌های سمت- سرویس‌دهنده پیش از ارسال برروی اینترنت، مورد پردازش قرار می‌گیرند و مرورگر کاربران در عمل چیزی غیر از یک صفحهٔ HTML استاندارد دریافت نمی‌کنند، لذا می‌توان از آن به عنوان زبان سمت- سرویس‌دهنده استفاده کرد.

۳-۳ اسکریپت‌های سمت سرویس‌گیرنده

اگر برنامه‌های اسکریپت را داخل HTML بنویسیم، طوری که روی سرویس‌گیرنده انتقال پیدا کند و توسط مرورگر اجرا شود، آن را اسکریپت‌های سمت سرویس‌گیرنده می‌گوییم، که بحث این بخش در این رابطه است.

با زبان‌های برنامه‌نویسی مثال VBScript و Javascript و غیره می‌توان در داخل HTML، برنامه‌نویسی نمود. در ضمیمهٔ کتاب به‌طور خلاصه دستورات مربوط به VBScript و Javascript بیان شده است. برچسب مربوطه عبارتند از:

```
< Script  language  = "      "    Runat = "    ">
          ____________________________________
          ____________________________________
         ____________________________________
      </ Script >
```

1. Client side-Jerver side

طوری در قسمت Language می‌توان از زبان‌های JavaScript,Vbscript,Jscript ,... و به جای Runat از Client یا Server استفاده کرد، که سمت سرویس‌گیرنده یا سمت سرویس‌دهنده بودن اسکریپت را مشخص می‌کند.

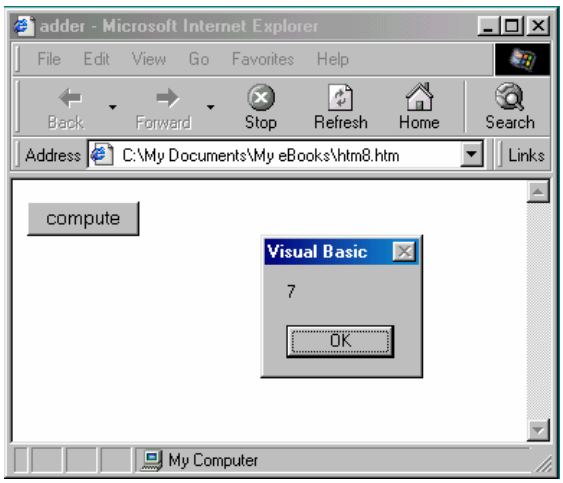
***توجه** : ممکن است بعضی از مرورگرها قدیمی، <Script> راپشتیبانی نکنند، از این‌رو می‌توان آن را در Comment قرار داد. در این صورت اگر مرورگر، اسکریپت را پشتیبانی کند، آن را اجرا می‌نماید و در غیر این صورت از آن به عنوان یک توضیح، رد می‌شود.*

مثال ۱ :

در این مثال، تابعی با Vbscript تعریف شده است، به نام add()، که در آن دو متغیر X, Y را تعریف کرده و مجموع آنها را در یک جعبهٔ پیغام نمایش می‌دهیم.

این تابع با زده شدن کلیدی از نوع Button، به نام Comute اجرا می‌شود. (شکل ۱- ۳)

```
<HTML>
<head>
 <Title>adder </title>

<Script  language ="vbscript">
 SUB  add()
 x=2
 y=5
 msgbox  X+Y
 END SUB
 </Script>

 </head>

 <body>
        <input  type="button"   value="compute" Onclick="add()" >
  </body>
 </HTML >
```



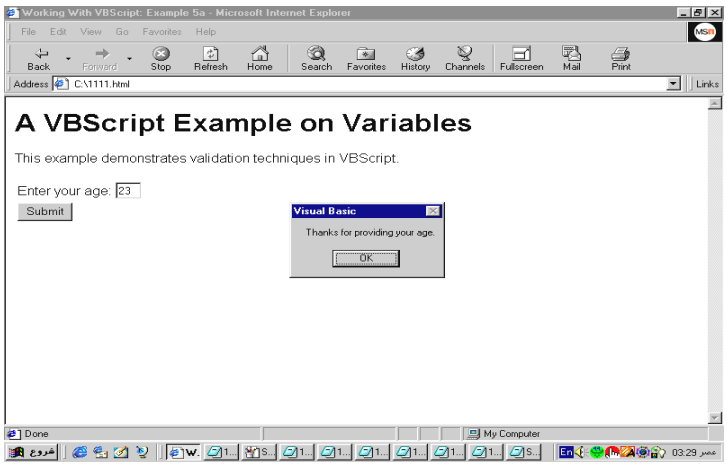
شکل ۱-۳ اجرا برنامه بالا

مثال۲ :

مثال زیر مربوط به برنامهٔ Vbscript می‌باشد، که به ازای ورود سن مشخص، اگر اطلاعات درسی به عنوان سن وارد نشده باشد، پیغام خطا خواهد داد. (شکل ۲- ۳)

```
<HTML>

    <HEAD>

        <TITLE>Working With VBScript:</TITLE>

            <SCRIPT LANGUAGE="VBScript">

                Sub cmdSubmit_OnClick

                   If  (Len(document.frmExample5a.txtAge.value)  =  0)
Then
                      MsgBox  "You  must  enter  your  age  before
submitting."
                      Exit Sub
                   End If
                   ' Check to see if the user entered a number.

                   If
(Not(IsNumeric(document.frmExample5a.txtAge.value))) Then
                      MsgBox "You must enter a number for your age."
                      Exit Sub
                   End If

                   ' Check to see if the age entered is valid.

                   If (document.frmExample5a.txtAge.value < 0) Or
                      (document.frmExample5a.txtAge.value > 100) Then

                      MsgBox "The age you entered is invalid."

                      Exit Sub
                   End If

                   MsgBox "Thanks for providing your age."
                End Sub
                -->

            </SCRIPT>
         </HEAD>
 <BODY>
     <H1>A VBScript Example on Variables</H1>

     <P> This example demonstrates validation techniques in VBScript.
</P>

     <FORM NAME="frmExample5a">

          Enter your age:<BR>
```

```
            <INPUT TYPE="Text" NAME="txtAge" SIZE="2">
            <BR>
              <INPUT TYPE="Button" NAME="cmdSubmit" ALUE="Submit">
            <BR>

    </FORM>

 </BODY>

</HTML>
```



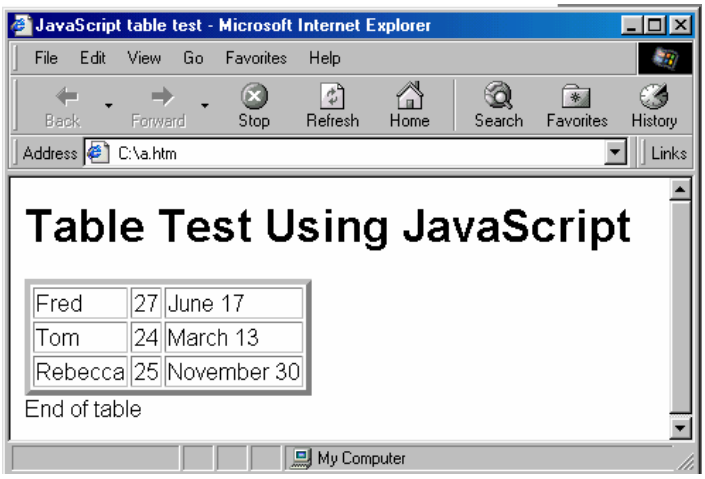
شکل ۲-۳ اجرای برنامهٔ VBSCRIPTقبل

مثال۳:

در این برنامه، یک جدول رسم خواهد شد، طوری که برای رسم هر خط از این جدول، یک فراخوانی به تابع PrintRow، که به زبان جاوا نوشته است، انجام می‌شود. (شکل ۳- ۳)

```
<HTML>
<HEAD>
    <TITLE>JavaScript table test </TITLE>
    <SCRIPT Language = "JAVASCRIPT">
        function printRow(name,age,birthday) {
        document.write("<TR><TD>", name,"</TD><TD>",age,"</TD><TD>",
         birthday,"</TD></TR>\n");
         }
    </SCRIPT>
    </HEAD >
    <BODY>
       <H1> Table Test Using JavaScript </H1>
       <TABLE border=4>
       <SCRIPT LANGUAGE ="JAVASCRIPT">
       printRow ( "Fred" , 27 , "June 17" );
       printRow ( "Tom" , 24 , "March 13");
       printRow ( "Rebecca" , 25 , "November 30 " );
      </SCRIPT>
      </TABLE>
       End of table
```

```
    </Body>
</HTML>
```



شکل ۳-۳ اجرای برنامهٔ Javascript

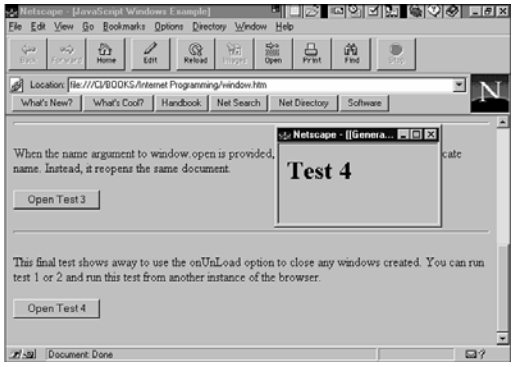
متد Document . Write همانند ( ) Printf در C می‌باشد و از طریق آن یک عبارت در خروجی چاپ می‌شود.

مثال ٤:

این مثال که با جاوا اسکریپت نوشته شده است، طریقهٔ ایجاد یک صفحهٔ جدید همچنین فراخوانی یک تابع خواص در JavaScript را مشخص می‌کند. (شکل ٤- ۳)

```
<HTML>
<HEAD>
<TITLE>JavaScript Windows Example</TITLE>
<SCRIPT LANGUAGE="JavaScript">

<!--

var win4Open=false

function winTest4() {
  winTst4=window.open("", "winS", "resizable,width=200,height=100")
  winTst4.document.open()
  winTst4.document.write("<H1>Test 4</H1>")
  winTst4.document.close()
  win4Open=true
}

function endIt() {

     if (win4Open) winTst4.close()
}
```

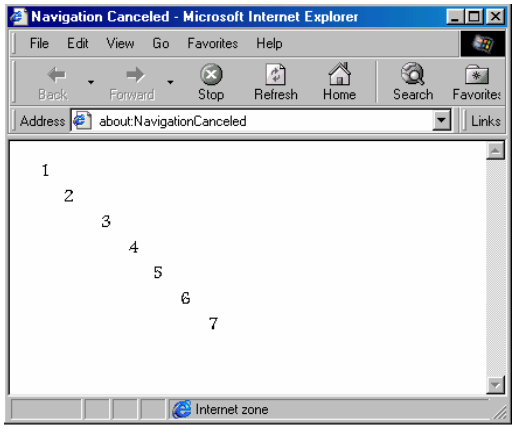
```
// -->
</SCRIPT>

</HEAD>

<BODY onUnLoad='endIt()'>

<H1 ALIGN="CENTER">A JavaScript Windows Example</H1>
<P>HTML lacks the means to create windows. Frames, yes -- windows ,
no. The best way to learn about the behavior of windows is to play
with this script.</P>
<HR>

<FORM>

<P>This window test generates a new instance of the browser,
with this document as the opening document. Close the first instance
of the browser to see what happens.<BR><BR>
<INPUT TYPE="button" NAME="test1" VALUE="Open Test 1"
   onClick='window.open("window.htm")'></P>

<HR>

<P>By declaring a named windowed object, it is still possible to

create multiple instance of the browser.<BR><BR>

<INPUT TYPE="button" NAME="test2" VALUE="Open Test 2"

   onClick='winTST2 = window.open("window.htm")'></P>

<HR>

<P>When the name argument to window.open is provided, it cannot
another copy with a duplicate name. Instead, it reopens the
same document.<BR><BR>

<INPUT TYPE="button" NAME="test3" VALUE="Open Test 3"
onClick='winTST3 = window.open("window.htm", "winTest")'></P>

<HR>
<P>This final test shows away to use the onUnLoad option to close
any windows created. You can run test 1 or 2 and run this test
from another instance of the browser.<BR><BR>

<INPUT TYPE="button" NAME="test4" VALUE="Open Test 4"
   onClick='winTest4()'></P>
</FORM>
</BODY>
</HTML>
```



شکل ٤-۳ خروجی برنامه جاوا اسکریپت بالا

۳-۴ تمرین‌ها

۱. تفاوت HTML و DHTML چیست؟
۲. تفاوت Script های سمت سرویس‌دهنده با سرویس‌گیرنده چیست؟
۳. عملکرد اجرای یک Script سمت سرویس‌گیرنده را با شکل نشان دهید؟
٤. جهت نوشتن یک برنامهٔ کوچک Vbscript در HTML، از چه تگی باید استفاده کرد؟
٥. آرگمان Runat در <Script> چه استفاده‌ای دارد؟
٦. برنامه‌ای باVBScript بنویسید، که یکForm داشته‌باشد، که کاربرحتماً مجبورباشد همهٔ فرم راپرکند.
۷. برنامه‌ای با جاوا اسکریپت بنویسید، که خروجی زیر را از طریق حلقه ایجاد نماید. (از دستورات موجود در ضمیمهٔ کتاب استفاده کنید). (شکل ٥- ۳)



شکل ٥-۳

فصل چهارم

روش‌های دیگر برنامه‌نویسی تحت وب

۴-۱ مقدمه

در فصل‌های گذشته در مورد برنامه‌نویسی به زبان HTML و DHTML صحبت کردیم و بیان کردیم که برنامه‌نویسی روی اینترنت و وب، به دو دسته برنامه‌نویسی سمت سرویس‌دهنده و برنامه‌نویسی سمت سرویس‌گیرنده تقسیم می‌شود. در این فصل روش‌های دیگر برنامه‌نویسی را در حد مقایسه به‌طور بسیار مختصر شرح خواهیم داد.

مطالبی که در این فصل گفته خواهد شد، در رابطه با برنامه‌نویسی به زبان جاوا، ISAPI، ActiveX، Plugins و ASP می‌باشد.

۴-۲ برنامه‌نویسی به زبان جاوا

جاوا زبان برنامه‌نویسی است، که توسط شرکت Sun Microsystem ارائه شده و شباهت‌های بسیاری به زبان C++ دارد. این زبان برنامه‌نویسی را می‌توان در بازه متنوعی از کاربردهای وب و مهم‌تر از همه، در کاربردهای توزیع شده به‌کار گرفت.

برنامه‌های کوچک نوشته شده به زبان جاوا، که اصطلاحاً اپلت[1] نامیده می‌شوند را می‌توان درون صفحات HTML قرار داد. بدین ترتیب این برنامه‌ها همراه صفحات HTML به ماشین کاربر منتقل شده به اجرا درمی‌آیند.

شرکت SUN جهت عمومیت دادن جاوا Source کامپایلر آن را به‌صورت رایگان در اختیار شرکت‌های مختلف خصوصاً شرکت‌های نویسندهٔ مرورگر قرار داد. این باعث شد که بسیار عمومیت پیدا کند، طوری که به عنوان یک زبان برنامه‌نویسی باز (Open) معرفی شود.

با استفاده از جاوا می‌توان کاربردهای توزیع شدهٔ وب به‌وجود آورده، چرا که منطق تجاری را می‌توان در سمت کاربر به اجرا در آورد. همچنین دسترسی به پایگاه‌های داده به‌طور مستقیم توسط مرورگر امکان‌پذیر است. (از طریق روش‌هایی چون RMI).

استانداردی به نام JDBC ایجاد شده است، که امکان دسترسی اپلت‌های جاوا به پایگاه‌های دادهٔ رابطه‌ای را فراهم می‌سازد. با این وجود جاوا برای کاربردهای بزرگ مناسب نیست، ولی می‌توان حداقل بعضی از قسمت‌های سمت کاربر را با استفاده از جاوا پیاده‌سازی کرد.

---

1. Applet

برنامه‌نویسی با جاوا به دو صورت انجام می‌گیرد:

۱. برنامه‌های کاربردی (Application)

۲. اپلت‌ها

برنامه‌های کاربردی، که به‌صورت EXE هستند، بعد از ترجمهٔ برنامهٔ جاوا ایجاد می‌شوند. جهت ایجاد آنها از کامپایلرجاوا استفاده می‌شود. شکل ترجمهٔ یک برنامهٔ جاوا در سیستم‌های Unix به‌صورت زیر است:

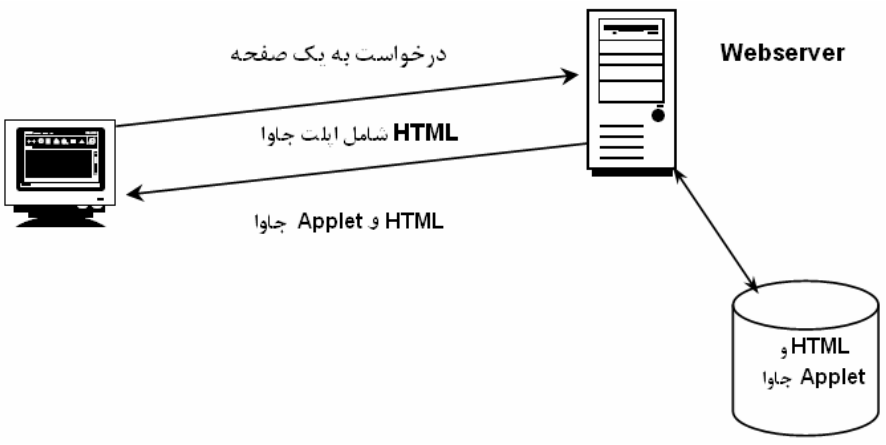
```
$ Javac hello.java
```

ولی خروجی یک اپلت بعد از کامپایل، توسط کامپایلر یک Bytecode می‌شود، که این کد روی هر ماشینی که در آن ماشین مجازی جاوا[1] موجود باشد، قابل اجرا است. این برنامه زبان ماشین نیست، بلکه یک کد میانی است، که قابل حمل به هر ماشینی می‌باشد واین خاصیت باعث قابل حمل شدن اپلت‌های جاوا می‌شود. (شکل ۱- ٤)

شکل ۱-٤ طریقهٔ اجرای برنامه‌های جاوا در محیط Command Prompt

***توجه** : اپلت‌های جاوا، هم امکان برنامه‌نویسی سمت سرویس‌گیرنده را فراهم می‌کند و هم امکان برنامه-نویسی سمت سرویس‌دهنده، که این روش برنامه‌نویسی، یعنی نوشتن اپلت سمت سرویس‌دهنده را Servlet گویند. (شکل ۲- ٤)*

1. Java Vitual Machin



شکل۲-٤ ارتباط اپلت‌های جاوا با سرویس‌دهندهٔ وب

عملکرد اپلت‌های جاوا

هنگامی که درخواستی به یک صفحه شامل اپلت جاوا ارسال می‌شود، مراحل زیر رخ خواهد داد:

۱. ارسال درخواست به سرویس‌دهنده، جهت انتقال صفحه و اجزای آن.
۲. انتقال صفحهٔ HTML و خود اپلت جاوا با پسوند class به سوی client.
۳. اجزای اپلت‌جاوا روی سرویس‌گیرنده ازطریق ماشین مجازی جاوا، روی مرورگر و نمایش خروجی روی مرورگر.

به‌کار بردن اپلت‌ها در صفحات وب

اپلت‌ها نیز مانند کنترل‌های Activex، می‌توانند هم روی سرویس‌دهنده اجرا شوند و هم روی سرویس-گیرنده، ولی بحث ما در مورد اپلت سمت سرویس‌گیرنده است. در HTML، برچسبی به نام <Applet> وجود دارد، که از طریق آن می‌توان یک Applet جاوا را فعال ساخت. مانند مثال زیر:

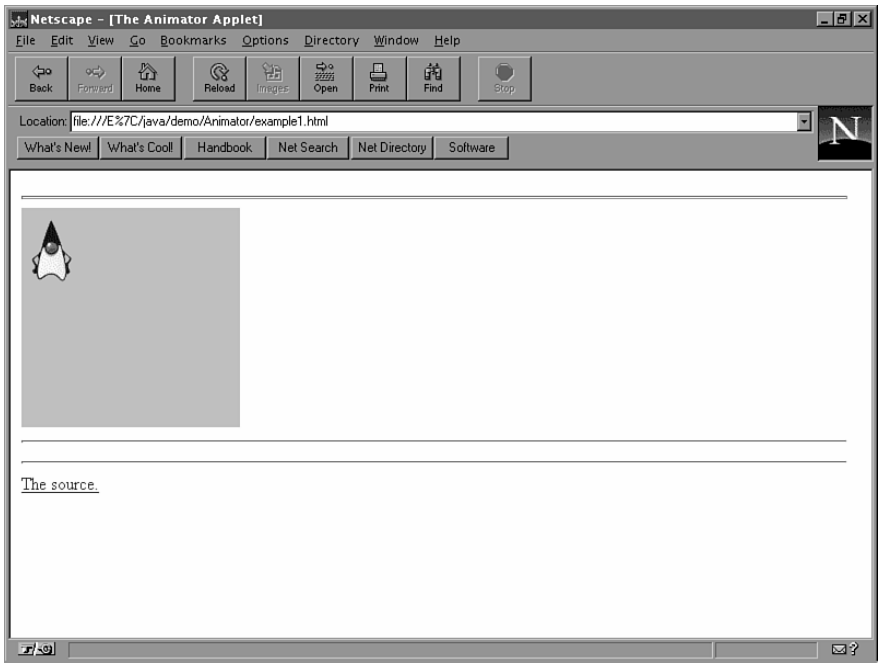
```
<HTML>

<Body>

<Applet   Codebase="Http://www.Applet.Com/Hello.Class"          Width =
"500 "
       Height    = "500">
<Param  Name = " Ver"  Value = "3.1">

</Applet>

</Body>

</HTML>
```

در تگ <Applet> آرگمان Codebase، آدرس برنامهٔ اپلت و Width و Height، ابعاد بخشی از مرورگر می‌باشند، که اپلت در آنجا اجرا می‌شود.

***توجه** : خصوصیتی که اپلت‌ها دارند، این می‌باشد، که برای دیدن یک صفحه شامل اپلت خود برنامهٔ اپلت با پسوند Class روی سرویس‌گیرنده انتقال می‌یابد. حال اگر صفحهٔ دیگری از همان اپلت مجدداً استفاده کرده باشد، باز هم اپلت باید روی سرویس‌گیرنده Download شود، که این بسیار زمان‌گیر می‌باشد.*

یعنی در واقع می‌توان گفت اپلت‌های جاوا خود را روی سرویس‌گیرنده، register نمی‌کنند.

در Activex های سمت سرویس‌گیرنده، که بعداً توضیح خواهد شد، نیز عملکرد بدین‌صورت است، با این تفاوت که آنها خود را روی register سیستم عامل ثبت می‌کنند و عمل انتقال تنها یک بار صورت می‌گیرد. در مورد کنترل‌های Activex بعداً شرح داده خواهد شد. (شکل ۳- ٤)



شکل۳-٤ اجرای یک اپلت جاوا

طریقهٔ نوشتن اپلت‌های جاوا

برخلاف برنامه‌نویسی جاوا در حالت Application، که یک تابع اصلی داریم و توابع فرعی را فراخوانی خواهیم کرد، جهت نوشتن یک اپلت کلاً برنامه‌نویسی متفاوت است، طوری که باید از یک کلاس خاص به نام Applet استفاده نمود. برای این منظور باید از کلاس اپلت در جاوا یک زیر کلاس به ارث ببرید. مثال زیر:

```
Public    Class    Myclass    Extends    Java.applet.Applet
{
    _____   Properties
    _____   Methods
    _____   Program
 }
```

توجه: *در اپلت‌ها نام زیرکلاس (Myclass) باید با نام برنامهٔ (Myclass.Java) یکی باشد، که بعد از کامپایل فایل Myclass.class ایجاد شود.*

کلاس اپلت در جاوا دارای تعدادی متد می‌باشد، که این متدها به ترتیب اجرا می‌شوند، حال اگر شما متدها را دوباره تعریف کنید، از متدهای شما استفاده خواهد شد. در غیر این صورت از همان متدهای پیش‌فرض استفاده می‌گردد.

این متدها عبارتند از :

۱. init ( )

جهت مقداردهی اولیه می‌باشد، که دارای خصوصیات زیر است.

- فقط یک بار اجرا می‌شود.
- متغیرها را در این قسمت تعریف و مقداردهی اولیه می‌کنیم.
- برای گرفتن پارامترها از برنامهٔ HTML از این قسمت استفاده می‌شود. این کار با استفاده از دستور ( )Get parameter انجام می‌شود.

۲. Start ( )

این متد به‌صورت خودکار، بعد از init اجرا می‌شود و بدنهٔ اصلی در این قسمت نوشته می‌شود، همچنین هنگامی که اپلت متوقف شده باشد و دوباره فعال شود، این متد اجرا خواهد شد.

۳. Paint ( )

این متد جهت قرار دادن اطلاعات روی ناحیه‌ای از مرورگر که مشخص شده است به‌کار می‌رود. ورودی این متد، یک نمونه از شیء Graphics می‌باشد.

```
Paint ( Graphics g )
```

حال می‌توان از متدهای شیء Graphics، برای قرار دادن اطلاعات روی صفحه استفاده کرد.

٤. Stop ( )

این متد اجرای اپلت را متوقف می‌کند، ولی منابع آن را آزاد نمی‌کند. قبل از Destroy کردن یک اپلت همواره باید آن را متوقف کرد (Stop).

۵. Destory ( )

این تابع با بستن پنجرهٔ مرورگری که اپلت را اجرا می‌کند رخ خواهد داد، طوری که حافظهٔ اختصاص داده شده، ProcessTime و Swap Disk Space و سایر منابع آزاد می‌گردد.

چند مثال از اپلت

مثال ۱:

در این مثال که Speaker نامیده می‌شود، یک تصویر گرافیکی نمایش داده شده و فایل صدا نواخته می‌شود.

```
/*      Speaker

     This applet displays a gif or jpeg while playing a sound (.au)
file.
```

```
*/
import java.awt.*;
import java.applet.*;
import java.lang.*;
import java.net.URL;
public class Speaker extends java.applet.Applet {

     Image image;
     AudioClip sound;
     String graphic;
     String clip;
     public String[][] getParameterInfo() {
          String[][] info = {
               {“graphic”,     “string”,     “The image file to be
displayed.”},
               {“clip”,     “string”,     “The sound file to be
played.”},
          };
     return info;
     }
     public void init() {
          graphic = getParameter(“graphic”);
          clip = getParameter(“clip”);
          image = getImage(getDocumentBase(), graphic);
          sound = getAudioClip(getDocumentBase(), clip);
     }
     public void paint(Graphics g) {
               g.drawImage(image, 0, 0, this);
         }
```

```
     public void start() {
          repaint();

          // This could also be sound.loop(); to loop the sound.
          sound.play();
     }
     public void stop() {
          sound.stop();
     }
}
```

مثال ۲:

در این مثال، طریقهٔ کار کردن با رنگ‌ها و ترکیب رنگ‌ها در جاوا اسکریپت را می‌بینید.

```
import java.awt.*;

public class ColorCycle extends java.applet.Applet {
         float hue = (float).5;
         float saturation = (float)1;
         float brightness = (float)0;
         Button b;


public void init() {
        b = new Button("Next Color");
        add(b);
}
public void start() {
            setBackground(Color.black);
            repaint();
}
public boolean action(Event evt, Object o) {
        if (brightness < 1)
              brightness += .25;
        else
```

```
                brightness = 0;

        Color c = new Color(Color.HSBtoRGB(hue, saturation,
brightness));

        setBackground(c);

         repaint();

         return true;

     }

  }
```

کد مربوط به بخش HTML هر کدام از اپلت‌ها می‌تواند مانند زیر باشد:

```
<html>

    <body>

           <applet
codebase="http://www.applet.com/class/ColorCycle.class" height=250
width=250>

           </applet>

   </body>

</html>
```

۴-۳ ISAPI (Internet Server Application Programming)

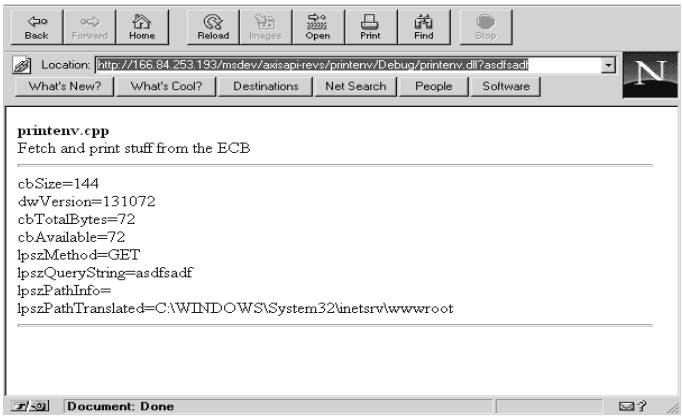
یک روش برنامه‌نویسی برای سرویس‌دهندهٔ مایکروسافت می‌باشد و عملکرد آن مانند Cgi است، یعنی شما یک HTML و یک برنامهٔ دروازه‌ای دارید، که در HTML از طریق یک Form یا SSI آن را فراخوانی می‌کنید. ISAPI هم مثل Cgi است، با این تفاوت که در Cgi برنامه‌های دروازه‌ای ExE هستند، ولی در ISAPI، DLL می‌باشند.

- به ازای یک درخواست از سرویس‌گیرنده، به یک برنامهٔ دروازه‌ای Cgi یک process روی سرویس-دهنده با یک ناحیهٔ کاری جدا و حافظهٔ جدا فعال می‌گردد و این کار با درخواست‌های بعدی مرتب تکرار می‌شود. این امر سبب خواهد شد، که در صورت زیاد شدن تعداد درخواست‌ها، سرویس-دهنده بسیار کند شود.
- در برنامه‌های دروازه‌ای ISAPI، که از نوع DII هستند، فقط یک بار برنامه در حافظه بار می‌شود. به ازای درخواست‌های بعدی، فقط یک thread از آن در حافظه فعال خواهد شد. همان‌طور که می-دانید thread کوچک‌ترین واحد پردازشی مجزا در سیستم عامل‌های Multi thread می‌باشد، که منابع بسیار کمتری نسبت به process اشغال می‌کند.

مثال :

می‌توان همانند cgi، از طریق یک form درخواستی به یک برنامهٔ Dll روی سرویس‌دهنده ارسال کرد. (شکل ٤- ٤)

```
       < Form  Method = " Post "     Action =
"http://www.sample.com/cgi-bin/hello.dll">
                .
                .
                .
                .
                .
      < / form >
```



شکل٤-٤ مثالی از یک فراخوانی ISAPI

Activex ۴-۴

برنامه‌نویسی ماژولار، سال‌ها برنامه‌نویسان را به خود مشغول کرده بود، لذا محرک اصلی که به توسعهٔ سیستم عامل ویندوز منجر شده، نیز کدهای قابل اشتراک و قابل استفاده مجدد بود. اولین گام در راه پیاده‌سازی عناصر ماژولار، تکنولوژی OLE[1] بود. هدف اولیهٔ OLE، ایجاد مستندات مرکب[2] با استفاده از برنامه‌های مختلف بود. به عنوان مثال، سندی که مقداری متن و نمودار دارد که هر کدام با نرم‌افزارهای خاص خود ایجاد شده اند، نمونه ای از یک سند مرکب است. در چنین مثالی، هنگامی که با متن کار می‌شود نرم‌افزار واژه‌پرداز کنترل را به دست می‌گیرد و هنگامی که با نمودارها کار صورت می‌گیرد، نرم‌افزار ترسیمی مسئولیت را به‌عهده می‌گیرد. ایراد اصلی OLE، سرعت کم آن بود.

تکنولوژی OLE بر مبنای استاندارد کلی‌تری به نام COM (Component Object Model) بنا شده بود. با پیشرفت تکنولوژی، تکنولوژی COM نیز توسعه یافت و از سطح سندهای مرکب فراتر رفت و DCOM (Distributed Com) پدید آمد. ویرایش های قدیمی Office ، Visual basic از این تکنولوژی استفاده می‌کنند.

1. Object Linking and Embedding
2. Compound documents

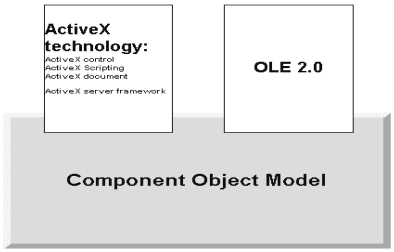
امروزه به بخشی از تکنولوژی‌های COM، که در آنها یک قطعه نرم‌افزاری، امکانات خود را در اختیار برنامه‌های دیگر گذارد و یا توانایی پشتیبانی از نرم‌افزارهای توزیع شده را داشته باشد، Activex گفته می‌شود.

Activex در سال ۱۹۹٦ به‌عنوان استراتژی اصلی مایکروسافت برای اشیاء توزیع شده و وب ارائه شد. خصوصیات اصلی Activex عبارتند از:

مؤلفه‌های Activex از COM وDCOM برای ارتباط با یکدیگر استفاده می‌کنند. به‌عبارت دیگر DCOM نقش ORB را در محیط‌های مبتنی بر Activex بازی می‌کند.

مرورگر می‌تواند همچون یک Container عمل کند. برای مثال Internet Explorer می‌تواند محتوای مؤلفه‌هایی همچون اسناد Word ، اپلت‌های جاوا و کدهای HTML باشد.

در کاربردهای توزیع شده می‌توان تکنولوژی Web (مرورگر، صفحات HTML ، اپلت‌های جاوا) را با ابزارهای رومیزی (صفحه گسترده‌ها، پردازشگرهای متن) در هم آمیخت. (شکل ٥- ٤)



شکل ٥-٤ ارتباط Activex،OLE،Com

کاربران Activex می‌توانند مؤلفه‌هایی همچون سرویس‌دهندهٔ SQL و پورت‌های وب را فراخوانی کنند.

انواع Activex

نکته‌ای که باید به آن دقت بسیار نمود، آن است که Activex امروزه صورت‌های گوناگونی به خود گرفته است، که اگر چه با یک نام خوانده می‌شوند، ولی کاربردهای متفاوتی دارند. ریشهٔ آن را باید در تعریف بسیار کلی مایکروسافت از Activex جستجو کرد:

Activex، نام تجاری ارائه شده توسط مایکروسافت برای تکنولوژی‌هایی است که ارتباط داخلی بین کاربردها را با استفاده از مدل COM فراهم می‌آورند. تکنولوژی Activex دربردارندهٔ "یک یا بیشتر" واسط است، که هر یک از آنها دارای "یک یا بیشتر" Property و Method هستند. تمامی واسطه‌های COM از کلاس پایه unknown منشعب شده‌اند. تکنولوژی‌هایی که بر مبنای مدل COM بنا شده‌اند عبارتند از: Activex و MAPI و OLE .

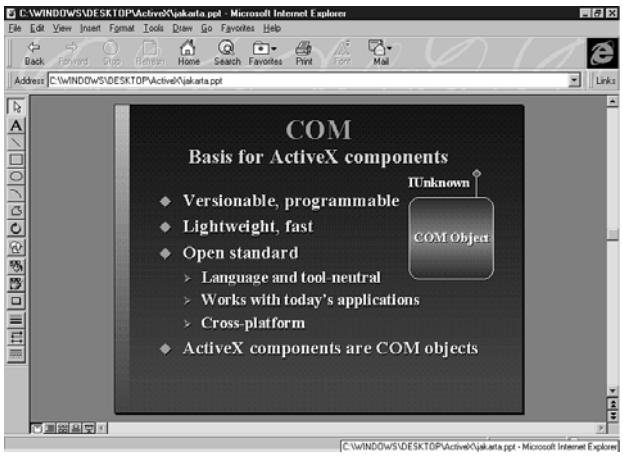
در وب امکان استفاده از این کنترل‌های Activex، به عنوان اشیای طراحی شده با زبان‌های ویژوال وجود دارد. می‌توان از طریق زبان‌های ویژوال، اشیاء موردنظر را طراحی کرد و به‌صورت فایل‌های OCX درآورده و در HTML استفاده کرد.

کنترل‌های Activex، نیز همانند اسکریپت‌ها می‌توانند سمت سرویس‌گیرنده یا سرویس‌دهنده باشند. درحالت سمت سرویس‌گیرنده، خود برنامهٔ OCX به‌سیستم سرویس‌گیرنده منتقل شده و روی Registery سیستم عامل سرویس‌گیرندهٔ خود را ثبت می‌کند. حال اگر دوباره درخواستی به آن ارسال شود، فایل OCX نیازی به بار شدن دوباره ندارد، درصورتی که برای اپلت‌های جاوا این مسئله وجود ندارد.

برای استفاده از کنترل‌های Activex، سمت سرویس‌گیرنده از برچسب <object> استفاده می‌شود. این تگ نیز مانند <Applet> دارای آرگمان codebase است، که آدرس یا URL کنترل Activex می‌باشد، ولی دارای یک آرگمان دیگر با نام ClassID است، که این شناسه همان نامی می‌باشد، که کنترل Activex خود را با این نام روی سرویس‌گیرنده، Register می‌نماید.

همانند اپلت‌های جاوا، در این حالت نیز می‌توان پارامترهایی را به کنترل Activex، از داخل HTML ارسال نمود. (شکل ٦- ٤)

```
< Object      ID = " XYZ " WIDTH = 100 HELGHT = 100
              ClassID = " ClsID = 593#1A b3DDK3097 "
              Code Base = "Http://www.Sample.Com/Sample.ocx ">
 <Param       Name = " p1 "        Value = "AA">
 <Param       Name = " p2 "        Value = "2.1">

 </Object >
```



شکل ٦-٤ مثالی از یک برنامهٔ Activex

۴-۵ برنامه‌های Plug-Ins

Plug_In نویسی، سوپاپ برنامه‌نویسی تحت وب می‌باشد، یعنی اگر از هیچ روشی نتوانیم سیستم خود را تحت وب پیاده کنیم، در این حالت باید امکانی را به مرورگر اضافه کنیم، که برنامه‌ها را اجرا کند.

Plug_In ها برنامه‌هایی هستند، که توسط شرکت‌های مختلف نوشته می‌شوند و این امکان را به مرورگر می‌دهد، که بتواند فایل‌هایی را با پسوند خاص که در حالت عادی قادر به نمایش آنها نیست، نمایش دهد، مثلاً فایل‌هایی به‌صورت زیر:

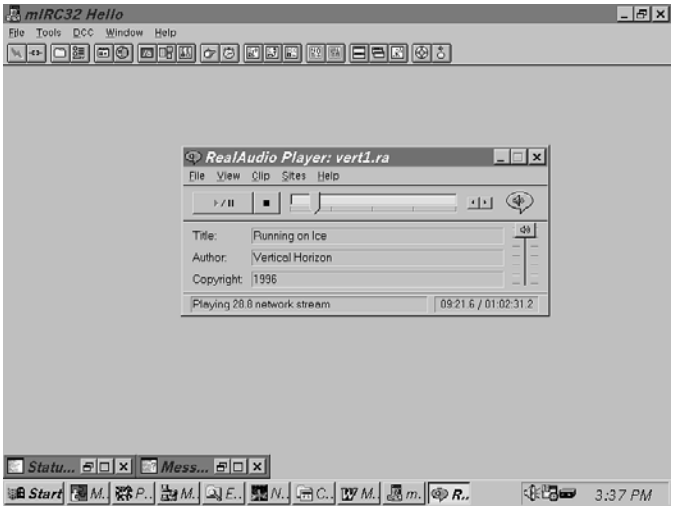
```
Adobe Acrobat    ( PDF) Portable Document Format
Macromedia Director
Quick Time
VebFX VRML
MP3
DAT
```

یک Tag به نام < EMBED > در HTML برای کنترل Plug In ها وجود دارد.

```
< EMBED   SRC =”http:www.server.com/amovie avi”>
```

مرورگر جهت اجرای این دستوراز روی پسوند فایل avi به کمک MIME[1] برنامهٔ اجرایی یا Plug-In آن را پیدا کرده و اجرا می‌کند. (شکل ۷- ٤)

توجه: *باید توجه کرد، که هرگاه در انتقال یک فایل تحت وب سرآیند Content_Type، به مرورگر می‌رسد در این حالت نیز اگر خود قادر به نمایش آن نباشد، از طریق MIIME، Plug_In مربوطه را یافته و فعال می‌کند. در این صورت مرورگر در یک لایهٔ پایین‌تر قرار گرفته و اطلاعات را در اختیار PlPlug_In قرار می‌دهد.*



شکل۷-٤ مثالی از اجرای یک Plugin

1. Multiperpouse Internet Message Extention

۴-۶ Asp (Active Server Page)

شرکت مایکروسافت از طریق این روش بهترین ویژگی‌های روش‌های قبلی مثل دسترسی آسان به بانک‌های اطلاعاتی، دسترسی به سیستم عامل و سرعت و امنیت بالا را مورد توجه قرار داده است. همان‌طور که قبلاً گفته شد، در:

برنامه‌نویسی وب دو نوع اسکریپت داریم:

- اسکریپت سمت سرویس‌دهنده (روی سرویس‌دهنده اجرا می‌شوند)
- اسکریپت سمت سرویس‌گیرنده (روی سرویس‌گیرنده اجرا می‌شوند)

Asp محیطی مبتنی بر اسکریپت نویسی سمت سرویس‌دهنده می‌باشد، که موجب ساخت برنامه‌های پویا و محاوره‌ای می‌شود. این برنامه‌ها زمان زیادی را برای پردازش تلف نمی‌کنند و هم برای طراحان حرفه‌ای وب و هم مبتدیان مطلوب می‌باشد.

Asp، چون محصول مایکروسافت است، وابسته به IIS می‌باشد و IIS نسخهٔ ٤ به بالا آن را پشتیبانی می‌کند. Asp همان فایل HTML است و می‌تواند هر چه که HTML پشتیبانی می‌کند، پشتیبانی نماید. مثل اپلت‌های جاوا - متن چشمک زن - اسکریپت‌های سمت سرویس‌گیرنده و Activex های سمت سرویس-گیرنده و... پشتیبانی نماید، ولی دستوراتی در آن وجود دارد که مربوط به اسکریپت‌های سمت سرویس‌دهنده است، که روی سرویس‌دهنده اجرا می‌شود. دستورات Asp، در داخل برچسبی به‌صورت <%___%> قرار می‌گیرند.

خاصیت‌های خاص Asp

Asp دارای خاصیت‌های زیر می‌باشد.

۱. ترکیب دو بخش برنامه‌نویسی HTML و برنامه دروازه‌ای در یک فایل، روی سرویس‌دهنده.
۲. سرعت بالا نسبت به روش‌های دیگر برنامه‌نویسی cgi، ISAPI و....
۳. پشتیبانی از تعداد زیادی شیء داخلی که امکانات بسیار زیادی را به اسکریپت‌های شما می‌دهند.
٤. پشتیبانی از اجزای Activex سمت سرویس‌دهنده جهت اضافه کردن اشیاء جدید به آن.
۵. امنیت بالا به جهت انتقال نیافتن کد برنامه روی سرویس‌دهنده.
٦. پشتیبانی از بانک‌های اطلاعاتی سمت سرویس‌دهنده از طریق SQL.

زبان‌های اسکریپت‌نویسی Asp

توجه به این نکته ضروری است، که Asp یک تکنولوژی است، نه یک زبان، لذا اسکریپت صفحهٔ Asp می‌تواند با هر زبانی (....c, Perl,Jscript,Javascript,Vbscript) نوشته شده باشد و تنها کافی است که سرویس‌دهنده ابزار لازم برای اجرای این اسکریپت را داشته باشد. از آنجا که VB script زبان بسیار ساده‌ای است و از سوی دیگر زبان پیش‌فرض IIS برای اسکریپت‌نویسی سمت سرویس‌دهنده می‌باشد، لذا می‌توان Vbscript را به عنوان زبان تولید اسکریپت‌های سمت سرویس‌دهنده در نظر گرفت. جهت تغییر دادن زبان از Vbscript به Jscript، کافی است دستورات زیر را بنویسید:

مثال :

```
<%@Longuage=Jscript%>
```

همچنین می‌توانید هرجا که می‌خواهید از جاوا استفاده کنید. در این صورت از برچسب script، با آرگمان Runat = Server می‌توان استفاده کرد:

```
<Script Language ="Jscript" RunAt ="Server">
```

۴-۷ تمرین‌ها

۱. تفاوت اپلت‌های جاوا با کنترل‌های Activex در چیست؟ معایب و مزایای هر کدام را شرح دهید.
۲. ASP و CGI چه نقاط مشترک و چه نقاط غیرمشترکی دارند؟
۳. مزایا و معایب cgi را شرح دهید.
٤. ISAPI با cgi چه تفاوتی دارد؟
٥. عملکرد برنامه‌های ASP را با یک شکل شرح دهید.
٦. مراحل مربوط به درخواست یک صفحهٔ HTML شامل اپلت جاوا را شرح دهید.
۷. Plugin چیست و معایب و مزایای آن را شرح دهید.
۸. ActiveX چیست و تفاوت ActiveX های سمت سرویس‌دهنده و سرویس‌گیرنده در چیست؟

فصل پنجم

برنامه‌نویسی Cgi (Common Gateway Interface)

۵-۱ مقدمه

Cgi، یک استاندارد برای برقراری ارتباط بین سند وب و برنامهٔ دروازه‌ای[1] روی سرویس‌دهنده است. Cgi روشی است، که برنامه‌های خارجی می‌توانند با سرویس‌دهنده ارتباط برقرار کنند. به کمک این روش می‌توان هر برنامه‌ای نوشت و هر سیستمی را روی اینترنت پیاده‌سازی کرد.

کارهایی که Cgi می‌تواند انجام دهد:

HTML	CGI+HTML	Task
No	Yes	Handle form
No	Yes	ایجاد هر چیز غیرStatic
No	Yes	Searching
No	Yes	ایجاد برنامه های کاربردی

برای استفاده از Cgi ابتدا باید سرویس‌دهندهٔ وب را آماده نمود، اگر سرویس‌دهندهٔ وب درست نصب شده باشد، یک زیرشاخه با نام ،cgi_bin یا Scripts در زیرشاخهٔ ریشه ایجاد می‌شود، که برنامه-های دروازه‌ای را می‌توان در آن قرار داد.

۵-۲ اجزای یک برنامهٔ CGI

اگر از روش Cgi، جهت برنامه‌نویسی تحت Web استفاده کنید، برنامهٔ شما دو بخش دارد:

```
HTML
Gateway Program
```

۱- برنامهٔ HTML

این بخش اکثر اوقات یک form می‌باشد، که اطلاعات را از کار بر گرفته و برنامهٔ دروازه‌ای را صدا می‌زند و یا از روش SSI استفاده می‌کند (این برنامه روی سرویس‌گیرنده توسط مرورگر اجرا می‌شود)

1. Gateway program

۲- برنامهٔ دروازه‌ای

این برنامه یک برنامهٔ اجرایی (EXE) است، که می‌تواند با هر زبانی نوشته شود و در زیر شاخهٔ Cgi-bin یا Scripts قرار گیرد. (این برنامه روی سرویس‌دهنده اجرا می‌شود و به ازای هر درخواست یک Process از آن روی سرویس‌دهنده، ایجاد می‌شود.)

۵-۳ عملکرد Cgi روی Server

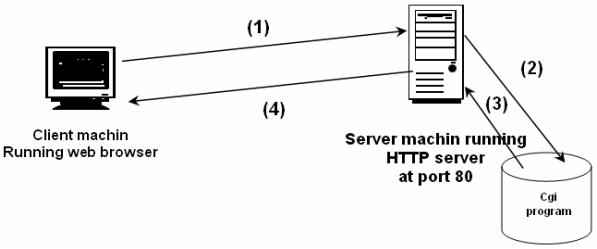
برنامه‌های CGI از نوع Event oriented (واقعه‌گرا) هستند به این معنی که هنگامی که یک سند به آنها دسترسی پیدا می‌کند، شروع می‌شوند.

هنگامی که یک درخواست به برنامهٔ دروازه از طریق SSI (بعداً شرح داده خواهد شد) یا زدن کلید Submit در یک form ارسال می‌شود. سرویس‌دهندهٔ HTTP بلافاصله به زیر شاخهٔ موجود در سیستم خود (Cgi-bin) مراجعه کرده و به دنبال برنامهٔ دروازه‌ای که آدرس آن در " "= Action مشخص شده است، می‌گردد و اگر یافت شود، یک Process ایجاد شده و یک ناحیهٔ کاری و حافظه به آن اختصاص یافته و فعال می‌گردد و پارامترها و متغیرهای محیطی آن Set می‌شوند. برنامهٔ اجرا شده و نتیجهٔ آن که یک HTML است، به مرورگر برمی‌گردد.

***توجه** : برنامه‌های CGI نیاز به فراخوانی‌های خاصی به سیستم عامل سرویس‌دهنده دارند. از این نظر باید روی سیستم عامل خاصی که به‌کار می‌برید تبحر حاصل کنید. اولین برنامه‌های Cgi با Perl و C روی Unix نوشته شدند.*

۵-۴ کاربردهای برنامه‌های CGI

- گرفتن اطلاعات از افراد ملاقات‌کنندهٔ سایت و ذخیرهٔ داده‌ها در یک فایل
- فرستادن اطلاعات از طریق فرم‌ها و ارسال آنها به یک آدرس خاص از طریق e-mail
- به‌روز در آوردن فایل‌های محلی روی سرویس‌دهنده
- تولید اطلاعات خودکار به‌صورت گرافیکی یا متنی
- ایجاد سطح امنیت
- تولید گرافیک یا تغییر فایل‌های گرافیکی به‌طور خودکار (شکل ۱- ۵)



شکل ۱-۵ عملکرد برنامه‌های Cgi

همانند شکل، مراحل عملکرد cgi به‌صورت زیر است:

۱. ابتدا درخواستی جهت اجرای برنامه دروازه‌ای به سرویس‌دهنده ارسال می‌شود و اطلاعات نیز در صورت وجود همراه آن ارسال می‌شود.
۲. سرویس‌دهندهٔ HTTP به ازای درخواست رسیده یک Process ایجاد کرده و اطلاعات را بسته به متد در اختیار آن متد قرار می‌دهد.
۳. خروجی برنامه دروازه‌ای به جای نمایش روی مانیتور سرویس‌دهنده، در اختیار سرویس‌دهنده HTTP قرار می‌گیرد.
٤. سرویس‌دهندهٔ HTTP، اطلاعات را به مرورگر برمی‌گرداند.

۵-۵ روش‌های صدا زدن برنامه‌های دروازه‌ای

الف - استفاده form

این کار از طریق form قابل انجام است. در این صورت حتماً باید از یک کلید، Submit وجود داشته باشد، که با زدن آن اطلاعات به برنامهٔ دروازه‌ای ارسال گردد.

```
 <HTML>
 <BODY>
< form action =" http://www.cgi-free.com/cgi-bin/cgiprogram.cgi "
method =" post " >
What is your Message: <Textarea name =" name">     < / Textarea >
< Input type ="submit"  Value = "send"> < / Input >
< Input type ="Reset"   Value = "Reset"> </ Input >
</Form >
</ Body>
 </ HTML>
```

ب- روش SSI (Server side include)

یکی از روش‌های فراخوانی برنامه‌های دروازه‌ای همان‌طور، که گفتیم از طریق form بود، طوری که در form حتماً باید کلید Submit باشد، که با زدن آن اطلاعات به برنامهٔ دروازه‌ای ارسال شود. بعضی اوقات لازم است بدون زدن هیچ کلیدی با دیده شدن صفحه در مرورگر، درخواستی به برنامهٔ دروازه‌ای ارسال گردد. این روش SSI می‌باشد.

در SSI از دستور EXEC #در HTML استفاده می‌شود، که به‌صورت زیر استفاده می‌شود. البته می‌توان در مرورگرهایی، که EXEC# را پشتیبانی نمی‌کنند، از دستور IMG نیز استفاده کرد، طوری که در آرگمان SRC در IMG نام برنامهٔ دروازه‌ای را می‌نویسیم:

```
 <HTML>
 <HEAD>
 <BODY>
 <TITLE> Counter program page </ TITLE>
 </ HEAD>
 See How Many hits this page has taken :
<IMG SRC =" http://www.sample.com/cgi_bin/Test.cgi">
```

```
<HR >
 </Body>
 </ HTML>
```

بلافاصله با View شدن این صفحه توسط مرورگر، یک درخواست ساخته شده و به برنامهٔ دروازه‌ای که آدرس آن مشخص شده است، ارسال خواهد شد.

۶-۵ متدهای CGI

متدهای ارسال اطلاعات به برنامهٔ دروازه‌ای روی سرویس‌دهنده چندین روش دارد، که دو تا از مهم‌ترین این روش‌ها عبارتند از:

Get

در این حالت برنامهٔ CGI داده‌ها را از متغیر محیطی[1] QUERY-STRING دریافت خواهد کرد. حال برنامهٔ دروازه‌ای باید رشته را پویش[2] کرده و داده‌ها را بیرون بکشد.

در این روش محدودیت ۱۰۲٤ بایت (محدودیت URL) داریم، زیرا اطلاعات ارسالی نمی‌تواند بیشتر از ۱۰۲٤ بایت باشد. (می‌توان این متغیر محیطی را باتابع ( ) getenv در C گرفت و پویش کرد، تا داده‌ها به‌دست آیند)

مثال:

```
strcpy(str , getenv("QUERY_STRING");
```

Post

در این حالت سرویس‌دهندهٔ وب داده‌ها را از طریق ورودی استاندارد یا stdin[3] به برنامه CGI انتقال می‌دهد. سرویس‌دهنده می‌تواند پایان داده‌ها را با یک کاراکتر EOF علامت‌گذاری کند، بنابراین برنامه باید از یک مقدار CONTENT-LENGTE برای خواندن ورودی استاندارد به‌صورت درست استفاده کند. اگر شما بخواهید داده‌های بیشتر از ۱۰۲٤ بایت را ارسال کنید، باید ازاین روش استفاده کنید.

مثال :

```
for (i=0 ; i<n ; i++)
      str[i] = getchar();
      str[i + 1] = '\0';
```

1. Environment Variable
2. Parse
3. Standad input

۵-۷ متغیرهای محیطی

هر سرویس‌دهندهٔ وب دارای تعدادی متغیر خاص می‌باشد، که هنگامی که برنامهٔ دروازه‌ای توسط سرویس‌دهندهٔ وب فعال می‌گردد، این متغیرها مقداردهی می‌شوند. لیست کامل این متغیرها در ضمیمهٔ کتاب بیان شده است. بعضی از این متغیرها عبارتند از:

QUERY_STRING

اطلاعات ارسالی از مرورگر به برنامهٔ دروازه‌ای، اگر از روش Get استفاده شده باشد، در این متغیر قرار خواهد گرفت. این متغیر در روش Post خالی می‌باشد، اطلاعات با علامت؟ شروع می‌شوند که بعد ازURL قرار می‌گیرد.

CONTENT_LENGTH

طول اطلاعات ارسالی به برنامهٔ CGI می‌باشد (برحسب بایت)، که در حالت Post مقداردهی می‌شود. اگر از روش Get استفاده شده باشد، این متغیر خالی است. باید توجه کرد که از آنجا که خروجی یک رشته کاراکتری است، باید به عدد تبدیل شود. این کار با تابع ()atoi در C انجام می‌شود.

مثال :

```
strcpy(num,getenv("CONTENT_LENGTH"));
n = atoi(num);
```

Content-Type

نوع یا Type اطلاعات ارسالی از سرویس‌دهندهٔ وب به مرورگر را مشخص می‌کند و مربوط به MIME است (هرگاه در برنامهٔ دروازه‌ای بخواهید اطلاعاتی را به مرورگر ارسال نمایید ابتدا باید Content-Type را ارسال کنید). به‌طور کامل نوع‌های مختلف MIME در ضمیمه بیان شده است، ولی بعضی از آنها عبارتند از:

Msword Zip pdf	application
Midi wav	audio
Aiff Gif Jpeg bmp	image
Plain Html	text
Mpeg Quick time	video

مثال :

```
Print f( "Content-Type:    Text / plain    \n\n")
```

این اطلاعات به مرورگر ارسال می‌شود، مرورگر خود را آمادهٔ دریافت این فرمت فایل خواهد کرد.
* این سرآیند همواره از سرویس‌دهندهٔ وب به مرورگر ارسال می‌شود.

PATH_INFO

اطلاعات اضافی Path، شامل نام برنامهٔ CGI و URL آن است، که برنامهٔ دروازه‌ای می‌تواند ازآن استفاده کند.

REMOTE_HOST, REMOTE_ADDR

آدرس IP کامپیوتری، که درخواست ارسال کرده است در REMOTE_ADDR قرار دارد و آدرس IP کامپیوتری که به آن درخواست ارسال شده در REMOTE_HOST قرار دارد.

REMOTE_IDENT

نام کاربر (User name) را با استاندارد RFC931 می‌دهد.

REQUEST_METHOD

روش ارسال اطلاعات را می‌دهد که یا Get است یا Post. از طریق این متغیر محیطی می‌توان برنامه‌هایی نوشت، که با هر دو روش کارکنند.

SERVER_NAME

نام حوزه یا آدرس IP سرویس‌دهنده است.

SERVER_PORT

شمارهٔ پورت مربوط به سرویس‌دهندهٔ وب می‌باشد، که معمولاً ۸۰ است.

SERVER_PROTOCOL

نام و نسخهٔ پروتکل سرویس‌دهندهٔ وب می‌باشد، مثلاً (http / 1.0).

SERVER_SOFTWARE

نام سرویس‌دهندهٔ وب، که برنامه CGI را اجرا می‌کند، مثلاً (NCSA / 1.5bs).
همچنین کامپیوترهای سرویس‌گیرنده ممکن است سرآیندهای HTTP را به برنامهٔ CGI ارسال کنند. این متغیرها با <HTTP> شروع می‌شوند، که گروهی از آنها در زیر بیان شده است.

HTTP_ ACCEPT

شامل لیستی از انواع فایل‌هایی که مرورگر می‌تواند از سرویس‌دهندهٔ وب دریافت کند:

```
audio / basic , image / gif , text  / plain
```

HTTP_USER_AGENT

نام مرورگری است، که به سند دسترسی پیدا کرده است، مثلاً Netscape 2.0 یا Internet Explorer 5

مثال :

```
strcpy (str , getenv ("HTTP_USER_AGENT "));
```

۸-۵ زبان‌های برنامه‌نویسی مورد استفاده در cgi

زبانی که انتخاب می‌کنید، دارای محدودیت‌هایی می‌باشد که به سرویس‌دهنده بستگی دارد. اغلب برنامه-های کاربردی سرویس‌دهندهٔ وب، از طریق cgi در اینترنت با یکی از سه زبان Perl ، ویژوال بیسیک و C نوشته می‌شوند.

(Practical Extraction language and Reporty) Perl

Perl زبان تفسیری است (کامپایلر ندارد) یعنی در آن حلقهٔ ویرایش، کامپایل و آزمایش سریع‌تر می‌شود. اولین خط اشاره‌گر به مفسر خودش است و ابتدا مفسر را به حافظه آورده و سپس برنامه را اجرا می‌کند. Perl برای کارهای کوچک بهترین است، ولی برای برنامه‌های بزرگ‌تر چندان خوب عمل نمی‌کند.

بیشتر در سیستم‌های مبتنی بر Unix از آن استفاده می‌شود. روی اینترنت اگر از سیستم عامل دیگری غیر از Unix استفاده می‌کنید، می‌توانید برنامه را روی سیستم نوشته و با FTP آن را به Unix ارسال کنید (همچنین از Telnet نیز می‌توان استفاده کرد).

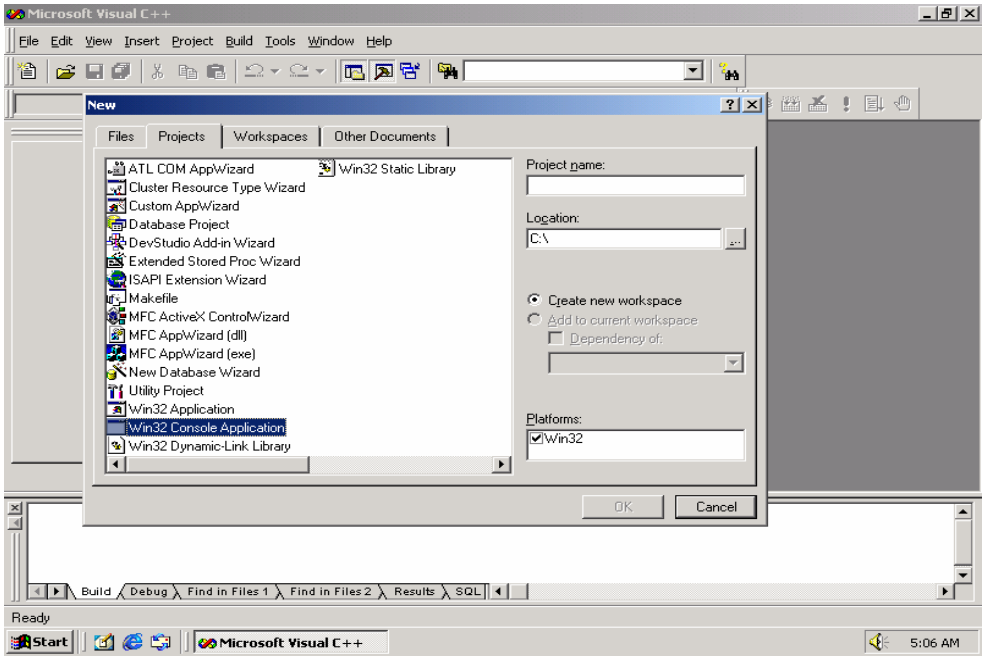
C++ و C

C++ از C متولد شده و یک مرحله بالاتر از C است C و++C روی خیلی از سیستم‌ها از Pc گرفته، تا Mac و Unix قابل انتقال است، البته C نیاز به کامپایلر دارد و حافظهٔ کمتری را نسبت به Perl مصرف می‌کند.

برنامه نوشتن با C دشوار است، زیرا کنترل همهٔ جنبه‌های برنامه‌نویسی و سیستم، برعهدهٔ برنامه-نویس است، ولی از نظر اجرا یک برنامه کامپایل شده، مطمئن‌تر از Perl است.

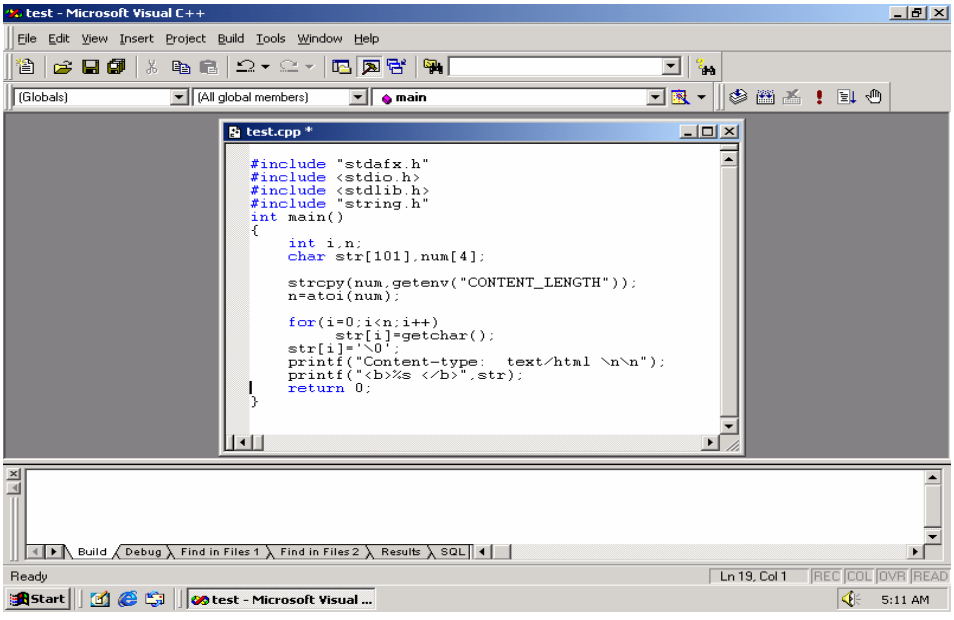
باید توجه کرد که جهت نوشتن برنامه‌های cgi از طریق C باید از یک کامپایلر C روی سرویس‌دهنده خود استفاده کنید. مثلاً اگر سرویس‌دهنده شما Advance ServerWindows2000، باشد باید برنامه‌های C را روی این ماشین به‌صورت ۳۲ بیتی کامپایل کنید. برای این منظور می‌توان از VisualC نصب شده روی سرویس‌دهندهٔ Win2000 استفاده کرد.

در VisualC کافی است یک پروژه به‌صورت Win32 Consols Application تعریف کرده و برنامه‌نویسی را در این محیط انجام دهیم. شکل زیر طریقهٔ ایجاد یک برنامهٔ کاربردی در این حالت را نشان می‌دهد. (شکل ۲- ۵)



شکل ۲-۵ ایجاد یک Win32 Consols Application

حال کافی است برنامهٔ مربوطه را نوشته و کامپایل نماییم، سپس برنامهٔ EXE را در زیر شاخهٔ Scripts قرار می‌دهیم. (شکل ۳- ۵)



شکل ۳-۵ محیط VisualC برای برنامه‌نویسی

ویژوال بیسیک (Visual Basic)

بیسیک در اواسط سال ۱۹۶۰ در کالج Portmouth، جهت تعلیم و آموزش تولید شد، ولی ویژوال بیسیک چندین سال بعد در سال ۱۹۹۱ ایجاد شد، که اساس کار آن بر پایهٔ[1] رخداد بود. از طریق OLE[2]، VB می‌تواند با برنامه‌های دیگر مایکروسافت، مثل EXEL یا ACCESS ارتباط داشته باشد.

البته برنامه‌نویسی VB برای اینترنت تقریباً غیر ویژوال است و باید کدهای برنامه نوشته شود.

جهت مقایسهٔ این سه زبان می‌توان گفت: Perl بسیار سازگار با Unix است ودر به‌کارگیری حافظه، کارایی ندارد و صلاحیت کمی در پشتیبانی از سیستم‌های مختلف دارد.

C++ زبان قابل حمل روی سیستم‌های مختلف بوده، ولی اشتباهات کوچک در برنامه‌نویسی باعث خطاهای بزرگی می‌شود.

VB نیزاز OLE استفاده می‌کند، ولی محدود به سکوهای مایکروسافت است.

۵-۹ چند مثال از CGI

برنامهٔ شمارندهٔ صفحات وب

این برنامه تعداد دفعات ملاقات یک صفحه وب را می‌شمارد .به این برنامه Hit counter می‌گویند. در این برنامه کافی است، که یک ناحیهٔ مشترک روی سرویس‌دهنده را در نظر بگیریم (مثلاً یک فایل مشترک) و شمارنده را درآن قرار دهیم. حال هر کاربر که صفحه را بخواهد ببیند، به‌صورت SSI درخواستی جهت اجرای برنامهٔ دروازه‌ای روی سرویس‌دهنده ارسال می‌کند و این برنامه شمارنده را از فایل خوانده و یکی اضافه نموده و برای کاربر ارسال می‌کند.

الف) بخش HTML

```
<HTML>
 <BODY>
     HitNumber of : <BR>
     <HR>
     < IMG SRL = "http://www.sample.com/cgi _bin/sample.cgi ">
</BODY>
</HTML >
```

ب) بخش Server برنامهٔ دروازه‌ای

این بخش همان‌طور که مشخص می‌باشد، فایلی با نام hit.dat روی سرویس‌دهنده را در حالت Append باز کرده و اطلاعات را از آن می خواند و سپس آن را به‌روزرسانی می‌کند.

باید توجه کرد که هرگاه بخواهیم خروجی به مرورگر ارسال کنیم، ابتدا باید سرآیند Context_typ را ارسال نماییم.

```
#include <stdio.h >
```

1. Event
2. Object Linking Embedding

```
#include <stdlib.h>

  void main(  )
{
       int  k ;

       FILE  *f;

       f = fopen ( "c:\test\hit.txt " , "a" );

       fscanf ( f , "%d" , &k );

       printf ( "Content -type :   Text / htm  \n\n" );

       printf ( "<B> %d </ B> " , K+1);

       fseek ( f , 0);  // به اول فایل پرش می کند

       fprintf ( f , " %d" , k+1 );

       fclose ( f );
}
```

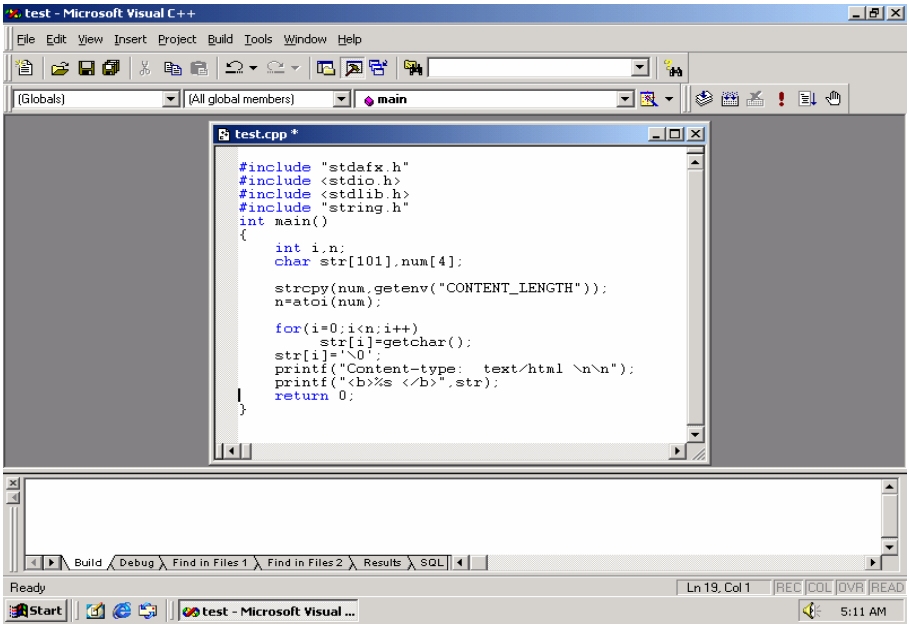
توجه: *خروجی برنامه‌های دروازه‌ای روی مانیتور سرویس‌دهنده، نمایش داده نمی‌شود، بلکه از طریق سرویس‌دهندهٔ HTTP به مرورگر ارسال می‌شوند.*

برنامهٔ Chat با Cgi

از Chat برای صحبت کردن کاربران روی اینترنت استفاده می‌شود. ایدهٔ اصلی این برنامه آن است که همانند مثال قبل یک ناحیهٔ مشترک داشته باشیم، که کاربران بتوانند اطلاعات به انتهای آن اضافه کنند و آن را تغییر دهند، برای این منظور می‌توان از یک فایل استفاده کرد. برای ایجاد یک برنامهٔ chat کافی است ابتدا از طریق یک برنامهٔ HTML اطلاعات کاربران دیگر را که قبلاً وارد نموده‌اند نمایش داد و فرمی جهت گرفتن یک خط اطلاعات از کاربر ایجاد کنیم. مانند (شکل ٤- ٥)

بخش HTML

```
 <HTML>
 <BODY>

     <IMG SRC ="http:\\www.sample.com\chat1.cgi">

     <Form    method ="get"      Action="http:\\www.sample.com\Cgi-
bin\Chat2.cgi">

        <Input  Type ="Text "    NAME="T1">

        <Input   Type ="Submit "  NAME="S1"   Value="send">

     </ Form>

</BODY>
</ HTML>
```



شکل ٤-٥ خروجی برنامه chat

حال برای نمایش اطلاعات موجود در فرم بالا کافی است، از طریق SSI درخواستی به برنامهٔ دروازه‌ای با نام chat1.cgi ارسال نموده و این اطلاعات را از فایل بخوانیم (مثلاً ۱۰ خط آخر فایل یا ۱۰۰ کاراکتر آخر فایل). همچنین برنامه دارای یک form برای گرفتن اطلاعات کاربر و کلید submit می‌باشد، که با زدن کلید Send اطلاعات به برنامه دروازه‌ای دومی با نام chat2.cgi ارسال شده و این برنامه اطلاعات را به انتهای فایل اضافه می‌کند.

متن این دو برنامه به‌صورت زیر است:

```
//CHAT 1
#include <stdlio.h>
#include <Stdio.h>
void main ( )
{
char   c,str[100];
FILE *f;
f=fopen (“c:\test\chat.txt”, “r”);
fseek (f,filesize(f)-100);
fread(f,100 ,Str);
print f (“Content-Type:   Text/Html   \n\n”);
print f (“%S”,Str);
fclose (f);
}


//CHAT 2
#include<stdlio.h>
#include<stdlib.h>
void main ( )
```

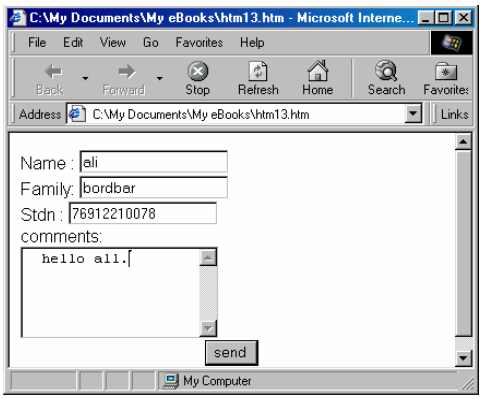
```
{
FILE  *f;
char str [100];
strcpy (str,getenv ("QUERY_STRING"));
f=fopen("c:\test\chat.txt","a");
fprintf(f,"%s<br>",str);
fclose(f);
}
```

برنامهٔ ثبت اطلاعات دانشجو

برنامهٔ زیر، اطلاعات یک دانشجو شامل نام، نام خانوادگی، شماره و توضیحات را از کاربر گرفته و این اطلاعات را در یک پایگاه داده روی سرویس‌دهنده ذخیره می‌کند. نکته‌ای که در این برنامه وجود دارد، این می‌باشد که قرار است هر دو متد را پشتیبانی نماید. برنامه دارای دو بخش HTML و دروازه‌ای می‌باشد، که در زیر مشخص شده است.

بخش HTML

```
  <HTML>
  <BODY>

  < Form  method = " post "  Action
="http://wwwsample.com/cgi_bin/student.cgi">
   Name : <Input  Name = "N">  <BR>
   Family : <Input  Name = "F ">  <BR>
  Stdn : <Input Name = " STU ">  <BR>

  Comment:<br>
   <textarea    name="t"  cols="20"  rows="5">
   </textarea>
  <Center>
  < Input Type = " Submit "   Value = " send" >
  </Center>
  </Form>
  </ Body>
  </Html >
```



شکل ۵-۵ فرم ثبت اطلاعات دانشجویان

بخش برنامهٔ دروازه‌ای

باید توجه کرد، که از آنجا که قرار است برنامهٔ هر دو متد را پشتیبانی نماید، ابتدا باید نام متد ارسالی که در متغیر محیطی REQUEST_METHOD می‌باشد، از سیستم عامل گرفته شده و سپس از طریق یک دستور IF_ELSE برای هر دو حالت get و post اطلاعات را از کاربر هم از داخل متغیر محیطی QYERY_STRING در حالت get و هم از ورودی استاندارد در حالت post بگیریم.

اطلاعات نهایی که در متغیر Str قرار می‌گیرد، کافی است یک مرحله pars روی آن انجام شده و اطلاعات بیرون کشیده شود و در پایگاه داده ثبت گردد.

```
#include <stdio.h >
#include <stdlib.h>
void main(  )
{
int  n,i;
Char   str[100] , Method[10] , Name=[15] , Family[20] , Number[11];
 FILE *f ;
Strcpy ( method , getenv ( "REQUEST_METHOD"));

If !strcmp ( method , " post ")
{
strcpy (num , getenv ("CONTENT_LENGTH"));
n= atoi (num);
for (i=0;i<n;i++)
     str[i]=getchar ( );
str[i+1]= "\0";
}
else
   Strcpy ( Str , getenv ( "QUERY-STRING" ));

Parser ( Str , &Name , &Family , &Number ");
f= fopen (" Student.dat" ، "a" );
fprintf (f ، "%s%s%s " ، Name ، Family ،Number" );
fclose (f );
}
```

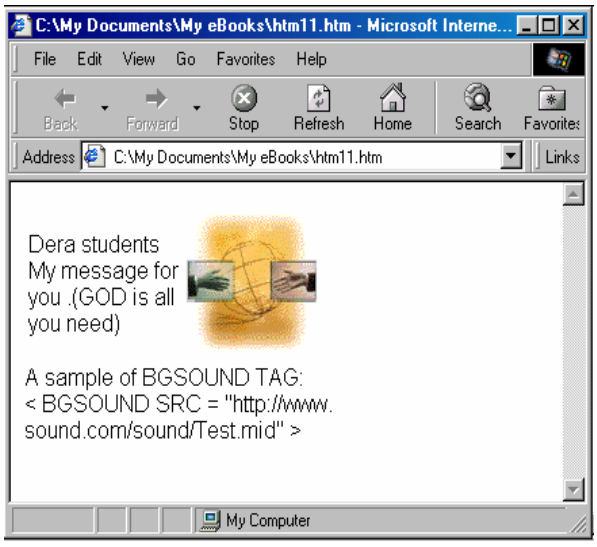
فرض می‌شود تابع Parser، اطلاعات داخل Str را پویش کرده و نام و نام خانوادگی و شماره را بیرون کشیده و در متغیرهای بالاتر قرار می‌دهد.

۱۰-۵ تمرین‌ها

۱. برنامه‌ای با cgi بنویسید، که جهت دیدن یک صفحه با آدرس www.sample.com از کاربر، رمز عبور سؤال کند و اگر رمز عبور عدد ۲۳۲ بود، اجازهٔ دیدن صفحهٔ Welcome page را بدهد، در غیر این صورت پیغام خطا صادر شود.

۲. برنامه‌ای بنویسید که اگر کاربر از سایتی با محدودهٔ IP، 194.224.1.0 تا 194.224.1.31 به آدرس www.sample.com وصل شود، امکان دیدن صفحهٔ اول را داشته باشد، در غیر این صورت امکان-پذیر نباشد.

۳. برنامه‌ای با cgi بنویسید، که در صفحهٔ اول نام کاربر را بپرسد و در سایر صفحات link به این صفحه نام کاربر بالای صفحه به عنوان title نمایش داده شود.
٤. برنامه‌ای با cgi بنویسید، که هر بار کاربر آدرس www.sample.com را زد هر بار از بین سه تا صفحهٔ خاص به‌طور تصادفی یکی را ببینید.
٥. یک Search engin از طریق cgi بنویسید.
٦. برنامهٔ chat را از طریق cgi کامل کرده، طوری که از کاربران Nickname (نام مستعار) سؤال کند.
۷. برنامه‌ای بنویسید، که از طریق یک form اطلاعات کاربر شامل نام و نام خانوادگی و آدرس و پست الکترونیکی را گرفته و با زدن کلید RUN، این اطلاعات به سرویس‌دهنده ارسال شده و از طریق یک برنامهٔ CGI روی سرویس‌دهنده اطلاعات دریافت و بدون Pars کردن دوباره به مرورگر ارسال شده و در وسط صفحه به‌صورت Bold، نمایش داده شود.
برنامه را طوری بنویسید که با هر دو متد post و get کار کند.
۸. برنامه‌ای با روش CGI بنویسید، که از طریق یک Form اطلاعات شما (name,family,number,sex) را گرفته و بدون تغییری در این اطلاعات اگر نام مرورگر NETSCAPE 3.0 باشد این اطلاعات را برای کاربر برگرداند. در غیر این صورت، پیغام Your browser is't match را دریافت کند. توجه کنید که متد ارسال، POST می‌باشد.
۹. برنامه‌ای از روش cgi بنویسید، که در صورتی که مرورگر کاربر Netscape 3.0 باشد، اجازهٔ دیدن Welcome page به‌صورت زیر را بدهد، در غیر این صورت پیغام خطا داده شود.



فصل ششم

برنامه‌نویسی با روش Active Server Page

۶-۱ مقدمه

این فصل رسماً شما را با ASP آشنا می‌کند و شما فرا می‌گیرید، که ASP چیست، چه کاربردهایی دارد و چگونه عمل می‌کند. این فصل یک تعبیر کلی از چگونگی به‌کارگیری اسکریپت‌های ASP در صفحات HTML و اطلاعاتی از اشیاء و اجزای ASP را پیش روی شما می‌گذارد.

۶-۲ ASP چیست؟

یک ASP، فایل استاندارد HTML ای است، که با یک سری ترکیبات الحاقی توسعه یافته است. همانند فایل استاندارد HTML، یک ASP می‌تواند شامل برچسب‌های HTML ای که توسط مرورگر وب تفسیر و نمایش داد می‌شود، باشد. هرچیزی مثل اپلت‌های جاوا، متن چشمک‌زن، اسکریپت‌ها و کنترل‌های اکتیوایکس روی سرویس‌گیرنده که بتوانید در یک فایل HTML جای دهید، در یک ASP نیز قابل جای دادن است. البته ASP پنج ویژگی منحصربه‌فرد نیز دارد، که به‌صورت زیر است:

- **یک ASP می‌تواند شامل اسکریپت‌های سمت سرویس‌دهنده باشد.**

در این بخش کتاب به شما یاد داده می‌شود که چگونه اسکریپت‌های ASP توسط جاوااسکریپت و VBScript ایجاد می‌شود. با به‌کارگیری اسکریپت‌های سمت سرویس‌دهنده، شما می‌توانید صفحات وبی با اجزای پویا ایجاد کنید. به عنوان یک نمونهٔ بسیار ساده، می‌توانید صفحهٔ وبی را ایجاد کنید که هر روز پیغام جدید یا تاریخ آن روز را نمایش دهد.

- **ASP شماری از اشیاء تعبیه شده را فراهم کرده است.**

با استفاده از اشیاء تعبیه شده قابل دسترس در یک ASP، می‌توانید اسکریپت‌های خود را بسیار قوی‌تر کنید. اشیاء به شما اجازهٔ دریافت و ارسال اطلاعات به مرورگر یا از آن را می‌دهد. برای مثال با استفاده از شیء Request، می‌توانید اطلاعاتی را که یک کاربر با فرم HTML ارسال کرده، دریافت و به آن پاسخ دهید.

- **یک ASP می‌تواند توسط اجزای الحاقی توسعه یابد.**

برنامه‌های ASP امکان دسترسی به اجزای Activex، روی سرویس‌دهنده را دارا می‌باشند. این اجزای به شما اجازهٔ کارهایی مثل تصمیم‌گیری در مورد قابلیت‌های مرورگرهای مختلف یا گنجاندن یک شمارندهٔ صفحه در سایت خود را می‌دهد.

شما نباید خود را فقط به اجزای استاندارد اکتیوایکس محدود کنید، هر چند که این اجزا بسیار مفید می‌باشند. شما می‌توانید اجزای الحاقی اکتیوایکس را برای خود ایجاد کنید، این بدان معنی است که هیچ محدودیتی در چگونگی توسعهٔ ASP ها برای شما وجود ندارد.

- **یک ASP می‌تواند اطلاعاتش را با پایگاه داده‌ای مثل سرویس‌دهندهٔ SQL مایکروسافت رد و بدل کند.**

با استفاده از مجموعهٔ خاصی از اشیاء که اشیای داده‌ای Activex (ADO)[1] نامیده می‌شود، شما می‌توانید از دستورات SQL جهت دسترسی به هر نوع پایگاه داده در ASP های خود استفاده کنید.

بحث ASP در این کتاب با این پیش‌فرض است، که شما از سکوهای مایکروسافت و سرویس‌دهندهٔ وب (IIS) بهره خواهید برد. هر چند شما می‌توانید به خوبی از ASP به همراه سرویس‌دهنده‌های وب دیگر نیز استفاده کنید. ASP ها می‌تواند توسط یک Personal Web Server که متعلق به‌ویندوز 9X می‌باشد و یا سرویس‌دهندهٔ وب ویندوز NT مورد استفاده قرار گیرند.

۶-۳ با ASP چه کاری می‌توان انجام داد؟

شما محدودیتی در اینکه چه کاری با ASP می‌توانید انجام دهید، ندارید.

تقریباً هر سایت وبی که امروزه در اینترنت وجود دارد با Asp ها قابل ایجاداست.

آنچه با ASP می‌توانید بسازید، عبارتند از اینکه :

- اطلاعات را از فرم‌ها دریافت و در یک پایگاه داده ذخیره کنید.
- یک صفحه وب شخصی بسازید، که شامل مشخصات متفاوت کاربران مختلف باشد.
- یک شمارندهٔ صفحه وب ایجاد کنید.
- با توجه به مشخصات مرورگرهای مختلف صفحات وب مختلفی را نمایش دهید.
- صفحات مختلف را به هم متصل کنید.
- فعالیت‌های کاربران سایت وب را دنبال و اطلاعات آنها را به‌دست آورده و این اطلاعات را در یک Logfile ذخیره کنید.

۶-۴ ASP چگونه کار می‌کند؟

بهترین راه برای فهمیدن اینکه ASP به چه صورت کار می‌کند، مقایسهٔ سرویس‌دهنده‌های وبی است، که ASP را حمایت کرده، با آنهایی که پشتیبانی نمی‌کنند. مایکروسافت، ASP را با چهارمین نسخهٔ IIS معرفی کرد. با معرفی ASP ،IIS یعنی سرویس‌دهنده‌ای با محتوای ثابت و ایستا به یک سرویس‌دهنده با محتوای پویا و تأثیرپذیر تبدیل شد. این کار به چه صورت انجام می‌شود؟ مهم‌ترین کار IIS، سرویس-دادن به صفحات HTML استاندارد بود. هنگامی که کسی نیاز به یک صفحه وب داشت، از سرویس-

1. Activex data objects

دهندهٔ IIS استفاده می‌کرد و آن فایل ثابت HTML را از دیسک یا حافظه به‌دست می‌آورد وبه مرورگر شخص می‌فرستد.

IIS با بقیهٔ سرویس‌دهنده‌های دیگر وب متفاوت است. مهم‌ترین هدف هر سرویس‌دهنده وب، سرویس دادن به فایل‌های HTML است. این مهم است که بدانیم سرویس دادن به فایل‌های HTML به چه صورت انجام می‌شود. همان‌طور که قبلاً هم در فصل اول گفته شد، مراحل دسترسی به فایل‌های HTML به‌صورت خلاصه به شکل زیر است.

۱. یک کاربر، آدرس اینترنتی فایل HTML ای را که می‌خواهد درون خط آدرس قرار می‌دهد و با فشار دادن کلید Enter، این درخواست را می‌فرستد.
(به عنوان مثال http://.www.aspsite.com/hello.htm)
۲. مرورگر این درخواست را برای یک سرویس‌دهندهٔ وب مثلاً IIS می‌فرستد.
۳. سرویس دهندهٔ وب درخواست را دریافت می‌کند و تشخیص می‌دهد، که یک فایل HTML درخواست شده است، به این دلیل که دارای پسوند htm یا html است .
٤. سرویس‌دهندهٔ وب آن فایل را از دیسک یا حافظه دریافت و آن را برای مرورگر ارسال می‌کند.
٥. فایل HTML توسط مرورگر شخصی ترجمه و تفسیر می‌شود و نتیجه در پنجرهٔ مرورگر نمایش داده می‌شود.

ASP همه چیز را تغییر می‌دهد. دیگر علاوه براینکه IIS می‌تواند برای سرویس HTML ثابت مورد استفاده قرار گیرد، با ASP ، IIS قادر به انجام خدمات HTML پویا و تأثیرپذیر نیز خواهد شد.

حال ببینیم جهت دسترسی به یک برنامهٔ ASP روی سرویس‌دهندهٔ وب چه عملیاتی رخ می‌دهد:

۱. یک کاربر، آدرس اینترنتی فایل ASP را درون خط آدرس قرار می‌دهد و با زدن کلید Enter درخواست ASP را می‌فرستد. (به عنوان مثال http://www.aspsite.com/hello.asp)
۲. مرورگر این درخواست را برای یک سرویس‌دهندهٔ وب، مثلاً IIS می‌فرستد.
۳. سرویس‌دهندهٔ وب درخواست را دریافت می‌کند، ولی از آنجا که درخواست، پسوند asp دارد، تشخیص می‌دهد که یک فایل ASP در خواست شده است.
٤. سرویس‌دهندهٔ وب، فایل ASP را از دیسک یا حافظه باز می‌کند.
٥. سرویس‌دهندهٔ وب، فایل را به برنامهٔ مخصوصی به نام ASP.dll می‌فرستد.
٦. فایل ASP از بالا تا پایین پردازش می‌شود و تمام script های سمت سرویس‌دهنده اجرا می‌شوند، نتیجهٔ این پردازش یک فایل HTML استاندارد است.
۷. فایل HTML برای مرورگر فرستاده می‌شود.
۸. فایل HTML توسط یک مرورگر شخصی ترجمه و تفسیر می‌شود و نتیجه در پنجرهٔ مرورگر نمایش داده می‌شود.

یک ASP با یک فایل HTML عادی خیلی تفاوت دارد. یک فایل HTML عادی، بدون پردازش به مرورگر فرستاده می‌شود، ولی در ASP، تمام script های سمت سرویس‌دهنده، باید اجرا شوند تا یک صفحهٔ HTML استاندارد را به وجودآورد.

خروجیASP در اغلب موارد به صورت کامل مانند یک HTML نرمال است، تنها فرقی که دارد این است، که پسوند آن به جای htm، asp است. وقتی درخواستی برای به دست آوردن یک ASP به وجود می‌آید، مرورگر یک صفحهٔ HTML عادی را دریافت می‌کند، همین امر باعث می‌شود که یک ASP با تمام مرورگرها سازگار شود.

۵-۶ به‌کار بردن اسکریپت‌ها درون ASP

یک ASP، ابتدایی‌ترین محیط اسکریپت‌نویسی است. می‌توان درون ASP، از هر دو نوع اسکریپت (JscriptVBScript) استفاده کرد. از بقیهٔ زبان‌های اسکریپت نیز می‌توان به همان خوبی استفاده کرد. راحت‌ترین راه برای استفاده از اسکریپت درون ASP استفاده از تگ <%-%> می‌باشد. هر متنی که درون این تگ قرار بگیرد، به عنوان اسکریپت سمت سرویس‌دهنده محسوب می‌شود.

به عنوان مثال:

```
<HTML>
< HEAD> <TITLE> ASP Script </TITLE> </HEAD>
<BODY>
      This is a
      <%  for   i=1  to  10%>

           very

       <%NEXT%>
        very long sentence.
</BODY>
<HTML/>
```

هنگامی که این ASP توسط مرورگر دریافت می‌شود، جملهٔ زیر نمایش داده خواهد شد:

```
This is a very very very very very very very very very very very  long
sentence.
```

این اسکریپت کلمهٔ very را ۱۱ بار تکرار می‌کند و این کار را با استفاده از حلقهٔ FOR...NEXT در VBscript انجام می‌دهد. به هرحال، سه روش برای استفاده از اسکریپت ها درون ASP وجود دارد.

می‌توان با استفاده از Internet Service Manager یک زبان اسکریپت را به عنوان پیش‌فرض برای تمام ASP ها در نظر گرفت. بعد از اینکه یک زبان اسکریپت را به عنوان پیش‌فرض انتخاب کرده‌اید، می‌توانید آن زبان را درون Asp به همان راحتی وبا استفاده از تگ <%- %> استفاده کنید. به عنوان مثال اگر می‌خواهید از جاوا اسکریپت استفاده کنید، باید این زبان را به عنوان پیش‌فرض انتخاب کنید.

همچنین می‌توانیم یک اسکریپت خاص را فقط در یک صفحه به‌کار ببریم. برای انجام این کار زبان موردنظر را در اولین خط از فایل ASP تعریف می‌کنیم، مانند زیر:

```
<%@LANGUAGE=JScript%>
<HTML>
<HEAD> <TITLE> ASP Script </TITLE> </HEAD>
<BODY>
    This is a
    <%for ( i=1 ; i<11 ; i++) { %>
```

```
        very

       <% } %>
        very long sentence.
</BODY>
</HTML>
```

دستوری که در خط اول قرار داده شده است، نشان می‌دهد که زبان پیش‌فرض در کل برنامه، جاوا اسکریپت است. هنگامی که از این دستوراستفاده می‌شود، باید توجه داشت که این دستور باید قبل از هر دستوری در صفحهٔ ASP قرار بگیرد.

سومین روش برای استفاده از اسکریپت‌ها درون ASP، استفاده از برچسب <Script> با آرگمان Runat = “server” در HTML است.

به مثال زیر توجه کنید:

```
<HTML>
<HEAD><TITLE>ASP Script</TITLE></HEAD>
<BODY>
<SCRIPT  LANGUAGE= "JScript"  RUNAT= "Server" >
function sayhello()
{
response.write(" Hello!”)
}
</SCRIPT>
<%

sayhello ()

%>
</BODY>
</HTML>
```

در اینجا از یک تابع جاوا اسکریپت استفاده شده است. LANGUAGE نشان‌دهندهٔ نوع زبان اسکریپتی است، که استفاده می‌شود. RUNAT نشان می‌دهد، که این اسکریپت باید در سرویس دهنده اجرا شود. در مثال بالا اگر عبارت "RUNAT=" Server فراموش شود، دستورات در سرویس‌گیرنده اجرا می‌شوند. در این حالت سرویس‌دهنده دستورات را نادیده می‌گیرد و سرویس‌گیرنده، تلاش در اجرای دستورات می‌نماید.

برچسب <SCRIPT> یک مزیت اصلی نسبت به تگ <٪ و ٪> برای تعریف اسکریپت دارد، زیرا شما می‌توانید از چندین زبان مختلف درون یک ASP استفاده کنید، مثال زیر را در نظر بگیرید:

```
<%@LANGUAGE= "Vbscript"%>
<HTML>
<HEAD><TITLE>ASP Script</TITLE></HEAD>
<BODY>
<SCRIPT  LANGUAGE=" JScript" RUNAT= "Server">
function sayhello()
{

 response.write(“Hello!”)

}
```

```
</SCRIPT>
<%
for i=1 to  10
Sayhello ()
NEXT
%>
</BODY>
</HTML>
```

این اسکریپت کلمه !Hello را ده بار در یک سطر، تکرار می‌کند، اما باید توجه داشت که این کار را به چه صورت انجام می‌دهد. اسکریپتی که درون کاراکترهای تعریف <% و %> قرار دارد، اسکریپت ویژوال بیسیک می‌باشد، که یک تابع به زبان JavaScript را که درون برچسب <SCRIPT> تعریف شده است، صدا می‌زند. هنگامی که از یک اسکریپت به عنوان زبان اصلی استفاده می‌کنید و همچنین نیاز به زبان اسکریپت از نوع دیگری دارید، از این روش استفاده می‌شود.

در اینجا به‌طور خلاصه سه روش استفاده ازاسکریپت‌ها درونASP بیان شده است:

- یک زبان اسکریپت با استفاده از Internet Service Manager به عنوان زبان اصلی تعریف می‌کنیم.
- یک زبان اسکریپت را برای یک صفحه با استفاده از دستورASP به صورت زیر تعریف می‌کنیم:

```
%>"  LANGUAGE=" Scripting language<%@
```

- با استفاده از <SCRIPT> می‌توانیم چندین زبان اسکریپت را در یک ASP استفاده نمائیم.

مثال :

با استفاده از کاراکتر(=) در تگ <٪ و ٪> می‌توانید مقدار متغیرها، توابع و متدها را چاپ نمایید. در مثال زیر با استفاده از تابع TIME درVBScript می‌توانیم زمان را در صفحهٔ نمایش چاپ کنیم.

```
<HTML>
<HEAD><TITLE>ASP Example</TITLE></HEAD>
<BODY>

At the tone , the time will be : <%=TIME%>

</BODY>
</HTML>
```

یک راه دیگر برای انجام این کار وجود دارد، مثال زیر را در نظر بگیرید:

```
<HTML>
<HEAD><TITLE>ASP Example </TITLE></HEAD>
<BODY>

At the tone , the time will be: <%Response.write(TIME)%>

</BODY>
</HTML>
```

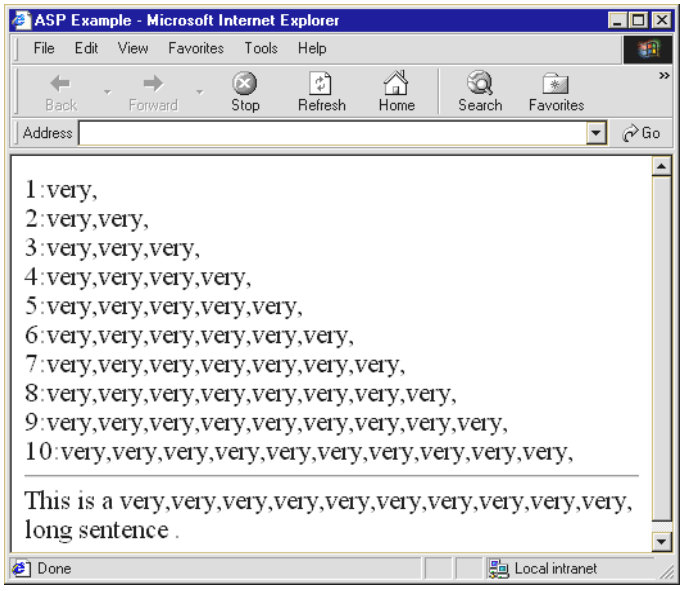
در این مثال مقدار تابع TIME در VBScript به‌وسیله یک شیء از ASP نمایش داده می‌شود. متد write از شیء Response مقدار متغیر خروجی را روی مرورگر نمایش می‌دهد. چه وقت باید از متد ( ) write به جای <٪ = ٪> به عنوان خروجی استفاده کنیم؟ این موضوع زیاد مهم نیست.ASP به صورت داخلی به

متد صدا شدهٔ Response.write() پاسخ می‌دهد. این دو متد برای نمایش دادن مقدار خروجی، به صورت داخلی بوده و قابل تبدیل به یکدیگر می‌باشند.

در مواقعی که یکی از روش‌های خروجی مناسب‌تر از دیگری است ؛ به عنوان مثال وقتی شما یک عبارت خروجی را می‌خواهید داخل یک اسکریپت تعریف کنید، راحت‌تر است که از متد Response.write() استفاده کنید، به عبارت دیگر وقتی می‌خواهید مقدار خروجی یک عبارت را به وسیلهٔ یک قسمت از کد HTML نشان دهید، راحت‌تر است از <٪و٪> استفاده کنید. در مثال زیر هر دو روش نشان داده شده است:

```
<HTML>
<HEAD><TITLE>ASP Example </TITLE></HEAD>
<BODY>
<%
    for i=1 to  10
    myvar = myvar &"very"
    Response.write(i&" : "& myvar& "<BR>")
    NEXT
%>
<HR>
This is a <%=myvar %> long sentence.
</BODY>
</HTML>
```

در این مثال متد Response.write( )درون یک حلقه برای نشان دادن مقدار متغیر myvar و برای کم کردن اندازهٔ برنامه استفاده می‌شود و از علامت <٪و٪> به‌صورت عادی در یک کد ساده HTML استفاده می‌شود. (شکل ۱- ٦)



شکل ۱-٦ کار با رشته ها در ASP

۶-۶ اشیاء ASP

ASP، شامل تعدادی از اشیاء استاندارد داخلی است. این اشیاء باعث افزایش قدرت اسکریپت های ما می‌شود. با استفاده از این اشیاء می‌توانید تعداد زیادی از در خواست‌های مرورگر را دریافت کنید و چگونگی پاسخ سرویس‌دهنده به این در خواست‌ها را کنترل نمایید. این اشیای تعبیه شده، امکان کنترل Session ها و برنامه‌های سرویس‌دهندهٔ وب را به‌وجود می‌آورند. با توجه به شیء تعبیه شدهٔ Response در مثال قبل از این شیء می‌توانید برای فرستادن خروجی‌هایتان به مرورگر استفاده کنید. این شیء دارای شماری از خاصیت‌ها، متدها ومجموعه‌های مهم می‌باشد.

در زیر اشیای Asp مورد بررسی قرار گرفته است.

شیء Response: این شیء برای فرستادن اطلاعات از سرویس‌دهنده به مرورگر استفاده می‌شود.

شیء Request: این شیء برای دستیابی به اطلاعات فرستاده شده از مرورگر به سرویس‌دهندهٔ وب قابل استفاده است. می‌توان ازاین شیء برای کار روی اطلاعاتی که کاربر درون یک فرم HTML وارد می‌کند، استفاده کرد.

شیء Application : این شیء برای کار روی متغیرهایی به‌کار می‌رود، که می‌توانند بین کاربران یک سایت به اشتراک گذاشته شوند، به عنوان مثال می‌توان از شیء Application برای فرستادن اطلاعات بین کاربران سایت وب خود استفاده کرد.

شیء Session: این شیء برای کار روی متغیرهای مربوط به جلسهٔ یک کاربر خاص، استفاده می‌شود. از این شیء برای ذخیرهٔ اطلاعات کاربری که دارد از سایت وب شما دیدن می‌کند، می‌توان استفاده نمود.

شیء Server: این شیء اجازهٔ استفاده از توابع گوناگون و مفیدی را از سرویس‌دهنده می‌دهد، به عنوان مثال با استفاده از آن می‌توان مدت زمان اجرای برنامه‌های ASP را تغییر داد. همچنین می‌توان با استفاده از این شیء، اشیای دیگری را به‌وجود آورد.

شیء Context: این شیء برای کنترل نقل و انتقالات ASP استفاده می‌شود. این نقل و انتقالات توسط نرم‌افزار Microsoft Transaction Server(MTS)مدیریت می‌شود.

باید توجه کرد که برای استفاده از اشیای بالا دیگر نیاز به ساخت شیء نداریم، مگر اینکه بخواهیم درون یک اسکریپت از شیء جدیدی استفاده کنیم. باید اضافه کرد که متدها، خواص و مجموعه‌های هر شیء به‌طور خودکار در دسترس قرار می‌گیرند.

اجزای ASP

اجزای ASP مانند اشیای تعبیه شده در آن برای قدرتمند کردن اسکریپت‌ها استفاده می‌شوند. اجزای با اشیای تعبیه شده تفاوت دارند، زیرا آنها برای کارهای خاص تری استفاده می‌شدند. در این کتاب این اجزاء شرح داده نشده‌اند. در زیر لیستی از اجزای ASP بیان شده است:

جزء Ad Rotator (MSWC.AdRotator): این جزء برای نمایش اعلان آگهی در صفحات سایت وب استفاده می‌شود. می‌توان از این جزء برای تعیین چگونگی تکرار اعلان آگهی که باید نمایش داده شوند استفاده کرد.

جزء Browser Capabilities (MSWC.BrowserType): این جزء برای نمایش محتوای HTML های مختلف مطابق با قابلیت‌های مرورگرها استفاده می‌شود.

جزء Content Linking (MSWC.NextLink): با استفاده از این جزء می‌توانید تعدادی از صفحات HTML را به هم متصل کنید، که باعث می‌شود هدایت کاربر آسان‌تر شود، به عنوان مثال با استفاده از این جزء می‌توانید صفحات یک کتاب را نمایش دهید.

جزء Counters (MSWC.Counters): این جزء می‌تواند کاربران مختلف را در سایت وب ردیابی کند. با استفاده از این جزء می‌توانید یک شمارنده برای صفحهٔ وب ایجاد کنید.

جزء Counters Rotator (MSWC.ContersRotator): این جزء شما را قادر به حرکت درون محتوای یکHTML می‌کند. به عنوان مثال می‌توان به‌صورت تصادفی آگهی‌های مختلفی را در صفحهٔ خانگی سایت وب نمایش داد.

جزء Page Counter (MSWC.PageCounter): دقیقاً مانند جزء Counters می‌باشد. این شمارنده می‌تواند برای پیگیری تعداد ملاقات کننده‌های یک صفحه وب استفاده شود. با استفاده از این جزء می‌توان یک شمارنده، برای یک صفحهٔ خاص وب قرار دارد.

جزء Permission Checker (MSWC.PermissionChecker): این جزء برای نمایش اتصال‌های مختلف به صفحات وب به شرط اینکه کاربر اجازهٔ دیدن آنها را داشته باشد، استفاده می‌شود. با استفاده از این جزء می‌توان صفحات وبی را به وجود آورد، که فقط مدیر سایت یا افراد خاصی می‌توانند آنها را ببینند.

ActiveX Data object (ADO) (ADODB.Connection): از این جزء جهت دسترسی به پایگاه دادهٔ مختلف از طریق دستورات SQL استفاده می‌شود.

۱-۶-۶ شیء Response

یکی از مهم‌ترین اشیای ASP شیء Response می‌باشد، همان‌طور که از معنی آن مشخص می‌باشد، جهت ارسال اطلاعات از برنامهٔ ASP روی سرویس دهنده به مرورگر سرویس‌گیرنده از این شیء می‌توان استفاده کرد. این شیء همانند سایر اشیای ASP دارای تعدادی خصوصیت یا Property و Method و Collection می‌باشد که مهم‌ترین آنها عبارتند از:

Collection:	
Cookies	جهت دسترسی به فایل Cooki روی سرویس‌دهنده ازاین مجموعه استفاده می‌شود.
Properties:	
Buffer	جهت فعال کردن بافر در ASP به‌کار می‌رود.
Cache Control	طریقهٔ Cache شدن صفحات در مرورگر را تعیین می‌کند.

Content Type	جهت مشخص کردن نوع خروجی برای مرورگر به‌کار می‌رود.
Expires	جهت تعیین کردن اینکه یک صفحه چه مدت در داخل cache نگهداری شود، به‌کار می‌رود.
IsClientConnect	نشان می‌دهد که آیا هنوز ارتباط بین مرورگر و سرویس‌دهنده برقرار است، یا خیر.
Expires Absolute	مشخص می‌کند، که نسخهٔ cache شدهٔ صفحه باید در تاریخی که ذکر شده، دوباره از سرویس‌دهنده دریافت شود.
Status	این خاصیت برای تعریف کد وضعیتی است، که پاسخ HTTP باز می‌گرداند.
Methods:	
Appen To Log	جهت نوشتن اطلاعات در LogFile از آن استفاده می‌شود.
Binary Write	جهت ارسال اطلاعات به‌صورت باینری به‌کار می‌رود.
Clear	جهت خالی کردن بافر به‌کار می‌رود.
End	جهت خاتمه دادن پردازش ASP به‌کار می‌رود.
Flush	جهت خالی کردن بافر و ارسال اطلاعات بافر به سرویس‌گیرنده به‌کار می‌رود.
Redirect	به‌طور خودکار کاربر را به URL معینی هدایت می‌کند.
Write	جهت ارسال اطلاعات به مرورگر به‌کار می‌رود.

حال به شرح هر یک با کاربردشان خواهیم پرداخت:

cookie

تنها فایلی روی سرویس‌گیرنده است، که سرویس‌دهنده اجازهٔ ایجاد و خواندن یا نوشتن اطلاعات روی آن را دارد. مرورگرهایی که از cookie پشتیبانی می‌کنند، این فایل‌های مخصوص را دارند. این فایل‌ها Cookie Files نامیده می‌شوند و برای ذخیره داده روی سرویس‌گیرنده استفاده می‌شوند. یک سرویس-دهندهٔ وب می‌تواند مقداری از اطلاعات را درون فایل‌های cookie قرار دهد. بعضی از کاربران وب عکس‌العمل شدیدی نسبت cookie ها نشان می‌دهند. بعضی از کاربران تصور می‌کنند، cookie حریم خلوت آنها را تهدید می‌کند، یا به حریم خصوصی آنها تجاوز می‌کند.

cookie های خاصی موقتی و بقیه ماندگار هستند. به عنوان مثال cookieهایی که در ASP برای دنبال کردن session های کاربر به‌کار می‌روند، پس از اینکه کاربری سایت وب را ترک می‌کند، تمام می‌شوند. بقیهٔ cookie ها درون فایل cookie می‌مانند، تا هنگامی که کاربر برگرددو توسط سرویس‌دهنده دوباره خوانده شوند. Cookie هایی که در فایل cookie می‌مانند، آنهایی هستند، که بیشتر به‌کار می‌روند. بیم آن می‌رود، که cookie ها بتوانند برای اهداف سودجویانهٔ شخصی مورد استفاده قرارگیرند. البته کاربر می‌تواند از نوشته شدن cookie درون شاخهٔ فایل‌های cookie خود با فقط خواندنی کردن آن جلوگیری کند، ولی واضح است، که خواندن فایل‌های شما انکارناپذیر خواهد بود.

cookie ها به چه صورت کار می‌کنند؟

cookieها توسط سرآیندهای HTTP بین سرویس‌دهنده و مرورگر، جابه‌جا می‌شوند. ابتدا سرویس‌دهنده در پاسخ خود به وسیلهٔ سرآیند Set cookie یک cookie به‌وجود می‌آورد و بعد توسط درخواست مرورگر این cookie درون سرآیند cookie بازگردانده می‌شود.

فرض کنید می‌خواهید یک cookie به‌نام UserName به وجود آورید، که شامل نام ملاقات کنندگان سایت شما باشد. برای به وجود آوردن این cookie ، سرویس‌دهنده باید سرآیند زیر را بفرستد:

```
 set_cookie : UserName =BILL+Gates ; Path=/ ;   domain =aspsite.com;
Expires= Tuesday , 01-Jan-2003   00:00:01  GMT
```

این سرآیند به مرورگر دستور می‌دهد، که یک cookie با نام UserName و با مقدار Bill Gates درون فایل cookie اضافه کند. همچنین سرآیند، مرورگر را آگاه می‌کند، که این cookie باید بدون توجه به مسیر به‌کار رفته در درخواست، به سرویس‌دهنده بازگردد. اگر مقدار مسیر به مقادیر دیگری همچون / Private تنظیم شده بود، cookie باید تنها در این مسیر برگردانده می‌شد.

صفت domain برای این است که نشان دهد، مرورگر cookie ها را به کجا می‌تواند بفرستد. در این مثال cookie می‌تواند فقط به سایت www.aspsite.com فرستاده شود و در این صورت cookie به هیچ سایت دیگری فرستاده نمی‌شود.

در آخر صفت Expries تعیین می‌کند، که cookie باید تا چه زمانی باقی بماند. سرآیند این مثال به مرورگر می‌گوید که cookie را تا نیمهٔ اول ژانویهٔ 2003 ذخیره کند، یک cookie ممکن است زودتر از این از بین برود، به این صورت که اگر یک cookie بیش از حد بزرگ بشود، مرورگر به‌طور خودکار آن را حذف می‌کند.

اولین بار، که مرورگر یک cookie را به وجود می‌آورد، آن را درون تمام درخواست‌هایی که به وجود می‌آورد قرار می‌دهد و به سایت‌های وبی که مورد درخواست واقع شده‌اند می‌فرستد. البته مرورگر cookieها را به سایت وبی که نام‌های Domain متفاوت دارند، نمی‌فرستد. مرورگر فرستادن cookie را ادامه می‌دهد، تا آن cookie از بین برود. یک سرآیند cookie شبیه زیر است.

```
cookie : User NAME: Bill +Gates
```

به وجود آوردن و خواندن cookie ها باASP

به منظور به وجود آوردن یک cookie، با ASP، باید از مجموعهٔ cookie از شیء Response استفاده کنید. شما می‌توانید دو نوع از cookie ها را به وجود آورید، که اولی یک cookie با یک مقدار تکی و دومی یک cookie dictionary شامل نام و مقدار آن می‌باشد.

برای به وجود آوردن یک cookie، با مقدار تکی می‌توانید از اسکریپت زیر استفاده کنید.

```
<%
Response.cookies (" UserName ") =" Bill Gates”
Response.cookies (" UserName "). Expires =" jan 1,2003”
%>
```

این اسکریپت یک cookie با UserName با مقدار کاراکتری " Bill Gates " به وجود می‌آورد. این cookie توسط مرورگر تا ژانویهٔ 2003 یا تا وقتی که مرورگر آن را پاک نکرده باشد برگردانده می‌شود. اگر زمان انقضاء برای cookie در نظر گرفته نشود هنگامی که کاربر سایت وب را ترک کند cookie نیز ازبین می‌رود. چون این اسکریپت یک سرآیند تولید می‌کند، بنابراین باید قبل از هرگونه خروجی درون فایل ASP قرار بگیرد. درنهایت شما می‌توانید صفحه را بافر کنید. اسکریپت زیر مثال ساده‌ای ازچگونگی ایجاد یک Cookie می‌باشد.

باید توجه کرد که مجموعهٔ cookie دارای خواص دیگری نیز می‌باشد، به مثال‌های زیر توجه کنید.

```
<%
Response.Cookies ("UserName")= "Steve Jobs"
Response.Cookies ("UserName").Expires= "Jan 1,2003"
Response.Cookies ("UserName").path= "/examples"
Response.Cookies ("UserName").Domain = "aspsite.com"
Response.Cookies ("UserName").secure="True"
%>
```

این اسکریپت یک cookie با نام UserName به وجود می‌آورد، که دارای صفات زیر است.

صفت path برای تعریف مسیر دقیق‌تر جایی که مرورگر باید cookie را بفرستد، استفاده می‌شود. در این مثال، cookie فقط به مسیرهای مورد درخواستی که با examples / شروع می‌شوند، فرستاده می‌شوند. به عنوان نمونه، درخواست‌های زیر فرستاده می‌شوند، examples/chapater/cookie.asp / examples/hello.asp/ ولی برای "/hello.asp" فرستاده نمی‌شوند. صفت Domain تعیین می‌کند که یک cookie چه وقت باید فرستاده شود در مثال قبل cookie تنها با درخواست‌هایی که به حوزهٔ aspsite.com می‌رود، ارسال می‌شود. این بدان معنی است، که cookie به مقاصد www. aspsite یا crieket.aspsite .com فرستاده می‌شود. اگر این خاصیت تعریف نشود، از نام حوزهٔ سرویس دهنده وب استفاده می شود.

نهایتاً صفت Secure مشخص می‌کند، که Cookie باید به‌صورت رمز شده انتقال یابد.

بافر کردن خروجی

معمولاً وقتی یک ASP در سرویس‌دهنده اجرا می‌شود، صفحهٔ خروجی بعد از اجرای هر کدام از دستورات در سرویس‌دهنده به روی مرورگر ارسال می‌شود. به عنوان مثال ASP زیر را در نظر بگیرید.

```
<HTML>
<HEAD> <TITLE> Buffer Example </ TITLE> </ HEAD>
<Body>
For i =1To 500
      Response.Write(I & "<br>")
 NEXT
 %>
</ BODY>
</ HTML>
```

این اسکریپت اعداد 1 تا 500 را از بالا به پایین در مرورگر نمایش می‌دهد. صفحهٔ خروجی دقیقاً بعد از اجرای هر دستور به مرورگر فرستاده می شود و می‌توانید هر عددی که ظاهر می‌شود را در همان لحظه ببینید. در بعضی از مواقع شاید بخواهید خروجی ASP را بافر کنید. وقتی خروجی ASP را بافر

می‌کنید، تا وقتی پردازش صفحات در سرویس‌دهنده به‌صورت کامل پایان نیافته، هیچ اطلاعاتی به مرورگر فرستاده نمی‌شود.

مثال زیر همان اسکریپت بالا است، که براساس مطالب گفته شده تغییر یافته است.

```
<%Response.Buffer = Ture%>
<HTML>
<HEAD>
<BODY>
< %
FOR i =1 To 500
Response. Write (I &"<BR>")
NEXT
%>
</ BODY>
</ HTML>
```

بین این اسکریپت وقبلی تنها یک تفاوت وجود دارد، اینکه در خط اول به خاصیت Buffer از شیء Response مقدار True داده شده است. وقتی این صفحه در مرورگر نمایش داده می‌شود، کل محتوای صفحه به‌صورت یک جا به مرورگر فرستاده می‌شود و صفحه تا وقتی پردازش اسکریپت پایان یابد، بافر می‌شود.

هرگاه بخواهید بافر را فعال کنید، این کار را باید قبل از هر نوع خروجی اسکریپت یا HTML ، انجام دهد. در غیر این صورت، پیغام خطا دریافت می‌کنید.

با بافرکردن یک صفحه می توانید صفحات مختلفی را با توجه به شرایط مختلف به‌دست آورید. به عنوان مثال، در زیر می‌توان دو صفحه مختلف به صورت تصادفی ایجاد کرد.

```
<% response . Buffer = True%>
<HTML>
<HEAD><TITLE> First Page </ TITLE> </ HEAD>
<BODY>
This is the first page
</ BODY>
</ HTML>
<% Randomize
 If int(2*RND)=1 Then Response.End
 Else
 Response.Clear
 End If
<HTML>
<HEAD> <TITLE> Second page </ TITLE> </ HEAD>
<BODY>
This is the second page.
</ BODY>
</ HTML>
```

در این مثال، دو متد از شیء Response به‌کار می‌رود، که عبارتند از متد Endو متد Clear. متد End فوراً پردازش یک صفحهٔ ASP را متوقف و نتیجهٔ پردازش را ارسال می‌کند. باید توجه کرد که متد End را بدون توجه به اینکه از بافر استفاده کرده‌ایم، می‌توان به‌کار برد. در این مثال متد End تا وقتی صفحهٔ اول نمایش داده نشده، مانع نمایش صفحهٔ دوم می‌شود. متد Clear بافر صفحهٔ جاری را بدون استخراج

محتوای بافر خالی می‌کند. از متد Clear وقتی که خروجی ASP بافر شده باشد، استفاده می‌کنیم. در مثال بالا متد Clear، مانع از نمایش صفحهٔ اول تا زمانی که صفحهٔ دوم در حال نمایش است، می‌شود و سپس صفحهٔ اول را از بافر پاک می‌کند. عموماً یکی دیگر از متدهای شیء Response نیز هنگام بافر کردن یک ASP به‌کار می‌رود. متد Flash محتویات بافر صفحه را مستقیماً خارج می‌کند. همانند متد Clear، اگر شما مبادرت به استفاده از این متد با صفحه‌ای که بافر نیست بکنید، پیغام خطا دریافت می‌کنید و برخلاف متد End، پس از فراخوانده شدن متد Flash، پردازش صفحه ادامه می‌یابد.

همیشه لازم نیست برای خروجی ASP از بافر استفاده کنیم. مثال زیر را درنظر بگیرید: در این مثال برنامهٔ قبل بدون استفاده از بافر نوشته شده است.

```
<%
Randomize
IF INT(2*RND) =1 Then
%>
<HTML>
<HEAD> <TITLE> First page</TITLE></HEAD>
</BODY>
</HTML>
<%ELSE%>
<HTML>
<HEAD> <TITLE> Second page </ TITLE> </ HEAD>
<BODY>
This is the second page
</ Body>
</ HTML>
<%END IF %>
```

بررسی برقراری ارتباط بین مرورگر و سرویس‌دهنده

اگر به اسکریپت‌ها اجازهٔ اجرا برای دوره‌های زمان طولانی را بدهید، این امکان وجود دارد که سرویس-دهندهٔ وب شما دچار کمبود منابع شود. در واقع ممکن است یک اسکریپت درون یک ASP حتی بعد از اینکه شخص درخواست کنندهٔ ASP از آن بیرون رفت، به اجرای خود ادامه بدهد. در این صورت اسکریپتی، که به اجرای خود ادامه می‌دهد، برای هیچ‌کس فایده‌ای نخواهد داشت. خوشبختانه خاصیتی از شیء Response وجود دارد که می‌تواند به ما در این مورد کمک کند. خاصیت IsClientConnected می‌تواند بررسی کند، که آیا هنوز ارتباطی بین مرورگر و سرویس‌دهنده برقرار است یا نه. برای مثال اسکریپت مثال بعد اجرا را تا هنگامی که زمان اسکریپت منقضی شود و یا اینکه ارتباط مرورگر از سرویس‌دهنده قطع شود، ادامه می‌دهد.

```
<HTML>
<HEAD><TITLE> Test  page </ TITLE> </ HEAD>
<BODY>
<%
WHILE  1 =1
Response.Write ( "Hello . How Are you?" )
If not Response.IsClientConnected THEN Response.End
WEND
%>
</ BODY>
```

```
</ HTML>
```

نکتهٔ قابل ذکر این است که خاصیت IsClientConnected تنها وقتی عکس‌العمل نشان می‌دهد، که مرورگر تا زمان فراخوانی متد Response.Write آخر وصل باشد. اگر شما یک اسکریپت با زمان طولانی، که هیچ چیزی را به مرورگر برنگرداند داشته باشید، آن موقع خاصیت مزبور بی‌فایده خواهد بود.

کار با سرآیندها و متغیرهای محیطی

همان‌طور که قبلاً گفتیم، هم درخواست‌های مرورگر و هم پاسخ‌های سرویس‌دهنده حاوی سرآیندهایی است، که در فصل اول در بخش Cgi توضیح داده شدند. سرآیندها اطلاعاتی اضافی در رابطه با محتوی درخواست یا پاسخ فراهم می‌کنند. همچنین می‌توانند حاوی اطلاعاتی در رابطه با مرورگر یا سرویس-دهنده باشند.

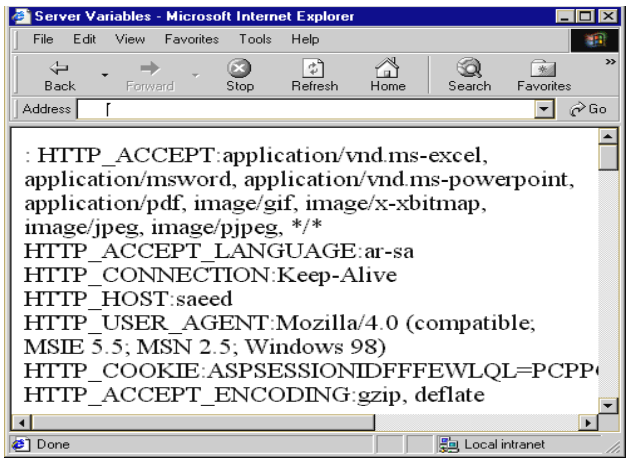
ASP دارای یک سری مجموعه‌ها و متدهایی است، که با استفاده از آنها می‌توانیم روی سرآیندها و متغیرهای محیطی کار کنیم. (لیست این سرآیندها و متغیرهای محیطی در ضمیمهٔ کتاب شرح داده شده است)

دریافت سرآیندها

وقتی مرورگر درخواست یک صفحه وب از سرویس‌دهنده می‌کند، این درخواست شامل تعدادی از سرآیندها می‌باشد. با استفاده از یکی از مجموعه‌های شیء Request به نام ServerVariables می‌توان این سرآیندها را به‌دست آورد. مجموعهٔ ServerVariables حاوی سرآیندها و عناصرو اطلاعات اضافی در رابطه با Server است.

برنامهٔ زیر از محتویات مجموعهٔ ServerVariables برای مرورگر یک کپی می‌فرستد. (شکل ۲- ٦)

```
<HTML>
<HEAD> <TITLE> Server Variables </ TITLE> </HEAD>
<BODY>
<%
FOR EACH Name IN Request.ServerVariables
Response.Write( "<P><B>"&"</B>: ")
Response.Write (Request.ServerVariables (Name))
NEXT
%>
</BODY>
</HTML
```



شکل ۲-۶ مجموعهٔ *ServerVariables*

سرآیندها ومتغیرهای سرویس‌دهنده در مجموعهٔ ServerVariables، دسته زیادی از اطلاعات را نگه می‌دارند.

در لیست زیر بعضی از مهم‌ترین‌های آنها آمده است.

می‌توان محتوای این متغیرها را با قرار دادن اسم آنها درون مجموعهٔ ServerVariables به‌دست آورد. به عنوان مثال، در اسکریپت زیر کسانی می‌توانند به این صفحه دسترسی داشته باشند، که از اسکریپت Origin.asp به این صفحه بیایند.

```
<HTML>
<HEAD> <TITLE> Server Variable </ TITLE> </ HEAD>
<BODY>
<%
WhereForm = Request.ServerVariable ( "HTTP_REFERER" )
IF WhereForm = "Http:// WWW.Mysite.Com/Origin.asp"  THEN
%>
Welcome to This page !
<%
ELST
%>
You are not authorized to viem this page !
<%
END IF
%>
</ BODY>
</ HTML>
```

در این اسکریپت سرآیند HTTP_REFERER، مسیر صفحه‌ای را که کاربر از آن به صفحهٔ موجود وصل شده است، نشان می‌دهد. اگر کاربر از مسیر Http://www.mysite.com/Origin.asp استفاده نکند، اجازه دیدن صفحه به آن داده نمی‌شود.

به‌کار بردن سرآیندها برای کنترل چگونگی Cache شدن صفحات

همانطور که قبلاً گفته شده بود، سرویس‌دهندهٔ پروکسی[1] باعث بالا رفتن سرعت دسترسی به صفحات وب می‌شود. این سرویس‌دهنده‌ها یک کپی محلی از صفحات وب را درون Cache نگهداری می‌کنند. وقتی کسی صفحه وبی را مجدداً درخواست می‌کند، می‌تواند آن را به جای سرویس‌دهندهٔ اصلی از پروکسی به‌دست آورد.

برای برنامه‌های ASP، سرویس‌دهندهٔ پروکسی مایهٔ دردسر است، چرا که تمام نکتهٔ یک ASP نمایش محتویات پویا و تأثرپذیر (دینامیک) می‌باشد. به عبارت دیگر، یک ASP می‌تواند صفحاتی را که در هر لحظه عوض می‌شوند، نشان دهد، ولی در سرویس‌دهندهٔ پروکسی نمی‌توان کپی تمام این صفحات را داشته باشیم. سرآیندی متعلق به ASP با نام CACHE_CONTROL دستورالعملی را برای چگونگی Cache کردن صفحات وب توسط سرویس‌دهندهٔ پروکسی، فراهم می‌کند. وقتی از ASP استفاده می‌کنید، به‌طور پیش‌فرض سرویس‌دهندهٔ پروکسی صفحات ما را Cache نمی‌کند. به هرحال می‌توانید این پیش-فرض را تغییر دهید. اگر سرویس‌دهنده‌های پروکسی را برای یک شخص به منظور Cache کردن ASP آن شخص می‌خواهید، می‌توانید خاصیت CacheControl را از شیء Response تغییر دهید. با استفاده از خط زیر در بالای صفحه، اجازه می‌دهید که صفحات توسط سرویس‌دهندهٔ پروکسی ذخیره شود.

```
<%
Response.CacheControl = "public"
%>
```

همچنین می‌توانید تعریف کنید، که به چه صورت مرورگرهای وب، صفحات ASP شما را Cache کنند. مرورگر عموماً Cache هایی روی حافظه و یا دیسک دارند. دو خاصیت از شیء Response چگونگی Cache شدن صفحات وب توسط مرورگر را کنترل می‌کند. با استفاده از خاصیت Expires متعلق به شیء Response، مقدار زمانی را که یک مرورگر صرف بازکردن یک صفحهٔ وب Cache شده می‌کند، مشخص می‌کنید. اگر این خاصیت را روی صفر تنظیم کنید، مرورگر به هیچ صورت نباید ASP ها را Cache کند. به مثال زیر توجه کنید.

```
<% Response.Expires= 0%>
```

شما همچنین می‌توانید تاریخ و زمانی را برای به پایان رساندن، کپی Cache در نظر بگیرید. در مثال زیر، مرورگر تا سال 2002 یک صفحه را Cache می‌کند.

```
<% #Response.ExpiresAbsolute =# Jan 1, 2002 00:00:00%>
```

تغییر دادن سرآیند Content-Type

سرآیند Content –Type، نشان‌دهندهٔ نوع محتویات ارسالی به مرورگر (نوع MIME) می‌باشد، که در ضمیمهٔ کتاب بیان شده‌اند. به عنوان چند مثال معمولی می‌توان به , "Image/Gif" , "Text/HTML" "Application / Msword" اشاره کرد. با استفاده از خاصیت Content-Type از شیء Response می‌توانید این سرآیند را تنظیم نمائید.

1. Proxy server

از خاصیت Content -Type به‌طور عمومی برای نمایش منابع اسناد HTML استفاده می‌شود. اگر خاصیت Content-Type را به‌صورت Text /Plain تنظیم کنیم، بدنهٔ پاسخ به جای HTML به‌صورت Text فرستاده می‌شود. مثال زیر را درنظر بگیرید.

```
<%
Response.Content Type = Text/Plain
%>
<HTML>
<HEAD> <TITLE> HTML Document </ TITLE> </ HEAD>
<BODY>
<H1> This is an HTML Document ! </H1>
</ BODY>
</ HTML>
```

وقتی این فایل نمایش داده می‌شود، تمام متنی که در زیر خط اسکریپت قرار دارد، به همان صورت به روی مرورگر نمایش داده می‌شود. با این کار آن را از حالت HTML درآورده‌ایم.

کد وضعیت

خاصیت Status برای تعریف کد وضعیتی است، که پاسخ HTTP بازمی‌گرداند. وقتی که سرویس‌دهنده‌ای به یک درخواست پاسخ می‌دهد، اولین خطی که فرستاده می‌شود، خط وضعیت[1] است. خط وضعیت شامل سه خانهٔ کد وضعیت و توضیحی دربارهٔ کد وضعیت (که Response نامیده می‌شود) می‌باشد. در زیر ۵ کلاس از کدهای وضعیت نمایش داده شده است.

- Information 1xx ، این کد, وضعیت در این کلاس آزمایشی است.
- Success 2xx ، کد وضعیت در این کلاس برای نشان دادن موفقیت درخواست به‌کار می‌رود. به عنوان مثال، کد 200 نشان می‌دهد، که صفحهٔ وب مورد درخواست با موفقیت دریافت شده است.
- Redirection 3xx کد وضعیت در این کلاس برای نشان دادن این است، که قبل از اینکه درخواست بتواند اجرا شود، باید عمل دیگری انجام شود. به عنوان مثال کد وضعیت 301 نشان می‌دهد که صفحه وب برای همیشه به آدرس دیگری منتقل شده است، در این حالت مرورگر به‌صورت خودکار به آدرس جدید برمی‌گردد.
- Cline Error 4xx این کد وضعیت هنگامی که مرورگر درخواستی را تولید کرده باشد که نتواند انجام شود، بازگردانده می‌شود. به عنوان مثال کد 404 موقعی که صفحهٔ وب وجود نداشته باشد بازگردانده می‌شود.
- Server Error 5xx این کد وضعیت نشان‌دهندهٔ مشکل درسرویس‌دهنده است. برای مثال کد وضعیت 503 نشانگر این است، که سرویس‌دهنده خراب شده است.

می‌توان از خاصیت Status در شیء Response برای تعریف کد وضعیتی که باید در یک پاسخ بازگردانده شود، استفاده کرد. به عنوان مثال اگر کسی در دریافت ASP زیر در روز چهارشنبه مشکل داشته باشد کد 407 Not Authorized برگردانده می‌شود. (این نتایج در جعبهٔ محاورهٔ کلمهٔ عبور ظاهر می‌شود)

1. Status

```
<%
IF WEEKDAYNAME (WEEKDAY (DATE)) = "Wednesday" THEN
Response.Status = "401 Not Authorized."
Response.End
ELSE
%>
<HTML>
<HEAD> <TITLE> Not Wednesday </ TITLE> </ HEAD>
<BODY>
 Welcome! Today is Not Wednesday.
</ BODY>
</ HTML>
<%END IF %>
```

هدایت یک کاربر به صفحهٔ دیگر

در بعضی مواقع نیاز است، که یک کاربر را به صفحه دیگری هدایت کنیم. به عنوان مثال اگر یک کاربر سعی کند فرم ثبت نام را به‌دست آورد، باید به صورت خودکار به صفحهٔ ثبت نام هدایت شود، یا اگر کاربری فرم اطلاعات را تکمیل کرد، باید به صورت خودکار به صفحهٔ دیگر راهنمایی شود. با استفاده از ASP هدایت یک کاربر به صفحهٔ دیگر بسیار ساده انجام می‌شود. متد Redirect از شیء Response به شما اجازه می‌دهد، تا کاربر را به صفحهٔ جدید هدایت کنید. به مثال زیر توجه کنید:

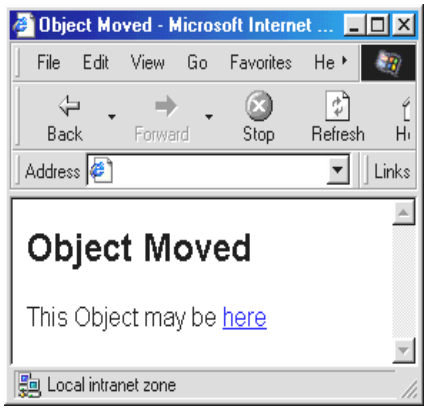
```
<%
IF Request.Form ("First Name ")= ""  THEN Response.Redirect
"http://www.asp.com/register.asp"
%>
<HTML>
<HEAD> <TITLE> Registration Results </TITLE> </ HEAD >
<BODY>
Tank you <%= Request.Form("Fistrname ")%> For registering!
</BODY>
</HTML>
```

تصور کنید، که کاربر فرم ثبت نام را کامل کرده و این صفحه آمده است. متد Response.Redirect در این مثال برای هدایت کاربر به فرم ثبت نام در مواقعی که کاربر نام خود را وارد نکرده، به‌کار می‌رود. باید از متد Response.Redirect قبل از اینکه متنی برای مرورگر کاربر فرستاده شود، استفاده کنید، از این‌رو استفاده از این متد در اسکریپتی که بالای آن برچسب <HTML > وجود دارد، ایدهٔ خوبی است. شما می‌توانید با استفاده از متد Response.Redirect کاربر را به هر URL قابل دسترسی راهنمایی کنید؛ که می‌تواند صفحهٔ دیگری روی سایت شما یا روی یک سایت دیگر در اینترنت باشد. البته تنها مشکل این است، که متد Response.Redirect با برگرداندن یک کد وضعیتی خاص کار می‌کند. وقتی که سرویس-دهنده به درخواستی پاسخ می‌دهد، کد وضعیتی را دراولین خط پاسخ قرار می‌دهد. به‌طور مثال وقتی که متد Response.Redirect صدا زده می‌شود، کد وضعیتی 302 که کد مربوط به Object Moved می‌باشد، بازگردانده می‌شود. یک سرآیند موقعیت نیز به پاسخ اضافه می‌شود، تا مکان صفحهٔ جدید را بدهد. بدین ترتیب کد وضعیتی و سرآیند موقعیت به صورت خودکار، مرورگر را به صفحهٔ جدید هدایت می‌کنند.

توجه: *متد Response.Redirect کاملاً هم ارز کدهای دوخط زیر است: (کدهای وضعیتی در ضمیمهٔ کتاب بیان شده‌اند)*

```
<%
Response.Status = "302 Object Moved”
Response.AddHeader  "Location " ,  "URL”
%>
```

در حقیقت همیشه اتفاق بالا نمی‌افتد و مرورگرهای قدیمی در حالت‌های خاصی با این نوع هدایت دچار مشکل می‌شوند. هنگامی که مرورگر نتواند به‌طور خودکار به کد وضعیت پاسخ دهد، پیغامی مشابه پیغام شکل ٤-٥ را دریافت می‌کنید، بنابراین برای جلوگیری از این مشکل از این متد اجتناب کرده و از متد هدایت شبیه‌سازی شده، که در بخش بعد آمده استفاده کنید.



شکل ۳-٦ نتیجهٔ راهنمایی سرویس‌دهنده

خواندن Cookie

برای خواندن یک cookie درون یک ASP، باید از مجموعهٔ cookie از شیء Request استفاده کنید. به‌عنوان مثال، برای استخراج مقدار cookie می‌توانید ازاسکریپت زیر استفاده کنید:

```
<%=Request .cookies ("UserName”)%>
```

این اسکریپت مقدار یک cookie با نام UserName را استخراج می‌کند. در اینجا نیز می‌توانید با استفاده از صفت count تعداد عناصر مجموعه در cookie هارا به‌دست آورید. همچنین می‌توانید با حلقه‌های FOR...NEXT , FOR...EACH بین عناصر مجموعهٔ cookie حرکت کنید. به مثال زیر توجه کنید:

```
<%
FOR   EACH    thing IN  Request.cookies
Response.Write ("<BR>"&thing&Request.cookies (thing))
NEXT
 %>
```

به وجود آوردن بیش از یک cookie

می‌توانید با استفاده از مجموعهٔ Response.cookie (همانند مثال قبل) بیش از یک cookie به وجود آورید. هر چند که خیلی از مرورگرها فقط سه یا چهار cookie را از یک سایت وب خاص پشتیبانی می‌کنند.

یک روش جایگزین برای به‌وجود آوردن چند cookie وجود دارد: شما می‌توانید یکcookie dictionary به وجود آورید. یک cookie dictionary یک cookie تکی با چند مقدار و نام می‌باشد. مثالی برای به وجود آوردن cookie dictionary در زیر آمده است.

```
<%
Response.cookies ("User")("Name")="Bill  Gates"
Response.cookies ("User")( "password")= "Bill ons"
%>
```

این اسکریپت cookie dictionary با نام user و کلیدهای Name و password به وجود می‌آورد. وقتی یک cookie dictionary به وجود آید، سرآیند زیر به مرورگر فرستاده می‌شود.

```
set . cookie : User = Name =Bill +Gates &password =billions
```

یک cookie با نام user به وجود می‌آید. نام و مقدار هر کلید از cookie dictiovary در یک cookie بزرگ قرار می‌گیرند.

برای به‌دست آوردن یک cookie dictionary می‌توانید، همانند مثال قبل از مجموعهٔ cookie dictionary استفاده کنید. برای به‌دست آوردن کلیدهای خاص از cookie dictionary نام هر کلید را درون مجموعه می‌آوریم. مثال زیر را در نظر بگیرید.

```
<%= Request. cookies ("User")%>
<%=Request. cookies ("User)( "Name")%>
<%=Request. cookies ("User ")("password")%>
```

برای تعیین اینکه آیا یک cookie از نوع cookie dictionary است، یا نه از صفت Haskeys استفاده می‌کنیم. به عنوان مثال، اسکریپت زیر اگر یک cookie جدید از cookie dictionary باشد، مقدار True و در غیر این صورت مقدار false را باز می‌گرداند.

```
<%=Request.cookies ("User").Haskeys%>
```

۲-۶-۶ شیء Request

این شیء همان‌طور که از نام آن مشخص است، جهت دسترسی به اطلاعات ارسال شده از سرویس-گیرنده به برنامهٔ ASP می‌تواند به‌کار رود. این شیء دارای متدها، خصوصیات و مجموعه‌های زیر است.

Collection:	
Cookies	جهت دسترسی به cookie ها به‌کار می‌رود.
Form	جهت دسترسی به اطلاعات ارسال شده از یک فرم به برنامهٔ ASP با متد POST به‌کار می‌رود.
QueryString	جهت دسترسی به اطلاعات ارسال شده از یک فرم به برنامهٔ ASP با متد

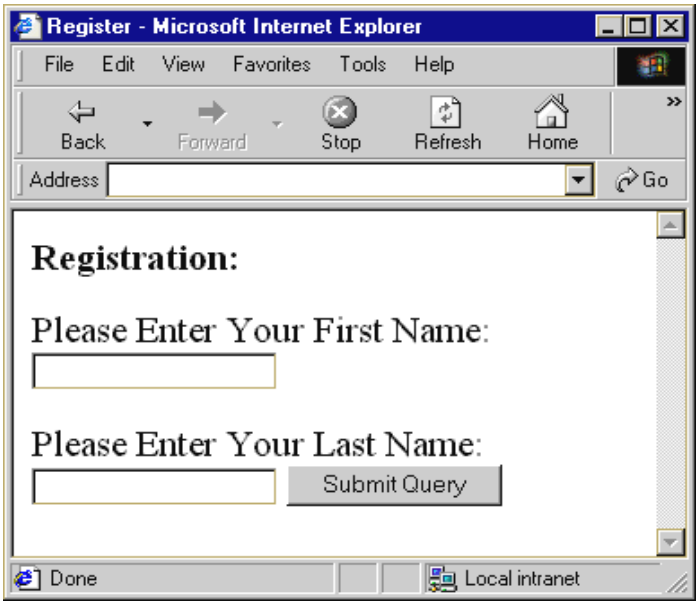
	GET به‌کار می‌رود.
ServerVariable	جهت دسترسی به متغیرهای محیطی سیستم عامل به‌کار می‌رود.
Properties:	
TotalByte	مجموعه بایت‌های ارسالی.
Method	
BinaryRead	جهت خواندن اطلاعات به‌صورت باینری به‌کار می‌رود.

حال هر کدام را با کاربردشان شرح خواهیم داد.

دریافت محتویات یک فرم HTML

بهترین راه برای گرفتن اطلاعات از کسی، که وارد سایت وب می‌شود یک فرم HTML است. همانطور که در قبل می‌دانستید، اطلاعات داخل یک فرم را می‌توان از طریق برنامهٔ دروازه‌ای که یک برنامهٔ EXE. یا DLL است پردازش کرد، این کار را از طریق یک برنامهٔ ASP نیز می‌توان انجام داد. فرض کنید می‌خواهید قبل از اینکه کسی از سایت شما دیدن کند، از آن شخص ثبت نام کنید. برای این ثبت نام نیز می‌توانید از فرم HTML استفاده کنید. این فرم نام ونام خانوادگی ملاقات‌کننده را می‌پرسد. بعد از اینکه کاربر نام ونام خانوادگی خودرا وارد کرد، کاربر می‌تواند دکمهٔ submit یا move را برای رفتن به صفحهٔ بعدی بزند. باید این اطلاعات از فرم دریافت شود. حال نیاز به متدی برای دریافت این اطلاعات می‌باشد. هنگامی که شما فرم HTML را می‌فرستید، آن فرم همانند یک درخواست پروتکل HTTP ارسال می‌شود. در اینجا با استفاده از متد POST اطلاعات را به سرویس‌دهنده می‌فرستیم. شما اطلاعات را می‌توانید از شیء Request از ASP دریافت کنید. (شکل ٤- ٦)

```
<HTML>
<HEAD> <TITLE>  Register  < /TITLE> </HEAD>
<BODY>
<H4> Registration :</H4>
<FORM  METHOD = "POST"  Action = "http://www.asp.com/regresults .asp”>
<P> Please  Enter  Your  First  Name:
<BR> <INPUT  NAME="FirstName”  Type = “TEXT”>
< p> Please  Enter Your  Last   Name:
<BR> <INPUT   NAME = "LastName”  Type  = “TEXT”>
<INPUT    TYPE = " SUBMIT " value = " continue”>
</FORM>
</BODY>
</HTML>
```



شکل ٤-٦ یک فرم ثبت نام ساده

شیء Request دارای مجموعهٔ Form می‌باشد، که از طریق آن می‌توان به تمامی اطلاعات وارد شده در فرم دسترسی پیداکرد. هر کلیدی روی Form به منزلهٔ یک ورودی در فرم HTML می‌باشد. به‌طور نمونه در مثال بالا اگر کاربر فرمی را پر کرده و سپس ارسال کرده باشد، مجموعهٔ Form شامل دو خط ورودی خواهد بود و اولین خط ورودی FirstName دومی LastName است.

به اسکریپت زیر توجه کنید:

```
<HTML>
<HEAD> <TITLE> Registration Results < /TITLE> </HEAD>
<BODY>
Thank  you  <%= Request.Form ("first Name ")%> for  registering!

</BODY>
</HTML>
```

وقتی که این صفحه روی مرورگر نمایش داده می‌شود، اسمی که کاربر درون فرم HTML وارد کرده است، نمایش داده می‌شود. اطلاعات وارد شده در این Form به برنامهٔ Asp مشخص شده در Action ارسال می‌شود. باید بدانید که وقتی فرم انتشار می‌یابد، شما قادر نیستید که به محتوای یک فرم HTML در صفحهٔ مشابه‌ای دسترسی پیدا کنید. فرم HTML باید اول قبل از اینکه شما بتوانید مقادیر و اجزای آن را ارسال کنید، با زدن دکمهٔ مربوطه ارسال شود. وقتی یک صفحه فرم HTML دارید، باید یک صفحه اضافه نیز برای پردازش فرم داشته باشید. همین امر نشان می‌دهد که سایت‌های وب اینترنت محدودیت زیادی در دادن سرویس دارند. هنگامی که فرم ثبت نام پردازش می‌شود، یک صفحه شامل نتیجهٔ کار ساخته شده و برای کاربر ارسال می‌شود.

دسترسی به همهٔ اطلاعات موجود درForm

راه‌های زیادی برای دریافت محتوای داخل Form وجود دارد. اگر بخواهید مروری بر محتوای داخل Form داشته باشید و هر کدام را که لازم بود نمایش دهید، باید از اسکریپتی مشابه اسکریپت زیر استفاده کنید:

```
<%
FOR  EACH    name  IN   Request.Form
 Response.Write("<BR>"& name&"=")
Response.write(Request.Form (name))
NEXT
%>
```

این اسکریپت، محتویات داخل Form را نمایش می‌دهد. به عنوان مثال اگر بیل گیتز از سایت دیدن کند و فرم ثبت نام را پرکند، خروجی این اسکریپت به‌صورت زیراست:

```
FIRSTNAME =Bill
LASTNAME = Gates
```

توجه کنید که در مثال‌های بالا، اسامی چطور به حروف بزرگ تبدیل شده‌اند. هنگامی که می‌خواهید یک آیتم را در فرم وارد کنید، نمی‌خواهد نگران کوچک و بزرگ بودن آنچه وارد می‌کنید باشید، به عبارتی تفاوتی بین First Name و First name وجود ندارد. به‌جای استفاده از حلقهٔ FOR....EACH می‌توان از حلقهٔ FOR....NEXT برای کپی کردن محتویات Form استفاده کرد. اسکریپت زیر مقادیر هر جزء داخل Form را برمی‌گرداند.

```
 <%
FOR  i = l  TO   Request.Form.Count
    Response.write ("<BR>"& Request . Form (i))
NEXT
 %>
```

در این اسکریپت، خاصیت Count مربوط به مجموعهٔ Form برای تعیین تعداد اجزای مجموعه Form استفاده می‌شود. با استفاده از خاصیت Count، هنگامی که فرم ارسال می‌شود می‌توان تعداد، فیلدهای فرم HTML ای که توسط کاربر پرشده است را به‌دست آورد.

در آخر اگر می‌خواهید تمام اطلاعات موجود در ورودی استاندارد را بدانید، می‌توانید از اسکریپت زیر استفاده کنید:

```
<%= Request.Form %>
```

خروجی این اسکریپت به‌صورت زیراست:

```
FirstName = Bill & LastName  = Gates
```

عناصر فرم با چندین مقدار

عناصر موجود در یک فرم HTML می‌توانند چندین مقدار داشته باشند. به عنوان مثال هم CheckBox وهم list Box ها می‌توانند، چندین مقدار مختلف داشته باشند: (شکل ٥- ٦)

```
<FORM  METHOD = "POST "   ACTION = " /regresults.asp">
How  did  you  hear   about   our    web  site?
<BR> <INPUT NAME="HowHear" Type ="CheckBox "   value =" A Newspaper">
A Newspaper  Article
```

```
BR><INPUT NAME =" How Hear "  Type =" checkBox "  value >
< ="SearchEngin"
search  Engine
< BR> <INPUT   NAME= "HowHear"  Type =" checkBox"  VALUE=" Afriend">
A Friend
 <BR> <INPUT  NAME = "HowHear"  Type ="checkBox "   stumbled Into It">
 VALUE="

stumbled Into It
< P> <INPUT  TYPE =" SUBMIT"  VALUE = "Continue">
</FORM>
```

شکل ۵-٦ فرمی با چندین مقدار

این فرم را می‌توانید برای اینکه بفهمید کاربران سایت وب شما آن را چطور پیدا کرده‌اند، مورد استفاده قرار دهید، ولی ممکن است کسی از طرق مختلفی به سایت شما رسیده باشد. به عنوان مثال هم فرم friend و هم فرم newspaper را خواسته باشید، این فرم به کاربر اجازه می‌دهد، که بیش از یک مقدار را در یک زمان انتخاب کنید. به چه صورت می‌توان مقادیر عناصری را که بیش از یک مقدار دارند به‌دست آورد؟ پاسخ این است که شما می‌توانید از یک پارامتر اضافی مجموعهٔ Form استفاده کنید. به مثال زیر توجه کنید:

```
<HTML>
<HEAD> <TITLE> Your Response < /TITLE> </HEAD>
<BODY>
Accding  to your respone ,  you heard about this web site in
Request.form ("How Hear"). count %> ways =%>
<P>  you heard about this form
<%
FOR  EACH  way IN Request.form("HowHear")
Response.write ("<P>" & way")
NEXT
%>
```

```
</BODY>
</HTML>
```

در این اسکریپت، خاصیت Count برای بازگرداندن تعداد گزینه‌های انتخاب شده به کار می‌رود. در این مثال، خاصیت Count تعداد کلی عناصر مجموعهٔ Form را باز نمی‌گرداند. به جای آن فقط شمارهٔ کلی مقادیر عنصر فرم، که HowHear نامیده می‌شود باز می‌گرداند. حلقهٔ تکرار FOR....EACH به تعداد مقادیر تکرار می‌شود. در این مثال اگر کسی هم friend و هم Newspaper را انتخاب کند، هر دو مقدار نمایش داده می‌شود.

text Area و مجموعهٔ Form

شما می‌توانید متنی را که درون TextArea قرار دارد، به همانند سایر عناصر فرم دریافت نمائید. متغیرهای رشتهٔ درون VBScript می‌توانند، کاملاً طولانی باشند. در این جا محدودیت ۲۵۵ کاراکتری، که در زبان‌های دیگر هست، وجود ندارد. مثال زیر یک مجموعهٔ Form به همراه یک TextArea است:

```
<FORM  METHOD ="POST" ACTION ="http://www.asp.com/Respoose .asp”>
Please enter any feedback on this web site below:
<P>
<TEXTAREA  NAME = "Feedback” cols =30  ROWS=10></TEXTAREA>
< P> <INPUT  TYPE = " SUBMIT"     VALUE =" feedback”>
</FORM>
```

این فرم HTML یک TextArea را که می‌تواند به عنوان خروجی کاربر روی سایت وب مورداستفاده قرارگیرد، نشان می‌دهد. هنگامی که کاربر روی کلید feedback کلیک می‌کند به روی صفحهٔ Response.asp می‌آید. اگر می‌خواهید متنی را که درون Text Area نوشته شده است، نمایش دهید، مانند زیر عمل کنید:

```
<HTML>
<HEAD> <TITLE> Feedback Response </TITLE > </HEAD>
<BODY>
Thank you for submiting feedback   you wrote:
<P>
<%=Request.Form ("feedback”)%>
</BODY>
</HTML>
```

بررسی وجود اجزای فرم

بعضی وقت‌ها باید بررسی کرد که شخص فرم را به‌صورت کامل پرکرده است یا خیر. به عنوان مثال یک فرم ثبت نام ساده یک سری فیلدهای خاص را لازم دارد. شما نمی‌خواهید اجازه بدهید، که کاربر بعضی از این فیلدها راخالی بگذارد. با اسکریپت زیر می‌توانید این مسئله را بررسی کنید.

```
<%
IF Request.Form ("firstname") =" " THEN
  Response.write("you must enter your first  name.”)
ELSE
  Response.write("Thank you for regstering.”)
END IF
%>
```

در این اسکریپت بررسی می‌شود، که فیلد firstname دارای مقدار باشد. عنصرfirstname با یک رشتهٔ تهی مقایسه می‌شود. اگر کاربر در وارد کردن نام خود کوتاهی کرده باشد از این مسئله آگاه خواهد شد. هنگامی که کاربر فرم اطلاعات را ناقص پرکرده باشد شما باید او را دوباره به آن فرم برگردانید. شما می‌توانید این کار را با گذاشتن یک پیوند مثلاً با نام Back یا هر چیز دیگر انجام دهید. البته بهتر خواهد بود، اگر بتوان این مسئله را خودکار کرد، یعنی اگر فرم ناقص بود دوباره فرم قبلی بیاید.

دریافت یک QueryString

یک QueryString، قسمت اعظم URL را تشکیل داده و بعد از علامت سؤال نمایش داده می‌شود. اگر از موتورهای جستجوگر اینترنت مثل AltaVista استفاده کرده باشید با رشتهٔ درخواست آشنا شده‌اید. این کاراکترها را شما می‌توانید در خط آدرس مرورگرتان موقع جستجو ببینید. شما می‌توانید از رشتهٔ درخواست، توسط یک پیوند برای فرستادن اطلاعات از یک صفحه به صفحهٔ دیگر استفاده کنید، به مثال زیر توجه کنید:

```
<HTML>
<HEAD> <TITLE> QueryString Example  </TITLLE > </HEAD>
<BODY>
A HREF = "http : //www.aspsite.com/newpag.asp?click=yes ">Click >
<me!</A
</BODY>
</HTML>
```

دراین مثال، پیوند به صفحهٔ newpage.asp برمی‌گردد. هرچند که پیوند می‌تواند شامل یک QueryString هم باشد، ولی به هر صورت وقتی کسی روی کلمهٔ !click meکلیک می‌کند. عبارت "click = "yes به صورت یک درخواست برای صفحهٔ جدید فرستاده می‌شود.

می‌توانید QueryString را مستقیماً با نوشتن آن درون خط آدرس مرورگر نیز بفرستید. این مسئله برای سرویس‌دهنده فرقی نمی‌کند. به عنوان مثال وارد کردن درخواست زیر در خط آدرس اثر کلیک کردن روی لینک را دارد.

```
HTTP://www.aspsite.com /newpage.asp? click =yes
```

QueryString برای زمانی که شما می‌خواهید منویی از گزینه‌ها بسازید مفیدتر است. اگر چندین لینک دارید، که به یک صفحه اتصال می‌یابند، می‌توانید ازQueryString برای تعریف لینک مخصوصی که روی آن کلیک می‌کنید، استفاده کنید. مثال زیر را در نظر بگیرید:

```
<HTML>
<HEAD> <TITLE>product List </TITLE > </HEAD>
<BODY>
<H3> Welcom  To Our Store:    </H3>
Please  select the  item you  want  to purchase form the   list
belon:
P><A  HREF = "http://www.asp.com/Purchas.asp ? ITEM=1" > Used Book >
<</A
P><A  HREF = "http://www.asp.com/Purchas.asp ? ITEM=2"> Broken >
<Typewriter </A
```

```
P> <A HREF = "http://www.asp.com/Purchase.asp? ITEM=3"> Horseshoe >
<</A
</BODY>
</HTML>
```

در این صفحه چندین عنصر که کاربر برای خرید می‌تواند آنها را انتخاب کند، وجود دارد. در هر نمونه یک پیوند تعریف شده است. وقتی کاربری روی هر عضو کلیک می‌کند، آن شخص به صفحهٔ purchase.asp وصل می‌شود. (شکل ٦-٦)

شکل ٦-٦ صفحه ای با لیست عناصر فروش

می‌توان از طریق متد QueryString فهمید، که کاربر روی کدام Link کلیک کرده است.

```
<HTML>
<HEAD> <TITLE> Purchase  </TITLLE > </HEAD>
<BODY>
<%
 SELECT CASE Request.QuerySting ("item")
     CASE"1"
          Response.Write ("Thank you for purchasing a used book.")
     CASE "2"
          Response.Write ("Thank you for purchasing a broken
typewriter")
     CASE "3"
          Response.write ("Thank you for purchasing  a  horseshoe.")
  END SELECT
%>
</BODY>
</HTML>
```

مجموعهٔ QuerySting در مثال بالا هر پیوندی که روی آن کلیک شده است، مشخص می‌کند. عبارت SELECTCASE بسته به نوع رشتهٔ درخواست، پاسخی را اختصاص داده است. به عنوان مثال اگر کسی Horseshoe را انتخاب کند، از کاربر برای این انتخاب تشکر می‌شود.

رشته‌های درخواست با پارامترها و مقادیر چندتایی

شما می‌توانید بیش از یک نام و مقدار را در QuerySting بفرستید. به عبارت دیگر شما می‌توانید QuerySting ای با چند پارامتر به وجود آورید. همان‌طور که می‌دانید برای فرستادن چند پارامتر باید از کارکتر (&) استفاده کنید. مثلاً در QuerySting زیر دو پارامتر فرستاده شده است.

```
<A  HREF ="http://www.asp.com/Response .asp ?Firstparam= <%=Server
.URLEncode ("this is the first parameter
.")%>&secondparam=<%server.URLEncode(this is the second parameter .
")>"%> Click Here </A>
```

این رشتهٔ درخواست، شامل دو پارامتر است که Firstparam وSecondparam نامیده می‌شود. پارامتر اول‌مقدارش برابربا this is the first parameter وپارامتر دوم مقدارش برابربا This is the seconde paramter می‌باشد. متد ( )server.URLEncode مقدار این پارامتر را عوض می‌کند تا آنها بتوانند فرستاده شوند. در صفحهٔ response.asp شما می‌توانید مقدار دو پارامتر را به‌صورت زیر خارج کنید:

```
P> <% = Request .QuerySting ("Firstparam")%> <
>P><%= Request .QueryString ("secondparam")% <
```

با فرستادن نام هر رشتهٔ درخواست در مجموعه QueryString، می‌توانید مقدار هر پارامتر را به‌دست آورید. خروجی دو جملهٔ قبلی به صورت خواهد بود.

```
This is the first parameter.
This is the second  parameter.
```

همچنین می‌توانید به یک پارامتر چندین مقدار را نسبت دهید، برای این منظور به‌راحتی می‌توانید یک نام را دوبار در QueryString همانند مثال زیر بیاورید:

```
<A HREF ="/Response.asp?onlyparam= <%=Server.URLEncode ("I am the
first value of The only parameter.")%>& onlyparam = <%= Server
.URLEncode  ("I am The second value of The only parameter. ") %>">
Click Here </A>
```

در این مثال، به پارامتر Onlyparam دو مقدار نسبت داده شده است. با استفاده از خاصیت count می‌توانید تعداد مقادیری که به هر پارامتر نسبت می‌دهید را مشخص کنید. مثال زیر تعداد مقادیری که به یک پارامتر نسبت داده‌اید، نمایش می‌دهد.

```
The  Onlyparam parameter  has
<% Request .QueryString ("onlyparam").Count%> values .
<P> There are
<%
  FOR EACH  pvalue IN Request.QueryString ("onlyparam ")
Response.write ("<BR>"& pvalue")
NEXT
%>
```

حلقهٔ FOR....EACH به تعداد مقادیر پارامتر Onlyparam تکرار می‌شود. اگر پارامتر Onlyparam دارای مقدار صفر باشد، خاصیت Count، شمارهٔ صفر را باز می‌گرداند و هیچ مقداری را نمایش نمی‌دهد.

دسترسی به همهٔ اطلاعات موجود در QueryString

اگر می‌خواهید تمام پارامترهای مجموعهٔ QueryString را به‌دست آورید باید از حلقهٔ FOR...EACH درون مجموعه استفاده کنید، به عنوان مثال، اسکریپت زیر را در نظر بگیرید:

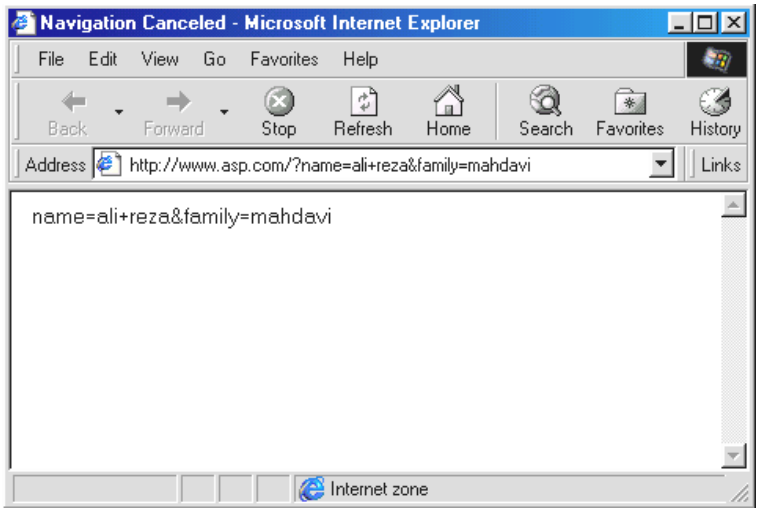
```
<%
FOR EACH  QSParam IN  Request.QueryString
     Response.Write ("<BR>"&QSParam&"=")
Response.Write (Requrest.QueryString (QSParam))
NEXT
%>
```

به جای حلقه FOR...EACH می‌توانید از حلقهٔ FOR...NEXT نیز استفاده کنید. برای این کار باید شمارهٔ هر عنصر درون مجموعهٔ QueryString را به‌دست آورید. این کار را می‌توانید با خاصیت Count انجام دهید. همانند مثال زیر:

```
<%
FOR i=1  TO    Request.QueryString.Count
 Response.Write ("<BR>"&Request.QueryString (i))
NEXT
%>
```

اگر ترجیح می‌دهید، که QueryString را به صورت غیرمجزا به‌دست آورید، می‌توانید از مجموعهٔ Request.QueryString بدون پارامتر استفاده کنید. که به صورت زیر می‌نویسید: (رجوع شود به شکل زیر)

```
<% =Request.QueryString %>
```



شکل ۷-۶ یک رشتهٔ پویش نشده

زمان‌هایی که از رشتهٔ درخواست استفاده نمی‌کنید

QueryString، برای وقتی مفید است که لازم است چند بیت یا اطلاعات محدود را از یک صفحه به دیگری بفرستید و در دو وضعیت از QueryString استفاده نمی‌کنیم: یکی وقتی که بخواهیم اطلاعات مخفی را بفرستیم و دیگری وقتی که بخواهیم اطلاعاتی با حجم زیاد را بفرستیم. یک QueryString به هیچ‌وجه نمی‌تواند مخفی باشدو معمولاً روی خط آدرس مرورگر ظاهر می‌شود. بنابراین فرستادن کلمهٔ عبور از یک صفحه به صفحهٔ دیگر از طریق QueryString، کار درستی نمی‌باشد.

QueryString برای اطلاعات باحجم بالا قابل استفاده نمی‌باشد. تعداد کاراکترهایی که یک QueryString می‌تواند داشته باشد، به چند پارامتر بستگی دارد؛ یکی از آنها مرورگری که استفاده می‌شود، می‌باشد. مرورگرهای مختلف محدودیت‌هایی در اندازه‌های QueryString دارند. به عنوان مثال Microsoft Internet Explorer نسخهٔ 4.0 نمی‌تواند QueryString با بیشتر از ۲۰۰۰ کاراکتر را پشتیبانی کند. بنابراین برای نگه داشتن رشتهٔ کوتاه QueryString، ایده خوبی است، ولی برای رشته‌های طولانی نه. اگر نیاز به فرستادن میزان زیادی اطلاعات از یک صفحه به صفحهٔ دیگری دارید، می‌توانید از Form استفاده کنید. بدین ترتیب دیگر محدودیتی وجود نخواهد داشت. پروتکل HTTP، فیلدهای یک فرم را به‌صورت‌های بهتر ی نسبت به QueryString می‌فرستد.

همراه کردن فایل‌ها در Asp

با استفاده از هدایت کنندهٔ INCLUDE به راحتی می‌توانید روی سرویس‌دهنده، یک فایل دیگر را با ASP همراه نمایید. هدایت کنندهٔ INCLUDE روی سرویس‌دهنده نباید درون اسکریپت قرار گیرد. بلکه آن را خارج از اسکریپت و مانند جزئی از کد HTML می‌آوریم؛ همانند مثال زیر:

```
<HTML>
< HEAD> <TITLE>Welcome </TITLE> </ HEAD>
<BODY>
<!--#INCLUDE  VIRTUAL ="mybanner.inc”-- >
  Welcom To Our Web Site!
</BODY>
</HTML>
```

در این مثال، فایل mybanner.inc را در یک ASP و زیر برچسب <BODY> وارد می‌کنیم. وقتی این ASP اجرا می‌شود، هر اسکریپت یا کد HTML ای که درون فایل mybanner.inc قرار دارد، شامل ASP بالا نیز می‌شود.

یک فایل را می‌توانید به دو صورت همراه کنید؛ یا با به وجود آوردن یک مسیر مجازی برای فایل همانند مثال قبل، یا با استفاده از مسیر فیزیکی فایل، که مثال زیر روش دوم را نشان می‌دهد.

```
<HTML>
< HEAD> <TITLE>Welcome </TITLE> </ HEAD>
<BODY>
<!--#INCLUDE  FILE ="mybanner.inc”-->
Welcom to our web site !
</BODY>
</HTML>
```

اگر برای همراه کردن فایل از مسیر فیزیکی استفاده می‌کنید، باید ازمشخصهٔ FILE استفاده کنید، فایل باید درون فهرست جاری یا زیر فهرست آن قرار گیرد. مسیر فایل به فهرست جاری بستگی دارد، این یک محدودیت مهم است، بنابراین شما معمولاً از مشخصهٔ VIRTUAL استفاده خواهید کرد.

فایل همراه هر نام یک پسوند می‌تواند داشته باشد. معمولاً فایل‌هایی را که همراه می‌کنیم، دارای پسوند inc است، اما می‌توان از فایل‌های htm. ،html. ،asp. و هر پسوند دیگری استفاده کرد.

همراه کردن فایل‌ها به دو دلیل مناسب است. اول آنکه شما می‌خواهید همان محتوا یا اجرای همان صفحهٔ اسکریپت را به‌صورت صفحات پی‌درپی نمایش دهید. به عنوان مثال غیرعادی نیست، که هر صفحه درون سایت وب bannerوfooter یکسان داشته باشد. به جای اینکه کدهای HTML را مرتباً تکرار کنید، می‌توانید به راحتی فایل Footer و Banner را مانند قسمتی از هر صفحه همراه آن کنید. شما می‌توانید یک اسکریپت ASP را در چندین صفحه با استفاده از هدایت کننده INCLUDE همراه کنید. به هرحال، چون هدایت کننده INCLUDE باید خارج از اسکریپت باشد، اسکریپت شما باید کاملاً در علامت-های جداکننده قرار بگیرد. همچنین آنها را به صورت قطعه قطعه نمی‌توان همراه کرد.

دلیل دوم برای اینکه یک فایل را همراه دیگری می‌کنیم، این است که می‌خواهیم هدایت سرویس‌دهنده را شبیه‌سازی کنیم. برای انجام این کار، تمام ASP های خود را همراه صفحهٔ دیگر می‌کنیم. برای انجام این کار باید یک Save As خالی در ASP دوم همراه کنیم. مثال زیرا را درنظر بگیرید.

```
<%
IF Request.Form ("FirstName ")=" "  THEN
%>
<!--#INCLUDE VIRTUAL = "Register.Asp”-->
%>
 Response .End
END   IF
<%
<HTML>
<HEAD> <TITLE>Registration  Results </TITLE> </ HEAD >
<BODY>
Thank you <%= Request.Form("FirstName ")%>For    registering
</BODY>
</HTML>
```

این مثال کاملاً همان اثر متد Response.Redirect را دارد. اگر کاربری فراموش کند، فیلد FirstName از فرم ثبت نام را پرکند، آن شخص به صفحهٔ ثبت نام بازگردانده می‌شود. به هر حال، چون همراه کردن فایل به صورت کامل درون سرویس‌دهنده انجام می‌شود، این هدایت شبیه‌سازی شدهٔ فایل، قابل اعتمادتر از هدایت کامل است.

به متد Reposone.End در این مثال توجه کنید. متد Reposone.End تا زمانی که صفحه ثبت نام نمایش داده می‌شود، از نمایش صفحات دیگر ASP و ماندن آنها روی صفحه جلوگیری می‌کند.

دانستن اینکه IIS، هدایت‌کننده INCLUDE را قبل از هر اسکریپت دیگری پردازش می‌کند، حائز اهمیت است. این بدان معناست که شما نمی‌توانید از دستور اسکریپت زیر در آن استفاده کنید و در صورت نوشتن کار نخواهد کرد.

```
<%
```

```
IF Request.Form("First Name ")=" "  THEN
myInclude = "register.asp"
ELSE
myInclude ="Homepape.asp"
END IF
%>
<!--# NCLUDE   VIRTUAL = <%=myInclude%> -->
```

این اسکریپت به این دلیل کار نخواهد کرد، که سرویس‌دهنده همراه کردن فایل را قبل از اجرای هر اسکریپت دیگر انجام می‌دهد. یعنی سرویس‌دهندهٔ اول، خط <٪ myinclude=٪> را اجرا می‌کند، که دراین صورت چیزی برای اجرا وجود ندارد.

۳-۶-۶ شیء Session

هنگامی که ارتباطی از مرورگر به سرویس‌دهنده برقرار می‌شود، یک جلسه ایجاد می‌گردد. این جلسه دارای شناسه‌ای خاص بوده و مدتی برقرار می‌ماند. اگر درخواست دیگری در این مدت زمان به سرویس‌دهنده ارسال نشود، جلسه خاتمه خواهد یافت.

از طریق session می‌توان متغیرهایی را ایجاد کرد، که در تمام طول جلسه بتوان آنها را از یک صفحه به صفحه دیگر انتقال داد.

Session ها می‌توانند سلیقه‌های بازدیدکنندگان را ذخیره کنند. به عنوان مثال اینکه آیا ملاقات کننده ترجیح می‌دهد صفحات وب دارای پس زمینهٔ آبی باشد یا سبز؟ آیا ملاقات کننده ترجیح می‌دهد که نسخهٔ غیرگرافیکی و متنی از سایت شما را ببینید؟ تمام این انتخاب‌ها می‌تواند توسط Sessionها پی‌گیری شود.

همچنین Session ها می‌توانند برای تهیهٔ کارت‌های خرید مجازی به‌کار بروند. هر وقت بازدید کننده‌ای، اقلامی را برای خرید از سایت وب شما انتخاب می‌کند، این نمونه‌ها به کارت خرید او اضافه می‌شود. وقتی کاربر می‌خواهد سایت شما را ترک کند او تمام چیزی‌های را که درکارت خریدش وجود دارد به یکباره خریداری کرده است. تمام اطلاعات دربارهٔ این نمونه‌ها، درون کارت خرید توسط Session ها ذخیره می‌شوند. در آخر Session ها می‌توانند اعمال بازدیدکنندگان سایت شما را تعقیب کنند.

شئ Session دارای مجموعه‌ها و متدها ورخدادهای زیر می‌باشد:

Collection:	
Contents	جهت ایجاد متغیر از نوع Session به‌کار می‌رود و خودش دارای متدهای Remove و Remove All می‌باشد.
Properties:	
Session ID:	جهت به‌دست آوردن شناسهٔ یک Session، از آن استفاده می‌شود.
Timeout	جهت مشخص کردن زمان خاتمهٔ یک Session، از این خصوصیت استفاده می‌شود.
Methods:	
Abandon	جهت از بین بردن یک شیء Session به‌کار می‌رود.

Events:	
Session-OnEnd	با شروع شدن یک Session این رخداد ایجاد می‌گردد.
Session-OnStart	با خاتمهٔ یک Session این رخداد ایجاد می‌گردد.

ردیابی کاربران با Session

Session برای محدودهٔ پروتکل HTTP ابداع شده بود. به‌یاد دارید، که پروتکل HTTP به چه صورت کار می‌کند. وقتی کاربر درخواستی را مطرح می‌کند، سرویس‌دهنده پاسخ آن را می‌فرستد. تمام این فعل و انفعالات، بین مرورگر و سرویس‌دهندهٔ وب به صورت جفت‌های درخواست و پاسخ مطرح می‌شود.

هیچ چیزی در پروتکل HTTP به سرویس‌دهنده اجازه نمی‌دهد، کاربری که کد درخواستی را به وجود آورده است تعقیب کند. پس از اینکه سرویس‌دهنده به درخواست فرستاده شده پاسخ داد، دیگر برای سرویس‌دهنده مرورگری که این درخواست را فرستاده، قابل شناسایی نیست.

سرویس‌دهندهٔ وب هر درخواست جدید را به عنوان شخص جدیدی درنظر می‌گیرد. این مسئله همان حالت State Less بودن پروتکل HTTP است، که قبلاً شرح دادیم. پروتکلHTTP وضعیت کاربر را نمی‌تواند نگه دارد. این یک محدودیت جدی است، به این معنی که شما نمی‌توانید کاربر را روی صفحات مختلف سایت وب تشخیص بدهید.

Session ها برای برطرف کردن این مشکل به وجود آمده‌اند. با استفاده از Session ها می‌توانید اطلاعاتی دربارهٔ کاربرانی که روی صفحات مختلف وب حرکت می‌کنند، به‌دست آورید. Session ها توانایی کارهای بسیار مشکل و یا غیرممکن را دارند.

ذخیره کردن اطلاعات Session

کار با SessionدرASP بسیار آسان است. توسط شیء Session از ASP می‌توان تمام کنترل‌های لازم را روی Session انجام داد. اگر لازم است، اطلاعات کاربر را درون یک Session ذخیره کنید. می‌توانید این داده‌ها را درون یک متغیر از نوع Session ذخیره کنید. به مثال زیر توجه کنید:

```
<HTML>
<HCAD> <TITLE> Session Example </TITLE> </HEAD>
<BODY>
<%
Session.contexts (" Greeting") = " Welcome”
Response. Write (Session("  Greeting”))
%>
</BODY>
</HTML>
```

با نمایش این ASP، روی مرورگر، جملهٔ Welcome نمایش داده می‌شود. جملهٔ اول درون این اسکریپت متن Welcome را به متغییر Greeting از شیء Session نسبت می‌دهد. خط دوم، متغیر Greeting را روی صفحه نمایش می‌دهد. شما این کارها را می‌توانید با VBScript انجام دهید. به هر حال تصور کنید همان کاربر، صفحهٔ دیگری را درخواست کند. به عنوان مثال صفحهٔ زیر را درخواست کند.

```
<HTML>
```

```
<HEAD > <TITLE > Another page < / TITLE > < / HEAD>
<BODY>
<%=Session.contexts("Greeting”)%>
</BODY>
</HTML>
```

وقتی کاربر این صفحه را می‌بیند، همان Welcome دوباره روی صفحه نمایش داده می‌شود. به هر حال متغیر Session مقداری به این صفحه نسبت نمی‌دهد. این متغیر Greeting ازشیء Session است که، مقداری را که به صفحهٔ قبلی نسبت داده شده است، نگه می‌دارد.

شما نمی‌توانید این کار را با متغیرهای معمولی اسکریپت انجام دهید. یک متغیر معمولی فقط درون یک صفحه دارای اعتبار است. برخلاف آن متغیر Session تا زمانی که کاربر سایت وب را ترک نکند، باقی می‌ماند.

باید بدانید متغیر Sessionای را که به‌وجود آورده‌اید، فقط در رابطه با کاربری خاص است. متغیر Session ای که به یک کاربر نسبت داده‌اید نمی‌تواند روی متغیر Session کاربران دیگر اثر بگذارد. به عبارت دیگر، یک داده که درون متغیر Session ذخیره می‌شود، میان کاربران دیگر به اشتراک گذاشته نمی‌شود. به عنوان مثال، اسکریپت زیر را که در یک ASP آمده است، در نظر بگیرید.

```
<%
Randomize
If  INT (2 *RND) = 1 THEN
Session.contexts( "Favoritecolor") =" Blue”
ELSE
Session.contexts("Favoritecolor")= " Red”
END IF
%>
```

این اسکریپت به‌صورت اتفاقی، مقدار Blue یا Red را به Favoritecolor، که یک متغیر شیء Session است، نسبت می‌دهد. این متغیر با توجه به کاربران مختلف می‌تواند مقدارهای مختلفی را داشته باشد. مقدار متغیر Favoritecolor، فقط به Session یک کاربر خاص مربوط می‌شود.

محتوای یک Session

بیشتر متغیرهای Session، درون مجموعه‌ای از شیء Session به نام Contents ذخیره می‌شود. البته چون Contexts حالت پیش‌فرض است، می‌توان آن را ننوشت. در مثال زیر دو عبارت هم ارزش هستند.

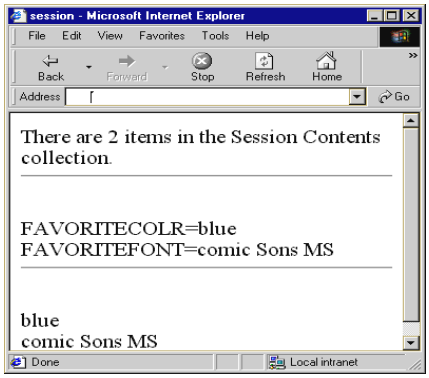
```
< % Session  ("myvar")= " Some data” %>
< % Session.Contents ("myvar ")= "Some data” % >
```

همان‌طوری که قبلاً در مجموعه‌ها بحث شد، می‌توانید از صفت count برای تعیین تعداد عناصر مجموعهٔ Contents استفاده کنید. همچنین با استفاده از حلقهٔ FOR... EACH , FOR...NEXT، می‌توانید تمام عناصر مجموعهٔ contents را نمایش دهید. مثال زیر از این متدها استفاده می‌کند.

```
<%
Session  (" Favoritecolor") =" blue”
Session  (" FavoriteFont")="  comic Sons”
% >
There are
<% =Session.Contents.Count %>
```

```
items in the Session  Contents collection
<HR>
<%
FOR EACH thiny IN Session.Contents
Response.write (" <BR>"  & thiny & " ="  & Session.contents (thiny))
NEXT
Response.write (" <HR >")
FOR i=l TO Session.Contents.count
Response.write(" <BR>" & Session.contents( i ))
NEXT
%>
```

در این اسکریپت، دو متغیر Session به نام‌های Favoritecolor , FavoriteFont به‌وجود آمده است. (شکل ۸- ٦)



شکل ۸-٦ محتویات مجموعهٔ contents

شناسایی یک Session

ASP، به هر کاربر، یک شناسهٔ Session منحصر به فرد نسبت می‌دهد. وقتی اولین بار یک کاربر وارد می‌شود و یک شناسه به آن اختصاص داده می‌شود ، این شناسه تا زمانی که او در سایت است می‌ماند. یک SessionID وقتی به وجود می‌آید، که کاربرجدیدی به وجود آید و بخواهد مدتی را در سایت بماند. برای به‌دست آوردن این SessionID، می‌توان از صفتSessionID از شیء Session استفاده کرد. مانند مثال زیر:

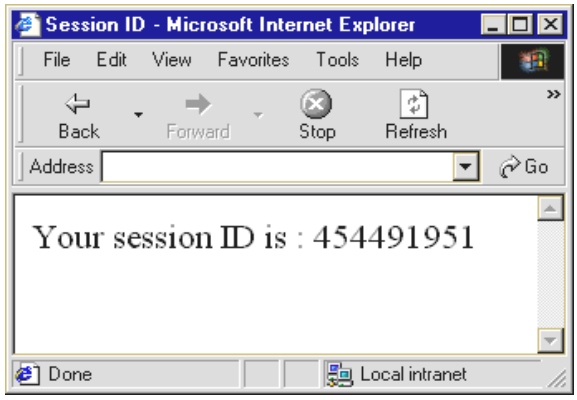
```
<HTML>
<HEAD > <TITLE > Session ID < / TITLE > < / HEAD>
<BODY>
Your session ID is :   < %= session.sessionID%>
</BODY>
</HTML>
```

ASP بالا به راحتی مقدار صفت SessionID را استخراج می‌کند. (رجوع به شکل بعدی). وقتی کاربران صفحه را دریافت می‌کنند، برای هر کدام SessionID متفاوتی نمایش داده خواهد شد. یکی از کاربردهای صفت SessionID، دنبال کردن حرکت یک ملاقات کننده است. به عنوان مثال می‌توانید صفحاتی را که یک

کاربر می‌بیند، درونlogfile سایت وب خود ذخیره کنید. برای این کار می‌توانید فایل زیر را بسازید و آن را درون صفحات خود استفاده کنید.

```
<%
who =Session.SessionID
Currentpage = Request.ServerVaribales (" SCRIPT_ NAME")
Response.AppendToLog (who&":"&Currentpage)
%>
```

در این اسکریپت با استفاده از متد AppendToLog از شیء Response، می‌توانید ورودی را به logfile سرویس دهنده اضافه کنید. در این مثال رشته‌ای که به فایل اضافه شده، یک SessionID می‌باشد که از صفت SessionID دریافت شده است. همچنین رشته شامل مسیر صفحهٔ جاری می‌باشد، که از متغیر محیطی SCRIPT-NAME به‌دست می‌آید. (شکل ۹- ٦)



شکل ۹-٦ مقدار sessionID

کنترل، هنگام پایان یافتن Session ها

چگونه یک سرویس‌دهنده، تشخیص می‌دهد، که یک Session پایان یافته است؟ به عبارت دیگر چگونه یک سرویس‌دهنده تشخیص می‌دهد، که یک کاربر، سایت وب شما را ترک کرده، یا دستگاه خود را خاموش کرده و رفته است. یک سرویس‌دهنده فرض را بر این می‌گذارد، که اگر کاربری بیش از ۲۰ دقیقه درخواستی نفرستاد و یا صفحهٔ خود را refresh نکرد، یعنی از سایت خارج شده است و زمان استفاده از Session برای آن تمام شده است. برای درخواست‌های سایت‌های وب خاص، این مدت زمانی بسیار کوتاه است. به عنوان مثال فرض کنید، شما یک سایت بازی دارید که شامل تعدادی سرگرمی است، که کاربر برای حل هرکدام نیاز به کاغذ و مداد دارد، بنابراین به مدت زمان بیشتری برای از دست دادن Session نیازدارید.

برای درخواست‌های بعضی دیگراز سایت‌های وب، مدت زمان ۲۰ دقیقه بسیار زیاد است. اگر شما یک سایت بسیار بزرگ داشته باشید و بخواهید بار روی سرویس‌دهنده را تا حد امکان کم کنید، مدت زمان کمتری برای خروج هر Session لازم دارید، بنابراین باید بیشترین زمان اختصاصی به هر Session را کنترل کنید. شیء Session دارای صفتی برای این منظور است. شما می‌توانید میزان اختصاص زمان

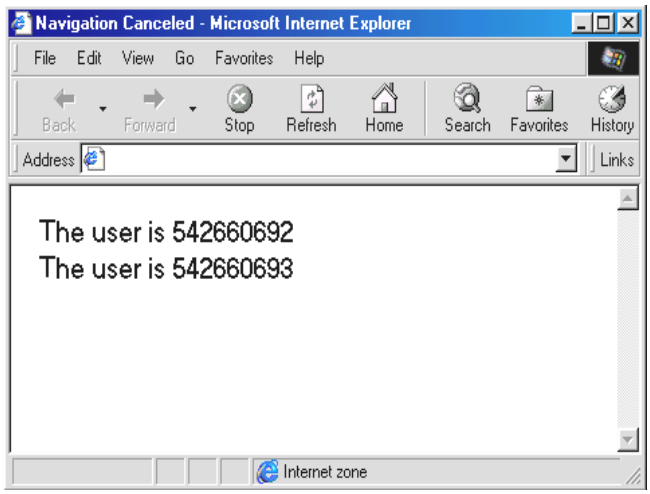
را قبل از اینکه زمان پایان یابد، با استفاده از صفت Timeout تنظیم کنید، به عنوان مثال اسکریپت زیر صفت Timeout را روی ۶۰ دقیقه تنظیم کرده است:

```
<%Session.Timeout =60% >
```

هنگامی که زمان یک Session پایان می‌یابد و کاربر تقاضایی می‌کند، سرویس‌دهنده مانند یک کاربر جدید با او رفتار می‌کند. سرویس‌دهنده یک Session جدید به وجود می‌آورد و تمام اطلاعات Session قبلی از دست می‌رود. شما می‌توانید با استفاده از متد Abandon از شیء Session این کار را انجام دهید. مثال زیر را در نظر بگیرید.

```
<HTML>
<HEAD > <TITLE >Abandon Session  </TITLE > </HEAD>
<BODY>
< BR> The   user   is   <%= session.sessionID%>
<% session.Abandon %>
< BR> The   user   is   <%= session.sessionID%>
</ BODY>
</HTML >
```

در این مثال، Session ID روی صفحه نمایش داده می‌شود. بعد از آن Session.Adandon صدا زده می‌شود. وقتی کاربرSessionID دوباره روی خروجی نمایش داده می‌شود، این ID شمارهٔ دیگری است. بعد از این که متد Abandon صدا زده می‌شود، سرویس‌دهنده با کاربر همانند یک کاربر جدید رفتار می‌کند. (رجوع به شکل زیر)



شکل ۱۰-۶ واگذاری یک session

رخدادهای Session

برخلاف دیگر اشیای مطرح شده، شیء Session چند رخداد نیز دارد. دوتای آنها عبارتند از Session_OnStart، که هنگام شروع Session رخ می‌دهد و Session_OneEND که هنگام پایان Session رخ می‌دهد. شما فقط یک اسکریپت را می‌توانید با هر یک از این رخدادها تلفیق کنید.

جملات درون اسکریپت وقتی که رخداد شروع شد، اجرا می‌شود. هردوی این اسکریپت‌ها، درون یک فایل مخصوص به نام Global.asa قرار می‌گیرند. هر تقاضا برای سایت وب می‌تواند یک فایل Global.asa داشته باشد. این فایل درون فهرست ریشهٔ سایت شما قرار دارد. فایل Global.asa دارای ساختار زیر است.

```
<SCRIPT  LANGUAGE =VBScrtpt    RUNAT= Server>
SUB  Application_Onstart
END SUB
</SCRIPT>
<SCRIPT   LANGUAGE = VBScript   RUNAT = Server>
SUB Application _onEnd
End SUB
</SCRIPT>
<SCRIPT  LANGUAGE = VBScript   RUNAT = Server>
SUB Session_Onstart
END SUB
</SCRIPT>
<SCRIPT  LANGUAGE = VBScript   RUNAT = Server>
SUB Session_On End
END SUB
</SCRIPT>
```

فایل Global.asa می‌تواند شامل چهار اسکریپت باشد. یکی از اسکریپت‌ها با رخداد Session_Onstart راه‌اندازی شده و یکی دیگر با رخداد Session_OnEnd.

توجه کنید در این فایل از برچسب <SCRIPT> به جای روش معمول، یعنی استفاده از کاراکترهای <٪و٪> استفاده می‌شود. دراین فایل از VBScript استفاده می‌کنیم، البته می‌توان از زبان‌های دیگر نیز استفاده کرد. در این فایل نمی‌توانید هیچ خروجی دیگری داشته باشید. به عنوان مثال نمی‌توانید از برچسب HTML یا متد ()Response.write استفاده کنید. فایل Global.asaخودش هیچ وقت نمایش داده نمی‌شود، این فایل فقط برای اشیاء و اسکریپت‌ها استفاده می‌شود. برای ایجاد اسکریپتی که هر وقت یک Session شروع می‌شود، اجرا شود، می‌توان به سادگی با اضافه کردن آن به قسمت Session_Start، از فایل Global.asa آن را انجام داد. همانند مثال زیر:

```
<SCRIPT  LANGUAGE = VBScript  RUNAT= Server>
SUB Session _Onstart
Session (" UserName") = " UnKnown"
Session (" UserPassword") = " UnKnown"
END SUB
</SCRIPT>
```

این اسکریپت مقدار "Un known" را به دو متغیر UserPassword و UserName، که متعلق به Session می‌باشند، نسبت می‌دهد. در این مثال یکی از توابع اصلی اسکریپت، یعنی Session_Onstart برای مقداردهی اولیه به متغیرهای Session ، تشریح شده است.

از Session_Onstart می‌توانیم برای منظورهای دیگری نیز استفاده کنیم. یکی از کاربردهای جالب Session_Onstart هدایت کاربر به صفحه‌ای جدید است. به عنوان مثال اگر بخواهید هنگامی که کاربری

وارد سایت شما می‌شود، ابتدا به صفحهٔ خانگی راهنمایی شود و نه صفحهٔ دیگری؛ می‌توانید با استفاده از متد Response.Redirect، این کار را انجام دهید. همانند مثال زیر:

```
<SCRIPT   LANGUAGE= VBScript  RUNAT = Server >
SUB  Session_Onstart
      MyHomepage="http://www.asp.com/homepage .asp"
      Requestpage  = Request.ServerVariables (" SCRIPT_ NAME")
      IF NOT (STRCOMP (MyHomepage,Requestpage)=0) THEN
          Response.Redirect (MyHomepage)
      END IF
END SUB
</SCRIPT>
```

در این اسکریپت، مسیر صفحهٔ در خواست شده با مسیر صفحهٔ خانگی مقایسه می‌شود. اگر همان صفحه نبود، به‌طور خودکار به صفحهٔ خانگی راهنمایی می‌شود. به یک مثال دیگر توجه کنید:

```
<SCRIPT   LANGUAGE=VBscript  RUNAT=Server >
SUB  Session_Onstart
      Response.AppendTolog (Session.Session ID&"starting")
END SUB
<SCRIPT/>
<SCRIPT   LANGUAGE= VBscript  RUNAT = Server >
SUB  Session_OnEnd
       Reponse.AppendTolog (Session.Session ID &"ending")
END SUB
</SCRIPT>
```

در این مثال از Session_Onstartو session_OnEnd برای ثبت sessionID یک کاربر درون logfile استفاده می‌شود، به‌خاطر اینکه Session_Onstart هنگامی اجرا می‌شود، که یک کاربر برای اولین بار وارد شود؛ این اسکریپت زمانی را که کاربر یک Session جدید را شروع کرد، ثبت می‌کند. Session_Onend نیز زمانی را که کاربر خارج می‌شود، ثبت می‌کند.

Session ها به چه صورت کار می‌کنند؟

Session ها از cookie ها استفاده می‌کنند. وقتی یک کاربر برای اولین بار تقاضای صفحه‌ای را از سایت شما می‌کند، سرویس‌دهنده روی مرورگر کاربر یک cookie برای دنبال کردن session به وجود می‌آورد. وقتی session پایان می‌یابد cookie نیز به خوبی پایان می‌یابد. cookieها برای هر کاربری، که ASPSESSIONID نامیده می‌شود، به وجود می‌آید. منظور از cookie به وجود آوردن یک شناسهٔ منحصربه‌فرد برای هر کاربر می‌باشد.

اگر شما بخواهید، ASP خود را refresh یا reload کنید، حداقل یک بار باید cookie را نمایش دهید. متغیرهای session، خودشان به تنهایی روی مرورگر کاربر ذخیره نمی‌شوند. به هر حال ASPSESSIONID cookie نیاز دارد که از متغیرهای session استفاده کند. یک سرویس‌دهنده از ASPSESSIONID cookie برای مرتبط کردن یک متغیر session خاص به یک کاربر خاص استفاده می‌کند. بدون cookie، یک سرویس‌دهنده راهی برای شناسایی یک کاربر که روی صفحات سایت وب حرکت می‌کند، ندارد. SessionID ذخیره شده درون ASPSESSIONID cookie به همان صورت صفت sessionID

نمی‌باشد. مایکروسافت از یک الگوریتم پیچیده برای تولید مقدار ASPSESSIONID cookie استفاده می‌کند.

چون session از cookie استفاده می‌کند، ممکن است با مرورگرهای قدیمی ناسازگار باشد. مرورگرهای قدیمی نمی‌توانند از cookie استفاده کنند، چون cookie با همهٔ مرورگرها سازگار نمی‌باشند و شما باید در استفاده از شیء session درون سایت وب خود محتاط باشید. البته بدون استفاده از session شاید خیلی از کارها را نتوانید انجام دهید، ولی می‌توان session را به روش دیگری شبیه‌سازی کرد.

نگه داشتن موقعیت بدون cookie

به‌کاربردن session و cookie خالی از دردسر نیست، زیرا همهٔ مرورگرها از آنها پشتیبانی نمی‌کنند. وقتی از cookie استفاده می‌کنید، از طرف مرورگرهای متفرقه، پیغامی مبتنی بر اینکه نمی‌تواند از Cookie ها استفاده کند، را دریافت می‌کنید. در ادامه شما یاد می‌گیرید که از متدهایی بدون نیاز به Cookie برای دریافت وضعیت استفاده کنید.

به‌دست آوردن موقعیت با استفاده از Querystring

شما می‌توانید یک Querystring به هر پیوندی در Asp ها اضافه کنید. با استفاده از Querystring می‌توانید اطلاعات را از یک صفحه به صفحهٔ دیگر بفرستید. همانند مثال زیر:

```
<HTML>
<HEAD> <TITEL> Query State <TITLE> </HEAD>
<BODY>
<%
UserName =Server.URLEncode("Bill  Gates”)
%>
A HREF ="http://www.asp.com/nextpage.asp ? <% = UserName %> ">Click >
<Here </A
</BODY>
</HTML>
```

این اسکریپت نام Bill Gates را به متغیری با نام UserName نسبت می‌دهد. مقدار این متغیر به صورت Query string در صورتی، که کاربر روی لینک کلیک کند، به صفحهٔ Nextpage.asp، فرستاده می‌شود. با به‌دست آوردن UserName از مجموعهٔ Query string می‌توانید آن را از صفحه‌ای به صفحهٔ دیگر بفرستید. به عنوان مثال، صفحهٔ Nextpage.asp به‌صورت زیر است:

```
<HTML>
<HEAD> <TITEL>Next page <TITLE> </HEAD>
<BODY>
<%
UserName =Server.URLEncode(Request.Querystring( "UserName”))
%>
A  HREF= "http://www.asp.com/nextpage .asp ? <%= User Name %>">Click >
<Here</A
</BODY>
</HTML>
```

مزیت این روش این است که با تمام مرورگرها کار می‌کند، اما خیلی پرزحمت است. اگر می‌خواهید یک کاربر را روی صفحات سایت وب خود دنبال کنید، باید با هر پیوند در سایت وب یک Querystring همراه کنید و هر Querystring باید شامل نام کاربر باشد.

ضرر این روش این است، که نمی‌توانیم اطلاعات با حجم بالا را بفرستیم، زیرا یک Querystring نمی‌تواند حجیم باشد. اگر یک Querystring بیشتر از 1024 کاراکتر باشد، اشکال به‌وجود می‌آید.

به‌دست آوردن وضعیت با استفاده از فیلدهای فرم مخفی

اگر شما می‌خواهید بدون استفاده از متغیرهای session، مقدار زیادی اطلاعات را از یک صفحه به صفحهٔ دیگری بفرستید، چاره‌ای جزء استفاده از فرم HTML ندارید. با استفاده از یک فیلد فرم مخفی، می‌توانید اطلاعات را همانند مثال زیر ارسال کنید.

```
<HTML>
<HEAD> <TITLE> Form State </TITLE ></HEAD>
<BODY>
<%
UserName ="Bill Gates”
%>
< FORM  METHOD = "POST" ACTION= "http://www.asp.com/nextpage.asp”>
<% INPUT NAME ="UserName "  Type ="HIDDEN "  VALUE =<%=UserName%>”>
<INPUT   TYPE ="SUBMET "  VALUE ="Nextpage”>
</FORM>
</BODY>
</HTML>
```

این صفحه شامل یک فرم HTML است. فرم، یک فیلد مخفی به نام UserName دارد، که حاوی مقدار متغییر Username است. فرم شامل یک دکمه نیز می‌باشد. هنگامی که روی دکمه کلیک می‌کنیم nextpage.asp فراخوانی می‌شود و داده‌ها در فیلد فرم مخفی به صفحهٔ جدید فرستاده می‌شود. شما می‌توانید داده‌ها را به این صورت ارسال کنید. در هر صفحه، از مجموعهٔ Form شیء Request برای دریافت داده‌ای فیلد مخفی استفاده کنید، بعد باید برای اینکه داده‌ها بتوانند به صفحهٔ جدید فرستاده شوند؛ یک فیلد مخفی جدید به وجود آورید. به مثال زیر توجه کنید:

```
<HTML>
<HEAD> <TITLE> Next page </TITLE ></HEAD>
<BODY>
<%
UserName =Request .Form("UserName”)
%>
< FORM  METHOD = "POST" ACTION = "http://www.asp.com/Nextpage.asp”>
<INPUT NAME ="UserName " Type ="HIDDEN” VALUE = " <%=User Name%>”>
<INPUT  TYPE ="SUBMET " VALUE ="Nextpage”>
</FORM>
</BODY>
</HTML>
```

روش‌های ترکیبی

هیچ یک از دو روش ذکر شده به تنهایی مناسب نمی‌باشند، هر چند که آنها تنها روش‌های جایگزین برای به‌دست آوردن موقعیت بدون استفاده از session و cookie می‌باشند. با استفاده از Querystring، به همراه فیلدهای مخفی فرم، با مرورگرها مشکلی نخواهید داشت.

اگر شما نیاز دارید کاربر روی تمام صفحات، وب خود را تعقیب کند، باید Querystring یا فیلد مخفی فرم را روی هر صفحه در سایت وب خود قرار دهید. شما می‌توانید این دو متد را برای به‌دست آوردن موقعیت با هم ترکیب کنید. برای مثال در بعضی از صفحات می‌توانید از Querystring و در بعضی صفحات دیگر از فیلد مخفی فرم استفاده کنید، اگر این کار را بکنید نیاز ندارید که هم مجموعه‌های form و هم Querystring را در هر صفحه بررسی کنید، اگر از متدRequest بدون تعریف مجموعه استفاده کنید، هر دو مجموعه به صورت خودکار بررسی می‌شود. به مثال زیر توجه کنید:

```
<HEAD> <TITLE> Next page </TITLE ></HEAD>
<BODY>
<%
 UserName= Request.Form("User name”)
%>
<FORM   METHOD = "POST" ACTION = "http://www.asp.com/Nextpage.asp” >
INPUT   NAME ="UserName "Type ="HIDDEN”>
<VALUE ="<%=UserName”%>
<INPUT  TYPE ="SUBMET"   VALUE ="Nextpage”>
</FORM>
<A  HREF ="http://www.asp.com/nextpage.asp ?<% =Server.URLEncode
(UserName) %>” Click Here </A>
</BODY>
</HTML>
```

در این مثال متغیر User Name بدون در نظر گرفتن اینکه نام کاربر با Querystring فرستاده شده است، یا با فیلد مخفی، فرم تخصیص می‌یابد. با صدا کردن Request("UserNme”) می‌توان مقدار UserName را از Querystring یا، Form دریافت کرد.

۴-۶-۶ شئ Application

شیء Application جهت ایجاد ناحیه‌های مشترک برای همهٔ کاربران یک سایت وب می تواند استفاده شود. از طریق این شیء می‌توان یک متغیر از نوع Application تولید کرد، که این متغیر برای همهٔ کسانی که به سایت وب شما متصل هستند، قابل اشتراک می‌باشد.

در این فصل شما طرز کار با Application را می‌آموزید. در بخش اول فصل، پیش‌زمینه‌ای دربارهٔ Application ها پیدا می‌کنید. سپس خواهید آموخت که از متدها، مجموعه‌ها و رخدادهای یک Application چگونه استفاده کنید و در آخر این فصل دو برنامه که در آنها از Application استفاده شده مورد بررسی قرار گرفته است، که یکی به شما چگونگی ساخت یک Chat چند کاربرهٔ ساده و دیگری طریقهٔ ایجاد یک Asp که به‌صورت Real time آمار استفاده از سایت وب شما را نمایش دهد، یاد می‌دهد.

در زیر لیستی از ویژگی‌های مشترک یک Application از ASP آمده:

- در Application ها داده را می‌توانید بین صفحات و در نتیجه بین چندین کاربر در سایت وب به اشتراک بگذارید.
- یک Application دارای رخدادهایی است، که می‌تواند اسکریپت‌های Application ها را راه‌اندازی کند. به عنوان نمونه، یک شیء می‌تواند بین تمام صفحات در برنامه‌های کاربردی به اشتراک گذاشته شود.
- Application های مجزا می‌توانند برای داشتن خصوصیات مختلف توسط Internet service Manager پیکره‌بندی شوند.
- Application مجزا می‌توانند به‌طور جداگانه روی قسمتی از حافظه که به خود آنها تعلق دارد، اجرا شوند. این بدان معنی است، که اگر یکی از Application ها از کار افتاد[1]، بقیه نباید از کار بیافتند.
- شما می‌توانید یک Application را بدون آنکه روی Application دیگر اثر بگذارد، متوقف کنید.

هنگامی که برای اولین بار ASP ها را نصب می‌کنید، تعدادی از Application ها به‌طور پیش‌فرض ایجاد می‌شوند. به‌طور مثال یک Application برای سایت وب پیش‌فرض شما به‌وجود می‌آید. گرچه شما می‌توانید بعداً با توجه به نیازتان، بسیاری از Application های دیگر را ایجاد کنید.

شیء Application، دارای مجموعه‌ها و متدها و رخدادهای زیر می‌باشد:

Collection:	
Contents	جهت ایجاد متغیر از نوع Application به‌کار می‌رود و خودش دارای متدهای Remove و Remove All می‌باشد.
Methods:	
Lock	جهت Lock کردن ناحیهٔ بحرانی ایجاد شده، از طریق متغیر Application به‌کار می‌رود.
Unlock	جهت Unlock کردن ناحیهٔ بحرانی ایجاد شده، از طریق متغیر Application به‌کار می‌رود.
Events :	
Application-OnEnd	هنگامی که اولین بازدید کننده به سایت وصل می‌شود، شروع می‌گردد.
Application-OnStart	هنگامی رخ می‌دهد، که سیستم Shutdown شده، یا Application، unload شود.

1. Proced

مقدمه‌ای برمتغیرهای Application

یک متغیرApplication شامل داده‌ای است، که تمام صفحات و تمام کاربرانی که به سایت متصل هستند می‌توانند به آن دسترسی داشته باشند. متغیرهای Application می‌توانند شامل هر نوع داده‌ای باشند، از جمله آرایه‌ها و اشیاء. یک متغیر Application از دو جهت با متغیرهای session تفاوت دارد:

۱. برخلاف متغیرهای session متغیرهای Application به cookieها وابسته نمی‌باشند و به عبارتی سرویس‌دهندهٔ وب در استفاده از متغیرهای Application، با مرورگرمشکلی ندارد.

۲. برخلاف متغیرهای session، یک متغیر Application می‌تواند بین کاربران به اشتراک گذاشته شود. یک داده می‌تواند توسط یک کاربر درون متغیر Application ها ذخیره شود و توسط کاربر دیگر خوانده شود.

برای مثال فرض کنید شما از یک متغیر Application برای ثبت تعداد دفعاتی که یک اعلام آگاهی کلیک شده، استفاده می‌کنید. بدین ترتیب هر وقت این اعلام کلیک می‌شود، اسکریپتی مشابه اسکریپت زیر اجرا می‌شود:

```
<%
NumClicks = Application.contents("BannerClicks")
NumClicks = Num Clicks + 1
Application.contents("BannerClicks") = NumClicks
%>
```

اسکریپت بالا به سادگی یکی یکی شمارهٔ ذخیره شده در BannerClicks را اضافه می‌کند، اما فرض کنید که دو کاربر، هم زمان روی یک آگهی کلیک کنند، یعنی یک اسکریپت همزمان برای دوکاربر باید اجرا شود. اگر چنین شود مقدار BannerClicks مقدار نادرستی خواهد بود، که ارزش ندارد، چرا که هر دو کاربر در آن واحد مقدار متغیر را اضافه کرده‌اند.

در زیر برخی از موارد استفاده از متغیرهای Application ،آورده شده است :

- یک متغیر Application می‌تواند برای نمایش اطلاعات زود گذر در صفحهٔ وب به‌کار رود.به عنوان مثال می‌توانید از متغیرهای Application برای نمایش نوع روزها یا برای تغییر دادن اخبار روزانه در هر صفحهٔ وب استفاده کنید.
- یک متغیر Application می‌تواند برای ثبت تعداد دفعاتی که یک اعلام آگهی[۱] در سایت وب شما مورد دسترسی واقع شده، استفاده شود.
- یک متغیر Application برای نگه داشتن اطلاعات به‌دست آمده از یک پایگاه داده استفاده می‌شود. برای مثال می‌توان لیستی از عناصر فروش را از پایگاه داده دریافت کرد و این لیست را توسط متغیرهای Application در چند صفحه نمایش داد.
- یک متغیر می‌تواند شامل تعداد ملاقات کنندگان سایت وب شما باشد.
- یک متغیر Application می‌تواند برای برقراری ارتباط بین کاربران سایت وب استفاده شود، به عنوان مثال می‌توانید با استفاده از متغیرهای Application، یک بازی چند نفره به وجود آورید.

۱. Banner Advertisement (همان لینکی است که اغلب در بالای صفحات وب قرار می‌دهند و بیشتر جنبهٔ تبلیغاتی دارد)

به وجود آوردن و خواندن متغیرهای Application

به وجود آوردن یک متغیر Application، بسیار ساده است. با قرار دادن نام متغیر درون شیء Application با استفاده از مجموعهٔ Contents، می‌توانید یک متغیر جدید به وجود آورید، مانند مثال زیر:

```
<HTML >
<HEAD> <TITLE> Application Example</TITLE> </HEAD>
<BODY>
<%
Application.contents("Greeting") ="Welcome! "
%>
<% =Application.contents("Greeting ") %>
</BODY>
</HTML>
```

در این مثال متغیر Greeting به وجود آمده و مقدار Welcome به آن نسبت داده شده است و در آخر محتوای متغیر روی مرورگر نمایش داده می‌شود. یک متغیر Application که به آن مقداری نسبت داده می‌شود، می‌تواند روی تمام صفحات سایت نمایش داده شود. به عنوان مثال صفحهٔ زیر نیز می‌تواند متغیر Greeting را نمایش دهد:

```
<HTML >
<HEAD> <TITLE> Another page</TITLE> </HEAD>
<BODY>
<% Application.contents("Greeting ") %>
</BODY>
</HTML>
```

باید توجه داشت که برخلاف متغیرهای session ،متغیرهایApplication بعد از ترک کاربر، از بین نمی‌روند. مقداری که به این متغیر نسبت داده شده است، تا زمانی که سرویس‌دهندهٔ وب shut down نشود، یا Application Unloade نشود باقی می‌ماند و اگر خوش شانس باشد روزها وحتی ماه‌ها باقی می‌ماند.

چون متغیرهای Application به‌طور خودکار بعد از ترک کاربر از بین نمی‌رود، باید مراقب باشید که آنها را بدون هدف ایجاد نکنید. البته از آن جایی که متغیرهای Application حافظه را اشغال می‌کنند، باید از آنها کم استفاده کرد.

بهتر است بدانید که متغیر Application مربوط به یک کاربرخاص نیست. اگر یک کاربر، درخواست یک صفحهٔ وب را بکند که به متغیر Application آن یک مقدار داده شده است و کاربر دیگری درخواست همان صفحه را با مقداری دیگری بکند، مقدار این متغیر برای هر دو صفحه عوض می‌شود. اسکریپت زیرا را در نظر بگیرید:

```
<%
Rondomize
If  INT(2* RND )=1        THEN
  Application.contents("Favoritecolor")= "Blue"
ELSE
Application.contents ("Favoritecolor") ="Red"
END IF
%>
```

این اسکریپت به صورت تصادفی به متغیر Application ای به‌نام Favoritecolor مقدار Blue یا Red را نسبت می‌دهد. به هر حال مقدار متغیر برای هر دوکاربر یکی می‌شود، این کار می‌تواند باعث مشکل شود. از آن جایی که هر دو کاربر در آن واحد می‌توانند به متغیر Application دسترسی داشته باشند، باید تداخل‌ها را از بین ببرد.

خوشبختانه شیء Application، دارای دو متد است، که می‌توانند در تصحیح این وضع به ما کمک کنند. دو متد Lock و Unlock وقتی کاربر متغیری را تغییر می‌دهد به‌طور موقت می‌توانند مانع شوند، که کاربر دیگر آن را تغییر دهد. به مثال زیر توجه کنید:

```
%>
 Application.Lock
NumClicks= Application.contents("BannerClick")
1+NumClicks =NumClicks
Application.contents("BannerClicks ")= NumClicks
Application.Unlock
%>
```

خط اول این اسکریپت، تمام متغیرهای شیء Application را قفل می‌کند. وقتی متغیرها قفل می‌شوند، تا وقتی باز نشوند، کاربران دیگر نمی‌توانند این متغیرها را تغییر دهند. متغیرهای Application، تازمانی که متد Unlock صدا زده شود، یا انتهای صفحه دریافت شود، قفل می‌ماند.

توجه داشته باشید که شما نمی‌توانید متغیرهای Application را به صورت انتخابی قفل نمایید یا همه را انتخاب کنید، یا هیچ کدام را؛ مثلاً اسکریپت قبل به‌طور موقت کاربران را از تغییر همهٔ متغیرهای Application ای که ممکن است وجود داشته باشد، بازمی‌دارد. مهم است بدانید که قفل کردن متغیرهای Application مانع تغییر متغیرها توسط کاربران به‌صورت دائمی نمی‌شود و فقط قفل کردن تغییرات را منظم می‌کند.

ذخیره کردن متغیرهای Application

بیشتر متغیرهای Application از طریق مجموعهٔ Contents از شیء Application ایجاد می‌شوند. وقتی یک متغیر Application جدید به وجود می‌آورید، یک عنصر جدید به مجموعه اضافه می‌شود، در مثال زیر دو عبارت معادل هم هستند.

```
<%  Application ("Favoritecolor") =" Blue"%>
<% Application.Contents ("Favoritecolor") ="Blue"%>
```

از آن جایی که متغیرهای Application، درون مجموعه ذخیره می‌شوند، می‌توانید هر نوع تغییری را با استفاده از متدها روی آنها انجام دهید. با استفاده از متد count می‌توانید تعداد متغیرهای Application را به‌دست آورید و با استفاده از حلقه‌های FOR...EACH و FOR...NEXT می‌توانید تمام عناصر مجموعهٔ Contents را نمایش دهید، که در مثال زیر از حلقهٔ FOR...EACH استفاده شده است.

```
<%
FOR  EACH    thing   IN Application.Contents
Response.Write("<BR> " &  Thing &"="& Application.Contents (thing ))
NEXT
%>
```

این اسکریپت با قرار دادن مجموعهٔ Contents Application، در حلقه، تمام متغیرهای Application را نمایش می‌دهد.

رخدادهای Application

مانند رخدادهای **Session**، **Application** نیز دارای دو رخداد است. **Application_Onstart** و **Application _OnEnd** اولین رخداد وقتی به وجود می‌آید، که یکApplication ازASP شروع شود و دیگری وقتی اتفاق می‌افتد، که آن یکی پایان یابد.

چه وقت یک Application شروع می‌شود؟ باید گفت یک Application شروع نمی‌شود، مگر اینکه اولین صفحهٔ آن در خواست شود. یک رخداد Application_Onstart همیشه قبل از session_Onstart به وجود می‌آید.

برخلاف رخداد Session_Onstart رخداد Application_Onstart، هنگامی که کاربری جدید از Application درخواست یک صفحه می‌کند، شروع نمی‌شود. رخداد Application_Onstart وقتی که اولین بازدید کننده به سایت وصل می‌شود، شروع می‌شود.

Application_OnEnd وقتی رخ می‌دهد که سرویس‌دهندهٔ وب shut down شود، یا Application،unloade شود. یک رخداد Application_onEnd بعد از آخرین رخداد OnEnd _session به وقوع می‌پیوندد. به‌طور مثال این رخداد پس از اینکه با ISM سرویس‌دهندهٔ وب را خاموش می‌کنید، به وقوع می‌پیوندد.

Application_onstart و Application_onEnd فقط یک اسکریپت را راه‌اندازی می‌کنند. هر دو اسکریپت باید درون فایل Global.asa قرار بگیرند. این اسکریپت‌های خاص نمی‌توانند توسط صفحات دیگر صدا زده شوند.

فایل Global.asa یک فایل مخصوص است، که در فهرست ریشهٔ Application قرار دارد. هر Application فقط یکی از این فایل‌ها را دارد. فایل Global.asa شامل تمام اسکریپت‌ها واشیایی است، که به صورت سراسری در یک Application قرار دارند. این فایل دارای ساختار زیراست:

```
<SCRIPT  LANGUAGE= VBScript      RUNAT = Server>
SUB Application_Onstart
END    SUB
</SCRIPT>

<SCRIPT       LANGUAGE = VBScript       RUNAT= Server>
SUB Application_OnEnd
</SCRIPT>
<SCRIPT      LANGUAGE =VBScript          RUNAT= Server>
SUB Session_Onstart
END SUB
</SCRTPT>

<SCRIPT      LANGUAGE =VBScript          RUNAT= Server>
SUB Session_OnEnd
END SUB
</SCRIPT>
```

آنچه را که می‌توانید در اسکریپت‌های Application_Onstart و Application_OnEnd بگنجانید، بسیار محدود است. در این اسکریپت‌ها هیچ عبارت دیگری که مقدار خروجی داشته باشد را نمی‌توانید قرار دهید. به عنوان مثال از HTML یا متد Response.Write نمی‌توانید استفاده کنید. علاوه براین باید در استفاده از اشیایی، که به همراه اسکریپت‌های Application_Onstart و Application-OnEnd به‌کار برده‌اید، محتاط باشید.

اسکریپت Application_Onstart برای مقدار دهی اولیهٔ متغیرهای محدودهٔ Application ، به‌کار می‌رود. به عنوان مثال یک کاربرد عمومی session_Onstart و Application_Onstart دنبال کردن تعداد کل ملاقات کنندگان از زمانی که یک Application شروع شده است، می‌باشد. مثال زیر چگونگی انجام این کار را نشان می‌دهد.

```
<SCRIPT  LANGUAGE= VBScript      RUNAT = Server>
SUB Application _Onstart
Applicationn ("TotalUsers")=0
END    SUB
</SCRIPT>
<SCRIPT       LANGUAGE = VBScript       RUNAT= Server>
SUB session_Onstart
Application.lock
Application.unlock("totalUseres")= Applicationn ("totalUsers")+1
END    SUB
</SCRIPT>
```

عبارت تکی اضافه شده به اسکریپت Applicationn_Onstart متغییری به نام TotalUsers مقدار اولیهٔ صفر می‌دهد. این اسکریپت تنها درشروع‌کار سرویس‌دهندهٔ وب اجرا می‌شود. اسکریپت session_Onstart هنگامی که یک کاربر جدید وارد سایت می‌شود، مقدار TotalUser را یکی اضافه می‌کند. توجه داشته باشید که یک متغیر Application قبل از هر تغییری قفل می‌شود. این مسئله، از ناسازگاری‌هایی، که با رسیدن چند کاربر رخ می‌دهد، جلوگیری می‌کند. بعد از اینکه فایل Global.asa را تغییر دادید، می‌توانید تعداد کاربران را توسط یک صفحهٔ ASP با اضافه کردن خط زیر نمایش دهید.

```
<% = Applicationn ("Total Users")%>
```

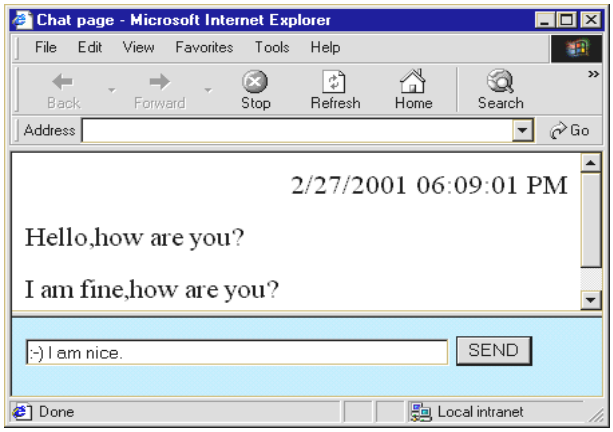
مثال‌هایی از کاربرد Application

صفحهٔ Chat

برای برقراری ارتباط بین دو کاربر برروی سایت وب، می‌توانید از صفحهٔ chat استفاده کنید. تمام کاربرانی که به صفحهٔ chat احتیاج دارند، می توانند پیغام‌های رسیده را ببینید. همچنین خودشان پیغام بفرستند. (رجوع شود به شکل بعد) برای به وجود آوردن صفحهٔ chat، باید سه فایل به وجود آیند یا تغییر کنند، که این سه فایل عبارتند از:

۱. صفحهٔ chat : این صفحه دو قاب دارد. فریم بالایی برای پیغام‌های کاربران دیگر و فریم پایینی برای قرار دادن پیغام خود کاربر می‌باشد.

۲. صفحهٔ نمایش Display : این صفحه کل پیغام‌های رسیده توسط کاربران دیگر را نمایش می‌دهد. با رسیدن هر پیغام جدید این صفحه به هنگام می‌شود.
۳. صفحهٔ Message : این صفحه به کاربر اجازه خواهد داد، پیغام جدید خود را قراردهد و شامل یک جعبهٔ ورودی متنی می‌باشد.
٤. فایل Global.asa که در آن یک اسکریپت در Application_Onstart وجود دارد. (شکل ۱۱- ٦)



شکل ۱۱-٦ صفحهٔ Chat

به‌وجود آوردن یک صفحهٔ Chat

اولین صفحه‌ای که لازم است به وجود آورید، صفحهٔ Chat است. این صفحه فقط وظیفهٔ جای دادن دو صفحهٔ دیگر را در خود دارد. از آن جایی که این صفحه حاوی هیچ اسکریپتی نمی‌باشد، باید آن را به نام Chatpage.htm به طوری که در لیست زیر آمده، ذخیره کنید:

```
<HTML>
<HEAD> <TITLE> Chat page  </TITLE> </HEAD>
<FRAMESET ROWS= "*;%10”>
<FRAME SRC=" Display.asp">
<FRAME SRC ="Meseage.asp">
</FRAMESET>
</HTML>
```

فریم‌ها را به این دلیل که صفحهٔ نمایش هر پنج ثانیه یک بار تجدید[1] می‌شود، به‌وجود می‌آوریم. این کار برای زمان‌هایی است، که نیاز باشد یک پیغام جدید به یک صفحهٔ جدید وارد کرد و از طرفی پیغام کاربر هنوز نیمه کاره باشد و نباید تجدید بشود.

1. Refresh

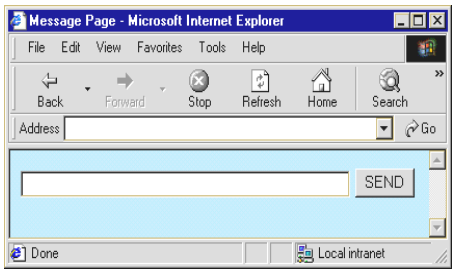
تغییر فایل Global.asa

برای اینکه صفحهٔ Chat به کار بیافتد، فایل Global.asa باید تغییر داده شود. اسکریپت‌های زیر متغیرهای Application های موردنیاز برای صفحهٔ Chat را مقداردهی می‌کند. متغیرها باید از نوع متغیرهای Application باشند، تا تمام کاربران بتوانند به آنها دسترسی داشته باشند. اولین متغیر Talk می‌باشد. این متغیرآرایه‌ای است، که تمام پیغام‌ها را نگه می‌دارد. آرایهٔ Talk به‌وسیلهٔ تخصیص TempArray به آن ایجاد می‌شود. دومین متغیر Tplace (Talk place) نام دارد و برای اشاره کردن به آخرین پیغام در آرایهٔ Talk به‌کار می‌رود. اسکریپت زیر به متغیر مقدار صفر می‌دهد:

```
<SCRIPT   LANGUAGE =VBScript      RUNAT= Server>
SUB      Application _Onstart
Dim     TempArray(5)
Application ("Talk  ") =TempArry
Application ("Tplace")=0
END    SUB
</SCRIPT>
```

به‌وجود آوردن صفحهٔ Message

منظور از به‌وجود آوردن صفحهٔ Message ، دادن امکان به کاربران برای وارد کردن پیغام جدید است. (مانند شکل زیر) این صفحه شامل یک فرم HTML است، که درون آن نیز از جعبهٔ ورودی متن[1] و دکمهٔ submit استفاده شده است. هنگامی که روی کلید submit، کلیک می‌شود، صفحه خودش را reload می‌کند. (شکل ۱۲- ٦)



شکل ۱۲-٦ صفحهٔ Message

در شکل بعد، اسکریپت دو کار را انجام می‌دهد. اول چک می‌کند، که آیا بیشتر از پنج پیغام وجود دارد یا نه، اگر بیشتر از پنج پیغام وجود داشته باشد، دوباره به متغیر Tplace، مقدار صفر داده می‌شود. این کار از سرریزشدن آرایه جلوگیری می‌کند، سپس اسکریپت یک پیغام تازه را به آرایهٔ Talk اضافه می‌کند و مقدار Tplace را یکی افزایش می‌دهد. Tplace معمولاً به مکان بعدی که پیغام باید درون آن قرار بگیرد، اشاره می‌کند. لیست زیر محتوای صفحهٔ Message.asp را نمایش می‌دهد.

```
<%
IF    NOT    Request.Form ("message")= "THEN"
```

1. Inpnt Box

```
Application.Lock
IF      Appication  ("Tplace ")>4   THEN
         Application("Tplace ")=0
END    IF
 TempArray = Application("Talk")
 TempArray = Application("TPlace") = Request.Form("Message")
 Application("Talk ")=Temparray
 Application("TPlace")=Application("TPLACE")+1
       Application.Unlock
END IF
 %>
<HTML>
<HEAD > <TITLE>  Message   Page   </TITLE >   </HEAD>
<BODY   BGCOLOR ="LIGHTBLUE">
<FORM     METHOD ="POST"    ACTION= "message.asp">
<INPUT    NAME=”MESSAGE”   TYPE=”TEXT” SIZE=50>
<INPUT    TYPE = "SUBMIT"   VALUE= "SEND">
</FORM>
</BODY>
</HTML>
```

به‌وجود آوردن Display

آخرین صفحه‌ای که باید به وجود بیاید، Display است. این صفحه است که پیغام همهٔ کاربران را نمایش می‌دهد. (رجوع شود به شکل ١٣- ٦)

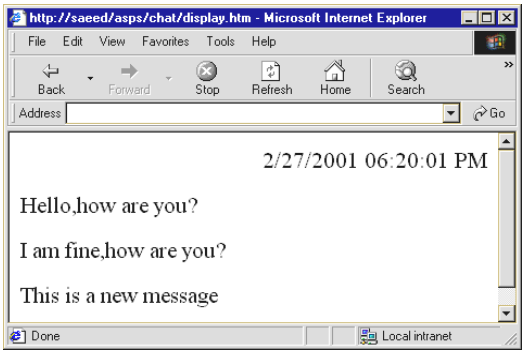
این صفحه به‌طور خودکار هر پنج دقیقه یک بار خودش را تجدید می‌کند. این کار را با استفاده از تگ <META> انجام می‌دهد. برچسب <META> شامل دستوراتی است، که این کار را انجام می‌دهد.

ابتدای اسکریپت صفحه زیر برای مشخص کردن صفحهٔ جاری به‌کار می‌رود. از مجموعهٔ ServerVariabels ، URL کامل صفحهٔ جاری را به‌دست آوریم وآن را به متغیر MySelf نسبت می‌دهیم. متغیر Myself نیز توسط برچسب <META>برای تعیین صفحه‌ای، که باید تجدید بشود به‌کار می‌رود.

اسکریپت اصلی برای نمایش محتوای آرایهٔ Talk استفاده می‌شود. با استفاده از حلقهٔ FOR...NEXT تمام پیغام‌های جاری نمایش داده می‌شود. لیست زیر، صفحهٔ Display.asp را نشان می‌دهد.

```
<%
 MyServer =Request.ServerVariables ("SERVER _ NAME ")
Mypath = Request.ServerVariables ("SCRIPT_ NAME ")
Myself = "HTTP:// "& Myserver&Mypath
%>
<HTML>
<HEAD >
<META   HTTP.EQUIV= REFRESH    CONTENT="5;<%=MySelf%>”>
<TITLE > Display    page    </TITLE>
</HEAD>
<BODY>
<P  ALIGN = RIGHT><%= NOW%> </P>
<%
TempArray =Application ("Talk")
FOR    i=0   TO    Application("Tplace")-1
 Response.Write  ("<P>"& Temparray (1))
NEXT
%>
```

```
</BODY>
</HTML>
```



شکل ۱۳-۶ صفحهٔ Display

طرح توسعهٔ صفحهٔ Chat

راه‌هایی برای تقویت صفحهٔ chat وجود دارد. به عنوان مثال بیشترین پیغامی، که صفحهٔ Chat می‌تواند در آنِ واحد نشان دهد، پنج پیغام است که این برای استفاده‌های بالا کافی نمی‌باشد.

می‌توانید حداکثر تعداد پیغام‌ها را با تغییر اندازهٔ TempArray در فایل Global.asa و تغییر مقداردهی مجدد TPlace در صفحهٔ Message ، تغییر دهید.

صفحهٔ chat به شما اجازه می‌دهد، پیغام‌هایی را که شامل فرمت HTML هستند، وارد کنید. به هرحال تغییر صفحهٔ Message به‌منظور ساده‌تر کردن آن مشکل نخواهد بود. به عنوان مثال می‌توانید check Box هایی برای قالب‌بندی پیغام داشته باشید، که به شما اجازهٔ تعریف فونت و رنگ پیغام را می‌دهد.

در آخر، صفحهٔ chat نمی‌تواند نام کاربر را همراه پیغام کند، رفع این عیب کار مشکلی نمی‌باشد. می‌توانید یک صفحهٔ logon را قبل از صفحهٔ chat قرار دهید، تا نام کاربر را دریافت کند. سپس این نام می‌تواند در ابتدا، همراه تمام پیغام‌هایی، که توسط کاربر فرستاده می‌شود، قرار گیرد.

صفحهٔ whoson

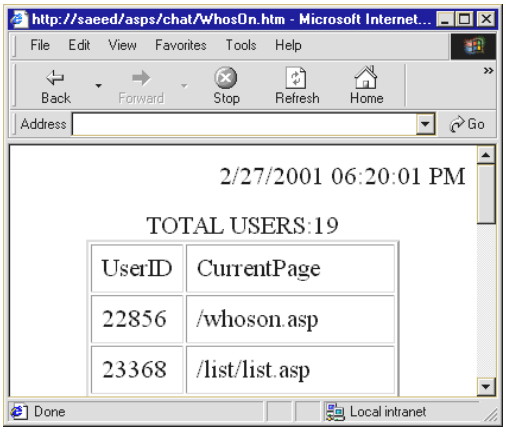
مدیران خوب وب، یک هدف در زندگی دارند؛ آنها می‌خواهد کاربران سایت آنها بیشتر شود. بخشی از این وسواس، مشخص کنندهٔ تعداد کاربرانی که روی خط می‌باشند است وتعیین می‌کند که آنها چه انجام می‌دهند. شما بلافاصله می‌توانید هر زمانی که بخواهید، شمار بازدید کنندهٔ سایت وب خود را تعیین کنید. همچنین می‌توانید بفهمید آخرین صفحه‌ای که توسط کاربر درخواست شده است، چه بوده است؟ (رجوع به شکل ۱۴-۶)

این طرح توضیح می‌دهد، که چگونه یک شیء را به یک متغیر Application نسبت دهیم و چگونه یک شیء Dictionary برای ذخیرهٔ اطلاعات کاربران مورد استفاده قرارگیرد. وقتی یک کاربر تقاضای یک صفحه را می‌کند، اطلاعات درون شیء Dictionary به روز در آورده می‌شود. برای این طرح باید فایل-های زیر به وجود آمده، یا تغییر داده شوند.

فایل Global.asa : برای این طرح هم اسکریپت Application_Onstart و هم اسکریپت Ses-sion_OnEnd باید تغییر داده شوند.

فایل Grabstats : این فایل شیء Dictionary را به‌روز می‌کند. شما باید این فایل را همراه همهٔ صفحاتی که می‌خواهید آنهارا دنبال کنید، بیاورید.

صفحهٔ WhosOn : این صفحه کاربران فعلی را روی سایت وب نشان می‌دهد.



شکل ۱۴-۶ صفحهٔ WhosOn

تغییر فایلGlobal .asa

برای به وجود آوردن این طرح، باید اسکریپتی که درون فایل Global.asa قرار دارد، تغییر دهید. اول باید شیء Dictionary را به وجود آورید، که اطلاعات کاربران را ذخیره می‌کند. (طریقهٔ ایجاد اشیای جدید در Asp را در بخش بعد شرح خواهیم داد). از آن جایی که این شیء باید فقط یک بار به وجود آید، این کار را درون Application _Onstart انجام می‌دهیم:

```
<SCRIPT   LANGUAGE=VBScript    RUNAT= Server>
SUB    Application_Onstart
Set  Applicationn("stats")=Server.CreateObject("Scripting.Dictionary")
END    SUB
<SCRIPT>
```

یک خط به اسکریپت قبلی اضافه شده است. عبارتی که یک نمونه از شیء Dictionary را به متغیر Application نسبت می‌دهد، Stats نامیده می‌شود. این متغیر به محض ایجاد می‌تواند در Application شما مورد استفاده قرار گیرد. اسکریپت Session_OnEnd درون فایل Global.asa باید تغییر داده شود. هدف از اسکریپت زیر، برداشتن کاربر از شیء Dictionary هنگام خاتمهٔ session کاربر، است.

```
<SCRIPT   LANGUAGE  = VbScript   RUNAT   = Server>
SUB     Session _OnEnd
IF   Application("stats").Exists(session.sessionID)   THEN
Applicationn .Lock
Application("stats").Remove  (session.sessionID)
Application.unlock
END   IF
```

```
END   SUB
</SCRIPT>
```

کاربران می‌توانند با sessionID هایشان دنبال شوند. کلیدهایی درون Dictionary با نام stats مطابق این sessionID ها می‌باشند. اسکریپت بالا بررسی می‌کند، که آیا sessionID مربوط به کاربر فعلی Dictionary وجود دارد، یا نه و درصورت وجود، آن را حذف می‌کند.

به‌وجود آوردن فایل Grabstats

برای تعیین صفحه‌ای که کاربر روی آن قرار دارد، باید دستورات زیر را روی هر صفحه‌ای که می‌خواهید دنبال کنید، قرار دهید. این دستورات شامل یک اسکریپت تک خطی می‌باشد، که در زیر نشان داده شده است.

```
<%
Application("stats") .item (session . session ID)=
Request .ServerVariables ("SCRIPT_NAME")
%>
```

این اسکریپت، مسیر هر صفحه را به شیء Dictionary اضافه می‌کند. با استفاده از متغیر محیطی سرویس‌دهندهٔ SCRIPT_NAME مسیر صفحه جاری را به‌دست می‌آوریم. بعد، مقدار این متغیر را به یک کلید Dictionary که مطابق با کاربرفعلی است، نسبت می‌دهیم. (اگر این کلید وجود نداشته باشد، به صورت خودکار به‌وجود می‌آید).

این فایل را با نام Grabstats.asp ذخیره می‌کنیم و آن را با هرصفحه‌ای که می‌خواهیم دنبال شود، همراه می‌کنیم. به راحتی می‌توانید این خط را در بالای ASP اضافه کنید.

```
<!-- # INCLUDE   VIRTUAL = "Grabstats.asp" -->
```

به‌وجود آوردن صفحهٔ whosOn

صفحهٔ whosOn برای نمایش کاربران جاری استفاده می‌شود. sessionID هر کاربر پس از آخرین صفحه‌ای که کاربر درخواست کرده، نمایش داده می‌شود. در زیر، صفحهٔ whoson.asp نشان داده شده است.

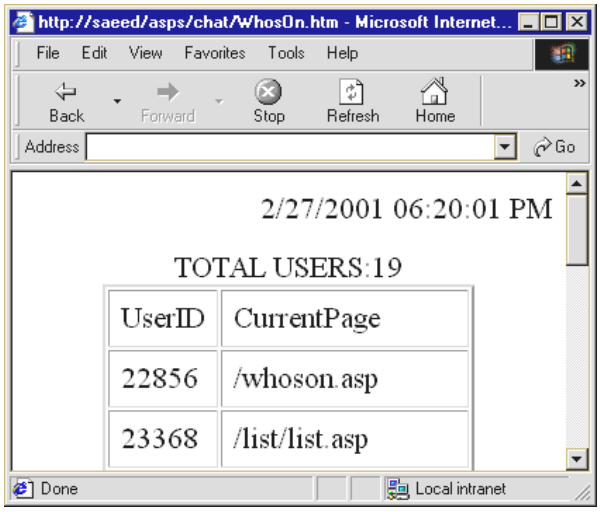
```
<!--#  INCLUDE   VIRTUAL = "Grabstats.asp" -->
<%
MyServer =Request.ServerVariables ("SERVER _NAME")
Myself ="HTTP =//"&Myserver&Mypath
%>
<HTML>
<HEAD>
<META   HTTP.EQUIV="REFRESH" CONTENT="20;<%=Myself %>">
<TITLE >WHOSON</TITLE>
</HEAD>
</BODY>
<% Application.Lock
Set  TempStats = Application("stats")
Application.unlock
%>
<CENTER>
<B> TOTAL    USERS:</B>    <%=TempStats.Counts%>
```

```
<TABLE    BORDER =1    CELLPADDING=10>
<TR> <TH> User   ID    </TH><TH>  Current   Page</TH></TR>
<% TempItems=Tempststs.items
Tempkeys = TempStats.keys
For   i=0   to    UBOUND(TEMPKEYS)
%>
<TR><TD><%= Tempkeys ( i) %> </TD><TD><%=TempItems(i)%></TD>
     </TR>
<%
NEXT
%>
</CENTER>
</TABLE>
</BODY>
</HTML>
```

خط اول، شامل فایل Grabststs.asp است. این کار به شماره اجازه می‌دهد، که بدانید چه وقت این صفحه را ببینید. مسیر صفحهٔ whosOn بعداً در sessionID شما ظاهر می‌شود.

اولین اسکریپت، مسیر صفحهٔ جاری را دریافت می‌کند. صفحهٔ whoson با استفاده از client_pull هر ۲۰ ثانیه به‌صورت خودکار تجدید می‌شود. برای تجدید کردن این صفحهٔ مسیر آن را لازم دارید. اسکریپت دوم، دیکشنری ذخیره شده در متغیر Application را در شیء موقتی Dictionary که Temp Stats نام دارد، ذخیره می‌کند. Temp Statsبه‌طور خودکار در زمان خاتمهٔ صفحه از بین می‌رود.

با استفاده از شمارش عناصر در dictionary، می‌توانید تعداد جاری کاربران را به‌دست آورید. اسکریپت آخر با قرار دادن کلیدها و عناصرش در یک حلقه، تمام مقادیر ورودی به dictionary را نمایش می‌دهد. (شکل ۱۵- ۶)



شکل ۱۵-۶ کاربران و آخرین صفحهٔ مورد دسترسی

توسعهٔ صفحهٔ WhosOn

صفحهٔ WhosOn اطلاعات تغییرپذیری را دربارهٔ کاربران سایت وب شما را فراهم می‌کند. راه‌های زیادی وجود دارد، که می‌توان توسط آن پروژهٔ قبل را توسعه داد. برای مثال اگر سایت وب شما نیاز به ثبت داشته باشد، شما می‌توانید اسامی کاربران را متناسب با SessionID ها نمایش دهید. برای این کار از اسامی کاربران همچون کلیدی در دیکشنری استفاده کنید.

به علاوه، فهمیدن اینکه هر کاربر چه مدت Online بوده، مفید خواهد بود. شما می‌توانید پروژه را به روش‌های مختلفی برای دنبال کردن این اطلاعات، تغییر دهید. به‌طور نمونه می‌توانید به سادگی متغیر Application ثانویه‌ای ایجاد کنید، که حاوی دیکشنری دیگری باشد. در این حالت می‌توانید طول مدت بازدید کاربر از سایت را با ذخیرهٔ این اطلاعات در این دیکشنری ثانویه به‌دست آورید.

۵-۶-۶ شیء Server

این شیء جهت مقداردهی بعضی از متغیرهای سرویس‌دهنده یا ایجاد اشیاء جدید روی سرویس‌دهنده می‌تواند استفاده شود. این شیء دارای متدها و متغیرهای زیر می‌باشد.

Properties:	
Scripttimeout	جهت تعیین کردن زمان Timeout برنامه‌های Asp از آن استفاده می‌شود.
Methods:	
CreatObject	جهت ایجاد یک شیء جدید از اجزای Activex روی سرویس‌دهنده می‌توان از آن استفاده کرد.
Execute	از طریق این متد می‌توان یک برنامهٔ Asp دیگر را داخل برنامه اجرا کرد.
GetLastError	آخرین Error روی سرویس‌دهنده را برمی‌گرداند.
HTMLEncode	این متد برچسب‌های HTML را به کدهای کاراکتر HTML ترجمه می‌کند. جهت ارسال متنی به مرورگر که آن را HTML تفسیر نکند، به‌کار می‌رود.
MapPath	جهت تبدیل مسیر مجازی به فیزیکی روی سرویس‌دهنده به‌کار می‌رود.
Transfer	جهت انتقال کنترل به صفحهٔ Asp دیگر به‌کار می‌رود. وقتی این متد فراخوانی گردد، تمام داده‌های مربوط به صفحهٔ فراخوانده شده، به صفحهٔ جدید منتقل می‌شوند.
URLEncode	از این متد جهت کد کردن URL و QueryString می‌توان استفاده کرد.

استفاده از اجزای[1] ASP

در فصل‌های پیشین چگونگی استفاده از اشیاء Asp مثل Request و Response شرح داده شده است. اجزای ASP به این اشیاء بسیار شبیه می‌باشند. با اینکه یک جزء آن چنان با ASP پیوندی ندارند، لیکن اجزاء یکی از بهترین گزینه‌ها برای توسعهٔ متدها و توابع فراهم آمده به‌وسیلهٔ اشیای تعبیه شده می‌باشند. شما می‌توانید با استفاده از زبان‌هایی مثل C++ و جاوا و ویژوال بیسیک، یا دلفی برای خودتان اجزایی را ایجاد کنید. همچنین می‌توانید اجزاء را از شرکت‌های نرم‌افزاری خریداری کنید. به غیر از اینها مایکروسافت، نیز تعدادی از اجزای اکتیوایکس را با ASP همراه کرده است. قبل از آنکه بتوانید از یک جزء استفاده کنید، اول باید نمونه‌ای از آن را ایجاد کنید. با این کار شما می‌توانید به خصوصیات، مجموعه‌ها و متدهای یک شیء از آن جزء دسترسی داشته باشید.

به وجود آوردن یک جزء با محدودهٔ عمل صفحه

در بیشتر موارد یک نمونهٔ جزء را با محدودهٔ عمل صفحه به وجود خواهید آورد. یک جزء با محدودهٔ عمل صفحه روی یک صفحهٔ تکی به وجود می‌آید و وقتی که پردازش صفحه پایان می‌یابد، آن نیز از بین می‌رود. شما نمی‌توانید یک جزء با محدودهٔ عمل صفحه را روی صفحهٔ دیگری که به‌طور صریح ایجاد نشده، استفاده کنید. برای ایجاد نمونه‌ای از یک جزء با محدودهٔ عمل صفحه باید از متد Server.Createobject ( ) استفاده کنید.

اینجا یک مثال از ایجاد یک جزء با محدودهٔ عمل صفحه وجود دارد:

```
<%
Set MyBrow=Server.CreateObject ("MSWC.BrowserType”)
%>
```

این اسکریپت یک نمونه‌ای از جزءِ امکانات مرورگر را ایجاد می‌کند. این اسکریپت متغیر MyBrow را به یک نمونهٔ این جزء نسبت می‌دهد. قابل توجه است که از عبارت set از VBScipt استفاده شده است. به علت اینکه یک نمونه از جزء رابه یک متغیر نسبت می‌دهید، باید از عبارت set استفاده کنید. متد ایجاد یک جزء با استفاده از جاواسکریپت، خیلی شبیه به این است، تنها باید به جای استفاده از set، از عبارت Var استفاده کنید. مثال زیر از عبارت Var استفاده کرده است.

```
<%
Var MyBrow = Server.CreateObject ("MSWC.AdRotator”)
%>
```

مایکروسافت توصیه می‌کند، که عملکرد اجزا را به‌وسیلهٔ محدودهٔ عمل صفحه به‌وجود آورید. با ایجاد اجزا با محدودهٔ عمل صفحه، شما فضای کمتری از منابع سرویس‌دهندهٔ وب را اشغال می‌کنید. وقتی پردازش خاتمه می‌یابد، یک جزء با محدودهٔ عمل صفحه هر حافظه و منبعی را که احتیاج داشته است، رها می‌کند.

1. Component

ایجاد اجزا با محدودهٔ عمل Session

دومتد برای ایجاد اجزا با session وجود دارد. یکی این است که یک جزء را با استفاده از server.CreatecObject () به متغیری از نوع session، به شکلی که در مثال زیر می‌بینید نسبت دهیم.

```
<%
Set session("MyBrow") = Server.CreateObject ("MSWC.PageCounter")
%>
```

این اسکریپت یک متغیر session به نام MyBrow را به یک نمونهٔ جزء شمارندهٔ صفحه نسبت می‌دهد. این متغییر session روی تمام صفحاتی که یک کاربرمشخص تقاضا می‌کند، قابل استفاده است. شما می‌توانید این اسکریپت را درون Session_Onstart از فایل Global.asa یا هر ASP دیگری قرار دهید.

روش دومی برای ایجاد یک جزء با محدودهٔ session وجود دارد. با استفاده از برچسب <object> می‌توانید یک جزء را درون فایل global.asa ایجاد کنید. مانند زیر:

```
<OBJECT   RUNAL="Server"  SCOPE= "session"  ID="MyBrow"
PROGID= "MSWC.BrowserType">
</OBJECT>
```

این مثال، چگونگی ایجاد یک نمونه از امکانات مرورگر با استفاده از برچسب <object> را نشان می‌دهد. ویژگی محدودهٔ عمل تعیین می‌کند، که جزء ایجاد شده محدودهٔ عمل session دارد. ویژگی ID به جزء، یک شناسهٔ خاص نسبت می‌دهد، به‌طوری که شما می‌توانید در اسکریپت‌های ASP خود به آن رجوع کنید. PROGID برای تخصیص یک نام ثبت شده می‌باشد. این نامی است که سرویس‌دهنده برای شناسایی جزء وقتی که یک نمونه ازآن به وجود می‌آید، استفاده می‌کند و همان نامی است که از آن در متد Server.Createobject استفاده می‌کنید.

هنگامی که از برچسب <object> درون فایل Global.asa استفاده می‌کنید، آن را باید خارج از هر اسکریپت دیگری قراردهید. از برچسب <object> درون Application_Onstart , Session_OnEnd , Session یا Application_OnEnd استفاده نمی‌کنیم.

وقتی یک جزء با محدودهٔ عمل Session با هر یک از دو روش شرح داده شده ایجاد شود، هر یک از متدها، مجموعه‌ها یا خواصش روی هر صفحه‌ای که یک کاربر درخواست می‌کند، در دسترس می‌باشد. به هر شکل، یک نمونه خاص از جزء، باید برای هر کاربر ایجاد شود. شبیه یک متغیر Session، یک جزء نیز که با محدودهٔ عمل Session ایجاد شده باشد، به یک کاربر خاص وابسته است.

به وجودآوردن اجزاء با استفاده ازمحدودهٔ عمل Application

هنگام ایجاد نمونهٔ یک جزء دارای محدودهٔ عمل Application، می‌توانید با آن، همانند یک شیء تعبیه شده رفتار کنید. به محض ایجاد، متدها، مجموعه‌ها و خصوصیات جزء آنها برای کاربران در هر صفحه‌ای قابل دسترسی می‌باشند و تا زمانی که سرویس‌دهنده، Shutdown نشده باشد، یا فایل global.asa تغییر داده نشده باشد و یا Unload Application نشود، در دسترس باقی می‌ماند.

می‌توانید جزء دارای محدودهٔ عمل Application را با همان متدی که در ایجاد جزء دارای محدودهٔ عمل session استفاده می‌کنید، به وجود آورید. این کار را می‌توانید توسط متد Server.CreateObject() همانند مثال زیر انجام دهید:

```
<%
Set Application("MyBrow") = Server.Ceateobject ("ADODB.Connection”)
%>
```

در بالا، جزء ADO به یک متغیر Application تخصیص داده شده است. شما می‌توانید این کار را در یک اسکریپت درون فایل global.asa مثل اسکریپت Application_Onstart انجام دهید. با این روش می‌توانید یک جزء را درون هر صفحهٔ ASP ای بگنجانید. بعد از اینکه یک جزء دارای امکانات مرورگر با محدودهٔ عمل Application به وجود آمد، از خواص آن می‌توانید در تمام صفحات استفاده کنید.

شما می‌توانید یک جزء با محدودهٔ عمل application را با برچسب <object > به وجود آورید.

مانند زیر:

```
<OBJECT  RUNAT= " server"  SCOPE="Application"  ID="MyBrow"
 PROGID= "MSWC.BrowserType">
 </OBJECT>
```

در این مثال برچسب <object> ، برای ایجاد یک نمونه از جزء امکانات مرورگر با محدودهٔ عمل Application به‌کار رفته است. ویژگی محدودهٔ عمل مشخص کرده است، که جزء بیشتر باید دارای محدودهٔ عمل Application باشد، تا session. ویژگی ID یک نام برای جزء فراهم می کند. ویژگی PROGID به سرویس‌دهنده اجازه می‌دهد، تا یک جزء را شناسایی کند. شما می‌توانید برچسب <object> را درون فایل global.asa قرار دهید، البته این برچسب باید خارج از هر اسکریپت دیگری قرار گیرد. همچنین برچسب <object> درون، اسکریپت‌های Application_Onstart ، session_Onstart, Session_OnEnd و Application_OnEnd قرار نمی‌گیرد.

به‌راستی چه وقت نیاز به ایجاد شیء با محدودهٔ عمل Application دارید؟ در برنامه‌نویسی WhosOn، که در فصل قبل آمده، شما چگونگی پی‌گیری درخواست‌های کاربران را فراگرفتید. این اطلاعات در یک دیکشنری ایجاد شده و دارای محدودهٔ عمل Application می‌شود. لازم است که این اجزاء با محدودهٔ عمل Application به وجود آیند، در غیر این صورت کاربر روی تمام صفحات نمی‌تواند به آنها دسترسی پیدا کند.

سرآیندهای اجازه

هنگامی که صفحهٔ وبی توسط کلمهٔ عبور سرویس‌دهندهٔ Proxy محافظت شده باشد، چهار سرآیند برای دریافت اطلاعاتی در مورد دسترسی کاربر به صفحه، مفید می‌باشد. سرآیند AUTH -TYPE روش محافظتی برای دسترسی به صفحهٔ مورد استفاده را نشان می‌دهد. سرآیندهای LOGON_USER , AUTH_USER حاوی نام کلمهٔ عبور کاربر ویندوز NT است. درنهایت وقتی از حفاظت سطح پایین استفاده می‌شود، AUTH_PASSWORD، حاوی کلمهٔ عبوری که برای دسترسی به صفحه مورد استفاده قرار گرفته، می‌باشد.

برای مثال، برنامهٔ ASP زیر بررسی می‌کند، که آیا کاربر از حفاظت سطح پایین برای دسترسی به صفحه استفاده کرده است یا نه. این کار را توسط AUTH_TYPE انجام می‌دهد. این سرآیند می‌تواند تنها دو مقدار داشته باشد: یکی Basic که برای حفاظت ابتدایی مورد استفاده قرار می‌گیرد، یا NTML که برای همراهی و پاسخ به NT مورد استفاده قرار می‌گیرد. به مثال زیر توجه کنید: (شکل ۱۶- ٦)

```
<HTML>
<HEAD> <TITLE> Password Protected </ TITLE> </ HEAD>
<BODY>

<%
IF Request.ServerVariables ( "AUTH_TYPE" ) = "BASIC"  THEN
%>

 You are Logged in using Basic Authentication.
You are Logged in Using NT challenge

<% Response.ServerVariables ("LOGON_ USER" )%>
<%ELSE%>

       You are Logged in Using NT Challenge and Response You Account
is

       <% =Request.ServerVariables ( "LOGON _USER" )% >

       <%END IF%>

       </ BODY>


       </ HTML>
```

شکل ۱٦-٦خروجی برنامهٔ بالا

برچسب ها در HTML

فرض کنید بخواهید سایت وبی را طراحی کنید، که برنامه‌نویسی را با استفاده از ASP به کاربر یاد بدهد. فرض کنید کسی یک کد نمونه را با استفاده از HTML ارسال کرد و شما می‌خواهید که برچسب-

های واقعی HTML نمایش داده شود و مرورگر آن را تفسیر[1] نکند، در این صورت به چه روشی باید عمل کنید؟

خوشبختانه ASP، دارای متدهای مخصوصی برای این منظور خاص می‌باشد. متد Server.HTMLEncode( )

برچسب‌های HTML را به کدهای کاراکتر HTML ترجمه می‌کند (این کدها در ضمیمهٔ کتاب شرح داده شده است). مثال زیر چگونگی استفاده از این متد را نشان می‌دهد :

```
< %=Server.HTMLEncode  "<B> This is Bold </B>"   %\>
```

در حالت عادی، اگر رشتهٔ "< B > This is bold </B>" توسط مرورگر دریافت شود، متنی نمایش داده می‌شود، که برجسته و ضخیم خواهد بود، ولی اگر قبل از آن با HTMLEncode آن را کدگذاری کنیم، چیزی که توسط مرورگر نمایش داده خواهد شد، همان رشته به همراه برچسب‌های مزبور خواهد بود.

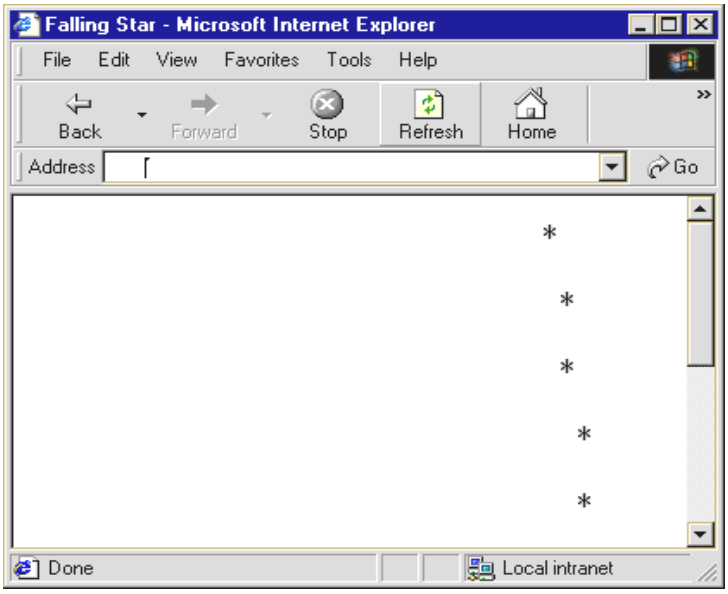
کارباسکریپت‌هایی بااجرای طولانی و صفحات طولانی HTML

بنا به تعریف، بیشرین زمانی که به اجرای یک اسکریپت درون ASP اختصاص می‌یابد، 90 ثانیه می‌باشد، که این باعث جلوگیری از حلقه‌های بی‌نهایت می‌شود. در بعضی موارد نیاز است، که به دستورات اسکریپت زمانی بیشتر از 90 ثانیه اختصاص دهیم. به عنوان مثال وقتی که می‌خواهیم از اسکریپت برای خروجی یک HTML طولانی استفاده کنیم و نمی‌خواهیم تا قبل از آنکه پردازش به‌صورت کامل تمام شود، وقت به پایان برسد. می‌توانیم با استفاده از خاصیت Script Timeout، متعلق به شیء Server حد اکثر زمان مورد نیاز برای اجراء را کنترل کنیم.

```
<HTML>
<HEAD> <TITLE> Falling Star </ TITLE> </ HEAD>
<BODY>
 <% Server.Scripttimeout=150
 RANDOMIZE
 Starx = 60
 FOR K =1 To 10
 NextScond = DATEADD ( " S " , 10 , Time )
 Do WHILE TIME<NextSecond
 Starx = Starx + 3 * RND (  ) -1
 FOR i =1 To Starx
 Response.Write ( "   ;" )
 NEXT
 Response . Write ( " *<p>" )
 NEXT
 %>
 </ BODY>
 </ HTML>
```

در این برنامه، زمان اجرای اسکریپت به‌طور معمول قبل از اینکه افتادن ستاره‌ها پایان یابد، تمام می‌شود. به‌هرحال خط اول این صفحه از وقوع این حالت جلوگیری می‌کند، چرا که خاصیت ScriptTimeout از شیء Server، روی زمان انقضای[2] 150 ثانیه تنظیم شده است. (شکل ۱۷– ٦)

1. Interprete
2. Time out



شکل ۱۷-۶ برنامهٔ نمایش ستاره‌ها

باید توجه کرد که نمی توان از خاصیت Server.scriptTimeout برای کاهش زمان انقضای اسکریپت‌ها کمتر از 90 ثانیه استفاده کرد. برای اینکه اسکریپت خود را وادار کنید، که قبل از 90 ثانیه خاتمه یابد، لازم است، که خاصیت ScriptTimeout را توسطInternet Service Manager تغییردهید. این خاصیت در گزینه‌های ASP در منوی Application Configuration قرار دارد. اگر خاصیت ScriptTimeout را روی 1 تنظیم کرده باشید، زمان اسکریپت هرگز خاتمه نخواهد یافت.

کد کردن یک QueryString

یک QueryString قبل از اینکه به صفحه دیگر فرستاده شود، باید به صورت یک کد URL تبدیل شود. به عنوان مثال تمام جاهای خالی به علامت جمع تبدیل می‌شود. اگر فراموش شود که QueryString را به صورت کد URL تبدیل کنیم، ممکن است نتیجهٔ غلطی به‌دست آوریم.

اتفاقاً در ASP تبدیل رشتهٔ درخواست به کد URL بسیار آسان است .متد ( )Server.URLEncode هر رشته را به کد URL تبدیل می‌کند. به مثال زیر توجه کنید:

```
A HREF = "/response .asp? message = <%= server.URLEncode  <
"This query    string  hasbeen  URL   encoder . ")%>"> Click  Here
</A> (
```

توجه داشته باشید، که نام QueryString و علامت مساوی را تبدیل به کد نکنید. با انجام این کار دچار مشکل می‌شوید. شما فقط باید مقدار QueryString را تبدیل به کد کنید. QueryString ای که در مثال قبلی تبدیل به کد شده است، به صورت زیر نوشته می‌شود.

```
Message = This + query+string+has+been +URL+encoded %2E
```

نباید نگران کد کردن رشته‌های خود باشید، زیرا این کار را ASP به‌صورت خودکار انجام می‌دهد. به عنوان مثال، صفحهٔ response.asp که شامل خط زیر است، در نظر بگیرید.

```
<%  =Request .QuerySting ("message")%>
```

خروجی این پیغام به صورت کد URL نمی‌باشد و شما می‌توانید آن را به صورت زیر ببینید.

```
This Querystring has been URL encoded
```

۷-۶ تمرین‌ها

۱. تفاوت بین بین مجموعهٔ Request.QueryString و مجموعهٔ Request.Form چیست؟
۲. چگونه می‌توانید کل QueryString را با یک متغیر بخوانید؟
۳. ازطریق Asp یک برنامه بنویسید، که یک ماشین حساب برای انجام چهار عمل اصلی + و− و× و/ داشته باشد.
٤. تفاوت Session و Application در چیست؟ به‌طور کامل شرح دهید.
٥. یک صفحهٔ Asp ایجاد کنید و در داخل آن کدی بنویسید، که تمامی Cookie های سایت وب شما را که بر روی کامپیوتر سرویس‌گیرنده ایجاد شده است، پاک نماید.
٦. برنامهٔ Chat.asp را بدون استفاده از frame با یک فایل asp بنویسید.
۷. برنامه‌ای بنویسید، که اطلاعات کاربر را گرفته و این اطلاعات را دوباره برای کاربر برگرداند. (هر دو متد را پشتیبانی نماید).
۸. برنامه‌ای با Asp بنویسید، که نام کاربر را در صفحهٔ اول گرفته و در سایر صفحات، نام کاربر را به عنوان Header نمایش دهد.
۹. برنامه‌ای بنویسید، که اطلاعات کاربر را گرفته و اگر آدرس IP کاربر 194.224.1.2 بود، این اطلاعات را در Logfile ثبت کند.

ضمیمه ۱

کدهای مشخصه کشورها و سازمان‌ها در اینترنت

Internet country identification codes.

Country identification code	*Country*
AE	United Arab Emirates
AF	Afghanistan
AI	Anguilla
AL	Albania
AM	Armenia
AN	Netherlands Antilles
AR	Argentina
AU	Australia
AZ	Azerbaijan
BA	Bosnia and Herzegovina
BD	Bangladesh
BH	Bahrain
BR	Brazil
BV	Bouvet Island
CA	Canada
CC	Cocos (Keeling) Islands
CF	Central African Republic
CG	Congo
CH	Switzerland
CK	Cook Islands

CM	Cameroon
CN	China
CO	Colombia
CR	Costa Rica
CS	Czechoslovakia (former)
CU	Cuba
DE	Germany
DK	Denmark
DZ	Algeria
ER	Eritrea
ES	Spain
FK	Falkland Islands (Malvinas)
FR	France
GB	Great Britain (UK)
GD	Grenada
GL	Greenland
GM	Gambia
GN	Guinea
GP	Guadeloupe
GQ	Equatorial Guinea
GR	Greece
GS	South Georgia and South Sandwich Islands
GT	Guatemala
HK	Hong Kong
HM	Heard and McDonald Islands
HN	Honduras
HR	Croatia (Hrvatska)
HT	Haiti

ID	Indonesia
IE	Ireland
IL	Israel
IN	India
IO	British Indian Ocean Territory
IQ	Iraq
IR	Iran
IS	Iceland
IT	Italy
JM	Jamaica
JO	Jordan
JP	Japan
KE	Kenya
KG	Kyrgyz Republic
KI	Kiribati
KM	Comoros
KN	Saint Kitts Nevis Anguilla
KP	Korea (North)
KR	Korea (South)
KW	Kuwait
KY	Cayman Islands
KZ	Kazachstan
LA	Laos
LB	Lebanon
LC	Saint Lucia
LI	Liechtenstein
LK	Sri Lanka
LV	Latvia

LY	Libya
MA	Morocco
MC	Monaco
MD	Moldova
MG	Madagascar
MH	Marshall Islands
MK	Macedonia
ML	Mali
MM	Myanmar
MN	Mongolia
MQ	Martinique
MR	Mauritania
MX	Mexico
MY	Malaysia
MZ	Mozambique
NE	Niger
NF	Norfolk Island
NG	Nigeria
NI	Nicaragua
NP	Nepal
NR	Nauru
NT	Neutral Zone
NZ	New Zealand (Aotearoa)
OM	Oman
PA	Panama
PE	Peru
PF	French Polynesia
PG	Papua New Guinea

PH	Philippines
PK	Pakistan
PM	Saint Pierre and Miquelon
PT	Portugal
PY	Paraguay
QA	Qatar
RO	Romania
RU	Russian Federation
SA	Saudi Arabia
SD	Sudan
SG	Singapore
SI	Slovenia
SJ	Svalbard and Jan Mayen Islands
SK	Slovak Republic
SL	Sierra Leone
SN	Senegal
SO	Somalia
SR	Suriname
ST	Sao Tome and Principe
SU	USSR (former)
SV	El Salvador
SY	Syria
TR	Turkey
TW	Taiwan
TZ	Tanzania
UA	Ukraine
UG	Uganda
US	United States

YU	Yugoslavia
ZA	South Africa
ZM	Zambia
ZR	Zaire
ZW	Zimbabwe

Internet organization identification codes.

Organization identification code	*Organization category*
com	U.S. Commercial
edu	U.S. Educational
gov	U.S. Government
int	International
mil	U.S. Military
net	Network
org	Non-profit Organization

Other Internet organization identification codes.

Organization identification code	*Organization category*
arpa	Old Style Arpanet
nato	NATO Field

ضمیمه ۲

تگ‌های HTML 3.0

CONTENTS

General Tags

<HTML> </HTML>

The <HTML> and </HTML> tags mark the beginning and end of the HTML file respectively.

<TITLE> </TITLE>

The text enclosed in the <TITLE> and </TITLE> tags is displayed by the browser in the title area-most often the browser's application title bar.

<HEAD> </HEAD>

Descriptive information, such as the title and document type, are enclosed in the HEAD tags.

<BODY> </BODY>

The BODY tags mark the beginning and end of the document body. These tags are not required for the document to look correct on most browsers, but it is proper HTML practice to put them at the beginning and end of the document, respectively. A number of attributes can be set in the body tags.

```
<BODY BACKGROUND = "back.gif">
```

The BACKGROUND attribute sets the background to a tiled series of the image back.gif. You can also set the color of the document text, links that have or have not been visited, and active links, as shown in the following lines:

<BODY TEXT = "*#rrggbb*">

<BODY LINK = "*#rrggbb*">

<BODY ALINK = "*#rrggbb*">

<BODY VLINK = "*#rrggbb*">

http://www.biola.edu/cgi-bin/colorpro/

This sets the text, link, activated link, and visited link to the color where rr as the amount of red, gg as the amount of green, and bb as the amount of blue. The numbers provided are hexadecimal in the range of 00 - FF. Many utilities generate the values of these values automatically from a color dialog box, or sliders for the red, green, and blue colors. A wonderful online resource for color setting is the ColorServer Pro site at http://www.biola.edu/cgi-bin/colorpro/.

<BODY BACKGROUND = "back.gif" TEXT = "*#rrggbb*" LINK = "*#rrggbb*"

ALINK = "*#rrggbb*" VLINK = "*#rrggbb*">

(Insert HTML document here)

</BODY>

Note
Microsoft Internet Explorer 3.0 supports named colors wherever a color attribute can be used. For example, <BODY TEXT=TEAL> causes the text in the document to be teal.

Formatting HTML Text

HTML provides many ways to format the look of text appearing in your Web documents. Table B.1 lists many of the tags for formatting textual information.

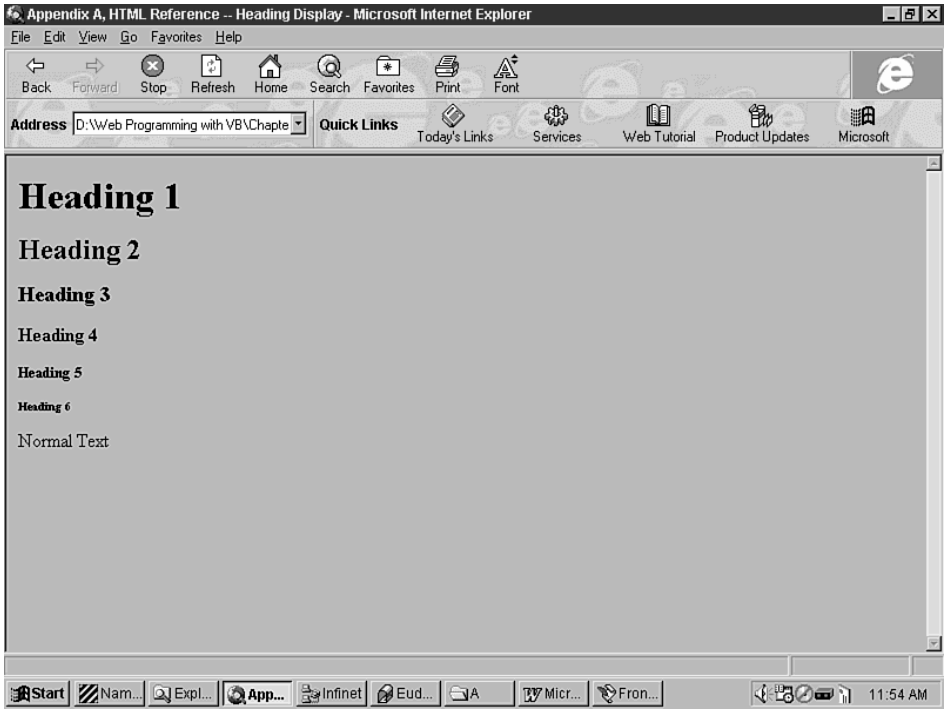
Table B.1. Tags for formatting text.

Tag	*Descr · ·iption*
<B> </B>	Applies boldface to text between tags.
<BLINK> </BLINK>	Causes any text between the tags to blink.
<CITE> </CITE>	Used to offset a citation from a book or magazine article. Browsers typically display citations in italics.
<CODE> </CODE>	Offsets a source code program listing. The browser usually displays this in a smaller monospaced font.
<DFN> </DFN>	Styles a definition of a term. Browsers typically display definitions in

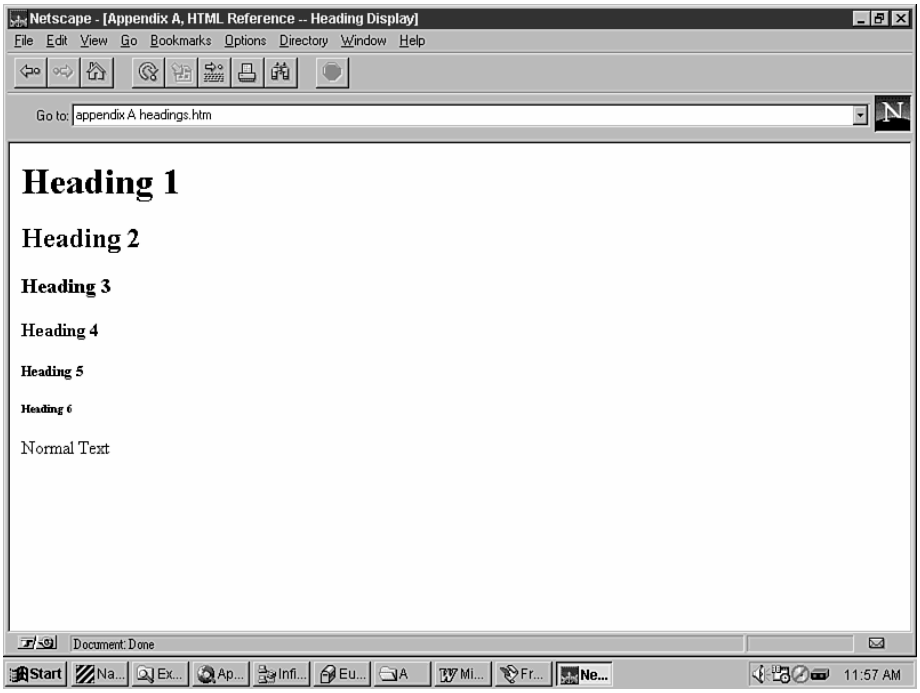
	italics.
<EM> </EM>	The text between these tags is emphasized. Most browsers use italic to show emphasized text.
<FONT SIZE = #> </FONT>	Sets the font size (#) used to display the HTML appearing between the tags. Default is 3; the range is 1 to 7.
<H#> </H#>The # in the <H#>	tag specifies the heading level-a number between 1 and 6, with 1 as the largest size and 6 as the smallest.
<I> </I>	Text occurring between the tags is displayed in italics.
<KBD> </KBD>	The Keyboard tag is used to represent keyboard keys such as the Del or Enter keys. On most browsers this is displayed in a smaller monospaced font.
<PRE> </PRE>	Preformatted text is displayed without any special formatting. This is a good tag to use if you want to maintain spacing over several lines; it's also an easy way to put tabs into your document.
<MARQUEE> </MARQUEE>	(Microsoft Internet Explorer 2.0/3.0 only.) This tag causes the text within to scroll by default from left to right across the screen. The attributes of the MARQUEE element are as follows:
ALIGN	Specifies that the text around the marquee should align to the TOP, MIDDLE or BOTTOM of the marquee.
BEHAVIOR	The valid settings for this attribute are SCROLL, SLIDE, and ALTERNATE.
SCROLL	The default, repeatedly scrolls the text from one side of the screen to the other, then repeats.
SLIDE	Scrolls the text across the screen, and when it hits the margin, stops.
ALTERNATE	Bounces the text from margin to margin.
BGCOLOR	This sets the background color of the marquee.
DIRECTION	The direction can be either the default LEFT, or RIGHT. This specifies the start direction either left to right, or right to left, of the marquee.
HEIGHT	Sets the height of the marquee either in pixels (HEIGHT = n) or in a percentage of the screen (HEIGHT = n%).
WIDTH	Similar to the HEIGHT attribute, but applies to the width of the marquee.
HSPACE	Sets the left and right margins for the outside of the marquee in pixels.
LOOP	Specifies the number of times the marquee should scroll. If left undefined, set to -1 or INFINITE, and the marquee scrolls until the user leaves the page.
SCROLLAMOUNT	Sets the number of pixels to move the text between redrawing of the marquee text.
SCROLLDELAY	The number of milliseconds between movements of the marquee text.

VSPACE	Sets the top and bottom margins of the outside marquee in pixels.
<SAMP> </SAMP>	Shows a group of literal characters. Most browsers display this as a monospaced font.
<STRIKE> </STRIKE>	Text enclosed in these brackets appears struckout (~~Example~~).
<STRONG> </STRONG>	Another way of drawing attention to a special section of text; <STRONG> usually is displayed in bold by most browsers.
<TT> </TT>	Displays in a typewriter style (monospaced) font.
<U> </U>	Text occurring between the tags is displayed underlined.

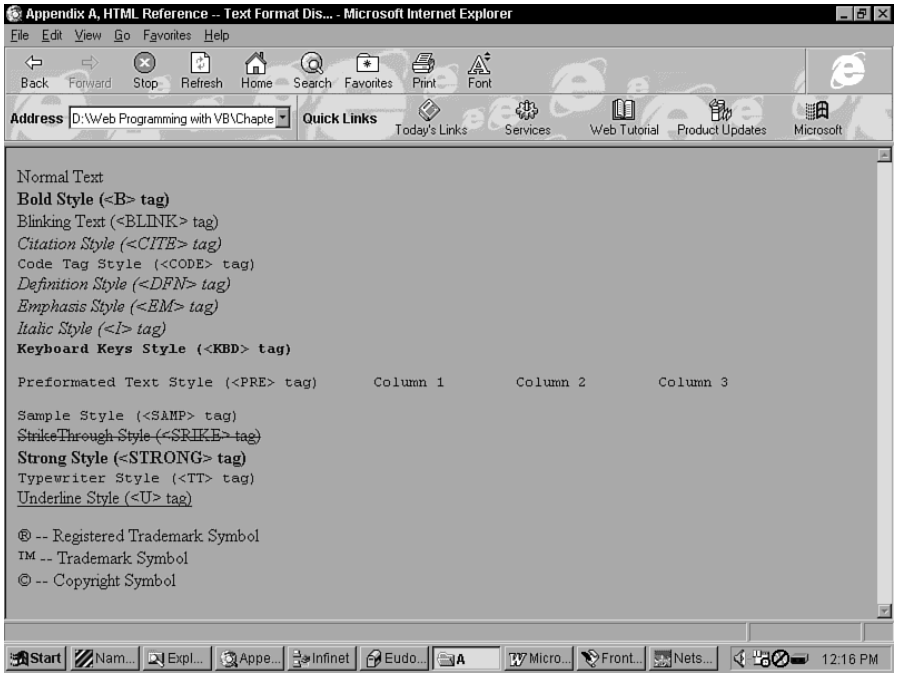
Figures B.1 through B.4 demonstrate how Netscape Navigator and Microsoft Internet Explorer display various headings and text formatting tags.



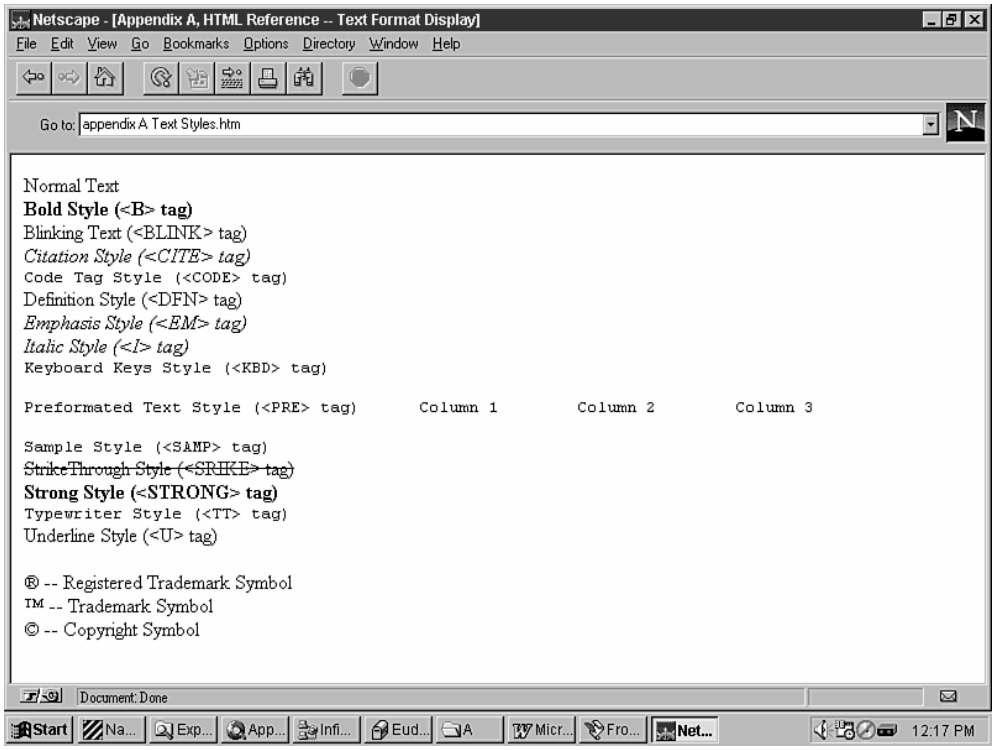
B.1 *Headings displayed by Microsoft Internet Explorer.*



B.2 *Headings displayed by Netscape Navigator.*



B.3 *Text formats displayed by Microsoft's Internet Explorer.*



B.4 *Text formats displayed by Netscape Navigator.*

Note
As you can see from Figure A.4, Navigator does not correctly render the <U> tag. The same HTML file was used for both Figure A.3 and A.4.

Text Alignment

<CENTER> </CENTER>

These tags center the text or image enclosed in the tags.

<P ALIGN=LEFT|CENTER|RIGHT> </P>

This aligns the text to the left, center, or right of the browser display.

<H# ALIGN=LEFT|CENTER|RIGHT> </H#>

This aligns the enclosed text to the left, center, or right and gives the text the heading level of the number that replaces the two # symbols.

Embedding Images Within Web Pages

To insert an image into a Web page, simply use the <IMG> tag. The tag has the following elements:

SRC=*URL* The URL that points to the image file.

ALIGN The ALIGN element specifies how the image will be aligned relative to text and other elements near the image. Many values are available and won't be covered here.

ALT=*text* Specifies the text (typically placed within quotation marks) to be displayed if the Web browser is incapable of displaying images or if the image was not loaded correctly for some reason.

HEIGHT=*h* **WIDTH=***w* Specifying the height and width of the image allows the Web browser to set aside enough space for the image before it actually retrieves the image. This aides the Web browser in rendering the entire document when all images haven't yet been retrieved.

An example of embedding an image into a Web page is

```
<IMG SRC=Picture.gif HEIGHT=100 WIDTH=65 ALIGN=TOP ALT="A Picture">
```

Linking to other Documents and Sites

HTML documents are hypertext documents. This means that they can contain links to other documents. The anchor (<A> </A>) tag is used to specify a link to another document or resource to be loaded when the user clicks the link. The text and any other HTML appearing between the anchor tags appears to the user as a hotlink, usually colored differently than regular browser text or underlined. The following sections describe the attributes of the anchor tag.

HREF

HREF = "*URL*" specifies the URL of the resource to be displayed after the user clicks on the anchor. As an example, <A HREF="http://www.microsoft.com"> specifies a link to the Microsoft Web site.

NAME

The NAME attribute specifies the name of a location within the current HTML document that can be linked to by a link from another page. This enables other documents to reference the specific anchor in a page. To reference a named anchor, the URL of the page is specified, then a #, followed by the name of the anchor. For example, <a name="Copyright"> defines a link named Copyright within the current document.

TITLE

This attribute, although not required, allows the page designer to include the title of the page that the anchor references.

URN

This specifies a URN (Universal Resource Name) for the target document.

Tables

Tables are implemented in both Internet Explorer 2.0 and 3.0, as well as Netscape 1.1 and later. Tables are a good way to logically group information that would otherwise appear in a spreadsheet. Using tables is also, in some cases, a convenient way to display database records.

The following code contains an example of a basic table:

```
<TABLE BORDER = 5>
<TR>
<TD>1</TD> <TD>2</TD> <TD>3</TD> <TD>4</TD>
</TR>
<TR>
<TD>5</TD> <TD>6</TD> <TD>7</TD> <TD>8</TD>
</TR>
</TABLE>
```

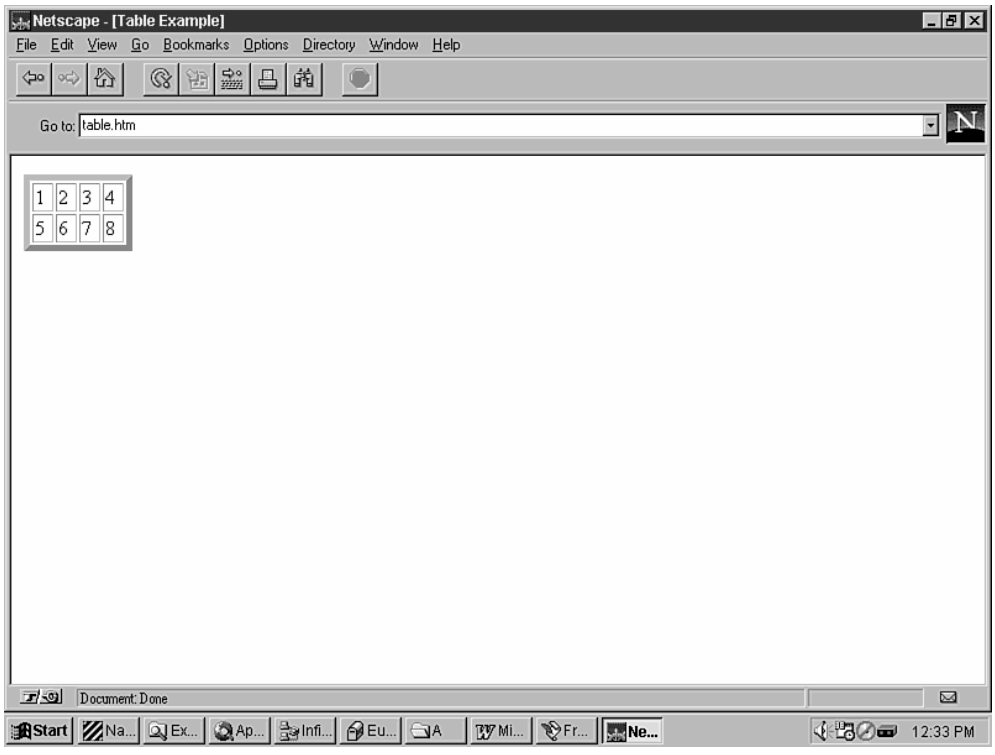


Figure B.5: *Simple table as displayed by Netscape Navigator.*

<TABLE> </TABLE>

The following are the attributes of the TABLE tag and their effects on the table.

BORDER

The BORDER attribute sets the width of the border as displayed by the browser. Experiment with the browser you use and the information within to find the optimum setting.

CELLSPACING

The CELLSPACING attribute determines the amount of space the browser places between each individual cell and the space between the cell and the border (if the cell is on the outer edge of the table). Again, experimentation works well to find an appropriate value.

CELLPADDING

The CELLPADDING attribute determines the amount of space the browser places between the data in the table cell and the border around the cell. The greater the value in CELLPADDING is, the more room around the actual value in the cell. Within the table tags, three other tags define the header, rows, and cells. These tags are the table header <TH>, table row <TR>, and table data <TD>.

The following are the attributes of the <TABLE> element.

WIDTH

The WIDTH attribute can be set either as a number defining the absolute width in pixels, or as a percentage of the browser's viewing space. The browser can stretch the table, but it does not compress the table if the size of the browser window is smaller than that of the table.

ALIGN

With ALIGN, you can specify left, center, or right alignment of the table.

BGCOLOR

As with the <BODY> tag, you can set the background color of tables using the BGCOLOR attribute, you can either use the *#rrggbb* color definition, or, if you're designing for Internet Explorer, you can specify the color by name (for example, <TABLE BGCOLOR = RED). The *#rrggbb* color values follow those of the <BODY> tag as described at the beginning of this appendix.

BORDERCOLOR (Microsoft Internet Explorer 2.0/3.0 Only)

This works similarly to the BGCOLOR attribute but applies to the tables external and internal borders.

BORDERCOLORLIGHT (Microsoft Internet Explorer 2.0/3.0 Only)

This setting is optional and sets the color of the thin line around the top and left sides of the entire exterior of the table and the bottom right sides of the interior of the table. These colors can be modified to enhance or change the default 3-D appearance.

BORDERCOLORDARK (Microsoft Internet Explorer 2.0/3.0 Only)

This setting functions as the exact opposite of BORDERCOLORLIGHT, coloring the bottom and right sides of the exterior and the upper and left sides of the interior cells of the table.

<TH> </TH>

The table header element functions the same as the table data <TD> element, except the table header element is displayed in a bold font. The valid attributes for the table header element are ROWSPAN, COLSPAN, ALIGN, VALIGN, NOWRAP, BGCOLOR, BGCOLORDARK, and BGCOLORLIGHT. The attributes function similarly to the table elements but affect only the header cell that they are set in.

<TD> </TD>

The table data element marks the beginning and end of each cell in the table. The valid attributes for the table data element are ROWSPAN, COLSPAN, ALIGN, VALIGN, NOWRAP, BGCOLOR, BGCOLORDARK, and BGCOLORLIGHT. These elements are described in the following sections.

ROWSPAN

This attribute specifies the number of rows the cell should stretch; for instance, if ROWSPAN is set to 3, that cell will span three normal rows in the table.

COLSPAN

This attribute specifies the number of columns the cell should stretch; for instance, if COLSPAN is set to 3, that cell will span three normal columns in the table.

NOWRAP

Inserting the NOWRAP attribute keeps the text of the cell all on one line.

<TR> </TR>

The table row element marks the beginning and end of each row in the table. The valid attributes of the table row element are ALIGN, VALIGN, BGCOLOR, BGCOLORDARK, and BGCOLORLIGHT.

<CAPTION> </CAPTION>

The CAPTION tags are placed within the table tags, but not within row, heading, or cell tags. The text in the CAPTION tags defines the associated caption of the table. The following are the valid attributes of the CAPTION tag.

ALIGN

The valid settings for the ALIGN attribute for Netscape are TOP and BOTTOM; Microsoft's Internet Explorer includes TOP, BOTTOM, LEFT, RIGHT, and CENTER.

VALIGN

With VALIGN, you can specify the vertical alignment of the data within the table cells. This attribute is set within the <TD> tag of the cell to be aligned. Valid settings are TOP, MIDDLE, and BOTTOM. Not

specifying the tag results in center aligned (MIDDLE) text. For example, <TD VALIGN=TOP>Top Aligned</TD> specifies a cell that will be aligned with the top of the row.

Forms

Forms are the way to get feedback from the visitors to your Web site. Beginning and ending with the <FORM> and </FORM> tags, the fields within these determine the size and type of input fields. Although you can have multiple forms on a page, you cannot nest forms.

The <FORM> Tag

The attributes of the form element are described in the following sections.

ACTION

This specifies the URL of the resource that will carry out the action on the form data, and give the user a response.

ACTION = "http://www.execpc.com/~haasch/search"

METHOD

This can either be the default GET or POST. Using GET, the query appends the form data to the end of the URL; using POST, the data is sent via an HTTP post transaction.

Elements Contained Within a Form

The following sections describe the various elements that can be placed within a form.

INPUT

The INPUT element specifies the user interface to enter information. The following are the attributes for the INPUT tag.

chECKED

For checkboxes and radio buttons, this attribute can be set to TRUE (checked) or FALSE (unchecked).

MAXLENGTH

MAXLENGTH indicates the maximum number of characters that can be entered into a textbox.

NAME

This specifies the name of the form control. This is used to identify the data elements on the form to the resource that processes those elements. It's also used by VBScript (described in Appendix B, "Visual Basic Script Reference") to provide similar functionality to control names in Visual Basic.

SIZE

This specifies the size of the form control. This can either be specified in a single value, representing the width in characters of the control, or in a width/height pair.

SRC

This specifies the image to be displayed with the control.

TYPE

This sets the type of control to use. The following is a list of available controls:

chECKBOX	The checkbox control is a simple TRUE/FALSE control, where checked is TRUE, and empty indicates FALSE.
HIDDEN	This control is not displayed to the viewer of the page. It can be used to send state information back to the form-processing program.
IMAGE	This causes the form data to be submitted immediately, and the value passed back from the image is the x, y coordinates in pixels.
PASSWORD	This functions the same as a textbox, but the text is echoed as asterisks instead of the letters typed.
RADIO	This functions much like the chECKBOX control, but only one option button in the group may be selected at a time.
RESET	When a RESET button is clicked, the form data returns to the initial values specified in the form's data element definitions. The VALUE attribute can be set to give the RESET button a label.
SUBMIT	Clicking this button submits the form data to the FORM ACTION URL.
TEXT	This control is used to gather a single line of text. SIZE and MAXLENGTH attributes can be set to restrict the size of user input and the appearance of the text control.
TEXTAREA	This control is used for multiline text input. The SIZE and MAXLENGTH attributes have the same function as the TEXT control.

SELECT

The SELECT tags mark the beginning and end of the data in a listbox or a drop-down selection list. The following are the attributes for the SELECT element.

MULTIPLE	The multiple attribute enables the user to select more than one item from the listbox. The user holds the CTRL button and clicks on the different items to select them.
NAME	This specifies the name of the SELECT element.
SIZE	This sets the height of the list control.

OPTION

The OPTION element sets off each selection in the textbox or listbox. The following are the attributes of the OPTION element:

SELECTED	This attribute sets the default value of the text- or listbox.
VALUE	This is the return value of the element that is selected.

Netscape 2.x/Microsoft Internet Explorer 3.x Frames

With Netscape 2.0 and above, the browser is able to display frames that contain different sets of HTML code.

<FRAMESET> </FRAMESET>

The FRAMESET element is the container element for a group of frames. Its two attributes are ROWS and COLS.

ROWS

The ROWS tag specifies the amount of space given to each row. The number can be specified in pixels, in percent by placing a % after the value, or as a relative value by placing an asterisk (*) in place of the value.

COLS

The syntax for the COLS attribute is the same as that of the ROWS attribute.

<FRAME>

The FRAME element specifies the properties for each individual frame in the frameset. Because it contains no text, there is no corresponding end-tag. The attributes of the FRAME tag are as follows.

SRC

The SRC attribute specifies the URL source for the frame.

NAME

The NAME attribute is used to assign a name to the frame, so it can act as a target from other URLs.

MARGINWIDTH

This allows the page designer to specify the width of the frame border in number of pixels.

MARGINHEIGHT

This works the same as MARGINWIDTH except it affects the height of the margin.

SCROLLING

The valid settings for the SCROLLING attribute are YES, NO, and AUTO. If set to YES, a scrollbar is provided for the frame; NO results in no scrollbar; and AUTO provides a scrollbar if the document size exceeds the size of the frame.

NORESIZE

Specifying this attribute prevents the user from dragging the margins of the frame to cause them to resize. Default is to allow the user to resize the frames.

<NOFRAMES> </NOFRAMES>

The data within the NOFRAMES tags is ignored by form-capable browsers. This set of tags allows information to be displayed to non-forms capable browsers.

ضمیمه ۳

کد رنگ‌های RGB در HTML

RGB color codes.

Name of color	***RGB color code***
Aquamarine	#70DB93
Baker's Chocolate	#5C3317
Black	#000000
Blue	#0000FF
Blue Violet	#9F5F9F
Brass	#B5A642
Bright Gold	#D9D919
Bronze	#8C7853
Bronze II	#A67D3D
Brown	#A62A2A
Cadet Blue	#5F9F9F
Cool Copper	#D98719
Copper	#B87333
Coral	#FF7F00
Cornflower Blue	#42426F
Cyan	#00FFFF
Dark Brown	#5C4033
Dark Green	#2F4F2F
Dark Green Copper	#4A766E
Dark Olive Green	#4F4F2F
Dark Orchid	#9932CD
Dark Purple	#871F78
Dark Slate Blue	#6B238E
Dark Slate Grey	#2F4F4F
Dark Tan	#97694F
Dark Turquoise	#7093DB
Dark Wood	#855E42
Dim Grey	#545454

Dusty Rose	#856363
Feldspar	#D19275
Firebrick	#8E2323
Forest Green	#238E23
Gold	#CD7F32
Goldenrod	#DBDB70
Green	#00FF00
Green Copper	#527F76
Green Yellow	#93DB70
Grey	#C0C0C0
Hunter Green	#215E21
Indian Red	#4E2F2F
Khaki	#9F9F5F
Light Blue	#C0D9D9
Light Grey	#A8A8A8
Light Steel Blue	#8F8FBD
Light Wood	#E9C2A6
Lime Green	#32CD32
Magenta	#FF00FF
Mandarin Orange	#E47833
Maroon	#8E236B
Medium Aquamarine	#32CD99
Medium Blue	#3232CD
Medium Forest Green	#6B8E23
Medium Goldenrod	#EAEAAE
Medium Orchid	#9370DB
Medium Sea Green	#426F42
Medium Slate Blue	#7F00FF
Medium Spring Green	#7FFF00
Medium Turquoise	#70DBDB
Medium Violet Red	#DB7093
Medium Wood	#A68064
Midnight Blue	#2F2F4F
Navy Blue	#23238E
Neon Blue	#4D4DFF
Neon Pink	#FF6EC7
New Midnight Blue	#00009C

New Tan	#EBC79E
Old Gold	#CFB53B
Orange	#FF7F00
Orange Red	#FF2400
Orchid	#DB70DB
Pale Green	#8FBC8F
Pink	#BC8F8F
Plum	#EAADEA
Quartz	#D9D9F3
Red	#FF0000
Rich Blue	#5959AB
Salmon	#6F4242
Scarlet	#8C1717
Sea Green	#238E68
Semi-Sweet Chocolate	#6B4226
Sienna	#8E6B23
Silver	#E6E8FA
Sky Blue	#3299CC
Slate Blue	#007FFF
Spicy Pink	#FF1CAE
Spring Green	#00FF7F
Steel Blue	#236B8E
Summer Sky	#38B0DE
Tan	#DB9370
Thistle	#D8BFD8
Turquoise	#ADEAEA
Very Dark Brown	#5C4033
Very Light Grey	#CDCDCD
Violet	#4F2F4F
Violet Red	#CC3299
Wheat	#D8D8BF
White	#FFFFFF
Yellow	#FFFF00
Yellow Green	#99CC32

ضمیمه ٤

جدول کد کاراکترهای خاص در HTML

The ISO8859-1 table of special characters.

Description	*Character*	*Numeric Code*	*Character Code*
Quotation mark	"	&#34;	&quot;
Ampersand	&	&#38;	&amp;
Less-than sign	<	&#60;	&lt;
Greater-than sign	>	&#62;	&gt;
Non-breaking space			
Inverted exclamation	¡	&#161;	&iexcl;
Cent sign	¢	&#162;	&cent;
Pound sterling	£	&#163;	&pound;
General currency sign	¤	&#164;	&curren;
Yen sign	¥	&#165;	&yen;
Broken vertical bar	¦	&#166;	&brvbar;
Section sign	§	&#167;	&sect;
Dieresis	¨	&#168;	&uml;
Copyright	"	&#169;	&copy;
Feminine ordinal	ª	&#170;	&ordf;
Left angle quote	«	&#171;	&laquo;
Not sign	¬	&#172;	&not;
Soft hyphen	-	&#173;	&shy;
Registered trademark	,	&#174;	&reg;
Macron accent	¯	&#175;	&macr;
Degree sign	°	&#176;	&deg;
Plus or minus	±	&#177;	&plusmn;
Superscript two	²	&#178;	&sup2;
Superscript three	³	&#179;	&sup3;
Acute accent	´	&#180;	&acute;
Micro sign	µ	&#181;	&micro;
Paragraph sign	¶	&#182;	&para;
Middle dot	·	&#183;	&middot;

Cedilla	ç	¸	¸
Superscript one	[dotlessi]	¹	¹
Masculine ordinal	º	º	º
Right angle quote	»	»	»
Fraction one-fourth	1/4	¼	¼
Fraction one-half	1/2	½	½
Fraction three-fourths	3/4	¾	¾
Inverted question mark	¿	¿	¿
Capital A, grave accent	À	À	À
Capital A, acute accent	Á	Á	Á
Capital A, circumflex accent	Â	Â	Â
Capital A, tilde	Ã	Ã	Ã
Capital A, dieresis	Ä	Ä	Ä
Capital A, ring	Å	Å	Å
Capital AE ligature	Æ	Æ	Æ
Capital C, cedilla	Ç	Ç	Ç
Capital E, grave accent	È	È	È
Capital E, acute accent	É	É	É
Capital E, circumflex accent	Ê	Ê	Ê
Capital E, dieresis	Ë	Ë	Ë
Capital I, grave accent	Ì	Ì	Ì
Capital I, acute accent	Í	Í	Í
Capital I, circumflex accent	Î	Î	Î
Capital I, dieresis	Ï	Ï	Ï
Capital Eth, Icelandic	q	Ð	Ð
Capital N, tilde	Ñ	Ñ	Ñ
Capital O, grave accent	Ò	Ò	Ò
Capital O, acute accent	Ó	Ó	Ó
Capital O, circumflex accent	Ô	Ô	Ô
Capital O, tilde	Õ	Õ	Õ
Capital O, dieresis	Ö	Ö	Ö
Multiply sign	*	×	×
Capital O, slash	Ø	Ø	Ø
Capital U, grave accent	Ù	Ù	Ù
Capital U, acute accent	Ú	Ú	Ú
Capital U, circumflex accent	Û	Û	Û
Capital U, dieresis	Ü	Ü	Ü

Capital Y, acute accent	Y	Ý	Ý
Capital THORN, Icelandic	t	Þ	Þ
Small sharp s	ß	ß	ß
Small a, grave accent	à	à	à
Small a, acute accent	á	á	á
Small a, circumflex accent	â	â	â
Small a, tilde	ã	ã	ã
Small a, dieresis	ä	ä	ä
Small a, ring	å	å	å
Small ae ligature	æ	æ	æ
Small c, cedilla	ç	ç	ç
Small e, grave accent	è	è	è
Small e, acute accent	é	é	é
Small e, circumflex accent	ê	ê	ê
Small e, dieresis	ë	ë	ë
Small i, grave accent	ì	ì	ì
Small i, acute accent	í	í	í
Small i, circumflex accent	î	î	î
Small i, dieresis	ï	ï	ï
Small eth, Icelandic	u	ð	ð
Small n, tilde	ñ	ñ	ñ
Small o, grave accent	ò	ò	ò
Small o, acute accent	ó	ó	ó
Small o, circumflex accent	ô	ô	ô
Small o, tilde	õ	õ	õ
Small o, dieresis	ö	ö	ö
Division sign	÷	÷	÷
Small o, slash	ø	ø	ø
Small u, grave accent	ù	ù	ù
Small u, acute accent	ú	ú	ú
Small u, circumflex accent	û	û	û
Small u, dieresis	ü	ü	ü
Small y, acute accent	y	ý	ý
Small thorn, Icelandic	w	þ	þ
Small y, dieresis	ÿ	ÿ	ÿ

ضمیمه ۵

زبان JavaScript

CONTENTS

Variables and Literals

Unlike Java, JavaScript recognizes only a few types of values. They are as follows:

- Numbers, both real and integer (such as 4.156 and 39)
- Strings (such as `"this is JavaScript"`)
- Logical (Boolean) values of true and false
- Null, which is a special keyword denoting a null value

Although the number of data types is small, they are sufficient for the tasks that JavaScript performs. Notice that there is no distinction between integers and real numbers; both data types are just numbers. JavaScript does not provide an explicit data type for a date. However, there are related functions and a built-in date object that enable the Web page designer to manage dates.

Defining Variables

The naming rules for variables require that a variable name begins with a letter or underscore (_) and that the remaining characters are either numbers (0-9), uppercase (A-Z) or lowercase letters (a-z), or the underscore. Following are examples of legal variable names:

```
First_Name
t99
_name
```

NOTE

JavaScript does not check the spelling of variable names. Therefore, the programmer is responsible for the correct spelling of all names. When variables contain unexpected values, check to be sure that the names are spelled correctly.

The only other restriction on variable names is that they must not be the same as a JavaScript reserved word. Table E.1 lists the JavaScript reserved words.

Table E.1. JavaScript reserved words.

abstract	extends	int	super
boolean	false	interface	switch
break	final	long	synchronized
byte	finally	native	this
case	float	new	throw
catch	for	null	throws
char	function	package	transient
class	goto	private	true
const	if	protected	try
continue	implements	public	var
default	import	return	val
do	in	short	while
double	instanceof	static	with
else			

Not every word in Table **E**.1 is currently used in JavaScript; some are reserved for future use. The reserved words cannot be used for variable names, function names, method names, or object names.

A variable in JavaScript accepts all valid data types. There is no way to automatically force strong typing of data. In the same script, variables can be set to different data types or even mixed data types in a single declaration. The following variable declarations are all valid:

```
temperature =

temperature = "The temperature is"

temperature = "The temperature is " +
```

Because JavaScript is loosely typed, it provides several functions for the manipulation of string and numeric values. The section "Built-In Functions" discusses the `eval, parseInt,` and `parseFloat` functions.

Scope of Variables

JavaScript supports two variable scopes:

- Global variables
- Local variables

The local variable applies only within a function and limits the scope of the variable to that function. To declare a local variable, the variable name must be preceded by `var`, as shown following:

```
var MaxValue=0
```

JavaScript considers any variable declaration not preceded by `var` as a global variable. Although JavaScript permits you to use the same variable name for local and global variables, it is not a recommended practice.

TIP
To ensure that functions inherit the correct value for a global variable, declare all global variables at the beginning of the script

Literally Literals

As opposed to variables, literals represent the fixed values used in JavaScript. JavaScript supports the following literals:

- Integer literals
- Floating point literals
- Boolean literals
- String literals
- Special characters

Integers can be in decimal, hexadecimal, or octal format. A decimal integer is any sequence of digits that is not prefixed by a zero (such as 4, 89, or 157). If the integer is prefixed by a zero, it is an octal value (such as 04, 065, or 0145). An integer expressed in hexadecimal format is prefixed by 0x or 0X (such as 0xff, 0X44, or 0xAE).

A floating-point literal consists of the following components: a decimal integer, a decimal point (.), a decimal fraction, and an exponent. This format allows for both fractional literals (such as 1.23 or 44.6389) or those expressed in scientific notation (3.6E-8, .4E12, or -2.7E12). Every floating-point literal must have at least one digit and a decimal point or an exponent.

Boolean literals are straightforward. Their values are either true or false**.**

String literals can be enclosed by either single (**'**) or double quotes (**"**). The beginning and ending quote mark must be the same, as shown in the following examples:

```
"a double quoted literal"
'a single quoted literal'
```

TIP
When you write event handlers, enclose string literals in single quotes because double quotes delimit attribute values

String literals also might contain special characters to provide a limited degree of line control. Table E.2 lists the special characters and their functions.

Table E.2. JavaScript special characters.

Description	*Special character*
Backspace	\b
Form feed	\f
Newline	\n
Carriage return	\r
Tab	\t

The backslash (\) is the escape character for JavaScript. When used at the end of a line, it acts as a line continuation character. When followed by another character, it escapes that character so that the following character loses its special function. In JavaScript, the programmer uses the backslash to escape another backslash, a single quote, or a double quote.

Expressions and Operators

When literals and variables are linked by operators, the resulting statement is an *expression*. Examples of different types of expressions are the assignment of a literal to a variable or the computation of several literals using mathematical operators. JavaScript provides a rich variety of operators that enable programmers to write expressions ranging from the most simple to the very complex.

The JavaScript operators fall into the following categories:

- Assignment operators
- Arithmetic operators
- Bitwise operators
- Logical operators
- Comparison operators
- String operators

JavaScript includes both binary and unary operators. A binary operator has the format

```
operand1 operator operand2
```

For example, `9 * 7` or `temp = 24` are expressions with binary operators.

The unary operator has two formats:

```
operand operator
```

or

```
operator operand
```

Examples of expressions using unary operators are `++y` or `y++`.

Assignment Operators

The assignment operator (=) is a binary operator that assigns a value to the left operand (usually a variable) based on the value of the right operand (such as `FirstName = "John" or x = y * 9`). For simple right-hand expressions, JavaScript prefixes the assignment operator with either an arithmetic or bitwise operator. Table E.3 lists these shorthand assignment operators.

Table E.3. JavaScript shorthand assignment operators.

Shorthand operator	*Meaning*	*Example*
`x += y`	`x = x + Y`	`x +=`
`x -= y`	`x = x - y`	`x -=`
`x *= y`	`x = x * y`	`x *=`
`x /= y`	`x = x / y`	`x /=`
`x %= y`	`x = x % y`	`x %=`
`x <<= y`	`x = x << y`	`x <<=`
`x >>= y`	`x = x >> y`	`x >>=`
`x >>>= y`	`x = x >>> y`	`x >>>=`
`x &= y`	`x = x & y`	`x &= 0xC0`
`x |= y`	`x = x | y`	`x |= 0x0F`
`x ^= y`	`x = x ^ y`	`x ^= 0XFF`

NOTE

For those who are not familiar with C programming, be careful of the difference between the assignment operator (=) and the comparison operator (==)

Arithmetic Operators

According to the specified arithmetic operators, the purpose of arithmetic operators is to compute a single numerical value from the numerical values of either literals or variables. JavaScript supports the standard arithmetic operators of addition (+), subtraction (-), multiplication (*), and division (/). It also includes operators for modulus (%), increment (++), decrement (--), and unary negation (-).

The modulus operator (%) is a binary operator that returns the remainder of the integral division of *operand1* by *operand2*. For example, the result of `27 % 6` is `4`.

The increment unary operators add one to the operand, while the decrement unary operators subtract one from it. However, the value returned depends on the order of the operator and the operand. If the operator is prefix (`++x` or `--x`), the value returned is x+1 or x-1, accordingly. When the operator is postfix (`x++` or `x--`), the value returned is `x` before it is incremented or decremented.

The other special unary arithmetic operator is the unary negation operator. It reverses the sign of the value assigned to a variable. For example, if `x = -7`, `-x` changes the value to 7.

Bitwise Operators

For those programmers who need to fiddle with bits, JavaScript provides a set of bitwise operators. For these operators, JavaScript converts the operand into a 32-bit integer before it performs the operation specified by the operator. The bitwise logical operators are

- Bitwise AND (&), which returns the results of the logical AND between each pair of bits. For example, `0x0f & 0x0a` returns `0x0a`.
- Bitwise OR (`|`), which returns the results of the logical OR between each pair of bits. For example, `0x05 | 0x0a` returns 0x0f.
- Bitwise XOR (^), which returns the results of the logical exclusive OR between each pair of bits. For example, 0x0f ^ 0x0a returns 0x05.

JavaScript also provides a set of bitwise shift operators that shift the bits of *operand1* by the amount specified in *operand2*. These operands are

- Shift left `(<<)`, which rotates the bits to the left by the amount specified. Excess bits shifted off to the left are discarded, while zero bits are shifted in from the right. For example, `0x0f << 2` returns `0x3c`.
- Sign propagating shift right (>>) keeps the value of the sign while shifting bits to the right by the amount specified. Excess bits shifted to the right are discarded, while, excluding the sign bit, zero bits are shifted in from the left. For example, `10 >> 2 r`eturns 2 and -`10 >> 2` returns -2.
- Zero-fill right shift (>>>) does not preserve the sign bit while shifting bits to the right by the amount specified. Excess bits shifted to the right are discarded while zero bits are shifted in from the left. For positive numbers, the results are the same as the sign propagating shift right. However, for negative numbers, the sign is lost.

Logical Operators

The logical operators require that the operands are Boolean values (true or false) and they return a logical value. In the case of logical operators, the operands are expressions that evaluate to a logical value. The logical operators are

- Logical AND (`&&`)
- Logical OR (`||`)
- Logical NOT (`!`)

The logical NOT operator is a unary operator that reverses the Boolean value of the expression.

NOTE
JavaScript performs a short-circuit evaluation on logical operations so that `false&&expr` resolves to false and `true||expr` resolves to true. Under these cases, `expr` is not evaluated

Comparison Operators

The comparison operators apply to comparisons between numerical and string values and not to Boolean values. Both operands must be of the same type: numbers compared to numbers or strings compared to strings. The result of a comparison, however, is a Boolean value. The comparison operators are

- Equal (==)
- Not equal (! =)
- Greater than (>)
- Greater than or equal to (>=)
- Less than (<)
- Less than or equal to (<=)

JavaScript also supports the conditional expression that takes the form

```
(condition) ? true_value : false_value
```

If the condition is true, the expression has the value of `true_value`. Otherwise, it has the value of `false_value`. Like its cousins in other C-based languages, the conditional expression is like a standard expression and can be used anywhere, as shown in the following:

```
battery_status = (voltage > 1.3) ? "good" : "weak"
```

String Operators

The string operator (+) concatenates two string values and returns a string that is a union of the two values. For example, the expression

```
"Java" + "Script"
```

returns

```
"JavaScript"
```

The shorthand operator += concatenates the string in the left operand with that in the right operand and assigns the new value to the left operand.

Order of Precedence

In complex expressions involving more than one operator, the precedence of the operators determines the order of evaluation. By using parentheses, the programmer overrides these rules. Table E.4 shows the order of precedence from lowest to highest.

Table E.4. JavaScript operator precedence from lowest to highest.

Description	*Operators*
Assignment	= += -= *= /= %= <<= >>= >>>= &= ^= |=

Conditional	?:
Logical OR	||
Logical AND	&&
Bitwise OR	|
Bitwise XOR	^
Bitwise AND	&
Equality	== !=
Relational	< <= > >=
Bitwise shift	<< >> >>>
Addition/subtraction	+ -
Multiply/divide	* / %
Negation/increment	! ~ - ++ -
Call, member	() []

Control Statements and Functions

To make a page dynamic and interactive, the Web page developer needs statements that control the flow of information. Depending on computation results or input from the users, the script makes decisions that alter the path of execution. This section covers the conditional and loop statements provided in JavaScript.

Although all of the code could be written in the event handlers, this is not good programming practice. Instead, by using functions, code becomes modularized and reusable. A discussion of the function statement and a more formal discussion of comments round out this section. There are some details of the function statement that apply to objects; the next section covers these details and other control statements related to objects.

Conditional Statements

In addition to the conditional expression discussed in the previous section, JavaScript has one conditional statement-the `if` statement. The syntax of the conditional statement is as follows:

```
if (condition) {
   statements1 }
[else {
   statements2}]
```

The `condition` is any JavaScript expression that evaluates to the Boolean type, either true or false. The conditional `statement` is a JavaScript statement (including another `if`) or JavaScript expression. The following is an example of a valid `if` statement:

```
if (n>3) {
   status = true
   if (j != n) j = 0 }
else j = n
```

CAUTION
C programmers beware of the JavaScript rules for condition evaluation: A numerical condition evaluating to zero is not the equivalent of a Boolean true in JavaScript. Conversely, a non-zero number is not the same as a Boolean false. In JavaScript, the result of the condition must be a Boolean data type

When there is only a single statement, the braces are not necessary. For example, the following is a legal statement:

```
if (a==b) j=0
else j=1
```

Loop Statements

JavaScript supports two loop structures: the `for` statement and the `while` statement. For control within the loop structure, JavaScript provides the `break` and continue statements.

The for Statement

The JavaScript `for` statement is the same as the one for Java and C. The `for` statement repeats a loop until the specified condition evaluates to true or the loop is exited by a `break` statement. The syntax of the `for` statement is as follows:

```
for ([initial-expression;] [condition;] [increment-expression]) {
   statements
}
```

The order of processing for the `for` statement is as follows:

1. The interpreter executes the `initial-expression`. This expression initializes any values needed for loop control.

2. The interpreter then checks the `condition`. If it is true, control passes to the next step. If it is false, control goes to the next statement after the loop.
3. The interpreter then executes the `increment-expression`, which updates the variables used for loop control.
4. The statements then are executed and, unless a `break` or `continue` statement is encountered, control returns to step 2.

Listing E.2 is an example of a `for` statement.

Example of a JavaScript `for` loop.

```
<HTML>

<HEAD>

<SCRIPT LANGUAGE="JavaScript">

<!-- hide script

for (i=1; i<=10; i++) {

   sq=i*i

   document.write("number: " + i + "square: " + sq + "<BR>")

}

// end script hiding -->

</SCRIPT>

</HEAD>

<BODY>

</BODY>

</HTML>
```

The `while` Statement

The `while` statement continues to repeat the loop as long as the `condition` is true. The syntax for the `while` statement is as follows:

```
while (condition) {
```

```
   statements
}
```

The condition test occurs when the loop in the `while` statement is first executed and at the end of every loop. When the test returns false, control passes to the next statement after the loop. The for statement turning into a `while` loop is shown in Listing E.3.

Listing E.3. Example of a JavaScript while loop.

```
<HTML>
<HEAD>
<SCRIPT LANGUAGE="JavaScript">
<!-- hide script
i=1
while (i<=10) {
   sq=i*i
   document.write("number: " + i + "square: " + sq + "<BR>")
   i++
}
// end script hiding -->
</SCRIPT>
</HEAD>
<BODY>
</BODY>
</HTML>
```

The `break` Statement

The `break` statement terminates the `for` or `while` loop and returns control to the next statement following the terminated loop. The following example illustrates how to use the `break` statement:

```
i=0
while (i<10) {
   if (i==3)
      break
   i++
}
```

The `continue` Statement

Like the `break` statement, the `continue` statement terminates the current iteration of a `for` or while loop; it does not exit the loop. Where it picks up the next iteration depends on the type of loop.

- In a `while` loop, control passes to the `condition`.
- In a `for` loop, it passes to the `increment-expression`.

The following shows an example of how to use the continue statement:

```
i=0
while (i<10) {
   if (i==3)
      continue
   i++
}
```

The `function` Statement

A `function` is a group of JavaScript statements that performs a specified task. This function can then be called from any point in the document and plays an important role in writing event handlers. The format of the function statement is as follows:

```
function FunctionName(argument list) {
   statements
}
```

The Hello World script example turns into a function like the one shown in Listing E.4.

Listing E.4. Hello World as a function.

```
<HTML>

<HEAD>

<SCRIPT LANGUAGE="JavaScript"

<!-- hide the script

function DisplayIt(LineToDisplay) {

   document.write(LineToDisplay + "<BR>")

}

// end hiding -->

</SCRIPT>

</HEAD>

<BODY>

<SCRIPT LANGUAGE="JavaScript"

<!-- hide it

LineToDisplay("Hello World")

// end hiding -->

</SCRIPT>

</BODY>

</HTML>
```

TIP
Because the browser reads the statements bound by <HEAD>...</HEAD> first, it is good practice to initialize all global variables and define all functions in the HEAD of the document. This prevents errors from non-initialized variables and undefined functions

The arguments to a function are any JavaScript data types or objects. The array object `functionName.argument[i]`, where `functionName.arguments.length` contains the total number of arguments passed to the function, references arguments by the declared variable names. This feature enables a variable number of arguments to be passed to the function as an argument array. Listing E.5 illustrates the use of a variable argument list.

Listing E.5. JavaScript with a variable number of arguments.

```
<HTML>

<HEAD>

<SCRIPT LANGUAGE="JavaScript">

<!-- hide it

UnorderList="UL"

function DisplayList(ListType) {            // Display Variable List

   if (ListType="OL" || ListType="UL") {    // Validate ListType

      document.write("<" + ListType + ">"      // Display type of list

      for (var i=1; i<DisplayList.arguments.length; i++)

         document.write("LI" + DisplayList.arguments[i])

      document.write("</" + ListType + ">")    // End list

      return true

   }

   else return false

}

// end hiding -->

</SCRIPT>

</HEAD>

<BODY>
```

```
<SCRIPT LANGUAGE="JavaScript"

<!-- hide it

if (DisplayList(UnorderList, "Bullet 1 text", "Bullet 2 text"))

   document.write("<P>List Display</P>")

else

   document.write("<P>Invalid List Type<p>")

// unhide it -->

</SCRIPT>

</BODY>

</HTML>
```

Listing E.5 illustrates several important features of functions. A global variable is initialized in the `<HEAD>`. The `var` statement declared `i` to be a local variable of the function. It also shows how the `return` statement can be used to ensure that a function executed properly. The return statement can also return string and numerical values, as the following illustrates:

```
    function RetExam(a, b) {

       var x=0

       x = a+b

       return x

    }


    TestResult=RetExam(5, 7)
```

Comments

JavaScript supports the two Java style formats:

The single-line comment preceded by a double slash (`//`)

The multiple-line comment preceded by a `/*` and terminated by a `*/`

The single-line comment, from C++, treats everything from the double slash to the end of the line as a comment. Thus, the following are valid examples of a single-line comment:

```
// this is a comment

if (a=b) c=1 // also a valid comment
```

The multiple-line comment follows the rules of its C equivalent and can be used to bracket any comment. The comment used to bracket the JavaScript statements, as shown in the previous examples, is an HTML comment. However, the last line of the comment needs the double slash to keep JavaScript from interpreting the line.

Fundamentals of Objects

JavaScript is an object-based language and not an object-oriented programming (OOP) language. The designers of JavaScript were not trying to create another OOP; instead, they sought to create a scripting language that provided a tool for integrating objects created with an OOP language into an HTML document. Thus, although JavaScript lacks encapsulation, inheritance, and abstraction features of C++ or Java, it has the means to access their external objects. The ability to access Java applets and plug-ins is limited in version 2.0 of Netscape Navigator, but LiveConnect and the enhanced JavaScript in version 3.0 fully support these features.

Although it lacks the Java class structure, the JavaScript class provides properties and methods for the creation objects. In addition to providing reusable objects, the object provides a means for JavaScript to implement arrays.

Objects and Their Properties

The following shows the notational system used by JavaScript to represent an object and its properties:

```
ObjectName.PropertyName
```

Without calling them objects, this chapter already used several objects and their properties: `ObjectName.arguments` and `ObjectName.arguments.length`. In addition to being referenced by name, JavaScript supports two methods of array referencing: by property name and by an index. For example, the object `mydog` has the following properties:

```
mydog.breed="small mut"

mydog.age=5

mydog.weight=25
```

The object also could be referenced as an array using the property name as an index:

```
mydog["breed"]="small mut"

mydog["age"]=5
```

```
mydog["weight"]=25
```

It also could be referenced as an array using the numerical index:

```
mydog[0]="small mut"

mydog[1]=5

mydog[2]=25
```

Defining Methods

A function associated with an object is referred to as a *method*. After defining a function in the manner described in the "Control Statements and Functions" section of this chapter, the following format associates the function with an object:

```
ObjectName.MethodName = function_name
```

The method then is referenced in the context of working with an object:

```
ObjectName.MethodName(parameters);
```

Before providing an example, I need to cover the JavaScript statements used to work with objects.

Working with Objects

The manipulation of objects requires these additional JavaScript features: for...in statement, with statement, new operator, and this keyword. Only a short description of these features is necessary, because other sections explore them in further detail.

The `for...in` Statement

The for...in statement provides a loop mechanism for iteration through all of the properties of an object. Its format is as follows:

```
for (variable in Objectname) {

   statements

}
```

The following example uses this statement to list the properties in an object and their associated values:

```
function ListProperties(obj, obj_name) {

   var result = ""

   for (var i in obj) {

      result += obj_name + "." + i " = " + obj[i] + "<BR>"

   }
```

```
    return result

PropList = ListProperties(mydog, "mydog")
```

In the preceding example, the variable `i` is the name of the property. The example then indexes the object using the property name-`obj[i]`.

The `with` Statement

In some situations the same object is referenced several times. The `with` statement establishes a default object for a bracketed set of statements. Its format is as follows:

```
with (ObjectName) {

   statements

i
```

The `Math` object provides an example of how to use the with statement:

```
var r =

var x =

with (Math) {

   r = p / (1 - cos(a))

   x = (2 * p * cos(a)) / (sin(a) * sin(a))

}
```

The `new` Operand

For a user-defined object type, the `new` operand provides a means of creating a new instance of that object type. The syntax of this operand is as follows:

```
ObjectName = new ObjectType(param1 [, param2,] É [, paramN])
```

The section "Creating New Objects" contains examples of the use of this operand.

The `this` Keyword

this refers to the current object. As the next section shows, it plays an important role in the writing of functions and methods.

Creating New Objects

Although JavaScript contains a large number of predefined objects, the developer can create additional user-defined objects. The creation of objects requires the developer to perform two steps:

Define the object type by writing a function.

Create instances of the object using the new operand.
The function defines the object type, the properties, and the methods. For example, to create an object type for dogs, the following function needs to be written:

```
function dog(breed, age, weight) {

   this.breed = breed;

   this.age = age;

   this.weight = weight;

}
```

In the preceding example, the this keyword refers to the instance of the object being created. It is important to note that the assignment of values to a property requires that the statement end with a semicolon.
Using this function, the new operand defines a new instance of the object. For example,

```
mydog = new dog("small mut", 5, 25);
```

In addition to the regular JavaScript data types (string, numeric, and Boolean), another object can be the property of an object. For example, to add a license number to the object type for a dog, the license number also could refer to another object.

```
function doglicense(owner, phone_number) {

   this.owner = owner;

   this.phone_number = phone.number;

}

AZ123 = new doglicense("John Smith", "999-9999");
```

The object type then needs to be modified to include the new information:

```
function dog(breed, age, weight, license) {

   this.breed = breed;

   this.age = age;

   this.weight = weight;

   this.license = license;

}
```

```
mydog = new dog("mixed mut", 5, 25, AZ123);
```

To reference the owner of `mydog`, the syntax would be

```
mydog.license.owner
```

NOTE
If a new property is added to an object without changing the object type, the additional property only affects that object and not all other instances of the object type

NOTE
A string variable or a string literal is a string object. String methods are associated with these objects, as discussed in the section "Built-In Objects and Functions.

Defining Arrays

Unlike other languages, JavaScript lacks an array data type. However, an equivalent function is performed by creating an object that emulates an array. The first step is to define an array object type:

```
function MakeArray(n) {

   this.length = n;

   for (var i = 1; i <= n; i++)

      this[i] = 0;

   return this

}
```

The next step is to define an instance of the `MakeArray` object type:

```
ExmpArray = new MakeArray(20);
```

When you assign values to array elements, it looks like assigning values to an array data type. The difference is that the array begins with one and not zero, because zero defines the length of the array:

```
ExmpArray[1] = "test1"

ExmpArrya[2] = "another test"
```

Built-In Objects and Functions

The objects and functions described in this section are part of the JavaScript environment and, therefore, they are browser independent. These objects and functions are as follows:

The `string` object and its associated methods

The `Math` object and its associated methods

The `date` object and its associated methods

The `eval, parseInt,` and `parseFloat` functions

Taken as a whole, these objects and their associated methods provide a powerful set of additions to the programmer's toolbox. The string and date objects extend the basic set of data types to cover most development needs. The Math object provides all the standard functions for performing complex mathematical routines.

The `String` Object

Whether a quoted string is a string variable or a string property of an object, it is a string object; everything placed between quotes is a string object. There are two ways to use a string object:

```
stringName.propertyName
stringName.methodName(parameters)
```

`String` Object Properties

The string object has only one property-length. Because it is a property, the following are all valid references:

```
StringLength = stringVariable.length;

StringLength = mydog.name.length;

StringLength = "This is a string".length;
```

`String` Object Methods

A large number of methods is associated with the `string` object. Besides the normal string manipulation functions, many of these objects wrap the string with HTML tags. The following is a list of the string object methods:

- anchor(nameAttribute). With the nameAttribute set to the value of the parameter and passed to the method, the anchor method brackets the text with the <A></A> tags. Here's an example:

```
document.write("Other Links".anchor("other_links"));
```

- `big()`. The string is wrapped with the <BIG></BIG> tags.
- `blink()`. The string is wrapped with the <BLINK></BLINK> tags.
- `bold()`. The string is wrapped with the <B></B> tags.
- `charAt(index)`. This method returns the character to the position specified by index, where index ranges from 0 to stringName.length - 1. If the index is out-of-bounds, a null string is returned.
- fixed(). The string is wrapped with the <TT></TT> tags.

- `fontcolor(color)`. The string is wrapped with the `<FONT COLOR="color"></FONT>` tags. The `color` parameter is specified in `rrggbb` format.
- `fontsize(size)`. The string is wrapped with the `<FONTSIZE=size></FONTSIZE>` tags. The size can be a value from one to seven, or it can be a relative change (+ or −) to the font size specified in the `<BASEFONT>` tag.
- `indexOf(searchValue, [fromIndex])`. This method returns the index position of the first occurrence of the string specified in `searchValue`. The optional `fromIndex` sets the starting value of the index. If the `searchValue` is not found, JavaScript returns `−1`.
- `italics()`. The string is wrapped with the `<I></I>` tags.
- `lastOf(searchValue, [fromIndex])`. Starting at the end of the string or the value specified by `fromIndex`, this method searches the string backward for `searchValue`. If the `searchValue` is not found, JavaScript returns `−1`.
- `link(hrefAttribute)`. The text is wrapped with the `<A HREF=hrefAttribute></A>` tags. The parameter `hrefAttribute` includes all valid URLs.
- `small()`. The string is wrapped with the `<SMALL></SMALL>` tags.
- `strike()`. The string is wrapped with the `<STRIKE></STRIKE>` tags.
- `sub()`. The string is wrapped with the `<SUB></SUB>` tags.
- `substring(indexA, indexB)`. This method returns the subset of `string`, as specified by `indexA` and `indexB`. As with all indexes, the string begins at `0` and ends at `stringName.length -1`. If `indexA` is less than `indexB`, the string returned begins at `indexA` and ends at `indexB−1`. If `indexA` is greater than `indexB`, the string returned begins at `indexB` and ends at `indexA−1`. When `indexA` equals `indexB`, JavaScript returns a null string.
- `sup()`. The string is wrapped with the `<SUP></SUP>` tags.
- `toLowerCase()`. This method converts the string to lowercase and returns it.
- `toUpperCase()`. This method converts the string to uppercase and returns it.

The `Math` Object

The `Math` object provides a set of standard mathematical values and methods that augment the set of mathematical operators provided with JavaScript. As opposed to other objects, the `Math` object does not require an instance of the object before using the math methods. To simplify the entering of names and to make the script more readable, the `Math` methods often are bounded by the `with` statement. The syntax for the `Math` object is as follows:

1. `Math.propertyName`
2. `Math.methodName(parameters)`

The `Math` Object Properties

The `Math` object provides eight properties. These properties define various mathematical constants. Table E.5 describes the properties and gives their approximate values.

Table E.5. The `Math` object properties and their values.

Property	*Description*	*Approx. value*
`E`	Euler's constant	2.718
`LN2`	Natural logarithm of 2	0.693
`LN10`	Natural logarithm of 10	2.302
`LOG2E`	Base 2 logarithm of e	1.442
`LOG10E`	Base 10 logarithm of e	0.434
`PI`	Ratio of circumference to diameter	3.14159
`SQRT1_2`	Square root of one-half	0.707
`SQRT2`	Square root of two	1.414

`Math` Object Methods

The `Math` object provides a number of `Math` methods that augment the set of mathematical operators:

- `abs(number)`. This method returns the absolute value of the number.
- `acos(number)`. The `number` must be a value between `-1` and `1`. For a valid `number`, `acos` returns arc cosine in radians. If the value of `number` is outside of the range of `-1` to `1`, `acos` returns `0`.
- `asin(number)`. The `asin` method returns the arc sine in radians. The rules for evaluation are the same as those for the `acos` method.
- `atan(number)`. The `atan` method returns the arc tangent in radians. The rules for evaluation are the same as those for the `acos` method.
- `ceil(number)`. This method returns the next integer greater than or equal to `number`.
- `cos(number)`. The `number` is the angle in radians. JavaScript returns the cosine.
- `exp(number)`. This method returns `e` to the power of `number`, where `e` is Euler's constant.
- `floor(number)`. This method returns the next integer that is less than or equal to the `number`.
- `log(number)`. The `log` method returns the natural logarithm (base e) of `number`, where `number` is any positive numeric expression or object. If the number is outside the range, the return value is always `1.797693134862316e+308`.
- `max(number1, number2)`. This method returns the greater of the two numbers.
- `min(number1, number2)`. Conversely, this method returns the lesser of two numbers.
- `pow(base, exponent)`. This method raises `base` to the power of `exponent`. If either the `base` or the `exponent` is an imaginary number, JavaScript returns zero.

- `random()`. This method is available only on UNIX platforms, where it returns a pseudorandom number between zero and one. In Netscape Navigator 3.0, the `random()` method is available on all platforms.
- `round(number)`. JavaScript returns the value of `number` rounded to the nearest integer. For the purpose of rounding, any value of `.5` or greater is rounded to the next highest integer.
- `sin(number)`. This method returns the sine of the angle, where `number` is expressed in radians.
- `sqrt(number)`. The `sqrt` method returns the square root of any non-negative number. If the `number` is out of range, JavaScript returns zero.
- `tan(number)`. This method returns the tangent of the angle, where `number` is expressed in radians.

The `Date` Object

Although JavaScript does not provide a `Date` data type, it provides a `Date` object that allows the handling of date and time information. One caveat is that all dates are in milliseconds from January 1, 1970, 00:00:00. Due to the way JavaScript handles dates, dates before 1970 are invalid dates.

The `Date` object requires that an instance of the `Date` object exist prior to the use of its methods. The instance can either be a new object or the property of an existing object. There are four ways to define the new instance:

```
dateObjectName = new Date()

dateObjectName=new Date("month day, year hours:minutes:seconds")

dateObjectName=new Date(year, month, day)

dateObjectName=new Date(year, month, date, hours, minutes, seconds)
```

Format one sets the date and time to the current date and time. Leaving out the time sets the value to zero. Because the `Date` object does not contain any properties, there is only one format for the `Date` methods:

```
dateObjectName.methodName(parameters)
```

The exceptions are `UTC` and `parse` methods, which are static methods and use

```
Date.UTC(parameters)

Date.parse(parameters)
```

Table E.6 describes the values returned by the various `get` commands.

Table E.6. Extracting information from the `Date` object.

`Date` *method*	*Returned value*
`getDate()`	Day of the month
`getDay()`	Day of the week
`getHours`	Hour of the day
`getMinutes`	Minutes in the hour
`getMonth`	The month
`getSeconds`	Seconds in the minute
`getTime`	Milliseconds since 1/1/1970
`getTimezoneOffset`	Offset between local time and GMT
`getYear`	The year

Besides being able to retrieve information on the `Date` object, the methods in Table E.7 show how to change date information.

Table E.7. Setting information into the `Date` object.

Date method	*Valid values*
`setDate(dayValue)`	1-31
`setHours(hoursValue)`	0-23
`setMinutes(minutesValue)`	0-59
`setMonth(monthValue)`	0-11
`setSeconds(secondsValue)`	0-59
`setTime(timeValue)`	>=0
`setYear(yearValue)`	>=1970

Two additional methods are used to convert the date to a string value. They are as follows:

- `toGMTString()`. This method converts the date in GMT (Greenwich Mean Time) and returns it as a string. The actual format of the string is dependent on the hardware platform.
- `toLocaleString()`. This method converts the string to the locale format, which varies according to locale.
- The `Date` object also provides two static methods for handling strings; it has the format of `Date.method()`. These methods are as follows:
- `parse(dateString)`. This method converts a date string into the number of milliseconds since January 1, 1970, 00:00:00. Local time is the default. However, it also supports a suffix specifying the GMT offset or U.S. standard time zones.
- `UTC(year, month, day [, hrs] [, min] [, sec])`. This method returns the number of milliseconds since January 1, 1970, 00:00:00 Universal Time Coordinate (in other words, GMT).

Built-In Functions

JavaScript supports several built-in functions that are not related to any object. These built-in functions are as follows:

- `eval(string)`. The `eval` function evaluates the `string`, which contains any JavaScript expression, statement, or sequence of statements, and returns the result. Some examples of the `eval` function are

```
var x =

var y =

var z = "if (x <= 9) (x*y) else (x/y);"

document.write(eval("x + y / 4"), "<BR>")

document.write(eval(z), "<BR>")
```

- `parseFloat(string)`. This function parses `string` and returns a floating point number. If the first character is not a number or sign, it returns zero on Windows platforms and `NaN` (not a number) on other platforms. Parsing of the string continues until it reaches the end of the string, or a character that is not a sign, number, decimal point, or exponent.
- `parseInt(string [, radix])`. The `parseInt` function parses the string and returns an integer according to the specified radix. The value of radix is 8 for octal, 10 for decimal, and 16 for hexadecimal. If the radix is not specified, a string that begins with `0x` is radix 16, `0` is radix 8, and any other value is radix 10. If `parseInt` encounters a character that is not a numerical value in the specified radix, it stops parsing and returns the value to that point. If `string` begins with a non-numerical value, it follows the same rules as `parseFloat`.
- `isNaN(testValue)`. This function exists only on UNIX platforms and evaluates `testValue` to determine whether it is a `NaN`. It returns either true or false.
- `escape("string")`. This function returns the ASCII encoding of an argument in the ISO Latin-1 character set. The `string` is a non-alphanumeric string or property of existing object. It returns the value as `%xx`, where `xx` is the ASCII encoding of a character in the argument. If it encounters an alphanumeric value, it passes the value to the output string without encoding it.
- `unescape("string")`. This function returns the ASCII character for the `%xx`, or hexadecimal values specified in the `string` parameter.

Netscape Objects

In addition to the JavaScript objects and methods, the Web page developer has access to the objects and methods in the Netscape browser. This section reviews the Netscape Navigator objects and methods. The next section looks at how to use the objects and methods in the management of windows and frames.

The Navigator Object Hierarchy

Netscape Navigator builds an instance hierarchy that reflects the document being processed. As an instance hierarchy, there are no classes as defined in Java. However, the instance hierarchy works with the JavaScript approach to objects. Figure E.1 shows the structure of the Navigator object hierarchy.

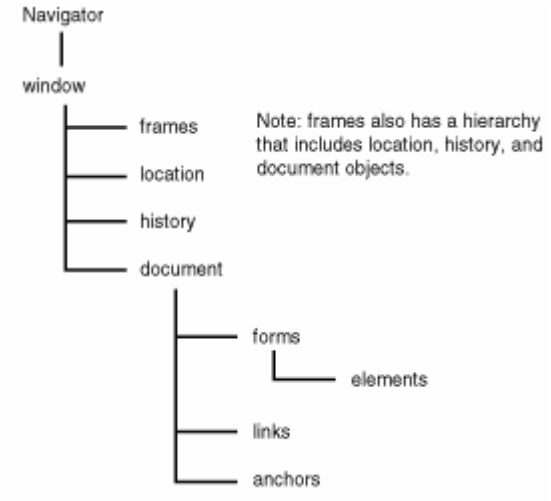


Figure E.1 : *The Netscape Navigator object hierarchy.*

This hierarchy is important for creating references to objects and their properties. The children of an object are properties of the parent object. Because all objects are descendants of the `window` object, the `window` object itself is not referenced when referencing any object or property within the current window. For example, the reference to the instance of `myform` is `document.myform`. However, referencing a document in another window requires the addition of the window name to the object reference.

The Importance of HTML Layout

The proper use of JavaScript depends on having a basic understanding of how Netscape Navigator processes an HTML document. When an HTML document is loaded, the browser starts processing at the first line of the document. The browser lays out the screen in the order of the HTML statements in the document. After the screen is drawn, it cannot be redrawn without processing a new document. How this affects frames and windows is the topic of the next section.

The corollary to this is that an instance of an object exists only after encountering the HTML that generates the instance. Thus, JavaScript cannot reference an HTML object, such as a form, or execute a function until the browser processes the statement. For example, JavaScript cannot reference a `form` object until the browser processes the HTML for the form. Similarly, the changing of a property after the browser used the property in the layout of the window does not affect its value or appearance.

Although the constraints seem onerous, understanding the behavior of processing HTML and JavaScript saves a lot of frustration. The key principle to remember is that an HTML document is sequentially processed and JavaScript is part of that sequential process.

The Window Object

The `window` object is the parent of all objects. It includes the properties that define all windows and frames within a window. When the browser initially loads the HTML document, it starts with a single instance of a `window` object. If the HTML document creates frames, the frame information is stored in a `frame` array object. By the same token, opening a new window creates a child `window` object. The power of JavaScript lies in its capability to utilize the properties and methods of the window object.
The section "Windows and Frames" goes into more detail about the `window` object itself. This section covers the objects that are properties of the `window` object. These objects are the

- `Location` object
- `History` object
- `Document` object

Of the three, the document is the most important object in the hierarchy. The `document` object itself contains properties that refer to other objects. The most important of these are the following:

- `form` object
- `anchor` object
- `link` object

The upcoming sections cover each of these objects. However, they cover only those objects, properties, and methods that are related to JavaScript.

The `location` Object

The `location` object contains information about the current URL. The reference to the object is as follows:

```
[windowReference.]location[.propertyName]
```

The properties of the `location` object refer to the individual parts of the URL:

```
protocol//hostname:port pathname search hash
```

NOTE
The `location` object and the `location` property of the `document` object (`document.location`) have different purposes. The `location` object can be changed, but the `location` property cannot be changed

The properties of the `location` object are as follows:

- `protocol`. The `protocol` specifies the access method of the URL and includes everything up to the first colon.
- `hostname`. The `hostname` contains the host and domain name, or IP address, of the destination host.
- `port`. The `port` is the TCP/IP port as discussed in Chapter 1 "An Overview of Internet Programming." If the `port` property is empty, it defaults to the port specified for the protocol as defined in the services file.
- `pathname`. The `pathname` specifies the path to the specified resource on the destination host.
- `search`. The `search` property is a string that begins with a question mark and is used for CGI scripts.
- `hash`. The `hash` property is a string that begins with a hash mark (#) and specifies an anchor name.
- `href`. This property specifies the entire URL. If reference is made to `[windowName.]location`, the `href` property is assumed.
- `host`. This property is equivalent to `hostname:port`.

NOTE
Although JavaScript permits the modification of the individual properties, sound programming practice calls for changing the `href` property. This approach prevents any errors resulting from the browser attempting to access the URL before all changes are made

The `History` Object

Access to the `History` object is a controversial subject because it enables the script to send the history back to the server. To prevent this from being misused, Netscape Navigator versions 2.01 and above no longer enable access to this object.

The `document` Object

The `document` object holds the properties, objects, and methods that define the presentation of the document. It refers to that part of the HTML document defined by the `<BODY></BODY>` tags. The following subsections discuss the components of the document object, except for the `form` object (which is covered in the "The `Form` Object" section, later in this chapter).

The `document` Object Properties

The HTML options to the `<BODY>` tag define the `document` object properties. JavaScript references all of these properties, except for the background image.

NOTE
The string required to change the color properties is in the format of `document.colorProperty = "#RRGGBB"` or `document.colorproperty="colorName"`. Any color property defined in the

<HEAD></HEAD> tags takes precedence over the <BODY> tag option. The outcome of color changes made in the <BODY> depends on the color property

The color document object properties are as follows:

- bgColor. This property defines the background color of the document. The bgColor property immediately updates the display.
- fgColor. This property defines the text color of the document. After the browser completes the layout of the HTML document, the browser ignores changes to this property. Instead, the <FONT> tag or the fontcolor method provide an alternative mechanism for changing the text color.
- linkColor. The linkColor represents the color of a link defined by HREF, without prior visits. As with all color involving the links, the colors change after the user selects the link.
- alinkColor. This property controls the color of an active link. In other words, it is the color of the link after it is selected and before the destination host replies.
- vlinkColor. After the user visits a site, the browser displays this color for the link.
- The document object also contains the following non-color related properties:
- lastModified. This read-only property reflects the date that the document was last modified.
- location. This read-only property usually matches the value of the location object unless redirection alters the URL.
- referrer. This read-only property contains the URL for the document that is linked to the current document.
- title. A read-only property that contains the value specified by the <TITLE></TITLE> tags.

The anchors Object

The anchors object contains an array of all anchors declared by the NAME attribute of the <A> </A> tags. The array begins at 0 and continues through document.anchors.length - 1. The value of document.anchors[index] is null.

TIP
Before using it to set a value such as location.hash, it is possible to check the validity of the anchor by comparing it to the array length; you use sequential numbers to identify anchors

The link Object

The link array contains the link objects defined by the <A></A> tags or the link method. The array includes objects for both the HREF and NAME attributes. With the addition of the TARGET attribute, the properties of each link object are identical to those of the location object.

NOTE
The link array is a read-only array. Additional entries are added via the <A></A> tags. The link method modifies existing entries in the link array

The `link` object provides two event handlers: `onClick` and `onMouseOver`. The section called "The `Form` Object," later in this chapter, describes how to use these event handlers.

The `cookie` Property

The `cookie` property contains a string value of the cookie entry from the `cookies.txt` file for the document. For a complete description of how to use cookies, see the Netscape cookie specification. The substring, `charAt`, `indexOf`, and `lastIndexOf` string methods can be used to dissect the cookie string. Chapter 18 contains an example of how to read the cookie string.

The `document` Object Methods

The `document` object contains five methods:

- `document.write()`
- `document.writeln()`
- `document.open()`
- `document.close()`
- `document.clear()`

As shown in previous examples, the `document.write` method, without a window reference, writes text to the current window. The `document.writeln()` method is the same as `document.write`, except that it inserts a newline character at the end of the argument. The format for these methods is as follows:

```
document.write(expression [, expression2] ... [expressionN])

document.writeln(expression [, expression2] ... [expressionN])
```

The default MIME type is `text/html`. However, the `document.open(["mimetype"])` method enables the opening of other MIME types, such as text/plain, image/gif, image/jpeg, image/x-bitmap, and plugIn. The `document.open()` method opens a stream to collect the output of the `write` and `write.ln` methods. If the MIME type is text or image, the browser opens a stream for layout; for plugIn, the browser opens it to a plug-in. If a document already exists in the target window, the `open` method clears it.

NOTE
Currently, it is not possible to print any text generated by JavaScript via the `write` or the `writeln` methods

The stream stays open until the browser encounters a `document.close()` method. The `document.close()` forces the content of the stream to display. The `document.clear()` method clears the content of the window.

The Form Object

The HTML <FORM></FORM> tags provide the means for user input of data, and output of variable data to the user. The user input might affect choices for client-side use, or it can be sent to the server. On the other hand, variable data such as marquees can be displayed on the form. On the input side, event handlers provide a means to invoke JavaScript routines to perform such tasks as editing data. On the output side, JavaScript plays a bigger role in the managing of data to be displayed in the form.
This section discusses the form object and its properties. Due to the length of forms examples, this section only presents snippets of code. The next chapter gives full-blown scripts illustrating the use of forms and event handlers.

Event Handlers

Event handlers constitute one of the major uses of JavaScript. Events are the result of user actions, such as the clicking of a mouse button, the checking of a box, or the submission of a form. Event handlers are defined in the HTML tags along with the JavaScript related to the event. The following example illustrates the coding of an event handler:

```
<INPUT TYPE="button" VALUE="Submit" onClick="validate(this.form)">
```

In the preceding example, the keyword `this` refers to the current object, which is the button object. By stating `this.form`, the reference is made to the `form` object containing the button. While the preceding example executes a function, JavaScript statements also are valid. When there is more than one statement, each statement must be separated by a semicolon.

TIP
Good programming practice calls for the use of functions because they make code easier to read and can be reused

NOTE
Until an HTML document is loaded into a window that contains the <BODY></BODY> tags, a window contains no event handlers

The following is a list of event handlers supported by JavaScript:

- `onBlur`. The JavaScript for this event handler executes when the user leaves the field causing it to lose focus.
- `onChange`. The browser executes this JavaScript for the event when the user leaves the field and the value of the object changes.
- `onClick`. The `onClick` event occurs when the user clicks on a form or link.

- onFocus. This event occurs when a field receives input focus by tabbing to the field or clicking with the mouse.
- onLoad. This event occurs when the browser finishes loading a document or all frames within the <FRAMESET> tag.
- onMouseOver. This event occurs when the mouse moves over an object from outside the object. The JavaScript routine for the event handle must return true for the status and defaultStatus properties to be set.
- onSelect. An onSelect event occurs when a user selects text within a text or textarea field.
- onSubmit. When the user submits a form, an onSubmit event occurs. If the JavaScript returns false, the form is not submitted. Any other value, including no return statement, enables the form to be submitted.
- onUnload. This event occurs when a document is exited.
- The event handlers are a part of various objects. Some objects support more than one event handler, and some event handlers appear in multiple objects. Table E.8 shows the relationship between event handlers and objects.

Table E.8. The relationship between event handlers and objects.

Object	*Event handlers*
button	onClick
checkbox	onClick
form	onSubmit
link	onClick, onMouseOver
radio	onClick
reset	onClick
select	onBlur, onChange, onFocus
submit	onClick
text	onBlur, onChange, onFocus, onSelect
textarea	onBlur, onChange, onFocus, onSelect
window	onLoad, onUnload

The forms Array

The forms array contains an entry for each form object created by the <FORM></FORM> tags. For JavaScript, it is a read-only array composed of the following properties:

- action. This property contains the value of the ACTION attribute.
- element. This is an array of element objects defined for the form.
- encoding. This property contains the value of the ENCTYPE attribute.
- length. This property contains the number of entries in the element array.
- method. This property contains the value of the METHOD attribute.

- `target`. This property contains the value of the `TARGET` attribute.

The following are the valid means of addressing the form objects:

```
formName.propertyName

formName.methodName(parameters)

forms[index].propertyName

forms[index].methodName(parameters)
```

`Form` Object Methods

The `form` object has only one method-`submit`. The `submit` method performs the same action as the submit button of an HTML form and has the following syntax:

```
document.formName.submit()
```

The `element` Objects

The `element` objects reflect the element entries in the `<FORM></FORM>` tags. Table E.9 lists the element objects and their properties.

Table E.9. The properties of `element` objects.

Element object	*Properties*
`button`	`name`, `value`
`checkbox`	`name`, `value`, `checked`, `defaultChecked`
`hidden`	`name`, `value`
`password`	`name`, `value`, `defaultValue`
`radio`	`name`, `value`, `checked`, `defaultChecked`, `length`
`reset`	`name`, `value`
`select`	`name`, `length`, `options` array, `selectedIndex`
`submit`	`name`, `value`
`text`	`name`, `value`, `defaultValue`
`textarea`	`name`, `value`, `defaultValue`

The properties are addressed as `document.elementName.property`, or as `document.formName. elements[index].propertyName`, where `elementName` is the value of the `name` property for the `element` object.

The `element` Methods

The `element` methods emulate their cousins, the event handlers. However, there are some caveats, as the following shows:

- `blur()`. This method removes focus from the specified object but does not establish the focus on another object.
- `click()`. This method simulates a mouse click on the specified object. When referencing the radio element, the form is `document.radioName[index].click()`.

- `focus()`. This method gives focus to the specified object.
- `select()`. This method selects the entire input area.

Except for the `radio` object, the methods are addressed as `document.elementName.methodname()`. Table E.10 lists the element objects and their corresponding methods.

Table E.10. The methods of `element` objects.

Element object	*Methods*
`button`	`click`
`checkbox`	`click`
`hidden`	(has no methods)
`password`	`blur`, `focus`, `select`
`radio`	`click`
`reset`	`click`
`select`	`blur`, `focus`
`submit`	`click`
`text`	`blur`, `focus`, `select`
`textarea`	`blur`, `focus`, `select`

Windows and Frames

Windows and frames create more confusion for Web page developers than any other aspect of the browser. When Netscape Navigator starts, it opens a window and, depending on how the options are set, loads a document into the window. If you select the menu option File | New Web Browser, a new window is opened. In this case, closing the original window does not close the new window.

On the other hand, frames are created according to the `<FRAMESET></FRAMESET>` tags in the HTML document. It subdivides the screen's real estate into a number of frames. When the document that defined the frames is closed, the frames go away because their existence depends on the document.

Chapter 18 provides examples that illustrate the difference in behavior between windows and frames, even though both are containers for documents. This section concerns itself with the properties and methods of the `window` and `frame` objects.

The `Window` Object Properties

One of the major features of JavaScript is its capability to create and manipulate windows. These windows are not limited to just displaying messages; depending on the parameters set, they can be another instance of the browser. The following properties of the `window` object reflect the flexibility of the browser window:

- `defaultStatus`. The `defaultStatus` is the message that appears in the status bar when no other message is being displayed. If it is set from a `onMouseOver` event handler, the event handler must return true for the status to change.

- `frames`. This property is an array that contains the `frame` objects. The frame inherits all of the properties and methods of the `window` object.
- `length`. The value of this property is the number of frames in the frame array.
- `parent`. From a frame reference, this is the window that the frameset resides in. A frame within the frameset can reference another frame in the frameset by using `parent.frames[index]` without having to reference the window by name.
- `self`. This is a synonym for the current window or frame.
- `status`. This is a transient message that is set by the `onMouseOver` event handler.
- `top`. This property is used to reference the topmost window. It can be used by child windows or embedded filesets to reference the originating window.
- `window`. This property is a synonym for the current window.
- The forms for referencing window properties are

```
window.propertyName

self.propertyName

top.propertyName

parent.propertyName

windowVar.propertyName

propertyName
```

The `Window` Object Methods

The following are the `window` or `frame` object methods:

- `alert("message")`. This method creates an alert dialog box with a single OK button. It is used to display a message that does not require a user decision.
- `close()`. This method closes the referenced window. It must contain a window reference such as `window.close` as `close()` with no reference is the equivalent of `document.close`.
- `confirm("message")`. The confirm method displays a confirm dialog box with OK and Cancel buttons. OK returns a value of true, and Cancel returns a value of false.
- `[windowVar = ][window.]open("URL", "windowName" ["windowFeatures"]`. This method opens a new Web browser window. The object name `windowVar` is the name of the new window and is used to reference its properties and methods. The URL specifies the URL to open in the new window. If the option is null, a blank window is opened. The variable `windowName` is

the name used in the TARGET attribute of the <FORM> and <A> tags. The variable windowFeatures is a comma-separated list of the following options:

```
toolbar=yes|no

location=yes|no

directories=yes|no

status=yes|no

menubar=yes|no

scrollbars=yes|no

resizable=yes|no

width=pixels

height=pixels
```

- If no features are set, all features default to true. If any feature is explicitly set, all features default to false. If a feature is set without specifying the value, the value is true. These features refer to the components of the Navigator window. Thus, location refers to the location entry field and directories refers to the standard Navigator buttons.

NOTE
After it is created, the window is independent of the parent window; if the parent window closes, the window created remains open. The onUnLoad event handler closes the windows created

- prompt("message" [, inputDefault]). The prompt method displays a prompt dialog box with a message, an input field, an OK button, and a Cancel button. The inputDefault is a string, integer, or property of an object that represents the default value for the input field. If the inputDefault is not specified, the input field displays the value <undefined>.
- timeoutID=setTimeout(expression, msec). With this method, the evaluation of the expression is delayed for the number of milliseconds specified. The timeoutID is only used by the clearTimeout method.
- clearTimeout(timeoutID). This method cancels the time-out set by the setTimeout method.

The preceding methods are referenced as follows:

```
window.methodName(parameters)
```

```
self.methodName(parameters)

top.methodName(parameters)

parent.methodName(parameters)

windowVar.methodName(parameters)

methodName(parameters)
```

CAUTION

The `open()` and `close()` methods need to be referenced as `window.open()` and `window.close()` to avoid any conflicts of scope with `document.open()` and `document.close()`

Dividing the Window into Frames

Frames divide a window into multiple, independently scrollable frames on a single screen. Frames are created via the `<FRAMESET></FRAMESET>` tags in an HTML document. Each document creates a frame array for that document. If a document opened in one of the frames contains a `<FRAMESET>` tag, that frame is divided into frames by that document. This hierarchy of framesets is important in referencing the properties and methods of frames.

NOTE

Frames have all the properties of a window. The entire hierarchy for the frame structure is the same as the window structure. (Refer to Figure E.1.

The structure under any window or frame can be referenced. Thus, object properties in one window or frame can change object properties in another window or frame using the structure shown in Figure E.1.

NOTE

The HTML document that uses `<FRAMESET></FRAMESET>` contains only frame statements. After the frames are opened, the original document is no longer visible. The HTML document is one frame that can control the other frames. Thus, the possibilities for screen management give the Web page developer tremendous freedom in the development of interactive Web documents

Summary

JavaScript is more than a simple scripting language and less than a full-blown, object-oriented programming language. It is the glue that binds many diverse elements into a single dynamic and interactive environment.

This chapter presented the fundamentals of JavaScript and described the objects and their properties that are available to JavaScript. This chapter dealt with the pieces to the puzzle, and Chap-ter 18 puts the pieces together to illustrate the uses of JavaScript.

ضمیمه ٦

زبان VBScript

CONTENTS

Visual Basic Script, herein referred to as VBScript or VBS, is Microsoft's answer to Sun's JavaScript. Both are meant to be intertwined in HTML code; both are meant to expand on the client side functionality of the Web, and neither of the two are fully released products. (VBScript is in beta with MS-Internet Explorer 3.0. JavaScript was first implemented in Netscape 2.0.)

```
http://www.microsoft.com/vbscript
```

The resource for information on Visual Basic Script is the Microsoft Web site (`http://www.microsoft.com/vbscript`). Here you can find complete documentation, some sample snippets of VBScript code, and links to sites using VBScript.

Variables

In most languages, variables are an important part of the language, its structure, and its implementation. In VBScript, the variables are all of the same type, `Variant`.

The `Variant` type is an "all things to all people" generic data type. Assign it a string, and it acts as a string; assign it a number, and it acts like a number. The following lines are an example of declaring a `Variant` in VBScript:

```
<SCRIPT LANGUAGE="VBScript">
Option Explicit
Dim vUserName, vUserNumber, vUserString
</SCRIPT>
```

The `SCRIPT LANGUAGE` tag is a proposed addition to the HTML standard and is first implemented in the Microsoft Internet Explorer 3.0 It informs the browser that code, which must be interpreted, is coming and of what type it is. The usual as Variant, (like you would see in a regular Visual Basic

program) is omitted because the `Variant` type is the only one available. Multiple variables per line are allowed-the procedure is to separate each by a comma. The `Option Explicit` keyword is implemented in Visual Basic Script, and if used, it should appear as the first statement after the `<SCRIPT LANGUAGE = "VBScript">` tag.

Naming Restrictions

The naming constraints in VBScript are similar to those of Visual Basic. Note that these rules apply to function names, constant names, and variable names. The name must begin with an alphabetic character. It cannot contain an embedded period, must be less than 255 characters, and it must be unique in the scope in which it is declared.

Scope Rules for VBScript Variables

Two scopes are available in VBScript: the script level and procedure level. Variables at the script level are instantiated at the time the script is run, and exist until the script has completed running. Procedure level variables exist from the beginning of the procedure they are declared in until the end of that procedure.

The `Static` keyword is supported when using persistent procedure level variables.

Arrays with multiple dimensions are also supported in VBScript:

```
Dim aSums(10) 'single dimension
Dim aTable (10, 10) 'two dimension
```

As you can see, the declaration is just as in Visual Basic, except again there is no `as <Variable Type>`.

The `ReDim` and `ReDim Preserve` keywords are also supported in the creation and recreation of arrays. They function in the same way as their Visual Basic counterparts.

Operators

VBScript includes all the operators found in Visual Basic: arithmetic (`+`, `-`, and so on), logical (`And`, `Or`, `Not`, etc.), and comparison (`=`, `< >`, and so on). VBScript also obeys the same rules for operator precedence as Visual Basic.

Note
Operator precedence is a set of rules that determines what order operators are evaluated in if more than one operator is present in an expression.

Control Constructs

The `If-Then-Else`, `Do-Loop`, `While-Wend`, `For-Next`, and `For Each-Next` control constructs are supported, just as they are in standard Visual Basic syntax.

Using Subroutines and Functions

Like its Visual Basic cousin, VBScript supports the `Sub` and `Function` procedures. Within the HTML file, the subroutines and function bodies must be placed before the code that calls them. It's advisable to put the VBScript functions and subroutines in the beginning of the `<HEAD>` portion of the HTML document. Then, following the subroutines and procedures, place the remainder of the VBScript code.

Visual Basic Script Statement Overview

Visual Basic Script supports a subset of the Visual Basic statements. The following list are the functions supported:

- `Call` statement
- `Dim` statement
- `Do-Loop` statement
- `Erase` statement
- `Exit` statement
- `For-Next` statement
- `For Each-Next` statement
- `Function` statement
- `If-Then-Else` statement
- `Let` statement
- `LSet` statement
- `Mid` statement
- `MsgBox` statement
- `On Error` statement
- `Private` statement
- `Public` statement
- `Randomize` statement
- `ReDim` statement
- `Rem` statement
- `RSet` statement
- `Set` statement
- `Static` statement
- `Sub` statement
- `While-Wend` statement

Visual Basic Script Functions Overview

The following is a list of functions and objects that are supported by Visual Basic Script and operate similarly to the VB counterparts:

- `Abs` function
- `Asc` function
- `Atn` function
- `Chr` function
- `Cos` function
- `Date` function

`DateSerial` function
`DateValue` function
`Day` function
`Exp` function
`Hex` function
`Hour` function
`InputBox` function
`InStr` function
`Int`, `Fix` functions
`LBound` function
`LCase` function
`Left` function
`Len` function
`Log` function
`LTrim`, `RTrim`, and `Trim` functions
`Mid` function
`Minute` function
`Month` function
`MsgBox` function
`Now` function
`Oct` function
`Right` function
`Rnd` function
`Second` function
`Sgn` function
`Sin` function
`Sqr` function
`Str` function
`StrComp` function
`String` function
`Tan` function
`Time` function
`TimeSerial` function
`TimeValue` function
`UBound` function
`UCase` function
`Val` function
`VarType` function
`Weekday` function
`Year` function

The following sections provide information regarding new functions or functions that behave differently than their VB counterparts.

The `Array` Function

The `Array` function returns a variant that contains an array. The following is an example:

```
Dim aNums as Variant
aNums = Array(1, 2, 3, 4, 5)
```

This would create an array with five values, 1 through 5. If no arguments are passed to the Array function, the array created is of zero length.

Accessing the elements of an array is done the same way in VBScript as in Visual Basic. For instance,

```
X = aNums(3)
```

would assign the value of the third position in the array aNums to the variable X.

Data Conversion Functions

The data conversion functions listed in Table F.1 take one Variant variable as an argument and return that value as the Variant subtype of that function.

For example, the CBool function would take a normal Variant as an argument and return a value into a Variant with the subtype Boolean.

Table F.1. VBScript's data conversion functions

Function	*Description*
Cbyte	Variant to subtype byte
Cdate	Variant to subtype date
OCDbl	Variant to subtype double
Cint	Variant to subtype integer
CLng	Variant to subtype long
CSng	Variant to subtype single
CStr	Variant to subtype string
CVErr	Variant to subtype error

The values passed to these conversion functions must be in the valid range for the type being converted to. For example, the argument passed to the CVErr function must be a valid error number.

The Is and Isx Functions

The Is functions determine whether the variable is of that data type. The IsArray function returns True if the Variant type variable passed to the function is of subtype array. The other Is functions (IsDate, IsEmpty, IsError, IsNull, IsNumeric, and IsObject) behave in the same way.

The Err Object's Properties and Methods

At this point, VBScript supports only the Err object. This contains any information about runtime errors during the processing of the script.

Description Property

The Description property can return or be set to a string that is displayed to the user when a runtime error occurs.

`HelpContext` Property

The `HelpContext` property returns or sets the help context ID used to find a topic in a help file.

`HelpFile` Property

`HelpFile` specifies or can be set to any path that contains the help file.

`Number` Property

This corresponds to the error number of the `Err` object. It can be read or set at runtime.

`Source` Property

The `Source` property returns or sets the object or code that caused the error.

`Clear` Method

This clears all properties of the `Err` object.

`Raise` Method

To generate a runtime error, use the `Raise` method on the `Err` object. The arguments to the `Raise` method are `number`, `source`, `description`, `helpfile`, and `helpcontext`.
These correspond to the properties of the `Err` object and enable the programmer to set these properties at the time the error is raised.

Visual Basic Script Examples

Currently, the only browser that supports VBScript is Microsoft's Internet Explorer 3.0 Beta. For the latest updates to the Internet Explorer and for VBScript examples, see Microsoft's Web site (`http://www.microsoft.com`), the Visual Basic Script page from Microsoft (`http://www.microsoft.com/vbscript/`), and the Internet Explorer page at (`http://www.microsoft.com/ie/`).
Listing F.1 provides the HTML for a quick and dirty Hello World VBScript application. If you have Internet Explorer 3, you can enter this into Notepad or load it from the CD-ROM included with this book.

Listing F.1. The Hello World VBScript example

```
<HTML>
<HEAD><TITLE>VBScript Hello World Example</TITLE></HEAD>
<BODY>

<CENTER><H2>Hello World Example</H2>
```

```
<INPUT TYPE=TEXT VALUE="Insert Name Here" NAME="txtName"><p>
<INPUT TYPE=BUTTON VALUE="Click Here" NAME="BtnHello">
</CENTER>

<SCRIPT LANGUAGE="VBScript">
<!-
Sub BtnHello_OnClick
    MsgBox "Hello, world!", 0, "From " + txtName.Value
End Sub
->
</SCRIPT>
</BODY></HTML>
```

The HTML in Listing F.1 is pretty straightforward. The first `<INPUT>` tag defines a text box with an initial value of `Insert Name Here` and a `Name` of `txtName`. The second `<INPUT>` tag defines a button named `BtnHello` that has a caption reading `Click Here` (a button's caption is defined by the `Value` element of the `<INPUT>` tag). The page produced is shown in Figure F.1.

Figure F.1: *The Hello World VBScript Web page.*

The VBScript portion begins with the <SCRIPT> tag. Note that outside of the <SCRIPT> tags, the actual script is placed within an HTML comment (<!– ... –>). This is to prevent Web browsers that are not script-compliant from attempting to interpret or display the code that is contained within the script.
The Sub defined in the script is the event procedure that is executed whenever BtnHello is clicked. This concept is identical to the event-driven nature of Visual Basic. The procedure executes the MsgBox statement. The text in the message box is Hello, world! and the title for the message box is the string From concatenated with the text contained in the txtName textbox.
A simple example, I know, but it demonstrates the ease-of-use and integration with HTML that VBScript provides.

ضمیمه ۷

کدهای وضعیت و خطاها در HTTP

Status codes and meanings.

Numeric	English	Meaning
1xx	Informational	This is not used; it is reserved for future use.
2xx	Success	The action was received, understood, and accepted.
3xx	Redirection	Further action must be taken in order to complete the request.
4xx	Client Error	The request contains bad syntax or cannot be fulfilled.
5xx	Server Error	The server failed to fulfill an apparently valid request.

Status codes for HTTP/1.0.

Code	Reason Field	Meaning
201	Created	The request has been fulfilled and resulted in a new resource being created. The newly created resource can be referenced by the URI(s) returned in the URI-header field of the response, with the most specific URI for the resource given by a `Location` header field.
202	Accepted	The request has been accepted for processing, but the processing has not been completed.
203	Non-Authoritative	The returned metainformation in the `Entity-Information` header is not the definitive set as available from the origin server, but is gathered from a local or a third-party copy.
204	No Content	The server has fulfilled the request, but there is no new information to send back.
300	Multiple Choices	The requested resource is available at one or more locations and a preferred location could not be determined via content negotiation.
301	Moved Permanently	The requested resource has been assigned a new, permanent URI; any future references to this resource should be done using one of the returned URIs.
302	Moved Temporarily	The requested resource resides temporarily under a different URI.
303	See Other	The requested resource resides under a different URI and should be accessed using a `Get` method on that resource.

304	Not Modified	If the client has performed a conditional Get request and access is allowed, but the document has not been modified since the date and time specified in the If-Modified-Since field, the server responds with this status code and does not send an Entity-Body header to the client.
400	Bad Request	The request could not be understood by the server because it has a malformed syntax.
401	Unauthorized	The request requires user authentication. The response must include a WWW-Authenticate header field containing a challenge applicable to the requested resource.
402	Payment Required	This code is not currently supported, but it is reserved for future use.
403	Forbidden	The server understood the request but is refusing to perform the request because of an unspecified reason.
404	Not Found	The server has not found anything matching the request URI.
405	Method Not Allowed	The method specified in the request line is not allowed for the resource identified by the request URI.
406	None Acceptable	The server has found a resource matching the request URI, but not one that satisfies the conditions identified by the Accept and Accept-Encoding request headers.
407	Proxy Authentication (Unauthorized) Required	This code is reserved for future use. Itis similar to 401, but it indicates that the client first must authenticate itself with the proxy. HTTP/1.0 does not provide a means for proxy authentication.
408	Request Timeout	The client did not produce a request within the time that the server was prepared to wait.
409	Conflict	The request could not be completed due to a conflict with the current state of the resource.
410	Gone	The requested resource is no longer available at the server, and no forwarding address is known.
411	Authorization Refused	The request credentials provided by the client were rejected by the server or insufficient to grant authorization to access the resource.
500	Internal Server Error	The server encountered an unexpected condition that prevented it from fulfilling the request.
501	Not Implemented	The server does not support the functionality required to fulfill the request.
502	Bad Gateway	The server received an invalid response from the gateway or upstream server it accessed when

		attempting to fulfill the request.
503	Service Unavailable	The server currently is unable to handle the request due to a temporary overloading or maintenance of the server.
504	Gateway Timeout	The server did not receive a timely response from the gateway or upstream server it accessed when attempting to complete the request.

ضمیمه ۸

نوع‌های MIME

MIME Type	What It is (If Noted)	File Extensions (ncSA)	File Extensions (CERN)
application/acad	AutoCAD drawing files		dwg, DWG
application/clariscad	ClarisCAD files		ccAD
application/drafting	MATRA Prelude drafting		DRW
application/dxf	DXF (AutoCAD)		dxf, DXF
application/i-deas	SDRC I-DEAS files		unv, UNV
application/iges	IGES graphics format		igs, iges, IGs, IGES
application/ octet-stream	Uninterpreted binary	bin	bin
application/oda		oda	oda
application/pdf	PDF (Adobe Acrobat)	pdf	pdf
application/postscript	PostScript, encapsulated PostScript, Adobe Illustrator		ai, PS, ps, eps
application/pro_eng	PTC Pro/ENGINEER	part	prt, PRT
application/rtf	Rich Text Format	rtf	rtf
application/set	SET (French CAD standard)		set, SET
application/sla	Stereolithography		stl, STL
application/solids	MATRA Prelude Solids		SOL
application/STEP	ISO-10303 STEPdata files		stp, STP, step, STEP
application/vda	VDA-FS surface data		vda, VDA
application/x-mif	FrameMaker MIF format	mif	
application/x-csh	C-shell script	csh	csh
application/x-dvi	TeX dvi	dvi	dvi
application/x-hdf	ncSA HDF data file	hdf	hdf

application/x-latex	LaTeX source	latex	latex
application/x-netcdf	Unidata netCDF	nc, cdf	nc, cdf
application/x-sh	Bourne shell script	sh	sh
application/x-tcl	TCL script	tcl	tcl
application/x-tex	TeX source	tex	tex
application/x-texinfo	Texinfo (emacs)	texinfo, texi	texinfo, texi
application/x-troff	troff	t, tr, roff	t, tr, roff
application/x-troff-man	troff with MAN macros	man	man
application/x-troff-me	troff with ME macros	me	me
application/x-troff-ms	troff with MS macros	ms	ms
application/x-wais-source	WAIS source	src	src
application/zip	ZIP archive	zip	
application/x-bcpio	Old binary CPIO	bcpio	bcpio
application/x-cpio	POSIX CPIO	cpio	cpio
application/x-gtar	GNU tar	gtar	gtar
application/x-shar	Shell archive	shar	shar
application/x-sv4cpio	SVR4 CPIO	sv4cpio	sv4cpio
application/x-sv4crc	SVR4 CPIO with CRC	sv4crc	sv4crc
application/x-tar	4.3BSD tar format	tar	tar
application/x-ustar	POSIX tar format	ustar	ustar
audio/basic	Basic audio (usually _-law)	au, snd	au, snd
audio/x-aiff	AIFF audio	aif, aiff, aifc	aif, aiff, aifc
audio/x-wav	Windows WAVE audio	wav	wav
image/gif	GIF image	gif	gif
image/ief	Image Exchange format	ief	ief
image/jpeg	JPEG image	jpeg, jpg jpe	jpg, JPG, JPE, jpe, JPEG,

			jpeg
image/tiff	TIFF image	tiff, tif	tiff, tif
image/x-cmu-raster	CMU raster	ras	ras
image/x-portable-anymap	PBM Anymap format	pnm	pnm
image/x-portable-bitmap	PBM Bitmap format	pbm	pbm
image/x-portable-graymap	PBM Graymap format	pgm	pgm
image/x-portable-pixmap	PBM Pixmap format	ppm	ppm
image/x-rgb	RGB image	rgb	rgb
image/x-xbitmap	X bitmap	xbm	xbm
image/x-xpixmap	X pixmap	xpm	xpm
image/x-xwindowdump	X Windows dump (xwd) format	xwd	xwd
multipart/x-zip	PKZIP archive		zip
multipart/x-gzip	GNU zIP archive		gzip
text/html	HTML	html	html, htm
text/plain	Plain text	txt	txt, g, h, C, cc, hh, m, f90
text/richtext	MIME rich text	rtx	rtx
text/tab-separated-values	Text with tab-separated values	tsv	tsv
text/x-setext	Struct-enhanced text	etx	etx
video/mpeg	MPEG video	mpeg, mpg, mpe	MPG, mpg, MPE, mpe, MPEG, mpeg
video/quicktime	QuickTime video	qt, mov	qt, mov
video/x-msvideo	Microsoft Windows video	avi	avi
video/x-sgi-movie	SGI movieplayer format	movie	movie

ضمیمه ۹

متغیرهای محیطی

۱: متغیرهای محیطی CGI (CGI Environment Variables)

Variable	Meaning
AUTH_TYPE	Contains the authentication method used to validate the Web browser, if any is used. An example of an authentication method is a username/password scheme.
CONTENT_LENGTH	The length of the user-provided content from the Web page requesting the CGI script, which is sent via the user's Web browser. Because the user-provided content is passed to the CGI script as a string, this value is in bytes, with each byte representing one character.
CONTENT_TYPE	Contains the type of the data that accompanies the browser's request for the CGI script. Examples are text/html or image/jpeg.
GATEWAY_INTERFACE	Holds the version of the Common Gateway Interface being used. For version 1.1 of the CGI specification, this variable would be CGI/1.1.
PATH_INFO	Holds additional path information for the CGI script. This is usually the virtual path to another document in the document root that the CGI script will use. This value is set from the information appended to the URL requesting the CGI script. See PATH_TRANSLATED for an example.
PATH_TRANSLATED	Holds additional path information for the CGI script. This is usually the virtual path to another document in the document root that the CGI script will use. This value is set from the information appended to the URL requesting the CGI script. See PATH_TRANSLATED for an example.
QUERY_STRING	Contains the user-provided data when the request method is GET. This data is appended along with a question mark to the referenced URL. For example, in the URL http://www.robertm.com/cgi-bin/answer.pl?state=CA, the QUERY_STRING would be "state=CA."
REMOTE_ADDR	Stores the IP address of the machine running the Web browser requesting the CGI script.
REMOTE_HOST	Stores the domain name of the machine running the Web browser requesting the CGI script. If this information is unavailable to the Web server, REMOTE_ADDR will be set and REMOTE_HOST will not be set.
REMOTE_IDENT	Stores the user's login name only if the Web server supports identification.

REMOTE_USER	Stores the username the Web browser specified for authentication. This is only set if the server supports authentication and the CGI script is protected.
REQUEST_METHOD	Contains the request method used to request the CGI script. This can contain any of the valid HTTP request methods such as GET, HEAD, POST, PUT, and so on.
SCRIPT_NAME	Stores the virtual path and name of the CGI script being executed. This is used for self-referencing URLs.
SERVER_NAME	Contains the name, either domain name or IP address, of the machine running the Web server.
SERVER_PORT	Contains the port number on which the Web browser sent the request to the Web server.
SERVER_PROTOCOL	Contains the name and version of the protocol being used to make the request for the CGI script. In most cases, this will be the HTTP protocol and will look something like HTTP/1.0.
SERVER_SOFTWARE	Stores the name and version of the Web server software that executed the CGI script. For example, for the Netscape Communications Server version 1.1, the variable would be set to Netscape-Communications/1.1.

۲:متغیرهای محیطی سرآیند درخواست HTTP

(HTTP Request Header Environment Variables)

HTTP Request Header	**Meaning**
HTTP_ACCEPT	Contains a comma-separated list of media types the browser can accept in response from the Web server. Examples are audio/basic, image/gif, text/*, */*. The last two examples contain the wildcard *, which is a stand-in for any string of characters. text/* means that all forms of text can be accepted; */* means that the browser will accept any content type.
HTTP_ACCEPT_ENCODING	Contains the valid encoding methods the browser can receive in response from the Web server. Examples are x-zip, x-stuffit, and x-tar.
HTTP_ACCEPT_LANGUAGE	Contains the browser's preferred language for a response from the Web server. However, responses in any language not specified in this variable are allowed. An example is en_UK, which is the English of the United Kingdom.
HTTP_AUTHORIZATION	Contains authorization information from the Web browser. Its value is used for the browser to authenticate itself with the Web server. There is not a single specific format for possible values of this field, and new formats may be added. One example is the user/password scheme, where the value, in my case, would be user robertm:mypassword.
HTTP_CHARGE_TO	Formats for this field are still undetermined. However, it is

	available to contain information for the account that is to be charged for the costs of receiving the requested data.
HTTP_FROM	Contains the name of the requesting user as supplied by the Web browser in an e-mail address format. Some examples are robertm@deltanet.com and rmcdanie@primenet.com.
HTTP_IF_MODIFIED_SINCE	Can contain a value specified in a valid ARPANET date standard, such as Weekday, DD-Mon-YY HH:MM:SS TIMEZONE. This field can be used in conjunction with the GET method to return the requested document only if it has changed since the date specified.
HTTP_PRAGMA	Holds the value of any special directives for the Web server. For instance, a proxy Web server has one valid value for a pragma request header, no-cache, which means that the proxy server should always request the document from the real Web server instead of returning a nonexpired cached copy.
HTTP_REFERER	Contains the URI (uniform resource identifier, which is a superset of URLs) of the document that contained the link to the currently requested document. An example would be http://www.thepalace.com/web-pages.html.
HTTP_USER_AGENT	Contains the name of the Web browser software that requested the document. An example is Mozilla/2.0 (Win95; I), which would be the user agent for the Netscape 2.0 browser for Windows 95.

Server Directives for Parsed Headers: ۳

Directive	**Meaning**
Content-type	Specifies to the Web server the MIME type of the data being returned by the CGI script.
Location	Contains either the virtual path or the URL of a document that your CGI script wants returned to the Web browser requesting your script.
Status	Returns to the Web server an HTTP status line, which will then be returned to the Web browser. Status lines consist of a three-digit status code and the reason string. Examples are 404 Not Found and 403 Forbidden.

HTTP Response Headers:۴

HTTP Response Header	**Meaning**
ALLOWED	Specifies to the requesting browser which request methods are allowed. Examples are GET, HEAD and PUT.
CONTENT-ENCODING	Specifies which encoding method is used. Examples are x-zip, x-stuffit, and x-tar.
CONTENT-LANGUAGE	Specifies the language the returning document is in. An example is en, which is English in one of its forms.
CONTENT-LENGTH	Specifies the size in bytes of the returning data.

CONTENT-TRANSFER-ENCODING	Specifies the encoding of the data between the Web server and the Web browser. The default is binary.
CONTENT-TYPE	Contains the type of the data being transferred. Examples are text/html and image/gif.
COST	Will contain the cost of the retrieval of the object being requested. The format of this header has not yet been specified.
DATE	Contains a creation date of the requested object in a valid ARPANET format.
DERIVED-FROM	Can contain a version number for the requested object, allowing for version control of editable documents.
EXPIRES	Contains an expiration date for the requested information, after which the document should be retrieved again. This header is used primarily for caching mechanisms and is in an ARPANET date format.
LAST-MODIFIED	Contains the date when the requested object was last modified. This header is in an ARPANET date format.
LINK	Holds information about the document being returned. You can use it to specify information such as the inclusion of another URL within the returned document or the creator of the returned object.
MESSAGE-ID	Contains a unique identifier for the HTTP message.
PUBLIC	Fairly similar to the ALLOW response header. However, it specifies the request methods that anyone can use, not just the requesting browser. Examples for this header are GET, HEAD, and TEXTSEARCH.
TITLE	Contains the title of the document being returned. For an HTML file, this is equivalent to the value contained within the <TITLE></TITLE> tags.
URI	Gives the URI (uniform resource identifier) where the requested object can be found. This will not always be the URL the user entered in the Web browser requesting the returned object. However, it will point to an object that should be the same as the one being returned, with some degree of variance. An example is http://www.robertm.com/Group-one/section1.htmlvary= language, version which gives a URI with the same document, which might vary in language or version.
VERSION	Defines the version of an object that can be changed. Its format is currently undefined.