به نام خدا

آموزش ساخت اتوران

مرجع نرم افزار
AutoPlay Media Studio 8

نویسندگان :
جواد احشامیان
حامد حیدری

کاری از انجمن وسوسه

تقدیم به تمام کسانی که
قلبمان برای آن ها می تپد

مقدمه :

این کتاب حاصل تلاش چند ماهه نویسندگان آن می باشد و به امید ارتقا سطح علمی کاربران نرم افزار AutoPlay Media Studio گردآوری شده است و فروش آن تحت هر عنوانی غیرقانونی و غیر شرعی است و نویسندگان آن راضی نخواهند بود . در صورت داشتن نظر یا پیشنهادی برای اصلاح نسخه کتاب ، یا مشاهده اشکالات احتمالی در کتاب به وسیله سایت vasva3.com با ما ارتباط برقرار کنید.

بر اساس نظر سنجی بین کاربران سایت قرار بر ارائه کتاب به صورت بسته نرم افزاری و با هزینه بالغ بر 6 الی 7 هزار تومان بود که پس از بررسی ، تصمیم بر ارائه این کتاب به صورت رایگان شد تا عموم کاربران به این کتاب دسترسی داشته باشند و در صورت استفاده از کتاب و رضایت مبلغ دلخواه خود را به صورت همیاری به حساب ما واریز نمایند .

اکثر مبالغ همیاری شده برای حفظ انجمن وسوسه هزینه خواهد شد تا کاربران براحتی بتوانند مشکلات خود را مطرح و حل کنند.

همچنین نام 3 نفر از برترین همیاران (در صورت تمایل) در پست اصلی این کتاب در انجمن وسوسه قرارخواهد گرفت.

بانک ملی

شماره کارت:6037 9911 9480 1191

شماره حساب:0302177622009

شماره شبا: IR19 0170 0000 0030 2177 6220 09

به نام : جواد احشامیان

بانک ملت

شماره کارت:6104 3370 5543 8772

شماره حساب:3540239659

شماره شبا:IR50 0120 0100 0000 3540 2396 59

به نام :حامد حیدری


ملت کارت
Mellat Access Card
بانک ملت
bank mellat
6104 3370 5543 8772
حامد حیدری


بانک ملی ایران
ملی کارت
6037 9911 9480 1191
جواد احشامیان

پیشگفتار

ساخت برنامه‌های چند رسانه ای و اتوران نویسی یکی از مباحث روز دنیای کامپیوتر می‌باشد که افراد، شرکت‌ها و ارگان‌های زیادی در سرتاسر جهان از آن بهره می‌برند. حال این برنامه می‌تواند آموزشی، تجاری، مذهبی، تبلیغاتی و ... باشد. این نوع برنامه‌ها اکثراً برای اجرا از روی CD یا DVD ساخته می‌شوند اما می‌توان از آن بر روی حافظه‌ی دیسک سخت نیز استفاده نمود و یا آن را به صورت برنامه‌ی قابل نصب ارایه کرد.

اتوران از اجزای معرفی شده توسط مایکروسافت بر روی سیستم عامل‌های خود می‌باشد که تعیین کننده عملیاتی است برای زمانی که درایو توسط سیستم شناخته می‌شود.

در زمان قرارگیری لوح فشرده یا حافظه جانبی سیستم عامل به دنبال فایلی با نام autorun.inf در شاخه اصلی می‌گردد و در صورت یافتن این فایل دستورات فایل را اجرا خواهد کرد.

درون این فایل می‌توان دستورات متعددی از قبیل اجرای فایل خاص، تغییر آیکون درایو، تغییر نام درایو و ... را نوشت.

برای مثال یک فایل autorun.inf ساده:

```
[autorun]
open=autorun.exe
Label=VaSvA3
icon=autorun.ico
```

با استفاده از اتوران می‌توان تعامل بیشتری با کاربر ایجاد کرد و همچنین، دسترسی به فایل‌ها و کار با آن‌ها را برای کاربر آسان ساخت.

برای این کار ابزارهایی طراحی و ساخته شده که می‌توان با استفاده از آن‌ها، اتورانی زیبا، با قابلیت‌های فراوان ایجاد نمود که می‌توان از بین این نرم افزارها به AutoPlay Media Studio، Multimedia Builder، Adobe Director، Demo Shield و ... اشاره کرد.

یکی از برترین و بهترین برنامه‌های اتوران سازی که متأسفانه در کشور ما چندان شناخته شده

نیست نرم افزار Autoplay Media Studio می‌باشد که به جرأت می‌توان گفت بی نظیر است.

حال چه دلیلی دارد که ما Autoplay Media Studio را انتخاب کردیم؟ چون این نرم افزار محیط کاربری آسانی دارد، انعطاف پذیری آن بسیار بالا است، کدهای از پیش نوشته شده زیادی در خود دارد، دارای محیط کد نویسی جذاب و دل نشین است و در کد نویسی دست برنامه نویس را باز گذاشته است به طوری که می‌توان افزونه‌هایی را برای برنامه نوشت یا از افزونه های نوشته شده توسط دیگران استفاده کرد و به آن کدها و قابلیت‌های جدیدی اضافه کرد، پیش زمینه ای برای برنامه نویسی است و اتوران‌های چند رسانه ای بسیار زیبا و حرفه ای را می‌توان با آن نوشت.

هم چنین Autoplay Media Studio از یک زبان برنامه نویسی محبوب (Lua) برای کد نویسی استفاده می‌کند و طی کار با این برنامه می‌توانید یک زبان برنامه نویسی را هم بیاموزید.

با مطالعه‌ی این کتاب شما می‌توانید برنامه نویسی چند رسانه‌ای، اتوران نویسی و به همراه آن کد نویسی را یاد بگیرید و با تمرین و ممارست در این راه به یک برنامه نویس حرفه ای تبدیل شوید و از این دانش خود در زمینه های مختلف همچون علمی، فرهنگی، تجاری و ... استفاده نمایید.

برای مشاهده فهرست به صفحات پایانی این نوشتار مراجعه نمایید .

فصل اول: نصب Autoplay Media Studio و آشنایی با محیط برنامه

Autoplay Media Studio یکی بهترین و قوی‌ترین نرم افزارهای ساخت و تولید برنامه‌های چند رسانه ای، اتوران نویسی، برنامه‌های آموزشی و ... است که توسط شرکت Indigo Rose طراحی و تولید شده است.

به وسیله Autoplay Media Studio می‌توانید برنامه‌هایی نظیر یک دیوان اشعار چند رسانه‌ای، آزمون گیر و آزمون ساز، فرهنگ لغت، بسته های نرم افزاری و ... طراحی و تولید کنید.

هیچ نیازی نیست که شما دانش برنامه نویسی داشته باشید حتی اگر تا به حال یک سطر هم کد نویسی نکرده‌اید نگران نباشید چون کدهای از پیش نوشته شده در Autoplay Media Studio به حدی است که حتی ممکن است یک برنامه بنویسید بدون اینکه حتی یک سطر کد را تایپ نمایید .

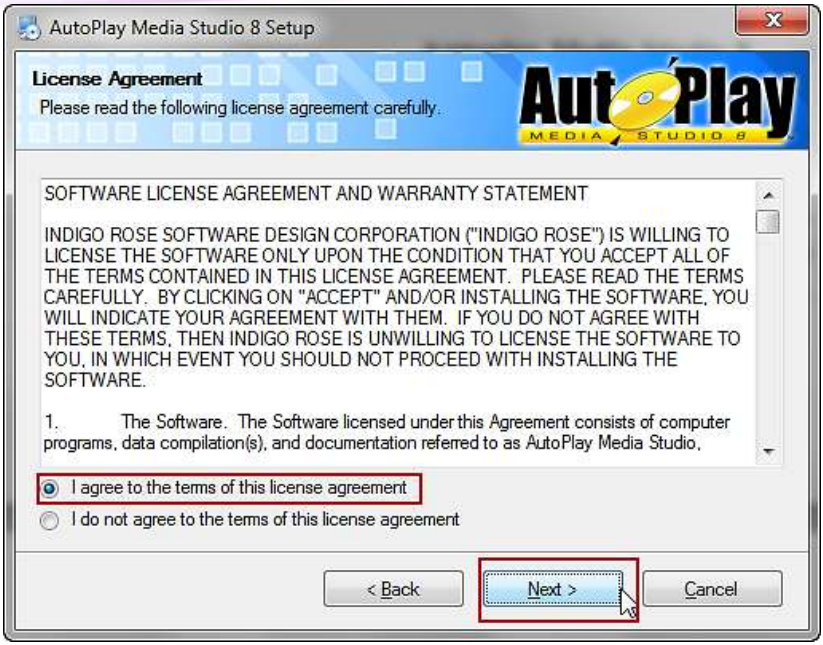
در طی مطالعه‌ی این کتاب شما به مثال‌هایی برخورد خواهید کرد که با تمرین و اجرای آن‌ها بر روی سیستم عامل بیشتر با جنبه های مختلف این برنامه آشنا خواهید شد.

نصب Autoplay Media Studio 8

برای استفاده از Autoplay Media Studio باید اول آن را بر روی سیستم عامل ویندوز خود نصب کنیم.

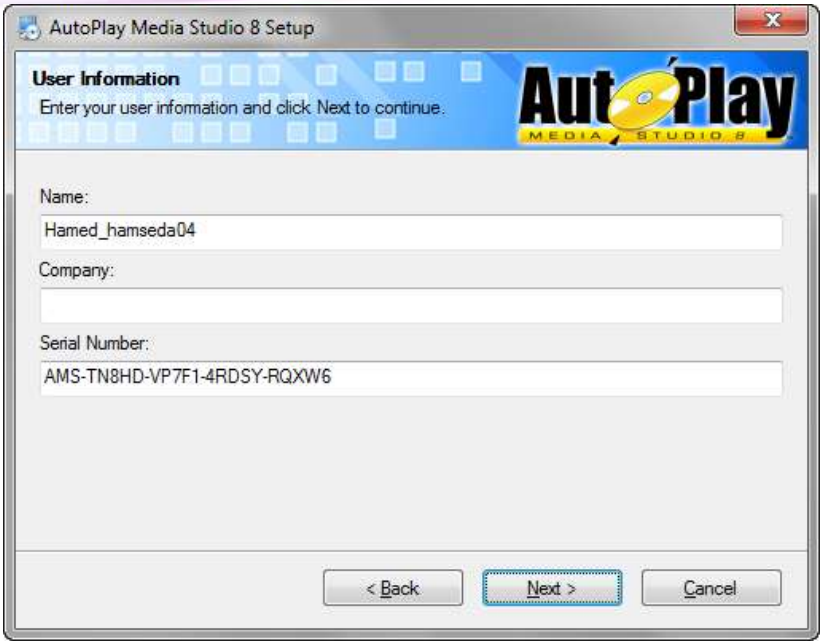
برای این کار ابتدا CD ارایه شده همراه کتاب را درون درایو قرار دهید و سپس فایل Setup.exe اجرا نمایید. در پنجره ظاهر شده روی دکمه‌ی Next کلیک کنید. (تصویر 1- 1) در مرحله‌ی بعد شما باید مطابق تصویر 1- 2 عمل کنید.

تصویر 1 - 1



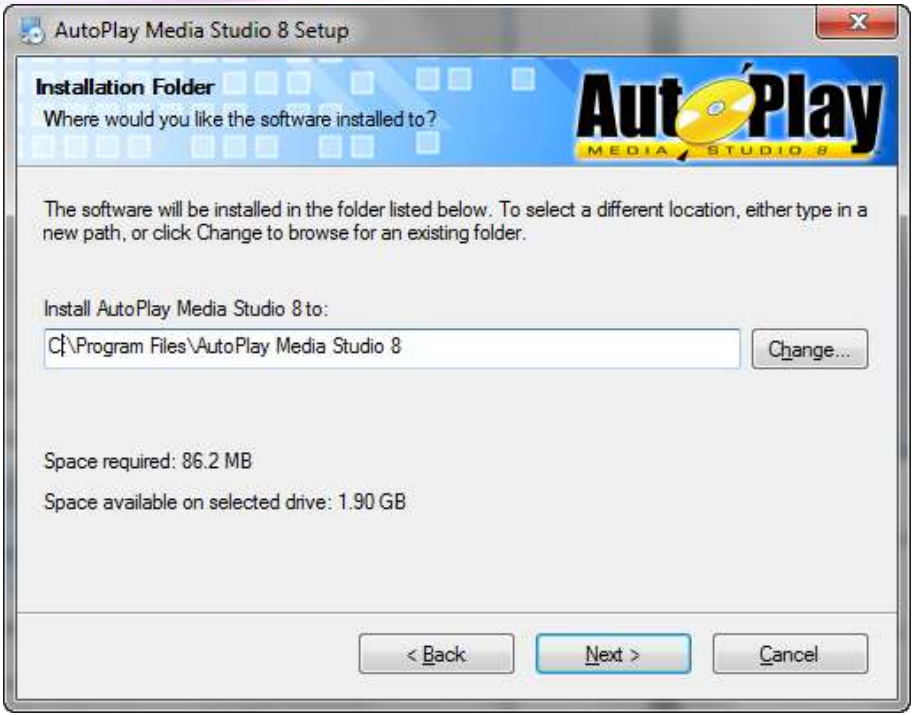
تصویر 1 - 2

در ادامه از شما شماره سریال درخواست می‌شود که آن را نیز در CD در فایل serial.txt قرار داده‌ایم که می‌بایست آن را در کادر مقابل Serial Number وارد کنید و بر روی دکمه‌ی Next کلیک نمایید .



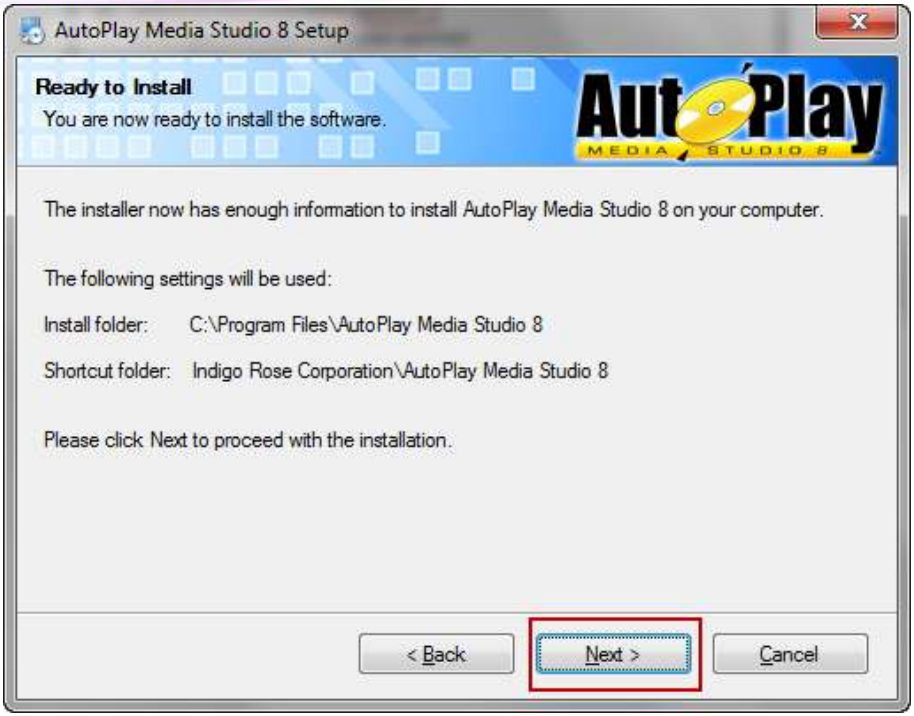
تصویر 1 - 3

در قسمت بعد مسیر نصب نرم افزار برای شما نمایان می‌شود که می‌توانید با تایپ مسیر مورد نظر یا کلیک بر روی Change و انتخاب شاخه میر پیش‌فرض را تغییر دهید.



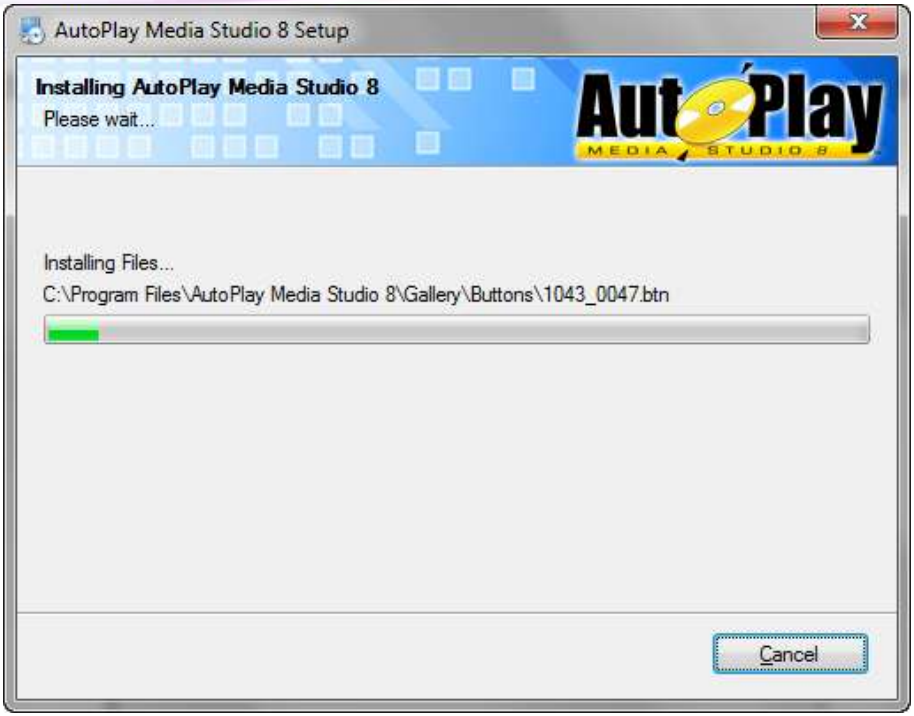
تصویر 1 - 4

با کلیک بر روی Next پنجره بازشده مسیر قرارگیری میانبر در Start Menu را مشاهده می‌کنید . بر روی Next کلیک کنید تا پنجره تایید اطلاعات نمایان شود .



تصویر 1 - 5

پس از کلیک بر روی Next برنامه با نمایش روند پیشرفت مراحل نصب بر روی سیستم شما نصب می‌شود .



تصویر 1 - 6

در آخرین پنجره ظاهر شده روی دکمه‌ی Finish کلیک کنید، بدین ترتیب شما Autoplay Media Studio 8 را روی سیستم عامل خود نصب کرده‌اید.

نکته: چنان چه پس از نصب و اجرای برنامه با فارسی نویسی در برنامه مشکل داشتید به ضمایم کتاب مراجعه نمایید.

کارکرد Auto Play :

نرم افزار Auto Play یک نرم افزار عالی برای تمام اقشار و گروه های سنی و با هر سطح دانشی تهیه شده است. چرا که با استفاده از حدود 850 اکشن آماده و کنترل کامل نرم افزار تا 8000 صفحه را مدیریت کرده و از برنامه‌های کوچک تا پروژه های سنگین را با این نرم افزار تولید و آماده بهره برداری نمایید. در این نرم افزار با مدیریت هر صفحه به طور جداگانه، کار برای کاربر آسان‌تر می‌شود.

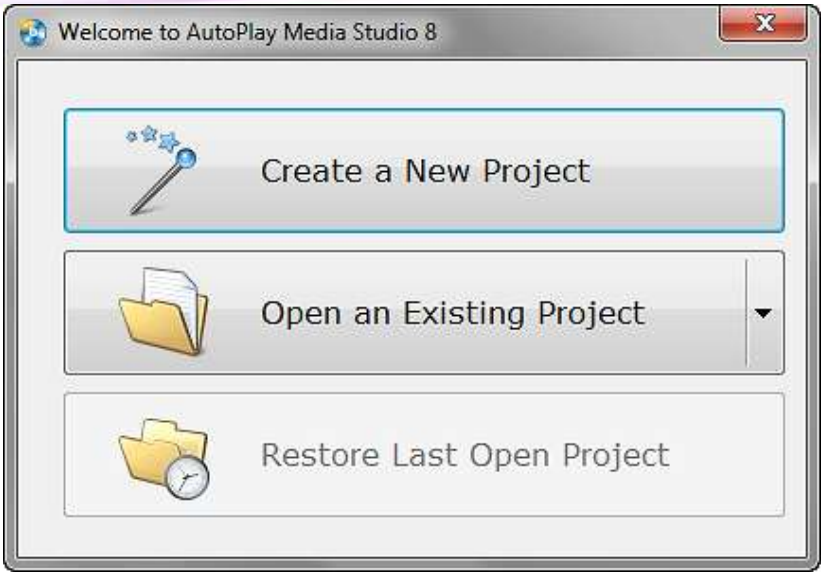
شما می‌توانید در هر صفحه از اشیاء گوناگونی مانند عکس، متن، موزیک، دکمه و ... استفاده کنید و برای هر یک دستورات جداگانه ای تعریف کنید .

همین طور در برنامه اجرایی فقط 1 صفحه قابل رویت خواهد بود و چگونگی و زمان اجرا شدن صفحات دیگر به دست شما خواهد بود.

در حالت کلی در یک پروژه ساخته شده توسط AMS، مجموعه ای از صفحات مختلف را در اختیار داریم که حرکت بین آن‌ها به سادگی قابل انجام است و در این صفحات می‌توانیم از اشیاء مورد نظرمان استفاده کنیم. هر یک از این اشیا نیز بسته به نیازمان می‌توانند کارهای متفاوت (مانند: نصب یک نرم افزار، نمایش یک پیغام ، باز کردن یک سایت، پخش آهنگ، مدیریت فایل یا پوشه، تغییر صفحه و .... صدها عملیات دیگر) را انجام دهند. پس در نتیجه تمام قابلیت‌های مورد نیاز برای ارتباط بین کاربر و رایانه در این نرم افزار قرار گرفته است.

ساخت پروژه جدید:

با اجرای نرم افزار، صفحه‌ی خوش آمد گویی نرم افزار ظاهر می‌گردد که در آن 4 گزینه دیده می‌شود.



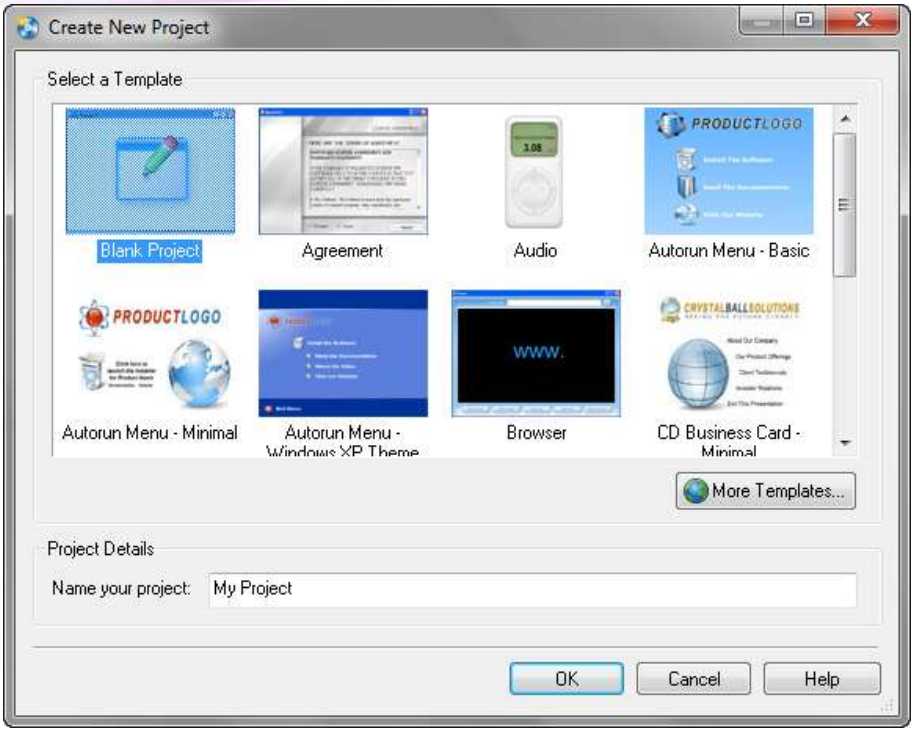
تصویر 1 - 7

- create a new project: این گزینه پروژه‌ی جدیدی را ایجاد خواهد کرد. برای آغاز کار بر روی این گزینه کلیک کنید.

- open An existing project: این گزینه، وظیفه‌ی باز کردن پروژه های ذخیره شده بر روی هارددیسک کامپیوترتان را دارد. البته این کار در هنگام اجرای پروژه‌ی دیگر، از مسیر file > open نیز امکان پذیر است.

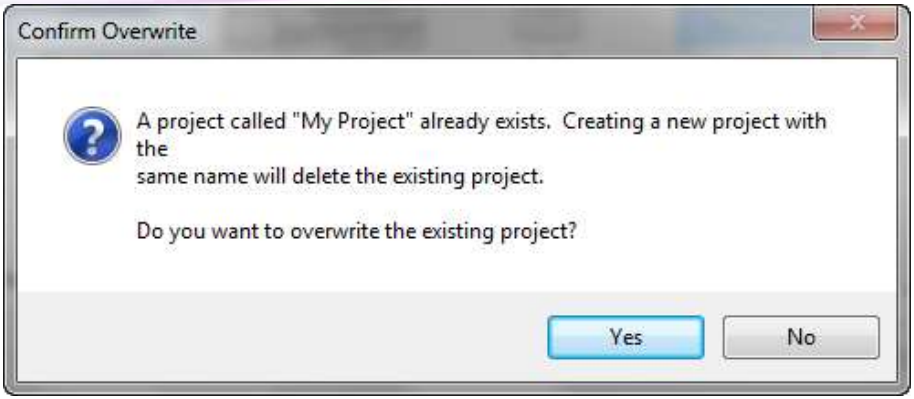
- Restore last open project: این گزینه، آخرین پروژه ای را که اجرا نموده‌اید را باز می‌کند.(در صورتی که برای اولین بار نرم افزار را باز کنید این گزینه غیر فعال خواهد بود)

با کلیک بر روی create a new project پنجره جدیدی برای ساخت پروژه ظاهر خواهد شد که شامل قسمتی برای انتخاب نام و قسمتی برای انتخاب نوع پروژه می‌باشد.



تصویر 1 - 8

پس از انتخاب نام پروژه و کلیک بر روی OK در صورتی که پروژه ای با این نـام وجـود داشـته باشد، پیغامی مبنی بر وجود پروژه دریافت می‌کنید، در صورتی که بر روی Yes کلیـک کنیـد، پروژه جدید بر روی قبلی ساخته می‌شود و در صورت انتخاب No پنجره برای تغییر نام پـروژه باز می‌ماند

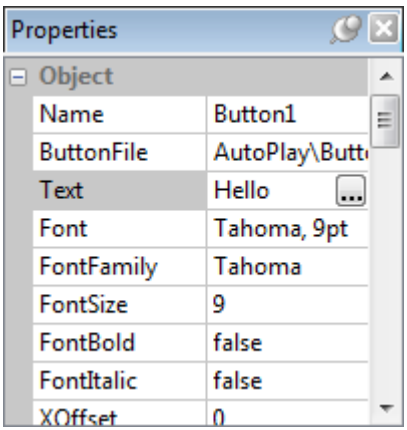


تصویر 1 - 9

شروع کار با AMS

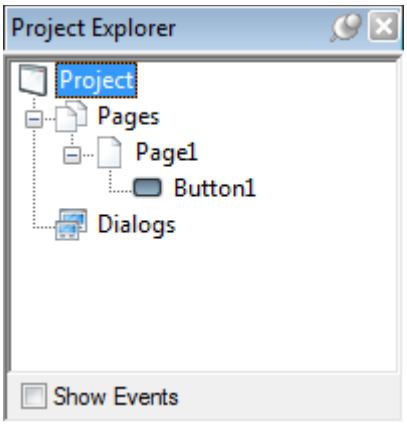
با ایجاد پروژه جدید به صورت پیش‌فرض پروژه دارای یک صفحه(Page) با نام Page1 است. حال قبل از اینکه تغییری در این پروژه ایجاد کنیم برخی از پنجره‌ها و تب‌های(سر برگ) موجود در محیط پروژه را که بسیار کاربردی هستند معرفی می‌کنیم و سپس به سراغ پروژه بر می‌گردیم.

1. پنجره Project Explorer: در این پنجره شما می‌توانید صفحه‌ها و اشیای موجود در آن‌ها را مشاهده نمایید. هم چنین می‌توانید با علامت‌گذاری قسمت Show Events رویدادهای هر یک از اشیا را مشاهده نمایید و با دو بار کلیک روی آن‌ها کد محتوای آن‌ها را نیز ببینید.



تصویر 1 - 10

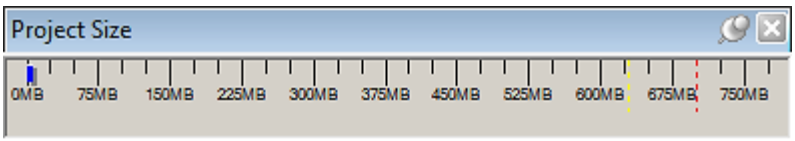
2. پنجره‌ی Properties: در این پنجره می‌توانید خصوصیات صفحه یا شی انتخاب شده را مشاهده کرده و آن‌ها را تغییر دهید.



تصویر 1 - 11

3. پنجره Project Size: در این پنجره می‌توانید مقدار حجم پروژه خود را مشاهده کنید. به صورت پیش‌فرض این پنجره از 0 تا 750 مگابایت را نمایش می‌دهد اما اگر خواستید مقدار نمایش آن را تغییر دهید از منوی زیر مقدار آن را تغییر دهید:

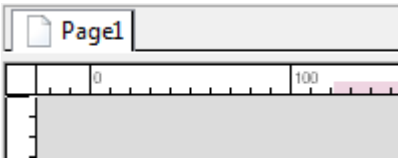
Edit>Preferences>Environment>Project Size



تصویر 1 - 12

نکته: هر پنجره را می‌توان از منوی View>Toolbar مخفی یا آشکار کرد.

4. Page Tab: در این تب یا سربرگ شما می‌توانید صفحه‌ها و یا دیالوگ‌هایی که ایجاد کرده‌اید را مشاهده کنید.(تصویر 1- 10)



تصویر 1 - 13

اضافه کردن صفحه جدید

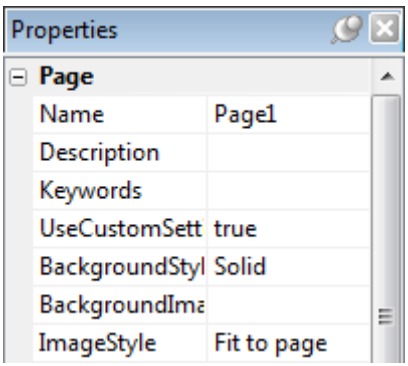
برای اضافه کردن صفحه جدید از منوی page گزینه add را انتخاب کنید تا یک صفحه خالی به پروژه شما اضافه گردد. شما می‌توانید در بین صفحات موجود رفت و آمد کنید. به این صورت که در نمای صفحات یک بار بر روی صفحه مورد نظر کلیک کنید.

- اضافه کردن دیالوگ

این قسمت هم مانند اضافه کردن صفحه می‌باشد.با انتخاب add از منوی dialog می‌توانید یک دیالوگ به پروژه اضافه نمایید.

- تغییر نام صفحه و دیالوگ

برای اینکه صفحات موجود را اشتباه نگیرید، پس از انتخاب یک صفحه(یا دیالوگ) از قسمت خصوصیات بر روی name کلیک و نام مورد نظر را جایگزین صفحه مورد نظر کنید. زمانی که نام را تغییر می‌دهید در قسمت نمای صفحات و دیالوگ‌ها نیز این نام تغییر می‌کند.



تصویر 1 - 14

- تغییر پس زمینه:

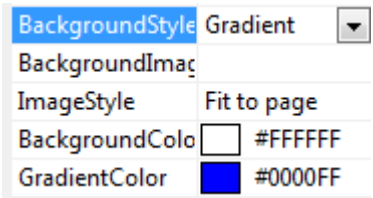
برای تغییر دادن رنگ زمینه‌ی پروژه باید در قسمت خصوصیات این کار را انجام دهید.

برای دسترسی به خصوصیات صفحه 2 روش وجود دارد. روش اول، بر روی صفحه کلیک راست کرده و properties را انتخاب کنید. پنجره‌ی خصوصیات صفحه باز می‌شود. که در این پنجره، نوع رنگ پس زمینه یا تصویر آن و ... مشاهده می‌شود که شما قادر به تغییر دادن آن نخواهید بود.

روش دوم، در سمت چپ و در قسمت پایین صفحه پنل خصوصیات(Properties Panes) را مشاهده می‌کنید که با 1 بار کلیک بر روی صفحه خصوصیات مربوط به صفحه، در این قسمت دیده می‌شوند.

زمانی که یک پروژه‌ی جدید با قالب Blank را انتخاب می‌کنید، به صورت پیش فرض رنگ پس زمینه، سفید خواهد بود. حال برای تغییر رنگ می‌توانید از لیست رنگ‌های موجود در پنل خصوصیات یک رنگ را انتخاب کنید. (Background color)

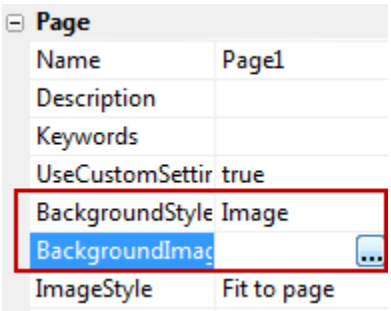
از دیگر خصوصیات برنامه‌ی AMS، 2 رنگ کردن پس زمینه می‌باشد. برای این کار در پنل خصوصیات، از لیست کرکره ای Background style گزینه‌ی solid را به Gradient تغییر دهید. حال برای انتخاب رنگ دوم، رنگ مربوط به Gradient color در پنل خصوصیات را به رنگ دلخواه خود تغییر دهید. مشاهده می‌کنید که رنگ صفحه به سایه روشن به 2 رنگ مجزا تبدیل شده است.



تصویر 1 - 15

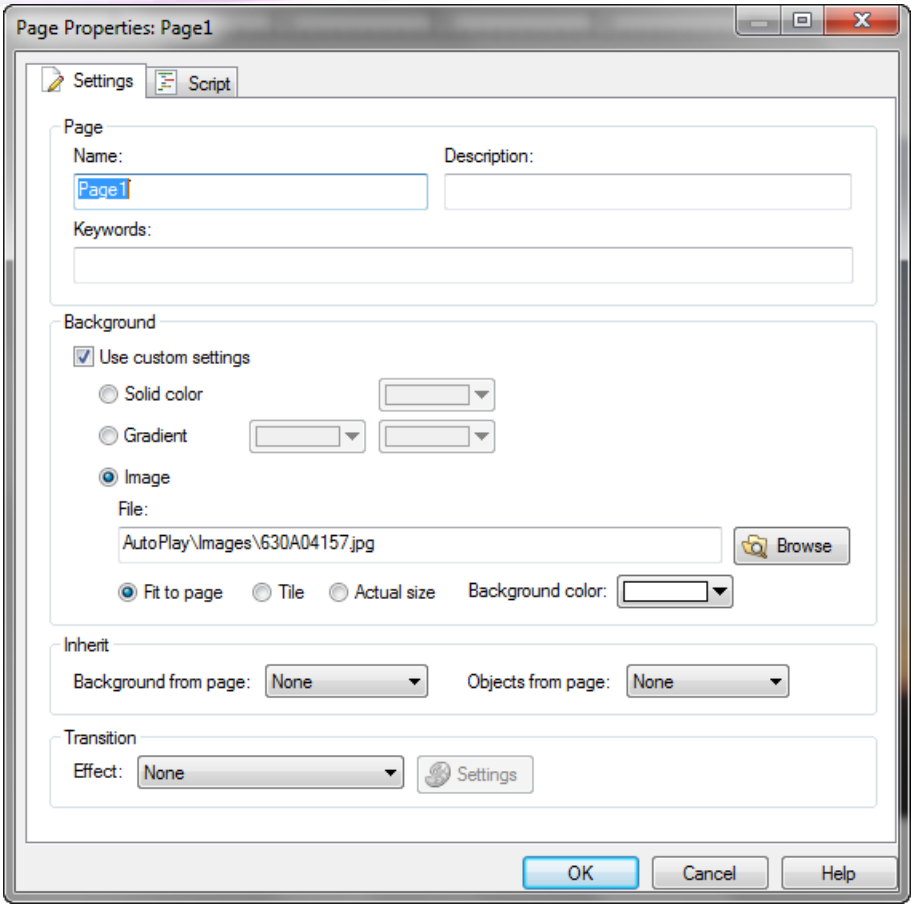
این بار از لیست Background Style, Image را انتخاب کنید. با کلیک بر روی Background image آیکون ... نمایان می‌شود( ) که با کلیک بر روی آن پنجره‌ی انتخاب فایل باز می‌شود. یک تصویر را انتخاب و بر روی ok کلیک کنید. با کلیک بر روی ok تصویر به صورت خودکار در شاخه‌ی اصلی پروژه شما ذخیره می‌شود و نیازی به انتقال آن فایل به طور

مجزا (که در بیشتر نرم افزارهای ساخت Autorun باید این کار را انجام دهید) نیازی نمی‌باشد.



تصویر 1 - 16

حال پس از تشریح کامل روش دوم توضیح مختصری از روش اول خواهم داد. پس از باز کردن پنجره‌ی خصوصیات، برای تغییر پس زمینه ابتدا گزینه Use Custom Setting را فعال کنید .3 قسمت image, Gradient, solid را مشاهده می‌کنید که دقیقاً مانند روش توضیح داده شده در قبل می‌باشد. با کلیک بر روی ok این پنجره را ببندید.

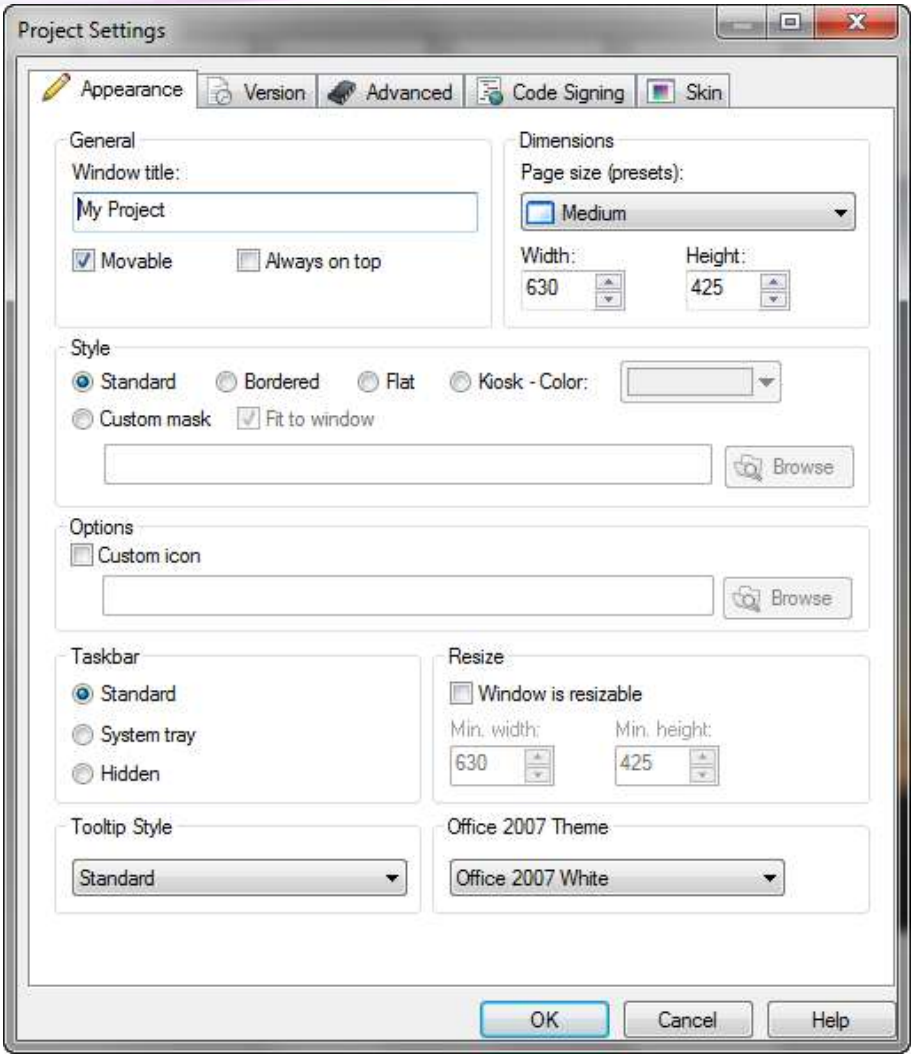


تصویر 1 - 17

تنظیمات پروژه (project setting)

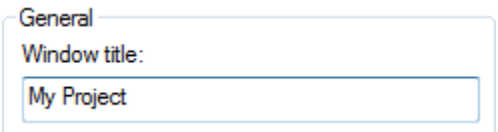
برای باز کردن صفحه‌ی تنظیمات پروژه‌ی خود از مسیر زیر پنجره‌ی مربوطه را باز کنید.

Project >setting



تصویر 1 - 18

تغییر نام پنجره (windows title)

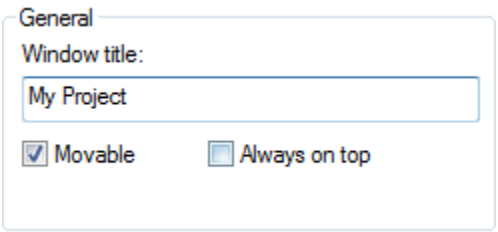


تصویر 1 - 19

نام پنجره‌ی اصلی برنامه به طور خودکار با نام پروژه شما تنظیم می‌شود برای تغییر نام آن، نام جدید را در فیلد windows title جایگزین نمایید.

در زمان اجرای پروژه، کاربر این نام را در نوار وظیفه‌ی ویندوز مشاهده می‌کنید .

در قسمت پایین فیلد windows title، 2 قسمت برای انتخاب مشاهده می‌کنید که به شرح هر یک از آن‌ها می‌پردازیم :



تصویر 1 - 20

Always on top: این گزینه باعث می‌شود پروژه (نرم افزار) همیشه روی همه‌ی پنجره‌ها قرار گیرد .

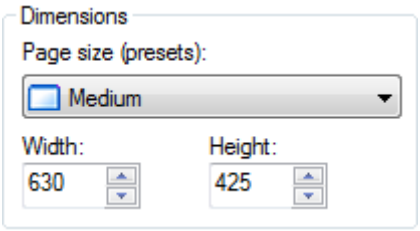
Movable: انتخاب این گزینه قابلیت جا به جایی نرم افزار را به پروژه‌ی شما اضافه خواهد کرد.

حال پس از تنظیم قسمت‌های یاد شده، به تنظیمات اندازه‌ی صفحات پروژه خواهیم پرداخت.

- ✓ اندازه‌ی صفحه (page size)

در قسمت **page size** تنظیمات اندازه‌ی صفحات انجام می‌شود که تمامی صفحات پروژه به این اندازه در خواهند آمد که برای تنظیم آن از 2 روش زیر می‌توانید استفاده کنید.

در روش اول از لیست کرکره ای یک مورد خاص مانند large را انتخاب کنید. و در روش دوم می‌توانید اعداد مورد نظرتان (مثلاً 1024 و 768) را در width و Height وارد نمایید.



تصویر 1 - 21

: style

در این قسمت تنظیمات مربوط به نمایش (نمـایش / عـدم نمـایش taskbar, titlebar و ...) انجام می‌پذیرد. با کلیک بر روی قسـمت custom mask دکمـه‌ی Browse زیـر آن فعـال خواهد شد. با کلیک بر روی آن صفحه‌ی انتخاب فایل را باز کنید. حال با استفاده از فایـل‌هـای گالری برنامه، فایل‌های ذخیره شده بر روی هارد (کامپیوتر شما) و یا فایل‌هـای پـروژه بـه کـار خود ادامه داده و شکل صفحه‌ی خود را به صورت تصویر انتخاب شده درآورید. بـرای ایـن کـار یک از تصاویر گالری را انتخاب کرده و بر روی ok کلیک کنید.البته شما می‌توانید این تصـاویر را خودتان با نرم افزاری مانند Adobe photoshop تولید و در پروژه‌هایتان اسـتفاده نماییـد. فقط دقت داشته باشید که فرمت این فایل‌ها png می‌باشد و همین طور این عکس‌ها (فایل‌هـا به صورت سیاه و سفید تولید شوند تا توسط نرم افزار بـه خـوبی تشـخیص داده شـوند.با تاییـد کردن، ظاهر و شکل صفحه به شکل فایل انتخابی تبدیل خواهد شد.

همین طور kiosk –color برای عدم نمایش taskbar (به صورت تمام صفحه) می‌باشد.

انتخاب یک آیکون براي پروژه:

در همـان پنجـره‌ی تنظیمـات، در قسـمت Custom icon, Option را انتخـاب کنیـد تـا Browse فعال شود. حال بر روی Browse کلیکی کنیـد. از قسـمت Gallery یـک آیکـون برای پروژه انتخاب کرده و بر روی ok کلیک کنید.

این آیکون بر روی taskbar و در زمان ضـبط اطلاعـات بـر روی CD نمـایش داده مـی‌شـوند (منظور این است که آیکون انتخاب شده جایگزین آیکون درایو DVD یا CD شما می‌شود.)

: taskbar

این قسمت محل تنظیم نحوه‌ی نمایش میله وضعیت (Stutsbar) می‌باشد کـه بـه شـرح هـر یک می‌پردازیم:

Standard: مانند بقیه نرم افزارها و My Computer نمایش داده می‌شـود. نـام پنجـره نیـز مشخص است.

تصویر 1 - 22

System Tray: آیکون برنامه مانند آیکون صدای ویندوز در کنار ساعت نمایش داده می‌شود. در این قسمت با نگه داشتن نشانه گر ماوس، نام پنجره مشخص می‌شود.

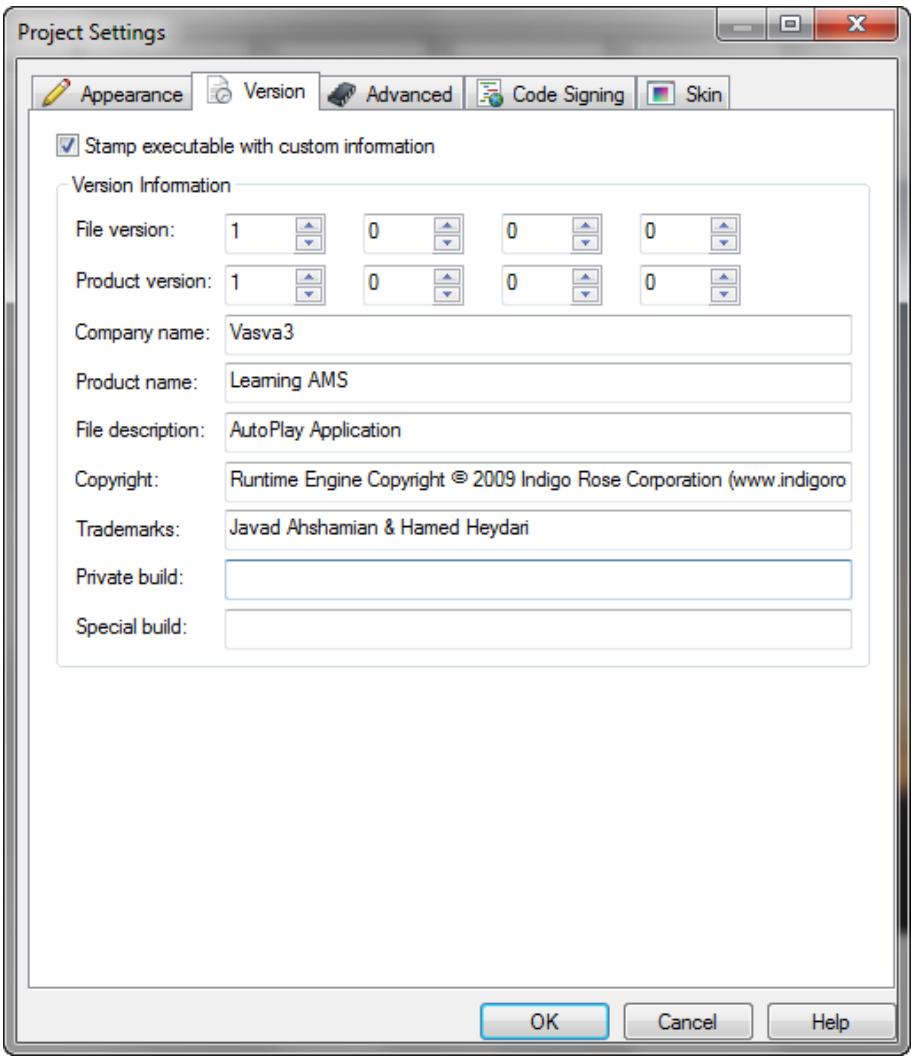
تصویر 1 - 23

Hidden: به هیچ صورت در میله‌ی وضعیت نمایش داده نمی‌شود.

ToolTip Style: با باز کردن منوی کشویی زیر این گزینه می‌توانید نوع نمایش ToolTip را انتخاب نمایید.

Resize :با فعال کردن این گزینه قابلیت تغییر اندازه پروژه توسط کاربر فعال می‌شود. دو قسمت MinWidthو Min height برای حداقل اندازه پروژه می‌باشد.

Version:



تصویر 1 - 24

در سربرگ version از همین پنجره اطلاعات مربوط به سازنده‌ی نرم افزار، نسخه‌ی آن و ... تنظیم می‌شود، انتخاب این سربرگ گزینه‌ی stamp executable with custom information دیده می‌شود این گزینه را علامت‌گذاری کنید. (تیک مربوط به آن را بزنید ☑)

با انتخاب تیک این گزینه اطلاعاتی برای تکمیل فعال می‌شود شما اطلاعات خود و نرم افزارتان را جایگزین این اطلاعات خواهید کرد. پس از تکمیل بر روی ok کلیک کنید.

در این قسمت نحوه کار با اشیا و چند شی اصلی را به طور مختصر توضیح خواهیم داد .

انتخاب چند شي

زمانی که شما یک بار بر روی شی کلیک می‌کنید اطراف شی به رنگ آبی در خواهد آمد. یکی از اشیاء موجود در صفحه را انتخاب و به قسمت خصوصیات آن توجه فرمایید. مشاهده خواهید کرد که تمامی اطلاعات آن از قبیل نام – طول- عرض– فاصله از بالا و غیره نمایش داده می‌شود. زمانی که شما چند شی را انتخاب می‌کنید این اطلاعات نمایش داده نمی‌شود. برای انتخاب شی دوم همراه شی انتخاب شده دکمه کنترل (ctrl) را از روی کیبورد پایین نگه داشته و بر روی شی دوم کلیک کنید مشاهده می‌کنید شی دوم نیز همراه با شی اول به انتخاب در می‌آید. علاوه بر آن کادر آبی دور شی اول را به صورت نقطه چین می‌بینید. معنی آن این است که شی اصلی انتخاب شده آن شی است که کادر دور آن به رنگ پر رنگ است اما در اصل هر دو شی انتخاب شده است. حال باز هم دکمه کنترل را پایین نگه داشته و بر روی شی دیگری که از نوع اشیاء دیگر نباشد کلیک کنید. ملاحظه می‌کنید که نوع شی هم در انتخاب آن مشکل ایجاد نخواهد کرد. حال باز هم دکمه کنترل را پایین نگه داشته و بر روی شی دوم انتخابی خود کلیک کنید با این کار این شی از حالت انتخاب در خواهد آمد.

بر روی صفحه پروژه خود کلیک کنید مشاهده می‌کنید که تمامی اشیاء از حالت انتخاب در خواهد آمد. حال نشانه گر ماوس را نزدیک اشیاء موجود قرار دهید. کلیک سمت چپ ماوس را نگه داشته و به سمت اشیاء مورد نظر حرکت دهید، کادر آبی رنگی ملاحظه می‌کنید. کلید سمت چپ را رها کنید. تمامی اشیاء که در مستطیل آبی رنگ قرار داشتند انتخاب شدند- به حرکت در آوردن چند شی زمانی که چند شی را همراه هم انتخاب کرده‌اید نشانه گر ماوس را بر روی یکی از آن‌ها قرار داده و کلید سمت چپ ماوس را نگه داشته و اشیاء را به اطراف حرکت دهید.

گروه بندي اشيا

بعضی اوقات برای ساخت یک Logo یا یک تصویر ترکیبی و یا حتی یک متن با نوع قرارگیری خاص، مجبورید از 2 یا چند شی استفاده نمایید ولی برای جابجایی آن‌ها دچار مشکل

می‌شوید. برای از بین بردن این مشکل باید آن‌ها را به یک گروه تبدیل کنید تا با حرکت دادن یکی از آن‌ها تمامی آن‌ها حرکت داده شود. برای این کار شما باید اشیاء مورد نظرتان را انتخاب کنید و از منوی edit بر روی Group کلیک نمایید، حال در محلی خالی از صفحه کلیک کنید تا اشیاء از حالت انتخاب درآید. دوباره بر روی یکی از اشیاء قبلی کلیک و آن را جابجا کنید. مشاهده می‌کنید که همه‌ی آن‌ها با هم جابجا خواهند شد.

برای از گروه درآوردن آن‌ها از منوی edit، Ungroup را انتخاب کنید.

قفل کردن یک شی

زمانی که در یک پروژه از تعداد زیادی شی استفاده می‌کنید، امکان ناکارا بودن بعضی از اشیاء وجود دارد در نتیجه باید آن را قفل کنید تا دیگر جابجا نشود. برای این کار یک شی را انتخاب و از منوی edit بر روی Lock کلیک کنید. حال شی قفل شده را انتخاب کنید.

مشاهده می‌کنید که دیگر شی قفل شده انتخاب نمی‌شود. حال بر روی شی کلیک راست کنید تمامی گزینه‌ها غیر از lock خاموش می‌باشد. شما نمی‌توانید با آن کار کنید. برای از بین بردن این حالت گزینه Lock را انتخاب کنید تا از حالت قفل شدن خارج شود. در صورتی که تعداد اشیاء قفل شده زیاد باشد و نیاز به از قفل درآوردن همه آن‌ها باشد از منوی unlock All کلیک کنید.

سنجاق کردن یک شی

منظور از سنجاق کردن یک شی این است که شما می‌توانید آن را انتخاب کنید ولی قادر به جابجا کردن آن نخواهید بود. برای این کار شی را انتخاب و از مسیر Edit > Pin آن را بر روی صفحه سنجاق کنید. هم اکنون برای از بین بردن حالت سنجاق از منوی edit، Unpin را انتخاب کنید.

مخفي کردن یک شی

در این حالت شی انتخاب شده از صفحه نمایش برنامه حذف می‌شود ولی در نمایش پروژه دیده خواهد شد. برای این کار بر روی شی کلیک راست کرده و بر روی Hide کلیک کنید. شی مورد نظر پنهان خواهد شد، برای ظاهر سازی اشیاء پنهان شده از منوی edit بر روی Unhide All کلیک نمایید.

بازگشت عملیات (undo)

زمانی که شما جابجایی و یا تغییراتی را بر روی اشیاء خود اعمال کرده باشید و متوجه اشتباه خود شوید لازم نیست دوباره تغییرات قبلی را بر روی شی اعمال کنید بلکه فقط با انتخاب گزینه undo از روی منو edit به تغییرات یک مرحله خود باز خواهید گشت. اگر باز هم این گزینه را انتخاب کنید تغییرات مرحله به مرحله به عقب باز خواهد گشت.

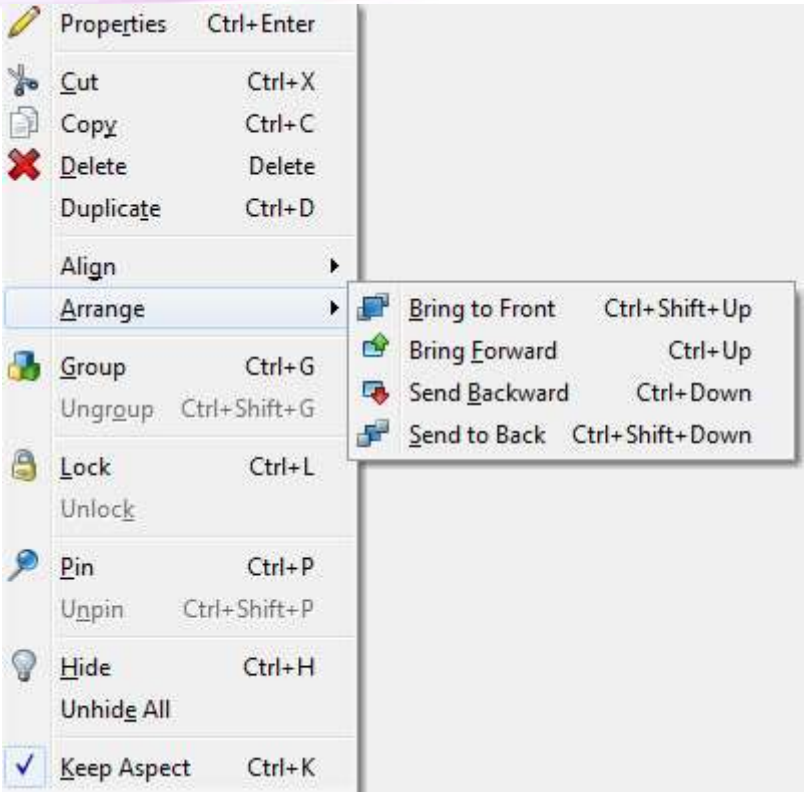
Redo در صورتی که از دکمه undo استفاده کرده باشید برای بازگشت به حالت قبلی از منوی edit گزینه Redo را انتخاب کنید.

مرتب سازي اشياء

چند شی را انتخاب کنید، بر روی یکی از آن‌ها کلیک راست کنید. در منوی باز شده گزینه به نام (align) مشاهده می‌کنید. گزینه های داخل align نحوه قرارگیری اشیاء انتخاب شده را تعیین می‌کند اگر گزینه to page از منو align انتخاب نشده باشد تمامی تنظیمات بر اساس شی که بر روی آن کلیک راست کرده‌اید انجام می‌شود ولی در غیر این صورت تمامی مرتب سازی بر اساس نحوه قرارگیری در صفحه تنظیم می‌شود. این گزینه (align) کاربرد فراوانی دارد. پس برای راحتی کار با آن از منو (view>toolbars) گزینه (align) را انتخاب کنید تا به صفحه شما اضافه شود.

نوع چیدن اشیاء

احتمالاً تا این جای کار امکان داشته که 2 شی به پروژه اضافه کنید و یک شی زیر یک شی دیگر قرار گیرد، در صورتی که شما نیاز به رو بودن شی پایینی داشته باشید. برای مثال یک تصویر و یک برچسب اضافه نموده‌اید و جای اینکه برچسب بر روی تصویر قرار گیرد تصویر بر روی بر چسب قرار می‌گیرد و برچسب دیده نمی‌شود. پس برای این کار بر روی شی مورد نظر کلیک راست کرده و در آن منو نشانگر ماوس را بر روی گزینه Arrange قرار داده تا لیست آن باز شود.



تصویر 1 - 25

حال 4 گزینه را ملاحظه می‌کنید که عبارتند از:

Bring to Front: با انتخاب این گزینه شی مورد نظر بر روی تمامی اشیاء قرار می‌گیرد.

Bring Forward: در صورت انتخاب این گزینه شی مورد نظر به اندازه یک شی بالا می‌آید.

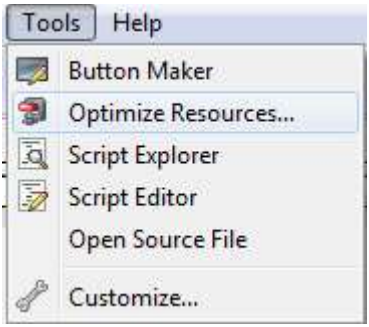
Send to Back: این گزینه شی را زیر تمامی گزینه‌ها قرار می‌دهد.

Send Backward: این گزینه، شی مورد نظر به اندازه یک شی پایین می‌رود.

حذف فایل‌های اضافه

همان طور که قبلاً ذکر شده بود تصاویری که شما وارد پروژه خود می‌کنید در یک شاخه نگهداری می‌شود که شما آن شاخه را در انتها بر روی یک CD رایت نمایید.

حال اگر شما تعداد زیادی تصویر را وارد پروژه کنید و از هیچ یک استفاده نکنید فضای زیادی از پروژه شما گرفته می‌شود. پس برای پاک کردن تمامی تصاویر و فایل‌ها و ... اضافه موجود در پروژه از منوی Tools>Optimize Resurce کلیک نمایید.



تصویر 1 - 26

حال برای خالی کردن شاخه‌ی اشیاء مورد نظر آن‌ها را انتخاب و بر روی ok کلیک نمایید. لیست فایل‌های موجود هر شاخه نمایش داده می‌شوند که با تیک زدن هر مورد و کلیک بر روی ok آن‌ها پاک خواهند شد.

تعویض نام اشیاء

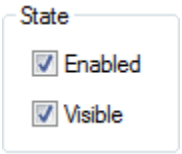
تنظیم نام اشیاء یکی از مهم‌ترین قسمت‌های ساخت پروژه می‌باشد چرا که وقتی شما یک پروژه را طراحی می‌کنید، برای استفاده از اشیاء باید نام آن‌ها را بدانید.

در غیر این صورت دچار مشکل شده و امکان درست کار نکردن پروژه شما نیز وجود دارد. برای مثال در صفحه اصلی پروژه شما 5 برچسب وجود دارد.

در قسمتی از برنامه دستوراتی را نوشته‌اید که از یک برچسب محتوای آن را برداشته و با اضافه کردن متنی به آن به برچسب دیگر منتقل کنید برای این کار اگر اسم برچسب‌ها را ندانید امکان اشتباه زدن نام آن‌ها وجود دارد. پس در نتیجه به جای عوض شدن مثلاً قسمت mail محتویات برچسب آدرس تغییر خواهد کرد. برای این کار باید نام هر برچسب را متناسب با کار برچسب تعیین کنید. پس بر روی برچسب خود 2 بار click کنید. سربرگ attributes را انتخاب و فیلد ورود اطلاعات Object Name را به نام مورد نظر تغییر دهد. این نام در هنگام اجرا پروژه نمایش داده نمی‌شود.

وضعیت‌ها

پس از رجوع به تنظیمات و سربرگ attributes یک عنوان به نام state ملاحظه می‌کنید. در این قسمت 2 گزینه برای فعال و غیر فعال کردن وجود دارد که عبارتند از:



تصویر 1 - 27

1. Enable: در صورت غیر فعال بودن این گزینه شی انتخاب شده غیر فعال می‌شود و کاربر نمی‌تواند هیچ عملیاتی روی آن انجام دهد. مثلاً اگر یک متن را ایجاد کرده‌اید که با قرار گرفتن نشانه گر ماوس روی آن به رنگ دیگری تبدیل شود با غیر فعال کردن این گزینه دیگر عملیاتی انجام نمی‌شود.

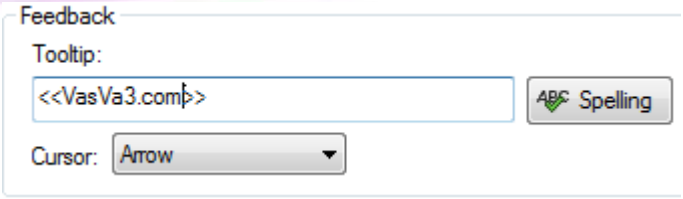
رنگ این اشیاء در تمام حالات به رنگ disable در قسمت پالت رنگ‌ها (خصوصیات) تغییر حالت خواهد داد.

2. visible: با غیر فعال کردن این گزینه در هنگام اجرای نرم افزار شی مورد نظر دیده نمی‌شود؛ یعنی مخفی می‌شود.

✓ اضافه کردن توضیحات

حتماً تا به حال در نرم افزاهای مختلف مشاهده کرده‌اید که با قرار گرفتن نشانه گر ماوس بر روی یک دکمه توضیحات در کادر ایجاد شده زیر آن نمایش داده می‌شود.

حال شما با این نرم افزار می‌توانید این قابلیت را به پروژه خود اضافه کنید. پس با رفتن به تنظیمات با دو بار کلیک کردن بر روی شی و رجوع به سربرگ attributes قسمتی به نام tooltip را مشاهده می‌کنید. حال در قسمت tooltip متن مورد نظر را تایپ کنید. در پایین tooltip برچسب cursor مشاهده می‌کنید. با انتخاب نوع cursor نشانه گر ماوس در زمان قرار گرفتن روی آن شی تغییر می‌کند.

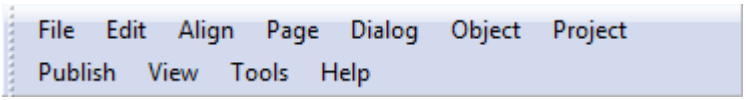


تصویر 1 - 28

تهیه نسخه دوم از اشیا

در صورتی که شما تنظیمات یک شی را اعمال کرده می‌توانید به جای ساخت اشیاء جدید و تنظیم تک تک آن‌ها نسخه المثنی از شی خود تهیه کنید و با تعویض نام آن را از دیگر اشیا متمایز سازید. برای این کار شی را انتخاب کرده و با انتخاب گزینه Duplication از منوی edit نسخه دیگری از آن ایجاد نمایید. علاوه بر این با کلیک راست کردن بر روی شی و انتخاب گزینه Duplication این کار ممکن است.

منو(Menu)



تصویر 1 - 29

File

با کلیک بر روی آن منوی فایل نمایان می‌شود. آیتم‌های مختلف این منو به شرح زیرند:

New: یک پروژه جدید ایجاد می‌کند.

Open: پروژه ای که از قبل آماده شده، بارگذاری می‌کند.

Save: آخرین تغییرات را در پروژه جاری ذخیره می‌کند.

Save As: پروژه جاری را با نام دیگری ذخیره می‌کند.

Export: فایل‌ها را با فرمت های خاص خود در حالت فشرده ذخیره می‌کند تا در موقع نیاز از آن استفاده کنید.

Revert: پروژه را به آخرین ذخیره باز می‌گرداند (مثلاً اگر چند دقیقه پیش برنامه را ذخیره کرده‌اید و پس از اعمال تغییرات متوجه اشتباهی شدید و نیاز به بازگشتن به وضعیت قبلی را دارید، می‌توانید این گزینه را انتخاب کنید.

Properties: اطلاعات پروژه را نمایش می‌دهد.

Recent File: نام آخرین پروژه های باز شده را نمایش می‌دهد.

Exit: از نرم افزار خارج می‌شود.

Edit

دومین گزینه در منو می‌باشد . با کلیک بر روی آن منو نمایش داده می‌شود که آیتم‌های آن به شرح زیر عمل می‌کنند:

Undo: برای بازگشت عمل آخری که انجام داده‌اید.

Redo: برای بازگرداندن عمل Undo شده.

Cut: بریدن شی برای استفاده در مکان و محلی دیگر.

Copy: از شی انتخاب شده کپی می‌گیرد.

Paste : شی Copy یا Cut شده را در محل مورد نیاز می‌چسباند.

Delete: شی ء انتخاب شده را پاک می‌کند.

Duplicate: برداشتن رو نوشت از روی شی

Select: انتخاب شی یا اشیاء

Arrange: تنظیم نحوه قرارگیری اشیاء.

Group & Ungroup: فعال و غیر فعال کردن گروه بندی اشیاء

Lock & Unlock All: قفل و آزاد کردن اشیاء

Pin & Unpin: سنجاق کردن و از بین بردن این حالت.

Hide & Unhide All: مخفی و نمایان کردن اشیاء.

Preferences: تنظیمات نرم افزار Auto Play

Align

سومین گزینه در منوها گزینه‌ی Align می‌باشد که گزینه‌هایی برای تنظیم چیدمان اشیای موجود در پروژه گنجانده شده است.

Left: سمت چپ شی را با سمت چپ شی انتخابی همتراز می‌کند

Center Horizontal: وسط(افقی) شی را با وسط شی انتخابی همتراز می‌کند

Right: سمت راست شی را با سمت راست شی انتخابی همتراز می‌کند

Top: شی را با بالای شی انتخابی همتراز می‌کند

Center Vertical: وسط(عمودی)شی را با وسط شی انتخابی همتراز می‌کند.

Bottom: شی را با پایین شی انتخابی همتراز می‌کند.

Distribute Vertical و Distribute Horizontal: فاصله بین اشیاء انتخابی را (افقی و عمودی) تنظیم می‌کند.

Make Same Width: طول شی را برابر طول شی دیگر قرار می‌دهد.

Make Same Height: عرض شی را برابر عرض شی دیگر قرار می‌دهد.

Make Same Size: طول و عرض را برابر طول و عرض شکل دیگر قرار می‌دهد.

Restor Size: طول عرض تغییر یافته را به حالت اولیه باز می‌گرداند.

To Page: با انتخاب این گزینه تمام عملیات بالا بدون نیاز به شی دیگر و با توجه به قرارگیری در صفحه انجام می‌دهد.

در تمام گزینه های بالا شی که در انتها انتخاب شده و کادر آبی رنگ آن پررنگ‌تر است،ثابت مانده و بقیه تغییر می‌کنند.

تصویر 1 - 30

Page

گزینه‌ی بعدی در منوها گزینه‌ی Page می‌باشد که آیتم‌های آن به صورت زیر عمل می‌کنند.

Add: ساخت صفحه ای جدید(صفحه جدید را در انتهای صفحات قرار می‌دهد)

Remove: پاک کردن صفحه‌ی انتخاب شده

Insert: ساخت یک صفحه جدید یا استفاده از صفحه Export شده در محل دلخواه(مانند گزینه‌ی Add در انتها قرار نمی‌دهد.)

Duplicate:کپی برداری از صفحه

Arrange: تنظیم ترتیب قرارگیری صفحات

Move Up: جا به جا کردن صفحه به اندازه‌ی یک خانه رو به عقب(بالا)

Move Down: جا به جا کردن صفحه به اندازه‌ی یک خانه رو به جلو(پایین)

Export: ذخیره صفحه انتخاب شده با فرمت XPG یا Zip

Preview: پیش نمایش صفحه

Properties: مشاهده مشخصات و خصوصیات صفحه

Dialog

دیالوگ‌ها در AMS[1] در اصل نوعی صفحه هستند با این تفاوت که دیالوگ‌ها بر روی صفحات ظاهر می‌شوند و تا زمانی که پنجره دیالوگ بسته نشود صفحه به صورت غیر فعال باقی می‌ماند.

آیتم‌های منوی دیالوگ به صورت زیر عمل می‌کنند:

Add: ساخت دیالوگ جدید

Remove: پاک کردن دیالوگ حاضر

Insert: ساخت یک دیالوگ جدید یا استفاده از دیالوگ Export شده در محل دلخواه(مانند گزینه‌ی Add در انتها قرار نمی‌دهد.)

Duplicate:کپی برداری

Arrange: تنظیم ترتیب قرارگیری دیالوگ‌ها

Move Up: جا به جا کردن دیالوگ به اندازه‌ی یک خانه رو به عقب(بالا)

Move Down: جا به جا کردن دیالوگ به اندازه‌ی یک خانه رو به جلو(پایین)

Export: ذخیره دیالوگ جاری با فرمت XDG یا Zip

Preview: پیش نمایش

Properties: مشاهده مشخصات و خصوصیات دیالوگ

Object

از دیگر گزینه های منو، Object می‌باشد که به شرح آیتم‌های آن می‌پردازم:

Button: اضافه کردن دکمه

تصویر 1 - 31

Image: اضافه کردن تصویر

2. در این کتاب منظور از AMS همان Autoplay Media Studio می باشد.

تصویر 1 - 32

Lable: اضافه کردن برچسب

New Label

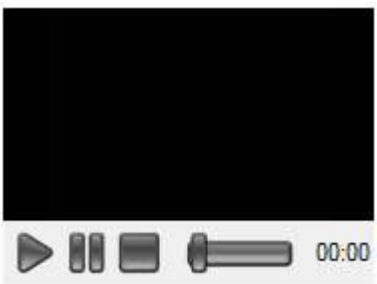
تصویر 1 - 33

Paragraph: اضافه کردن یک پاراگراف

By Hamed and Javad
My Forum is: Vasva3.com
By Hamed and Javad
My Forum is: Vasva3.com

تصویر 1 - 34

Video:اضافه کردن فایل ویدئویی



تصویر 1 - 35

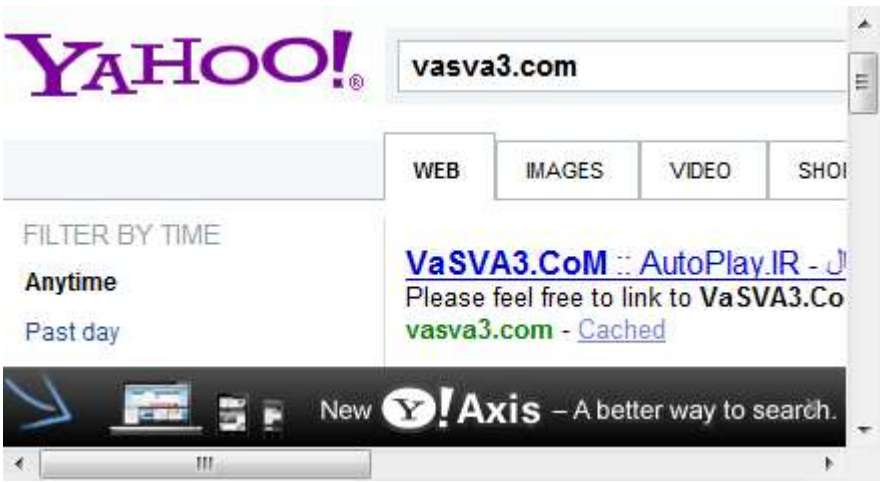
Quick Time: اضافه کردن فایل ویدئویی و کنترل به وسیله نرم افزار Quick Time

Flash: اضافه کردن یک فلش

Slide Show: اضافه کردن یک اسلاید شو به پروژه

PDF: اضافه کردن فایلPDF و کنترل به وسیله نرم افزار Adobe Reader

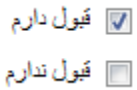
Web:اضافه کردن شی Web برای باز کردن سایت‌های اینترنتی و فایل‌های متنی داخل برنامه Auto Play



تصویر 1 - 36

Xbutton: اضافه کردن یک Xbutton به پروژه

CheckBox: اضافه کردن شی برای چک کردن یک موضوع یا مطلب



تصویر 1 - 37

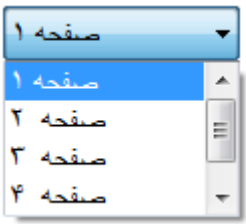
RadioButton: اضافه کردن شی برای انتخاب از بین چند گزینه(مثل سؤالات تستی)

تصویر 1 - 38

Input: اضافه کردن شی ورودی

تصویر 1 - 39

ComboBox:اضافه کردن لیست کرکره ای

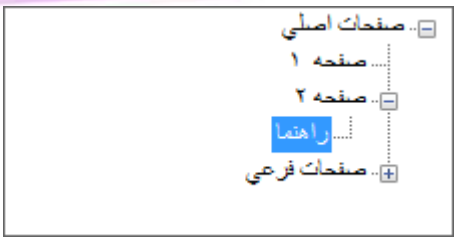


تصویر 1 - 40

ListBox: اضافه کردن یک لیست

تصویر 1 - 41

Tree: اضافه کردن لیست درختچه ای



تصویر 1 - 42

Grid: اضافه کردن یک گرید

نام	نام خانوادگی
جواد	احشامیان
حامد	حیدري

تصویر 1 - 43

Progress: اضافه کردن یک شی پیش رونده(مثل نصب نرم افزار)

تصویر 1 - 44

RichText: اضافه کردن باکس متن (RTF Document)

کاری از حامد حیدری و جواد احشامیان
مدیران انجمن وسو۳

تصویر 1 - 45

Hotspot: اضافه کردن شی نامرئی برای اضافه کردن دستورات

Plugins: اضافات برنامه برای استفاده(از قبیل تقویم، مدیا پلیر و ...)

Properties: مشاهده خصوصیات شی انتخاب شده

Project:

گزینه‌ی بعدی در منوها Project می‌باشد که از ده آیتم به شرح زیر تشکیل شده است:

Setting: تنظیمات مربوط به پروژه را در اینجا تغییر می‌دهید

Menu bar: ساخت منو بار برای پروژه(مثل File , Edit,...)

Audio:تنظیمات آهنگ و صداهای نرم افزار

Startup Movie: نمایش تصویر، فایل فلش یا ویدئو در آغاز اجرای نرم افزار

Dependencies:چک کردن نرم افزارهای خاص که در پروژه نیاز است و پیغام دادن به کاربر در صورت نصب نبودن آن نرم افزار

Action: دستورات برای اجرا در پروژه

Databases: استفاده از دستورات دیتابیس در پروژه

Global Function: تعیین متغیرها و توابع سراسری

Plugins: افزونه های نرم افزار

File Layout: مشاهده فایل‌های موجود و استفاده شده در پروژه و قابلیت اضافه یا کم کـردن این فایل‌ها

Publish

گزینه‌ی بعدی Publish است که 2 آیتم آن به صورت زیر عمل می‌کنند.

Preview: مشاهده پروژه

Build: ساخت و گرفتن خروجی برای استفاده از پروژه

View

آیتم‌های این گزینه که مخصوص چگونگی نمایش گزینه‌ها و پنل‌ها می‌باشد به شرح زیرند:

Toolbars: اضافه کردن یا حذف تولبارها

Panels: اضافه کردن یا حذف پنل‌ها

Layouts: تغییر ظاهری برنامه

Previous Page:رفتن به صفحه قبلی

Next Page: رفتن به صفحه بعدی

Find Page: پیدا کردن یک صفحه

Refresh: رفرش(تجدید و بازبینی دوباره)

Grid: نمایش یا عدم نمایش جدول بندی صفحه

Snap to Grid: قرار دادن اشیاء در جدول بندی‌ها

Snap to Page: قرار دادن اشیا در صفحه

Guidelines:نمایش یا عدم نمایش خطوط راهنما

Ruler:نمایش یا عدم نمایش خط کش

Tools

گزینه‌ی نهم که آیتم‌های آن به صورت زیر عمل می‌کنند:

Button Maker: نرم افزار همین شرکت برای ساخت دکمه دلخواه

Optimize Resources:پاک کردن فایل‌های زائد پروژه

Script Explorer: نمایش تمام دستورات نوشته شده

Script Editor: ویرایش دستورات در برنامه کوچک جداگانه

Open Source File:باز کردن فایل اصلی شی انتخاب شده

Customize: تنظیمات مربوط به منوها

Help

در این منو گزینه های برای کمک به کاربر در مورد نحوه‌ی کار با برنامه، اطلاعاتی در مورد برنامه و مواردی از این قبیل موجود هستند که نیازی به توضیح ندارند.

نوارهای ابزار

در محیط AMS چندین نوار ابزار وجود دارد که دو مورد از آن‌ها به صورت پیش‌فرض فعال می‌باشند که پرکاربردترین آن‌ها هستند(Standard و Objects). برای حذف یا اضافه کردن نوارهای ابزار می‌توانید از منوی View >Toolbars استفاده کنید.

این نوار ابزارها باعث می‌شوند شما به صورت سریع به دستورات پرکاربرد دسترسی داشته باشید و مجبور نشوید منوها را بررسی کنید و دنبال گزینه مورد نظر بگردید. برای مثال گزینه‌ی File >New از نوار منو، در نوار ابزار استاندارد همان گزینه اول در سمت چپ می‌باشد.

تصویر 1 - 46

هم اکنون کاربرد مختصر هر یک از ابزارهای موجود در نوار ابزار Standard و سپس نوار ابزار Objects را با توجه به تصویر 1- 5 را با هم مرور می‌کنیم.

نوار ابزار Standard :

1. ایجاد یک پروژه جدید
2. باز کردن یک پروژه ذخیره شده
3. ذخیره پروژه
4. بریدن شی یا اشیای انتخاب شده
5. کپی اشیای انتخاب شده
6. جایگذاری اشیای کپی شده یا بریده شده
7. بازگشت مرحله به مرحله به حالت قبلی

8. رفتن به جلو به صورت مرحله ای
9. اجرای پروژه جهت تست
10. ساخت پروژه نهایی

نوار ابزار Objects :

11. افزودن دکمه به پروژه
12. افزودن عکس
13. افزودن متن برچسب
14. افزودن متن به صورت پاراگراف
15. افزودن فیلم
16. افزودن فیلم‌های قابل بخش در نرم افزار Quick Time
17. افزودن فایل فلش swf
18. افزودن اسلاید شو
19. افزودن فایل pdf
20. افزودن فایل وب یا html
21. افزودن دکمه‌ی سیستمی با امکاناتی خاص
22. افزودن CheckBox (علامت گذاری)
23. افزودن RadioButton
24. افزودن متن از نوع وردی یعنی می‌توان در حین اجرا متنی را درون آن تایپ کرد.
25. افزودن منوی کشویی (ComboBox)
26. افزودن یک لیست (ListBox)
27. افزون لیستی از نوع درخت‌واره که هر قسمت از لیست می‌تواند دارای زیر شاخه‌هایی باشد.

28. افزون Grid که نوعی جدول به حساب می‌آید و بسته به تنظیم کاربر می‌توان تعداد سطرها و ستون‌های آن را کم و زیاد کرد.

29. افزودن Progress یا نوار پیشرفت که به وسیله آن می‌توان مقدار پیشرفت یک مورد را مشخص کرد مثل نمایش مقدار فایل کپی شده

30. افزودن متن از نوع RichText یا همان rtf

31. افزودن HotSpot که در هنگام اجرای برنامه به چشم نمی‌آید اما می‌توان دستوراتی را برای آن تعیین نمود که با کلیک ماوس بر روی آن اجرا شوند.

32. زمانی که شما یک شی را بر روی صفحه پروژه انتخاب کنید این گزینه فعال می‌شود که با کلیک بر روی آن می‌توان خصوصیات آن شی را مشاهده نمود.

نکته: اگر فراموش کردید که هر آیکون در نوارهای ابزار چه کاری انجام می‌دهند نشانگر ماوس را روی آن قرار دهید تا نام آن آیکون در یک کادر کوچک نمایان شود.

برای دیدن بقیه ابزار و پنجره‌ها می‌توان از منوی View اقدام به این کار کرد اما بهتر است آن‌ها را در طول نوشتن یک پروژه‌ی ساده با AMS بررسی کنیم.

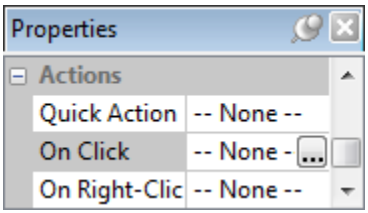
اکنون به سراغ ایجاد پروژه می‌رویم، از منوی Project روی گزینه‌ی Settings کلیک کنید و سپس از قسمت Dimension، از منوی کشویی، Tiny را انتخاب کنید تا اندازه صفحه پروژه شما به حداقل پیش‌فرض برسد و Ok را کلیک کنید.

از نوار ابزار Objects و یا از منوی Objects روی شی Button (دکمه) کلیک کنید سپس در پنجره ظاهر شده یک مورد را انتخاب کنید و روی دکمه‌ی Ok کلیک نمایید. با این کار یک دکمه به صفحه پروژه شما اضافه می‌گردد و در ابتدای صفحه قرار می‌گیرد که شما می‌توانید با فشردن کلیک ماوس و پایین نگه داشتن آن بر روی شی دکمه آن را جا بجا نمایید و در محل مورد نظر رها کنید.

برای نوشتن متن درون دکمه روی آن کلیک کنید و سپس در پنجره Properties در مقابل عبارت Text کلمه‌ی Hello را تایپ نمایید. هم چنین می‌توانید برای این کار روی شی دکمه دو بار کلیک کنید و سپس پنجره ظاهر شده در قسمت Text متن را یادداشت نمایید.

حال به قسمت Action از پنجره‌ی Properties بروید و مقابل عبارت On Click روی دکمه‌ای با علامت ... کلیک کنید.(تصویر 1- 11)

نکته: با دو بار کلیک کردن روی شی و رفتن به سربرگ Script نیز امکان‌پذیر است.



تصویر 1 - 47

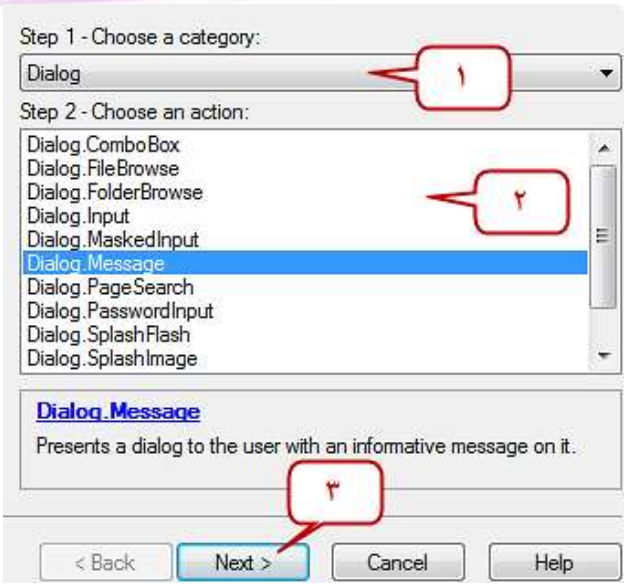
سپس در پنجره ظاهر شده روی دکمه‌ی Add Action کلیک کنید کلیک کنید و در پنجره New Action Wizard از قسمت Step1 از منوی کشویی گزینه‌ی Dialog را انتخاب کنید (در این حال مشاهده می‌کنید که گزینه های لیست موجود در قسمت Step2 نیز تغییر کرده‌اند) حال از Step2 گزینه‌ی Dialog.Message را انتخاب کنید و روی دکمه‌ی Next کلیک کنید(تصویر 1-12).

و در قسمت Step3 جلوی عبارت Title کلمه‌ی "Welcome" را تایپ نمایید دقت کنید که Welcome باید میان دو کاراکتر "" قرار گیرد.

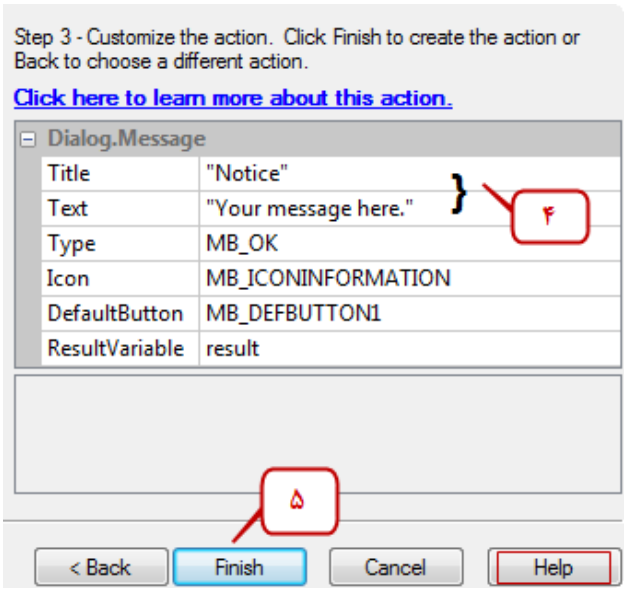
حال در جلوی عبارت Text جمله‌ی "Welcome to AMS !" را تایپ کنید و بر روی دکمه‌ی Finish کلیک کنید(تصویر 1 – 13) و سپس روی دکمه‌ی Ok کلیک نمایید.

هم اکنون می‌توانید نتیجه کار خود را مشاهده نمایید برای این کار دکمه‌ی F5 را از صفحه کلید فشار دهید. و یا از نوار ابزار روی دکمه‌ی Preview کلیک نمایید.

برنامه ساخته شده توسط شما اجرا می‌گردد که تقریباً شبیه به شکل (1- 14) است که با کلیک بر روی دکمه‌ی hello پیامی در یک کادر با عنوان خوش آمد گویی به محیط AMS به شما نمایش داده خواهد شد.



تصویر 1 - 48



تصویر 1 - 49

تصویر 1 - 50

دیدید که کار بسیار راحت و آسانی بود شما یک برنامه ساده نوشتید آن هم بدون اینکه یک سطر کد به صورت دستی بنویسید و تمامی مراحل کد نویسی به صورت جاودیی یا همان Wizard انجام گرفت و آن کد زیر است:

```
result = Dialog.Message("Welcome", "Welcome to AMS !", MB_OK,
MB_ICONINFORMATION, MB_DEFBUTTON1);
```

نکته: به علت کوچک بودن صفحه نمی‌توان کد را در یک سطر جا داد اما در محیط برنامه این کد در یک سطر قرار می‌گیرد. هم چنین به خاطر داشته باشید که کدها می‌توانند بیشتر از یک سطر باشند و گاهی حتی به صدها سطر می‌رسند.

نکته: علامت ; نشانگر پایان کد مورد نظر می‌باشد.

برای بستن کادر پیغام روی دکمه‌ی Ok کلیک کنید و برای بستن برنامه روی دکمه‌ی Close که شبیه ضربدر است کلیک کنید.

نام‌گذاری محاسباتی اشیا: مورد دیگری شاید برایتان جالب باشید این است که هر شی‌ای که به پروژه اضافه کنید با توجه به نوع آن شی نام‌گذاری می‌شود و یک عدد به انتهای آن اضافه می‌شود. مانند: Button1

توصیه می‌کنیم برای اشیا نام مناسب انتخاب کنید تا هنگام کد دهی به آن‌ها دچار مشکل نشوید.

✓ اضافه کردن شی تصویر (Image object)

در نرم افزار AMS تمام اشکال، متون و ... که برای کار وجود دارند شی نام می‌گیرند. یکی از این اشیاء تصویر می‌باشد که در تعامل با کاربر نقش بسزایی دارند. چرا که هم مفهوم را بیشتر به کاربر رسانده و هم پروژه را جذاب‌تر می‌کند. برای اضافه کردن یک تصویر از منوی object بر روی Image کلیک کنید .

و یا از نوار وظیفه بر روی آیکون Image کلیک کنید.( ) حال از پنجره‌ی انتخاب فایل، یک تصویر را انتخاب کنید و بر روی ok کلیک نمایید با اضافه کردن تصویر، این شی به سمت چپ و بالای صفحه اضافه می‌شود.

تصویر 1 - 51

✓ **تغییر اندازه و موقعیت اشیاء** :

بر روی تصویری که اضافه کرده‌اید 1 بار کلیک کنید تا آن را انتخاب نمایید. دور تصویر انتخاب شده، کادر آبی رنگی مشاهده می‌شود که مشخص کننده‌ی شی انتخاب شده می‌باشد. در روش اول 8 دایره را در اطراف تصویر مشاهده می‌کنید که با قرار گرفتن نشانه گر ماوس بر روی آن‌ها، نشانه گر به اشکال ↔ و و ↙↗ تغییر می‌کند. با نگه داشتن کلیک چپ ماوس و کشیدن ماوس به اطراف، اندازه‌ی شی شما تغییر می‌کند.

تصویر 1 - 52

همچنین برای تغییر مکان شی نشانه گر ماوس را بر روی شی برده (داخل کادر آبی رنگ) کلیک سمت چپ ماوس را نگه داشته و به اطراف منتقل کنید.

روش دوم و سوم روش‌هایی دقیق‌تر می‌باشند که به شرح هر یک خواهم پرداخت.

در روش دوم با انتخاب شی در پنل خصوصیات، خصوصیات مربوط به شی انتخاب شده نمایش داده می‌شود که شما باید به دنبال position بگردید. در پایین Position، 4 فیلد را مشاهده می‌کنید.

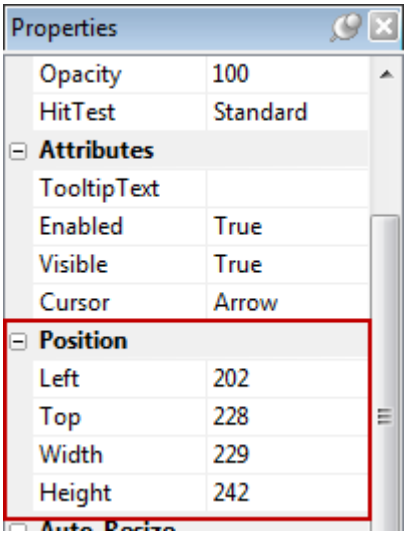
Left: فاصله‌ی شما از سمت چپ صفحه

Top: فاصله‌ی شی از بالای صفحه

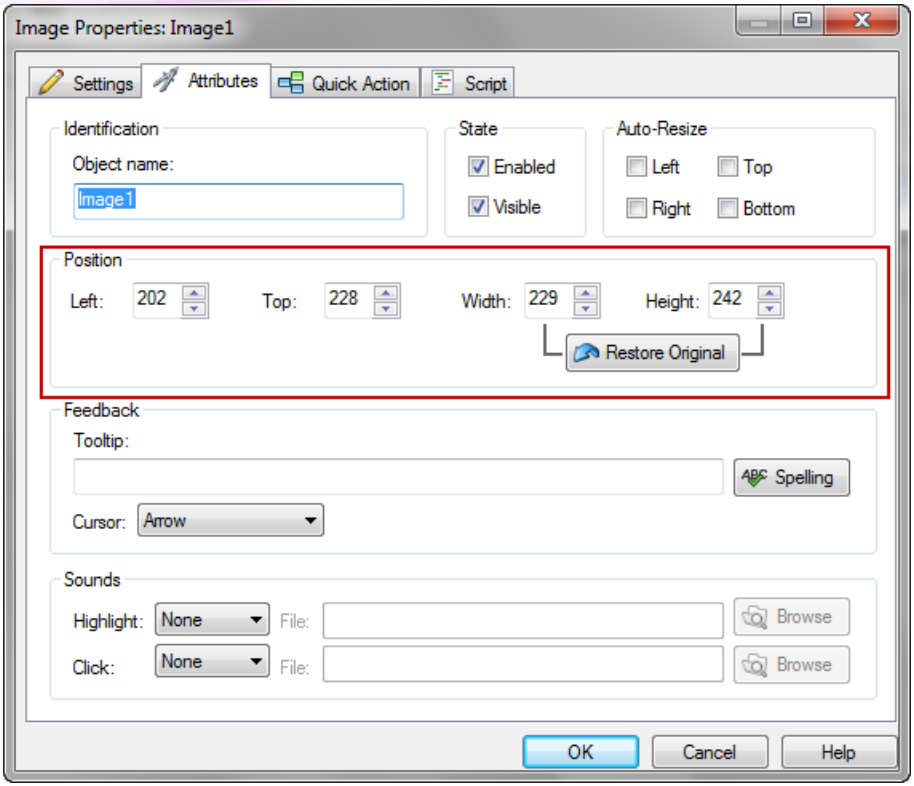
Height, width: به ترتیب پهنا و ارتفاع (طول و عرض) شی را تنظیم می‌کنند.

در روش سوم با 2 بار کلیک کردن روی شی پنجره‌ی خصوصیات آن باز می‌شود. از بالای صفحه سربرگ Attributes را انتخاب کنید. در قسمت position مانند روش دوم 4 فیلد

Left، Top، Height, width را می‌بینید که تنظیمات آن‌ها، تنظیمات روش قبلی یکسان می‌باشد. بر روی ok کلیک کنید.



تصویر 1 - 53

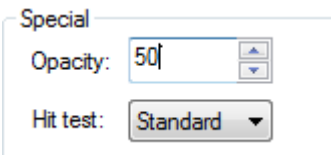


تصویر 1 - 54

در صورتی که چند شی یا تصویر را وارد پروژه کرده‌اید و نیاز به حذف آن‌ها دارید، با انتخاب شی و فشردن دکمه‌ی Del از روی صفحه کیبورد، می‌توانید شی را حذف کنید.

کمرنگ کردن تصاویر

در صورتی که نیاز به کمرنگ کردن تصویر (شیشه ای کردن آن) داشته باشید به قسمت خصوصیات رفته (2 بار کلیک بر روی تصویر) و فیلد متغیر opacity را کم کنید . (بازه opacity بین صفر تا صد می‌باشد.)



تصویر 1 - 55

✓ اضافه کردن شي برچسب :

برچسب‌ها نوشته های کوتاهی برای توضیح یک شی یا نوشتن ایمیل خود یا ... داخل آن است.

با انتخاب گزینه‌ی Label از منوی object یک برچسب به صورت خودکار بر روی صفحه و در قسمت بالا، سمت چپ صفحه ایجاد خواهد شد. البته لازم به ذکر است با راست کلیک کردن بر روی صفحه و کلیک بر روی Label و یا کلیک بر روی آیکون Label از قسمت نوار وظیفه امکان پذیر است.( )

تصویر 1 - 56

حال برای تغییر خصوصیات برچسب، دو/2 بار بر روی آن کلیک کنید تا صفحه‌ی تنظیمات آن باز گردد. در ابتدا به توضیح روش نوشتن متن دلخواه در برچسب می‌پردازیم. برای این کار در فیلد مقابل Text متن دلخواه خود را جایگزین متن نمایید. (در این پروژه ایمیل خود را وارد نمایید.)

متن را به صورت زیر تایپ کنید:

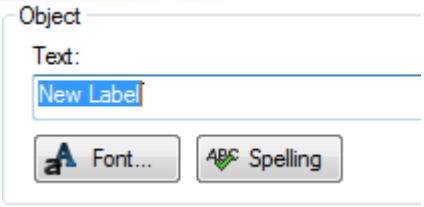
My E- mail: My_Email@AutoPlay.IR

پیکر بندي برچسب

حال برای پیکربندی برچسب خود دوباره بر روی برچسب کلیک کنید تا صفحه‌ی تنظیمات آن باز شود. همچنین برای این کار می‌توانید با کلیک راست کردن بر روی برچسب و انتخاب گزینه‌ی properties این پنجره را باز کنید.

جایگزین کردن متن:

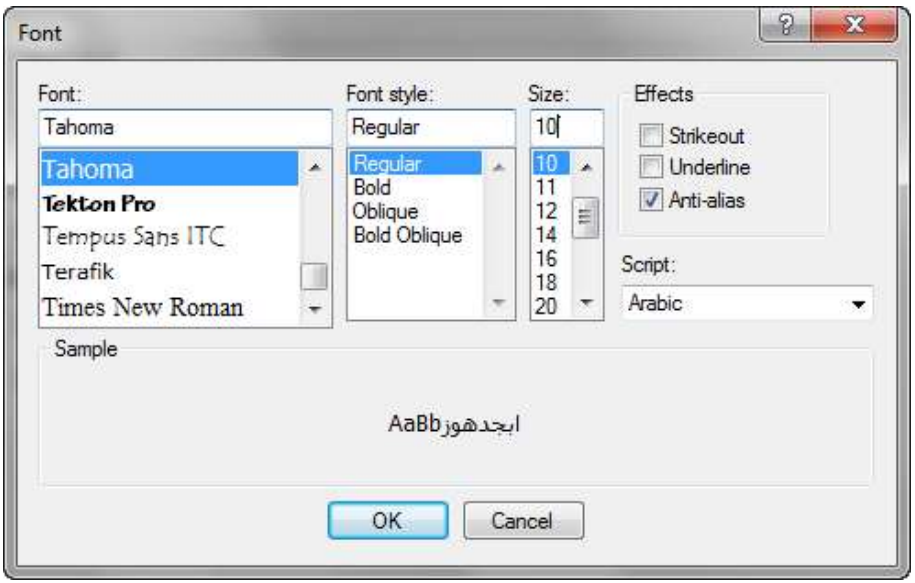
برای جایگزین کردن متن مورد نظر خود به جای متن قبلی و یا ویرایش آن در زیر قسمت Text و در فیلد ورود اطلاعات آن متن خود را وارد نمایید.



تصویر 1 - 57

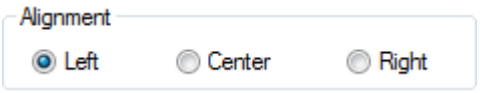
تنظیم فونت متن:

برای تنظیم فونت، سایز و ... متن خود بر روی دکمه‌ی font کلیک کنید در پنجره‌ی باز شده شکل خط نوشته را انتخاب کنید همچنین برای تغییر اندازه‌ی نوشته می‌توانید از قسمت سایز استفاده کنید.



تصویر 1 - 58

جاي گيري مناسب



تصویر 1 - 59

در پنجره تنظیمات برچسب Alignment را مشاهده می‌کنید. این گزینه نوع قرارگیری متن را در برچسب مشخص می‌کند که به شرح زیر می‌باشد.

Right: متن را راست چین می‌کند. (از سمت راست به چپ قرار می‌دهد.)

Center: متن را وسط چین می‌کند. (در وسط قرار می‌دهد.)

Left: متن را چپ چین می‌کند. (از سمت چپ به راست قرار می‌دهد.)

زاویه‌ی برچسب:

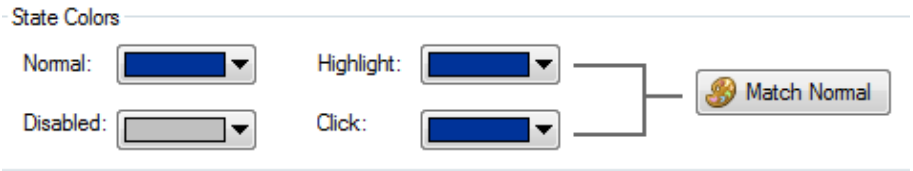
تصویر 1 - 60

در پنجره‌ی تنظیمات برچسب Orientation وجود دارد که برای تنظیم زاویه (جهت) شی برچسب مورد استفاده قرار می‌گیرد.

رنگ برچسب:

رنگ‌ها در نرم افزارهای مولتی مدیا نقش مهمی را ایفاء می‌کنند. چرا که این رنگ‌ها برای رساندن بهتر مفهوم، بالا بردن گرافیک نرم افزار، رفع خستگی از کاربر و ... استفاده می‌شود.

در پنجره‌ی تنظیمات، State color برای تنظیم رنگ‌ها قرار داده شده است که متشکل از چهار نوع جعبه‌ی رنگ به شرح ذیل می‌باشد.



تصویر 1 - 61

Normal: رنگ اصلی نوشته را از این قسمت می‌توان انتخاب کرد.

Highlight: رنگی که در این قسمت انتخاب می‌کنید زمانی نشان داده می‌شود که نشانه گر ماوس بر روی آن قرار گیرد.

Click: زمانی که بر روی برچسب کلیک می‌کنید رنگ آن به رنگ انتخاب شده در این قسمت تبدیل خواهد شد.

Disable: این رنگ زمانی مورد استفاده قرار می‌گیرد که شی غیر فعال شود.

دکمه‌ی match normal: این دکمه رنگ جعبه رنگ‌های Highlight و Click را به رنگ نرمال normal تغییر خواهد داد.

تصویر 1 - 62

اضافه کردن شی پاراگراف

از منوی object گزینه paragraph را انتخاب کنید، در این حالت در گوشه صفحه متنی را مشاهده می‌کنید، لازم به ذکر است برای اضافه کردن پاراگراف با کلیک راست کردن بر روی صفحه و انتخاب گزینه paragraph و همین طور با انتخاب آیکون مربوطه از نوار وظیفه می‌توانید این شی را به پروژه خود اضافه کنید.

تغییر اندازه شی

این شی را هم مانند اشیاء دیگر می‌توان با انتخاب و فشردن کلیک ماوس بر روی دایره های اطراف شی اندازه آن را تغییر داد، ولی بهتر آن است که از قسمت خصوصیات این کار را انجام . در صورتی که متن شما جای کافی برای قرارگیری در شی پاراگراف را نداشته باشد، اسکرول بار سمت راست به طور خودکار فعال شده و نمایش داده می‌شود.

غیر فعال کردن scroll bar

برای غیر فعال کردن scroll bar باید از پنل خصوصیات استفاده نمایید، در اینجا scroll bar را یافته و برچسب vertical را در قسمت پایین آن بیابید. حال از لیست کرکره ای مقابل Vertical Auto را به off تغییر دهید، این گزینه اسکرول بار سمت راست را از بین می‌برد.

ویرایش متن

بر روی شی پاراگراف خود دو بار کلیک کنید، پنجره خصوصیات پاراگراف " properties paragraph " باز می‌شود، از سربرگ setting دکمه font را کلیک کنید، پنجره مربوط به font گشوده خواهد شد. در این پنجره فونت Arial را انتخاب و قالب فونت را تو پُر، اندازه فونت را 14 و همچنین اسکریپت آن را به Arabic تغییر دهید، حال بر روی ok کلیک نمایید. پنجره فونت بسته شده و پنجره خصوصیات پاراگراف مشاهده می‌شود. در قسمت Alignment گزینه center را انتخاب کنید تا متن در وسط شی پاراگراف قرار گیرد.

حال از قسمت state colors رنگ‌های خود را انتخاب کنید و بر روی ok کلیک کنید.

تفاوت شی پاراگراف با شی Rich text

به جای استفاده از پاراگراف، می‌توانید از شی Rich text استفاده کنید. از تفاوت‌های این دو شی می‌توان به قابلیت بارگذاری فایل‌های RTF توسط Rich text اشاره کرد .

کار با این شی نیز آسان است. پس از 2 بار کلیک کردن بر روی شی می‌توانید متن خود را تایپ کنید و با استفاده از دکمه‌ها آن را Bold، Italic و با زیر آن خط انداختن و رنگ و اندازه آن را تغییر دهید.

اضافه کردن یک شي ویدئو

در ابتدا کار لازم است ذکر کنم که زمانی یک شی ویدئو را به پروژه خود اضافه می‌کنید، این شی به صورت پیش فرض برنامه به روی تمامی اشیاء موجود در صفحه قرار می‌گیرد. و به هیچ عنوان به زیر آن‌ها نخواهد رفت حتی با (arrange) برای اضافه کردن یک شی برنامه از منوی object گزینه ویدئو را انتخاب کنید، با انتخاب این گزینه پنجره select File باز خواهد شد. در این پنجره به محل قرارگیری فایل تصویری بروید و آن را به پروژه اضافه کنید.

یک ویدئو جدید به پروژه شما اضافه خواهد شد. به صورت پیش فرض یک نوار کنترل بر روی ویدئو شما قرار می‌گیرد . بر روی این نوار کنترل دکمه های پخش- مکث – توقف و همچنین زمان سپری شده فیلم و میله وضعیت فیلم وجود دارد.

تصویر 1 - 63

✓ Control Style :

حال از پنل خصوصیات control style را از standard به Basic blue تغییر دهید تا تغییرات style نوار کنترل را مشاهده کنید.

همان طور که مشاهده می‌کنید برای شی ویدئو و هم چنین پاراگراف style وجود دارد. حال اگر از قالب‌های موجود راضی نیستید و خودتان یک قالب جدید در نظر دارید خواهید توانست آن را ایجاد کنید. به این صورت که به شاخه‌ی اصلی نرم افزار مراجعه کرده و از آنجا شاخه‌ی plugins را باز نموده و در شاخه های transports , scrollbars می‌توانید قالب‌ها را اصلاح کنید. لازم به ذکر است که در شاخه‌ی transports نام فایل‌ها باید با "VT" و در شاخه‌ی (Scrollbars) فایل‌ها باید با نام "SB" شروع شوند.

در پنل خصوصیات و دسته control panel شش قسمت تنظیمات وجود دارد که عبارتند از:

style: تنظیم قالب نوار کنترل که در صفحه قبل توضیح داده شد.

Time: برای نمایش زمان است که عبارتند از:

None: حذف زمان

Elapsed: زمان سپری شده

Length: زمان کل ویدئو

Both: هم زمان سپری شدن و هم زمان کل

panel : تنظیم رنگ زمینه نوار کنترل

Text: رنگ نوشته‌ها

Control button: حذف یا نمایش دکمه‌ها (نمایش- مکث – توقف) که متشکل از true و false است(بر روی true تنظیم نمایید.)

Slider: نمایش یا عدم نمایش موقعیت فیلم

در دسته‌ی special لیست start Auto را باز کرده و بر روی true تنظیم نمایید. این کار باعث می‌شود به محض اجرای صفحه ویدئو فایل video شما شروع به نمایش نماید و loop را نیز به False تغییر دهید تا پس از پایان، video دوباره شروع به نمایش ننماید.

حال در دسته‌ی object، لیست کرکره ای video scaling mode را به maintain Aspect تنظیم نمایید. در این حالت اندازه صفحه‌ی شی ویدئو به طور خودکار تنظیم می‌شود.

در حقیقت طول و عرض صفحه‌ی شی متناسب با یکدیگر تغییر می‌کنند. در حالی که اگر آن را به stretch تغییر دهید، سایز صفحه‌ی شی با سایز خود ویدئو با هم برابر خواهند شد.

در صورتی که نیاز دارید سایز شی ویدئو دقیقاً برابر سایز اصلی فایل تصویر بشود، بر روی شی کلیک راست کرده و Restore size را انتخاب نمایید. شی ویدئو را بر روی قاب تنظیم نمایید.

افزودن شی فلش

برای اضافه کردن یک شی فلش از منوی Object گزینه‌ی Flash را انتخاب و یا بر روی آیکون Flash کلیک کنید.( ) فایل‌های فلش توسط نرم افزار Adobe Flash ساخته می‌شوند ولی با نرم افزارهای دیگری مانند Swish Max نیز می‌توانید این فایل‌ها را تولید و برای جلوه و زیبایی و بالا بردن گرافیک نرم افزار از آن استفاده کنید.تمام عملیات‌های عادی و مورد نیاز مثل جابجا کردن، تغییر سایز و... مانند اشیاء دیگر می‌باشد.

تصویر 1 - 64

تنظیمات فلش :

بر روی شی 2 بار کلیک کنید تا پنجره‌ی خصوصیات آن باز شود.تمام تنظیمات اصلی این شی در سربرگ Setting می‌باشد.

Alignment: این قسمت برای زمانی است که اگر شی فلش شما از فایل فلش بزرگ‌تر باشد، نوع قرارگیری آن در شی را تنظیم می‌کند.

Context Menu: نوع نمایش منو را در هنگام کلیک راست کردن بر روی شی نشان می‌دهد.

تصویر 1 - 65

در صورتی که فایل فلشی را در پروژه استفاده می‌کنید که نیاز به دسترسی به عناصر آن ندارید، با غیرفعال کردن آن (برداشتن تیک Enable) قابلیت نمایش کلیک راست از غیرفعال می‌شود.

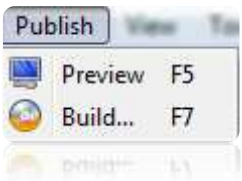
Quality: میزان کیفیت فایل فلش را تعیین می‌کند.

Scaling Mode: اگر شی را بزرگ یا کوچک کنید، با این گزینه می‌توانید نوع قرارگیری آن در داخل شی را تنظیم کنید.بدین گونه که در داخل -شی قرار گیرد یا به نسبت سایز آن تغییر کند.

انتشار پروژه

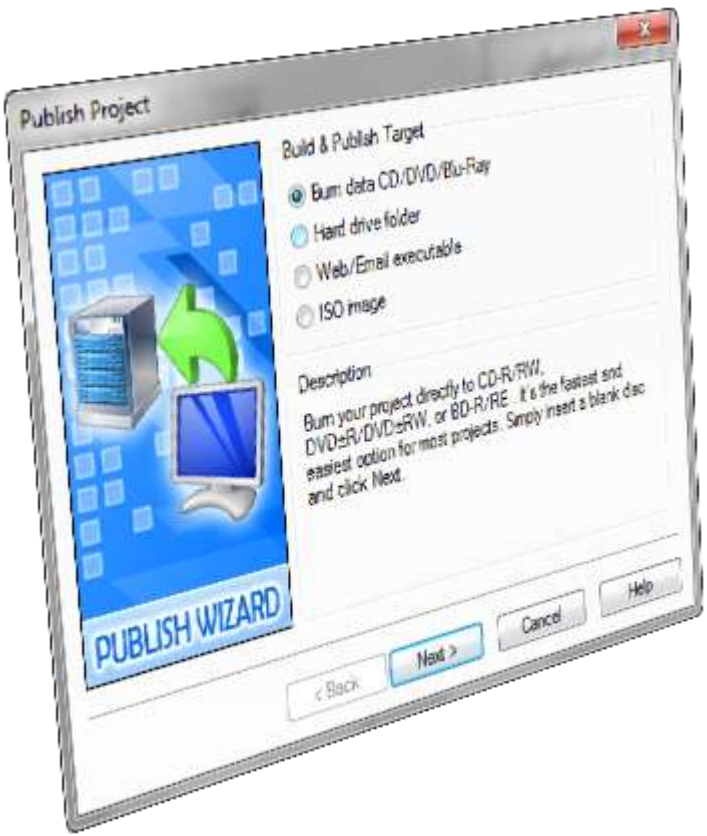
در این قسمت درباره‌ی روش‌های ساخت و انتشار برنامه(پروژه) تان بحث خواهیم کرد.

از منوی Publish، Build را انتخاب کنید. (برای این کار می‌توانید از دکمه‌ی F7 از صفحه کلید استفاده نمایید.)



تصویر 1 - 66

پنجره publish project باز خواهد شد. در این پنجره 2 قسمت مجزای Build publish Target و دیگری Description می‌باشد که Build & publish Target مربوط به انتخاب چگونگی انتشار پروژه و Description توضیحات مربوط به Build & publish Target می‌باشد. به شرح Build publish Target می‌پردازیم.

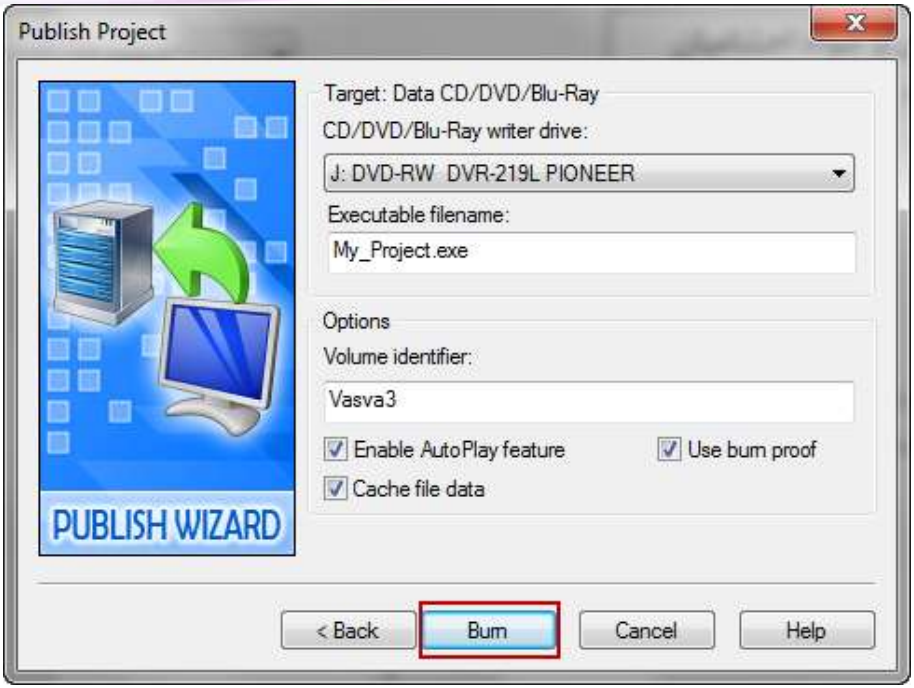


تصویر 1 - 67

رایت بر روي CD

یکی از گزینه های موجود در این بخش Burn date DVD/CD است. زمانی که پروژه به پایان رسید و نیاز به ذخیره‌ی مستقیم آن بر روی CD, DVD دارید باید این گزینه را انتخاب کنید. لازم به ذکر است که برای این کار نیاز به رایتر دارید. این گزینه را انتخاب کنید.یعنی بر روی آن کلیک کنید تا توپ کنار آن (دکمه رادیویی) توپر شود. بر روی next کلیک کنید.

تصویر 1 - 68



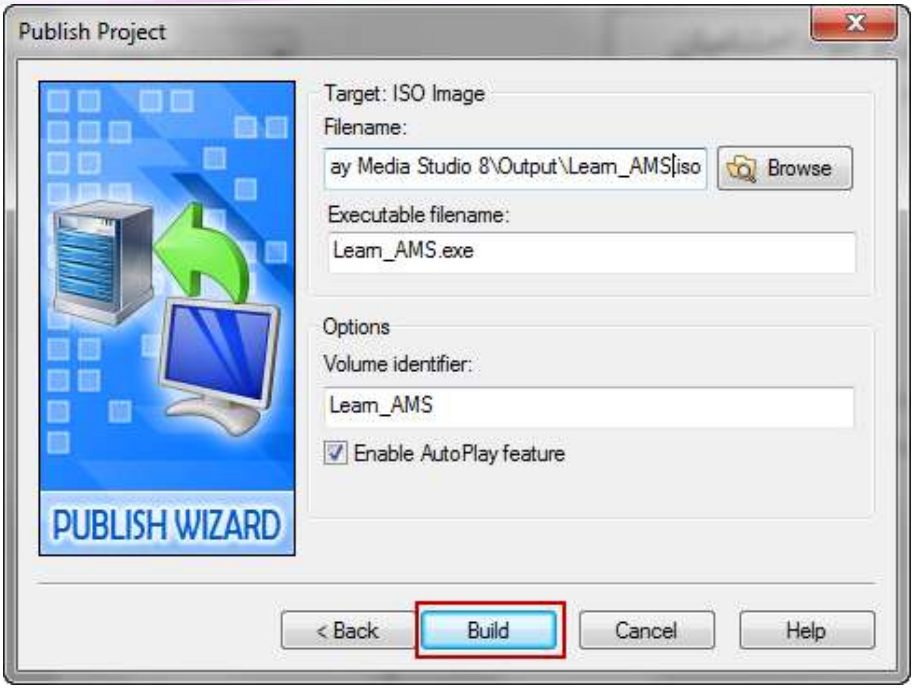
تصویر 1 - 69

یک CD (یا DVD) خام درون درایو CD Writer (یا CD Writer) قرار دهید. در صفحه‌ی جدید در زیر برچسب CD/DVD/Blu-Ray writer Drive یک لیست کرکره ای مشاهده می‌کنید. لیست را باز کرده و درایو رایتر خود را انتخاب کنید. در فیلد Volume identifier نام سی دی خود را وارد نمایید. فیلد executable file name، نام فایل اجرایی (همان فایل Auto run) شما می‌باشد که باید پسوند آن "exe" بماند. Burn را زده و منتظر پایان رایت باشید.

ذخیره بر روي فايل ISO

تصویر 1 - 70

در قسمت بعدی به ضبط فایل‌ها بر روی فایل ISO می‌پردازیم. در اصل فایل ISO یک ایمیج از سی دی شما می‌باشد. یعنی در صورتی که سی دی خام همراه نداشته باشید، می‌توانید آن را به صورت تک فایلی با پسوند ISO درآورید و بعداً با برنامه‌های رایت سی دی (Nero, clone, Alcohol ...) آن را باز کرده و بر روی سی دی رایت کنید. با فشردن دکمه‌ی F7 صفحه‌ی (پنجره) publish project باز می‌شود و بر روی ISO image کلیک کرده و next را بزنید. در قسمت file name مسیر فایل ISO و در قسمت volume identifier نام سی دی را وارد نمایید. بر روی Build کلیک کنید تا فایل در مسیر ثبت شده ذخیره گردد.



تصویر 1 - 71

منتظر باشید تا عملیات ذخیره سازی به پایان رسد. بر روی finish کلیک کنید. با کلیک بر روی close (البته اگر open output folder تیک خورده باشد) شاخه مربوطه باز شده و شما فایل ساخته شده را می‌بینید

نکته: توجه داشته باشید در تمامی مراحل کار با publish project، Enable Autoplay feature را علامت دار کنید. با این کار یک فایل با پسوند ini ساخته می‌شود. در صورتی که این فایل وجود نداشته باشد باید به صورت دستی فایل Autorun اجرا شود. یعنی کاربر سی دی را باز کرده و بر روی Autorun.exe کلیک کند تا Autorun اجرا شود.

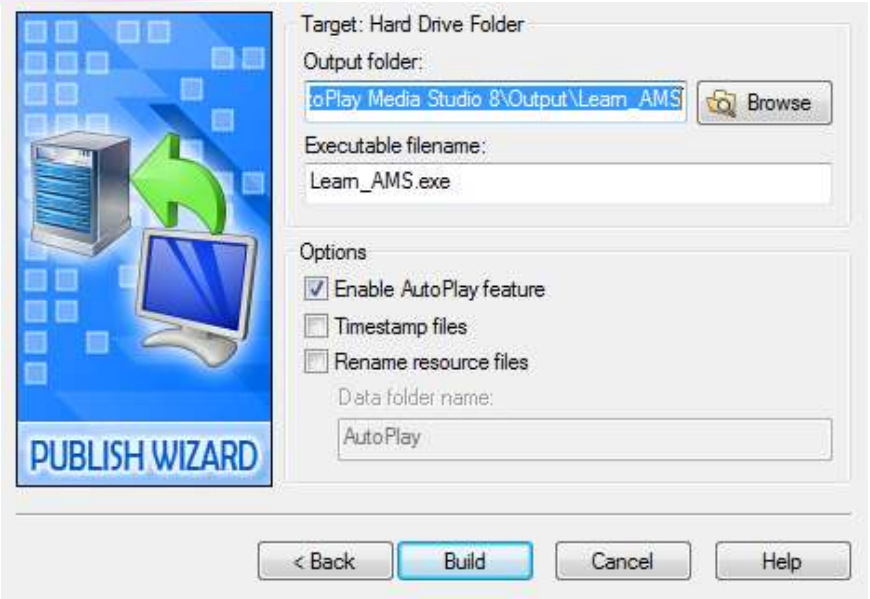
ذخیره روي هارددیسک (دیسک سخت)

تصویر 1 - 72

با کلیک بر روی منوی project و انتخاب publish پنجره‌ی مربوطه را باز کنید. این بار روش ساخت فایل‌های پروژه بر روی هارددیسک را توضیح خواهم داد. که با این کار می‌توانید بعداً آن‌ها را بر روی سی دی رایت کنید.یا با وسایلی مانند حافظه های جانبی آن‌ها را از روی رایانه خارج نمایید.

بر روی Hard drive folder کلیک و next را بزنید.

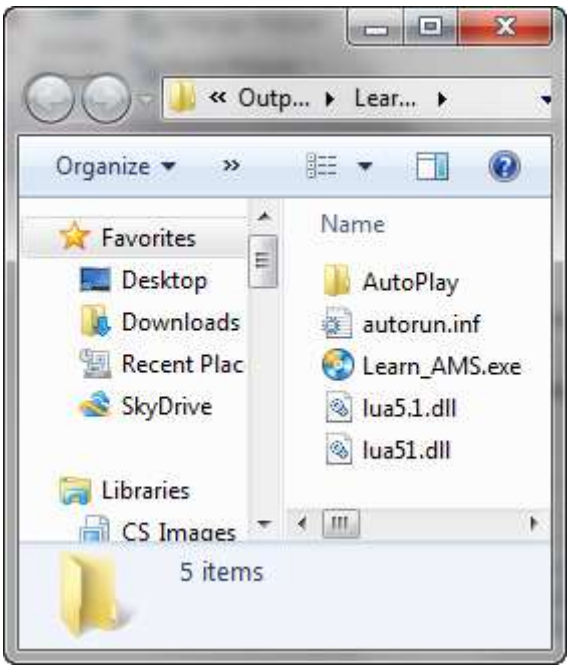
در قسمت output folder شاخه ای را که می‌خواهید فایل‌ها بر روی آن ذخیره شوند را وارد کنید. این شاخه حکم سی دی شما را دارد. یعنی زمانی که قصد رایت فایل‌ها بر روی سی دی دارید، باید محتویات این پوشه را رایت کنید. بر روی Build کلیک کنید و منتظر باشید تا عملیات ذخیره سازی به پایان رسد.



تصویر 1 - 73

بر روی close کلیک کنید. با کلیک بر روی close (البته اگر open output folder تیک خورده باشد) شاخه مربوطه باز شده و شما فایل‌های داخل آن را می‌بینید که این فایل‌ها باید تماماً بر روی سی دی منتقل شوند تا برنامه به طور صحیح کار کنند.

تصویر 1 - 74



تصویر 1 - 75

ساخت فایل فشرده

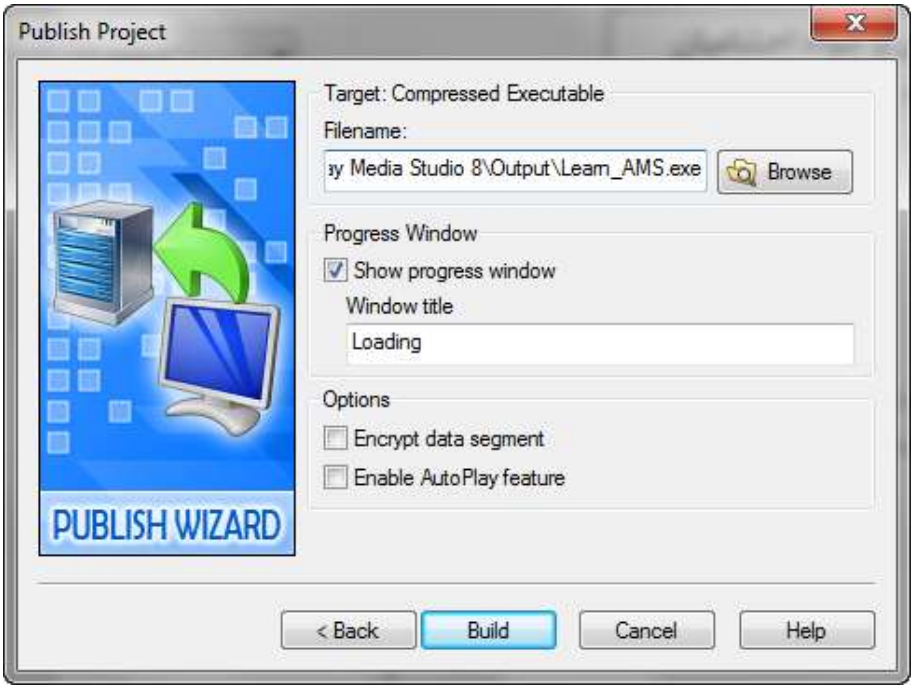
بعضی اوقات مجبورید یک فایل ساخته شده توسط Auto play را به وسیله‌ی E-mail و یا upload بر روی فضاهای خالی در web به شخص دیگری برسانید، برای این کار نمی‌توانید تمام فایل‌ها را تک تک برای شخص بفرستید. Auto play کار شما را آسان کرده، چرا که می‌توانید توسط این برنامه فایلی ایجاد نمایید که پس از باز شدن فایل‌های خود را درون temp ویندوز بارگذاری کرده و از آنجا بخواند و پس از اتمام کار، فایل‌ها پاک شوند و دوباره همان 1 فایل باقی بماند.

پیشنهاد می‌کنم که در صورتی که حجم فایل شما زیاد باشد، از این گزینه استفاده نکنید. چرا که امکان دارد زمان زیادی برای بارگذاری فایل‌ها داخل temp سپری شود و یا حتی درایوی که ویندوز در آن نصب است این ظرفیت را نداشته باشد و کاربر با پیغام خطا روبرو شود.

با فشردن F7، publish project را فراخوانی کنید و Web/Email executable را انتخاب کرده و بر روی next کلیک کنید.

تصویر 1 - 76

در قسمت file name مسیر فایل را انتخاب کنید. Show progress windows برای زمانی است که برنامه در حال بارگذاری (کپی کردن) فایل‌ها در temp می‌باشد. متن درون فیلد windows title به کاربر نمایش داده می‌شود. بر روی Build کلیک کرده و سپس close را بزنید.



تصویر 1 - 77

با کلیک بر روی فایل ساخته شده مشاهده می‌کنید که فایل‌ها با نمایش پیغام تنظیم شده، بارگذاری می‌شوند.

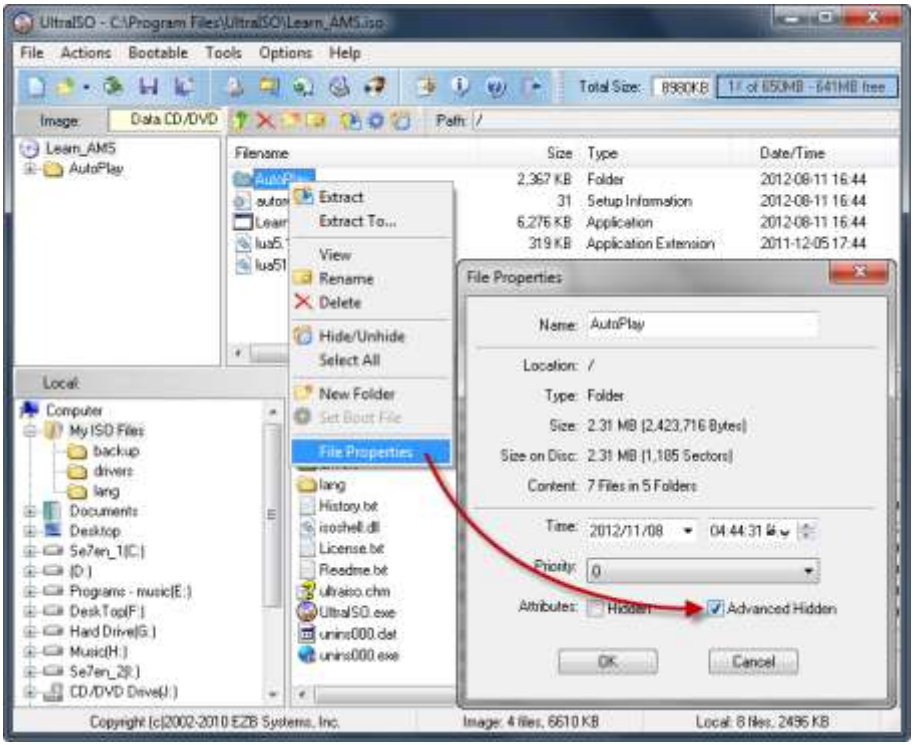
مخفي كردن فايل‌ها در داخل سي دي

بر روی CD همراه نرم افزار UltraISO قرار دارد. این برنامه یکی از برنامه‌های رایت سی دی می‌باشد. می‌خواهیم با این نرم افزار فایل‌های اصلی داخل سی دی را مخفی کنیم. این کار باعث می‌شود زمانی که فایل‌های داخل CD در داخل هارد کپی می‌شود، برنامه اجرا نشود. برای این کار پس از نصب برنامه‌ی UltraISO فایل‌هایی را که توسط قسمت Hard drive

folder ایجاد کرده‌اید را به داخل برنامه بارگذاری کنید. یا فایل iso ساخته شده توسط نرم افزار را با UltraISO باز کنید.

حال بر روی پوشه‌ی AutoPlay کلیک راست کرده و properties را انتخاب کنید. سپس تیک Advance Hide را بزنید و ok را کلیک کنید. سپس سی دی را رایت کنید. پس از رایت سی دی داخل سی دی رام گذاشته و آن را open کنید. مشاهده می‌کنید که شاخه‌ی Auto play در سی دی وجود ندارد.

حال 2 فایل موجود در CD را روی قسمتی از هارد دیسک کپی کنید. سپس آن را اجرا کنید. ملاحظه می‌کنید که پیغام خطایی ظاهر می‌شود.



78 - تصویر 1

فصل دوم: نوشتن برنامه با AMS

تا کنون شما توانستید AMS را اجرا کرده و یک پروژه ساده ایجاد کنید حال در این فصل می‌بایست مبانی نوشتن یک برنامه را بگیرید تا بتوانید با کنار هم قرار دادن کدها برنامه‌های جذابی طراحی کنید. هم چنین با نحوه‌ی استفاده از راهنمای برنامه آشنا خواهید شد تا در مواقع ضروری بتواند به شما در نوشتن یک برنامه کمک کنید.

الگوریتم‌ها

به مجموعه ای از دستورالعمل‌ها و فرمول‌های دقیق که به همراه جزئیات لازم و به صورت مرحله به مرحله به گونه ای اجرا شده باشند که هدف خاصی را دنبال کنند الگوریتم گفته می‌شود.

برای مثال فرض کنید می‌خواهید برنامه ای بنویسید که دو عدد از ورودی دریافت شود و سپس تعیین شود که مجموع دو عدد کوچک‌تر از 10 است یا خیر.

1. شروع .
2. دو عدد a و b را از ورودی در یافت کن.
3. a+b را محاسبه کن.
4. آیا 10> a+b است؟ اگر بلی به مرحله 6 برو.
5. بنویس خیر.
6. به مرحله 7 برو.
7. بنویس بلی.
8. پایان.

نوشتن چنین الگوریتمی معمولاً خیلی ساده به نظر می‌رسد اما برخی از الگوریتم‌ها بسته به برنامه ای که می‌خواهید بنویسید پیچیده تر می‌شوند و ممکن است چند ساعت برای حل مسئله فکر کنید.

آشنایی با لوا، زبان برنامه نویسی در AMS

چون کتاب حاضر در زمینه آموزش AMS نگاشته می‌گردد، فقط به معرفی مختصری درباره‌ی لوا بسنده می‌شود؛ به خاطر داشته باشید که در طول مطالعه‌ی در این کتاب شما کدهایی را می‌آموزید که به زبان لوا می‌باشند و با نحوه‌ی کد نویسی در این زبان تا حد زیادی آشنایی پیدا می‌کنید.

AMS یک نرم افزار چند رسانه‌ای ساز با محیط کد نویسی پیشرفته است اما زبان برنامه نویسی نیست و تحت یک زبان دیگر در آن کد نویسی می‌کنیم که آن زبان لوا (Lua) نام دارد.

لوا یک زبان برنامه نویسی بسیار قوی و ساده است که در سال 1993 توسط استادان دانشگاه PUC-RIO ریودوژانیرو برزیل طراحی و تولید شد که به مرور زمان توسعه پیدا کرده است و تا به امروز به طور گسترده توس شرکت‌های بزرگ برنامه نویسی مورد استفاده قرار گرفته است.

متغیرها

متغیر یعنی هر آنچه که مقداری را در خود ذخیره کند و در طول اجرای برنامه بتوان از آن مقدار استفاده کرد.

در اینجا باید گفت که متغیرها در لوا یعنی زبان برنامه نویسی AMS به نوع خاصی تقسیم بندی نمی‌شوند و فقط مقادیر آن‌ها نوع دارند و دیگر این که یک متغیر در لوا هرگز مقدار ساختار یافته ای به خود نمی‌گیرد و فقط به آن اشاره می‌کند. برای اینکه بهتر متوجه قضیه بشوید به مثال زیر دقت کنید:

```
n = 10;
m = n+12;
n = “Autoplay”;
m = n..“ Media”;
```

در این مثال در سطر اول متغیر n را مساوی 10 قرار داده‌ایم یعنی یک عدد و در سطر دوم متغیر m را مساوی با n+10 قرار داده‌ایم یعنی می‌شود m=22 ؛ در عین حال در سطر سوم n را مساوی ”Autoplay“ قرار داده‌ایم یعنی رشته و در سطر چهارم هم m را که مقدارش

22 شده بود مساوی با رشته ای از ترکیب n و Media قرار داده‌ایم که می‌شود:m=”Autoplay Media”

این مثال به خوبی نشان می‌دهد که متغیر در لوا نوع خاصی ندارد و مقدار آن نوع آن را تعیین می‌کند که در طول اجرای برنامه قابل تغییر است.

نکته: علامت .. برای اتصال دادن یک رشته به رشته ای دیگر به کار می‌رود.

در نام‌گذاری متغیرها به موارد زیر دقت نمایید.

- ✓ نام متغیرها به کوچک و بزرگ بودن حروف حساس است یعنی Test_var با test_Var متفاوت می‌باشد.
- ✓ یک متغیر از حروف، خط زیر (_) و اعداد تشکیل شده است. لازم به ذکر است نام متغیر با عدد شروع نمی‌شود.
- ✓ از خط تیره (-) برای نام‌گذاری استفاده نکنید.
- ✓ سعی کنید نام شی به کار انجام دهنده توسط آن نزدیک باشد .
- ✓ از ایجاد متغیر با نام‌های یکسان در قسمت‌های مختلف نرم افزار خودداری کنید (احتمال تداخل داده‌ها وجود دارد).

چند نمونه از نام‌های غیر مجاز :

1Value , %name ,G+L ,Test Name

همچنین چندین نام برای متغیرهای پیش فرض رزرو شده که از آن‌ها نیز نمی‌توانید استفاده کنید و مانند :

And , do , else , table , in , nil , not و ...

انواع داده‌ها

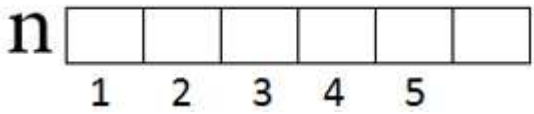
حال که با متغیر آشنا شدید بهتر است با انواع داده‌ها نیز آشنا شوید.

وقتی که متغیری را تعریف می‌کنید باید بدانید چه اطلاعاتی را در آن ذخیره می‌کنید و آن‌ها چه کاربردی در برنامه دارند.

1. اعداد: شامل هر نوع عددی است که در متغیر ذخیره می‌گردد؛ چه عدد اعشاری و چه عدد صحیح. مثل: n = 12.5;
2. رشته‌ها: شامل هر عبارتی که مابین علامت "" و یا '' قرار می‌گیرد و هم چنین با علامت .. به یکدیگر متصل می‌گرداند. مثل:

n="Computer"; یا N = n.." name";

3. داده های بولین: شامل هر نوع داده ای که دارای دو مقدار صحیح(true) و غلط(false) می‌باشد. مثل: n = false; یا n= true;
4. آرایه‌ها: هر چند آرایه‌ها برای خود بحثی جداگانه‌اند اما در لوا یک متغیر را می‌توان مساوی مجموعه ای از اعداد یا رشته قرار داد که به آن ارایه می‌گویند. برای مثال: n= {1,2,3,4,5}; برای اینکه بتوانیم به یک مقدار از یک آرایه دسترسی داشته باشیم باید شماره آن را بنویسم. برای مثال: m=n[1] یعنی m مساوی با اولین مقدار از آرایه‌ی n

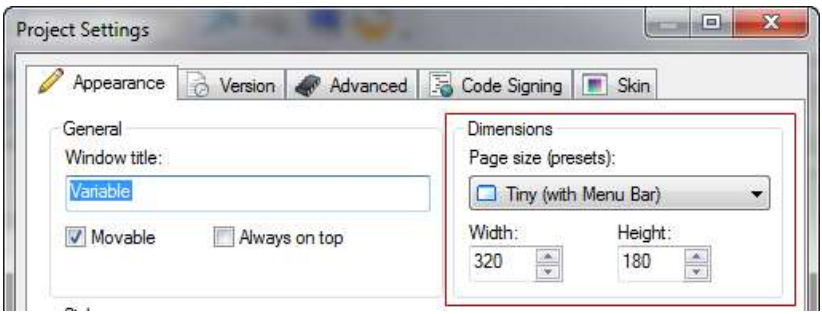


تصویر 2 - 1

برای مثال پروژه ای می‌نویسیم که بتواند یک متن را به عنوان ورودی دریافت کند و آن را در یک متغیر بریزد و سپس آن متغیر را به متغیری دیگر متصل کند و در یک کادر پیغام نمایش دهد.

1. پروژه ای با نام variable ایجاد کنید .
2. از منوی Project>setting و از قسمت Dimensions اندازه از منوی کشویی پروژه را برابر با Tiny انتخاب کنید. هم چنین می‌توانید اندازه دلخواه را در همان قسمت در ورودی Width (پهنا) و Height (درازا) وارد نمایید و برای اعمال اندازه Ok را کلیک کنید.(تصویر 2- 1)

3. از نوار ابزار Objects یک Input و دو Button به پروژه اضافه کنید.
4. Button1 را با کلیک روی آن انتخاب کنید و در پنجره‌ی Properties خاصیت Text را برابر OK قرار دهید. هم چنین خاصیت Text را برای Button2 برابر با Close قرار دهید.



تصویر 2 - 2

5. در این مرحله نوبت به کد نویسی می‌رسد :

روی Button1 کلیک کنید و در پنجره Properties بر روی دکمه‌ی ... مقابل On click کلیک کنید سپس در اینجا روی دکمه‌ی Add Action کلیک کنید و از منوی کشویی step1گزینه‌ی Input را انتخاب نمایید و از لیست step2 گزینه‌ی Input.GetText را انتخاب کرده و Next را کلیک کنید و در قسمت Result Variable کلمه‌ی result پاک کنید و به جای آن MyName را تایپ کنید. در آخر روی Ok کلیک کنید.

سطر دوم کد را به شکل زیر بنویسید:

```
Dialog.Message("My Name", "My Name is: "..MyName);
```

حال روی دکمه‌ی Ok کلیک کنید.

Button2 را انتخاب کرده و از پنجره‌ی Properties به محیط کد نویسی On click آن بروید و کد زیر را در آن تایپ کنید و سپس با کلیک روی Ok کد را تایید کنید.

```
Application.Exit(0);
```

برنامه را اجرا کنید و نام خود را در قسمت Input یا همان ورودی متن وارد کنید و سپس دکمه‌ی Ok را کلیک کنید تا نتیجه‌ی کار خود را ببینید.

نتیجه کار باید شبیه به شکل زیر باشد:

تصویر 2 - 3

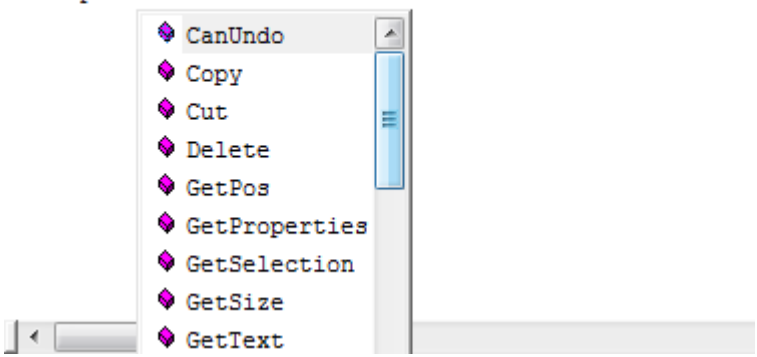
و با کلیک روی Close می‌توانید.

با توجه به این مثال ما می‌توانیم مقدار متغیرها را برخی اشیای موجود در پروژه نیز دریافت کنیم. همان طور که ما در این مثال مقدار متغیر MyName را از ورودی Input1 گرفتم و به متن My Name is: متصل کردیم.

نکات مهم و قابل توجه :

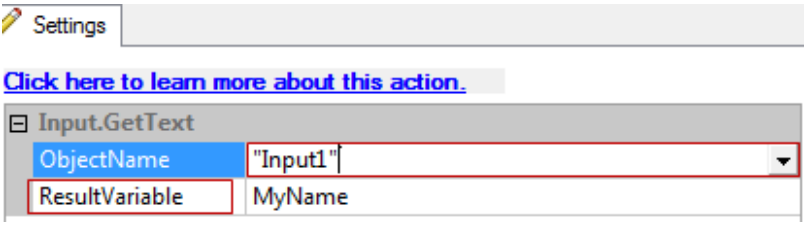
- با کلیک روی هر دکمه‌ی می‌توان آن را انتخاب کرد و در پنجره‌ی Properties در قسمت Name نام آن را مشاهده کرد.
- با نوشتن نام شی و گذاشتن یک نقطه جلوی آن در قسمت کد نویسی و فشردن کلیدهای Ctrl+Space می‌توان تمامی کدهای مربوط به آن شی را مشاهده کرد و با کلیک روی هرکدام می‌توان از کدها استفاده نمود.(تصویر 2- 3)

```
01 MyName = Input.GetText("Input1");
02 Dialog.Message("My Name", "My Name is: "..MyName);
03 Input.
```



تصویر 2 - 4

- می‌توان روی کد انتخاب شده به روش بالا دو بار کلیک کرد تا Wizard آن را مشاهده نمود و تغییراتی در آن اعمال نمود.(تصویر 2-4)



تصویر 2 - 5

- به قسمت‌هایی که در wizard مربوط به کد نویسی همانند تصویر 2-4 دارای یک علامت بازشو رو به پایین می‌باشد اکثراً قابل تغییر می‌باشند برای مثال اگر ما در پروژه variable دو تا Input داشتیم می‌توانستیم از قسمت ObjectName هر کدام را که نیاز داشتیم انتخاب کنیم.
- قسمت Result Variable را چنان چه برای هر کدی در Wizardدیدید بدانید که آن شی یک مقداری را برمی‌گرداند که می‌توانید از آن استفاده نمایید. همانند پروژه variable که ما از این کار را برای شی Input1 انجام دادیم.

توضیحات و فضای خالی در کد نویسی

توضیحات: زمانی که یک برنامه نویس می‌خواهد برنامه بنویسد که بعد از گذشت یک سال به راحتی بتواند در سورس برنامه تغییراتی ایجاد کنید می‌بایست کدهایش دقیق و همراه با توضیحات باشد تا ویرایش این کدها برایش وقت گیر نباشد.

در AMS برای مشخص کردن و نوشتن توضیحات از دو خط تیره (--) استفاده می‌شود و متن بعد از این علامت به رنگ سبز در آید و از کدها متمایز می‌گردد. (تصویر 2- 5)

برای مثال شما در کد زیر ابتدا یک توضیح می‌آوریم:

```
--tarife motaghayer
MyName = Input.GetText("Input1");
--nemayeshe matn dar dialog
Dialog.Message("My Name", "My Name is: "..MyName);
```

در صورتی که نیاز به توضیح چند خط داشته باشید، لازم به قرار دادن – در ابتدای هر سطر ندارید.

کافی است به صورت زیر عمل نمایید:

```
--[[
Write By :
Hamed Heydari
AND
Javad Ahshamian
--]]
```

فضای خالی: زمانی که کد نویسی را انجام می‌دهید باید به خوانا بودن کد نیز توجه داشته باشید تا در آینده در بررسی پروژه‌ی خود دچار زحمت و سردرگمی نشوید. با ایجاد فضای خالی در برخی از کدها خوانا بودن کدها را تأمین می‌کنیم برای مثال: به کدهای زیر دقت کنید:

```
n = 10;
m = 12;
```

```
if m > n then
        if m > 13 then
        Dialog.Message("Notice","m= "..m);
        end
Dialog.Message("Notice","m> n");
end
```

در فصل بعد شما با دستور if آشنا خواهید شد پس زیاد نگران درک آن نباشید و به فضای خالی ایجاد شده برای خوانا شدن کد دقت کنید؛ همان طور که می‌بینید در این کد از if دو بار استفاده کرده‌ایم اما برای اینکه آن‌ها را با هم اشتباه نگیریم و محل شروع و پایان هر کدام را بدانیم if دوم و کدهای مربوط به آن را با فاصله یک Tab یکی هشت کاراکتر فاصله نوشته‌ایم. این فضای خالی باعث شده است که به راحتی بتوان کاربرد کد مورد نظر را فهمید. (تصویر 2-5)

به خاطر داشته باشید که این موارد جزو اصول نوشتن یک برنامه‌ی استاندارد هستند.

```
01 --tarife motaghayer
02 MyName = Input.GetText("Input1");
03 Dialog.Message("My Name", "My Name is: "..MyName);
04 -----Fazaye khali
05 n = 10;
06 m = 12;
07 if m > n then
08     if m > 11 then
09     Dialog.Message("Notice","m= "..m);
10     end
11 end
```

تصویر 2 - 6

چگونگی استفاده از راهنمای برنامه

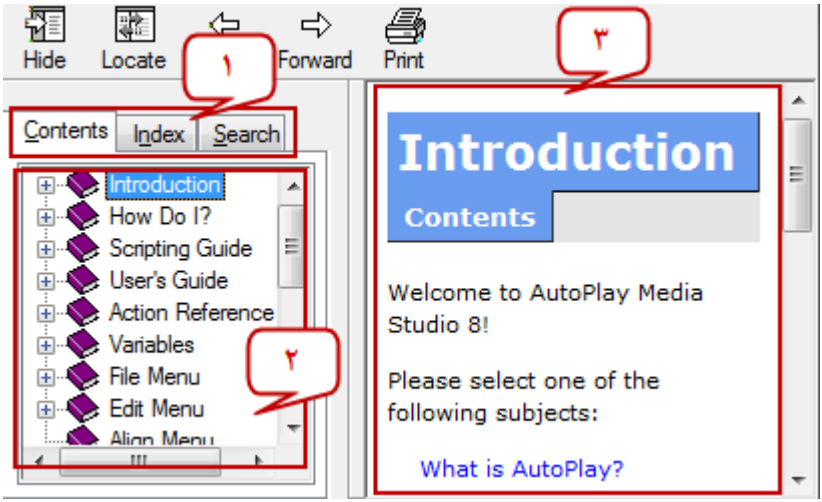
در محیط AMS چندین نوع راهنما وجود دارد که نوشتن پروژه و کد نویسی را برایمان آسان می‌سازد.

راهنمای برنامه

برای مشاهده راهنمای برنامه ابتدا یک پروژه را ایجاد یا باز کنید سپس از منوی Help گزینه Autoplay Media Studio Help را کلیک کنید تا راهنمای برنامه باز شود.(تصویر 1-15)

این راهنما شامل توضیحات و مثال‌های کاملی در مورد کار با برنامه به زبان انگلیسی می‌باشد. اما چنانچه زبان انگلیسی شما قوی نیست جای نگرانی ندارد چون با کمی تمرین متوجه خواهید شد که فهم عبارت‌های این راهنما بسیار آسان است.

همان طور که در تصویر 1- 15 مشاهده می‌کنید این راهنما به 3 قسمت تقسیم شده است: قسمت 1 شامل سه تب(سربرگ) می‌باشد که **تب Contents** شامل تقسیم بندی موضوعی مطالب موجود در راهنمای برنامه می‌باشد؛ **تب Index** شامل تقسیم بندی مطالب بر اساس کلمات کلیدی می‌باشد که راهنمای بسیار مناسبی است؛ **تب Search** نیز مخصوص جستجوی کلمات در راهنمای برنامه می‌باشد.

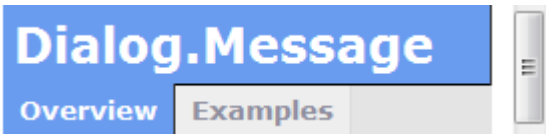


تصویر 2 - 7

استفاده از راهنما در هنگام افزودن کد به وسیله Wizard

اگر در حال افزودن یک کد به برنامه هستید که نمی‌دانید کاربرد آن چیست می‌توانید در همان جا از راهنمای برنامه استفاده کرده و کاربرد آن را بدانید برای این کار کافی است بر روی

دکمه‌ی help کلیک کنید (تصویر 1- 13) تا راهنمای دستور مورد نظر را مشاهده نمایید. این قسمت از راهنما شامل دو بخش می‌شود که با کلیک روی هر کدام مطالب مجزای آن بخش را مشاهده می‌کنید: 1. Overview که در برگیرنده توضیحاتی در مورد قسمت‌های مختلف کد مورد نظر می‌باشد. 2. Example که دربردارنده‌ی مثال یا مثال‌هایی در رابطه با کد مورد نظر می‌باشد. (تصویر 1- 16)



تصویر 2 - 8

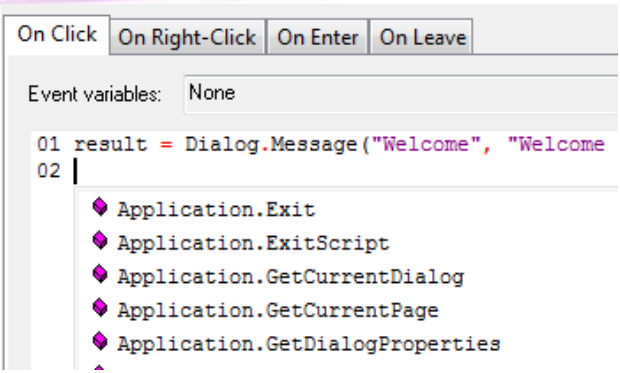
استفاده از راهنما در هنگام کد نویسی دستی

زمانی در حال تایپ کد بودید و نیاز به استفاده از راهنما را احساس کردید می‌بایست بر روی کد مورد نظر کلیک کنید و سپس در پایین کادر بر روی دکمه‌ی علامت سؤال کلیک کنید تا راهنمای آن کد نمایش داده شود. (تصویر 1- 17)

تصویر 2 - 9

مشاهده‌ی کدها

هنگامی که در محیط کد نویسی برنامه هستید برای اینکه بتوانید تمام کدها را مشاهده کنید از کلیدهای Ctrl+Space (کلید Ctrl را پایین نگه داشته و سپس کلید خط فاصله را بفشارید) را فشار دهید.(تصویر 1- 18)



تصویر 2 - 10

از دیگر راهنماها و مراجع موجود برای این نرم افزار به موارد زیر می‌توان اشاره کرد :

1. سایت اصلی نرم افزار و انجمن آن به آدرس زیر:

http://www.indigorose.com

2. سایت لوا :

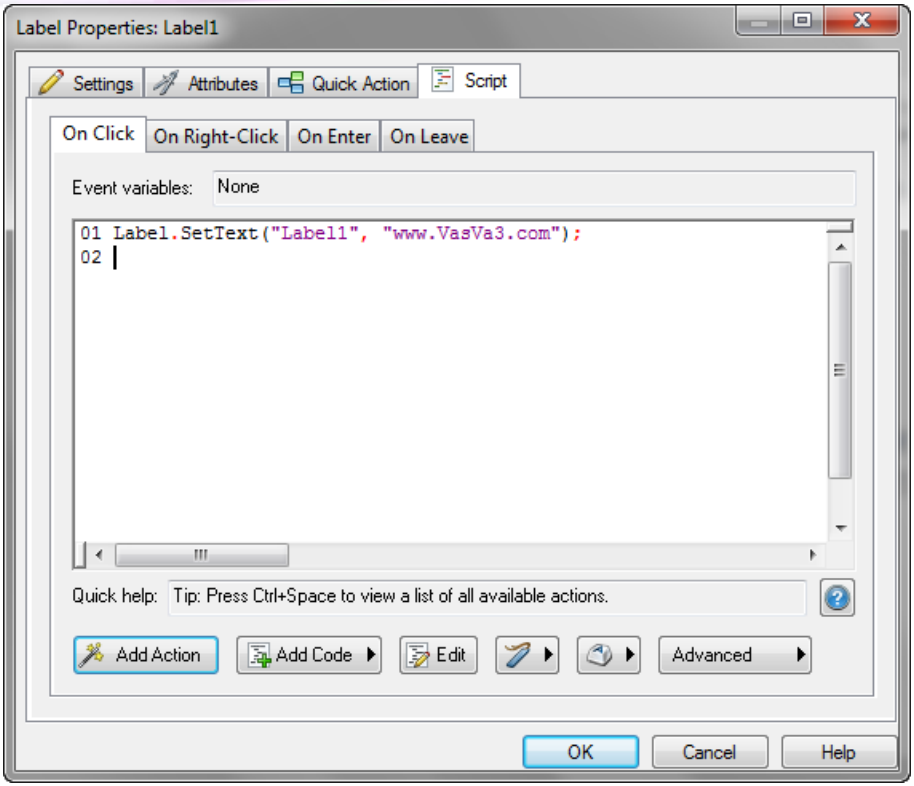
http://www.lua.org

3.انجمن تخصصی AMS به زبان پارسی :

http://www.AutoPlay.IR
http://www.vasva3.com

سربرگ کد نویسی(Script Tab)

یکی از مهم‌ترین سربرگ‌هایی که در حین نوشتن یک برنامه‌ی پیش رفته به کار خواهد آمد سربرگ کد نویسی می‌باشد. شما با استفاده از این سربرگ می‌توانید از به نوشتن، ویرایش و ذخیره‌ی کدهای مورد نیاز خود بپردازید. در تصویر زیر نمایی از این سربرگ را مشاهده می‌نمایید:



تصویر 2 - 11

همان طور که در تصویر مشاهده می‌کنید این سربرگ یک از قسمت‌های پنجره‌ی Properties می‌باشد که با دو بار کلیک کردن روی یک شی و یا قسمت خالی صفحه به نمایش در می‌آید.

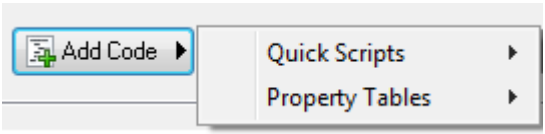
این سربرگ دارای چندین قسمت می‌باشد که هر کدام از آن‌ها برای انجام کار خاصی طراحی شده‌اند از جمله: دکمه‌ی Help که با آن آشنا شدید. در قسمت شما با کاربرد برخی از این قسمت‌ها آشنا خواهید شد و برخی دیگر از آن‌ها را به طور مفصل در فصل‌های بعدی مطالعه خواهید نمود.

دکمه های موجود در سربرگ کد نویسی

1. دکمه‌ی Add Action: از این دکمه برای افزودن دستورات موجود در AMS استفاده می‌شود که در این فصل با آن آشنا شدید.

2. دکمه‌ی Add Code: این دکمه برای افزودن کدهای از پیش نوشته شده مورد استفاده قرار می‌گیرد که خود دارای یک منوی فرعی دو گزینه ای با نام‌های Quick Script و Property Table می‌باشد که هر کدام از آن‌ها نیز دارای زیر منوی خاص خود می‌باشند:

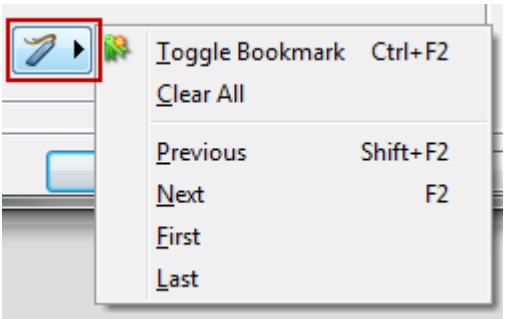
- ✓ Quick Script: این منو برای افزودن کدهایی همانند حلقه‌ها(for,while,...)، شرط(if)، خطا گیری و ... به کار می‌رود که برای کنترل روند اجرای برنامه مورد استفاده قرار می‌گیرند.
- ✓ Property Table: از این منو برای افزودن کدهای از پیش تعریف شده ای استفاده می‌گردد که توسط آن‌ها خصوصیاتی را به اشیا یا صفحه‌ها اختصاص می‌دهیم.



تصویر 2 - 12

3. دکمه‌ی Edit: چنان چه شما بر روی یک دستور کلیک کنید و سپس دکمه‌ی Edit را کلیک کنید پنجره‌ی خصوصیات (پارامترها) آن کد ظاهر خواهد شد که می‌توانید به صورت سریع کد را ویرایش کنید.

دکمه‌ی Bookmarks: این دکمه برای علامت گذاری کد انتخاب شده و مدیریت آن‌ها به کار می‌رود. این دکمه‌ی دارای یک منوی فرعی پنج گزینه ای می‌باشد: گزینه‌ی Toggle Bookmark برای علامت گذاری کردن کد، Clear All برای پاک کردن نشانه های علامت گذاری شده، Previous جهت رفتن به کد نشانه گذاری شده‌ی قبلی، Next برای رفتن به کد نشانه گذاری شده‌ی بعدی، First برای رفتن به اولین کد نشانه گذاری شده‌ی موجود در سربرگ کد نویسی و گزینه‌ی Last برای رفتن به آخرین کد علامت گذاری شده در سربرگ کد نویسی به کار می‌رود.



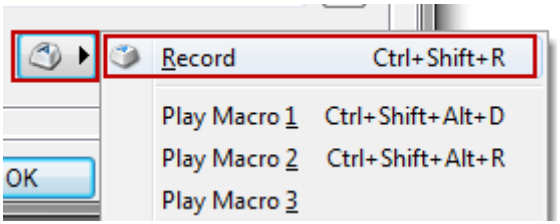
تصویر 2 - 13

در تصویر زیر نمونه ای از کد علامت گذاری شده را مشاهده می‌کنید:

```
01 Label.SetText("Label1", "www.VasVa3.com");
02
```

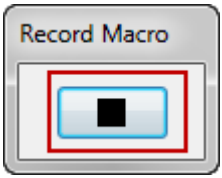
تصویر 2 - 14

دکمه‌ی Macros: این دکمه جهت ضبط کردن فعالیت‌های شما در محیط کد نویسی یا همان سربرگ کد نویسی به کار می‌رود که دارای یک منوی فرعی می‌باشد که در آن می‌توانید با کلیک روی گزینه‌ی Record شروع فعالیت‌های خود را ضبط کنید:



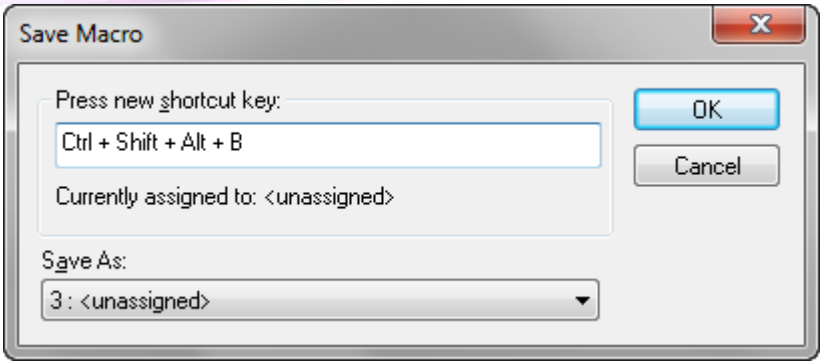
تصویر 2 - 15

و سپس در انتها کلیک روی دکمه‌ی Stop ظاهر شده عمل ضبط کردن را متوقف نمایید



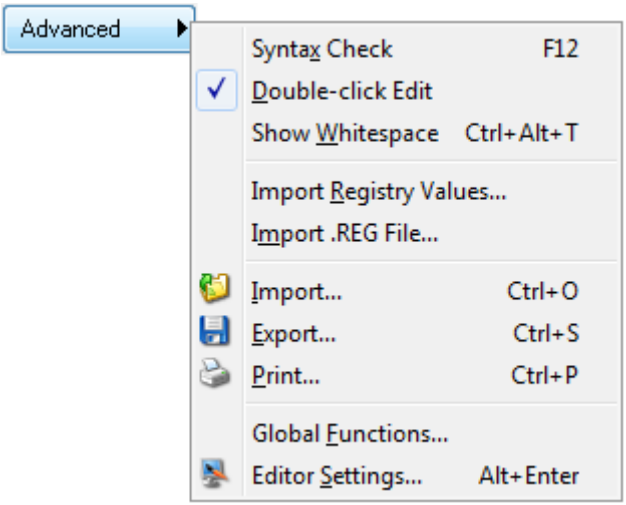
تصویر 2 - 16

و در پنجره‌ی ظاهر شده یک کلید میانبر برای انجام فعالیت مورد نظر تعریف کنید و در آخر روی دکمه‌ی Ok کلید نمایید و یا کلیک روی Cancel از انجام این کار منصرف شوید.



تصویر 2 - 17

4. Advanced: این دکمه نیز شامل گزینه‌هایی به شرح زیر می‌باشد:



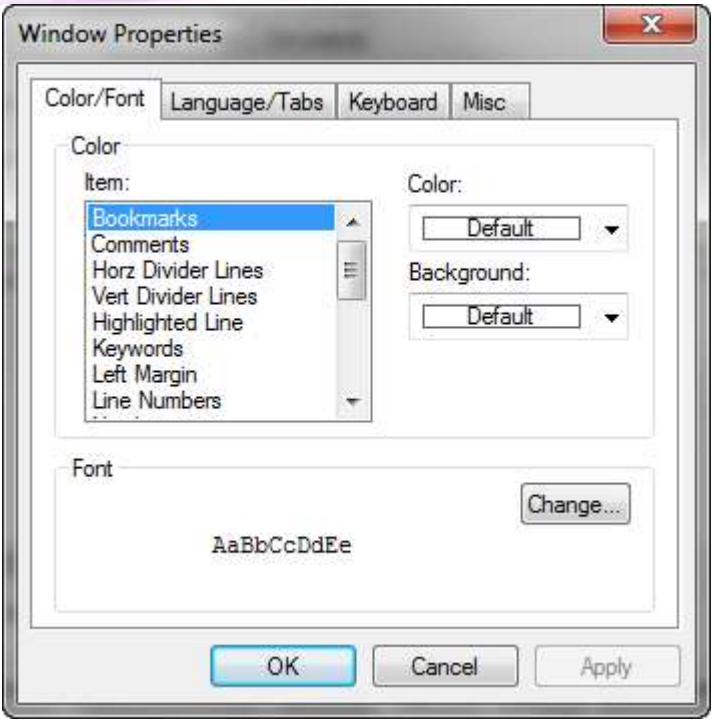
تصویر 2 - 18

- ✓ Syntax Checker: این گزینه برای بررسی صحیح نوشته شدن کدها به کار می‌رود که در فصل‌های بعدی بیشتر با کاربرد آن آشنا خواهید شد.
- ✓ Double-click Edit: اگر این گزینه فعال باشد می‌توانید با دو بار کلیک کردن روی دستورها پنجره‌ی خصوصیات آن‌ها را مشاهده کنید.
- ✓ Show Whitespace: با فعال کردن این گزینه می‌توانید فضاهای خالی بین کدها را مشخص‌تر کنید.

- ✓ Import Registry Values: این گزینه برای وارد کردن مقدار خاصی از رجیستری ویندوز به کار می‌رود و آن مقدار را در دستور Registry.SetValue قرار می‌دهد تا کاربر به صورت دستی اقدام به نوشتن این دستور و مقادیر ان ننماید.
- ✓ Import .REG Files: از این گزینه برای وارد کردن محتویات یک فایل رجیستری ذخیره شده به برنامه استفاده می‌کنیم و برنامه به طور خودکار آن را در دستور Registry.SetValue قرار می‌دهد.

توجه: در فصل‌های بعدی با دستورات مربوط به رجیستری آشنا خواهید شد.

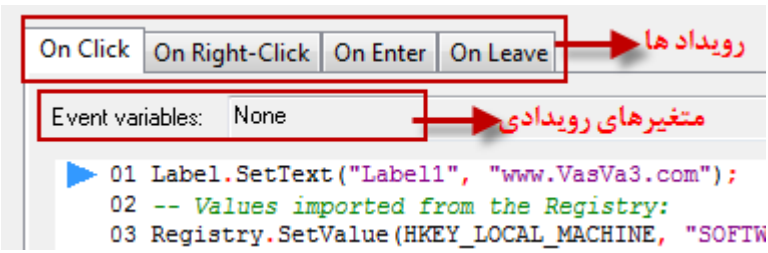
- ✓ Import: این گزینه برای وارد نمودن فایلی با پسوند Lua به کار می‌رود که حاوی کدهایی به زبان لوا باشد و محتویات آن فایل در سربرگ کد نویسی قرار می‌گیرد.
- ✓ Export: از این گزینه برای ذخیره سازی کدهای نوشته شده در سربرگ کد نویسی استفاده می‌کنیم
- ✓ Print: این گزینه برای چاپ کدهای نوشته شده در سربرگ کد نویسی به کار می‌رود.
- ✓ Global Functions: با استفاده از این گزینه می‌توان به پنجره کد نویسی Global Functions یعنی کدهایی که در سراسر برنامه قابل دسترس هستند، دست یافت.
- ✓ Editor Settings: با استفاده از این گزینه می‌توانید تنظیمات سربرگ کد نویسی را تغییر دهید: از جمله رنگ‌ها و قلم‌ها، تنظیمات مربوط به زبان برنامه نویسی و عملکرد کلید Tab، تنظیمات مربوط به صفحه کلید و کلیدهای میانبر و تنظیمات متفرقه.



تصویر 2 - 19

با سایر دکمه های این قسمت همانند: Help(راهنما)، Ok(تایید کدهای نوشته شده)، Cancel(منصرف شدن) نیز آشنا شده‌اید و نیازی به ذکر مجدد آن‌ها نیست.

اما چند قسمت دیگر نیز در این پنجره قابل مشاهده می‌باشد که عبارتند از رویدادها و متغیرهای رویدادی که هر دو مورد در فصل‌های بعدی به تفصیل بیان شده‌اند.



تصویر 2 - 20

فصل سوم: کنترل روند اجرای برنامه

در فصل دوم در مورد الگوریتم‌ها و کاربرد آن‌ها در برنامه مطالبی را آموختید. در این فصل نحوه‌ی کنترل روند اجرای در طول این الگوریتم‌ها را مشاهده خواهید کرد. برای مثال خواهید دید که چگونه می‌توانید تصمیماتی از قبیل "اگر X مساوی این بود، A را انجام بده در غیر این صورت B را انجام بده" را در برنامه خود پیاده سازی کنید. هم چنین مشاهده خواهید کرد که چگونه می‌توانید یک قطعه کد را به دفعات مشخص و یا تا زمانی که یک شرط درست است اجرا کنید.

تصمیم گیری در برنامه

الگوریتم‌ها همواره دارای تصمیماتی هستند. در واقع، این تصمیمات هستند که باعث می‌شوند کامپیوتر بتواند وظیفه‌ی خود را به خوبی انجام دهد. هنگام کد نویسی با تصمیم گیری‌های زیادی مواجه خواهید شد. مثلاً فرض کنید می‌خواهید فایلی را باز کنید باید کدی بنویسید که "آیا فایل مورد نظر وجود دارد؟". در صورت وجود فایل را باز کند و در غیر این صورت الگوریتم به پایان برسد.

تمام این تصمیم گیری‌ها به یک نحو در برنامه پیاده سازی می‌شوند. در ابتدا به بررسی دستور if برای کنترل روند اجرای برنامه می‌پردازیم.

دستور if:

در AMS برای تصمیم گیری از دستور if استفاده می‌شود.

شکل کلی دستور:

```
If ( شرط مورد نظر ) then
دستوراتی که باید اجرا شوند
end
```

در مثال عملی زیر با نحوه‌ی کاربرد این دستور آشنا خواهید شد:

مثال: یک دستور if ساده

1. یک پروژه به نام Simple if ایجاد کنید. سپس با استفاده از نوار ابزار Standard یک شی Button به پروژه اضافه کنید و آن را در وسط صفحه قرار دهید. و خاصیت Text آن را در پنجره‌ی Properties برابر با if قرار دهید. سپس روی شی Button دو بار کلیک کنید و به سربرگ Script بروید و سپس در قسمت On Click کدهای زیر را بنویسید :

```
Number = 20;
if (Number == 20) then
Dialog.Message("Sample if","Number is indeed 20!");
end
```

و در انتها روی دکمه‌ی Ok کلیک کنید.

2. برنامه را اجرا با کلید F5 اجرا کنید و بر روی دکمه‌ی if کلیک کنید. کادر پیغامی را مشابه تصویر 3- 1 خواهید دید.

در این مثال ابتدا یک متغیر به نام Number ایجاد کرده و مقدار آن را برابر با 20 قرار می‌دهید.(تعریف متغیر)

تصویر 3 - 1

سپس با استفاده از دستور if مشخص می‌کنید که باید چه کاری انجام دهید. در این مثال شما می‌گویید: اگر Number برابر با 20 بود کادر پیغام بالا را نمایش دهد.

نکته: قطعه کدی که درون دستور if قرار دارد تنها زمانی اجرا می‌گردد که Number برابر با 20 باشد. به عبارتی دیگر هنگامی که شرط مورد نظر صحیح باشد، کد درون دستور if اجرا

می‌گردد. هم چنین نوشتن شرط درون پرانتز ضرورتی ندارد و می‌توان بدون آن هم اقدام به نوشتن شرط نمود.

اما اگر نتیجه یک شرط نادرست باشد، چه اتفاقی می‌افتد؟ در ادامه به بررسی این حالت خواهیم پرداخت.

مثال: نادرست بودن شرط

1. اگر Simple if در حال اجرا است آن را ببندید. شی Button دیگری به فرم اضافه کنید، خاصیت Text آن را برابر با Another if قرار دهید. روی دکمه دو بار کلیک کنید و کد زیر را قسمت Script > On Click بنویسید:

```
Number = 20;
if (Number == 33) then
Dialog.Message("Sample if","Number is indeed 20!");
end
```

در این حالت هم مشاهده می‌کنید، چون جواب سؤال "آیا Number برابر با 33 است؟" خیر است و دستورات درون if هم فقط در حالتی اجرا می‌شود که نتیجه شرط درست باشد، در نتیجه کدهای درون if اجرا نخواهد شد. در چنین شرایطی کنترل برنامه بلافاصله به خط بعد از پایان if منتقل می‌شود و کد مربوط به آن را در صورت وجود اجرا می‌کند.

دستور else

این دستور زمانی استفاده می‌شود که بخواهید برنامه هم در صورت درست بودن شرط کاری را انجام دهد و هم در صورت نادرست بودن شرط کاری را انجام دهد.

مثال: دستور else

1. در پروژه Simple if، کد درون رویداد On Click مربوط به شی Button2 (Another if)را به صورت زیر تغییر دهید:

```
Number = 20;
if (Number == 33) then
Dialog.Message("Sample if","Number is indeed 20!");
else
Dialog.Message("Sample if","Number is not 33!");
```

```
end
```

2. برنامه را اجرا کنید و روی دکمه‌ی Another if کلیک کنید. کادر پیغامی مشابه تصویر 3-2 را مشاهده خواهید کرد.

کدی که در قسمت else وارد شده است، فقط زمانی اجرا می‌شود که شرط نادرست باشد. در این حالت مقدار Number برابر با 20 است، اما چون در شرط با عدد 1000 مقایسه شده است بنابراین شرط غلط است و کد نوشته شده در بخش else اجرا خواهد شد.

تصویر 3 - 2

بررسی چند شرط با دستور else if :

اگر می‌خواهید بیش از یک حالت را مورد بررسی قرار دهید باید از ترکیب دستور else و if استفاده کنید. در مثال بعدی برنامه Simple if را به نحوی تغیر می‌دهیم که برابری Numberرا با چند عدد مختلف بررسی کند و نتیجه را نمایش دهد.

مثال: دستور else if

1. در پروژه Simple if، کد درون رویداد On Click مربوط به شی Button2 (Another if)را به صورت زیر تغییر دهید:

```
Number = 27;
if (Number == 1000) then
Dialog.Message("Sample if","Number is indeed 1000!");
else if (Number == 27) then
Dialog.Message("Sample if","Number is 27!");
```

```
else
Dialog.Message("Sample if","Number is neither 1000 or 27!");
end
end
```

2. برنامه را اجرا کنید و روی دکمه‌ی Another if کلیک کنید. کادر پیغامی مشابه تصویر 3- 3 را خواهید دید.

تصویر 3 - 3

در اینجا دستورات بخش else if اجرا می‌شوند، زیرا Number برابر با عدد 27 است و بنابراین شرط داخل else if درست خواهد بود. توجه داشته باشید که اگر شرط داخل else if نیز غلط می‌بود، کد های بخش else اجرا می‌شدند.

در یک سری از دستورات if و else if متوالی، شرط‌ها از بالاترین if به سمت پایین بررسی می‌شوند و اولین عبارتی که درست ارزیابی شد، دستورات مربوط به آن اجرا می‌شوند. پس در برنامه‌ی قبل اگر شرط اول را به گونه ای تنظیم کنیم که درست باشد (برای مثال داخل پرانتز عبارت Number > 10 را قرار دهیم که به علت بزرگ‌تر بودن Number از 10 شرط ما درست از آب در می‌آید)، با وجود اینکه شرط دوم هم درست است دستورات شرط اول اجرا می‌شوند و کنترل برنامه به اولین خط بعد از سری دستورات if می‌رود.

```
else if (Number == 27) then
```

```
Dialog.Message("Sample if","Number is 27!");
else
Dialog.Message("Sample if","Number is neither 1000 or 27!");
end
```

شما می‌توانید به هر تعداد که بخواهید قسمت‌های else if را به یک دستور if برای بررسی حالت‌های مختلف اضافه کنید. اما همان طور که ذکر شد، هنگامی که AMS به اولین دستور if رسید شرط داخل آن را بررسی می‌کند؛ اگر عبارت داخل پرانتز درست ارزیابی شود دستورات درون if اجرا می‌شوند و کنترل برنامه به اولین خط بعد از سری دستورات if و else می‌رود. در غیر این صورت، عبارت مربوط به اولین else if ارزیابی می‌شود. این روند ادامه پیدا می‌کند تا برنامه به قسمتی از دستورات برسد که حاصل آن درست باشد. در این حالت دستورات این قسمت اجرا شده و کنترل برنامه به بعد از دستورات if و else می‌رود.

نکته هنگام بررسی یک سری از حالت‌ها، بهتر است آن‌هایی را که احتمال درست بودنشان بیشتر است، ابتدا بررسی کنید. این مورد باعث می‌شود برنامه هنگام اجرا، شرایط اضافی را بررسی نکند و کد سریع‌تر اجرا شود.

دستورات if تو در تو:

علاوه بر استفاده‌ی متوالی از دستورات if، می‌توانید در داخل یک if از دستورات if دیگری استفاده کنید:

```
n = 10;
m = 13;

if (m > n) then
        if (m > 13) then
        Dialog.Message("Notice","m= "..m);
        end
Dialog.Message("Notice","m> n");
end
```

در استفاده از دستورات if تو در تو هیچ محدودیتی نیست . البته باید دقت کنید که هر چه تعداد if های تو در تو در برنامه بیشتر باشد، درک آن مشکل‌تر می‌شود. بنابراین سعی کنید تا جایی که می‌توانید تعداد if های تو در تو را در برنامه کم کنید.

عملگر های مقایسه ای:

در قسمت‌های قبلی، نحوه‌ی بررسی برابر بودن یک متغیر را با مقادیر مختلف برای استفاده در شرط یک دستور if دیدیم. اما دستور if بسیار انعطاف پذیر است. شما می‌توانید برای شرط این دستور از سؤال‌هایی مثل موارد زیر استفاده کنید که پاسخ همه آن‌ها بله یا خیر است:

- ✓ آیا Number از 20 بزرگ‌تر است؟
- ✓ آیا Number از 20 کوچک‌تر است؟
- ✓ آیا Number بزرگ‌تر یا مساوی 20 است؟
- ✓ آیا Number کوچک‌تر یا مساوی 20 است؟
- ✓ آیا Name برابر با Javad است؟

هنگام کار با متغیرهای رشته ای، معمولاً از شرط‌های برابر بودن یا مخالف بودن استفاده می‌کنند. اما هنگام کار با متغیر های عددی می‌توانید از تمام عملگرهای ریاضی استفاده کنید.

استفاده از عملگر مخالف:

تا کنون از عملگر مخالف استفاده نکرده ایم. بنابراین در مثال بعدی نحوه‌ی استفاده از این عملگر را با متغیرهای رشته ای خواهیم دید.

مثال: استفاده از عملگر مخالف

1. یک پروژه جدید به نام if Demo ایجاد کنید.
2. هنگامی که محیط طراحی Page1 را مشاهده کردید، یک شی Input و یک شی Button را از نوار ابزار Standard (یا از منوی Objects) بر روی صفحه قرار دهید. خاصیت Text را برای شی Input برابر با Robbin قرار دهید. سپس خاصیت Text شی Button را برابر با Check قرار دهید.
3. کد زیر را پس از دو بار کلیک کردن روی Button1 در قسمت Script > On Click آن بنویسید و با کلیک Ok کد را تایید کنید:

```
strName = Input.GetText("Input1");
if (strName ~= "Mehdi") then
Dialog.Message("If Demo", "The name is *not* Mehdi.");
end
```

4. برنامه را با کلید F5 اجرا کنید و روی دکمه‌ی Check کلیک کنید. کادر پیغامی را مشاهده خواهید نمود که می‌گوید نام داخل جعبه متنی برابر با Mehdi نیست.

عملگر مخالف در AMS به صورت ~= نوشته می‌شود. هنگامی که کاربر روی دکمه‌ی Check کلیک می‌کند، اول متن درون Input1 با خاصیت GetText به دست آورده می‌شود و درون یک متغیر قرار می‌گیرد.

```
strName = Input.GetText("Input1");
```

بعد از این که نام وارد شده متغیر قرار گرفت، از عملگر مخالف درون مقدار درون متغیر که رشته است با یک عبارت رشته ای دیگر مقایسه شده و نتیجه‌ی این مقایسه برای اجرای دستورات if استفاده می‌شود.

```
if (strName ~= "Mehdi") then
Dialog.Message("If Demo", "The name is *not* Mehdi.");
end
```

همان طور که گفتیم، دستورات درون if فقط در صورتی اجرا می‌شوند که نتیجه‌ی داخل پرانتز صحیح باشد. ممکن است عملگر مخالف در ابتدا مقداری گیج کننده به نظر برسد اما توجه کنید سؤالی که در این قسمت پرسیده می‌شود این است:"آیا مقدار strName مخالف Mehdi است؟". در این حالت پاسخ سؤال به صورت "بله، مقدار strName مخالف با Mehdi است" خواهد بود. بنابراین پاسخ این سؤال به صورت بله است که درست ارزیابی می‌شود. دقت کنید که اگر مقدار Mehdi در Input1 وارد کنید، مقدار شرط برابر با غلط خواهد بود و دستورات if اجرا نخواهد شد.

نکته: اگر می‌خواهید متن Mehdi را در Input1 وارد کنید، توجه کنید که حتماً این متن به همان صورت که در کد نوشته شده بررسی می‌شود(G به صورت حروف بزرگ). زیرا بررسی

شرط در این قسمت به صورت حساس به حروف انجام می‌شود و اگر فرضاً عبارت mehdi را وارد کنید، دو مقدار برابر نخواهد بود و مجدداً کادر متنی نمایش داده می‌شود.

استفاده از عملگرهای مقایسه ای:

در این بخش چهار عملگر مقایسه ای دیگر را معرفی خواهیم کرد. در مثال‌های بعدی عملگرها آشنا خواهید شد:

مثال: استفاده از عملگر کوچک‌تر (<)

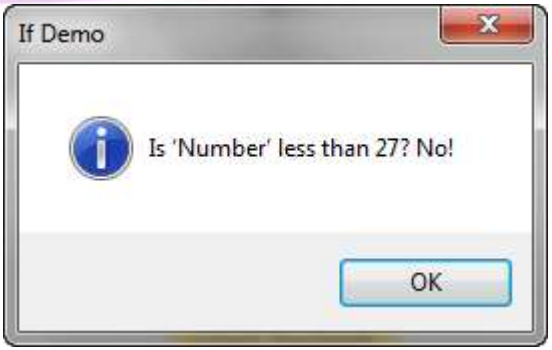
1. اگر برنامه if Demo در حال اجرا است آن را ببندید. یک Input و یک Button به پروژه اضافه کنید. خاصیت Text را در Button2 برابر با Check Numbers قرار دهید.

یادآوری: برای مشاهده‌ی نام شی وارد شده به خاصیت Name آن در پنجره‌ی Properties مراجعه کنید.

2. کد زیر را پس از دو بار کلیک کردن روی Button2 در قسمت Script > On Click آن بنویسید و با کلیک Ok کد را تایید کنید.(هم چنین می‌توان از پنجره‌ی Properties به رویداد On Click دسترسی پیدا کرد):

```
Number  = Input.GetText("Input2");
Number = String.ToNumber(Number);
if (Number < 27 ) then
Dialog.Message("If Demo", "Is ‘Number’ less than 27? Yes!");
else
Dialog.Message("If Demo", "Is ‘Number’ less than 27? No!");
end
```

3. برنامه را اجرا کنید. عددی را در Input2 وارد کنید و روی دکمه‌ی Check Numbers کلیک کنید. کادر پیغامی را مشاهده خواهید کرد که به شما می‌گوید عدد وارد شده در Input2 بزرگ‌تر از 27 است یا نه؟(تصویر شماره 3-4)



تصویر 3 - 4

در این برنامه ابتدا باید مقدار عددی وارد شده در Input2 را به دست آورد. برای این کار ابتدا باید مطمئن شویم متن داخل Input2 شامل عدد است. زیرا کاربر برنامه آزاد است که هر متنی را در اینجا وارد کند و ممکن است متن وارد شده شامل هیچ عددی نباشد که در این حالت برنامه در تبدیل آن به یک عدد با شکست مواجه می‌شود و بقیه کد اجرا نمی‌شود. بنابراین در این قسمت باید کدی را برای مدیریت این مورد وارد کنید تا اگر مقداری غیر از عدد را وارد کرد متغیر Number برابر با صفر شود، در غیر این صورت عدد وارد شده توسط کاربر در آن قرار گیرد.

```
Number  = Input.GetText("Input2");
Number = String.ToNumber(Number);
```

برای تبدیل یک رشته که شامل عدد است به یک عدد صحیح باید از دستور String.ToNumber استفاده کنید. این دستور یک رشته را که شامل عدد است دریافت کرده و عدد معادل آن را بر می‌گرداند. اگر رشته ای که به این تابع فرستاده می‌شود شامل عدد نباشد، یا حتی دارای کاراکتری غیر عددی باشد، تابع عدد صفر را بر می‌گرداند.

در بخش بعدی، به وسیله‌ی دستور if بررسی می‌کنید که عدد وارد شده در Input2 بزرگ‌تر از 27 است یا نه و بر اساس آن پیغام مناسبی را به کاربر نمایش می‌دهید.

```
if (Number < 27 ) then
Dialog.Message("If Demo", "Is ‘Number’ less than 27? Yes!");
else
Dialog.Message("If Demo", "Is ‘Number’ less than 27? No!");
```

```
end
```

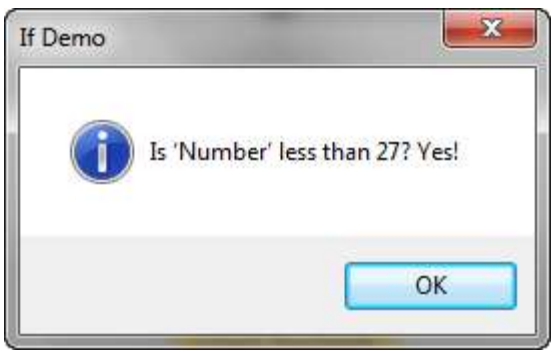
جالب اینجاست که اگر دقیقاً مقدار 27 را وارد کنید، گفته می‌شود که عدد کوچک‌تر از 27 نیست. زیرا عملگر کوچک‌تر فقط در صورتی مقدار درست را بر می‌گرداند که عدد کوچک‌تر باشد نه بزرگ‌تر یا مساوی. برای این که شرط بالا خود عدد 27 را نیز شامل شود از عملگر کوچک‌تر مساوی استفاده کنید که در مثال بعدی شرح داده شده است.

مثال: استفاده از عملگر کوچک‌تر مساوی(<=)

1. در پروژه‌ی if Demo کد موجود در رویداد On Click مربوط به شی Button2 را به صورت زیر تغییر دهید.

```
Number  = Input.GetText("Input2");
Number = String.ToNumber(Number);
if (Number < 27 ) then
Dialog.Message("If Demo", " Is ‘Number’ less than or equal to 27? Yes!");
else
Dialog.Message("If Demo", " Is ‘Number’ less than or equal to 27? No!");
end
```

2. حال برنامه را اجرا کنید و عدد 27 را در Input2 وارد کنید. کادر پیغامی مشابه تصویر 3- 5 را مشاهده خواهید نمود.



تصویر 3 - 5

ملاحظه می‌کنید، تنها تفاوتی که این برنامه نسبت به برنامه قبلی دارد این است که شرط دستور if خود عدد 27 را نیز شامل می‌شود. یعنی اگر در ورودی متن عدد 27 را وارد کنید، کادر پیغامی مشابه تصویر 3- 5 را مشاهده خواهید کرد.

دو عملگر دیگر مشابه عملگرهای قبلی می‌باشند که در ادامه آن‌ها را بررسی خواهیم نمود.

مثال: استفاده از عملگر بزرگ‌تر و بزرگ‌تر مساوی

1. در پروژه‌ی if Demo در ادامه‌ی کد موجود در رویداد On Click مربوط به شی Button2 کد زیر را بنویسید.

```
if (Number > 27 ) then
Dialog.Message("If Demo", "Is ‘Number’ greater than 27? Yes!");
else
Dialog.Message("If Demo", "Is ‘Number’ greater than 27? No!");
end

if (Number >= 27 ) then
Dialog.Message("If Demo", "Is ‘Number’ greater than or equal to 27? Yes!");
else
Dialog.Message("If Demo", "Is ‘Number’ greater than or equal to 27? No!");
end
```

2. برنامه را با فشردن کلید F5 اجرا کنید و مقدار 99 را در جعبه متنی Input2 وارد کنید. سپس روی دکمه‌ی Check Numbers کلیک کنید. سه کادر پیغام متوالی را مشاهده خواهید کرد. در کادر پیغام اول گفته می‌شود که عدد کوچک‌تر یا مساوی 27 نیست. در کادر پیغام دوم و سوم به ترتیب مشاهده می‌کنید که عدد بزرگ‌تر و هم چنین بزرگ‌تر مساوی 27 است.

عملگر های بزرگ‌تر و بزرگ‌تر مساوی دقیقاً برعکس عملگر کوچک‌تر و کوچک‌تر مساوی عمل می‌کند. سؤالاتی که در این قسمت برای شرط if می‌پرسید به صورت " آیا Number بزرگ‌تر از 27 است؟" و" آیا Number بزرگ‌تر یا مساوی 27 است؟" خواهد بود که بر اساس پاسخ آن‌ها کد مربوط به شرط if اجرا خواهد شد.

عملگرهای and و or:

در بعضی از شرایط ممکن است بخواهید به جای یک شرط، چند شرط را در دستور if بررسی کنید. برای مثال می‌خواهید بدانید آیا Number بزرگ‌تر از 27 و کوچک‌تر از 10 است یا نه و یا می‌خواهید بدانید آیا Name برابر با "Hamseda" و یا "Stephanie " است یا نه. در این موارد می‌توانید شرط‌های درون if را به وسیله عملگرهای and و or منطقی ترکیب کنید. در مثال بعدی، روش استفاده از عملگر or را خواهیم دی. نتیجه‌ی ترکیب چند شرط به وسیله‌ی این عملگر فقط هنگامی درست است که حداقل یکی از شرط‌ها برابر با درست باشد.

مثال: استفاده از عملگر های and و or

1. پروژه جدیدی به نام And Or Demo ایجاد کنید.
2. دو شی Input و یک شی Button به برنامه اضافه کنید. خاصیت Text مربوط به Input1 را برابر با hamseda و خاصیت Text مربوط به Input2 را برابر با Hamed قرار دهید و در آخر خاصیت Text شی Button1 را برابر با Or Check قرار دهید. بعد از این تنظیمات پروژه شما با ید مشابه تصویر 3- 6 باشد.

تصویر 3 - 6

3. کد زیر را پس از دو بار کلیک کردن روی دکمه‌ی Button1 در قسمت Script > On Click آن بنویسید و با کلیک Ok کد را تایید کنید:

```
strName1 = Input.GetText("Input1");
strName2 = Input.GetText("Input2");
```

```
if (strName1 == "hamseda" or strName2 == " hamseda") then
Dialog.Message("And Or Demo","One of the names is hamseda.");
else
Dialog.Message("And Or Demo","Neither of the names is hamseda.");
end
```

4. برنامه را با فشردن کلید F5 اجرا کنید و روی دکمه‌ی Or Check کلیک کنید. کادر پیغامی را مشابه تصویر 3- 7 مشاهده خواهید کرد.

تصویر 3 - 7

5. روی دکمه‌ی Ok در کادر پیغام کلیک کنید تا به صفحه اصلی برنامه برگردید. حال متن داخل Input1 را به Hamed و متن Input2 را به hamseda تغییر دهید. مجدداً روی دکمه‌ی Or Check کلیک کنید. کادر پیغامی را مشاهده می‌کنید که می‌گوید یکی از متن‌ها شامل hamseda است.
6. مجدداً روی Ok کلیک کنید و در صفحه اصلی متن‌های داخل صفحه را به گونه ای تغییر دهید که هیچ کدام از آن‌ها شامل hamseda نباشد و روی دکمه‌ی Or Check کلیک کنید. کادر پیغامی را مشاهده خواهید کرد که می‌گوید هیچ یک از آن‌ها شامل hamseda نیست.

عملگر or (با حروف کوچک) در AMS به عنوان "یا" منطقی استفاده می‌شود و معمولاً در بررسی شرط‌ها، برای ترکیب دو شرط متفاوت از هم به کار می‌رود. در رویداد On Click ابتدا دو متغیر تعریف می‌کنیم و مقادیری که کاربر در دو Input وارد کرده است را در آن‌ها قرار می‌دهیم.

```
strName1 = Input.GetText("Input1");
```

```
strName2 = Input.GetText("Input2");
```

حال که هر دو نام را از داخل Input ها به دست آوردید، می‌توانید آن‌ها را در یک شرط if با استفاده از عملگر or ترکیب کنید. در این حالت سؤالی که شما در بخش شرط if ایجاد می‌کنید به صورت " آیا مقدار strName1 برابر با hamseda است و یا مقدار strName2 برابر با hamseda است؟" خواهد بود. در این حالت هر یک از متغیرها که برابر با Hamseda باشد موجب می‌شود که پاسخ سؤال برابر با درست باشد.

```
if (strName1 == "hamseda" or strName2 == "hamseda") then
Dialog.Message("And Or Demo","One of the names is hamseda.");
else
Dialog.Message("And Or Demo","Neither of the names is Hamseda.");
end
```

استفاده از عملگر and منطقی:

این عملگر هم مانند عملگر or منطقی است به جز این که برای درست بودن شرط آن، باید تک تک شرط‌ها درست ارزیابی می‌شوند، این عملگر به صورت and (با حروف کوچک) نوشته می‌شود.

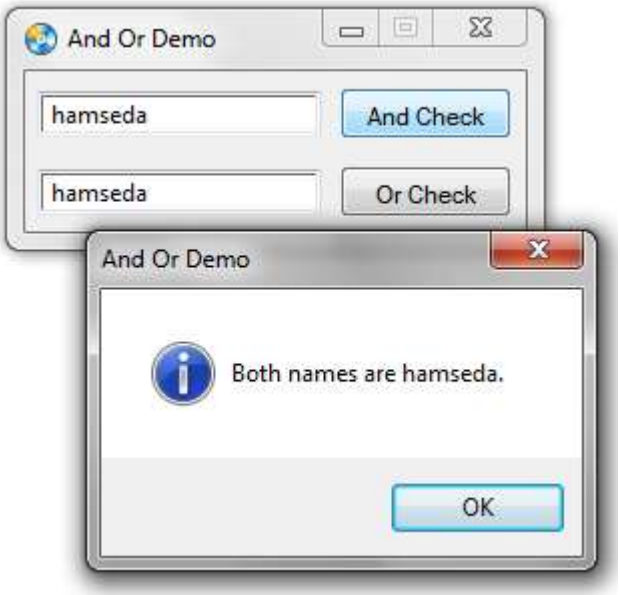
1. شی Button دیگری به پروژه‌ی And Or Demo اضافه کنید، خاصیت Text آن را برابر با And Check قرار دهید. سپس بر روی این کنترل دو بار کلیک کرده و در قسمت Script > On Click کد زیر را وارد کنید:

```
strName1 = Input.GetText("Input1");
strName2 = Input.GetText("Input2");
if (strName1 == "hamseda" and strName2 == "hamseda") then
Dialog.Message("And Or Demo", "Both names are hamseda.");
else
Dialog.Message("And Or Demo", "One of the names is not Hamseda.");
end
```

2. برنامه را با فشردن کلید F5 اجرا کنید و روی دکمه‌ی And Check کلیک کنید. کادر پیغامی را خواهید دید که به شما می‌گوید یکی از متن‌های وارد شده شامل hamseda نیست.
3. البته اگر متن داخل هر دو Input را به hamseda تغییر دهید، نتیجه ای مشابه تصویر 3 – 8 را خواهید دید.

در این مثال بعد از اینکه نام‌های موجود در Input را به دست آوردید، آن‌ها را با هم مقایسه می‌کنید. در اینجا با استفاده از عملگر and می‌پرسید "آیا strName1 برابر با hamseda و strName2 برابر با hamseda است؟". واضح است جواب این سؤال هنگامی درست است که هر دو Input محتوی کلمه‌ی Hamseda باشند.

```
if (strName1 == "hamseda" and strName2 == "hamseda") then
Dialog.Message("And Or Demo", "Both names are hamseda.");
else
Dialog.Message("And Or Demo", "One of the names is not
hamseda.");
end
```



تصویر 3 - 8

مطالب بیشتر در رابطه با عملگرهای and و or منطقی:

تا کنون با استفاده از عملگرهای and و or در رشته‌ها آشنا شده‌اید، اما می‌توانید این عملگرها را با اعداد نیز همانند زیر به کار ببرید:

```
X = 2;
Y = 2.3;
if (X == 2 and Y == 2.3) then
Dialog.Message("Hello", "the conditions has been satisfied!");
end

if (X == 2 or Y == 2.3) then
Dialog.Message("Hello", "the conditions have been satisfied!");
end
```

هم چنین در استفاده از عملگرهای and و or در یک دستور if هیچ محدودیتی نیست. به عبارت دیگر می‌توانید در برنامه خود دستوری مشابه زیر داشته باشید:

```
if (X1 == 1 and X2 == 2 and X3 == 3 and X4 == 4) then
Dialog.Message("Hello", "That's quite an If statement!");
end
```

البته باید توجه داشته باشید که استفاده زیاد از این عملگرها از خوانایی برنامه می‌کاهد و درک آن را مشکل‌تر می‌کند. پس تا حد امکان باید از شرط‌های کمتر در دستور if استفاده کنید.

هم چنین می‌توانید از چند عملگر and و or در شرط خود استفاده کنید. در این مواقع می‌توانید با استفاده از پرانتزها این عملگرها را دسته بندی کنید. برای مثال می‌خواهید اگر مقدار متغیری بین 10 تا 20 و یا بین 25 تا 35 باشد، دستورات داخل شرط اجرا شوند، در این صورت می‌توانید از دستور if زیر استفاده نمایید:

```
if (X > 10 and X < 20) or (X > 25 and X < 35) then
دستورات
end
```

حلقه های تکرار

هنگامی که در حال نوشتن یک برنامه هستید ممکن است بخواهید یک عمل مشخص را چندین بار متوالی انجام دهید تا نتیجه‌ی مطلوب خود را دریافت کنید. مثلاً ممکن است

بخواهید 10 عدد را فراخوانی کرده و مجموع آن‌ها را حساب کنید و یا 10 فایل را از روی درایوی بخوانید.

در برنامه نویسی، برای انجام این امور معمولاً از حلقه‌ها استفاده می‌شود در این قسمت شما با سه دسته کلی از حلقه‌ها که در AMS کاربرد دارند آشنا می‌شوید.

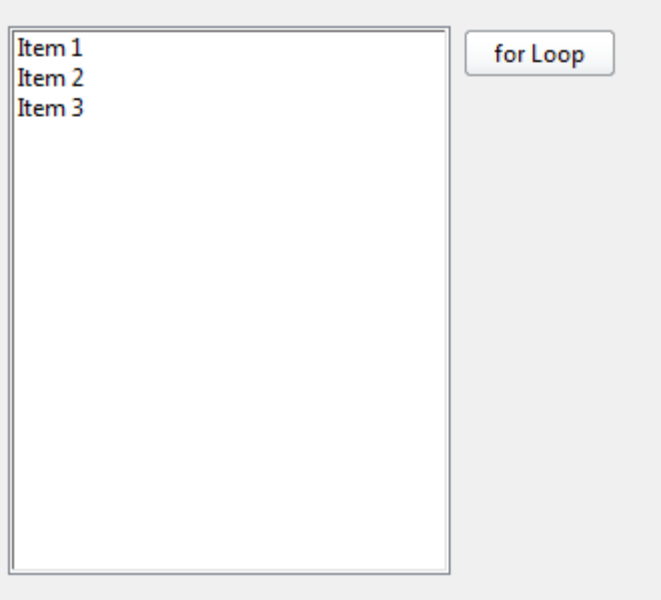
1. حلقه های for: این دسته از حلقه‌ها به تعداد معمولاً به تعداد مرتبه‌ی مشخصی اجرا می‌شوند(برای مثال: 10 بار)
2. حلقه‌ی while: این حلقه معمولاً تا هنگامی که نتیجه یک شرط درست شود ادامه می‌یابند.
3. حلقه‌ی repeat: عملکرد این حلقه نیز همانند حلقه‌ی while است با این تفاوت که شرط حلقه‌ی while در ابتدا بررسی می‌شود ولی در این حلقه شرط در انتها بررسی می‌گردد.

حلقه های for:

درک نحوه کارکرد حلقه‌ی for بسیار راحت است و به سه شکل به کار می‌رود که با هر سه شکل در مثال‌های زیر آشنا خواهیم شد.

مثال: ایجاد یک حلقه‌ی for نوع 1

1. یک پروژه جدید به نام Loops ایجاد کنید.
2. یک شی ListBox و یک شی xButton از منوی Objects به پروژه بیفزایید.
3. خاصیت Text را برای xButton1 برابر با For Loop قرار دهید و اندازه پروژه را از منوی Project>Settings و قسمت Page Size در گروه Dimensions برابر با 330x300 قرار دهید.
4. پروژه‌ی شما باید مشابه تصویر 3 – 10 باشد.

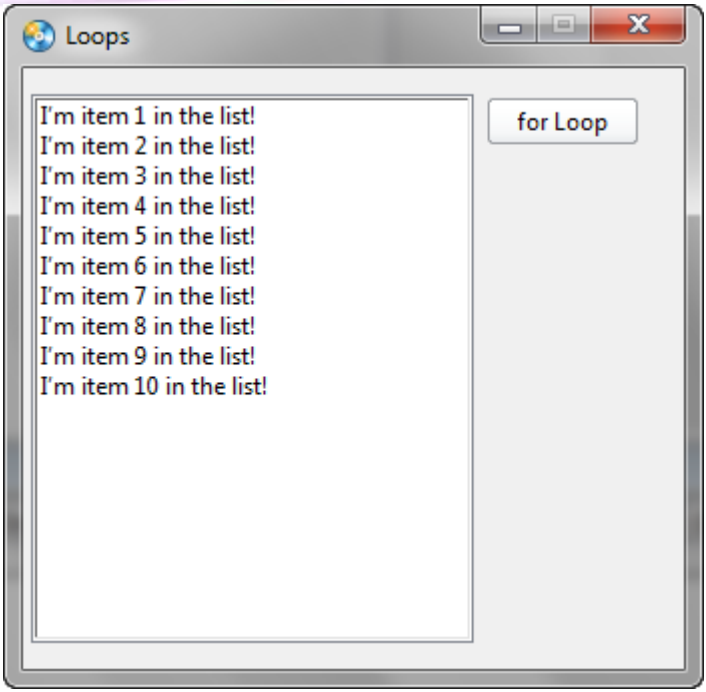


تصویر 3 - 9

5. بر روی دکمه‌ی xButton1 دو بار کلیک کنید و در قسمت Script>On Click کد زیر را وارد کنید:

```
ListBox.DeleteItem("ListBox1", -1);
for i = 1, 10 do
ListBox.AddItem("ListBox1", "I’m item "..i.." in the list!", "");
end
```

برنامه را اجرا کنید و بر روی دکمه‌ی for Loop کلیک کنید. نتیجه ای مشابه تصویر 3- 11 را مشاهده خواهید کرد.



تصویر 3 - 10

ابتدای کد مورد نظر ما دستوری نوشتم که تمام آیتم‌های موجود در لیست را پاک کند.

```
ListBox.DeleteItem("ListBox1", -1);
```

برای ایجاد حلقه از کلمه‌ی for استفاده نمودیم. این کلمه به برنامه می‌گوید که می‌خواهیم یک حلقه با تعداد دفعات تکرار مشخص ایجاد کنیم. تمام کلمات و علامت‌هایی که بعد از این کلمه می‌آیند، برای مشخص کردن نحوه‌ی عملکرد این حلقه به کار می‌روند. برای تعیین نحوه‌ی کارکرد این حلقه دو مورد را باید جلوی آن مشخص کنیم. این دو مورد با علامت ; از یکدیگر جدا می‌شوند.

در قسمت اول باید متغیری از با نام دلخواه ایجاد کرد و مقدار آن را برابر با یک عدد قرار داد. در این مثال چون می‌خواهیم شمارش از عدد 1 شروع شود مقدار متغیرمان(i) را مساوی با 1 قرار می‌دهیم.

در قسمت دوم باید تعیین کنیم که حلقه، شمارش را تا چه عددی ادامه دهد. در این مثال تا زمانی که متغیر i مساوی 10 قرار گیرد شمارش ادامه می‌یابد و در هر بار شمارش دستور درون حلقه اجرا می‌گردد

```
for i = 1, 10 do
ListBox.AddItem("ListBox1", "I'm item "..i.." in the list!", "");
end
```

نکته: حلقه های for همیشه همراه با do می‌باشند.

دستور درون حلقه نیز عبارتی مشخص را به اضافه‌ی شماره‌ی شمارنده متغیر i در هر بار شمارش به لیست اضافه می‌کند.

همان طور که تا کنون متوجه شده‌اید، در حلقه اجباری نیست که مقدار شروع حلقه را عدد یک در نظر بگیرید.

مثال: ایجاد یک حلقه‌ی for نوع 2

1. یک شی xButton دیگر به پروژه اضافه کنید و مقدار خاصیت Text ان را برابر با for Loop 2 قرار دهید و آن را با کشیدن ماوس زیر دکمه‌ی اول قرار دهید.
2. بر روی دکمه‌ی xButton2 دو بار کلیک کنید و در قسمت Script>On Click کد زیر را وارد کنید:

```
ListBox.DeleteItem("ListBox1", -1);
for i = 1, 10, 2 do
ListBox.AddItem("ListBox1", "I'm item "..i.." in the list!", "");
end
```

3. برنامه را اجرا کنید و روی دکمه‌ی for Loop 2 کلیک کنید. نتیجه مشابه تصویر 3- 12 خواهد بود.

تصویر 3 - 11

این نوع حلقه به غیر از دو قسمتی که در مثال نوع 1 بیان شد قسمتی سومی نیز دارد که تعیین می‌کند در هر مرحله مقدار متغیر قسمت اول یعنی i در مثال بالا چه تغییری بکند. که در این مثال در هر بار شمارش دو واحد به مقدار متغیر i اضافه می‌گردد.

شمارش معکوس: در این نوع از حلقه می‌توان از مقدار متغیر کم کرد و شمارش را به صورت معکوس انجام داد به کد زیر دقت کنید:

```
for i = 10, 1, -1 do
ListBox.AddItem("ListBox1", "I'm item "..i.." in the list!", "");
end
```

این کد شمارش را از عدد 10 به سوی عدد 1 انجام می‌دهد تا اینکه به عدد 1 برسد و اعداد را از بزرگ به کوچک در لیست اضافه می‌کند.

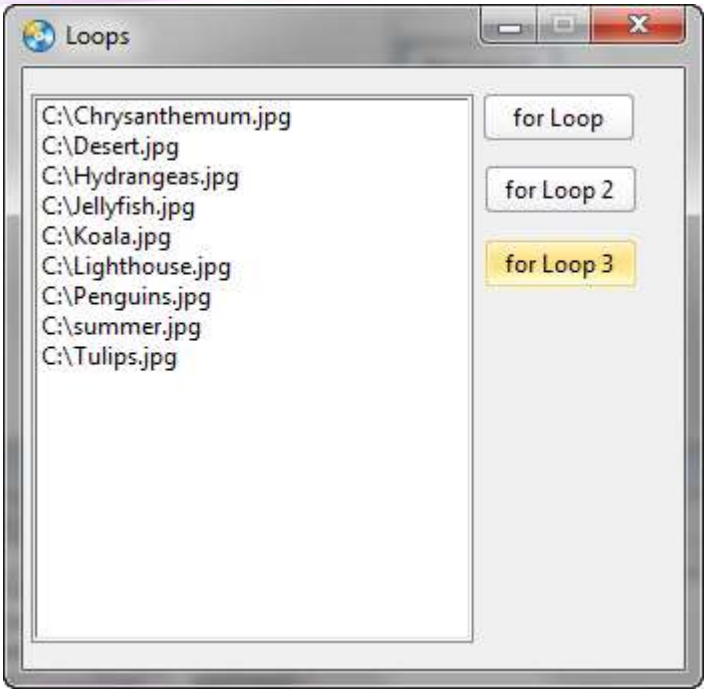
مثال: ایجاد یک حلقه‌ی for نوع 3

این نوع از حلقه‌ی for زمانی کاربرد دارد که شما می‌خواهید در بین آرایه ای از داده‌ها که تعداد آن‌ها مشخص نیست جا به جا شوید برای مثال ممکن است بخواهید درون یک پوشه بگردید و فایل‌هایی را که پسوند خاصی دارند لیست کنید.

1. یک شی xButton دیگر به پروژه اضافه کنید و مقدار خاصیت Text ان را برابر با for Loop 3 قرار دهید و آن را با کشیدن ماوس زیر دکمه‌ی اول قرار دهید.
2. بر روی دکمه‌ی xButton3 دو بار کلیک کنید و در قسمت Script>On Click کد زیر را وارد کنید:

```
ListBox.DeleteItem("ListBox1", -1);
files = File.Find("C:\\", "*.jpg", false, false, nil, nil);
for index, address in pairs(files) do
ListBox.AddItem("ListBox1", address, "");
end
```

3. برنامه را اجرا کنید و روی دکمه‌ی for Loop 3 کلیک کنید. نتیجه مشابه تصویر 3- 13 خواهد بود.



تصویر 3 - 12

سطر دوم کد بالا یعنی File.Find برای جستجو در یک مسیر به کار می‌رود که توسط کاربر یا برنامه نویس مشخص می‌شود این دستور آرایه ای از آدرس فایل‌های مشخص شده را بر می‌گرداند. سطر سوم یعنی سطری که حلقه‌ی ما از آن جا شروع می‌شود. index در ابن حلقه متغیر شمارنده ای است که تعداد دفعات تکرار حلقه در آن ذخیره می‌شود. address نیز متغیری است که آدرس فایل مورد نظر در آرایه را با توجه به شمارنده address در خود نگه می‌دارد یعنی اگر index برابر با 2 باشد address هم برابر با آدرس فایل دوم در آرایه‌ی files می‌باشد. in pairs(files) نیز به معنی این است که در آرایه ما files می‌باشد که با هر بار تکرار حلقه باید سطری از آن مطابق شمارنده به address تعلق گیرد. سطر چهارم نیز رشته‌ی آدرس بدست آمده را از متغیر address می‌گیرد و در لیست می‌ریزد.

نکته: همان طور که می بنید به جای استفاده از رشته‌ی "C:\" به عنوان پارامتر برای تابع File.Find از رشته‌ی "C:\\" استفاده کرده‌ایم. در زبان برنامه نویسی لوا کاراکتر \ به عنوان یک کاراکتر کنترلی در نظر گرفته می‌شود، فرض کنید می‌خواهید رشته‌ی A " Sign را در

یک متغیر رشته ای ذخیره کنید. این رشته شامل کاراکتر " است که برای مشخص کردن انتهای رشته به کار می‌رود. پس نمی‌توان به حالت عادی این رشته را در یک متغیر رشته ای قرار داد. برای اینکه به برنامه بگوییم کاراکتر " جزیی از رشته است، باید از کاراکتر \ قبل از آن استفاده کنیم. به همین ترتیب برای اینکه به برنامه بگوییم در رشته‌ی "C:\\" کاراکتر \ جزیی از رشته است، باید دو \ به طور متوالی استفاده کنیم و رشته را به صورت "C:\\" بنویسیم. اگر این عبارت را به صورت "C:\" بنویسیم برنامه تصور می‌کند که شما انتهای رشته را مشخص نکرده‌اید و خطا ایجاد می‌کند زیرا علامت " را در انتهای رشته به عنوان بخشی از رشته، به عنوان بخشی از عبارت محسوب می‌کند.

حلقه‌ی while

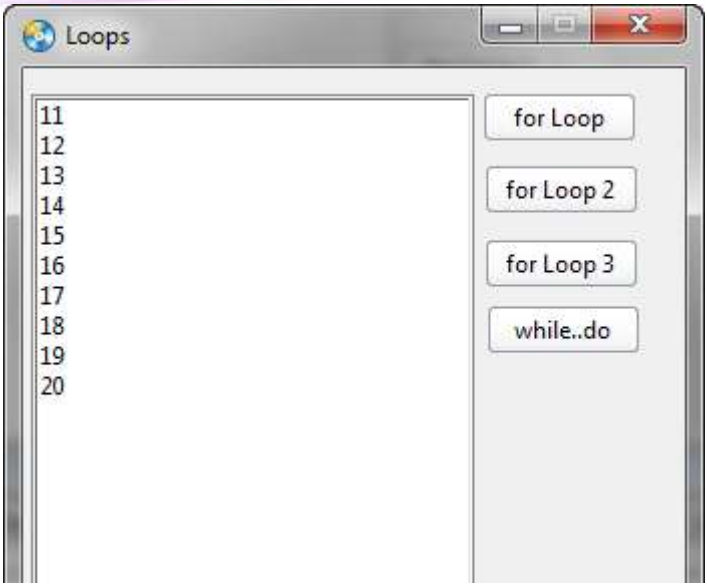
نوع دیگری از حلقه‌هایی که می‌توانید در برنامه‌های خود استفاده کنید، حلقه‌هایی هستند که تا زمان برقراری یک شرط مشخص اجرا می‌شوند. حلقه‌ی while یکی از این نوع حلقه‌ها است؛ شرط این حلقه در ابتدا بررسی می‌شود و سپس در صورت درست بودن دستورات درون حلقه اجرا می‌گردد.

مثال: استفاده از حلقه‌ی while

1. به پروژه‌ی Loops یک شی xButton دیگری اضافه کنید و خاصیت Text آن را برابر با while..do قرار دهید.
2. به رویداد On Click مربوط xButton4 بروید و کد زیر را در آن بنویسید:

```
ListBox.DeleteItem("ListBox1", -1);
testvar = 10;
while (testvar < 20) do
testvar = testvar + 1;
ListBox.AddItem("ListBox1", testvar);
end
```

3. برنامه را اجرا کنید و روی دکمه‌ی while..do کلیک کنید. نتیجه ای را مشابه تصویر 3- 14 مشاهده خواهید کرد:



تصویر 3 - 13

همان طور که گفتیم شرط حلقه‌ی while در ابتدا بررسی می‌شود و در صورت درست بودن آن، دستورات حلقه ادامه پیدا می‌کند. در غیر این صورت برنامه به اولین خط بعد از حلقه می‌رود. در این مثال ابتدا بررسی می‌شود که آیا مقدار testvar کمتر از 20 است یا نه؟ در صورتی که شرط درست باشد دستورات داخل حلقه اجرا می‌شوند.

```
while (testvar < 20) do
testvar = testvar + 1;
ListBox.AddItem("ListBox1", testvar);
end
```

بعد از این که حلقه برای مرتبه‌ی اول اجرا شد 1عدد به مقدار متغیر testvar اضافه می‌شود این کار تا زمانی که مقدار متغیر testvar برابر با 20 شود ادامه می‌یابد و پس از آن برنامه از دستورات حلقه خارج می‌شود و به سطر بعدی می‌رود و در صورت وجود دستورات آن را اجرا می‌کند.

نکته: عبارت‌های شرطی در این حلقه با دستور if تفاوتی ندارد. به عبارتی هر عبارتی که در دستور if استفاده می‌کنید می‌توانید آن را به عنوان شرط حلقه while نیز به کار ببرید. برای مثال به کد زیر دقت کنید:

```
while (testvar >= 10 and testvar < 30) do
testvar = testvar + 1;
ListBox.AddItem("ListBox1", testvar);
end
```

حلقه‌ی repeat.. until

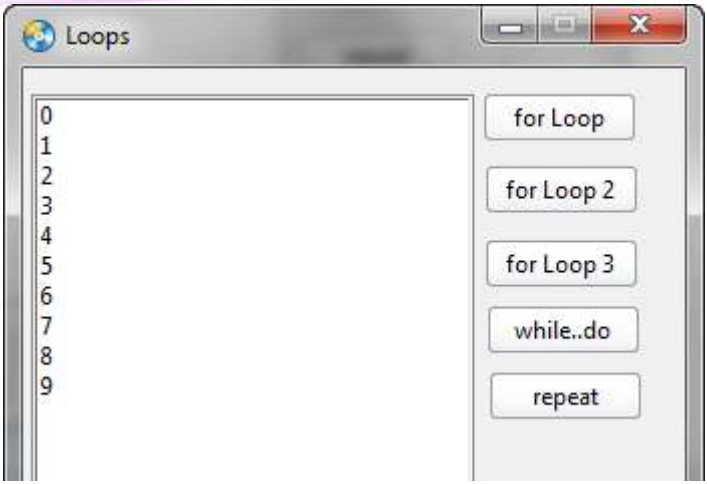
حلقه‌ی repeat نیز همانند حلقه‌ی while از حلقه‌هایی است که تا زمان برقراری یک شرط اجرا می‌شوند با این تفاوت که شرط این حلقه در آخر بررسی می‌شود.

مثال: استفاده از حلقه‌ی repeat

1. به پروژه‌ی Loops یک شی xButton اضافه کنید و خاصیت Text آن را برابر با repeat قرار دهید.
2. به رویداد On Click مربوط xButton5 بروید و کد زیر را در آن بنویسید:

```
ListBox.DeleteItem("ListBox1", -1);
numLoopCount = 0;
repeat
        ListBox.AddItem("ListBox1", numLoopCount, "");
        numLoopCount = numLoopCount + 1;
until numLoopCount == 10;
```

4. پروژه را با فشردن کلید F5 اجرا کنید. نتیجه‌ای را مشابه تصویر 3- 15 مشاهده خواهید کرد:



تصویر 3 - 14

همان طور که در تصویر 3 – 15 می‌بینید شمارش از عدد 0 شروع شده و تا عدد 9 ادامه یافته است و شامل عدد 10 نیست این بدان خاطر است که در این حلقه شرط ما در انتها مورد بررسی قرار گرفته است و درست زمانی که مقدار متغیر numLoopCount برابر با 10 قرار گرفته است برنامه از حلقه خارج شده است. البته می‌توان جای متغیر را عوض کرد و قبل از دستور ListBox.AddItem قرار داد اما این جابه‌جایی در تعداد شمارش حلقه تأثیری نخواهد داشت و فقط مقدار متغیر numLoopCount هنگام اضافه شدن به لیست تغییر خواهد کرد.

شرط این حلقه در پایان بعد از کلمه‌ی until مورد بررسی قرار می‌گیرد:

```
repeat
        ListBox.AddItem("ListBox1", numLoopCount, "");
        numLoopCount = numLoopCount + 1;
until numLoopCount == 10;
```

حلقه های تو در تو:

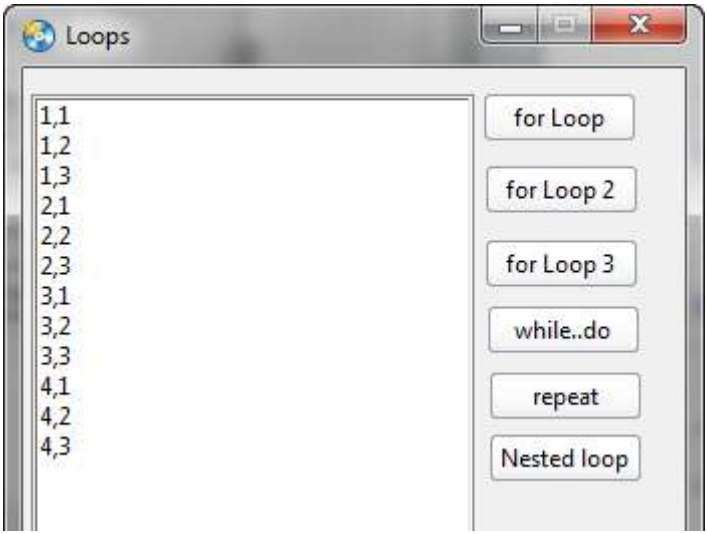
در مواقعی ممکن است نیاز داشته باشیم در حین این که درون یک حلقه هستیم، حلقه‌ی جدید را شروع کنیم؛ به این نوع حلقه‌ها، حلقه های تو در تو می‌گویند و تقریباً همانند دستورات if تو در تو هستند.

مثال: استفاده از حلقه های تو در تو

1. یک شی xButton به پروژه‌ی Loops اضافه کنید و خاصیت Text آن را برابر با Nested loop قرار دهید.
2. به رویداد On Click مربوط xButton6 بروید و کد زیر را در آن بنویسید:

```
ListBox.DeleteItem("ListBox1", -1);
for i = 1 , 4 do
        for x = 1 , 3 do
        ListBox.AddItem("ListBox1", i..","..x, "");
        end
end
```

3. پروژه را با فشردن کلید F5 اجرا کنید. نتیجه‌ای را مشابه تصویر 3- 16 مشاهده خواهید کرد:



تصویر 3 - 15

این کد کاملاً مشخص است و جای ابهامی ندارد. حلقه اول(حلقه بیرونی) با استفاده از شمارنده‌ی i از عدد 1 تا 4 و حلقه دوم(حلقه درونی) با استفاده از شمارنده‌ی x از 1 تا 3

حرکت می‌کند. در حلقه درونی کدی برای نمایش مقدار متغیر های i و x وجود دارد که آن‌ها را به لیست اضافه می‌کند:

```
for i = 1 , 4 do
        for x = 1 , 3 do
        ListBox.AddItem("ListBox1", i..","..x, "");
        end
end
```

در حلقه های تو در تو توجه داشته باشید حلقه ای که درون یک حلقه‌ی دیگر شروع می‌شود در همان حلقه نیز تمام می‌شود. به عبارت دیگر کدهای مربوط به حلقه‌ی درونی نمی‌توانند در خارج از حلقه قرار گیرند. در این مثال اولین end نشانه بسته شدن حلقه درونی است. هنگامی که برنامه برای اولین بار وارد حلقه i می‌شود حلقه‌ی x را سه بار اجرا می‌کند. این تکرار ادامه پیدا می‌کند تا دفعات تکرار حلقه به پایان برسد. به این ترتیب برنامه از هر دو حلقه خارج می‌شود و به سطر بعدی می‌رود.

خروج زود هنگام از حلقه:

برخی از مواقع، در برنامه نیازی نیست که یک حلقه‌ی for تا انتها اجرا شود. مثلاً در یک لیست به دنبال موردی خاص می‌گردید و می‌خواهید با پیدا شدن آن مورد از حلقه خارج شوید. در مثال بعدی کدی خواهید نوشت که به برنامه محض رسیدن به عددی خاص از حلقه خارج شود.

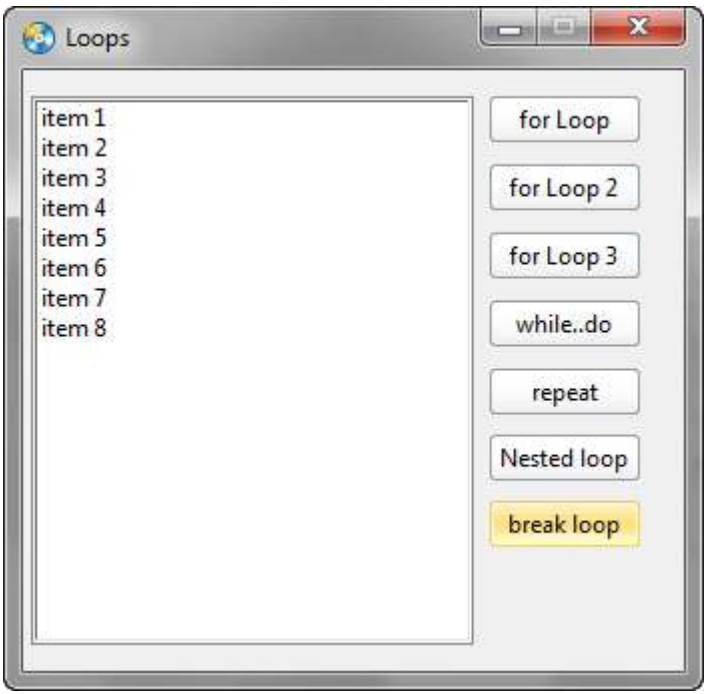
مثال: خروج زود هنگام از حلقه

1. یک شی xButton به پروژه‌ی Loops اضافه کنید و خاصیت Text آن را برابر با break loop قرار دهید.
2. به رویداد On Click مربوط xButton7 بروید و کد زیر را در آن بنویسید:

```
ListBox.DeleteItem("ListBox1", -1);
for i = 1 , 15 do
        ListBox.AddItem("ListBox1", "item "..i, "");
        if i == 8 then
```

```
        break;
        end
end
```

3. پروژه را با فشردن کلید F5 اجرا کنید. نتیجه‌ای را مشابه تصویر 3- 17 مشاهده خواهید کرد:



تصویر 3 - 16

هر بار که حلقه اجرا می‌شود پس از افزوده شدن مقدار item و مقدار متغیر i به لیست در یک دستور if بررسی می‌شود که آیا مقدار متغیر i برابر با 8 است؟ یا نه؟ در صورتی که مقدار متغیر i برابر با 8 باشد حلقه توسط دستور break شکسته خواهد و برنامه به سطر بعد از حلقه مراجعه می‌کند.

```
        if i == 8 then
        break;
        end
```

نکته: چنانچه شرط مورد نظر ما درست نباشد حلقه تا آخر ادامه خواهد یافت.

نکته: به جای break می‌توانید از دستور زیر استفاده کنید:

```
Application.ExitScript();
```

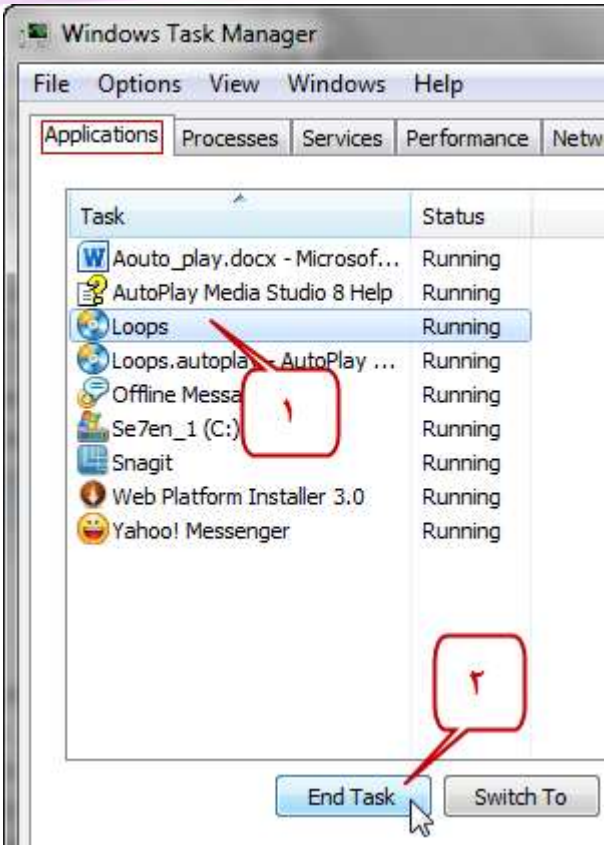
تفاوت این دستور با دستور break این است که در این دستور از کل بلوک خارج می‌شود ولی دستور break فقط از حلقه مورد نظر خارج می‌شود .

حلقه های بی نهایت:

هنگام ایجاد حلقه‌ها، می‌توانید حلقه‌هایی ایجاد کنید که بی نهایت بار اجرا شوند. به عبارت دیگر هنگامی که شروع شوند هیچ وقت تمام نشوند. به حلقه ایجاد شده در کد زیر دقت کنید:

```
testvar = 1;
while (testvar > 0) do
ListBox.AddItem("ListBox1", testvar);
testvar = testvar + 1;
end
```

با شروع حلقه مقدار شمارنده‌ی testvar برابر با عدد 1 است و چون از صفر بزرگ‌تر است حلقه برای بار اول اجرا می‌شود بعد از اجرای دستورات حلقه شمارنده تغییر می‌کند و برابر با 2 می‌شود. در این مرحله نیز چون عدد 2 از صفر بزرگ‌تر است، حلقه مجدداً اجرا می‌شود. همان طور که مشاهده می‌کنید، شمارنده‌ی این حلقه هیچ گاه به عددی کوچک‌تر از صفر نمی‌رسد، پس شرط حلقه همواره درست است. این حلقه بی نهایت بار اجرا می‌شود. در این حین ممکن است برنامه توقف کند و یا اینکه به کلیک‌های شما پاسخ ندهد. در چنین شرایطی اگر ویندوز شما به طور خودکار برنامه را نبندد می‌توانید با راست کلیک روی Taskbar و انتخاب گزینه‌ی Start Task Manager برنامه را مطابق تصویر 3- 18 ببندید و یا ممکن است در صورت عدم اجرای Task Manager اقدام به Restart کردن ویندوز نمایید.



تصویر 3 - 17

آشنایی با تابع (Function)

تابع در AMS یعنی قطعه کدی که قابلیت فراخوانی دارد و باعث سرعت بخشیدن و دقت در برنامه نویسی می‌شود. توابع به غیر از اسمشان معمولاً دارای قسمتی هستند که پارامتر نام دارد و با اجرای تابع آن پارامتر به دستورات داخل تابع فرستاده می‌شود تا کار مورد نظر بر روی آن انجام گیرد.

پارامتر یعنی مقداری که به یک تابع فرستاده می‌شود حال این مقدار می‌تواند هر مقداری باشد از قبیل: رشته، عدد و ... .

برای تعریف تابع از واژه‌ی function می‌شود که مشخص کننده توابع نوشته شده توسط برنامه نویس می‌باشد.

نکته: در AMS معمولاً تابع در پنجره‌ی کد نویسی Globals نوشته می‌شود که در منوی Projects > Global Function... قابل دسترس است.

در اینجا فقط به یک کد مثال بسنده می‌کنیم و در فصل بعدی در مثال‌ها با توابع به صورت عملی بیشتر آشنا خواهید شد.

```
function MessageBox(Title, Message)
Dialog.Message(Title,Message);
end
```

همان طور که مشاهده می‌کنید تابعی به نام MessageBox ایجاد کردیم با دو پارامتر با نام‌های Title و Message؛ سپس در داخل تابع دستور Dialog.Message را نوشتیم که نمایش دهنده‌ی یک کادر پیغام است؛ سپس مقدار عنوان آن را برابر با پارامتر Title و مقدار پیام آن را برابر با Message قرار دادیم و در انتها بعد از پایان تعریف تابع، باید در جایی که نیاز است آن را فراخوانی نماییم.

```
MessageBox("Hello", "How are you ?!");
```

هنگام فراخوانی تابع باید مقدار پارامترها را نیز تعیین کنیم که بسته به نیاز ما می‌تواند از نوع رشته، عدد، بولین و ... است.

در تابع مثال ما نیاز به متنی داریم که به کاربر نشان دهیم پس در هنگام نوشتن کد فراخوانی تابع می‌بایست از رشته استفاده کنیم که ابتدا نام تابع را می‌نویسیم (MessageBox) و سپس مقدار پارامترها را به ترتیب وارد می‌کنیم. ابتدا کلمه‌ی Hello را به جای عنوان کادر پیغام وارد می‌کنیم و سپس عبارت How are you ?! را به جای متن کادر پیغام وارد می‌کنیم.

```
function MessageBox(Title, Message)
Dialog.Message(Title,Message);
end
MessageBox("Hello", "How are you ?!");
```

اگر ما این کد را درون رویداد On Click یک شی xButton قرار دهیم و پس تایید کد، برنامه را اجرا کنیم ؛ با کلیک کردن روی شی xButton مورد نظر ابتدا تابع پارامترها را به درون خود ارسال می‌کند و سپس دستور درون خود را اجرا می‌کند.

توجه داشته باشد که تا زمانی که یک تابع فراخوانی نشود کد درون آن به هیچ وجه اجرا نخواهد شد.

از دیگر امکانات تابع می‌توان به مقدار(های) بازگشتی اشاره کرد.

برای مثال شما نیاز به محاسبه چند عدد با فرمولی خاص دارید .برای جلوگیری از تکرار کد برای هر بار محاسبه تابع را می‌نویسیم:

```
function Func(x, y)
t=(x*2)*(y+2) ;
return t;
end
```

این تابع به صورت زیر فراخوانی می‌شود

```
My_Var= Func(5,12);
```

پس مقدار 140 در My_Var قرار می‌گیرد .

حال در بعضی توابع نیاز به بازگردانی چند متغیر داریم که به صورت زیر نوشته می‌شود:

```
function Func(x, y)
t=(x*2)*(y+2) ;
s=(x*2)*(y)
return t,s;
end
```

فراخوانی این تابع به صورت زیر می‌باشد که مقدار My_Var برابر 140 و مقدار My_Var2 برابر 120 است :

```
My_Var , My_Var2 = Func(5,12);
```

توابع سراسری اصلی:

برخی از توابع در Ams از پیش تعیین شده می‌باشند که با استفاده از آن‌ها در Global Function عملیاتی در برنامه انجام می‌شود که به توضیح مختصر هر یک آن‌ها می‌پردازیم .

1.g_OnSystemTrayMenu (number X, number Y)

این تابع امکان کلیک راست بر روی آیکون پروژه در System Try را فعال می‌کند.

```
function g_OnSystemTrayMenu(X, Y)
  tblMenu = {};
  tblMenu[1] ={};
  tblMenu[1].Text = "&Ams ";
  tblMenu[1].ID = 100;
  tblMenu[1].Checked = false;
  tblMenu[1].Enabled = true;
  tblMenu[1].SubMenu = {};
  tblMenu[1].SubMenu[1] = {};
  tblMenu[1].SubMenu[1].Text = "&VaSvA3";
  tblMenu[1].SubMenu[1].ID = 101;
  tblMenu[1].SubMenu[1].Checked = false;
  tblMenu[1].SubMenu[1].Enabled = true;
  tblMenu[1].SubMenu[2] = {};
  tblMenu[1].SubMenu[2].Text = " &AutoPlay";
  tblMenu[1].SubMenu[2].ID = 102;
  tblMenu[1].SubMenu[2].Checked = false;
  tblMenu[1].SubMenu[2].Enabled = true;

  result = Application.ShowPopupMenu(X, Y, tblMenu,
TPM_RIGHTALIGN, TPM_BOTTOMALIGN, true, false);
```

```
  if(result ~= -1)then
     Dialog.Message("Menu Item Selected",result);
  end
end
```

2. QueryAllowProjectClose ( )

این تابع در هنگام کلیک بر روی دکمه ضربدر بالای پنجره اجرا شده و برای پرسش سؤال برای خروج از نرم افزار مورد استفاده قرار می‌گیرد.

```
function QueryAllowProjectClose()
   result= Dialog.Message("Application Exit", "Are you sure that you want to quit?", MB_YESNO, MB_ICONEXCLAMATION, MB_DEFBUTTON1);
   --if they choose yes
   if result == IDYES then
      --allow the app to close
      return true;
   else
      --cancel close
      return false;
   end
end
```

3. g_OnGetMinMaxInfo ( )

در صورت فعال بودن تغییر اندازه پروژه، این تابع میزان حداکثر و حداقل طول و عرض را تعیین می‌کند.

```
function g_OnGetMinMaxInfo()
tbReturn = {};
tbReturn.MinX = 150;
tbReturn.MinY = 150;
tbReturn.MaxX = 990;
tbReturn.MaxY =730;

return tbReturn;
end
```

4. QueryAllowDialogClose (string DialogName)

این تابع در هنگام کلیک بر روی دکمه ضربدر بالای دیالوگ اجرا شده و برای پرسش سؤال برای بستن دیالوگ مورد استفاده قرار می‌گیرد.

```
function QueryAllowDialogClose(strDialogName)
result= Dialog.Message("Close Dialog", "Are you sure that you want to close the dialog?", MB_YESNO, MB_ICONEXCLAMATION, MB_DEFBUTTON1);
--if they choose yes
if result == IDYES then
--allow the dialog to close
return true;
else
--cancel close
return false;
end
end
```

5. boolean g_OnUpdateMenuEnabled (number CommandID, table ItemInfo)

این تابع در صورتی که نیاز به فعال و غیر فعال کردن یک منو داشته باشید مورد استفاده قرار می‌گیرد.

```
function g_OnUpdateMenuEnabled(CommandID,tblInfo)
-- Disable command 2000
if (CommandID == 2000) then
return false;
else
return true;
end
end
```

6. boolean g_OnUpdateMenuCheck (number CommandID, table ItemInfo)

نوع علامت دار بودن منو با این تابع به دست می‌آید.

```
function g_OnUpdateMenuCheck(CommandID,tblInfo)
   -- Uncheck command 2000
   if (CommandID == 2000) then
      return false;
   else
      return true;
   end
end
```

فصل چهارم: آرایه‌ها

در فصل‌های قبلی نحوه‌ی استفاده از متغیرها مانند متغیرهای رشته ای و عددی را مشاهده کردید. با وجود اینکه این نوع‌های داده ای بسیار پرکاربرد هستند، اما برنامه‌های پیچیده تر نیاز دارند که از آرایه‌ها استفاده کنند.

در این فصل با موارد زیر آشنا می‌شویم:

- آرایه‌ها
- انتقال آرایه‌ها به عنوان پارامتر
- استفاده از حلقه‌ی for در آرایه‌ها
- استفاده از آرایه‌ها به عنوان یک پارامتر برای توابع
- معکوس کردن آرایه‌ها

مفهوم آرایه:

نگهداری لیستی از اطلاعات مشابه و مرتبط به هم یکی از عمومی‌ترین نیازها در برنامه نویسی است. برای این کار می‌بایست از آرایه‌ها استفاده کنید. آرایه‌ها لیستی از داده‌ها می‌باشند که معمولاً همه از یک نوع می‌باشند. به کار بردن کلمه‌ی معمولاً به این دلیل است که در زبان لوا می‌توان یک آرایه را به صورت لیستی از چندین نوع داده تعریف کرد.

تعریف و استفاده از آرایه‌ها:

هنگامی که در برنامه یک آرایه تعریف می‌کنید، در حقیقت متغیری ایجاد می‌کنید که بتواند بیش از یک عنصر را در خود نگهداری کند.

اگر بخواهید یک متغیر را به صورت آرایه تعریف کنید باید در قسمت نوع داده ای متغیر از علامت {} استفاده کنید. برای مثال به کد زیر دقت کنید:

```
strFriends = {};
```

هنگامی که آرایه یک آرایه را ایجاد کردید می‌توانید به تک تک عناصر آن با استفاده از اندیس[۲] آن عنصر دسترسی پیدا کنید. اندیس عناصر یک آرایه همواره عددی بین 1 و شماره آخرین عنصر آرایه است.

برای مثال اگر بخواهید مقدار عنصر سوم آرایه‌ی strName را برابر با ali قرار دهید باید از کد زیر استفاده کنید:

```
strName[3] = “ali”;
```

برای دسترسی به مقدار یک عنصر از ارایه هم می‌توانید از همین روش استفاده نمایید برای مثال:

```
strName = {“A”,”B”,”C” };
Dialog.Message("Table", strName[3]);
```

نکته‌ی مهم در اینجا است که اگر یکی از عناصر آرایه را تغییر دهید، بقیه عناصر تغییری نخواهد کرد. برای مثال اگر کد زیر را در برنامه به کار ببرید:

```
strName[3] = "F";
```

مقدار متغیر strName[2] هم چنان برابر با B خواهد بود.

برای اینکه بهتر فهمیدن آرایه‌ها و اینکه چگونه کار می‌کنند بهتر است پروژه ای با استفاده از آن‌ها بنویسیم.

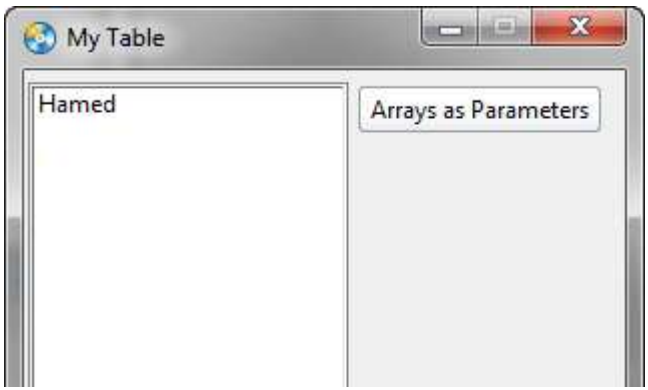
مثال: تعریف و استفاده از یک آرایه

1. نرم افزار AMS را اجرا کنید و یک پروژه جدید به نام My Table ایجاد کنید.
2. اندازه‌ی پروژه را از منوی Project>Settings… برابر با 300x300 قرار دهید.
3. یک شی ListBox و یک شی xButton به پروژه اضافه کنید و خاصیت Text را برای xButton1 را برابر با Array Elements قرار دهید.
4. روی کنترل xButton1 دو بار کلیک کنید و سپس در رویداد Script>On Click آن کد زیر را وارد کنید و با Ok تایید کنید:

۲- شماره 4

```
strFriends = {};
strFriends[1] = "Hamed";
strFriends[2] = "Javad";
strFriends[3] = "Mohsen";
strFriends[4] = "Soheyl";
strFriends[5] = "Farshid";
ListBox.DeleteItem("ListBox1", -1);
ListBox.AddItem("ListBox1", strFriends[1], "");
```

5. برنامه فشردن کلید F5 اجرا نمایید و روی دکمه‌ی Array Elements کلیک کنید. شی ListBox1 با نام Robbin پر می‌شود. (تصویر 4 – 1)



تصویر 4 - 1

بررسی: ابتدای کار یک آرایه تعریف به نام strFriends تعریف نمودیم و سپس 5 عنصر از نوع رشته به آن افزودیم.

به این ترتیب آرایه ای به طول 5 عنصر ایجاد کرده‌ایم و می‌توانیم به هر یک از عناصر آن با استفاده از اندیس آن دسترسی پیدا کنیم. برای دسترسی به یک عنصر خاص باید اندیس آن را داخل کروشه بعد از نام آرایه بیاورید. اندیس‌ها همان شماره عنصرها در آرایه می‌باشند.

```
strFriends[1] = "Hamed";
```

همان طور که برای مقدار دهی به یک عنصر باید از اندیس آن استفاده کنید برای دسترسی به مقدار آن عنصر از آرایه نیز می‌توانید از اندیس آن استفاده کنید. در این برنامه، مقدار موجود در خانه‌ی یکم را که برابر با اولین مقدار آرایه ("Robbin") است را به کاربر نمایش می‌دهید:

```
ListBox.AddItem("ListBox1", strFriends[1], "");
```

استفاده از حلقه‌ی for:

یکی از متداول‌ترین روش‌های استفاده از آرایه‌ها، به کار بردن حلقه‌ی for می‌باشد. این نوع حلقه‌ها را در فصل سوم با هم آموختیم. در مثال بعدی ملاحظه خواهید کرد که چگونه می‌توانیم از این حلقه‌ها در آرایه‌ها استفاده کنیم.

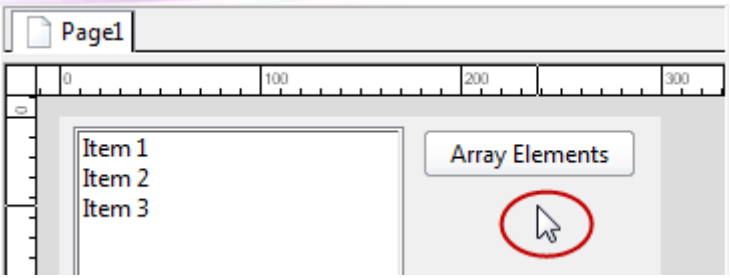
مثال: استفاده از حلقه‌ی for با آرایه‌ها

1. بر روی قسمت خالی پروژه مطابق با تصویر 4-2 دو بار کلیک کنید تا پنجره Properties مربوط به صفحه 1 پروژه نمایان شود[۳].
2. کدهای زیر را در قسمت Script>On Show بنویسید و با Ok تایید کنید:

```
strFriends = {};
strFriends[1] = "Hamed";
strFriends[2] = "Javad";
strFriends[3] = "Mohsen";
strFriends[4] = "Soheyl";
strFriends[5] = "Farshid";
ListBox.DeleteItem("ListBox1", -1);
for index , strName in pairs(strFriends) do
        ListBox.AddItem("ListBox1", strName, "");
end
```

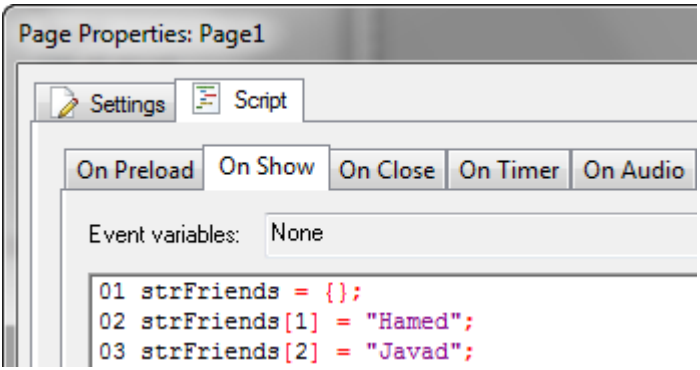
---

5. هم چنین می‌توانید با استفاده از پنجره‌ی Properties به این قسمت دسترسی پیدا کنید.



تصویر 4 - 2

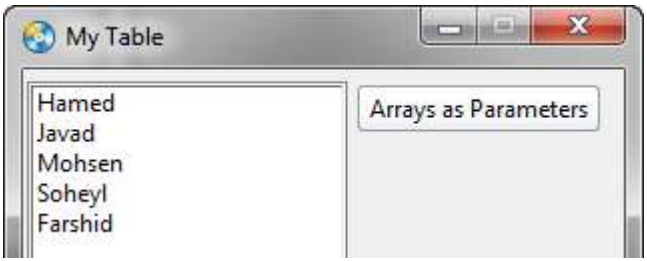
(قسمت خالی صفحه‌ی 1 پروژه)



تصویر 4 - 3

تصویر 4- 3 (محیط کد نویسی Script>On Show در پنجره‌ی Properties)

3. برنامه را اجرا کنید تا نتیجه‌ی این کد نویسی را مشاهده کنید.(تصویر 4- 4)



تصویر 4 - 4

بررسی مثال: در این مثال ما کدی نوشتم که به محض نمایش داده شدن صفحه‌ی اول پروژه اجرا شوند برای این کار می‌بایست کدهایمان را در رویداد On Show صفحه‌ی مورد نظر بنویسیم که در تصویر 4- 3 را مشاهده کردید.

کدهای مربوط به تعریف آرایه و مقدار دهی عناصر آرایه نیازی به توضیح ندارند و فقط به بررسی دستور for که در آرایه استفاده شده می‌پردازیم:

در فصل 3 مشاهده کردید که چگونه یک حلقه‌ی for در بین عناصر یک آرایه از رشته‌ها حرکت می‌کند. در این مثال هم همان مراحل را تکرار می‌کنیم. یک متغیر بعد از شمارنده‌ی index تعریف می‌کنیم(strName) و آن را برای دسترسی به تک تک عناصر آرایه مورد استفاده قرار می‌دهیم.

حلقه‌ی for از عنصر اول آرایه شروع می‌کند و تا رسیدن به آخرین عنصر در آرایه بین تمام عنصرها جا به جا می‌شود. در هر بار تکرار حلقه می‌توانید از عنصری که در متغیر کنترل کننده قرار گرفته است استفاده کنید. در این مثال این مقدار را به لیست اضافه می‌کنیم.

```
for index , strName in pairs(strFriends) do
        ListBox.AddItem("ListBox1", strName, "");
end
```

انتقال آرایه‌ها به عنوان پارامتر:

در برخی از حالت‌ها ممکن است نیاز داشته باشید یک آرایه را که دارای چندین عنصر است به عنوان یک پارامتر به یک تابع بفرستید. در مثال بعد، نحوه‌ی انجام این عمل را خواهیم دید.

مثال: انتقال آرایه‌ها به عنوان پارامتر

1. یک پروژه‌ی جدید با نام Arrays as Parameters ایجاد کنید. یک شی Listbox و یک شی xButton به پروژه اضافه کنید و خاصیت Text را برای xButton1 برابر با Arrays as Parameters قرار دهید.
2. بر روی قسمت خالی پروژه دو بار کلیک کنید و در قسمت Script>On Show پنجره‌ی Properties کد زیر را وارد کنید:

```
strFriends = {};
strFriends[1] = "Hamed";
strFriends[2] = "Javad";
strFriends[3] = "Mohsen";
strFriends[4] = "Soheyl";
```

```
strFriends[5] = "Farshid";
function AddItemsToList(arrayList)
        ListBox.DeleteItem("ListBox1", -1);
        for index , strName in pairs(strFriends) do
        ListBox.AddItem("ListBox1", strName, "");
        end
end
```

3. روی xButton1 دو بار کلیک کنید و در رویداد Script > On Click کد زیر را وارد کنید:

```
AddItemsToList(strFriends);
```

برنامه را اجرا کنید و روی دکمه‌ی Arrays as Parameters کلیک کنید. نتیجه ای مشابه با تصویر 4- 4 را مشاهده خواهید نمود.

بررسی مثال: کد وارد شده در قسمت On Show در صفحه‌ی اول پروژه تا سطر 6 مربوط به تعریف و مقدار دهی آرایه‌ی strFriends می‌باشد. در سطر 7 شما با واژه‌ی function رو به رو هستید، این واژه برای تعریف تابع AddItemsToList به کار رفته است که دارای یک پارامتر به نام arrayList می‌باشد. کد درون تابع هم مشخص است و در مثال مربوط **به استفاده از حلقه‌ی for با آرایه‌ها** آن را بررسی کردیم. و نیازی به توضیح مجدد آن نیست.

در رویداد On Click مربوط به xButton1 هم کد فراخوانی تابع را نوشتیم:

```
AddItemsToList(strFriends);
```

با این کار ما آرایه strFriends را به تابع می‌فرستیم تا دستورات درون خود را روی آن اعمال نماید.

مرتب سازی آرایه‌ها:

مرتب کردن آرایه‌ها یکی از مواردی است که در حین کار با آرایه‌ها بدان نیاز پیدا می‌کنیم. در مثال بعدی با چگونگی مرتب سازی آرایه‌ها آشنا خواهید شد.

مثال: مرتب سازی آرایه‌ها

1. یک شی xButton دیگر به پروژه‌ی Arrays as Parameters اضافه کنید و خاصیت Text آن را برابر با Sorting Arrays قرار دهید.
2. روی xButton2 دو بار کلیک کنید و در رویداد Script > On Click کد زیر را وارد کنید:

```
-- Sort the array
Table.Sort(strFriends);
-- List your friends
AddItemsToList(strFriends);
```

3. برنامه را اجرا کنید و روی دکمه‌ی Sorting Arrays کلیک کنید. مشاهده خواهید کرد که لیست اسامی موجود در آرایه به صورت الفبایی مرتب شده‌اند.

بررسی مثال: یکی از دستورات AMS که برای کار با آرایه‌ها به ما کمک می‌کند دستور Table.Sort می‌باشد که زیر مجموعه ای از دستور Table می‌باشد. دستور Table.Sort می‌تواند آرایه‌ها را به عنوان یک پارامتر دریافت نماید و سپس بر حسب نوع داده ای آرایه، آن‌ها مرتب کند. که در این مثال به علت رشته ای بودن عناصر آرایه به مرتب سازی الفبایی آن‌ها اقدام نموده است.

معکوس کردن آرایه‌ها:

گاهی اوقات در برنامه‌ها نیاز پیدا می‌کنیم که یک آرایه را به صورت معکوس یعنی از آخر به اول مرتب سازی نماییم. در AMS برای این کار دستور Table.Sort به غیر از پارامتر آرایه، می‌تواند یک تابع را نیز به عنوان پارامتر دریافت کند و بر اساس آن تابع اقدام به معکوس کردن آرایه نماید. در مثال بعدی نمونه ای از معکوس کردن آرایه‌ها را خواهیم دید.

مثال: معکوس کردن یک آرایه

1. یک شی xButton دیگر به پروژه‌ی Arrays as Parameters اضافه کنید و خاصیت Text آن را برابر با Descending Arrays قرار دهید.
2. بر روی xButton3 دو بار کلیک کنید و در رویداد Script > On Click کد زیر را وارد کنید:

```
function sorter(v1,v2)
```

```
    if (v1 > v2)then
        return true;
    else
        return false;
    end
end
-- Sort the array
Table.Sort(strFriends, sorter);
AddItemsToList(strFriends);
```

4. برنامه را اجرا کنید و روی دکمه‌ی Descending Arrays کلیک کنید. مشاهده خواهید کرد که لیست اسامی موجود در آرایه بر حسب الفبا از آخر به اول مرتب شده‌اند. (تصویر 4- 5)

تصویر 4 - 5

بررسی مثال: این کد با استفاده از تابع sorter دو مقدار را با هم مقایسه می‌کند و هر کدام که بزرگ‌تر باشد مقدار true یا همان صحیح را برمی‌گرداند و در غیر این صورت مقدار false یا همان غلط را بر می‌گرداند. سپس ما تابع sorter را در دستور Table.Sort قرار داده‌ایم که این امر باعث می‌شود آرایه‌ی ما از آخر به اول مرتب شود چون در اینجا مقدار آرایه رشته است به همین دلیل مقدار بزرگ‌تر یعنی هر حرفی ردیف الفبایی آن بالاتر باشد. به عبارتی دیگر این تابع باعث می‌شود دستور مرتب سازی به صورت معکوس انجام گیرد.

اگر خواستید آرایه ای را معکوس کنید بدون اینکه ردیف‌های آن جا به جا شوند می‌توانید از حلقه‌ی for استفاده کنید و آرایه را از آخر به اول مرتب کنید.

برای مثال به کد زیر دقت کنید:

```
ListBox.DeleteItem("ListBox1", -1);
for i = Table.Count(strFriends), 1, -1 do
ListBox.AddItem("ListBox1", strFriends[i], "");
end
```

در این کد ما از حلقه‌ی معکوس کمک می‌گیریم که در فصل قبل با آن آشنا شدیم. مقدار شمارنده‌ی i را برابر با تعداد کل عناصر آرایه می‌دهیم؛ برای گرفتن تعداد کل عنصر های آرایه از دستور Table.Count استفاده می‌کنیم. سپس کوچک‌ترین عنصری که حلقه تا آن ادامه خواهد داشت را تعیین می‌کنیم که 1 می‌شود یعنی حلقه از تعداد کل خود باید تا عدد 1 به صورت معکوس حرکت کند. هم چنین لازم است به حلقه بگوییم که در هر بار تکرارش چند واحد از تعداد کل کم کند که بر حسب نیازمان 1- را تعیین می‌کنیم یعنی اینکه هر بار که حلقه تکرار می‌شود یک واحد از تعداد کل کم می‌کند و به سمت کوچک‌ترین عنصر حرکت می‌کند.

فصل پنجم: متغیرهای پیش فرض و متغیرهای رویدادی

تا کنون با متغیرهای رشته ای و عددی آشنا شده‌اید. در این فصل با دو نوع دیگر از متغیرها در AMS آشنا خواهید شد که برخی از آن‌ها با شروع برنامه تا پایان آن قابل استفاده می‌باشند و برخی دیگر متغیرهایی هستند که در هنگام وقوع یک رویداد همانند فشردن یک کلید از صفحه کلید قابل استفاده می‌باشند.

در این فصل با موارد زیر آشنا می‌شویم:

- متغیرهای سراسری
- رویدادها
- متغیرهای رویدادی

متغیرهای سراسری (Global variables):

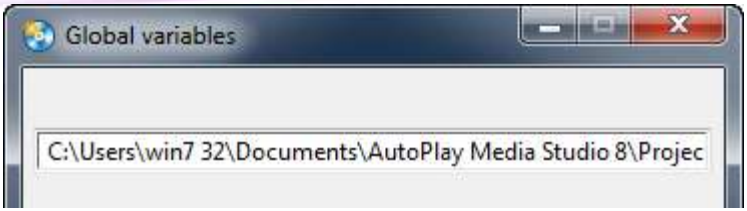
در AMS متغیرهایی وجود دارند که مقادیر آن‌ها به صورت قراردادی از قبل تعریف شده‌اند و می‌توان آن‌ها را به نوعی شبه ثابت نامید این متغیرها را متغیرهای سراسری می‌نامند زیرا با شروع برنامه این متغیرها نیز قابل استفاده می‌باشند. برای این که بیشتر با این متغیرها آشنا شوید به مثال زیر دقت کنید:

مثال: متغیرهای سراسری

1. یک پروژه جدید ایجاد کنید و نام آن را Global variables قرار دهید، هم چنین اندازه آن را برابر با 400x200 قرار دهید(Project > Settings).
2. از منوی Objects یک شی Input به پروژه اضافه کنید.
3. بر روی قسمت خالی پروژه دو بار کلیک کنید و در قسمت Script>On Show کد زیر را وارد نمایید و با Ok کد را تایید کنید:

```
Input.SetText("Input1", _SourceFolder);
```

4. پروژه را اجرا کنید، نتیجه ای همانند تصویر 4- 6 را مشاهده خواهید کرد.



تصویر 5 - 1

بررسی مثال: در این مثال ما فقط یک سطر کد نوشتیم که آن هم کدی است که متنی را در ورودی متن برنامه یعنی Input1 وارد می‌کند، نکته مهم همین متن ورودی است که به صورت یک متغیر نوشته شده است یعنی SourceFolder_ و درون علامت "" نیست و جالب‌تر آن که ما در حین کد نویسی هیچ مقداری را به این متغیر نسبت نداده‌ایم پس متن موجود در Input1 از کجا آمده است؟ جواب ساده است این متغیرها به صورت از پیش تعریف شده در برنامه موجود هستند و توسط برنامه نویس یا اتوران نویس تعیین نمی‌گردد. به این گونه متغیرها متغیر سراسری یا Global variables می‌گویند. متغیر سراسری SourceFolder_ برای گرفتن مسیر فایل اجرایی به کار می‌رود یعنی همان مسیری که پروژه از آن جا اجرا می‌گردد.

تا اینجا با متغیر سراسری آشنا شدید و توانستید یکی از آن‌ها را به کار بگیرید اما تعداد آن‌ها فقط به یک یا دو مورد ختم نمی‌شود، برای آشنایی با دیگر متغیرهای سراسری به ادامه مطلب توجه کنید.

AutoTabOrder_: در حالت عادی وقتی از اشیایی همانند Input، ListBox، RadioButon در پروژه‌ی خود استفاده کنید در هنگام اجرای پروژه می‌توانید با کلید Tab بین آن اشیا جا به جا شوید. AutoTabOrder_ یک متغیر از نوع بولین می‌باشد که اگر آن را برابر با false قرار دهید(AutoTabOrder=false;_) کلید Tab به دیگر اشیا پرشی انجام نخواهد داد.

CommandLineArgs_: این متغیر می‌تواند مسیر فایلی را که شما آن را با پروژه خود اجرا می‌کنید، در اختیار شما قرار دهید. دقت داشته باشید که این متغیر به صورت آرایه می‌باشد و می‌بایست شماره عنصر را هم در کنار متغیر بنویسید. همانند: CommandLineArgs[1]_

_DesktopFolder: این متغیر مسیر میز کاری سیستم عامل کاربر فعلی را برمی‌گرداند.

_DesktopFolderCommon: این متغیر مسیر میز کاری کاربران عمومی را برمی‌گرداند.

_DoFlashCheck: متغیری است از نوع بولین برای زمانی که شما پیش نیاز فلش پلیر را از منوی Project>Dependencies فعال کرده باشید. اگر مقدار این متغیر برابر با true باشد برنامه بررسی می‌کند که آیا فلش پلیر موجود است یا نه ولی اگر مقدار آن false باشد بررسی برای نصب فلش انجام نمی‌گردد. می‌توانید از این تابع در Global Function استفاده کنید.

_IR_ProductID: این متغیر رشته ای حاوی ورژن نرم افزار سازنده برنامه حاضر را باز می‌گرداند .برای مثال اگر شما با AutoPlay Media Studio 8 نرم افزارتان را ساخته باشید مقدار AMS8 داخل آن و اگر با نسخه 7 یا 7.5 ساخته باشید رشته حاوی عبارت AMS70 می‌باشد.

_NoExitScriptOnPageJump: متغیری از نوع بولین است که اگر مقدار آن را برابر با true قرار دهیم، برنامه کد بعد از پرش به صفحه‌ی قبلی و بعدی را اجرا خواهد نمود در حالت پیش‌فرض این متغیر برابر با false می‌باشد.

مثال:

```
_NoExitScriptOnPageJump = true;
Page.Navigate(PAGE_NEXT);
Dialog.Message("Notice", "Your message here.");
```

_ProgramFilesFolder: این متغیر مسیر پوشه‌ی Program Files را بر می‌گرداند.

_ShowIntroVideo: متغیری از نوع بولین که اگر مقدار آن را برابر با true قرار دهیم برنامه، فایل ویدئویی را که از منوی Project>Startup Movie به عنوان آغازگر برنامه تعیین کرده‌ایم اجرا خواهد کرد.

_SoundInitialized: تغیری از نوع بولین که اگر مقدار آن مساوی با true باشد نشان دهنده‌ی این است که کارت صوتی به خوبی کار می‌کند.

_SourceDrive: به وسیله‌ی این متغیر می‌توانیم نام درایوی را که برنامه در آن قرار دارد به دست آوریم.

_SourceFilename: این متغیر نام فایل اجرایی را برمی‌گرداند. مثل:

autorun.exe

_SourceFolder: با این متغیر می‌توانیم مسیر اجرای فایل اجرایی خود را به دست آوریم.

_SystemFolder: این متغیر مسیر پوشه سیستمی ویندوز را برمی‌گرداند.(C:\Windows\System32)

_tblErrorMessages: یک متغیر آرایه ای است که تمام خطاهای موجود در برنامه در آن موجود می‌باشد. مثال:

_tblErrorMessages[1000]

_TempFolder: متغیری که مسیر پوشه‌ی Temp را برمی‌گرداند(یک پوشه برای مصرف عموم برنامه‌ها به صورت موقتی).

_WindowsFolder: متغیری که مسیر پوشه ویندوز شما را برمی‌گرداند. مثل:

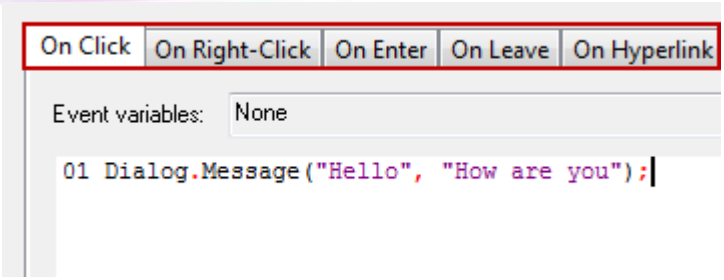
C:\Windows

نکته: این متغیرها همان طور که از نام متغیر مشخص است قابل تغییرند و می‌توان مقدار آن‌ها را در طول اجرای برنامه تغییر داد.

نکته: متغیرهای سراسری در ابتدای خود یک خط زیرین دارند همانند: _SourceFolder یا _TempFolder

رویدادها و متغیرهای رویدادی:

رویداد: زمانی که با یک برنامه کار می‌کنید اتفاقاتی رخ می‌دهد که باعث انجام کارهایی در آن برنامه می‌شوند این اتفاق‌ها می‌توانند شامل حرکت نشانگر ماوس، کلیک کردن بر روی یک شی، راست کلیک بر روی یک شی، فشردن یک کلید از صفحه کلید و ... باشد؛ هر یک از این اتفاقات را یک رویداد می‌نامند. برای مثال زمانی که شما بر روی یک دکمه کلیک می‌کنید رویداد کلیک رخ می‌دهد.(تصویر 4 - 7)



تصویر 5 - 2

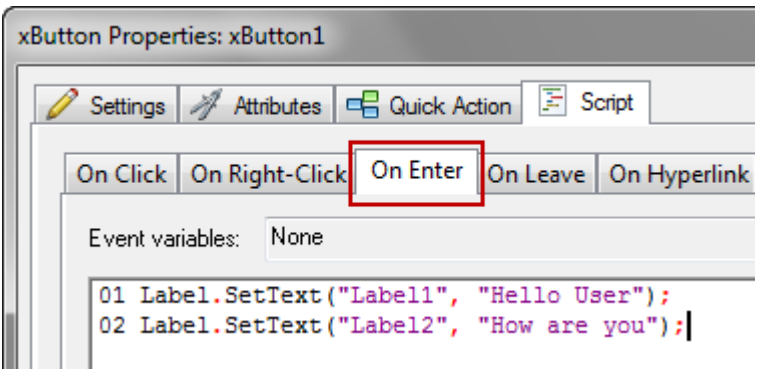
مثال: رویدادها

1. یک پروژه جدید به نام Event ایجاد کنید .
2. اندازه پروژه را برابر 250x100 قرار دهید، سپس یک شی xButton و دو شی Label به پروژه اضافه کنید.
3. خاصیت Text را برای xButton1 برابر با Change Event قرار دهید سپس روی آن دو بار کلیک کنید و کد زیر را در قسمت Script > On Enter پنجره‌ی Properties بنویسید:

```
Label.SetText("Label1", "Hello User");
Label.SetText("Label2", "How are you");
```

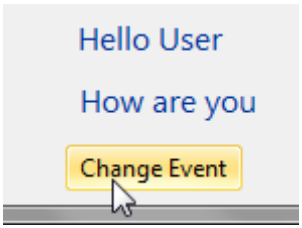
حال به قسمت سربرگ On Leave در همین پنجره بروید و کد زیر را در قسمت کد نویسی آن وارد کنید و با Ok کدها را تایید کنید(تصویر 4 - 8):

```
Label.SetText("Label1", "See you Later");
Label.SetText("Label2", "Good bye");
```

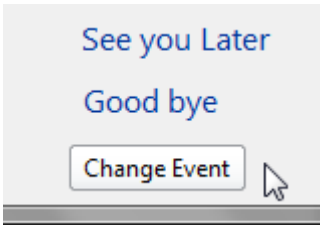


تصویر 5 - 3

4. برنامه را اجرا کنید، نشانگر ماوس را روی دکمه‌ی Change Event قرار دهید و سپس نشانگر از روی آن بر دارید. باید نتیجه‌ای مشابه به تصویر 4- 9 و 4- 10 داشته باشید:



تصویر 5 - 4



تصویر 5 - 5

بررسی مثال: وقتی روی xButton1 دو بار کلیک کردیم و به پنجره Properties و از آنجا به سربرگ Script رفتیم در آن جا با 5 سربرگ دیگر که زیر مجموعه‌ی سربرگ Script هستند مواجه شدیم که هر کدام یک رویداد محسوب می‌شوند. در رویداد On Enter (وارد شدن به محیط شی) کدی را نوشتیم که متنی را در Label1 و Label2 قرار دهد و سپس در رویداد On Leave (خارج شدن از محیط شی) کدی نوشتیم که متن دیگری را در Label1 و Label2 قرار دهد.

بدین ترتیب توانستیم در دو رویداد دیگر به غیر از رویداد کلیک دستوراتی را اجرا کنیم.

به دلیل آنکه رویدادها و متغیرهای رویداد در ارتباط با هم دیگر عمل می‌کنند بقیه‌ی رویدادها را نیز پس از آشنایی با **متغیرهای رویدادی** به صورت ترکیبی مرور خواهیم نمود.

متغیرهای رویدادی: برخی از رویدادها دارای متغیرهای از پیش تعریف شده ای هستند که در هنگام وقوع آن رویداد می‌توان از آن متغیرها استفاده نمود. اگر رویدادی که دارای چنین متغیری باشد در سربرگ Script در قسمتی به نام Event Variable نام متغیرهایش ذکر

خواهد شد و اگر آن رویداد متغیری نداشته باشد در کادر مربوطه عبارت None نمایش داده خواهد شد. برای روشن‌تر شدن موضوع به مثال زیر دقت کنید.

مثال: متغیرهای رویدادی

1. پروژه جدیدی به نام Event Variable ایجاد کنید و اندازه آن را برابر با 220x50 قرار دهید.
2. یک شی Input به پروژه‌ی خود اضافه کنید و سپس روی آن دو بار کلیک کرده و کد زیر را در رویداد Script>On Key وارد کنید و با Ok کدها تایید کنید:

```
if e_Key == 13 then
Dialog.Message("Event Variable", "Key Name is Enter");
end
```

3. کد زیر را نیز پس از دو بار کلیک در قسمت خالی صفحه‌ی پروژه در قسمت Script>On Show بنویسید و با Ok تایید کنید:

```
Page.SetFocus("Input1");
```

4. پروژه را اجرا کنید و سپس کلید Enter را در محیط برنامه فشار دهید. نتیجه ای مشابه با تصویر 4- 11 مشاهده خواهید کرد.

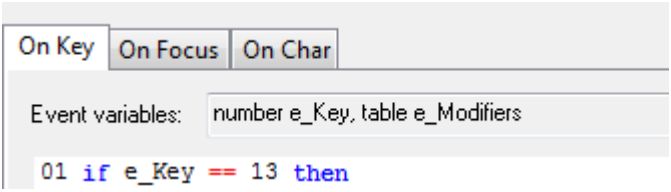
دقت داشته باشید که مکان نمای چشمک زن باید در قسمت Input(ورودی متن) باشد.

تصویر 5 - 6

بررسی مثال: زمانی که روی Input1 دو بار کلیک کردیم تا به قسمت کد نویسی Script>On Key برویم. وقتی یک شی دارای رویداد On Key باشد به این معنا است که

در این رویداد می‌توان کلیدهای فشرده از صفحه کلید را دریافت کرده و با توجه به آن‌ها دستوراتی را تعریف نمود.

در قسمت Event Variables با دو متغیر رو به رو شدیم که عبارت بودند از number e_Key, table e_Modifiers ؛ که در ابتدا نوع این متغیرها نوشته شده است و سپس متغیرها با یک علامت , از هم جدا شده‌اند: متغیر e_Key از نوع number (عددی) و متغیر e_Modifiers از نوع table (جدول یا آرایه) می‌باشد. این دو متغیر در این رویداد یعنی On Key قابل استفاده می‌باشند.(تصویر 4 -12)



تصویر 5 - 7

هر کلید در برنامه یک کد خاصی دارد که با آن کد می‌توان تشخیص داد که کدام کلید فشرده شده است متغیر e_Key نیز این کد را در خود نگه می‌دارد تا اگر نیاز شد مورد استفاده قرار دهد.

برای مثال کد کلید Enter عدد 13 می‌باشد. ما کدی را نوشتیم که با آن به برنامه بگوییم اگر کلید فشرده شده برابر با Enter بود کادر پیغامی را کاربر نشان دهد.

```
if e_Key == 13 then
Dialog.Message("Event Variable", "Key Name is Enter");
end
```

توجه: برای آشنایی با کد تمام کلیدها می‌توانید به پیوست کتاب مراجعه نمایید.

در قسمت On Show کدی نوشتیم که به محض نمایش داده شدن صفحه شی Input1 در برنامه مورد انتخاب قرار گیرد این کار بدان دلیل است که اگر این شی انتخاب نشود با فشردن کلید Enter دستورات اجرا نخواهند شد مگر اینکه کاربر به صورت دستی روی آن شی کلیک کند.

```
Page.SetFocus("Input1");
```

متغیر دیگری که در رویداد On Key قابل توجه می‌باشد متغیر e_Modifiers می‌باشد که می‌تواند کلیدهای ترکیبی فشرده شده را در خود نگهداری کند برای مثال کد زیر کلید Ctrl+Enter را بررسی می‌کند که اگر فشرده شده باشند کادر پیغامی به کاربر نمایش داده خواهد شد.

```
if e_Modifiers.ctrl == true and e_Key == 13 then
Dialog.Message("Event Variable", "Ctrl+Enter");
end
```

رویدادها و متغیرهای رویدادی موجود در آن‌ها:

همان طور که گفتیم برخی از رویدادها دارای متغیرهای از پیش تعریف شده ای می‌باشند. در این بخش از کتاب با رویدادهای مختلف و متغیرهای موجود در آن‌ها آشنا خواهیم شد.

- ✓ **رویدادها و متغیرهای رویدادی در منوی Project>Actions...**

On Startup: این رویداد در هنگام شروع برنامه و قبل از نمایش صفحات فعال می‌گردد و می‌توان دستوراتی را در آن به اجرا در آورد. این رویداد متغیر از پیش تعریف شده ای ندارد.

On Shutdown: در صورتی که می‌خواهید دستوراتی قبل از بسته شدن کامل برنامه اجرا شوند می‌بایست آن دستورات را در این رویداد وارد نمایید. این رویداد متغیر از پیش تعریف شده ای ندارد.

On Size: این رویداد زمانی به کار می‌آید که اندازه‌ی پنجره‌ی پروژه‌ی ما تغییر کند. این رویداد دارای متغیرهای رویدادی زیر می‌باشد:

e_WindowWidth: متغیری که پهنای پنجره‌ی پروژه را در خود نگه می‌دارد.

e_WindowHeight: متغیری که درازای پنجره‌ی پروژه را در خود نگه می‌دارد.

e_PageWidth: متغیری که پهنای صفحه‌ی پروژه را در خود نگه می‌دارد.

e_PageHeight: متغیری که درازای صفحه‌ی پروژه را در خود نگه می‌دارد.

e_Type: متغیری که نوع اندازه پنجره را بر می‌گرداند و سه حالت را بر می‌گرداند: 1. زمانی که پنجره در حالت اندازه‌ی معمولی قرار دارد عدد 0 را بر می‌گرداند. 2. زمانی که پنجره در

حالت حداکثر اندازه‌ی خود قرار دارد عدد 2 را بر می‌گرداند. 3 زمانی که پنجره حداقل اندازه را دارد عدد 1 را بر می‌گرداند.

On Menu: در این رویداد می‌توانید کدهای مربوط به منوی برنامه را بنویسید که در فصل مربوط به منوها به طور مفصل آموزش داده شده است.

این رویداد دارای دو متغیر e_ID، e_ItemInfo می‌باشد که e_ID از نوع عددی و e_ItemInfo از نوع آرایه می‌باشد.

برای آشنایی با این قسمت ابتدا از منوی Project>Menu Bar گزینه‌ی Show Menu Bar را علامت بزنید و Ok را کلیک کنید و سپس کد زیر را در رویداد On Menu بنویسید و سپس برنامه اجرا کنید و یکی از منوها را کلیک کنید تا نتیجه‌ی کد را مشاهده نمایید(تصویر 5 - 7)

```
Dialog.Message("Menu", "Text: "..e_ItemInfo.Text.."\r\n".."ID: "..e_ID);
```

تصویر 5 - 8

نکته: علامت "\r\n" برای ایجاد یک سطر به کار می‌رود.

✓ **رویدادها و متغیرهای رویدادی در منوی Page>Properties>Script**

در هر صفحه از پروژه رویدادهایی وجود دارد که هم با دو بار کلیک کردن روی قسمت خالی صفحه و هم چنین با رفتن به منوی Page>Properties>Script قابل دسترسی هستند.

On Preload: این رویداد قبل از نمایش صفحه‌ی مورد نظر فعال می‌گردد و می‌توان دستوراتی را در آن به اجرا در آورد. این رویداد متغیر از پیش تعریف شده ای ندارد.

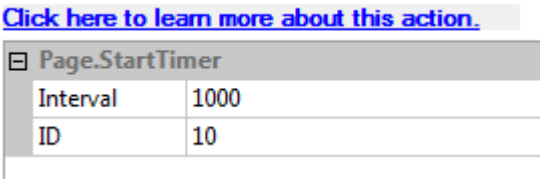
On Show: این رویداد همزمان با نمایش صفحه‌ی مورد نظر فعال می‌گردد و می‌توان دستوراتی را در آن به اجرا در آورد و متغیر از پیش تعریف شده‌ای ندارد.

On Close: این رویداد هنگام خارج شدن از صفحه ای که در آن هستیم فعال می‌گردد و متغیر از پیش تعریف شده‌ای ندارد.

On Timer: این رویداد دارای یک متغیر به نام e_ID می‌باشد. گاهی اوقات نیاز است که پروژه‌ی ما هر چند دقیقه یا هر چند ثانیه یک بار مواردی را بررسی کند مثلاً ساعت سیستم را بررسی کند و آن را در یک شی Label نمایش دهد. برای این کار ما باید از زمان شمار استفاده کنیم. زمان شمارها بر حسب هزارم ثانیه تعریف می‌گردند. برای مثال وقتی ما کدی همانند کد زیر در رویداد On Show می‌نویسیم:

```
Page.StartTimer(1000, 10);
```

اگر روی کد بالا دو بار کلیک کنید می‌توانید خصوصیات آن را مشاهده نمایید(تصویر 5- 8):



تصویر 5 - 9

در قسمت Interval عدد 1000 نمایش دهنده‌ی فاصله‌ی زمانی است که برنامه کدهای رویداد On Timer را بر حسب ID (که همان عدد 10 می‌باشد) مورد بررسی و اجرا قرار می‌دهد.

بر فرض مثال اگر در رویداد On Timer کد زیر را داشته باشیم:

```
if e_ID == 10 then
Time = System.GetTime(TIME_FMT_MIL);
Label.SetText("Label1", Time);
end
```

برنامه هر 1 ثانیه کدهای قسمت On Timer را بررسی می‌کند و اگر متغیر e_ID را برابر با 10 ببیند کدهای درون if مربوطه را اجرا خواهد کرد. پس در نتیجه مقدار متغیر رویدادی e_ID همان مقداری است که در قسمت ID برای کد Page.StartTimer تعریف نمودیم.

در رویداد On Timer می‌توان شرط‌های زیادی را مورد بررسی قرار داد و محدود به یک شرط نیست.

On Audio: این رویداد زمانی فعال است که ما در پروژه‌ی خود از فایل‌های صوتی استفاده کنیم و آن‌ها را پخش، قطع یا مکث کنیم. متغیرهای رویدادی این قسمت عبارتند از: e_Channel و e_State.

e_Channel: این متغیر کانال‌های مختلف پخش را بررسی و آن‌ها را شناسایی می‌کند.

در AMS می‌توان همزمان از 7 کانال صدا را پخش نمود. پس ما 7 کانال صوتی در اختیار داریم که هر کدام از آن‌ها دارای نام و شماره خاصی هستند:

نام ثابت	شماره	توضیح
CHANNEL_BACKGROUND	5	صدای پس زمینه
CHANNEL_EFFECTS	0	جلوه های صوتی
CHANNEL_NARRATION	6	صدای دوم
CHANNEL_USER1	1	کانال صوتی 1
CHANNEL_USER2	2	کانال صوتی 2
CHANNEL_USER3	3	کانال صوتی 3
CHANNEL_USER4	4	کانال صوتی 4
CHANNEL_ALL	-3	تمامی کانال‌ها

e_State: این متغیر حالت‌های پخش صدا را بررسی می‌کند که در جدول زیر آمده‌اند:

مقدار	نوع	توضیح
Finish	رشته	پخش به پایان رسیده است.
Pause	رشته	پخش فایل صوتی با مکث مواجه شده است.

فایل صوتی پخش شد.	رشته	Play
پخش فایل صوتی متوقف شده است.	رشته	Stop

برای روشن شدن موضوع مثالی می‌زنیم:

در رویداد On Audio می‌توان کدی نوشت که اگر صدای پس زمینه تمام شد برنامه یک پیغام نمایش دهد.

```
if e_Channel == 5 and e_State "Finish" then
Dialog.Message("Notice", "Your message here.");
end
```

On Size: توضیح این رویدادها در قسمت **رویدادها و متغیرهای رویدادی در منوی Project>Actions...** بیان شد و میان این دو هیچ تفاوتی وجود ندارد مگر اینکه متغیرهای On Size در این قسمت زمانی بررسی می‌شوند که در صفحه مورد نظر هستیم در حالی که متغیرهای On Size قسمت **Project>Actions...** مربوط به کل پروژه ما می‌شوند.

On Menu: توضیح این رویدادها در قسمت **رویدادها و متغیرهای رویدادی در منوی Project>Actions...** بیان شد و میان این دو هیچ تفاوتی وجود ندارد مگر در محل اجرای کدها.

On Key: این رویداد دو متغیر یعنی e_Key و e_Modifiers را دارا است که با متغیر e_Key آشنا شدید. اما متغیر e_Modifiers: این متغیر آرایه ای دارای سه حالت دارد و نتیجه ای بولین در پی دارد: e_Modifiers.shift: یعنی کلید shift پایین است، e_Modifiers.ctrl: یعنی کلید ctrl پایین است و e_Modifiers.alt: که به معنی پایین بودن کلید alt می‌باشد.

مثال:

```
if e_Modifiers.shif == true then
Dialog.Message("Notice", "Your message here.");
```

```
end
```

On Mouse Button: این رویداد دارای سه متغیر رویدادی از پیش تعریف شده است: e_Type، e_X و e_Y که هر سه عددی می‌باشند. متغیر e_Type دکمه‌ی فشرده شده از ماوس را در خود نگهداری می‌کند، متغیر e_X حرکت افقی ماوس را در خود ذخیره می‌سازد و متغیر e_Y حرکت عمودی ماوس را در خود نگه می‌دارد.

متغیر e_Type چهار حالت دارد :

حالت کلیک ماوس	شماره
کلیک چپ ماوس فشرده شده است	0
کلیک چپ ماوس رها است	1
کلیک راست ماوس فشرده شده است	2
کلیک راست ماوس رها شده است	3

برای مثال کد زیر را در این رویداد بنویسید:

```
if e_Type == 2 then
Dialog.Message("Notice", "Your message here.");
end
```

On Mouse Move: این رویداد زمانی رخ می‌دهد که نشانگر ماوس را حرکت می‌دهیم و دارای دو متغیر e_X و e_Y می‌باشد که توضیح آن‌ها در رویداد On Mouse Button بیان شد.

برای مثال این کد را در رویداد مورد نظر بنویسید:

```
if e_Y > 10 and  e_X < 50 then
Dialog.Message("Notice", "Your message here.");
end
```

On Mouse Wheel: این رویداد مربوط به چرخ ماوس یا همان Middle Button می‌باشد و دارای چهار متغیر e_Flags از نوع آرایه، e_Delta، e_X و e_Y که از نوع

عددی می‌باشند. با دو متغیر e_X و e_Y آشنا شده‌اید پس ما فقط به توضیح دو متغیر e_Flags و e_Delta می‌پردازیم:

متغیر e_Flags: متغیری است از نوع آرایه می‌تواند 5 حالت را ارزیابی نماید و نتیجه‌ی هر یک را به صورت بولین یعنی صحیح یا غلط برگرداند. این 5 حالت عبارتند از: e_Flags.ctrl(زمانی که کلید Ctrl را پایین نگه داشته و سپس چرخ ماوس را حرکت می‌دهید)، e_Flags.LButton(زمانی که کلیک چپ ماوس را پایین نگه داشته و سپس چرخ ماوس را حرکت می‌دهید)، e_Flags.MButton(زمانی که Middle Button ماوس را پایین نگه داشته و سپس چرخ ماوس را حرکت می‌دهید)، e_Flags.RButton(زمانی که کلیک راست ماوس را پایین نگه داشته و سپس چرخ ماوس را حرکت می‌دهید).

برای مثال کد زیر را در این رویداد بنویسید و سپس پروژه را اجرا کرده و در حالی که کلید Ctrl را از صفحه کلید پایین نگه داشته‌اید چرخ ماوس را حرکت دهید:

```
if e_Flags.ctrl == true  then
Dialog.Message("Notice", "Your message here.");
end
```

متغیر e_Delta: متغیری است عددی، که در زمان حرکت چرخ ماوس به طرف جلو عدد 120 را بر می‌گرداند و هنگام حرکت به عقب عدد 120- را بر می‌گرداند.

رویدادها و متغیرهای رویدادی موجود در اشیا

محیط پروژه‌ی ما از تعدادی شی تشکیل شده‌اند همانند: شی Button، شی Input و ... که هر کدام از آن‌ها دارای رویدادهایی می‌باشند و برخی از آن رویدادها نیز دارای متغیرهایی می‌باشند که در این قسمت با رویدادها و متغیرهای موجود اشیا آشنا خواهید شد.

نکته: با دو بار کلیک کردن روی اشیا و رفتن به سربرگ Script رویدادهای اشیا رو می‌توانید مشاهده کنید.

On Click: رویدادی است که هنگام کلیک روی شی مورد نظر رخ می‌دهد این رویداد در هر کدام از اشیا که موجود باشد متغیر رویدادی ندارد.

On Right-Click: رویدادی است که هنگام راست کلیک روی شی مورد نظر رخ می‌دهد این رویداد در هر کدام از اشیا که موجود باشد متغیر رویدادی ندارد.

On Enter: رویدادی است که هنگام رفتن نشانگر ماوس روی شی مورد نظر رخ می‌دهد این رویداد در هر کدام از اشیا که موجود باشد متغیر رویدادی ندارد.

On Leave: رویدادی است که هنگام کنار کشیدن نشانگر ماوس از روی شی مورد نظر رخ می‌دهد این رویداد در هر کدام از اشیا که موجود باشد متغیر رویدادی ندارد.

On Play: رویدادی است که هنگام پخش شدن یک فایل ویدئویی در شی Video یا شی Media Player رخ می‌دهد این رویداد در هر کدام از اشیا که موجود باشد متغیر رویدادی ندارد.

On Pause: رویدادی است که هنگام مکث کردن یک فایل ویدئویی در شی Video یا شی Media Player رخ می‌دهد این رویداد در هر کدام از اشیا که موجود باشد متغیر رویدادی ندارد.

On Stop: رویدادی است که هنگام متوقف یک فایل ویدئویی در شی Video یا شی Media Player رخ می‌دهد این رویداد در هر کدام از اشیا که موجود باشد متغیر رویدادی ندارد.

On Finish: رویدادی است که هنگام پایان یافتن یک فایل ویدئویی در شی Video یا شی Media Player رخ می‌دهد این رویداد در هر کدام از اشیا که موجود باشد متغیر رویدادی ندارد.

On Mouse Button Down: یکی از رویدادهای شی QuickTime که هنگام فشردن شدن کلیک چپ ماوس رخ می‌دهید و دارای متغیرهای رویدادی زیر می‌باشد:

e_Button: متغیری است عددی که سه حالت دارد: عدد 1 نشان دهنده‌ی آن است که کلیک چپ ماوس فشرده شده است، عدد 2 نشان دهنده‌ی آن است که کلیک راست ماوس فشرده شده است، عدد 3 نشان دهنده‌ی فشرده شدن Middle Button می‌باشد.

e_Modifiers: که با آن آشنا شدید.

e_XObject: مختصات افقی ماوس بر روی شی QuickTime را بر می‌گرداند.

e_YObject: مختصات عمودی ماوس بر روی شی QuickTime را بر می‌گرداند.

e_X: مختصات افقی ماوس را بر روی صفحه را بر می‌گرداند.

e_Y: مختصات عمودی ماوس را بر روی صفحه را بر می‌گرداند.

On Mouse Button Up: یکی از رویدادهای شی QuickTime که هنگام رها شدن کلیک چپ ماوس رخ می‌دهید و دارای متغیرهای رویدادی e_Button، e_Modifiers، e_XObject، e_YObject، e_X، e_Y می‌باشد که با آن‌ها آشنا هستید.

On Mouse Button Move: یکی از رویدادهای شی QuickTime که هنگام رها شدن کلیک چپ ماوس رخ می‌دهید و دارای متغیرهای e_Button، e_Modifiers، e_XObject، e_YObject، e_X، e_Y می‌باشد که با آن‌ها آشنا هستید.

On Rate Change: این رویداد زمانی رخ می‌دهد که سرعت پخش فایل ویدئو در شی QuickTime تغیر می‌کند. و داری متغیری عددی با نام e_Rate می‌باشد که سرعت پخش را بر می‌گرداند.

On Movie End: این رویداد زمانی رخ می‌دهد که پخش فایل ویدئویی در شی QuickTime به پایان برسد. این رویداد متغیر رویدادی ندارد.

On Error: این رویداد زمانی رخ می‌دهد که پخش در شی QuickTimeخطایی رخ بدهد. این رویداد دو متغیر رویدادی عددی با نام‌های e_ErrorCode و e_ErrorOrigin دارد. متغیر e_ErrorOrigin سه مقدار را بر می‌گرداند: عدد 0 که نشان دهنده‌ی خطا در ActiveX شی QuickTime می‌باشد. عدد 1 نشان دهنده‌ی خطا در فایل ویدئویی می‌باشد و عدد 2 نشان دهنده‌ی خطا در خود شی QuickTime می‌باشد.

مثال: فرض کنید کاربری یک فایل ویدئویی را به با آدرسی نادرست به شی QuickTime فراخوانی می‌کند اگر شما در قسمت رویداد On Error کد زیر را بنویسید می‌توانید با توجه به شماره ای که برنامه به کاربر اعلام می‌دارد به کاربر بگویید که با چه خطایی مواجه شده است.

```
Dialog.Message("Error", "Error Code: "..e_ErrorCode, MB_OK,
MB_ICONEXCLAMATION);
```

این کد شماره خطا را در یک کادر پیغام به کاربر اعلام خواهد کرد و شما می‌توانید با مراجعه به قسمت "QuickTime Error Codes " در راهنمای برنامه شماره خطا را یافته و کاربر را راهنمایی کنید.

On FSCommand: این رویداد مخصوص شی Flash می‌باشد و شامل دو متغیر رویدادی e_FSCommand و e_FSArgs می‌باشد.

e_FSCommand: متغیری رشته ای است که از فایل Flash دریافت می‌شود و با توجه به مقدار آن شرط‌های تعریف شده اجرا می‌گردند.

نکته: از FSCommand برای ارتباط بین AMS و فایل فلش استفاده می‌گردد.

e_FSArgs: متغیری رشته ای است که از فایل Flash دریافت می‌شود و با توجه به مقدار آن شرط‌های تعریف شده اجرا می‌گردند.

مثال: فرض کنید یک فایل فلش داریم و در آن دکمه ای با عنوان خروج قرار داده‌ایم و fscommand آن را برابر با Exit نوشته‌ایم با کد زیر می‌توانیم fscommand مورد نظر را به دست آورده و به برنامه دستور خروج بدهیم:

```
if e_FSCommand == "Exit" then
Application.Exit(0);
end
```

On FlashCall: این رویداد نیز مخصوص شی Flash می‌باشد و شامل یک متغیر رویدادی به نام e_FlashCall می‌باشد.

e_FlashCall: متغیری است رشته ای با فرمت XML و قابل استفاده با دستور XML موجود در AMS می‌باشد. برای کار با این قسمت می‌بایست از کد نویسی فلش نیز اطلاعات داشته باشید چرا این رویداد زمانی رخ می‌دهد که توابع فلش اجرا می‌گردند.

On Slide Changed: رویدادی است که هنگام عوض شدن عکس‌های اسلاید در شی Slide Show رخ می‌دهد و شامل متغیرهای رویدادی زیر می‌باشد:

e_Index: اندیس یا شماره اسلاید در حال نمایش را بر می‌گرداند.

e_FilePath: متغیری رشته ای است که مسیر فایل عکسی را که در حال حاضر در شی Slide Show نمایش داده می‌شود برمی‌گرداند.

On Navigate: این رویداد که متعلق به شی Web می‌باشد زمانی رخ می‌دهد که لینک یا آدرس شی web تغییر کند و دارای یک متغیر رویدادی با نام e_URL می‌باشد.

e_URL: متغیری است رشته ای که حاوی آدرس صفحه ای است که شی web در حال رفتن به آن می‌باشد.

On Loaded: این رویداد زمانی به وقوع می‌پیوندد که صفحه‌ی مورد نظر در شی Web فراخوانی شده باشد و دارای یک متغیر رویدادی با نام e_URL می‌باشد.

e_URL: متغیری است رشته ای که حاوی آدرس صفحه ای است که شی Web آن را فراخوانی کرده است.

On Hyperlink: این رویداد مختص شی xButton است و زمانی فعال می‌گردد که با دو بار کلیک کردن روی شی xButton خاصیت EnableMarkup را علامت گذاری کرده باشید. کاربرد این قسمت این است می‌توانید از متن این نوع دکمه‌ها به عنوان لینک استفاده نمایید. این رویداد دارای متغیری به نام e_Hyperlink می‌باشد.

e_Hyperlink: متغیری است رشته ای که متن لینک شده در شی xButton را بر می‌گرداند.

مثال: یک پروژه‌ی جدید ایجاد کنید و یک شی xButton به آن اضافه کنید و سپس روی آن دو بار کلیک کنید و در سربرگ Settings خاصیت EnableMarkup علامت گذاری کنید و سپس متن زیر را در قسمت Text یادداشت کنید:

```
<TextBlock>Hello <Hyperlink>AMS</Hyperlink></TextBlock>
```

حال به سربرگ Script بروید و در سربرگ On Hyperlink کد زیر را بنویسید:

```
Dialog.Message("Hyperlink", e_Hyperlink);
```

بعد از اجرا برنامه مشاهده خواهید کرد که عبارت AMS به صورت لینک در آمده است و می‌توان به طور جداگانه روی آن کلیک نمود.

On Focus: این رویداد زمانی رخ می‌دهد که یک شی بر روی صفحه پروژه انتخاب گردد و شامل اشیایی مثل: Input، RichText، و ... می‌شود. این رویداد متغیر رویدادی ندارد.

On Char: این رویداد مخصوص اشیایی است که ورودی متنی دارند و زمانی رخ می‌دهد که کاراکتری را درون آن تایپ کنیم. مثل: Input ؛ این رویداد دو متغیر رویدادی e_Char و e_Modifiers دارد.

e_Char: این متغیر کد عددی (کد دسیمال) کاراکترهای وارد شده را بر می‌گرداند.

برای مثال کد زیر را در یک شی Input وارد نمایید و پس از اجرای پروژه حرفی را درون آن تایپ کنید.

```
Dialog.Message("e_Char", e_Char);
```

e_Modifiers: با متغیر هم در قسمت On Key آشنا شدید.

On Select: این رویداد مختص اشیایی همانند ComboBox و ListBox می‌باشد و زمانی رخ می‌دهد که یک مورد را از لیست این اشیا انتخاب کنیم. این رویداد در شی ListBox متغیر رویدادی ندارد اما در شی ComboBox یک متغیر رویدادی با نام e_Selection دارد.

e_Selection: متغیری عددی که شماره‌ی گزینه‌ی انتخاب شده از لیست را بر می‌گرداند.

رویداد On Select در شی Tree نیز متغیری به نام e_NodeIndex که همانند e_Selection عمل می‌کند.

On Double-click: این رویداد که مختص شیListBox و Tree می‌باشد و زمانی رخ می‌دهد که روی یکی از گزینه های آن دو بار کلیک کنیم. این رویداد در شی Tree متغیری به نام e_NodeIndex دارد که شماره‌ی گزینه‌ی انتخاب شده را بر می‌گرداند.

On Check: این رویداد نیز مختص شی ListBox و Tree می‌باشد و زمانی رخ می‌دهد که یک گزینه را علامت گذاری کنیم. البته ما می‌بایست قبلاً در تنظیمات شی ListBox خاصیت Checklist box علامت گذاری کرده باشیم و هم چنین در شی Tree گزینه‌ی Show checkboxes را علامت زده باشیم. این رویداد در شی Tree دو متغیر با نام‌های e_NodeIndex و e_Checked .

e_NodeIndex: شماره‌ی گزینه‌ی انتخاب شده

e_Checked: وضعیت علامت دار بودن گزینه کلیک شده را بر می‌گرداند آن هم به صورت بولین یعنی true و false که true نشان دهنده‌ی علامت دار شدن گزینه‌ی مورد نظر می‌باشد.

On EditLabel: این رویداد زمانی رخ می‌دهد که بخواهیم یکی از گزینه های شی Tree را ویرایش کنیم و سه متغیر رویدادی دارد:

e_NodeIndex: شماره‌ی گزینه‌ی انتخاب شده

e_NewText: متن جدید

e_OldText: متن قدیمی

برای مثال کد زیر را در این رویداد On EditLabel یک شی Tree بنویسید:

```
Dialog.Message("Tree", "Index: "..e_NodeIndex.."\r\n".."New Text: ".. e_NewText.."\r\n".."Old Text: "..e_OldText);
```

البته قبل از اجرا در قسمت تنظیمات Tree گزینه‌ی Edit labels را علامت گذاری کنید.

پس از اجرای پروژه روی یکی از گزینه های Tree دو بار کلیک کنید و سپس آن را ویرایش نمایید و سپس گزینه دیگری را انتخاب نمایید.

On Expanded: این رویداد مختص شی Tree می‌باشد و زمانی رخ می‌دهد که روی علامتی + کنار گزینه‌ها کلیک می‌کنیم تا زیر مجموعه های آن‌ها را مشاهده کنیم. و دو متغیر رویدادی با نام‌های e_Expanded و e_NodeIndex.

e_Expanded: متغیری از نوع بولین می‌باشد و زمانی که گزینه مورد نظر جمع شده باشد و زیر مجموعه‌های آن قابل مشاهده نباشند مقدار false را بر می‌گرداند و علامت کنار آن به شکل - می‌باشد در غیر این صورت مقدار true را بر می‌گرداند.

برای مثال کد زیر را در این رویداد On Expanded یک شی Tree بنویسید و پروژه را تست کنید:

```
if e_Expanded == true then
Dialog.Message("Notice", "Expanded.");
```

```
end
```

On Cell Changed: این رویداد مختص شی Grid می‌باشد و زمانی رخ می‌دهد که متن سلول انتخاب شده تغییر یابد و دارای متغیرهای زیر می‌باشد:

e_Row: متغیری عددی که شماره سطر انتخاب شده را بر می‌گرداند.

e_Column: شماره‌ی ستونی که سلولی از آن انتخاب شده است را برمی‌گرداند.

e_OldText: متغیری رشته ای که متن قدیمی سلول ویرایش شده را در خود نگه می‌دارد.

e_NewText: متغیری رشته ای که متن جدید سلول ویرایش شده را در خود نگه می‌دارد.

On Selection Changed: این رویداد مختص شی Grid و RichText می‌باشد و زمانی رخ می‌دهد که سلول یا متن دیگری انتخاب شود. در شی Grid دارای دو متغیر رویدادی با نام‌های e_Row و e_Column می‌باشد که با آن‌ها آشنا شدید. اما در RichText دارای دو متغیر e_Min و e_Max می‌باشد.

e_Min: متغیری که شماره‌ی اولین کاراکتر متن انتخاب شده را بر می‌گرداند.

e_Max: متغیری که شماره‌ی آخرین کاراکتر متن انتخاب شده را بر می‌گرداند.

On Link: رویدادی است مختص شی RichText که هنگام کلیک بر روی یک متن لینک دار درون این شی، رخ می‌دهد. این رویداد دارای سه متغیر رویدادی می‌باشد:

e_Link: متغیری است که لینک کلیک شده را بر می‌گرداند.

e_Min: متغیری که شماره‌ی اولین کاراکتر متن لینک شده را بر می‌گرداند.

e_Max: متغیری که شماره‌ی آخرین کاراکتر متن لینک شده را بر می‌گرداند.

مثال: ابتدا یک شی RichText به پروژه‌ی خود اضافه کنید و سپس روی آن دو بار کلیک کنید و در سربرگ Settings گزینه‌ی Auto-detect URL را علامت بزنید و سپس کد زیر را در سربرگ Script> On Link وارد نمایید:

```
File.OpenURL(e_Link, SW_SHOWNORMAL);
```

رویدادها و متغیرهای رویدادی موجود در اشیای افزونه ای

برخی دیگر از اشیا وجود دارند که توسط افراد یا شرکت‌های مختلف برای AMS طراحی می‌گردند و سپس با برنامه افزوده می‌شوند (روش افزودن این افزونه‌ها در پیوست کتاب آمده است.) در این قسمت به معرفی رویدادهای اشیای افزونه ای پیش‌فرض AMS می‌پردازیم:

On PosChange: این رویداد مختص افزونه‌ی Slider می‌باشد و هنگامی که موقعیت سنج این شی جا به جا می‌گردد و دارای متغیر رویدادی e_Pos می‌باشد.

e_Pos: متغیری عددی که شماره‌ی موقعیت سنج شی Slider را بر می‌گرداند.

On Select: این رویداد مختص شی افزونه ای Calendar و ThumbList می‌باشد که در شی Calendar دارای دو متغیر با نام‌های e_StartDate وe_EndDate می‌باشد.

e_StartDate: متغیری رشته ای اولین تاریخ را از بین تاریخ‌های انتخاب شده را بر می‌گرداند.

e_EndDate: متغیری رشته ای آخرین تاریخ را از بین تاریخ‌های انتخاب شده را بر می‌گرداند.

On Month Select: رویدادی که مختص شی افزونه ای Calendar می‌باشد و زمانی رخ می‌دهد که از تقویم ماه را انتخاب کنیم.

e_CurrMonthYear: متغیری رشته ای که تاریخ کنونی را به صورت (ماه - سال) برمی‌گرداند.

e_SelMonthYear: متغیری رشته ای که تاریخ انتخاب شده از شی Calendar را به صورت (ماه - سال) برمی‌گرداند.

On Double-Click: این رویداد مختص شی افزونه‌ای ThumbList می‌باشد و زمانی رخ می‌دهد که روی یکی از گزینه‌های آن دو بار کلیک شود. و دارای متغیرe_Item می‌باشد.

e_Item: متغیری است عددی که شماره گزینه انتخاب شده را بر می‌گرداند.

فصل ششم: منوها

منوها یکی از اجزای تفکیک ناپذیر هر برنامه خوب محسوب می‌شوند که موجب دسترسی راحت به تمام قسمت‌های برنامه می‌شوند. برای مثال برنامه AMS با استفاده از منوها به شما این امکان را می‌دهد به راحتی به ابزارهای این برنامه دسترسی داشته باشید و بتوانید از آن‌ها استفاده نمایید. هم چنین در محیط‌های کد نویسی با استفاده از منوهای فرعی[۴] می‌توانید به راحتی کارهایی مانند برش متن، رونوشت، دسترسی به کدها و یا جستجو بین کدها را انجام دهید.

در این فصل چگونگی ایجاد منو در پروژه های خود آشنا خواهید شد. هم چنین نحوه‌ی ایجاد زیر منوها و منوهای فرعی را مشاهده خواهید نمود.

در این فصل به موارد زیر می‌پردازیم:

- با چگونگی ایجاد منوها آشنا می‌شوید.
- با نحوه ایجاد زیر منوها آشنا خواهید شد.
- چگونگی استفاده از منوهای فرعی را مشاهده خواهید کرد.

ویژگی‌های یک منو:

منوها در AMS دارای چندین ویژگی کلیدی می‌باشند. یکی از مهم‌ترین ویژگی‌های منوها در AMS سادگی و راحتی ایجاد آن می‌باشد.

منوهایی که در پروژه خود ایجاد می‌کنید علاوه بر متنی که برای هر منو وجود دارد، می‌توانند شامل عکس، کلید دسترسی و علامت قابل انتخاب باشند.

عکس:

اگر به منوهای برنامه‌هایی مانند Internet Explorer دقت کنید عکس‌هایی را در کنار گزینه های آن‌ها خواهید دید. در پروژه های ساخته شده با AMS نیز می‌توانید در کنار منوها عکس‌هایی را نمایش دهید.

6- منظور از منوی فرعی Pop-Up Menu است که با راست کلیک نمایش داده می شود.

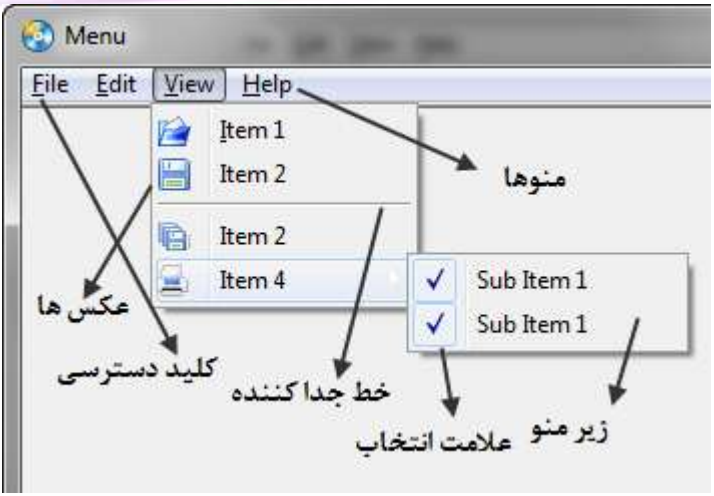
کلیدهای دسترسی:

یک کلید دسترسی به کاربر اجازه می‌دهد که با فشار کلید Alt و آن کلید در صفحه کلید، به منوی مورد نظر دسترسی داشته باشد. کلیدهای دسترسی برای هر منو به صورت یک حرف هستند که در نام منو با یک خط زیرین مشخص می‌شوند. وقتی که کلید Alt همراه با کلید دسترسی منو در صفحه کلید فشرده شد، منوی مربوطه در صفحه نمایش داده می‌شود و کاربر می‌تواند به وسیله‌ی کلیدهای جهت دار بین گزینه‌ها جا به جا شود.

علامت گذاری:

در کنار بعضی از منوها به جای عکس علامتی قرار می‌گیرد و مشخص می‌کند که این گزینه هم اکنون انتخاب شده است و یا در حال استفاده است. برای مثال اگر منوی View را از محیط AMS انتخاب کرده سپس به زیر منوی Toolbars بروید، نام تمام نوار ابزارهای موجود در AMS را مشاهده خواهید کرد که در کنار برخی از آن‌ها یک علامت ✓ قرار گرفته است. این به این معنی است که نوار ابزار مربوط به آن گزینه هم اکنون در محیط AMS نمایش داده می‌شود.

تصویر 6- 1 ویژگی‌هایی که یک منو می‌تواند داشته باشد را نمایش می‌دهد. همان طور که مشاهده می‌کنید، این منوی نمونه علاوه بر ویژگی‌های ذکر شده یک گزینه جدا کننده هم دارد. یک گزینه جدا کننده به صورت یک خط افقی نمایش داده می‌شود و برای دسته بندی گزینه های مربوط به هم در منو به کار می‌رود.



تصویر 6 - 1

شماره‌ی اختصاصی:

هر گزینه از منو یک شماره‌ی اختصاصی دارد که به وسیله‌ی آن می‌توان متوجه شد که کاربر کدام یک از گزینه های منو را کلیک کرده است.

ایجاد منوها:

هم اکنون وقت آن است که بدانیم چگونه می‌توان در یک پروژه منوهایی را ایجاد کرد. در مثال زیر با نحوه‌ی ایجاد منوها و استفاده از آن‌ها آشنا می‌شویم:

مثال: ایجاد منوها

1. یک پروژه جدید با نام Menus ایجاد کنید و اندازه آن را برابر با Small قرار دهید.
2. یک شی Input به پروژه خود اضافه کنید و سپس با دو بار کلیک روی آن و از گروه Multiline در سربرگ Settings گزینه‌ی Enabled را علامت گذاری کنید. هم چنین از گروه Special خاصیت Border Mode را برابر با Flat قرار دهید و با دکمه‌ی Ok تنظیمات جدید را تایید کنید.
3. برای ایجاد کردن منو برای پروژه‌ی خود، در محیط AMS از منوی Project گزینه‌ی Menu Bar... را کلیک کنید تا پنجره‌ی Menu Bar ظاهر شود. سپس گزینه‌ی Show menu

bar را فعال کنید. پیش فرض را با انتخاب آن‌ها و سپس استفاده از دکمه‌ی Delete پاک کنید.(تصویر 6 - 2)

تصویر 6 - 2

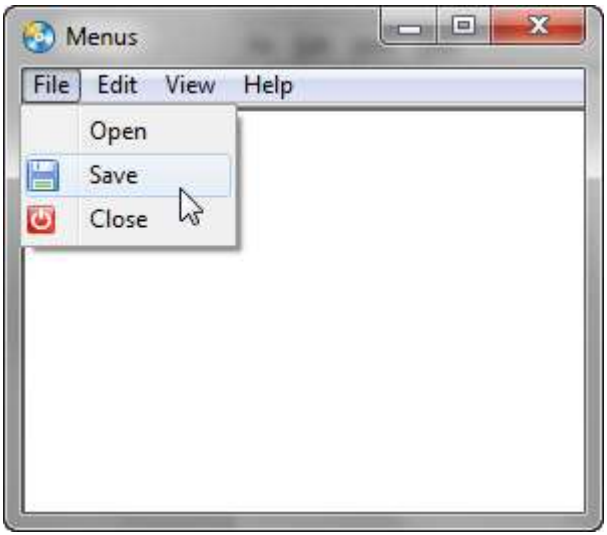
4. با استفاده از دکمه‌ی Add ابتدا چهار منو با نام‌های &File، &Edit، &View، &Help به پروژه بیفزایید و سپس دوباره با استفاده از دکمه‌ی Add برای منوی &File سه زیر منو با نام‌های &Open، &Save، &Close ایجاد نمایید. می‌توانید برای جا به جا کردن گزینه‌ها پس از ایجاد، از چهار دکمه‌ی جهت دار در پایین پنجره‌ی Menu Bar استفاده نمایید.
5. جهت افزودن عکس به منوها گزینه‌ی Use image list را علامت بزنید و با استفاده از دکمه‌ی Browse مسیر عکسی را که به صورت یک لیست از تصاویر 16x16 تشکیل شده

است؛ به برنامه بدهید(مطابق با تصویر 6 – 3 طراحی شود). از قسمت Transparent color رنگی را که نمی‌خواهید در هنگام نمایش عکس‌های منو نمایش داده نشود مشخص کنید. گزینه‌ی &Save را انتخاب کنید و سپس روی دکمه‌ی Properties کلیک کنید و از در پنجره ظاهر شده مقدار Icon ID را برابر با شماره‌ی 2 قرار دهید با کلیک روی دکمه‌ی Ok آن را تایید کنید. برای گزینه‌ی &Close همین مراحل را طی کنید و فقط Icon ID را برابر با 5 قرار دهید و در آخر با کلیک روی دکمه‌ی Ok تمامی تنظیمات مربوط به منوها را ذخیره کنید.

تصویر 6 - 3

نکته: شماره گذاری عکس‌ها از عدد صفر شروع می‌گردد مثلاً مورد اول برابر با 0 و مورد دوم برابر با 1 می‌باشد.

6. پروژه را با فشردن کلید F5 اجرا کنید. سپس روی منوی File کلیک کنید. نتیجه کار شما باید مشابه تصویر 6- 4 باشد.



تصویر 6 - 4

برای اینکه یک شی Input بتواند متن‌های بیشتر از یک سطر را نمایش دهد از خاصیت Multiline کمک می‌گیریم و آن را فعال می‌کنیم و سپس با استفاده از Virtical میله‌ی لغزان عمودی آن را فعال می‌کنیم تا بتوانیم با استفاده از حرکت دادن آن متن‌هایی که دیده نمی‌شوند را نیز مشاهده کنیم. اگر بخواهیم نحوه نمایش داده شدن حاشیه‌ی وردی متنی را تغییر دهیم از خاصیت Border Mode استفاده می‌کنیم و آن را برابر با Flat قرار می‌دهیم.

برای افزودن عکس قبل هر چیز شما باید یک عکس با طول مورد نیاز و عرض 16 پیکسل طراحی کنید و به تعداد نیاز در آن، عکس‌های منو را با اندازه‌ی 16x16 طراحی کنید و پشت سر هم بچینید به طوری که 16 پیکسل به هر عکس اختصاص یابد و هیچ عکسی نباید از محدوده خود خارج شود. برای این کار از برنامه‌های طراحی عکس مثل فتوشاپ استفاده نمایید. بعد از طراحی عکس می‌بایست آن را با فرمت bmp ذخیره نمایید در غیر این صورت AMS آن را نمایش نخواهد داد.

بعد از اینکه عکس مورد نظر را ذخیره کردیم مسیر آن را به برنامه بدهید و سپس هر عکسی را که می‌خواهید به یک گزینه خاص از یک منو اختصاص دهید اول آن گزینه را انتخاب کنید و سپس روی دکمه‌ی Properties کلیک کنید و در قسمت Icon ID شماره‌ی آن عکس را در قسمت وارد کنید. توجه داشته باشید که شمارش ردیف عکس‌ها باید از صفر شروع شود. پس از تایید تنظیمات و اجرای پروژه می‌بینیم که منوی ما ایجاد شده است و با انتخاب هر کدام از آن‌ها می‌توانیم گزینه های موجود در آن‌ها را مشاهده کنیم.

نکته: اگر می‌خواهید منوی شما هیچ عکسی را در کنار خود نداشته باشد در قسمت Icon ID عدد -1 را وارد کنید.

تا اینجا توانستیم منوهایی را به پروژه بیفزاییم اما هنوز آن‌ها هیچ کاری انجام نمی‌دهند در ادامه نحوه‌ی تعیین دستورات برای منو آشنا خواهید شد.

نوشتن کد برای منوها:

برای آنکه بتوانیم برای هر گزینه از منوها فعالیت مشخصی را تعیین کنیم می‌بایست با استفاده از کد نویسی این کار را انجام دهیم.

مثال: کد نویسی برای منوها

1. پروژه‌ی Menus را باز کنید و از منوی Project گزینه‌ی Menu Bar... را کلیک کنید تا پنجره‌ی Menu Bar ظاهر شود روی علامت + در کنار منوی &Files کلیک کنید سپس به ترتیب با انتخاب گزینه‌های &Open، &Save، &Close و کلیک روی دکمه‌ی Properties مقدار Menu ID را برابر با 101، 102 و 103 قرار دهید و آخر سر با Ok آن‌ها را تایید کنید.
2. از منوی Projects گزینه‌ی Actions... را کلیک کنید. در پنجره باز شده به سربرگ On Menu بروید و کدهای زیر را در آن وارد نمایید و با Ok کدها را تایید نمایید:

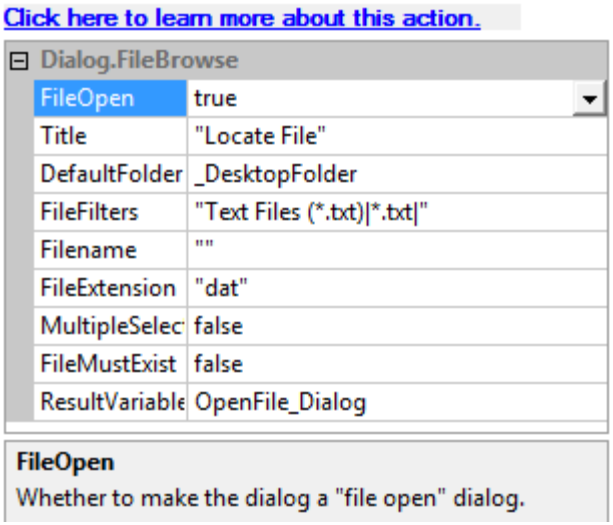
```
if e_ID == 101 then
  OpenFile_Dialog = Dialog.FileBrowse(true, "Locate File",
_DesktopFolder, "Text Files (*.txt)|*.txt|", "", "dat", false, false);
  if OpenFile_Dialog[1]~= "CANCEL" then
  text_in_file=TextFile.ReadToString(OpenFile_Dialog[1]);
  Input.SetText("Input1", text_in_file);
 end
elseif e_ID == 102 then
  SaveFile_Dialog = Dialog.FileBrowse(false, "Locate File",
_DesktopFolder, "Text Files (*.txt)|*.txt|", "", "dat", false, false);
  if SaveFile_Dialog[1]~= "CANCEL" then
  input_text = Input.GetText("Input1");
  TextFile.WriteFromString(SaveFile_Dialog[1],input_text, false);
  end
elseif e_ID == 103 then
        Application.Exit(0);
end
```

3. پروژه را اجرا نمایید و گزینه های منوی File را یکی یکی امتحان کنید تا نتیجه‌ی کار خود را ببینید.

در این قسمت از مثال ما با یکی دیگر از خصوصیات گزینه های منو آشنا شدیم که عبارت بود از Menu ID که همان شماره‌ی اختصاصی است . وقتی که ما از منوها استفاده می‌کنیم به هر گزینه از منوها یک شماره اختصاص داده می‌شود که با استفاده از آن شماره می‌توان شرط‌هایی را برای منوها تعریف نمود این شماره در یک متغیر رویدادی به نام e_ID ذخیره می‌گردد و فقط در هنگام استفاده از منوها قابل استفاده می‌باشد. پس از آن که به هر گزینه یک شماره اختصاص دادیم (101، 102 و 103) از منوی Projects>Action... به سربرگ On Menu می‌رویم تا با استفاده از دستور if شرط‌هایی را برای منوهای پروژه‌ی خود تعریف کنیم.

ابتدا بررسی می‌کنیم که اگر مقدار متغیر e_ID برابر با 101 بود برنامه با استفاده از دستور Dialog.FileBrowse یک دیالوگ برای فراخوانی فایل نمایش دهد و دوباره بررسی می‌کنیم که اگر مقدار بازگشتی از Dialog.FileBrowse با "CANCEL" برابر نبود یعنی Ok بود محتوای فایل متنی انتخاب شده را بگیرد و سپس در شی Input نمایش دهید.

اگر روی دستور Dialog.FileBrowse دو بار کلیک کنید می‌توانید تمام خصوصیت‌های مربوط به این دستور را مشاهده نمایید.(تصویر 6 - 5)



تصویر 6 - 5

همان طور که در تصویر مشاهده می‌کنید این دستور شامل 9 خصوصیت می‌باشد که می‌توان آن‌ها را تغییر داد.

1. خصوصیت FileOpen: اگر مقدارش را true قرار دهید برای فراخوانی فایل به کار خواهد رفت و اگر مقدارش را false قرار دهید برای ذخیره کردن فایل به کار خواهد رفت یعنی نوع دیالوگ ظاهر شده در هر دو حالت متفاوت خواهد بود.

2. Title که برای تعیین عنوان پنجره دیالوگ به کار می‌رود.

3. DefaultFolder: مسیر پیش‌فرض را برای نمایش در دیالوگ تعیین می‌کند.

4. FileFilters: فرمت فایل‌هایی را که می‌خواهید پشتیبانی شود در این قسمت مشخص کنید. همان طور که در تصویر هم مشاهده می‌نمایید ابتدا متنی که فرمت فایل را توضیح می‌دهد وارد کرده‌ایم Text File (*.txt) و سپس میان دو علامت || پسوند فایل را مشخص نموده‌ایم یعنی |*.txt| .

برای نمونه مثالی دیگر:

"|Pictures (.png, .jpg, .tif)|*.png;*.jpg;*.tif"

5. FileName: می‌توانید نام پیش فرضی را برای فایل وارد نمایید.

6. FileExtension: برای تعیین پسوند پیش‌فرض فایل به کار می‌رود.

7. MultipleSelect: اگر مقدار این خاصی برابر با true باشد می‌توان چندین فایل را در پنجره‌ی دیالوگ به صورت همزمان انتخاب نمود.

8. FileMustExist: اگر این خصوصیت را برابر با true قرار دهید برنامه بررسی خواهد کرد که آیا فایل مورد نظر ایجاد شده است یا خیر؟، اگر ایجاد شده بود به ادامه‌ی کارها می‌پردازد.

9. ResultVariable: نتیجه‌ی کار دیالوگ را در خود نگه می‌دارد و مشخص می‌کند که آیا کاربر دکمه‌ی Cancel را کلیک کرده یا دکمه‌ی تایید ؟ در این قسمت اسم مناسبی را یادداشت می‌کنیم تا نتیجه کار در آن ذخیره گردد که ما نام OpenFile_Dialog را انتخاب نموده‌ایم. نتیجه بازگشتی به صورت یک آرایه خواهد بود و برای همین منظور هنگام بررسی این متغیر اندیس 1 را قرار داده‌ایم.

بعد با دستور if بررسی می‌کنیم که اگر OpenFile_Dialog[1] برابر با "CANCEL" نبود با استفاده از دستور TextFile.ReadToString محتوای فایل انتخاب شده را در متغیر text_in_file بریز و سپس در شی Input نمایش بده:

```
if e_ID == 101 then
  OpenFile_Dialog = Dialog.FileBrowse(true, "Locate File",
_DesktopFolder, "Text Files (*.txt)|*.txt|", "", "dat", false, false);
  if OpenFile_Dialog[1]~= "CANCEL" then
  text_in_file=TextFile.ReadToString(OpenFile_Dialog[1]);
  Input.SetText("Input1", text_in_file);
 end
```

سپس بررسی می‌کنیم که اگر مقدار متغیر e_ID برابر با 102 بود برنامه با استفاده از دستور Dialog.FileBrowse یک دیالوگ برای ذخیره‌ی فایل نمایش دهد(تفاوت این کد با کد بالا در خصوصیت FileOpen می‌باشد که مقدار آن را برابر با false قرار داده‌ایم). در این قسمت برای ذخیره‌ی فایل از دستور TextFile.WriteFromString استفاده می‌کنیم. مسیر ذخیره را نیز از متغیر بازگشتی SaveFile_Dialog[1] می‌گیریم:

```
elseif e_ID == 102 then
  SaveFile_Dialog = Dialog.FileBrowse(false, "Locate File",
_DesktopFolder, "Text Files (*.txt)|*.txt|", "", "dat", false, false);
  if SaveFile_Dialog[1]~= "CANCEL" then
  input_text = Input.GetText("Input1");
  TextFile.WriteFromString(SaveFile_Dialog[1],input_text, false);
  end
```

در آخر هم بررسی می‌کنیم که اگر مقدار متغیر e_ID برابر با 103 بود برنامه با استفاده از دستور Application.Exit بسته شود.

```
elseif e_ID == 103 then
        Application.Exit(0);
```

```
end
```

نکته: متغیر رویدادی e_ID در قسمت On Timer صفحه‌ها نیز وجود دارد ولی این دو متغیر هرگز با همدیگر تداخل پیدا نمی‌کنند.

منوهای فرعی:

تا اینجا توانستیم به پروژه‌ی خود منوهایی اضافه کنیم که در قسمت بالای صفحه نمایان می‌شود و منوی اصلی پروژه تلقی می‌شوند. نوع دیگری از منو نیز وجود دارد که معمولاً با راست کلیک در برنامه نمایان می‌شوند. این منوها را منوی فرعی می‌نامیم. در مثال زیر با روش ایجاد یک منوی فرعی آشنا می‌شویم.

مثال: ایجاد منوی فرعی

1. پروژه‌ی Menus را باز کنید و از پنجره‌ی Properties قسمت On Mouse Button را پیدا کرده و سپس بر روی دکمه‌ی ... مقابل آن کلیک کنید تا پنجره کد نویسی این رویداد ظاهر شود. (هم چنین می‌توانید با رفتن به منوی Page>Properties و سپس انتخاب سربرگ On Mouse Button به این محیط دسترسی داشته باشید.
2. کدهای زیر را در این قسمت وارد نمایید:

```
if(e_Type == 2)then
  tblMenu = {};
  tblMenu[1] ={};
  tblMenu[1].Text = "&Open";
  tblMenu[1].ID = 104;
  tblMenu[1].IconID = -1;
  tblMenu[1].Checked = false;
  tblMenu[1].Enabled = true;

  tblMenu[2] ={};
  tblMenu[2].Text = "&Save";
```

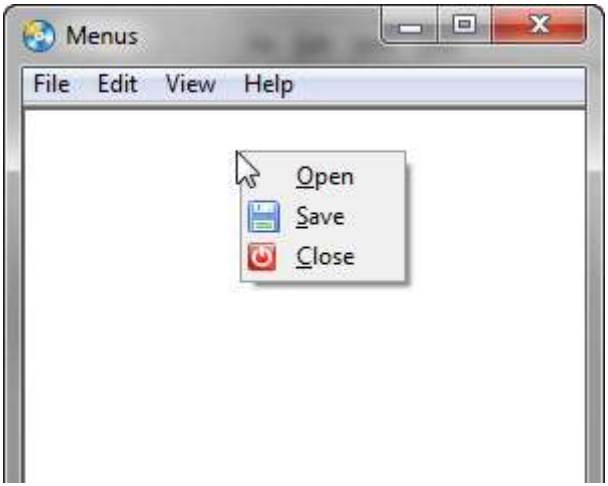
```
 tblMenu[2].ID = 105;
 tblMenu[2].IconID = 2;
 tblMenu[2].Checked = false;
 tblMenu[2].Enabled = true;

 tblMenu[3] ={};
 tblMenu[3].Text = "&Close";
 tblMenu[3].ID = 106;
 tblMenu[3].IconID = 5;
 tblMenu[3].Checked = false;
 tblMenu[3].Enabled = true;
  n_ID = Application.ShowPopupMenu(e_X, e_Y, tblMenu,
TPM_LEFTALIGN, TPM_TOPALIGN, true, true);
if n_ID == 104 then
  OpenFile_Dialog = Dialog.FileBrowse(true, "Locate File",
_DesktopFolder, "Text Files (*.txt)|*.txt|", "", "dat", false, false);
  if OpenFile_Dialog[1]~= "CANCEL" then
  text_in_file=TextFile.ReadToString(OpenFile_Dialog[1]);
  Input.SetText("Input1", text_in_file);
 end
elseif n_ID == 105 then
  SaveFile_Dialog = Dialog.FileBrowse(false, "Locate File",
_DesktopFolder, "Text Files (*.txt)|*.txt|", "", "dat", false, false);
  if SaveFile_Dialog[1]~= "CANCEL" then
  input_text = Input.GetText("Input1");
  TextFile.WriteFromString(SaveFile_Dialog[1],input_text, false);
  end
elseif n_ID == 106 then
        Application.Exit(0);
```

```
end
end
```

روی دکمه‌ی Ok کلیک کنید تا کدها تایید شوند.

3. پروژه را اجرا کنید و در قسمتی از برنامه راست کلیک کنید تا نتیجه کار خود را مشاهده نمایید.(تصویر 6- 7)



تصویر 6 - 6

وقتی از پنجره‌ی Properties به رویداد On Mouse Button می‌رویم با سه متغیر رویدادی از پیش تعریف شده رو به رو هستیم: e_Type ، e_X، e_Y که هر سه عددی می‌باشند. متغیر e_Type دکمه‌ی فشرده شده از ماوس را در خود نگهداری می‌کند، متغیر e_X حرکت افقی ماوس را در خود ذخیره می‌سازد و متغیر e_Y حرکت عمودی ماوس را در خود نگه می‌دارد.

متغیر e_Type چهار حالت دارد :

حالت کلیک ماوس	شماره
کلیک چپ ماوس فشرده شده است	0
کلیک چپ ماوس رها است	1
کلیک راست ماوس فشرده شده است	2

3 کلیک راست ماوس رها شده است

جدول 6- 1

در کد نوشته شده ما ابتدا بررسی می‌کنیم که آیا مقدار e_Type برابر با 2 هست یا نه؟ اگر برابر با 2 بود کدهای درون if اجرا می‌شوند و شروع به ساخت و نمایش منوی فرعی می‌کنند.

منوی باید به صورت آرایه تعریف شود چون شامل چندین عنصر است:

```
tblMenu = {};
```

نام منوی مان tblMenu قرار دادیم. سپس نوبت به ساخت اولین گزینه از منوی tblMenu می‌رسد که آن را با اندیس 1 مشخص می‌کنیم:

```
tblMenu[1] ={};
```

یک متن به آن اختصاص می‌دهیم:

```
tblMenu[1].Text = "&Open";
```

حال یک ID به آن اختصاص می‌دهیم:

```
tblMenu[1].ID = 104;
```

شماره عکس کنار منو را که در مثال قبل به پروژه افزودیم تعیین می‌کنیم که در اینجا هیچ عکسی تعیین نشده است:

```
tblMenu[1].IconID = -1;
```

سپس مشخص می‌کنیم که آیا منوی ما علامت گذاری شده باشد یا خیر؟ که با false از این کار صرف نظر می‌کنیم و با true این مورد را تایید می‌کنیم.

```
tblMenu[1].Checked = false;
```

در این قسمت نیز مشخص می‌کنیم که منوی ما باید فعال باشد. توجه داشته باشید که اگر این قسمت برابر با false باشد نمی‌توانید از این گزینه‌ی منو در هنگام مشاهده استفاده کنید.

```
tblMenu[1].Enabled = true;
```

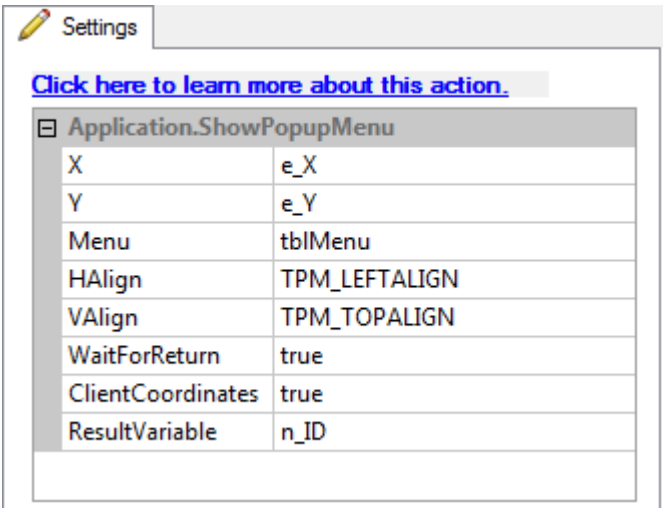
سپس تعیین می‌کنیم آیا این گزینه خط جداکننده باشد یا خیر؟

```
tblMenu[1].Separator = false;
```

نکته: چنانچه بخواهید گزینه از منو به عنوان خط جدا کننده عمل کند می‌بایست به مقدار خاصیت Text را برای آن برابر با "---" قرار دهید و هم چنین مقدار Separator را برابر با true قرار دهید:

```
tblMenu[1].Text = "---";
tblMenu[1].Separator = true;
```

به همین منوال تا tblMenu[3] پیش می‌رویم و سپس با استفاده از دستور Application.ShowPopupMenu منوی فرعی را ایجاد می‌کنیم خصوصیات این دستور را با استفاده از دو بار کلیک روی آن مشاهده کنید(تصویر 6- 8).



تصویر 6 - 7

خصوصیت X و Y به ترتیب محل افقی و عمودی ظاهر شدن منو را تعیین می‌کنند که ما آن‌ها را از متغیر رویدادی e_X و e_Y گرفته‌ایم یعنی مختصات خود نشانگر ماوس، خاصیت Menu نیز باید شامل نام منوی ایجاد شده ما باشد که همان tblMenu می‌باشد. HAlign و VAlign به ترتیب محل افقی و عمودی نمایش منوی فرعی را از نشانگر ماوس مشخص می‌کنند. WaitForReturn نیز تعیین می‌کند که آیا پروژه تا پایان کار منو صبر کند یا خیر؟ که ما با true این مورد را تایید کرده‌ایم. ClientCoordinates نیز اگر برابر با true باشد منوی فرعی با توجه به مختصات صفحه‌ی کنونی پروژه ظاهر خواهد شد و اگر برابر با false

باشد با توجه با تنظیمات مختصات مانیتور ظاهر خواهد شد. در قسمت ResultVariable نیز نام متغیری که نتیجه‌ی دستور Application.ShowPopupMenu را در خود ذخیره می‌کند می‌نویسیم یعنی همان n_ID (که ID تعیین شده برای هر گزینه از منو را که کلیک شود در خود ذخیره می‌کند). اگر برفرض کاربر از منوی ظاهر شده گزینه‌ی Open را انتخاب کند مقدار 104 در n_ID ذخیره می‌گردد.

در ادامه نیز با یک دستور if شرط تعیین کردیم که اگر n_ID برابر با 104 بود فلان کار را انجام بده و ... که در مثال قبلی توضیح آن بیان شد.

```
n_ID = Application.ShowPopupMenu(e_X, e_Y, tblMenu, TPM_LEFTALIGN, TPM_TOPALIGN, true, true);
if n_ID == 104 then
...
elseif n_ID == 105 then
...
elseif n_ID == 106 then
...
end
```

فرستادن منوی اصلی به منوی فرعی:

ممکن است گاهی اوقات نیاز پیدا کنیم که منوی اصلی برنامه را به منوی فرعی نیز بفرستیم به طوری که با راست کلیک در محیط پروژه منوی اصلی در کنار نشانگر ماوس نمایان گردد. در مثال زیر با روش انجام این کار آشنا خواهیم شد:

مثال: فرستادن منوی اصلی به منوی فرعی

1. یک پروژه‌ی جدید با نام Menus_in_pop_up ایجاد کنید و یک شی Input به آن اضافه نمایید مطابق با تنظیمات شی Input در پروژه‌ی Menus . سپس منوهای اصلی پروژه‌ی قبلی یعنی پروژه‌ی Menus را برای این پروژه نیز ایجاد کنید با همان تنظیمات پروژه‌ی

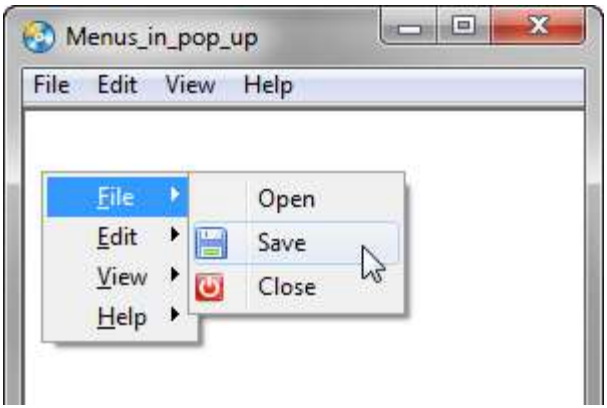
Menus. هم چنین کدهای مربوط به قسمت Project>Actions...>On Menu را نیز مطابق با پروژه‌ی Menus بنویسید.

2. حال از منوی Page>Properties به سربرگ Script>On Mouse Button بروید و کدهای زیر را در آن بنویسید:

```
if(e_Type == 2)then
Main_Menu = Application.GetMenu();
n_ID = Application.ShowPopupMenu(e_X, e_Y, Main_Menu,
TPM_LEFTALIGN, TPM_TOPALIGN, true, true);
        if n_ID == 101 then
                OpenFile_Dialog = Dialog.FileBrowse(true, "Locate
File", _DesktopFolder, "Text Files (*.txt)|*.txt|", "", "dat", false,
false);
                if OpenFile_Dialog[1]~= "CANCEL" then
                text_in_file =
TextFile.ReadToString(OpenFile_Dialog[1]);
                Input.SetText("Input1", text_in_file);
                end
        elseif n_ID == 102 then
                SaveFile_Dialog = Dialog.FileBrowse(false, "Locate
File", _DesktopFolder, "Text Files (*.txt)|*.txt|", "", "dat", false,
false);
                if SaveFile_Dialog[1]~= "CANCEL" then
                input_text = Input.GetText("Input1");
                TextFile.WriteFromString(SaveFile_Dialog[1],
input_text, false);
                end
        elseif n_ID == 103 then
                Application.Exit(0);
        end
```

```
end
```

3. پروژه را اجرا کرده و سپس روی قسمتی از آن راست کلیک کنید. نتیجه کار می‌بایست مشابه تصویر 6 – 9 باشد.



تصویر 6 - 8

بررسی مثال: در این پروژه با استفاده از دستور Application.GetMenu توانستیم تمامی مشخصات منوی پروژه‌ی خود را به دست بیاوریم و در متغیری به نام Main_Menu ذخیره نماییم و سپس آن مشخصات را به وسیله‌ی این متغیر به خصوصیت Menu از دستور Application.ShowPopupMenu فرستادیم تا برنامه منوی فرعی را مطابق با منوی اصلی ایجاد کند و تعیین کردیم که نتیجه را در متغیری به نام n_ID ذخیره کند. سپس با استفاده از دستور if به بررسی نتایج متغیر n_ID پرداختیم که توضیح آن در پروژه‌ی Menus بیان شد.

منوی فرعی را می‌توان برای کلیک چپ نیز تعیین نمود برای مثال به کد زیر دقت کنید:

```
if (e_Type == 0)then
دستورات مربوط به منوها
end
```

فصل هفتم: کار با بانک‌های اطلاعاتی

بیشتر نرم افزارهایی که امروزه در دسترس ما قرار می‌گیرند با داده‌ها و اطلاعات مختلفی رابطه دارند و از آن‌ها استفاده می‌کنند. این برنامه این اطلاعات را در بانک‌های اطلاعاتی نگهداری می‌کنند. در نتیجه در دینای برنامه نویسی نیاز است که بتوانیم با بانک‌های اطلاعاتی رابطه برقرار کنیم تا اطلاعات خود را در آن‌ها نگهداری و مورد استفاده‌ی مجدد قرار دهیم بانک‌هایی اطلاعاتی همانند: SQL Server ،MySQL ، Oracel، SQLite و ... .

در این فصل به موارد زیر می‌پردازیم:

- با مفهوم بانک‌های اطلاعاتی آشنا می‌شویم.
- مفاهیم پایه ای بانک‌های اطلاعاتی را می‌آموزیم.
- با بانک اطلاعاتی SQLite کار خواهیم کرد.
- روش کار با اطلاعاتی MySQL را خواهیم آموخت.
- چگونگی اتصال به دیگر بانک‌های اطلاعاتی موجود در AMS را می‌آموزید.

مفهوم بانک‌های اطلاعاتی

بانک اطلاعاتی شامل یک یا چندین فایل بزرگ و پیچیده است که داده‌ها در آن در یک قالب ساخت یافته و منظم ذخیره می‌شوند. موتور بانک اطلاعاتی معمولاً به برنامه ای گفته می‌شود که این فایل و یا فایل‌ها و نیز داده های درون آن‌ها را مدیریت می‌کند. در طی این فصل با دو موتور بانک اطلاعاتی SQLite و MySQL کار خواهیم کرد که شباهت‌های بسیار زیادی در به‌کارگیری دستورات دارند.

چند مفهوم پایه‌ای:

جدول(Table)

همان‌طور که از نام آن مشخص است یک جدول مجموعه‌ای از سطرها و ستون‌هاست، هر جدول از تعدادی ستون که فیلد نامیده می‌شود تشکیل می‌گردد.

فیلد (Field)

در هر یک از جدول به هر ستون یک فیلد اطلاعاتی گفته می‌شود. هر فیلد اطلاعاتی در هر جدول حداقل می‌بایستی نام، نوع و اندازه داشته باشد .

رکورد(Record)

به هر ردیف اطلاعاتی در هر جدول یک رکورد آن جدول گفته می‌شود که شامل اطلاعات کامل آن ردیف است.

کلید اصلی(Primary Key)

به هر مشخصه منحصر به فرد در یک جدول که در جداولی دیگر نیز دارای تکرار باشد کلید اصلی(Primary Key) گفته می‌شود(البته در یک جدولِ تنها نیز می‌توان از Primary Keyاستفاده نمود).

دیتابیس SQLite3

اس‌کیوال لایت (به انگلیسی: SQLite) یک پایگاه داده کوچک (حدود ۵۰۰ کیلوبایت) که به زبان C در قالب یک کتاب خانه نوشته شده است و یک پایگاه داده‌ای رابطه‌ای به حساب می‌آید. این پایگاه داده، به صورت آزاد و متن‌باز منتشر می‌شود. از ویژگی‌های اس‌کیوال لایت آن است که پایگاه داده‌ای متشکل از یک پرونده، با حجم کم و عدم وابستگی به سیستم‌عامل، دارای محیط مدیریتی خوب که تمامی امکانات آن را پوشش می‌دهد در اختیار کاربر می‌گذارد. با این حال اس‌کیوال لایت از تمامی امکانات اس‌کیوال پشتیبانی نمی‌کند. برخلاف انواع دیگر پایگاه داده، اس‌کیوال لایت یک پروسه جداگانه نیست که توسط برنامه‌ی اصلی فراخوانی شود، بلکه جزئی از خود برنامه‌ی اصلی است و با اجرای برنامه‌ی اصلی، اس‌کیوال لایت هم اجرا می‌گردد.

دستورات مهم و کاربردی دیتابیس SQlite3

1. Create Table
2. Execute
3. Insert
4. Select

5. Update
6. Delete
7. Drop
8. Index
9. Alter

شرح مختصر دستورات SQlite3

1. Create: از این دستور برای ساخت جدول در دیتابیس استفاده می‌شود.
2. Execute: دستورات یا فعالیت‌های مورد نظر شما را به دیتابیس انتقال می‌دهد.
3. Insert: با استفاده از این دستور می‌توان مقداری را در دیتابیس ثبت (ذخیره) نمود حال آن مقدار می‌تواند از نوع رشته‌ای و یا عددی و یا ... باشد.
4. Select: با استفاده از این دستور می‌توان مقداری را از دیتابیس انتخاب نمود (یا باز گرداند) تا در ادامه مورد استفاده قرار گیرد.
5. Update: با استفاده از دستور Update می‌توان مواردی (رکوردها) را در دیتابیس ویرایش یا به روز رسانی کرد.
6. Delete: با استفاده از دستور Delete می‌توان مواردی (رکوردها) را در دیتابیس حذف کرد.
7. Drop: با استفاده از دستور Drop می‌توان جدول یا ستونی را از دیتابیس حذف کرد.
8. Index: هر ایندکس در بانک اطلاعاتی ترتیب قرار گرفتن صعودی یا نزولی یک یا چند ستون را در یک جدول شامل می‌گردد و کاربرد عمده آن روش دسترسی سریع به اطلاعات در چند جدول نیز محسوب می‌گردد.
9. Alter: به منظور تغییر در ساختار یک جدول بکار می‌رود مثلاً افزودن فیلد به جدول، تغییر نوع فیلد و ...

عبارات و علایم مورد استفاده در دستورات دیتابیس

✓ علایم مورد استفاده عمومی:

***:** برای انتخاب تمام موارد از قبیل تمام رکوردها در دیتابیس

'': عبارتی را که می‌خواهید به صورت رشته متنی به دیتابیس بفرستید میان این علامت قرار می‌دهید.

,: جدا کننده

✓ عبارت و علایم مورد استفاده در جستجو:

Like: مناسب برای جستجو در میان رکوردهای رشته‌ای

Where: تعیین شرط هنگام جستجو

=: مناسب برای جستجو در میان رکوردهایی که می‌خواهید عین عبارت یا عدد را بیابید.

%%: چنانچه عبارت مورد جستجو با دستور Like در میان این علامت قرار بگیرد عبارت مورد جستجو را در هر جای رکورد که بیابد بر می‌گرداند؛ هم چنین می‌توان این عبارت را در ابتدا یا انتهای عبارت قرار داد که در آن صورت در جستجو ابتدا و انتها را مد نظر قرار خواهد داد.

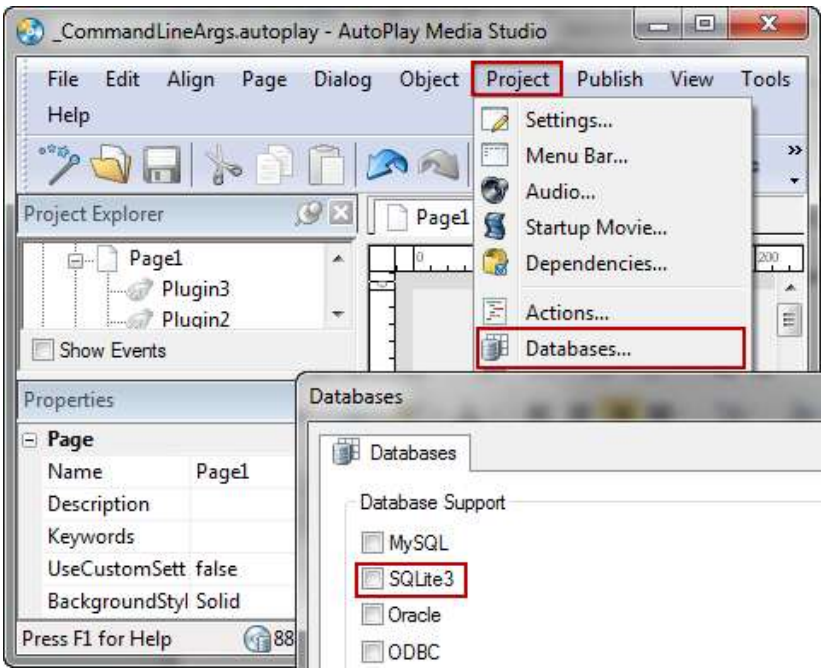
Order by: جهت مرتب سازی بر اساس یک فیلید (ستون)

توابع موجود در Auto Play Media Studio برای کار با دیتابیس SQlite3:

```
1) SQLite3:connect(sourcename[,username[,password]])
2) SQLite3: close()
3) SQLite3Connection:close()
4) SQLite3Connection:commit()
5) SQLite3Connection:execute(statement)
6) SQLite3Connection:rollback()
7) SQLite3Connection:setautocommit(boolean)
8) SQLite3Cursor:fetch([table[,modestring]])
9) SQLite3Cursor:getcolnames()
10) SQLite3Cursor:getcoltypes()
11) SQLite3Cursor:close()
```

پیش نیازها:

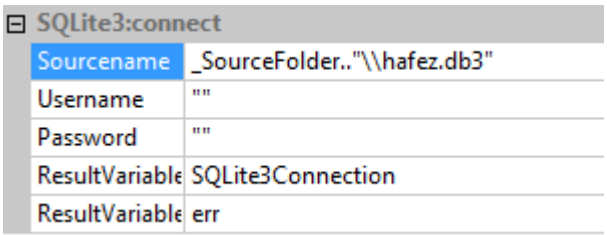
1. برای مشاهده و کار با توابع SQLite3 لازم است ابتدا دیتابیس مورد نظر را در برنامه فعال نمایید؛ برای این کار به منوی Project>Databases... بروید و سپس گزینه مربوط به دیتابیس SQLite3 را علامت گذاری نمایید. (تصویر 7- 1)



تصویر 7 - 1

شرح توابع:

SQLite3:connect(): این تابع برای اتصال به دیتابیس مورد نظر به کار می‌رود.(تصویر 7-1)



تصویر 7 - 2

SQLite3:close(): این تابع برای قطع ارتباط برنامه با دیتابیس به کار می‌رود. (قطع ارتباط کلی)

نکته: هرگز از این دستور برای بستن اتصال با دیتابیس استفاده نشود.

SQLite3Connection:close(): برای بستن اتصال دیتابیس در حال استفاده به کار می‌رود.

مثال :

```
SQLite3Connection, err =
SQLite3:connect(_SourceFolder.."\\hafez.db3", "", "");
```

در این مثال:

SQLite3Connection متغیری است که نتیجه اتصال به دیتابیس را در خود نگه می‌دارد.

err که با علامت , از متغیر قبلی جدا شده است متغیری است که خطاها را در خود ذخیره می‌کند.

_SourceFolder.."\\hafez.db3" مسیر ذخیره دیتابیس شما می‌باشد که اگر دیتابیس مورد نظر موجود نباشد برنامه خودش دیتابیسی خالی را با همان نام و پسوند ایجاد می‌کند.

در ادامه می‌بینید که دو علامت "" پست سر هم قرار گرفته‌اند که اولی "نام کاربری دیتابیس" و دومی "کلمه عبور" اتصال به دیتابیس می‌باشد. اما در این دیتابیس امکان تعریف نام کاربری و کلمه‌ی عبور امکان پذیر نمی‌باشد و باید خالی بمانند؛ و برای حفاظت از دیتابیس در مقابل نفوذ دیگران راه‌هایی وجود دارد که با آن‌ها اشاره خواهد شد.

نکته مهمی که برای این دستور وجود دارد آن است که هر اتصالی که ایجاد شد بعد از انجام فعالیت مورد نظر باید بسته شود که دستور آن در زیر آمده است:

```
SQLite3Connection:close()
```

SQLite3Connection:commit(): برای حصول اطمینان از فرایند اتصال با دیتابیس و انجام فعالیت‌ها به کار می‌رود.

مثال:

```
commit, err = SQLite3Connection:commit()
```

این تابع برای دریافت اطمینان از انجام اتصال و هم چنین دستورات ارسال شده به کار می‌رود اما در نظر داشته باشید که این دستور برای AMS در تمام سیستم‌ها عامل‌ها به طور عادی در دسترس نیست و برای استفاده از این تابع بایستی transaction را در دیتابیس فعال کنید و البته نیازی مبرمی هم به این دستور ندارید چون تمام دستورها متغیر err را مورد پشتیبانی قرار می هند که می‌توان با بررسی مقدار آن که در ادامه خواهد آمد از انجام فعالیت‌ها اطمینان حاصل کرد.

حال اگر سیستم شما از این تابع پشتیبانی کرد commit یعنی متغیر مربوط به دستور که نتیجه را در خود می‌ریزد در صورت موفقیت true را بر می‌گرداند و اگر ناموفق باشد false را بر می‌گرداند.

SQLite3Connection:execute(statement): با این تابع می‌توانید دستورات مورد نظر خودتان را به دیتابیس بفرستید تا اجرا شوند.

مثال:

```
SQLite3Cursor, err = SQLite3Connection:execute("CREATE TABLE hamseda (ID INTEGER PRIMARY KEY, Name, Last_Name)");
```

این تابع در مثال بالا دستور CREATE TABLE را به دیتابیس می‌فرستد که برای ساخت یک جدول در دیتابیس متصل شده به کار می‌رود که در مثال hamseda نام جدول، ID، Name، Last_Name که داخل پرانتز آمده‌اند نام فیلد می‌باشند؛ INTEGER PRIMARY KEY هم نوع فیلد می‌باشد که تعریف شده است اما اگر برای فیلدی نوع آن را تعریف نکنید به طور خودکار نوعی به آن داده خواهد شد و می‌توان در آن اطلاعاتی از قبیل متن و ... را ذخیره کرد.

و اما باید بدانید که نحوه‌ی ارسال دستورات با هم دیگر یکسان نیست، به همین منظور در زیر برای شما از نحوه‌ی ارسال هر کدام از دستورات مثالی ارایه می‌گردد:

جدول فرضی ما Dic نام دارد با سه ستون با نام‌های: ID , Loghat , Mani . دستورات بایستی طبق مثال زیر فرستاده شوند:

```
SQLite3Connection:execute("دستورات");
```

1. SELECT * from Dic

   انتخاب تمام رکوردها از جدول

2. DELETE from Dic where ID = 1

   حذف تمام رکوردهایی که ستون ID آن‌ها مساوی با 1 می‌باشد

3. INSERT INTO Dic(Loghat,Mani) VALUES ('hello' , 'سلام')

   - افزودن کلمه‌ی hello به سلول ستون Loghat و افزودن کلمه‌ی سلام به سلول ستون Mani

4. UPDATE Dic SET Loghat = 'bye' where ID= 1

   ویرایش یا به روز رسانی ستون Loghat در هر جایی که ستون ID آن مساوی با 1 می‌باشد.

5. Drop Table Dic

   حذف جدول Dic از دیتابیس

6. ALTER TABLE Dic ADD Mani2 Text

   تغییر در دیتابیس که در این مثال Mani2 به عنوان فیلدی جدید به دیتابیس افزوده شده است.

7. CREATE UNIQUE INDEX hamseda ON Dic(ID, Loghat)

   ایجاد یک ایندکس با نام hamseda بر روی جدول Dic از دو فیلد ID, Loghat

SQLite3Connection:rollback(): این تابع برای بازگشت دادن عملیات به عقب به کار می‌رود و به همین صورت به کار می‌رود.

مثال:

```
result, err = SQLite3Connection:rollback();
```

این دستور زمانی کاربرد پیدا می‌کند که برنامه در حین انجام دستورات به مشکلی برخورد می‌کند و چنانچه عملیات را ادامه دهد دیتابیس نقص پیدا می‌کند به همین دلیل بازگشت به عقب بهترین عمل ممکن می‌باشد این دستور برای AMS بر روی تمام سیستم‌ها کار نمی‌کند

و برای استفاده از این تابع بایستی transaction را در دیتابیس فعال کنید و در عوض ما از دستورات خود AMS برای بازگشت استفاده خواهیم نمود. مثل:

```
Application.ExitScript();
```

حال اگر سیستم شما از این تابع پشتیبانی کرد result یعنی متغیر مربوط به دستور که نتیجه را در خود می‌ریزد در صورت موفقیت true را بر می‌گرداند و اگر ناموفق باشد false را بر می‌گرداند.

SQLite3Connection:setautocommit(boolean): این تابع برای تنظیم اتوماتیک حالت commit که توضیح داد شده به کار می‌رود و آن را روشن یا خاموش می‌کند.

SQLite3Cursor:fetch([table[,modestring]]): این تابع برای فراخوانی داده‌ها تحت شرایط خاص به کار می‌رود.

شرح و مثال:

Cursor یا کرسر: هنگام استفاده از دستورات SQLite نظیر Select کلیه رکوردهای درخواستی به طور کامل استخراج می‌گردد اما در مواردی نیاز است که رکوردهای استخراج شده تحت شرایطی خاص مورد پردازش مجدد قرار گرفته و به برنامه درخواست کننده ارسال گردد در این صورت استفاده از کرسرها بسیار حائز اهمیت خواهد بود در واقع برای استفاده از یک کرسر می‌توان به ترتیب مراحل عمل نمود.

این تابع به همراه دو پارامتر a و n به کار می‌رود که به شکل زیر نوشته می‌شوند:

```
row = SQLite3Cursor:fetch({},"a") ;
```

داده‌های درخواستی را به صورت یک آرایه برمی‌گرداند.

چنانچه بخواهید مقادیر این تابع را مشاهده کنید می‌بایست آن را به صورت زیر فراخوانی کنید:

```
row = SQLite3Cursor:fetch({},"a")
if row ~= nil then
M_text = row.Mani → ( Mani = نام فیلد)
Input.SetText("Input2", M_text)
row = SQLite3Cursor:fetch(row,"a");
```

```
end
row = SQLite3Cursor:fetch({},"n");
```

تعداد داده‌های بازگشتی درخواست شده را بر می‌گرداند.

چنانچه بخواهید مقدار این تابع را مشاهده کنید می‌بایست آن‌را به صورت زیر فراخوانی کنید:

```
count_row = SQLite3Cursor:fetch({},"n")
count = count_row[1]
```

چون مقدار بازگشتی به صورت آرایه می‌باشد می‌بایست شماره یا اندیس سطر آن را بنویسم که 1 می‌باشد.

یک شکل دیگر نیز از این تابع وجود دارد که بدون پارامترهای {} به کار می‌رود و برای رفتن به شماره بعدی در آرایه‌ی فراخوانی شده به کار می‌رود که به این شکل است:

```
row = SQLite3Cursor:fetch(row,"a");
```

برای بستن این تابع هم باید از دستور زیر استفاده شود:

```
SQLite3Cursor:close();
```

SQLite3Cursor:getcolnames():این تابع برای فراخوانی نام جدول‌هایی که به آن‌ها متصل شده‌ایم به کار می‌رود و نام آن‌ها را به صورتی آرایه فراخوانی می‌کند.

شرح و مثال کامل:

برای استفاده از این تابع بایستی ابتدا به دیتابیس مورد نظر وصل شده و سپس به جدول‌های مورد نظر وصل شوید و در نهایت برای اینکه بخواهید نام آن جدول‌ها را بگیرید از تابع بالا استفاده نمایید که مثال آن در زیر آمده است:

```
SQLite3Connection, err = SQLite3:connect(.."\\ hafez.db3", "", "");
SQLite3Cursor, err = SQLite3Connection:execute("SELECT * FROM Dic");
Column_name = SQLite3Cursor:getcolnames();
Debug.ShowWindow(true);→نمایش پنجره ای متنی جهت نمایش نام ستون‌ها
for index , name in pairs(Column_name) do
```

```
Debug.Print(index..":"..name.."\r\n");→ فرستادن نام ستون‌ها به پنجره‌ی متنی
end
SQLite3Cursor:close();
SQLite3Connection:close();
```

SQLite3Cursor: getcoltypes(): این تابع برای فراخوانی نوع ستون جدول‌هایی که به آن‌ها متصل شده‌ایم به کار می‌رود و نام آن‌ها را به صورتی آرایه فراخوانی می‌کند.

مثال کامل از تابع:

```
SQLite3Connection, err = SQLite3:connect(.."\\ hafez.db3", "", "");
SQLite3Cursor, err = SQLite3Connection:execute("SELECT * FROM Dic");
Column_name = SQLite3Cursor:getcoltypes();
Debug.ShowWindow(true);
for index , type  in pairs(Column_name) do
Debug.Print(index..":"..type .."\r\n");
end
SQLite3Cursor:close();
SQLite3Connection:close();
```

SQLite3Cursor: close(): این تابع برای بستن Cursor (کرسر) به کار می‌رود که توضیح و مثال آن در تابع شماره 8 و 9 و 10 آمد.

سؤال و جواب برای درک کامل توابع مهم

تا اینجا تقریباً دستورات کاربردی را فراگرفته‌اید پس لازم است یک مثال نسبتاً کامل ارایه شود تا هم دستورات قبلی مرور شوند و هم با کاربرد Cursor آشنا شوید:

سؤال 1: یک دیتابیس ایجاد کنید با نام hafez.db3 و در آن یک جدول با نام Dic ایجاد کنید با سه فیلد با نام‌های:

```
ID(Integer Primary Key) , Loghat(Char) , Mani(Char)
```

جواب سؤال 1 :

برای این کار می‌توانیم یک پروژه جدید با نام SQLite ایجاد کنیم و کد زیر را در رویداد On Click یک دکمه‌ی xButton بنویسیم و پس از اجرا، روی آن کلیک کنیم تا دیتابیس با مشخصات مورد نظر در مسیر پروژه ایجاد گردد:

```
SQLite3Connection, err =
SQLite3:connect(_SourceFolder.."\\hafez.db3", "", "");
if not SQLite3Connection or err then
Dialog.Message("Error", err);
end
SQLite3Cursor, err = SQLite3Connection:execute("CREATE TABLE
Dic(ID INTEGER PRIMARY KEY, Loghat Char, Mani Char)");
if not SQLite3Cursor or err then
Dialog.Message("Error", err);
end
SQLite3Connection:close();
```

سؤال 2: دو شی Input و دو شی Button به پروژه اضافه کنید و متن شی Button1 را برابر با Add و متن Button2 را برابر با Find قرار دهید.

حال در رویداد On Click شی Button1 کدی بنویسید که متن Input1 را در فیلد Loghat و متن Input2 را در فیلد Mani ذخیره کند و ID هم به طور خودکار شماره خواهد گرفت.

جواب سؤال 2 :

در جواب این سؤال شما را با یک نکته دیگر نیز آشنا خواهیم کرد که مربوط به مشکلات زبان فارسی در حروف ک و ی که هر کدام در سیستم عامل‌های مختلف در یک کلید به دو صورت تعریف شده‌اند و این جستجوی ما را با مشکلاتی رو به رو خواهد کرد که یک برنامه نویس خوب باید بتواند این مسایل را حل کند.

شکل‌های ک و ی: ک، ك | ی، ي

شما برای حل این مشکل باید به برنامه بگویید هر صورت از شکل این حروف را که دید به یک شکل تبدیل نماید مثلاً ک را به ک کند و سپس به دیتابیس بفرستد.

البته ما در ویندوز 7 و در AMS 8 تست کردیم و دیدیم دیگر در حرف ی مشکلی پیش نمی‌آید به طوری که AMS خود به طور اتوماتیک هر دو نوع ی را به یک ی تبدیل می‌نماید و این در حالی بود که وقتی با Notepad خود ویندوز تست کردیم هنوز این مشکل وجود داشت. پس ما در AMS با ی مشکلی نداریم و مشکل فقط در حرف ک می‌باشد که شکل استاندارد آن ک می‌باشد که در کدهای زیر آمده است:

```
word = Input.GetText("Input1");
word = String.Replace(word, "ک" ,"ک", false);
explain = Input.GetText("Input2");
explain = String.Replace(explain, "ک" ,"ک", false);
if word ~= "" then
SQLite3Connection, err =
SQLite3:connect(_SourceFolder.."\\hafez.db3", "", "");
if not SQLite3Connection or err then
Dialog.Message("Error", err);
Application.ExitScript();
end
SQLite3Cursor, err = SQLite3Connection:execute("INSERT INTO
Dic(Loghat, Mani) VALUES ('"..word.."','"..explain.."')");
 if not SQLite3Cursor or err then
Dialog.Message("Error", err);
Application.ExitScript()
end
SQLite3Connection:close();
Dialog.Message("اطلاع رسانی", "لغت جدید با موفقیت ثبت شد");
else
Dialog.Message("خطا", "کادر لغت نباید خالی باشد.");
end
```

بررسی: ابتدا متن موجود در Input1 و Input2 را در متغیرهای word و explain ریختیم و سپس در هر دو متغیر با استفاده از دستور String.Replace ک را با ک جایگزین نمودیم.

```
word = Input.GetText("Input1");
word = String.Replace(word, "ک", "ک", false);
explain = Input.GetText("Input2");
explain = String.Replace(explain, "ک", "ک", false);
```

سپس به دیتابیس متصل شدیم :

```
SQLite3Connection, err =
SQLite3:connect(_SourceFolder.."\\hafez.db3", "", "");
```

چنانچه برنامه در اتصال به دیتابیس با خطایی برخورد نماید این دستور یک پیام خطا صادر می‌کند و ادامه‌ی کار را لغو می‌کند:

```
if not SQLite3Connection or err then
Dialog.Message("Error", err);
Application.ExitScript();
end
```

هم اکنون دستورات لازم را به دیتابیس ارسال نمودیم.

نکته: آن چه که در کدها اهمیت به سزایی دارد وارد کردن اطلاعات گرفته شده از متغیرهای مربوط به دو تا input می‌باشد که می‌بایست به صورت رشته‌ای به دیتابیس ارسال شوند که برای این منظور متغیرهای word و explain بین دو علامت " قرار داده‌ایم:

```
SQLite3Cursor, err = SQLite3Connection:execute("INSERT INTO
Dic(Loghat, Mani) VALUES ('"..word.."','"..explain..."')");
```

چنانچه برنامه در ارسال اطلاعات به دیتابیس با خطایی برخورد نماید این دستور یک پیام خطا صادر می‌کند و ادامه‌ی کار را لغو می‌کند:

```
if not SQLite3Cursor or err then
Dialog.Message("Error", err);
```

```
Application.ExitScript()
end
```

سپس در رویداد On Click مربوط به شی Button2 کد زیر را بنویسید:

```
Page.Jump("Page2");
```

این کد برای رفتن به صفحه‌ی 2 به کار می‌رود.

سؤال 3: حال یک صفحه‌ی دیگر به پروژه اضافه کنید(منوی Page>Add) و در آن دو شی input و یک شی ListBox و یک شی xButton اضافه کنید ؛ متن xButton را برابر Back قرار دهید و کاری کنید که با کلیک روی آن به صفحه‌ی قبل برگردید. حال در قسمت On Key مربوط به شی input1 کدی بنویسید که هر وقت شما در input1 حرفی را تایپ می‌کنید برنامه به دیتابیس وصل شود و آن حرف را در فیلد لغت بیابد و نتیجه را در ListBox1 بریزد و با انتخاب از ListBox1 معنی آن لغت هم در input2 ظاهر شود.

جواب سؤال 3:

کد زیر را در رویداد On Click شی xButton بنویسید:

```
Page.Jump("Page1");
```

کد زیر در رویداد On Key از input1 :

```
search = Input.GetText("Input1")
if search ~= ""  then
ListBox.DeleteItem("ListBox1", -1)
added_item = 0;
search = String.Replace(search, "ک" ,"ک", false);
search = search.."%"
SQLite3Connection, err =
SQLite3:connect(_SourceFolder.."\\hafez.db3", "", "");
 if not SQLite3Connection or err then
 Dialog.Message("Error", err);
 Application.ExitScript();
 end
```

```
        SQLite3Cursor, err = SQLite3Connection:execute("SELECT * from Dic where Loghat like ""..search.."""");
        if not SQLite3Cursor or err then
        Dialog.Message("Error", err);
         Application.ExitScript();
        end
        row = SQLite3Cursor:fetch({},"a");
        if row ~= nil then
                while row do
                ListBox.AddItem("ListBox1", row.Loghat, row.ID)
                added_item = added_item +1
                row = SQLite3Cursor:fetch(row,"a");
                end
        end
        SQLite3Cursor:close();
        SQLite3Connection:close();
else
ListBox.DeleteItem("ListBox1", -1);
end
```

بررسی: ابتدا مقدار Input1 را در متغیر search می‌ریزم و سپس در صورت نامساوی بودن مقدار آن با جای خالی ListBox1 را خالی می‌کنیم و متغیری با نام added_item تعریف می‌کنیم و مقدار آن را 0 قرار می‌دهیم. سپس مقدار search را با دستور String.Replace مورد بررسی قرار می‌دهیم و چنانچه در آن ک یافته شد ان را با ک جایگزین می‌کنیم.

```
search = Input.GetText("Input1");
if search ~= ""  then
ListBox.DeleteItem("ListBox1", -1);
added_item = 0;
search = String.Replace(search, "ک" ,"ک", false);
```

در ادامه‌ی به متغیر search یک کاراکتر "%" اضافه می‌کنیم که در جستجو به معنای آن می‌باشد که تمام مواردی را که حرف اولشان مشابه مقدار search می‌باشد پیدا کند.

```
search = search.."%"
```

سپس به دیتابیس متصل می‌شویم و دستورات مورد نظر را به آن می‌فرستیم که با کاربرد این دستورات آشنا شده‌اید:

```
SQLite3Connection, err =
SQLite3:connect(_SourceFolder.."\\hafez.db3", "", "");
if not SQLite3Connection or err then
Dialog.Message("Error", err);
Application.ExitScript();
end
SQLite3Cursor, err = SQLite3Connection:execute("SELECT * from
Dic where Loghat like "..""..search.."");
if not SQLite3Cursor or err then
Dialog.Message("Error", err);
 Application.ExitScript();
end
```

پس از اتصال به دیتابیس و فرستادن دستور جستجو به آن تمام مواردی را که یافت شده‌اند فراخوانی می‌کنیم:

```
row = SQLite3Cursor:fetch({},"a");
```

پس از فراخوانی موارد یافت شده ابتدا بررسی می‌کنیم که آیا موردی فراخوانی شده است یا خیر؟

```
if row ~= nil then
```

در صورت صحیح بودن شرط با استفاده از یک دستور while به برنامه می‌گوییم تا زمانی که موارد فراخوانی شده موجود می‌باشند یکی یکی آن‌ها را به ListBox1 اضافه کن به طوری که متن موجود در ستون Loghat به عنوان ItemText و شماره موجود در ستون ID به عنوان ItemData به ListBox افزوده شوند.

```
while row do
ListBox.AddItem("ListBox1", row.Loghat, row.ID)
```

و هم چنین در هر بار تکرار شدن حلقه‌ی while یک عدد به مقدار متغیر added_item اضافه گردد تا اینکه آخر سر بدانیم چند مورد به ListBox افزوده شده است.

```
added_item = added_item +1;
```

در انتهای حلقه نیز دستوری می‌نویسیم که حلقه به ردیف بعدی اطلاعات فراخوانی شده از دیتابیس برود.

```
row = SQLite3Cursor:fetch(row,"a");
```

و در آخر حلقه را می‌بندیم و سپس if را نیز می‌بندیم و پرس و جو و اتصال به دیتابیس را می‌بندیم.

```
end
end
SQLite3Cursor:close();
SQLite3Connection:close();
```

و چنان چه شرط اول ما نادرست باشد لیست خالی می‌گردد و if بسته می‌شود.

```
else
ListBox.DeleteItem("ListBox1", -1);
end
```

کد زیر در رویداد on select در ListBox1 :

```
select = ListBox.GetSelected("ListBox1");
if select ~= nil then
search = ListBox.GetItemData("ListBox1", select[1]);
        SQLite3Connection, err =
SQLite3:connect(_SourceFolder.."\\hafez.db3", "", "");
        if not SQLite3Connection or err then
        Dialog.Message("Error", err);
         Application.ExitScript();
        end
        SQLite3Cursor, err = SQLite3Connection:execute("SELECT
Mani from Dic where ID = ""..search..""");
        if not SQLite3Cursor or err then
        Dialog.Message("Error", err);
         Application.ExitScript()
        end
```

```
        row = SQLite3Cursor:fetch({},"a")
        if row ~= nil then
        M_text = row.Mani
        Input.SetText("Input2", M_text)
        row = SQLite3Cursor:fetch(row,"a");
        end
        SQLite3Cursor:close()
        SQLite3Connection:close()
L_text = ListBox.GetItemText("ListBox1", select[1])
Input.SetText("Input1", ListBox.GetItemText("ListBox1", select[1]))
L_ID = ListBox.GetItemData("ListBox1", select[1])
end
```

بررسی: ابتدا با دستور ListBox.GetSelected بررسی می‌کنیم که آیا گزینه ای از لیست انتخاب شده است یا خیر؟

```
select = ListBox.GetSelected("ListBox1");
```

این دستور شماره‌ی گزینه‌ی انتخاب شده از لیست را بر می‌گرداند و به صورت آرایه می‌باشد پس می‌بایست برای دریافت شماره‌ی گزینه متغیر را در هنگام استفاده به صورت زیر بنویسیم:

```
select[1]
```

اگر جواب شرط صحیح باشد شماره‌ی ID که به قسمت ItemData لیست اضافه کرده‌ایم را در متغیری به نام search می‌ریزیم.

```
if select ~= nil then
search = ListBox.GetItemData("ListBox1", select[1]);
```

سپس به دیتابیس مورد نظر متصل می‌شویم و دستوری به آن می‌فرستیم که مقدار ستون Mani را در هر ردیفی که مقدار ستون ID آن برابر با search است پیدا کند و در آرایه‌ی row بریزد.

```
SQLite3Connection, err = SQLite3:connect(_SourceFolder
.."\\hafez.db3", "", "");
```

```
if not SQLite3Connection or err then
Dialog.Message("Error", err);
 Application.ExitScript();
end
SQLite3Cursor, err = SQLite3Connection:execute("SELECT Mani from Dic where ID = ""..search..""");
if not SQLite3Cursor or err then
Dialog.Message("Error", err);
 Application.ExitScript();
end
row = SQLite3Cursor:fetch({},"a");
```

سپس بررسی می‌کنیم که اگر row برابر با nil یعنی هیچ نبود مقدار متن فراخوانی شده از ستون Mani را در متغیری به نام M_text بریزد و سپس در Input2 نمایش دهد و بعد if را می‌بندیم:

```
if row ~= nil then
M_text = row.Mani
Input.SetText("Input2", M_text)
row = SQLite3Cursor:fetch(row,"a");
end
```

سپس اتصال‌ها را نیز می‌بندیم

```
SQLite3Cursor:close();
SQLite3Connection:close();
```

و برای اینکه متن لغت انتخاب شده از لیست ما هم در Input1 نوشته شود از کد زیر کمک گرفتیم:

```
L_text = ListBox.GetItemText("ListBox1", select[1]);
Input.SetText("Input1", L_text);
```

امنیت در دیتابیس SQLite:

دیتابیس SQLite هیچ ابزاری جهت حفاظت از خود دارا نیست و اطلاعات آن به راحتی قابل دسترسی می‌باشد؛ حال این سؤال پیش می‌آید که برای حفظ کردن اطلاعات خود چه کنیم؟ برای حفظ اطلاعات دیتابیس مان مجبور هستم از روش‌های حافظت فایل‌ها توسط AMS بهره ببریم

حال دو روش برای حفاظت فایل‌های دیتابیس معرفی می‌گردد:

1. استفاده از روش رمزنگاری با تابع Crypto
2. استفاده از روش رمزگذاری با تابع Zip

در هر دو روش بالا شما می‌بایستی ابتدا با استفاده از توابع ذکر شده فایل دیتابیس خود را رمزنگاری و رمز گذاری کنید و سپس در هنگام نیاز به استفاده از دیتابیس آن را از حالت رمزی شده در آورید و در یکی از پوشه های سیستم عامل مثل Temp استخراج نمایید و سپس عملیات مورد نظرتان را انجام دهید و پس از انجام عملیات دوباره فایل را دیتابیس را اگر نیاز است رمزنگاری کرده و جانشین فایل قبلی نمایید

مثال برای استفاده از روش رمزنگاری با تابع Crypto:

رمز نگاری کردن:

```
Crypto.BlowfishEncrypt(_SourceFolder.."\\Mydb.db3",_TempFolder..
"\\Mydb.db3", "MyPassword");
```

استخراج از حالت رمز نگاری شده با تابع Crypto:

```
Crypto.BlowfishDecrypt(_SourceFolder.."\\Mydb.db3",
_TempFolder.. "\\Mydb.db3", "MyPassword");
```

مثال برای استفاده از روش رمزگذاری با تابع Zip:

```
Zip.Add(_SourceFolder.."\\Mydb.zip",
{_SourceFolder.."\\Mydb.db3"}, true, "MyPassword", 5, nil, true);
```

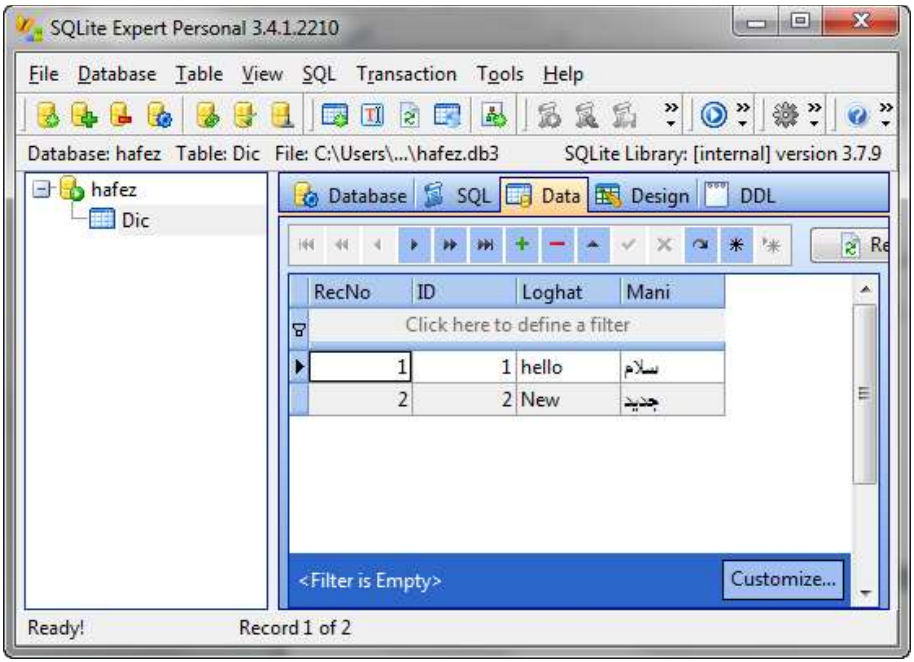
استخراج از حالت رمز شده با تابع Zip:

```
Zip.Extract(_SourceFolder.."\\Mydb.zip", {"*.*"},
_SourceFolder.."\\", true, true, "", ZIP_OVERWRITE_ALWAYS,
nil);
```

تا اینجا با دستورات مهم و کاربردی دیتابیس SQLite آشنا شده‌اید این دیتابیس یک دیتابیس قدرتمند و بی نیاز به نصب برنامه ای خاص برای اجرای خود می‌باشد اما باید بدانید که این دیتابیس یک دیتابیس تحت ویندوز بوده و برای استفاده در دیتابیس‌های آنلاین اینترنتی مناسب نیست.

آموزش نرم افزار SQLite Expert Personal

یکی از رایج‌ترین نرم افزارها جهت کار با دیتابیس SQLite نرم افزار SQLite Expert Personal که به وسیله آن می‌توان دیتابیس مورد نظر را ویرایش نمود و یا دیتابیس‌های جدیدی ساخت.



تصویر 7 - 3

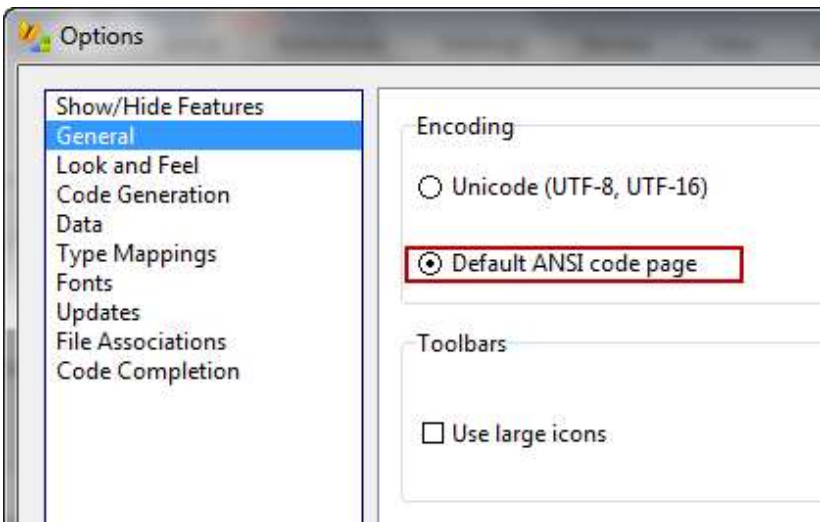
در این قسمت از کتاب کار با قسمت‌های مهم این نرم افزار را به طور خلاصه می‌آموزید

برای دریافت این نرم افزار می‌توانید به سایت سازنده آن مراجعه نمایید و نسخه رایگان ان را دریافت کنید.

آدرس سایت: http://www.sqliteexpert.com

پس از نصب نرم افزار آن را اجرا کنید و سپس برای تنظیم آن مطابق با استاندارد متنی ANSI مراحل زیر را دنبال کنید:

1. از منوی Tools به گزینه Option بروید.
2. در سربرگ General و در قسمت Encoding گزینه‌ی Default ANSI Code Page را علامت گذاری کنید و دکمه‌ی Ok را کلیک نمایید:



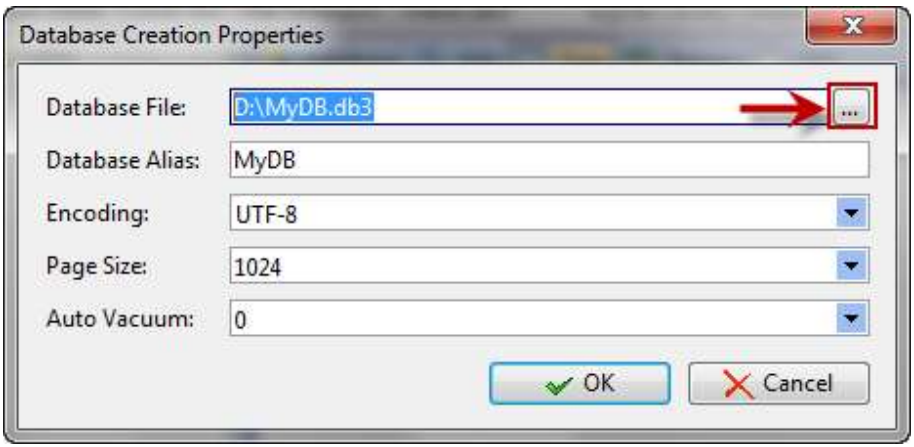
تصویر 7 - 4

روش ساخت یک دیتابیس

برای ساخت یک دیتابیس با نرم افزار SQLite Expert Personal به روش زیر عمل نمایید:

1. از منوی File گزینه‌ی New DataBase را کلیک کنید.

2. در پنجره ظاهر شده روی دکمه‌ی مشخص شده در تصویر زیر کلیک کنید و سپس در پنجره باز شده محل ذخیره و نامی را برای دیتابیس خود مشخص کنید و روی دکمه‌ی Open کلیک کنید و سپس در پنجره DataBase Creation Properties روی دکمه‌ی Ok کلیک کنید.



تصویر 7 - 5

با این کار فایل دیتابیس شما در مسیر مورد نظر ساخته می‌شود و می‌توانید روی آن جدول‌هایی (Table) ایجاد کنید.

ایجاد جدول بر روی دیتابیس

برای ایجاد جدول بر روی دیتابیس ساخته شده به روش زیر عمل کنید:

1. از منوی Table روی گزینه‌ی New Table کلیک کنید و سپس سربرگ ظاهر شده Design در قسمت Table Name نامی را برای جدول خود وارد نمایید برای مثال: My_Table.

2. هنوز جدول ما ایجاد نشده است زیرا جدول بدون ستون یا فیلد ایجاد نمی‌گردد. برای افزودن ستون به جدول My_Table در قسمت سربرگ Fields روی دکمه‌ی Add کلیک کنید و پس از وارد کردن نام و نوع آن ستون را ایجاد کنید:

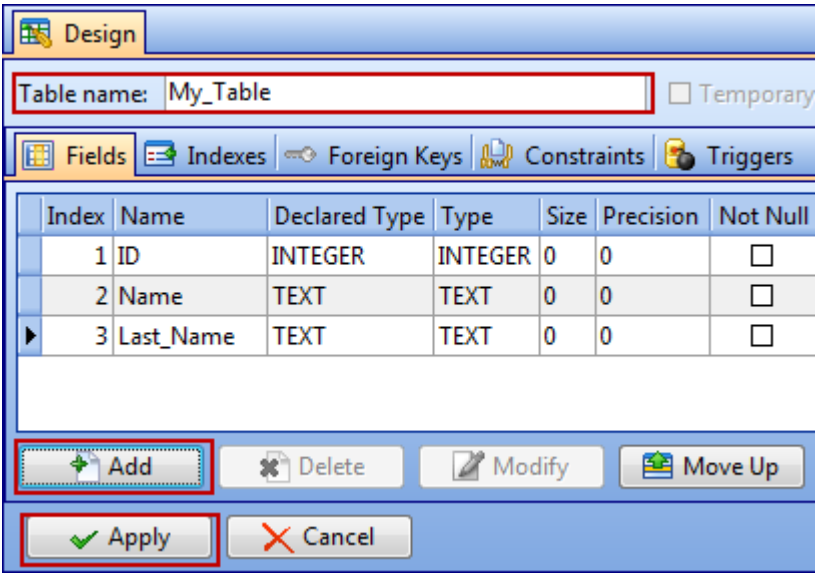
برای مثال ستون‌های زیر ایجاد نمایید:

ستون با نام ID و قسمت Type (نوع) آن را برابر با INTEGER قرار دهید.

ستون با نام Name و قسمت Type (نوع) آن را برابر با TEXT قرار دهید.

ستون با نام Last_Name و قسمت Type (نوع) آن را برابر با TEXT قرار دهید.

و در آخر می‌بایست برای ایجاد شدن این ستون‌ها روی دکمه‌ی Apply کلیک نمایید.

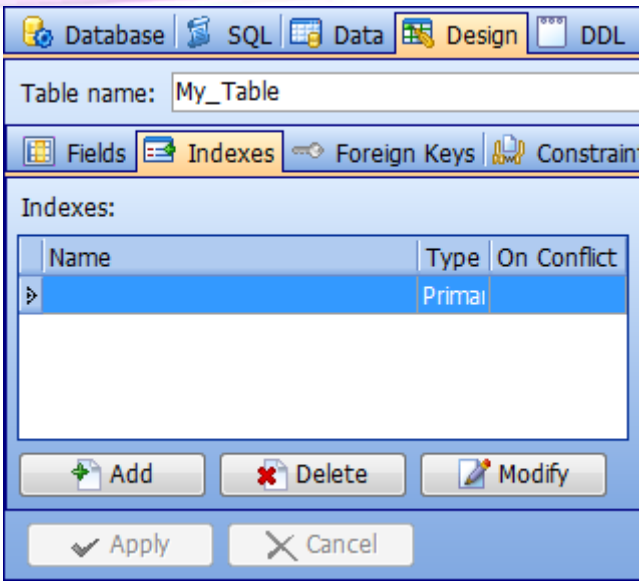


تصویر 7 - 6

با کلیک روی دکمه‌ی Apply جدول ما بر و ری دیتابیس ایجاد می‌گردد. که می‌توانیم بر روی آن اطلاعاتی را ذخیره نماییم.

بهتر است ستون ID را به عنوان کلید تعیین کنید برای این کار به سربرگ Indexs بروید و سپس روی دکمه‌ی Add کلیک نمایید، در پنجره‌ی ظاهر شده ستون ID را از لیست انتخاب کنید، گزینه‌ی Primary را علامت بزنید، سپس روی دکمه‌ی Add کلیک کنید، Ok را کلیک کنید و در آخر روی دکمه‌ی Apply کلیک نمایید تا تنظیمات اعمال گردند.

این کار باعث می‌شود تا ستون ID به عنوان کلید جدول در نظر گرفته شود خودکار شماره گذاری گردد.



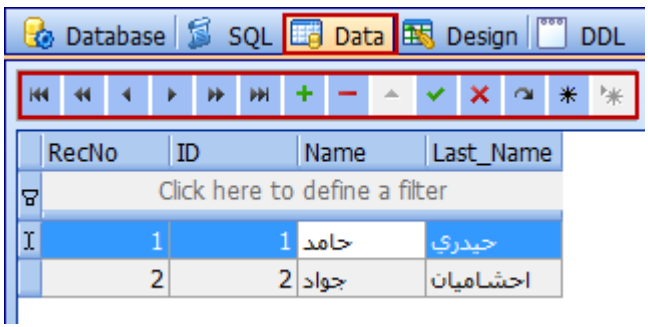
تصویر 7 - 7

وارد کردن اطلاعات در جدول

برای وارد نمودن اطلاعات در جدول دیتابیس می‌توانید از روش زیر استفاده کنید:

1. بر روی سربرگ Data کلیک کنید و سپس با استفاده از دکمه‌ی + اطلاعات خود را در سلول‌های هر ستون وارد کنید .

2. در آخر روی دکمه‌ی ✓ کلیک نمایید تا اطلاعات شما ذخیره شوند.

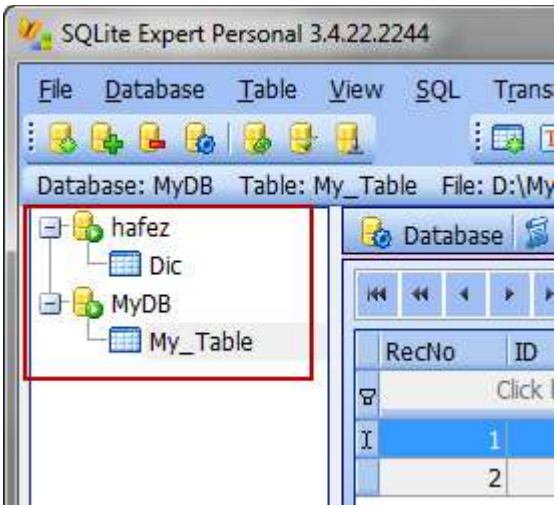


تصویر 7 - 8

باز کردن یک دیتابیس ایجاد شده

برای باز کردن یک دیتابیس ایجاد شده به منوی File بروید و گزینه‌ی Open Database را کلیک کنید، از مسیر مورد نظر دیتابیس خود را انتخاب کنید و سپس روی دکمه‌ی Open کلیک کنید.

دیتابیس باز شده در قسمت مشخص شده‌ی تصویر زیر قابل مشاهده می‌باشد و جدول‌های موجود در آن نیز به صورت درخت واره در آن جای گرفته‌اند که با انتخاب آن‌ها می‌توان در قسمت سربرگ Data اطلاعات آن‌ها را مشاهده نمود.



تصویر 7 - 9

این آموزشی بود خلاصه برای کار کردن با نرم افزار SQLite Expert Personal تا در مواقع لازم بتوانید فایل‌های دیتابیس SQLite خود را به صورت دستی و سریع ویرایش کنید و یا اقدام به ایجاد دیتابیس‌های جدید از این نوع بنمایید.

دیتابیس MySQL

دیتابیس MySQL دیتابیسی است متن باز، قدرتمند و پرکاربرد که برای استفاده‌ی آنلاین و تحت شبکه مناسب می‌باشد.

نسخه های اولیه MySQL را به هیچ عنوان نمی‌توان با نسخه های جدید آن مقایسه نمود. نسخه های قدیمی به دلیل ماهیت عمومی سیستم عاملی که برای آن در نظر گرفته شده بودند (یعنی یونیکس و لینوکس‌های اولیه) دارای واسط کاربر چندان جالبی نبودند و تمام فرامین مربوط به طراحی و مدیریت بانک اطلاعاتی در آن‌ها از طریق دستورات خط فرمان انجام می‌گرفت. اما به تدریج و با پیدایش محیط‌های گرافیکی توانمند و زیبا برای لینوکس، MySQL نیز همانند سایر نرم افزارهای متن باز تحت لینوکس مراحل تکامل و بهینه شدن هسته و ابزارهای جانبی خود را پیمود تا به جایی رسید که اکنون به عنوان یکی از سریع‌ترین، کاراترین و مقرون به صرفه‌ترین برنامه‌های بانک اطلاعاتی جهان شناخته می‌شود.

در این قسمت با نحوه‌ی کار با دیتابیس MySQL به وسیله‌ی AMS آشنا خواهید شد. بهتر است بدانید که دستورات موجود در دیتابیس SQLite در دیتابیس MySQL نیز قابل استفاده می‌باشند و به عبارتی دستورات مشترک زیادی دارند و هر آنچه که در رابطه با دستورات SQLite توضیح داده شد در مورد این دیتابیس نیز صدق می‌کند مگر در مواردی اندک؛ پس به همین سبب از ذکر مجدد آن‌ها خودداری می‌کنیم

و برای اینکه سریع‌تر بتوانید کار با این دیتابیس را فرا بگیرید با یک مثال عملی و سپس بررسی آن به آموزش این دیتابیس می‌پردازیم.

❖ **پیش نیازها:**

1. فعال سازی موتور دیتابیس: ابتدا از منوی Project>Databases... دیتابیس MySQL را علامت گذاری کنید تا توابع این دیتابیس در پروژه شما فعال گردند.
2. داشتن یک دیتابیس آنلاین از نوع MySQL
3. در اختیار داشتن نام کاربری و کلمه‌ی عبور دیتابیس(User Name & Password)
4. فعال بودن قابلیت کنترل از راه دور در دیتابیس(Remote).

امروزه اکثر هاست‌های خریداری شده دارای دیتابیس MySQL می‌باشند و قابلیت کنترل از راه دوره نیز در اکثر آن‌ها وجود دارد ولی احتمال آن نیز زیاد است که این قابلیت فعال نباشد، برای فعال کردن قابلیت کنترل از راه دور در پنل مدیریت هاست خود به قسمت مدیریت دیتابیس بروید و سپس روی گزینه‌ی کنترل از راه دور(Remote) کلیک کنید و سپس در

صفحه‌ی ظاهر شده در کادر متنی موجود عبارت ٪ وارد کنید و سپس روی دکمه‌ی افزودن (Add یا Allow) کلیک کنید تا قابلیت کنترل از راه دور دیتابیس شما فعال گردد.

در مثال زیر شما نحوه‌ی کار کردن با دیتابیس MySQL را فرا خواهید گرفت و با تفاوت‌های جزیی آن با SQLite نیز آشنا خواهید شد.

مثال: کار با دیتابیس MySQL (دفتر تلفن ساده)

در این مثال فرض می‌کنیم شما یک دیتابیس آنلاین با مشخصات زیر دارید:

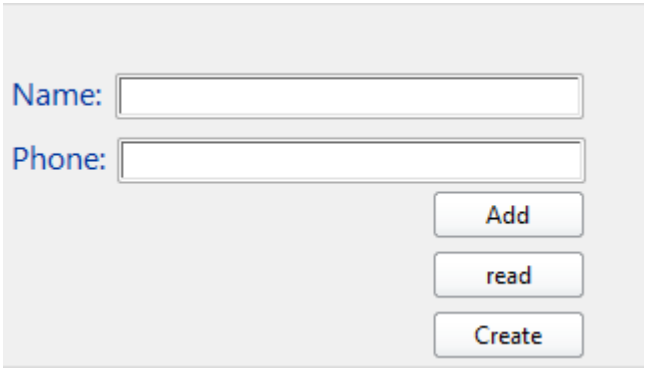
نام(DataBase): Mydb

نام کاربری(User Nmae): hamseda

کلمه‌ی عبور(Password): vasva3

نام هاست(Host Name) یا ip: www.vasva3.com

1. یک پروژه‌ی جدید با نام MySQL ایجاد کنید، اندازه‌ی آن را برابر با 320x180 قرار دهید و سه شی xButton و دو شی Input و دو شی Label به آن اضافه نمایید و آن‌ها را مرتب بچینید.(تصویر 7 -2)



تصویر 7 - 10

2. روی xButton1 دو بار کلیک کنید و خاصیت Text آن را برابر با Create قرار دهید و سپس کد زیر را در رویداد On Click آن وارد کنید:

```
MySQLConnection, err = MySQL:connect("Mydb", "hamseda",
"vasva3", "www.vasva3.com");
```

```
if not MySQLConnection or err then
   Dialog.Message("Error1", err);
      Application.ExitScript();
 end
MySQLCursor, err = MySQLConnection:execute("Create Table
MyTable(ID int Primary Key AUTO_INCREMENT,Name
Text,Phone Text)");
 if not MySQLCursor or err then
   Dialog.Message("Error2", err);
      Application.ExitScript();
 end
 result, err = MySQLConnection:close();
Dialog.Message("Success", "OK");
```

3. روی xButton2 دو بار کلیک کنید و خاصیت Text آن را برابر با Add قرار دهید و سپس کد زیر را در رویداد On Click آن وارد کنید:

```
Name = Input.GetText("Input1");
Phone = Input.GetText("Input2");
MySQLConnection, err = MySQL:connect("Mydb", "hamseda",
"vasva3", "www.vasva3.com");
if not MySQLConnection or err then
   Dialog.Message("Error1", err);
      Application.ExitScript();
 end
                  MySQLCursor, err =
MySQLConnection:execute("INSERT INTO MyTable(Name,Phone)
VALUES ('"..Name.."','"..Phone.."')");
         if not MySQLCursor or err then
```

```
        Dialog.Message("Error2", err);
        Application.ExitScript();
        end
 MySQLConnection:close();
Dialog.Message("Success", "OK");
```

4. روی xButton3 دو بار کلیک کنید و خاصیت Text آن را برابر با Read قرار دهید و سپس کد زیر را در رویداد On Click آن وارد کنید:

```
MySQLConnection, err = MySQL:connect("Mydb", "hamseda", "vasva3", "www.vasva3.com");
if not MySQLConnection or err then
   Dialog.Message("Error1", err);
     Application.ExitScript();
 end
 MySQLCursor, err = MySQLConnection:execute("select * from MyTable");
 if not MySQLCursor or err then
   Dialog.Message("Error2", err);
   Application.ExitScript();
end
Debug.ShowWindow(true);
Name_Phone = "";
for i = 1, MySQLCursor:numrows() do
   local ID,Name,Phone = MySQLCursor:fetch();
   Name_Phone = Name_Phone.."ID: "..ID..", Name:  "..Name.." Phone: "..Phone.."\r\n";
end
Debug.Print(Name_Phone);
MySQLCursor:close();
```

```
MySQLConnection:close();
```

5. برنامه را اجرا کنید، ابتدا دکمه‌ی Create را کلیک کنید و پس از ظاهر شدن کادر پیغام Ok یک نام یک شماره تلفن در ورودی‌های متن وارد نمایید و روی دکمه‌ی Add کلیک نمایید و در آخر پس از اعلام نتیجه‌ی Ok روی دکمه‌ی Read کلیک نمایید تا نام و شماره‌ی ذخیره شده را مشاهده نمایید.

بررسی مثال: دستور MySQL:connect برای اتصال به دیتابیس MySQL مورد استفاده قرار می‌گیرد برای مشاهده‌ی خصوصیات این دستور روی آن دو بار کلیک کنید تا کادری مشابه به تصویر 7-3 مشاهده نمایید.

Click here to learn more about this action.

MySQL:connect	
Sourcename	"Mydb"
Username	"hamseda"
Password	"vasva3"
Hostname	"www.vasva3.com"
Port	
ResultVariable	MySQLConnection
ResultVariable	err

تصویر 7 - 11

با کد زیر نیز اتصال به دیتابیس را بررسی می‌کنیم و اگر خطایی رخ دهد آن را اعلام می‌کند و سپس از اجرای کدهای بعدی خودداری می‌کند.

```
if not MySQLConnection or err then
   Dialog.Message("Error1", err);
     Application.ExitScript();
end
```

سپس با تابع MySQLConnection:execute دستورات مورد نظر را به دیتابیس می‌فرستیم.

کد Debug.ShowWindow برای ظاهر کردن پنجره‌ی متنی مخصوص Debug به کار می‌رود.

Name_Phone نیز متغیری است که قصد داریم نتیجه‌ی فراخوانی شده‌ی متنی را در آن بریزیم.

با استفاده از کد MySQLCursor:numrows() می‌توانیم تعداد سطرهایی را در نتیجه‌ی پرس و جو در دیتابیس فراخوانده‌ایم به دست آوریم.

```
MySQLCursor, err = MySQLConnection:execute("select * from MyTable");
 if not MySQLCursor or err then
    Dialog.Message("Error2", err);
    Application.ExitScript();
end
Debug.ShowWindow(true);
Name_Phone = "";
for i = 1, MySQLCursor:numrows() do
    local ID,Name,Phone = MySQLCursor:fetch();
    Name_Phone = Name_Phone.."ID: "..ID..", Name:  "..Name.." Phone: "..Phone.."\r\n";
end
Debug.Print(Name_Phone);
```

کد MySQLCursor:fetch نیز که در حلقه‌ی for به کار رفته است جهت فراخوانی مقدار ستون‌هایی به کار می‌رود که قبل از علامت = قرار گرفته‌اند به طوری که در هر بار تکرار حلقه یک سطر به جلو می‌رود تا اینکه تعداد سطرها به پایان برسد.

کلمه‌ی local نیز نشان دهنده‌ی محلی بودن متغیرها می‌باشد متغیر محلی متغیری است که با پایان یافتن یک دستور یا رویداد، از بین می‌رود و دیگر قابل استفاده نمی‌باشد.

```
local ID,Name,Phone = MySQLCursor:fetch()
```

آخر سر هم متغیر Name_Phone را با دستور Debug.Print به پنجره‌ی متنی Debug می‌فرستیم.

بعد از این نوبت به بستن پرس و جو و اتصال دیتابیس می‌رسد:

```
MySQLCursor:close();
MySQLConnection:close();
```

به این ترتیب توانستیم به دیتابیس MySQL نیز متصل شویم و با دستورهای آن نیز آشنا شویم.

تنها دستوری که در دیتابیس MySQL وجود دارد و در دیتابیس SQLite وجود نداشت دستور MySQLCursor:getnumrows که برای گرفتن تعداد سطرهای فراخوانده شده به کار می‌رود که این هم به خاطر تفاوت در موتور این دو دیتابیس می‌باشد. به هر حال با روش‌های دیگری توانستیم تعداد سطرهای فراخوانی شده را در SQLite نیز به دست آوریم.

امید است که توانسته باشید کار کردن با دو دیتابیس مورد نظر را فراگرفته باشید. توصیه می‌شود برای کسب مهارت در کار کردن با دیتابیس‌های موجود در AMS به مطالعه و تمرین بیشتر در رابطه با این دیتابیس‌ها بپردازید.

دیتابیس‌های دیگر

دیتابیس‌های دیگری که می‌توانید در AMS با آن‌ها کار کنید عبارتند از ODBC، Oracel و PostgreSQL که هر سه دیتابیس‌های قدرتمندی هستند که کار کردن با آن‌ها تفاوت چندانی با دو دیتابیس توضیح داده شده ندارد. در انتهای مطلب فقط به آموزش نحوه‌ی اتصال به این سه دیتابیس بسنده می‌کنیم و باقی کار را به شما خوانندگان گرامی واگذار می‌نماییم.

اتصال به دیتابیس Oracel:

پیش‌نیازها:

1. فعال کردن این دیتابیس از منوی Project>Databases…

برای اتصال به این دیتابیس به می‌بایست خصوصیت sourcename را در دستور اتصال به بانک اطلاعاتی برابر با نام سرویس(service name) قرار دهید.

مثال:

```
OracleConnection, err = Oracle:connect("service_name ", "
User_Name ", " Password ");
```

اتصال به دیتابیس ODBC:

پیش‌نیازها:

1. فعال کردن این دیتابیس از منوی Project>Databases…
2. ایجاد اتصال به یک دیتابیس با استفاده از Data Sources(ODBC) ویندوز

برای اتصال به این دیتابیس به می‌بایست خصوصیت sourcename را در دستور اتصال به بانک اطلاعاتی برابر با نام DSN ایجاد شده با ODBC ویندوز قرار دهید.

مثال:

```
ODBCConnection, err = ODBC:connect("mydb", "User_Name", "Password");
```

اتصال به دیتابیس PostgreSQL(آنلاین):

1. فعال کردن این دیتابیس از منوی Project>Databases…

برای اتصال به این دیتابیس به می‌بایست خصوصیت sourcename را در دستور اتصال به بانک اطلاعاتی برابر با نام دیتابیس قرار دهید.

مثال:

```
PostgreSQLConnection, err = PostgreSQL:connect("mydb", " User_Name", " Password", "www.Your_Host_Name.com");
```

فصل هشتم: کنترل خطاهای پروژه

گرفتن خطاهای پروژه یکی از قسمت‌های مهم هر پروژه به حساب می‌آید. خطاها می‌توانند روند اجرای برنامه را دچار مشکل کنند و ارزش آن را زیر سؤال ببرند بنابراین ضروری است که برنامه نویس با نحوه‌ی خطا گیری و اشکال زدایی برنامه آشنا باشد و تا جایی که می‌تواند سعی کند برنامه‌اش بدون خطا اجرا گردد.

در AMS برای خطا گیری ابزارها و کدهایی ارایه شده است که با به کار گیری مناسب آن‌ها می‌توان از به وجود آمدن خطاها در هنگام اجرای برنامه و یا اجرای کدها جلوگیری کرد و برنامه ای بدون نقص به کاربران ارایه داد.

جدای از اینکه شما کدهایتان را تا چه اندازه خوب و مرتب نوشته باشید، ممکن است برنامه به شرایط پیش بینی نشده ای مواجه شود که باعث توقف اجرای آن شود. در این شرایط اگر کدهای شما نتوانند موقعیت به وجود آمده را کنترل کنند کاربر با پیغام خطای پیش‌فرض مواجه خواهد شد که توضیحاتی در مورد خطای به وجود آمده به کاربر ارایه خواهد داد اما کاربران هرگز نمی‌توانند دلیل این خطا را پیدا کنند و آن را رفع نمایند. در چنین موقعیت است که اهمیت خطایابی در برنامه مشخص می‌شود.

در این فصل به بررسی نحوه‌ی اشکال زدایی پروژه‌ها خواهیم پرداخت.

در این فصل به موارد زیر می‌پردازیم:

- انواع خطاها و نحوه‌ی تصحیح آن‌ها
- چگونگی یافتن خطاهای موجود در یک برنامه و تصحیح آن‌ها
- چگونگی کنترل خطاها و شرایط پیش بینی نشده

انواع خطاها:

خطاهایی که در یک برنامه رخ می‌دهند به سه دسته‌ی کلی تقسیم می‌شوند: خطاهای دستوری، خطاهای زمان اجرا و خطاهای منطقی.

خطاهای دستوری:

خطاهای دستوری[5] ساده‌ترین نوع خطاها هستند و به سادگی قابل یافتن و رفع کردن هستند. این گونه خطاها معمولاً زمانی رخ می‌دهند که کد نوشته شده توسط شما از نظر AMS خطا داشته باشد و نتواند آن را تفسیر کند. ممکن است دستوری را ناقص وارد کرده باشید، ترتیب نوشتن دستورات را رعایت نکرده باشید و مواردی از این قبیل.

برای مثال در محیط کد نویسی یک شی کد زیر را بنویسید و سپس با دکمه‌ی Ok کد را تایید نمایید:

```
Debug.ShowWindow(true);
for i , 10 do
Debug.Print(i.."\r\n");
end
```

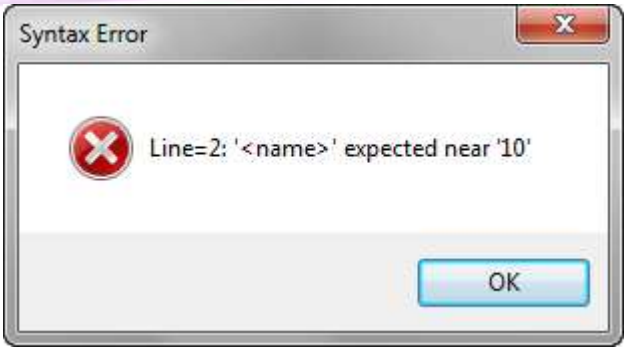
که کد سطر دوم باید به صورت زیر باشد:

```
for i=1 , 10 do
```

مشاهده می‌کنید که با یک پیام خطا از طرف AMS مواجه می‌شوید که شماره سطری را که در آن خطای دستوری وجود دارد اعلام کرده و توضیح مختصری در مورد آن بیان می‌کند.(تصویر 8- 1)

روش دیگری نیز برای مشاهده‌ی خطاهای دستوری وجود دارد و راست کلیک کردن در محیط کد نویسی و سپس انتخاب گزینه‌ی Advanced> Syntax Errors و یا فشردن کلید F12 از صفحه کلید می‌باشد.

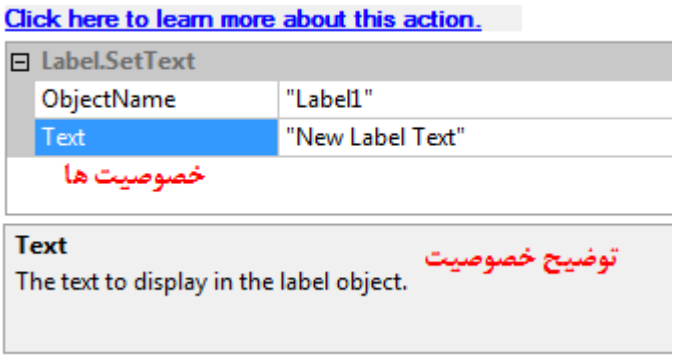
7- Syntax Errors



تصویر 8 - 1

یکی دیگر از ویژگی‌های AMS که باعث می‌شود خطاهای دستوری کمتر شوند ویژگی تکمیل خودکار متن دستورات و توابع از پیش تعریف شده می‌باشد با استفاده از این ویژگی وقتی که می‌خواهید کدی را بنویسید کادری باز می‌شود و بر اساس حروفی که وارد نموده‌اید دستور مشابه آن را نمایش می‌دهد که با فشردن کلیدهای Ctrl+Space به صورت خودکار در محیط کد نویسی افزوده می‌شود و اگر جلوی آن علامت "." بگذارید زیر مجموعه‌ی آن دستور را به شما نشان می‌دهد که می‌توانید گزینه‌ی دستور مورد نظر را انتخاب کنید تا در ادامه‌ی کد قرار گیرد(تصویر 8 - 2) و اگر از خصوصیات کد چیزی نمی‌دانید می‌توانید روی کد مورد نظر دو بار کلیک کنید تا در کادر مشابه تصویر 8- 3 خصوصیات آن دستور و هم چنین توضیحات آن‌ها را مشاهده نمایید.

تصویر 8 - 2



تصویر 8 - 3

خطاهای اجرایی

خطاهای اجرایی [6] و یا **خطاهای زمان اجرا** [7] خطاهایی هستند که در زمان اجرای یک برنامه رخ می‌دهند. این خطاها عموماً به این خاطر رخ می‌دهند که برخی از عوامل خارج از برنامه مانند کاربر، بانک اطلاعاتی، دیسک سخت و یا ... رفتار غیر قابل پیش بینی از خود بروز نمایند.

در هنگام نوشتن یک برنامه می‌بایست این مسایل را نیز مدنظر قرار داد و کد مناسبی برای کنترل این نوع خطاها نوشت. هر چند نمی‌توان از رخ دادن چنین خطاهایی در برنامه جلوگیری کرد، اما می‌توان با نوشتن کدهای مناسب برای کنترل آن‌ها، هنگام بروز چنین خطاهایی با نمایش پیغام خطای مناسب از برنامه خارج شد و یا بدون در نظر گرفتن خطا از اجرای بقیه کد صرف نظر کرد و به کاربر اطلاع داد که چگونه از بروز این خطا جلوگیری کند. نحوه‌ی کنترل خطاهای زمان اجرا در قسمت‌های بعدی این فصل شرح داده شده است.

خطاهای اجرایی معمولاً هنگام تست قسمت‌های مختلف در زمان نوشتن برنامه مشخص می‌شوند. در نتیجه بعد از تشخیص آن‌ها می‌توانید کدی بنویسید که رفتار برنامه را در آن شرایط کنترل کند و از توقف ناگهانی برنامه جلوگیری کنید. با این روش در ادامه‌ی فصل آشنا خواهید شد.

8- Execution Errors
9- Run-Time Errors

خطاهای منطقی

خطاهای منطقی[8] یا خطاهای مفهومی، خطاهایی هستند که باعث می‌شوند برنامه نتیجه‌ی نامطلوبی را ارایه کند و ریشه در اشتباهات کد نویسی برنامه نویس دارند. معمولاً بیشترین خطاهای منطقی که در یک برنامه به وجود می‌آیند، حلقه های بی نهایت هستند. برای مثال کد زیر را در نظر بگیرید:

```
index = 11;
while index < 10 do
دستورات--
index = index+1;
end
```

مقدار index در این حلقه هرگز به عدد کوچک‌تر 10 نخواهد رسید، در نتیجه برنامه در یک حلقه بی نهایت قرار می‌گیرد. این یک مثال ساده از خطاهای منطقی در برنامه بود، اما حتی برنامه‌نویسان با تجربه نیز ممکن است در هنگام نوشتن کد دچار چنین خطاهایی در برنامه‌ی خود شوند.

خطای منطقی دیگری که ممکن است در برنامه رخ دهد، خطا در مقایسه‌ها است. برای مثال می‌خواهید مقدار یک متغیر را با ورودی کاربر مقایسه کنید و در صورت برابر بودن این دو مقدار، عمل خاصی را در برنامه انجام دهید. در این شرایط معمولاً نمی‌خواهید که مقایسه نسبت به بزرگی و کوچکی حروف حساس باشد. مثل کد زیر:

```
if (strFileName == Input.GetText("Input1")) then
دستورات--
end
```

در این شرایط برای مثال اگر مقدار متغیر strFileName برابر Index.Html و مقدار موجود در Input1 برابر با index.html باشد، نتیجه‌ی مقایسه نادرست خواهد بود و کد داخل

10- Logic Errors

دستور if اجرا نخواهد شد. یکی از روش‌های جلوگیری از این خطاها این است که در ابتدا، هر دوی مقداری که می‌خواهید با هم مقایسه کنید را به حروف بزرگ و یا حروف کوچک تبدیل کنید(برای این کار می‌توانید از تابع String.Upper و String.Lower استفاده کنید). به این ترتیب اگر متنی که کاربر در ورودی متن وارد کرده است با متن موجود در متغیر strFileName برابر باشد و فقط از نظر بزرگی و کوچکی کاراکترها با هم تفاوت داشته باشند حاصل مقایسه درست خواهد بود.

نکته: از آن جایی که خطاهای منطقی در یک برنامه سخت‌ترین نوع خطاها از نظر تشخیص و رفع هستند و هم چنین ممکن است باعث توقف اجرای برنامه و یا تولید نتیجه‌ی نامطلوب توسط آن شوند باید هنگام نوشتن کد از درست بودن منطق آن اطمینان حاصل کنید و مطمئن شوید که روشی که برای اجرای این قسمت از برنامه به کار می‌برید درست است. هم چنین باید تمام خطاهایی که ممکن است به وسیله‌ی کاربر در برنامه ایجاد شود را نیز بررسی کرده و کنترل کنید. هر چقدر که بیشتر در برنامه نویسی تجربه کسب کنید، بیشتر با خطاهای عمومی و خطاهایی که ممکن است توسط کاربر ایجاد شوند آشنا خواهید شد.

کنترل خطاها در برنامه

در این قسمت از کتاب با روش‌های کنترل خطاهای مختلف در برنامه آشنا خواهید شد.

هر دستوری که در AMS نوشته می‌شود دارای روش خاصی برای خطا گیری دارد، حال ممکن است این دستور مربوط به یک شی باشد و یا ممکن است مربوط به یک فعالیت غیر قابل مشاهده باشد که به شرح این روش‌ها می‌پردازیم:

خطا گیری با دستور Application.GetLastError

یکی از معمول‌ترین روش‌های خطا گیری برای دستوراتی که احتمال بروز خطا در آن‌ها وجود دارد استفاده از دستور Application.GetLastError می‌باشد.

این دستور به نحوی عمل می‌کند که اگر خطایی در اجرای کدهای سطرهای قبلی رخ داده باشد آن‌ها را شناسایی می‌کند و شماره‌ی خطا را بر می‌گرداند سپس با رجوع به لیست خطا و یافتن شماره‌ی مورد نظر می‌توان آن را به کاربر اعلام نمود.

و چنانچه خطایی پیدا نکند عدد 0 را بر می‌گرداند.

مثال: به روش خطا گیری کد زیر دقت کنید:

```
Zip.Add(_SourceFolder.."\\Mydb.zip",
{_SourceFolder.."\\Mydb.db3"}, true, "MyPassword", 5, nil, true);
error = Application.GetLastError();
if (error ~= 0) then
        Dialog.Message("Error", _tblErrorMessages[error], MB_OK,
MB_ICONEXCLAMATION);
end
```

بررسی مثال همان طور که در مثال بالا مشاهده می‌کنید برای خطا گیری احتمالی دستور Zip.Add از تابع Application.GetLastError استفاده نموده‌ایم و سپس با یک دستور if شرطی تعریف کردیم که اگر مقدار بازگشتی از متغیر error نامساوی با عدد 0 بود به وسیله‌ی یک کادر پیغام به لیست خطاها یعنی _tblErrorMessages مراجعه می‌کنیم و با توجه به شماره‌ی دریافت شده از متغیر error پیام مورد نظر را از لیست انتخاب می‌کنیم و به کار بر نمایش می‌دهیم.

نکته: _tblErrorMessages یک لیست بلند از خطاهای احتمالی پروژه های ساخته شده با AMS می‌باشد که همراه با اجرای پروژه به صورت یک آرایه قابل دسترس می‌باشد.

و اما سؤال اینجا است که چگونه بفهمیم که کدام یک از دستورات با استفاده از تابع Application.GetLastError قابل خطا گیری می‌باشند؟

برای این منظور در محیط کد نویسی روی کد مورد نظر دو بار کلیک کنید تا پنجره‌ی خصوصیات کد مورد نظر باز شود، سپس روی دکمه‌ی help در همان پنجره کلیک کنید. با این کار فایل راهنمای برنامه باز خواهد شد و به صورت خودکار به قسمت توضیحات کد مورد نظر می‌رود. در توضیحات کد مورد نظر به قسمت Returns مراجعه کنید؛ در این قسمت توضیحاتی در مورد مقداری که از کد مورد نظر برگردانده می‌شود نوشته شده است که یکی از آن توضیحات نحوه‌ی خطا گیری کد مورد نظر می‌باشد.

در مورد دستوراتی که با تابع Application.GetLastError خطا گیری می‌شوند در توضیحات آن عبارتی همانند عبارت زیر نوشته شده است:

You can use Application.GetLastError to determine whether this action failed, and why.

که می‌گوید: چنان چه این دستور با خطایی مواجه شد می‌توانید از تابع Application.GetLastError استفاده نمایید.

خطا گیری با بررسی مقدار بازگشتی

برخی دستورات و نیز برخی از متغیرهای رویدادی دارای یک مقدار بازگشتی می‌باشند که معمولاً در هنگام کد نویسی محل ذخیره‌ی آن مقدارها را در یک متغیر تعیین می‌کنیم. با بررسی این مقدار این متغیرها می‌توانیم جلوی برخی از خطاهای احتمالی را بگیریم و یا یک پیام مناسب به کاربر ارایه دهیم این خطاها از نوع خطاهای منطقی هستند.

مثال: فرض کنید در یک لیست از شی ListBox کدی نوشته‌ایم که با انتخاب یک گزینه‌ی متن آن گزینه در یک کادر پیغام ظاهر می‌گردد.

```
index = ListBox.GetSelected("ListBox1");
text = ListBox.GetItemText("ListBox1", index[1]);
Dialog.Message("Show Text", text);
```

حال اگر کاربری برای بار اول بیاید و بر روی قسمت خالی ListBox کلیک کنید هیچ متنی برای نمایش موجود نخواهد بود و متغیر text ما به صورت تعریف نشده به کادر پیغام فرستاده می‌شود و به همین دلیل پیام خطای پیش‌فرض AMS نمایش داده خواهد شد که نشان از ضعف برنامه‌ی ما خواهد داشت.

برای رفع این مشکل می‌بایست از با استفاده یک دستور if از اجرای کد در هنگام انتخاب جای خالی از ListBox جلوگیری کنیم. بدین منظور می‌بایست متغیر index را بررسی کنیم که آیا مقدار بازگشتی آن برابر با nil (پوچ) است یا خیر؟ اگر مقدار آن برابر با nil باشد یعنی یک جای خالی از لیست انتخاب شده است که در این صورت برنامه دستورات درون شرط if را اجرا نخواهد کرد:

```
index = ListBox.GetSelected("ListBox1");
if index ~= nil then
text = ListBox.GetItemText("ListBox1", index[1]);
```

```
Dialog.Message("Show Text", text);
end
```

برای آن که بدانید آیا دستور مورد نظر مقدار بازگشتی دارد یا خیر؟ و یا مقدار بازگشتی آن چیست و یا اینکه چه زمانی مقدار بازگشتی آن nil خواهد بود؟ روی آن دستور دو بار کلیک کنید و در پنجره‌ی خصوصیات آن روی دکمه‌ی help کلیک کنید و سپس در توضیحات کد مورد نظر به قسمت Returns مراجعه کنید و سپس ببینید که مقدار بازگشتی آن کد چیست یا از چه نوع است.

معمولاً دستوراتی که در هنگام بروز خطا دارای مقدار بازگشتی nil نیز می‌باشند در میان توضیحات این قسمت عبارتی همانند عبارت زیر موجود است:

nil is returned.

خطایابی منحصر به فرد

برخی از دستورات در AMS نیز خود دارای قسمتی مخصوص برای بازتاب خطاهای ایجاد شده می‌باشند همانند دستورات مربوط به دیتابیس‌ها که توضیح آن‌ها بیان شد. به مثال زیر دقت کنید:

```
MySQLConnection, err = MySQL:connect("Mydb", "hamseda",
"vasva3", "www.vasva3.com");
if not MySQLConnection or err then
   Dialog.Message("Error1", err);
     Application.ExitScript();
 end
```

در این مثال تابع MySQL:connect خطاهای ایجاد شده را به متغیر err می‌فرستد و سپس با بررسی آن می‌توانیم بفهمیم چه خطایی رخ داده است.

هم چنین برخی از اشیا دارای چنین رویدادها و متغیرهایی می‌باشند که از جمله می‌توان به شی QuickTime اشاره نمود که دارای رویدادی به نام On Error می‌باشد و چنان چه این شی با خطایی مواجه شود کدهای نوشته شده در این قسمت را اجرا خواهد کرد در این رویداد

متغیرهایی نیز وجود دارد که برای کنترل خطاهای آن شی مناسب می‌باشند که توضیح آن‌ها در **متغیرهای رویدادی** بیان شد.

فصل نهم: توابع و دستورات AMS

برای آنکه بتوانیم برنامه‌های پیچیده و حرفه ای را با AMS طراحی و تولید نماییم لازم است با توابع و دستورات این برنامه آشنا باشیم و کاربرد هر یک از آن‌ها را بدانیم تا در جای مناسب از بهترین دستور ممکن استفاده نماییم. تا اینجا با روش‌های برنامه نویسی و خطا گیری در پروژه های نوشته شده با AMS آشنا شدید و دیگر وقت آن رسیده است که کاربرد هر یک از دستورات موجود در این برنامه را فرا بگیرید. در این فصل از کتاب با تمامی دستورات موجود در AMS آشنا شده و کاربرد هر یک از آن‌ها را خواهید آموخت.

در این فصل به موارد زیر می‌پردازیم:

- دستورات موجود AMS
- کاربرد هر یک دستورات
- پارامترهای ورودی دستورات
- خصوصیات برخی از دستورات

دستورات موجود AMS

Application

این تابع برای کار با پروژه‌ی اجرایی به کار می‌رود و دارای زیر مجموعه‌هایی به شرح زیر می‌باشد:

Application.Exit: از این دستور برای خروج از نرم افزار استفاده می‌شود. این دستور مانند فشردن دکمه "ضربدر" در نوار عنوان پنجره می‌باشد.

مقدار بازگشتی **ندارد** .

Application.ExitScript: از بلوک اسکریپت حاضر خارج می‌شود. تفاوت این دستور با دستور breakاین است که در این دستور از کل بلوک خارج می‌شود ولی دستور break فقط از حلقه مورد نظر خارج می‌شود .

مقدار بازگشتی **ندارد** .

Application.GetCurrentDialog: با این دستور می‌توان نام دیالوگ موجود را گرفت .

مقدار بازگشتی این دستور نام دیالوگ (به صورت رشته) می‌باشد. در صورت وجود مشکل یا عدم وجود دیالوگ، یک رشته خالی ("") بازگردانده می‌شود .

برای مثال:

```
DLG_N = Application.GetCurrentDialog();
if DLG_N ~=”” then
Result = Dialog.Message("Name", DLG_N);
end
```

Application.GetCurrentPage: این دستور هم مانند دستور قبلی عمل می‌کند، با تفاوت اینکه بجای نام دیالوگ، نام صفحه را باز می‌گرداند.

مقدار بازگشتی این دستور هم نام صفحه (به صورت رشته) می‌باشد و در صورت وجود خطا، یک رشته خالی بازگردانده می‌شود .

Application.GetDialogProperties: این دستور خصوصیات دیالوگ را باز می‌گرداند .به عنوان ورودی باید نام دیالوگ مورد نظر را وارد کنید تا خصوصیات دیالوگ دلخواه شما باز گردانده شود.

مقدار بازگشتی این دستور، یک آرایه با متغیرهای زیر می‌باشد :

DialogTitle: عنوان صفحه به صورت رشته

Movable: قابل حرکت بودن به صورت بولین(در صورت فعال بودن True بازگردانده می‌شود)

AlwaysOnTop: نمایش قابلیت بالا بودن پنجره نسبت به تمام پنجره های باز شده در ویندوز به صورت بولین

Width: عرض دیالوگ به صورت پیکسل (مثلاً 300 پیکسل)

Height :طول دیالوگ به صورت پیکسل

DialogStyle: استیل دیالوگ را با یکی از اعداد 0، 1، 2 یا 3 را نمایش می‌دهد :

STANDARD (0), BORDERED (1), FLAT (2),MASK (3).

Resizable :نمایش قابلیت تغییر اندازه دیالوگ به صورت بولین

MinWidth: کمترین عرض ممکن، در صورتی که Resizable برابر True باشد.

MinHeight: کمترین طول ممکن، در صورتی که Resizable برابر True باشد.

UseCustomIcon: مقدار True در صورت انتخاب آیکون برای دیالوگ (بولین) .

CustomIcon: مسیر آیکون در صورت True بودن UseCustomIcon .

UseCustomSettings: True در صورتی که رنگ اختصاصی انتخاب شده باشد .

BackgroundType: عدد مربوط به نوع بک گراند به صورت زیر :

SOLID (0), GRADIENT (1), IMAGE (2)

BackgroundColor: شماره رنگ

GradientColorTop: شماره رنگ قسمت بالای Gradient در صورتی که BackgroundType=1 باشد.

ImageFilename: مسیر تصویر استفاده شده در بک گراند در صورتی که BackgroundType=2 باشد.

ImageStretchMode: نوع کشیده شدن و قرار گرفتن تصویر در بک گراند در صورتی که BackgroundType=2 باشد.

FITPAGE (0), TILE (1), ACTUALSIZE (2).

CustomMask: مسیر فایل استفاده شده برای ماسک در صورتی که DialogStyle=3 باشد.

FitCustomMaskToWindow :در صورتی که هم اندازه شدن تصویر ماسک با دیالوگ انتخاب شده باشد، True بازگردانده می‌شود.

مثال :

```
DLG_P = Application.GetDialogProperties("Dialog1");
if DLG_p then
result = Dialog.Message("Title", DLG_P.DialogTitle);
end
```

Application.GetDialogs :با استفاده از این دستور می‌توانید نام تمام دیالوگ‌های موجود را در یک آرایه داشته باشید.

مقدار بازگشتی به صورت یک آرایه حاوی نام دیالوگ‌ها می‌باشد.

Application.GetDialogScript :این دستور، اکشن‌ها (دستورات) قرار گرفته در رویداد خاصی در دیالوگ را بازمی‌گرداند.نام دیالوگ و رویداد مورد نظر باید به عنوان ورودی وارد شوند.

مقدار بازگشتی این دستور، رشته می‌باشد .در صورت بروز خطا، رشته خالی بازگردانده می‌شود.

مثال:

```
Time_Script = Application.GetDialogScript ("Dialog2", "On Time");
```

Application.GetLastError :کد آخرین خطای رخ داده بازگردانده می‌شود .در صورتی که آخرین دستور اجرا شده موفق آمیز باشد، مقدار صفر در غیر این صورت مقداری بزرگ‌تر از 0 / صفر بازگردانده می‌شود.

Application.GetMenu: بازگرداندن اطلاعات منوها وظیفه‌ی این دستور می‌باشد.

مقدار بازگشتی این دستور یک آرایه عددی با زیر آرایه های زیر می‌باشد :

ID: عدد شناسه منو

Text: متن منو(در صورتی که جدا کننده باشد رشته برابر "--- " خواهد بود.

IconID: شناسه آیکون منو

Enabled: فعال یا عدم فعال بودن منو به صورت بولین

Checked :قابلیت تیک خوردن منو به صورت بولین

SubMenu: جدولی متشکل از نام زیر منوهای موجود .در صورتی که زیر منویی وجود نداشته باشد مقدارnil باز می‌گردد.

Application.GetPageProperties: این دستور خصوصیات صفحه را باز می‌گرداند .به عنوان ورودی باید نام دیالوگ مورد نظر را وارد کنید تا خصوصیات دیالوگ دلخواه شما باز گردانده شود.

مقدار بازگشتی این دستور، یک آرایه با متغیرهای زیر می‌باشد :

UseCustomSettings:True در صورتی که رنگ اختصاصی انتخاب شده باشد .

BackgroundType: عدد مربوط به نوع بک گراند به صورت زیر :

SOLID (0), GRADIENT (1), IMAGE (2)

BackgroundColor: شماره رنگ

GradientColorTop:شماره رنگ قسمت بالای Gradient در صورتی که BackgroundType=1 باشد.

ImageFilename: مسیر تصویر استفاده شده در بک گراند در صورتی که BackgroundType=2 باشد.

ImageStretchMode: نوع کشیده شدن و قرار گرفتن تصویر در بک گراند در صورتی که BackgroundType=2 باشد.

FITPAGE (0), TILE (1), ACTUALSIZE (2).

Description: توضیحات مربوط به صفحه

Keywords: کلمات کلیدی به صورت آرایه

Application.GetPages: با استفاده از این دستور می‌توانید نام تمام صفحات موجود را در یک آرایه داشته باشید.

مقدار بازگشتی به صورت یک آرایه حاوی نام صفحات می‌باشد.

Application.GetPageScript: این دستور مانند GetDialogScript، اکشن‌ها (دستورات) قرار گرفته در رویداد خاص را بازمی‌گرداند.نام صفحه و رویداد مورد نظر باید به عنوان ورودی وارد شوند.

مقدار بازگشتی این دستور، رشته می‌باشد .در صورت بروز خطا، رشته خالی بازگردانده می‌شود.

Application.GetWndHandle: این دستور شماره Handle پنجره موجود بازمی‌گرداند.

مثال:

```
handle = Application.GetWndHandle();
Window.Hide(handle);
Application.Sleep(1520);
Window.Show(handle);
```

Application.LoadActionPlugin: با استفاده از این دستور می‌توانید فایلی پلاگین ها (با فرمت LMD) را بارگذاری نمایید. مسیر فایل مورد نظر به عنوان ورودی این دستور خواهد بود. در حالت معمولی می‌توانید از مسیر Project > Plugins یک پلاگین را بارگذاری کنید.

مقدار بازگشتی **ندارد**.

Application.LoadScript: با استفاده از این دستور می‌توانید دستورات نوشته شده از قبل که در فایل ذخیره کرده‌اید را بارگذاری کنید.مسیر فایل به عنوان ورودی این دستور خواهد بود .

مقدار بازگشتی **ندارد** .

Application.LoadValue: بارگذاری یک دستور که با Application.SaveValue ذخیره شده است. section و key به عنوان ورودی در دستور می‌باشد .

مقدار بازگشتی عبارت ذخیره شده می‌باشد .در صورت بروز خطا یک رشته خالی برگشت داده می‌شود.

Application.MakeKeywordIndex: این دستور با جستجوی تمام صفحات، لیستی از "کلمات کلیدی " را در آرایه می‌ریزد، ورودی این دستور یک عبارت بولین می‌باشد که برای مرتب سازی به کار می‌رود.

مقدار بازگشتی آرایه ای از کلمات کلیدی است در صورتی که کلمه کلیدی تعریف نشده باشد عبارت nil بازگردانده می‌شود.

Application.Minimize :این دستور وظیفه کمینه کردن (مینی مایز) نرم افزار در Taskbar (اگر در Setting پروژه تغییر نداده باشید) را بر عهده دارد.

مقدار بازگشتی **ندارد** .

Application.Restore: این دستور پنجره را به اندازه و موقعیتی که قبل از Minimize داشته ، برمی‌گرداند.

مقدار بازگشتی **ندارد** .

Application.RunScriptFile: فایل‌های Lua که از قبل آماده کرده‌اید را با این دستور اجرا کنید .

مقدار بازگشتی **ندارد** .

Application.SaveValue: ذخیره یک داده در قسمت User ویندوز با این دستور انجام می‌پذیرد. برای استفاده از آن در زمانی دیگر باید از Application.LoadValue استفاده نمود.

مقدار بازگشتی **ندارد** .

Application.SearchKeywords: با استفاده از این کد بین "کلمه های کلیدی " جستجو نمایید.ورودی این کد، یک رشته برای جستجو و دو عبارت بولین برای AllowPartialMatch و CaseSensitive می‌باشد .

مقدار بازگشتی یک آرایه با نام صفحات می‌باشد .

Application.SetDialogProperties: با این دستور می‌توانید خصوصیات جدید برای دیالوگ خود تعریف کنید . ورودی‌های این دستور نام دیالوگ و یک آرایه با مشخصه های DialogTitle , Movable , AlwaysOnTop , Width , Height , DialogStyle , MinWidth , MinHeight , UseCustomIcon , CustomIcon , Resizable , UseCustomSettings , BackgroundType , BackgroundColor , GradientColorTop , ImageFilename , ImageStretchMode , CustomMask و FitCustomMaskToWindow می‌باشد که قبلاً در Application.GetDialogProperties توضیح داده شده است .

مقدار بازگشتی **ندارد** .

Application.SetDialogScript :با استفاده از این دستور هم می‌توانید در رویداد مورد نظر خود در دیالوگ، دستورات جدید قرار دهید.ورودی این دستور نام دیالوگ، رویداد مورد نظر (برای مثال On Time) و همچنین دستوراتی که نیاز به قرارگیری در رویداد دارند می‌باشد .

مقدار بازگشتی **ندارد** .

مثال:

```
Action_DLG3 = Application.GetDialogScript ("Dialog3", "On Show");
Action_DLG3 = Action_DLG3 .. "\r\n" .. "Application.Exit()";
Application.SetDialogScript("Dialog3", "On Show", Action_DLG3);
```

Application.SetLastError :با این دستور می‌توانید کد آخرین پیغام خطای رخ داده را تغییر دهید .ورودی این کد عددی مربوط به یکی از خطاها می‌باشد.

مقدار بازگشتی **ندارد** .

Application.SetMenu: با استفاده از این دستور می‌توانید برای نرم افزار منو ایجاد کنید، البته باید نمایش منو بار قبلاً فعال شده باشد. پارامتر ورودی یک جدول می‌باشد با مشخصه های ID,Text,IconID,Enabled ,Checked و SubMenu که قبلاً در Application.GetMenu توضیح داده شده است.

مقدار بازگشتی **ندارد** .

Application.SetMenuBarActive :با این دستور می‌توانید منو بار را فعال یا غیرفعال نمایید.پارامتر ورودی این دستور یک بولین برای فعال کردن یا غیرفعال کردن منو می‌باشد.

مقدار بازگشتی **ندارد** .

Application.SetPageProperties :این کد برای تنظیم خصوصیات جدید برای صفحه استفاده می‌شود. ورودی این دستور آرایه ای با مشخصه های , UseCustomSettings , BackgroundColor , GradientColorTop , BackgroundType , Description , ImageStretchMode , ImageFilename و Keywords می‌باشد که در قسمت Application.GetPageProperties توضیح داده شده است.

Application.SetPageScript: با استفاده از این دستور هم می‌توانید در رویداد مورد نظر خود در صفحه، دستورات جدید قرار دهید.ورودی این دستور نام صفحه، رویداد مورد نظر (برای مثال On Preload) و همچنین دستوراتی که نیاز به قرارگیری در رویداد دارند می‌باشد.

Application.SetRedraw :با این دستور می‌توانید رسم دوباره‌ی اشیا را فعال یا غیر فعال کنید.ورودی این دستور یک بولین می‌باشد.

مقدار بازگشتی **ندارد** .

Application.SetSysTrayIcon: با این دستور می‌توانید آیکون System Tray مربوط به نرم افزار را تغییر دهید. ورودی‌های این کد IconPath و IconIndex می‌باشد که اولی به صورت رشته ای است که مسیر آیکون با فرمت ico یا exe می‌باشد و دیگری شماره آیکون در فایل‌های exe می‌باشد.(برای فایل با پسوند ico این مقدار را برابر 0 قرار دهید.)

مقدار بازگشتی این دستور بولین می‌باشد که موفقیت آمیز بودن یا غیر موفق آمیز بودن اجرای کد را باز می‌گرداند.

Application.SetSysTrayTooltip :با استفاده از این دستور می‌توانید tooltip آیکون System tray را تغییر دهید. ورودی‌های این دستور یک رشته (متن برای ToolTip) و یک عبارت بولین برای اینکه همان زمان متن جدید را نمایش دهد یا منتظر قرارگیری ماوس روی آیکون توسط کاربر شود.

مقدار بازگشتی این دستور هم بولین می‌باشد که موفقیت آمیز بودن یا غیر موفق آمیز بودن اجرای کد را باز می‌گرداند.

Application.ShowPopupMenu: با این دستور می‌توانید PopUp Menu ساخته و نمایش دهید . این دستور دارای 7 ورودی است، فاصله از چپ و بالای صفحه، اجزای منو، چینش افقی و عمودی و دو بولین برای "انتظار برای پاسخ دهی " و "نحوه محاسبه فاصله از بالا و چپ" می‌باشد . در ClientCoordinates در صورتی که True انتخاب شود، X و Y

از گوشه پنجره نرم افزار محاسبه می‌شود در غیر این صورت از گوشه صفحه نمایش کاربر محاسبه خواهد شد.

مقدار بازگشتی ID منوی انتخاب شده (عدد) می‌باشد. در صورت بروز خطا یا عدم انتخاب -1 بازگردانده می‌شود.

Application.Sleep: با استفاده از این دستور می‌توانید، نرم افزار را اصطلاحاً به خواب ببرید.یعنی برای مدت زمان دلخواه شما برنامه غیر فعال می‌شود.در این دستور ورودی عدد به صورت میلی ثانیه وارد می‌شود که هر 1000 میلی ثانیه برابر 1 ثانیه خواهد بود.

مقدار بازگشتی **ندارد** .

Audio

این تابع برای کار با فایل‌های صوتی به کار می‌رود که دستورات زیر مجموعه‌ی آن به شرح زیر است:

Audio.GetCurrentPos: برای بدست آوردن میزان سپری شده صوت از کانال مورد نظر باید از این دستور استفاده کنید .مقدار ورودی این دستور کانال مورد نظر می‌باشد به جای نام کانال‌ها می‌توانید از عدد به جای آن‌ها (از0 تا 6) استفاده نمایید.

نام ثابت کانال	شماره	توضیح
CHANNEL_BACKGROUND	5	صدای پس زمینه
CHANNEL_EFFECTS	0	جلوه های صوتی
CHANNEL_NARRATION	6	صدای دوم
CHANNEL_USER1	1	کانال صوتی 1
CHANNEL_USER2	2	کانال صوتی 2
CHANNEL_USER3	3	کانال صوتی 3
CHANNEL_USER4	4	کانال صوتی 4
CHANNEL_ALL	-3	تمامی کانال‌ها

مقدار بازگشتی، موقعیت کنونی در آهنگ به ثانیه می‌باشد.

Audio.GetFilename: از این دستور برای گرفتن مسیر فایل بارگذاری شده در هر یک از کانال‌های دلخواه استفاده می‌شود. مقدار ورودی همان کانال می‌باشد .

مقدار بازگشتی دستور، رشته ای با محتوای مسیر فایل است .در صورت بروز مشکل رشته خالی باز گردانده می‌شود.

Audio.GetLength: از این دستور برای بدست آوردن طول آهنگ استفاده کنید.این دستور هم کانال را به عنوان ورودی می‌پذیرد.

مقدار بازگشتی طول آهنگ به ثانیه است .

Audio.GetOggTags: با این دستور برچسب‌های یک فایل OGG را به صورت آرایه بدست آورید.مقدار ورودی کانال می‌باشد.

مقدار بازگشتی آرایه ای متشکل از TITLE , VERSION , COPYRIGHT و... می‌باشد.در صورتی که مشکلی به وجود آید مقدار nil بازگشت داده می‌شود.

Audio.GetVolume: این دستور بلندی صدای کانال را نمایش می‌دهد.ورودی همان کانال می‌باشد .

مقدار بازگشتی عددی بین 0 و 255 می‌باشد که نشان دهنده میزان بلندی صدا در کانال مورد نظر است. در صورت بروز خطا 1- بازمی‌گردد.

Audio.IsLooping: با ورود کانال مورد نظر در این دستور فعال یا عدم فعال بودن صدا بازگردانده می‌شود.

مقدار بازگشتی این دستور از نوع بولین می‌باشد. True نشان دهنده فعال بودن تکرار و False غیرفعال بودن تکرار می‌باشد.

Audio.Load: با این دستور می‌توانید یک فایل صوتی را در کانال مورد نظر بارگذاری کنید.مقدار ورودی اول، کانال دلخواه برای قرارگیری آهنگ مدنظر است. ورودی دوم مسیر صوت مورد نظر می‌باشد . ورودی سوم از نوع بولین برای پخش اتوماتیک یا عدم پخش اتوماتیک صوت پس از بارگذاری است و ورودی آخر فعال یا غیر فعال کردن تکرار آهنگ می‌باشد.

مقدار بازگشتی **ندارد**.

Audio.Pause: این دستور با گرفتن نام کانال به عنوان ورودی، پخش صوت را متوقف می‌کند.با پخش مجدد صوت، از ادامه پخش خواهد شد.

مقدار بازگشتی **ندارد**.

Audio.Play: این دستور صوت بارگذاری شده در کانالی که به عنوان ورودی وارد کردید را پخش می‌کند.

مقدار بازگشتی **ندارد**.

Audio.Seek: پرش به موقعیت دلخواه صوت در کانال مورد نظر. ورودی اول طبق روال کانال می‌باشد، ورودی دوم نوع پرش است:

SEEK_BEGINNING: پرش به اول صوت

SEEK_END: پرش به انتهای صوت

SEEK_FORWARD: پرش به جلو

SEEK_BACKWARD: پرش به عقب

SEEK_SPECIFIC: پرش به زمان خاص

ورودی سوم هم زمان مناسب برای پرش (برای موارد سوم تا پنجم) استفاده می‌شود.

مقدار بازگشتی **ندارد** .

Audio.SetLooping: فعال یا غیرفعال کردن تکرار صوت با این دستور امکان پذیر است. ورودی‌ها کانال و یک عبارت بولین برای فعال و غیرفعال کردن تکرار صوت می‌باشند.

مقدار بازگشتی **ندارد** .

Audio.SetVolume: با ورود کانال و همچنین عددی بین 0 تا 255 می‌توانید بلندی صدای کانال دلخواهتان را تنظیم نمایید.

مقدار بازگشتی **ندارد** .

Audio.Stop: توقف کامل صوت با استفاده از این دستور امکان پذیر است .با پخش مجدد، صوت از ابتدا شروع به پخش می‌شود.

مقدار بازگشتی **ندارد** .

Audio.ToggleMute: بی صدا کردن فایل صوتی.در صورتی که در حال پخش باشد، آن را بی صدا می‌کند و اگر بی صدا باشد، دوباره صدای آن را وصل می‌کند (در خلال بی صدا بودن، فایل پخش خواهد شد.)

مقدار بازگشتی **ندارد**.

Audio.TogglePlay: مانند گزینه‌ی قبل عمل می‌کند با تفاوت اینکه با اولین اجرای دستور فایل نیز متوقف می‌شود.

مقدار بازگشتی **ندارد**.

Button

این تابع برای کار با شی Button(دکمه) به کار می‌رود و دستورات زیر مجموعه‌ی آن به شرح زیر است:

Button.GetPos: با استفاده از این دستور می‌توانید موقعیت قرارگیری دکمه در صفحه را بدست آورید .این موقعیت نسبت به گوشه‌ی بالا و سمت چپ محاسبه می‌شود.ورودی این تابع نام دکمه می‌باشد .

مقدار بازگشتی یک آرایه با متغیرهای X و Y می‌باشد.

مثال :

```
B_pos = Button.GetPos("Button 1");
result = Dialog.Message("X is :", B_pos.X);
```

Button.GetProperties: با این دستور می‌توانید خصوصیات دکمه را در یک آرایه داشته باشید. ورودی این تابع نام دکمه می‌باشد.

مقدار بازگشتی این دستور، یک آرایه با متغیرهای زیر می‌باشد :

ButtonFile: مسیر فایل (با فرمت btn.)

Text: متن موجود در دکمه

ObjectName: نام شی

FontName: نام فونت استفاده شده

FontSize: اندازه فونت

FontStrikeout: فعال یا غیر فعال بودن strikeout در تنظیمات فونت (بولین)

FontUnderline: فعال یا غیر فعال بودن underline در تنظیمات فونت (بولین)

FontAntiAlias: فعال یا غیر فعال بودن anti alias در تنظیمات فونت (بولین)

FontItalic: فعال یا غیر فعال بودن italic در تنظیمات فونت (بولین)

FontWeight: نوع نوشته (تیرگی متن) به صورت آرایه ای متشکل از DONTCARE ,THIN ,EXTRALIGHT ,LIGHT ,NORMAL ,MEDIUM, SEMIBOLD, BOLD, EXTRABOLD و HEAVY .

FontScript: نوع اسکریپت فونت به صورت آرایه ای متشکل از ANSI , BALTIC , CHINESEBIG5 , DEFAULT , EASTEUROPE , GB2312 , GREEK ,

HANGUL , MAC , OEM , RUSSIAN , SHIFTJIS ,SYMBOL و TURKISH .

XOffset: تغییر فاصله افقی متن نسبت به حالت پایه.

YOffset: تغییر فاصله عمودی متن نسبت به حالت پایه.

LeftMargin: فاصله (حاشیه) متن از سمت چپ.

RightMargin: فاصله (حاشیه) متن از سمت راست.

Alignment: نوع چینش متن به صورت آرایه ای سه عضوی با اعضای: LEFT, CENTER و RIGHT

Style: نوع استیل دکمه با آرایه ای تشکیل شده از STANDARD و TOGGLE

ToggleState: وضعیت کنونی دکمه (UP یا DOWN)

ColorNormal: رنگ متن دکمه در حالت normal

ColorHighlight: رنگ متن دکمه در حالت highlight

ColorDisabled: رنگ متن دکمه در حالت disabled

ColorDown: رنگ متن دکمه در حالت down

Enabled: فعال یا عدم فعال بودن دکمه (بولین)

Visible: دیده شدن یا عدم دیده شدن دکمه (بولین)

X: فاصله دکمه از سمت چپ صفحه

Y: فاصله دکمه از سمت راست صفحه

Width: عرض دکمه (پیکسل)

Height: طول دکمه (پیکسل)

TooltipText: متن نمایش دهنده زمانی که نشانگر ماوس روی آن دکمه قرار می‌گیرد.

Cursor: شکل نشانگر ماوس. آرایه ای با اعضای: ARROW , HAND , BLACK_ARROW , CROSSHAIR , EXPLORE , HELP , MAGNIFY , MEDIA , MONEY , NOTEPAD , PENCIL , PRINTER , SPEAKER , UP_ARROW یا

ResizeLeft: قابلیت تغییر اندازه از سمت چپ در زمان تغییر اندازه پروژه (بولین)

ResizeRight: قابلیت تغییر اندازه از سمت راست در زمان تغییر اندازه پروژه (بولین)

ResizeTop: قابلیت تغییر اندازه از بالا در زمان تغییر اندازه پروژه (بولین)

ResizeBottom: قابلیت تغییر اندازه از پایین در زمان تغییر اندازه پروژه (بولین)

HighlightSound: نوع صوت در هنگام قرارگیری نشانگر ماوس بر روی دکمه (NONE, STANDARD یا CUSTOM)

HighlightSoundFile: مسیر فایل صوتی پخش شونده در هنگام قرارگیری نشانگر ماوس بر روی دکمه

ClickSound: نوع صوت در هنگام کلیک بر روی دکمه (NONE, STANDARD یا CUSTOM)

ClickSoundFile: مسیر فایل صوتی پخش شونده در هنگام کلیک بر روی دکمه

Button.GetSize: نمایش اندازه دکمه به پیکسل با این دستور امکان پذیر است.ورودی دستور نام دکمه می‌باشد.

مقدار بازگشتی دستور، آرایه ای 2 عضوی با اعضای "Width" و "Height" می‌باشد.

Button.GetState: با این دستور می‌توانید حالت UP یا Down بودن دکمه در هنگام فعال بودن toggle را بدست آورید.ورودی این کد نام دکمه است.

مقدار بازگشتی کد، عدد0، 1 یا 1- می‌باشد که نمایش دهنده UP، Down یا خطا می‌باشد.

Button.GetText: متن موجود در دکمه با استفاده از این دستور بدست می‌آید. مقدار ورودی نام دکمه است.

مقدار بازگشتی رشته ای است حاوی متن دکمه . در صورت بروز خطا رشته خالی بازگردانده می‌شود.

Button.IsEnabled: با استفاده از این دستور می‌توانید فعال یا غیرفعال بودن دکمه را بدست آورید. مقدار ورودی نام دکمه است.

مقدار بازگشتی Trueبه معنای فعال بودن یا False به معنای غیرفعال بودن دکمه می‌باشد.

Button.IsVisible: با استفاده از این دستور می‌توانید نمایش یا عدم نمایش دکمه در نرم افزار را بدست آورید. مقدار ورودی نام دکمه است.

مقدار بازگشتی Trueبه نمایش یا False به معنای عدم نمایش دکمه می‌باشد.

Button.SetEnabled: با این دستور می‌توانید دکمه را فعال یا غیر فعال کنید.ورودی دستور نام دکمه مورد نظر و همچنین عبارت بولین برای فعال یا غیرفعال کردن دکمه می‌باشد

مقدار بازگشتی **ندارد** .

Button.SetPos: با این کد می‌توانید دکمه را به محل دلخواه در صفحه منتقل نمایید. ورودی‌های این کد نام دکمه، عددی برای جای گیری در قسمت X (فاصله از چپ) و عدد دیگری برای جای گیری در Y (فاصله از بالا می‌باشد).

مقدار بازگشتی **ندارد** .

Button.SetProperties: با این دستور می‌توانید خصوصیات جدیدی بر روی دکمه خود قرار دهید. ورودی‌های این تابع یکی نام دکمه مورد نظر و دیگری آرایه ای متشکل از خصوصیات دکمه است که این خصوصیات در Button.GetProperties توضیح داده شده.

مقدار بازگشتی **ندارد** .

Button.SetSize: با این دستور می‌توانید دکمه را به اندازه دلخواه خود تغییر دهید.ورودی‌های این کد نام دکمه، عددی برای عرض و عددی دیگر برای طول دکمه می‌باشد.

مقدار بازگشتی **ندارد** .

Button.SetState: تغییر حالت دکمه به Up یا Down (در صورت فعال بودن Toggle) با استفاده از این دستور امکان پذیر می‌باشد. ورودی‌های این دستور عبارتند از نام دکمه و عدد 0 یا 1 برای نمایش Up یا Down .

مقدار بازگشتی **ندارد** .

Button.SetText: با این دستور می‌توانید متن نمایش داده شده بر روی دکمه را تغییر دهید. ورودی‌های این دستور عبارتند از نام دکمه و متن جدید به صورت رشته.

مقدار بازگشتی **ندارد** .

Button.SetVisible: کنترل نمایش یا عدم نمایش دکمه در نرم افزار هم با این دستور امکان پذیر است. ورودی‌های کد، نام دکمه و عبارتی بولین برای فعال یا غیر فعال کردن نمایش دکمه می‌باشد.

مقدار بازگشتی **ندارد** .

CheckBox

از این تابع برای کار کردن با شی CheckBox استفاده می‌شود که دستورات زیر مجموعه آن به شرح زیر می‌باشد:

CheckBox.GetChecked: با ورود نام شی به عنوان ورودی می‌توانید از علامت خوردن یا عدم علامت خوردن آن مطلع شوید.

مقدار بازگشتی عبارتی بولین برای نمایش این موضوع است.

CheckBox.GetPos: با استفاده از این دستور می‌توانید موقعیت قرارگیری شی را بدست آورید .این موقعیت نسبت به گوشه‌ی بالا و سمت چپ محاسبه می‌شود.ورودی این تابع نام شی CheckBox می‌باشد.

مقدار بازگشتی یک آرایه با متغیرهای X و Y می‌باشد.

CheckBox.GetProperties: با استفاده از این دستور می‌توانید خصوصیات CheckBox را در یک آرایه در اختیار داشته باشید.ورودی دستور نام شی می‌باشد.

مقدار بازگشتی این دستور آرایه ای است با متغیرهای :

ObjectName: نام شی

Checked: نمایش علامت خورده بودن یا عدم علامت خوردن با بولین

Text: متن نمایش داده شده بر روی شی

FontName: نام فونت استفاده شده

FontSize: اندازه فونت

FontStrikeout: فعال یا غیر فعال بودن strikeout در تنظیمات فونت (بولین)

FontUnderline: فعال یا غیر فعال بودن underline در تنظیمات فونت (بولین)

FontAntiAlias: فعال یا غیر فعال بودن anti alias در تنظیمات فونت (بولین)

FontItalic: فعال یا غیر فعال بودن italic در تنظیمات فونت (بولین)

FontWeight: نوع نوشته (تیرگی متن) به صورت آرایه ای متشکل از DONTCARE, THIN, EXTRALIGHT, LIGHT, NORMAL, MEDIUM, SEMIBOLD, BOLD, EXTRABOLD و HEAVY.

FontScript: نوع اسکریپت فونت به صورت آرایه ای متشکل از ANSI , BALTIC , CHINESEBIG5 , DEFAULT , EASTEUROPE , GB2312 , GREEK , HANGUL , MAC , OEM , RUSSIAN , SHIFTJIS ,SYMBOL و TURKISH .

TextAlignment: نوع چینش متن در شی (راست چین، چپ چین و وسط چین)

ButtonAlignment: نوع چینش دکمه شی (مربع سفید در سمت چپ یا راست)

ReadOrder: نوع قرارگیری متن در شی (راست به چپ یا معمولی)

ColorNormal: رنگ متن در حالت normal

ColorHighlight: رنگ متن در حالت highlight

ColorDisabled: رنگ متن در حالت disabled

ColorDown: رنگ متن در حالت down

Enabled: فعال یا عدم فعال بودن (بولین)

Visible: دیده شدن یا عدم دیده شدن (بولین)

X: فاصله از سمت چپ صفحه

Y: فاصله از سمت راست صفحه

Width: عرض (پیکسل)

Height: طول (پیکسل)

TooltipText: متن نمایش دهنده زمانی که نشانگر ماوس روی شی قرار می‌گیرد.

Cursor: شکل نشانگر ماوس. آرایه ای با اعضای: ARROW , HAND , BLACK_ARROW , CROSSHAIR , EXPLORE , HELP , MAGNIFY , MEDIA , MONEY , NOTEPAD , PENCIL , PRINTER , SPEAKER , UP_ARROW یا

ResizeLeft: قابلیت تغییر اندازه از سمت چپ در زمان تغییر اندازه پروژه (بولین)

ResizeRight: قابلیت تغییر اندازه از سمت راست در زمان تغییر اندازه پروژه (بولین)

ResizeTop: قابلیت تغییر اندازه از بالا در زمان تغییر اندازه پروژه (بولین)

ResizeBottom: قابلیت تغییر اندازه از پایین در زمان تغییر اندازه پروژه (بولین)

HighlightSound: نوع صوت در هنگام قرارگیری نشانگر ماوس بر روی شی (NONE, STANDARD یا CUSTOM)

HighlightSoundFile: مسیر فایل صوتی پخش شونده در هنگام قرارگیری نشانگر ماوس بر روی شی

ClickSound: نوع صوت در هنگام کلیک بر روی شی (NONE, STANDARD یا CUSTOM)

ClickSoundFile: مسیر فایل صوتی پخش شونده در هنگام کلیک بر روی شی

CheckBox.GetSize: نمایش اندازه شی CheckBox به پیکسل با این دستور امکان پذیر است.ورودی دستور نام شی می‌باشد.

مقدار بازگشتی دستور، آرایه ای است با دو عضو "Width" و "Height" .

CheckBox.GetText: متن نمایش داده شده در شی با استفاده از این دستور به دست می‌آید. مقدار ورودی نام شی CheckBox است.

مقدار بازگشتی رشته ای است حاوی متن دریافت شده . در صورت بروز خطا رشته خالی بازگردانده می‌شود.

CheckBox.IsEnabled: با استفاده از این دستور می‌توانید فعال یا غیرفعال بودن شی را بدست آورید. مقدار ورودی نام شی CheckBox است.

مقدار بازگشتی Trueبه معنای فعال بودن یا False به معنای غیرفعال بودن CheckBox می‌باشد.

CheckBox.IsVisible: با استفاده از این دستور می‌توانید نمایش یا عدم نمایش CheckBox را بدست آورید. مقدار ورودی نام شی CheckBox است.

مقدار بازگشتی Trueبه نمایش یا False به معنای عدم نمایش CheckBox می‌باشد.

CheckBox.SetChecked:با استفاده از این دستور می‌توانید CheckBox را علامت بزنید یا از علامت دار بودن در آورید.ورودی تابع نام شی و متغیری بولین است .

مقدار بازگشتی **ندارد**.

CheckBox.SetEnabled: با این دستور می‌توانید CheckBox را فعال یا غیر فعال کنید.ورودی دستور نام شی و همچنین عبارت بولین برای فعال یا غیرفعال کردن آن است.

مقدار بازگشتی **ندارد** .

CheckBox.SetPos: با این کد می‌توانید شی CheckBox را به محل دلخواه در صفحه منتقل نمایید.ورودی‌های این کد نام شی، عددی برای فاصله از چپ و عدد دیگری برای فاصله از بالا می‌باشد.

مقدار بازگشتی **ندارد** .

CheckBox.SetProperties: با این دستور می‌توانید خصوصیات جدیدی برای شی CheckBoاعمال کنید.ورودی‌های این تابع یکی نام شی و دیگری آرایه ای متشکل از خصوصیات CheckBox است که این خصوصیات در CheckBox.GetProperties توضیح داده شده.

مقدار بازگشتی **ندارد** .

CheckBox.SetSize: با این دستور می‌توانید CheckBox را به اندازه دلخواه خود تغییر دهید.ورودی‌های این کد نام شی، عددی برای عرض و عددی دیگر برای طول دکمه می‌باشد.

مقدار بازگشتی **ندارد** .

CheckBox.SetText: با استفاده از این دستور می‌توانید متن CheckBox را تغییر دهید.ورودی دستور نام شی و رشته ای برای جایگزینی متن است.

مقدار بازگشتی **ندارد** .

CheckBox.SetVisible: کنترل نمایش یا عدم نمایش CheckBox هم با این دستور امکان پذیر است. ورودی‌های کد، نام شی و عبارتی بولین برای فعال یا غیر فعال کردن نمایش شی CheckBox می‌باشد.

مقدار بازگشتی **ندارد** .

ComboBox

این تابع برای کار با شی ComboBox به کار می‌رود که دستورات زیر مجموعه آن به شرح زیر می‌باشد:

ComboBox.AddItem: با استفاده از این کد می‌توانید یک آیتم جدید به کمبوباکس اضافه کنید .در صورتی که گزینه Sort فعال نباشد، آیتم جدید در انتهای لیست اضافه می‌شود .یکی از ورودی‌های تابع نام کمبوباکس می‌باشد.ورودی‌های دیگر 2 رشته به عنوان متن و دیتا خواهند بود.

مقدار بازگشتی، عدد ردیفی است که آیتم جدید در آنجا قرار گرفته است. در صورت خطا 1- بازگشت داده می‌شود.

ComboBox.DeleteItem: برای حذف یک آیتم باید از این دستور استفاده کنید .نام کمبوباکس و عدد ردیف مورد نظر برای حذف از ورودی‌های تابع اند.

مقدار بازگشتی **ندارد** .

ComboBox.FindItem: با استفاده از این کد می‌توانید در آیتم‌های موجود، یک متن را جستجو نمایید.یکی از ورودی‌های این دستور نام کمبوباکس می‌باشد .ورودی دیگر عددی است برای مشخص کردن شروع جستجو (برای مثال با وارد کردن عدد 5، جستجو از سطر 5 به بعد انجام می‌شود). ورودی بعدی نوع جستجو می‌باشد .عدد 1 برای جستجو میان متن آیتم‌ها، 2 برای جستجو میان اطلاعات (دیتا) آیتم‌ها و گزینه 3 میان هر دو مورد. آخرین ورودی هم رشته ای است که نیاز دارید بین آیتم‌ها مشابهش را بیابید.در صورتی که بخواهید همان رشته را بیابید متن را میان دو علامت "" وارد کنید . در غیر این صورت می‌توانید از * و ؟ استفاده کنید تا مشابه آن متن را بیابید. برای مثال اگر دنبال کلمه‌ی Hamed می‌گردید، به صورت "Hamed" بنویسید و در صورتی که فقط میدانید ابتدای نام با H شروع شده به صورت "H*" تایپ نمایید.

مقدار بازگشتی این دستور عدد ردیفی است که متن در آن موجود است.در صورتی که متن یافت نشود یا خطایی صورت گیرد 1- بازگردانده می‌شود.

ComboBox.GetCount: این دستور نشان دهنده تعداد آیتم‌های موجود در کمبوباکس می‌باشد. ورودی دستور، نام شی می‌باشد.

مقدار بازگشتی نشان دهنده تعداد آیتم‌ها است.

ComboBox.GetItemData: با ورود نام شی و شماره ردیف مورد نظر به عنوان ورودی این دستور، می‌توانید دیتا موجود در آیتم مورد نظرتان را در یک رشته داشته باشید.

مقدار بازگشتی رشته ای حاوی دیتا آیتم می‌باشد.

ComboBox.GetItemText: با این دستور هم می‌توانید متن آیتم را بدست آورید. شی و شماره ردیف مورد نظر به عنوان ورودی این دستور خواهند بود.

مقدار بازگشتی رشته ای حاوی نام آیتم می‌باشد.

ComboBox.GetPos: با استفاده از این دستور می‌توانید موقعیت قرارگیری شی کمبوباکس در صفحه را بدست آورید .این موقعیت نسبت به گوشه‌ی بالا و سمت چپ محاسبه می‌شود.ورودی این تابع نام شی کمبوباکس می‌باشد.

مقدار بازگشتی یک آرایه با متغیرهای X و Y می‌باشد.

Combobox.GetProperties: این دستور خصوصیات یک کمبوباکس را در اختیار شما قرار می‌دهد .ورودی تابع نام شی می‌باشد.

مقدار بازگشتی یک آرایه با متغیرهای زیر می‌باشد :

ObjectName: نام شی

ComboStyle: نوع کمبوباکس، عدد 0 نشان دهنده غیر فعال بودن قابلیت تایپ و ویرایش توسط کاربر، 1 نشانگر فعال بودن این قابلیت

Sort: متغیری بولین برای نمایش فعال یا غیرفعال بودن مرتب سازی اتوماتیک

LinesToDisplay: تعداد آیتم‌های نمایش داده شونده در هنگام باز کردن شی

FontName: نام فونت استفاده شده

FontSize: اندازه فونت

FontStrikeout: فعال یا غیر فعال بودن strikeout در تنظیمات فونت (بولین)

FontUnderline: فعال یا غیر فعال بودن underline در تنظیمات فونت (بولین)

FontAntiAlias: فعال یا غیر فعال بودن anti alias در تنظیمات فونت (بولین)

FontItalic: فعال یا غیر فعال بودن italic در تنظیمات فونت (بولین)

FontWeight: نوع نوشته (تیرگی متن) به صورت آرایه ای متشکل از DONTCARE, THIN, EXTRALIGHT, LIGHT, NORMAL, MEDIUM, SEMIBOLD, BOLD, EXTRABOLD و HEAVY.

FontScript: نوع اسکریپت فونت به صورت آرایه ای متشکل از ANSI , BALTIC , CHINESEBIG5 , DEFAULT , EASTEUROPE , GB2312 , GREEK , HANGUL , MAC , OEM , RUSSIAN , SHIFTJIS ,SYMBOL و TURKISH .

BackgroundColor: شماره رنگ بک گراند (پس زمینه)

TextColor: شماره رنگ متن

ReadOrder: نوع نمایش متن در کمبوباکس (استاندارد یا راست به چپ_0 یا 1)

Enabled: فعال یا عدم فعال بودن دکمه (بولین)

Visible: دیده شدن یا عدم دیده شدن دکمه (بولین)

X: فاصله دکمه از سمت چپ صفحه

Y: فاصله دکمه از سمت راست صفحه

Width: عرض دکمه (پیکسل)

Height: طول دکمه (پیکسل)

TooltipText: متن نمایش دهنده زمانی که نشانگر ماوس روی کمبوباکس قرار می‌گیرد.

ResizeLeft: قابلیت تغییر اندازه از سمت چپ در زمان تغییر اندازه پروژه (بولین)

ResizeRight: قابلیت تغییر اندازه از سمت راست در زمان تغییر اندازه پروژه (بولین)

ResizeTop: قابلیت تغییر اندازه از بالا در زمان تغییر اندازه پروژه (بولین)

ResizeBottom: قابلیت تغییر اندازه از پایین در زمان تغییر اندازه‌ی پروژه (بولین)

WindowHandle: شماره هندل شی

ComboBox.GetSelected: این دستور شماره آیتم انتخاب شده را باز می‌گرداند .ورودی این دستور نام شی می‌باشد.

مقدار بازگشتی این دستور عدد ردیف انتخاب شده است .

ComboBox.GetSize: نمایش اندازه شی کمبوباکس به پیکسل با این دستور امکان پذیر است.ورودی دستور نام کمبوباکس می‌باشد.

مقدار بازگشتی دستور، آرایه ای 2 عضوی با اعضای "Width" و "Height" می‌باشد.

ComboBox.GetText: این دستور، متنی که در حال حاضر در کمبوباکس نمایش داده می‌شود را باز می‌گرداند.

مقدار بازگشتی متن موجود به صورت رشته است.

ComboBox.InsertItem: با استفاده از این کد می‌توانید آیتمی را در محل دلخواه خود اضافه نمایید.ورودی‌های دستور، نام شی، شماره ردیفی که می‌خواهید آیتم جدید در آنجا قرار گیرد و همچنین متن و دیتا مربوط به آیتم جدید می‌باشد.

مقدار بازگشتی، عدد ردیفی است که آیتم جدید در آنجا قرار گرفته است. در صورت خطا 1- بازگشت داده می‌شود.

ComboBox.IsEnabled: با استفاده از این دستور می‌توانید فعال یا غیرفعال بودن کمبوباکس را بدست آورید. مقدار ورودی نام شی کمبوباکس است.

مقدار بازگشتی Trueبه معنای فعال بودن و False به معنای غیرفعال بودن کمبوباکس می‌باشد.

ComboBox.IsVisible: با استفاده از این دستور می‌توانید نمایش یا عدم نمایش کمبوباکس در نرم افزار را بدست آورید. مقدار ورودی نام شی کمبوباکس است

مقدار بازگشتی True به نمایش یا False به معنای عدم نمایش کمبوباکس می‌باشد.

ComboBox.ResetContent: این دستور تمام اطلاعات کمبوباکس رو پاک می‌کند.نام شی به عنوان ورودی این دستور خواهد بود.

مقدار بازگشتی **ندارد** .

ComboBox.SetEnabled: با این دستور می‌توانید کمبوباکس مورد نظر را فعال یا غیر فعال کنید.ورودی دستور نام دکمه مورد نظر و همچنین عبارت بولین برای فعال یا غیرفعال کردن دکمه می‌باشد.

مقدار بازگشتی **ندارد** .

ComboBox.SetItemData: این دستور وظیفه‌ی قرار دادن عبارت جدید به عنوان دیتا برای آیتمی خاص را بر عهده دارد.ورودی‌های دستور، نام شی، ردیف مورد نظر به صورت عدد و رشته ای برای جای گیری در دیتا می‌باشد.

مقدار بازگشتی **ندارد** .

ComboBox.SetItemText: با این دستور می‌توانید متن آیتم خاص در کمبوباکس را تغییر دهید و متن جدید جایگزین کنید .این دستور هم سه ورودی نیاز دارد که یکی نام شی، دیگری شماره ردیف برای جای گیری متن جدید و سومین ورودی هم متن جدید می‌باشد.

مقدار بازگشتی **ندارد** .

ComboBox.SetPos: با این کد می‌توانید کمبوباکس را به محل دلخواه در صفحه منتقل نمایید.ورودی‌های این کد نام شی، عددی برای جای گیری در قسمت X (فاصله از چپ) و عدد دیگری برای جای گیری در Y (فاصله از بالا می‌باشد).

مقدار بازگشتی **ندارد** .

Combobox.SetProperties: با این دستور می‌توانید خصوصیات جدیدی برای کمبوباکس تعریف کنید.ورودی‌های این دستور یکی نام شی کمبوباکس و دیگری آرایه ای متشکل از خصوصیات دکمه است که این خصوصیات در Combobox.GetProperties توضیح داده شده است.

مقدار بازگشتی **ندارد** .

ComboBox.SetSelected: با این دستور می‌توانید آیتم خاصی را در کمبوباکس به حالت انتخاب در آورید.ورودی این تابع نام شی و شماره ردیف مورد نظر برای انتخاب می‌باشد .برای اینکه اگر آیتمی انتخاب شده، آن را از حالت انتخاب در آورید کافی است مقدار ورودی را برابر 1- قرار دهید.

مقدار بازگشتی **ندارد** .

ComboBox.SetSize: با این دستور می‌توانید طول و عرض کمبوباکس را به اندازه دلخواه خود تغییر دهید.ورودی‌های این کد نام شی کمبوباکس، عددی برای عرض و عددی دیگر برای طول دکمه می‌باشد.

مقدار بازگشتی **ندارد** .

ComboBox.SetText: با این دستور می‌توانید متن نمایش داده شده بر روی کمبوباکس را تغییر دهید. ورودی‌های این دستور عبارتند از نام شی و متن جدید به صورت رشته.این قابلیت زمانی قابل استفاده است که حالت Dropdown edit فعال باشد.

مقدار بازگشتی **ندارد** .

ComboBox.SetUpdate: این دستور این امکان را می‌دهد که در زمانی که نیاز ندارید کاربر تغییرات صورت گرفته بر روی کمبوباکس را ببیند، آپدیت کمبوباکس را غیر فعال و در صورت نیاز فعال کنید.، نام کمبوباکس و عبارتی بولین برای فعال و غیر فعال کردن آپدیت از ورودی‌های تابع هستند.

مقدار بازگشتی **ندارد** .

ComboBox.SetVisible: کنترل نمایش یا عدم نمایش کمبوباکس هم با این دستور امکان پذیر است. ورودی‌های کد، نام کمبوباکس و عبارتی بولین برای فعال یا غیر فعال کردن نمایش دکمه می‌باشد.

مقدار بازگشتی **ندارد** .

Crypto

از این تابع برای رمزنگاری و رمز گشایی فایل‌های مختلف استفاده می‌شود که دستورات زیر مجموعه‌ی آن به شرح زیر است:

Crypto.Base64DecodeFromFile: با استفاده از این دستور می‌توانید فایل رمزنگاری شده را به فایل اصلی بازگردانید. ورودی این دستور مسیر فایل رمزنگاری شده و مسیر برای فایل اصلی (رمزگشایی شده) می‌باشد.

مقدار بازگشتی **ندارد** .

Crypto.Base64DecodeFromString: با استفاده از این کد هم می‌توان رشته ای رمزنگاری شده را در یک فایل از حالت رمز گذاری خارج کرد. ورودی این کد رشته رمزنگاری شده و مسیر برای فایل اصلی (رمزگشایی شده) می‌باشد.

مقدار بازگشتی **ندارد** .

Crypto.Base64EncodeToFile: با این دستور می‌توانید فایلی را به روش Base64 رمزنگاری کنید و به صورت یک فایل ذخیره نمایید. ورودی‌های دستور، مسیر فایل برای رمزنگاری، مسیر فایل رمزنگاری شده و بیشترین طول برای فایل رمزنگاری شده (حالت استاندارد 76) می‌باشد.

مقدار بازگشتی **ندارد** .

Crypto.Base64EncodeToString: با این دستور می‌توانید فایلی را به روش Base64 رمزنگاری کنید و دریک رشته ذخیره نمایید. ورودی‌های دستور، مسیر فایل و بیشترین طول برای فایل رمزنگاری شده (حالت استاندارد 76) می‌باشد.

مقدار بازگشتی رشته رمزنگاری شده است.

Crypto.BlowfishDecrypt: این دستور فایل رمزنگاری شده به روش Blowfish را به حالت عادی باز می‌گرداند .ورودی‌های دستور، مسیر فایل رمزنگاری شده، مسیر برای فایل از رمزنگاری خارج شده و کد امنیتی که اطلاعات با آن رمزنگاری شده‌اند می‌باشد.

مقدار بازگشتی **ندارد** .

Crypto.BlowfishDecryptString: با استفاده از این دستور می‌توانید رشته رمزنگاری شده توسط دستور Crypto.BlowfishEncryptString را به حالت عادی بازگردانید.ورودی‌های دستور متن رمزنگاری شده و کد امنیتی آن است.

مقدار بازگشتی، متن رمزنگاری شده است.

Crypto.BlowfishEncrypt: این دستور نسخه ای رمزنگاری شده از یک فایل را در مسیر مشخص ایجاد می‌کند. ورودی دستور مسیر فایل اصلی، مسیر برای ذخیره فایل رمزنگاری شده و کد امنیتی می‌باشد.

مقدار بازگشتی **ندارد** .

Crypto.BlowfishEncryptString: با این دستور می‌توانید رشته ای را رمزنگاری کنید .ورودی‌های این دستور متن، کد امنیتی و بیشترین طول برای رشته رمزنگاری شده است.

مقدار بازگشتی رشته رمزنگاری شده می‌باشد.

Crypto.MD5DigestFromFile: رمزنگاری فایل مورد نظر به MD5 از طریق این دستور امکان پذیر است. ورودی دستور مسیر فایل برای رمزنگاری است.

مقدار بازگشتی متنی حاوی پیغام محاسبه شده برای فایل به صورت MD5 می‌باشد.

لازم به ذکر است این نوع رمزنگاری بازگشت ندارد و قابل رمزگشایی نمی‌باشد.

Crypto.MD5DigestFromString: رمزنگاری رشته به MD5 هم از طریق این دستور امکان پذیر است. ورودی دستور رشته برای رمزنگاری است .

مقدار بازگشتی متنی حاوی پیغام محاسبه شده برای فایل به صورت MD5 می‌باشد.

Crypto.Rot13: نوع ساده ای از رمزنگاری که بجای هر حرف، 13 حرف بعد آن را نمایش می‌دهد .مثلاً به جای H حرف Uرا نمایش می‌دهد و بالعکس. ورودی دستور متن برای تغییر است.

مقدار بازگشتی هم متنی تغییر یافته می‌باشد.

Debug

این تابع برای کار با پنجره‌ی مخصوص اشکال زدایی یعنی همان پنجره Debug به کار می‌رود. دستورات زیر مجموعه‌ی از تابع به شرح زیر می‌باشد:

Debug.Clear: این دستور وظیفه پاک کردن تمام اطلاعات و نوشته های داخل پنجره Debug را بر عهده دارد.

مقدار بازگشتی **ندارد** .

Debug.GetEventContext: با استفاده از این دستور می‌توانید محل اجرای آخرین کد را بدست آورید.

مقدار بازگشتی رشته ای است نشان دهنده محل اجرای آخرین کد، مثل :

Page2 -> Label -> On Click

Debug.GetTraceMode: این دستور TraceMode وضعیت فعال یا غیر فعال بودن TraceMode را باز می‌گرداند.

مقدار بازگشتی عبارتی بولین می‌باشد.

Debug.Print: نمایش متن دلخواه که ورودی تابع می‌باشد، در پنجره Debug توسط این دستور امکان پذیر است.

مقدار بازگشتی **ندارد** .

Debug.SendToFile: با استفاده از این دستور می‌توانید تمام خروجی‌های Debug را در فایل ذخیره نمایید.ورودی‌های دستور مسیر فایل برای ذخیره سازی و متغیری بولین برای زمانی که فایل موجود باشد (جایگزینی اطلاعات روی فایل قبلی یا نوشتن در ادامه فایل) می‌باشد

مقدار بازگشتی عددی نمایش دهنده وضعیت فایل می‌باشد.

Debug.SetTraceMode: این دستور TraceMode را فعال یا غیر فعال می‌کند .ورودی، متغیر بولین می‌باشد.

مقدار بازگشتی **ندارد** .

Debug.ShowWindow: نمایش یا عدم نمایش پنجره Debug با استفاده از این دستور امکان پذیر است.ورودی، متغیر بولین می‌باشد.

مقدار بازگشتی **ندارد** .

Dialog

این تابع برای کار با کادر پیغام به کار می‌رود و دستورات زیر مجموعه‌ی آن به شرح زیر است:

Dialog.ComboBox: با استفاده از این کد می‌توانید دیالوگی نمایش دهید که متشکل از یک کمبوباکس است.مقادیر ورودی این دستور عبارتند از: رشته ای به عنوان دیالوگ، رشته ای برای نمایش در بالای کمبوباکس، آیتم‌های تشکیل دهنده کمبوباکس، آیتم پیش‌فرض، عبارت بولین برای نمایش با مرتب سازی یا بدون مرتب سازی، عبارتی بولین برای فعال یا غیر فعال کردن قابلیت ویرایش کمبوباکس توسط کاربر و عددی برای نمایش آیکون

مقدار بازگشتی رشته ای حاوی متن انتخاب شده از کمبوباکس یا متن وارد شده توسط کاربر است.در صورت بروز خطا رشته خالی بازگردانده می‌شود .در صورت انتخاب دکمه cancel، عبارت "CANCEL " در رشته قرار می‌گیرد.

مثال:

```
Items = {"vasva3.com", "autoplay.ir"};
Selected = Dialog.ComboBox("Select:", "Please Select Default URL:", Items, "vasva3.com", true, false, MB_ICONQUESTION);
if ((Selected ~= "") and (Selected ~= "CANCEL")) then
   Dialog.Message("Result", "The chosen URL is "..Selected);
end
```

Dialog.FileBrowse: نمایش دیالوگ برای بدست آوردن آدرس فایل برای باز کردن یا ذخیره با این دستور امکان پذیر است.ورودی‌های دستور عبارتند از: عبارت بولین برای مشخص کردن نوع دیالوگ (True برای Open و false برای Save)، رشته ای برای عنوان دیالوگ، پوشه پیش‌فرض، فرمت فایل‌های انتخابی به صورت رشته، نام فایل برای نمایش در قسمت file name ، فرمت پیش‌فرض فایل، عبارت بولین برای فعال کردن انتخاب همزمان چند فایل و مقدار بولین برای مشخص کردن وجود یا عدم وجود فایل برای باز کردن .

مقدار بازگشتی، آرایه ای حاوی خانه‌هایی به صورت رشته حاوی مسیر فایل (فایل‌های) انتخاب شده است.در صورتی که دکمه Cancel انتخاب شود، رشته "CANCEL " در خانه اول آرایه قرار می‌گیرد. (Array[1] = "CANCEL")

Dialog.FolderBrowse: با این دستور می‌توانید دیالوگی را برای انتخاب یک پوشه توسط کاربر نمایش دهید .ورودی‌های تابع، یکی عنوان پنجره و دیگری مسیر پیش‌فرض برای نمایش می‌باشد.

مقدار بازگشتی مسیر پوشه خواهد بود .در صورت انتخاب Cancel رشته برابر "CANCEL" و در صورت بروز خطا رشته خالی خواهد بود.

Dialog.Input: با این دستور می‌توانید دیالوگی با یک فیلد ورودی برای گرفتن اطلاعات از کاربر نمایش دهید .تابع شامل 4 ورودی عنوان، متن برای نمایش بالای فیلد ورودی (Input)، متن پیش فرض و آیکون می‌باشد.

مقدار بازگشتی، رشته ای حاوی اطلاعات وارد شده در Input می‌باشد . در صورت انتخاب Cancel رشته برابر "CANCEL" و در صورت بروز خطا رشته خالی خواهد بود.

Dialog.MaskedInput: با این دستور می‌توانید دیالوگی با یک فیلد ورودی برای گرفتن اطلاعات از کاربر نمایش دهید .تابع شامل 5 ورودی عنوان، متن برای نمایش بالای فیلد ورودی (Input)، متن پیش فرض +، آیکون و نوع عبارت ورودی می‌باشد.

مقدار بازگشتی، رشته ای حاوی جدولی با متغیرهای Displayed و Data می‌باشد . در صورت انتخاب Cancel رشته برابر "CANCEL" و در صورت بروز خطا رشته خالی خواهد بود.

Dialog.Message: با استفاده از این دستور می‌توانید پیغامی را به صورت دیالوگ به کاربر نمایش دهید .ورودی‌های دستور عبارتند از :

عنوان دیالوگ، متن دیالوگ، نوع دکمه های دیالوگ (0: فقط دکمه OK، 1 :دکمه های OK و Cancel، 2: دکمه های Abort، Ignore و Retry، 3:دکمه های Yes، No و Cancel ، 4: دکمه های Yes وNo، 5:دکمه های Retry و Cancel)، آیکون، حالت پیش‌فرض دکمه

مقدار بازگشتی عدد مربوط به دکمه فشرده شده می‌باشد.

عدد مربوط به کدها عبارتند از :

No:7 Yes:6 Ignore:5 Retry:4 Abort:3Cancel :2 OK:1

در صورت بروز خطا کد مربوط به Cancel یعنی عدد 2 بازگردانده می‌شود.

Dialog.PageSearch: جستجو میان صفحات بر اساس کلمات کلیدی توسط این دستور امکان پذیر است.ورودی تابع جدولی است متشکل از عناصر دیالوگ.

مقدار بازگشتی **ندارد** .

Dialog.PasswordInput: با استفاده از این دستور می‌توانید دیالوگ همراه با فیلد ورودی به کاربر نشان دهید که با ورود متن در فیلد متنی، کاراکترها به صورت * نمایش داده شوند .از این قسمت بیشتر برای دریافت پسورد استفاده می‌شود .ورودی‌های دستور، عنوان دیالوگ، متن نمایش داده شده در بالای فیلد و آیکون می‌باشد.

مقدار بازگشتی رشته ای حاوی متن وارد شده می‌باشد . در صورت انتخاب Cancel رشته برابر "CANCEL" و در صورت بروز خطا رشته خالی خواهد بود.

Dialog.SplashFlash: نمایش فایل فلش در دیالوگ توسط این دستور انجام می‌پذیرد.ورودی‌های دستور، مسیر فایل فلش (با فرمت swf)، زمان برای نمایش فلش (در صورتی که از 0 استفاده می‌کنید، فلش تا زمانی نمایش داده می‌شود که در

FSCOMMAND به مقدار quit برسد)، و مقداری بولین برای فعال یا غیرفعال کردن قابلیت بسته شدن پنجره توسط کلیک ماوس بر روی آن است.

مقدار بازگشتی **ندارد** .

Dialog.SplashImage: نمایش تصویر در دیالوگ توسط این دستور انجام می‌پذیرد.ورودی‌های دستور، مسیر تصویر، زمان برای نمایش تصویر (به ثانیه)، و مقداری بولین برای فعال یا غیرفعال کردن قابلیت بسته شدن پنجره توسط کلیک ماوس بر روی آن است.

مقدار بازگشتی **ندارد** .

Dialog.SplashVideo: نمایش فایل ویدئو در دیالوگ توسط این دستور انجام می‌پذیرد. ورودی‌های دستور، مسیر فایل، زمان برای نمایش ویدئو (به ثانیه، در صورت ورود 0 تا انتهای ویدئو پخش خواهد شد)، و مقداری بولین برای فعال یا غیرفعال کردن قابلیت بسته شدن پنجره توسط کلیک ماوس بر روی آن است.

مقدار بازگشتی **ندارد** .

Dialog.TimedMessage: با استفاده از این دستور هم می‌توانید پیغامی را در دیالوگ برای زمان معین به نمایش درآورید. ورودی‌های دستور عنوان پنجره، متن قابل نمایش در دیالوگ، مدت زمان نمایش دیالوگ (میلی ثانیه) و آیکون می‌باشد.

مقدار بازگشتی **ندارد** .

DialogEx

با استفاده از این تابع می‌توان پنجره های فرعی از نوع DialogEx را مدیریت نمود. دستورات زیر مجموعه این تابع به شرح زیر می‌باشد:

DialogEx.ClickObject: این دستور با گرفتن نام شی مورد نظر در دیالوگ، کدهای On Click آن شی را اجرا می‌کند.

مقدار بازگشتی **ندارد** .

DialogEx.Close: با استفاده از این دستور می‌توانید دیالوگ حاضر را ببندید.

مقدار بازگشتی متغیری بولین، نشان دهنده موفقیت آمیز بودن یا نبودن بسته شدن دیالوگ است.

DialogEx.CreateObject: با استفاده از این دستور می‌توانید شی خاصی را در دیالوگ بسازید. ورودی‌های این دستور، عددی برای مشخص کردن نوع شی درخواستی، نام شی جدید

(دقت داشته باشید نام شی با اشیاء داخل دیالوگ مشابه نباشد)، جدولی متشکل از خصوصیات شی است.(جدول خصوصیات هر شی در دستورات مربوط به همان شی نوشته شده است.)

مقدار بازگشتی **ندارد** .

DialogEx.DeleteObject: برای حذف شی مورد نظر از دیالوگ می‌توانید از این دستور استفاده کنید .ورودی کد، نام شی می‌باشد.

مقدار بازگشتی **ندارد** .

DialogEx.EnumerateObjects: برای بدست آوردن نام تمام اشیاء موجود در دیالوگ، از این دستور باید استفاده نمایید.

مقدار بازگشتی، آرایه ای متشکل از نام اشیاء می‌باشد. در صورت عدم وجود شی در دیالوگ یا بروز مشکل مقدار nil بازگردانده می‌شود.

DialogEx.GetFocus: با استفاده از این دستور می‌توانید شی که بر روی آن Focus شده را بدست آورید.

مقدار بازگشتی نام شی که بر روی آن Focus شده خواهد بود.

DialogEx.GetObjectScript: با استفاده از این دستور می‌توانید دستورات رویداد مورد نظرتان در شی خاص در دیالوگ حاضر را بدست آورید.ورودی‌های دستور، نام شی و رویداد مورد نظر می‌باشد.

مقدار بازگشتی رشته ای حاوی دستورات بدست آمده می‌باشد.

DialogEx.GetObjectType: این دستور با گرفتن نام شی به عنوان ورودی، نوع شی را باز می‌گرداند.

مقدار بازگشتی عددی نمایش دهنده نوع شی است.

BUTTON(0) , LABEL(1) , PARAGRAPH(2) , IMAGE(3) , FLASH(4) , VIDEO(5) , WEB(6) , INPUT(7) , HOTSPOT(8) , LISTBOX(9) , COMBOBOX(10) , PROGRESS(11) , TREE(12) , RADIOBUTTON(13) , RICHTEXT(14) , CHECKBOX(15) , SLIDESHOW(16) , GRID(17) , PDF(18) , QUICKTIME(19) , XBUTTON(20) , PLUGIN(40)

DialogEx.GetRadioValue: با استفاده از این دستور می‌توانید اطلاعات مربوط به دکمه رادیویی انتخاب شده موجود در دیالوگ را بدست آوردید.ورودی دستور، کد گروه دکمه رادیویی و نوع داده نیاز (Value، نام شی یا متن آن) می‌باشد.

مقدار بازگشتی رشته ای حاوی نوع داده خواسته شده است .در صورتی که دکمه رادیویی انتخاب نشده باشد یا خطایی صورت گرفته باشد رشته ای خالی بازگردانده می‌شود.

DialogEx.GetSize: با این دستور می‌توانید اندازه دیالوگ حاضر را به پیکسل به دست آورید.

مقدار بازگشتی آرایه ای متشکل از خانه های Width و Height می‌باشد.

DialogEx.GetWndHandle: این دستور شماره Handle دیالوگ موجود را بازمی‌گرداند.

مقدار بازگشتی شماره Handle دیالوگ می‌باشد.در صورت بروز خطا، 1- بازگردانده می‌شود.

DialogEx.Print: با استفاده از این دستور می‌توانید از دیالوگ موجود پرینت بگیرید .شی ویدئو در پرینت نمایش داده نمی‌شود.ورودی‌ها، دو متغیر بولین است .یکی نمایش و عدم نمایش دیالوگ پرینت قبل از عمل پرینت، دیگری یکی کردن اندازه تصویر دیالوگ با صفحه پرینت شده است.

مقدار بازگشتی **ندارد** .

DialogEx.Redraw: با این دستور می‌توانید اشیاء را دوباره رسم کنید.

مقدار بازگشتی **ندارد** .

DialogEx.SetFocus: با استفاده از این دستور می‌توانید بر روی شی خاص Focus نمایید. فوکوس تنها بر روی اشیاء Flash, input, listbox, combobox, tree و Web انجام می‌شود .ورودی دستور نام شی مورد نظر است.

مقدار بازگشتی **ندارد** .

DialogEx.SetObjectScript: با استفاده از این دستور می‌توانید دستورات خاصی را در رویداد مورد نظر شی موجود در دیالوگ قرار دهید.ورودی‌های تابع، نام شی، رویداد مورد نظر، و دستورات مورد نیاز می‌باشد.

مقدار بازگشتی **ندارد** .

DialogEx.SetObjectZOrder: با استفاده از این دستور می‌توانید نوع چینش شی را تعیین کنید، نام شی، نوع چینش (FRONT(0) , BACK(1) , FORWARD(2) , BACKWARD(3) , INSERT_BEFORE(4) یا BEHIND(5))

مقدار بازگشتی **ندارد** .

DialogEx.SetRadioValue: با این دستور می‌توانید دکمه رادیویی خاصی را در دیالوگ به انتخاب درآورید ورودی‌ها عبارتند از: متن داده انتخاب شده، کد گروه دکمه رادیویی و نوع داده (VALUE, OBJECTNAME, TEXT که متن آن در ورودی اول وارد شده) .

مقدار بازگشتی **ندارد** .

DialogEx.SetRedraw: با این دستور می‌توانید نمایش تغییرات را فعال یا غیر فعال کنید.ورودی این دستور یک بولین می‌باشد.

مقدار بازگشتی **ندارد** .

DialogEx.Show: با این دستور می‌توانید دیالوگ را به نمایش درآورید.ورودی تابع نام دیالوگ، مقداری بولین (true فاصله دیالوگ را از پنجره والد و false فاصله از کل صفحه را تعیین می‌کند.)، فاصله از چپ و فاصله از بالا (فاصله‌ها نسبت به گوشه سمت چپ، بالا محسوب می‌شود.)

مقدار بازگشتی عددی نمایش دهنده کارکرد درست کد است .

DialogEx.StartTimer: با اجرای این کد، تایمر (شمارنده) مربوط به دیالوگ فعال می‌شود .ورودی دستور زمان به میلی ثانیه و کد (ID) تایمر می‌باشد.

مقدار بازگشتی **ندارد**.

DialogEx.StopTimer: با استفاده از این دستور می‌توانید تایمر فعال شده با کد قبلی را غیرفعال نمایید.برای این کار کافی است کد تایمر را به عنوان ورودی دستور وارد نمایید.

مقدار بازگشتی **ندارد**.

DLL

این تابع برای فراخوانی توابع موجود در فایل‌های DLL به کار می‌رود که شامل یک زیر مجموعه به شرح زیر است:

DLL.CallFunction: با استفاده از این دستور قادر به اجرای توابع مدنظر از فایل DLL خاص خواهید بود. به این صورت که مسیر فایل DLL را به عنوان ورودی وارد می‌کنید . ورودی‌های دیگر نیز عبارتند از نام تابع داخل فایل، پارامترهای ورودی تابع (با , از هم جدا می‌شوند)، نوع داده بازگشتی (INTEGER(0) , LONG(1) یا STRING(2)) و نوع فراخوانی تابع (CDECL(0) یا STDCALL(1)) .

مقدار بازگشتی رشته ای است حاوی مقدار بازگشتی تابع .در صورت بروز خطا رشته خالی بازگردانده می‌شود.

Drive

این تابع برای کار با درایوهای موجود در سیستم شما به کار می‌رود که دستورات زیر مجموعه‌ی آن به شرح زیر است:

Drive.Eject: با این دستور می‌توانید درایوCD یا DVD خود را باز کنید .ورودی دستور نام درایو خواهد بود. برای مثال برای باز کردن درایو H کافی است یکی از مقادیر زیر وارد شو .(فقط کاراکتر اول به عنوان نام درایو خوانده می‌شود.)

"H" , "H:" , "H:\" , "H:\Test.vdb" یا ....

برای باز کردن درایوی که برنامه از روی آن اجرا شده نیاز به وارد کردن نام درایو نیست .کافی است این دستور را اجرا نمایید:

```
type = Drive.GetType(_SourceDrive);
if (type == DRIVE_CDROM) then
   Drive.Eject(_SourceDrive);
end
```

مقدار بازگشتی **ندارد** .

Drive.Enumerate: با استفاده از این دستور می‌توانید نام درایور های موجود در سیستم کاربر را بدست آورید.

مقدار بازگشتی آرایه ای متشکل از نام درایورهاست. در صورت بروز مشکل nil بازگردانده می‌شود.

Drive.GetFreeSpace: با استفاده از این دستور می‌توانید مقدار فضای خالی درایو را بدست آورید .نام درایو مورد نظر به عنوان ورودی تابع است.

مقدار بازگشتی میزان فضای خالی درایو بر حسب مگابایت است.

Drive.GetInformation: اطلاعات مربوط به درایو را می‌توانید با این دستور به دست آورید. نام درایو ورودی دستور خواهد بود.

مقدار بازگشتی آرایه ای با متغیرهای زیر می‌باشد:

Label: برچسب درایو

FileSystem: نوع درایو (Fat, Fat32, NTFS, …)

SerialNumber: سریال درایو

DisplayName: نام نمایش داده شده بر روی درایو

Drive.GetSize: حجم کلی درایو با استفاده از این دستور بدست می‌آید. نام درایو به عنوان ورودی خواهد بود.

مقدار بازگشتی، حجم کلی درایو بر حسب مگابایت است.

Drive.GetType: با استفاده از این دستور می‌توانید نوع درایو را بدست آورید.نام درایو به عنوان ورودی دستور است.

مقدار بازگشتی، عددی است نمایش دهنده نوع درایو :

UNKNOWN(0) , NO_ROOT_DIR(1) , REMOVABLE(2) , FIXED(3) , REMOTE(4) ,CDROM(5) , RAMDISK(6)

Drive.GetUsedSpace: میزان پرشده از درایو مورد نظر که به عنوان ورودی تابع خواهد بود، با این دستور به دست می‌آید.

مقدار بازگشتی، میزان حجم پر شده بر حسب مگابایت است.

File

این تابع برای کار با فایل‌ها مورد استفاده قرار می‌گیرد که دارای دستورات زیر مجموعه های زیر می‌باشد:

توجه: تابع بازگشت داده شده توسط دستورات به صورت مجزا توضیح داده خواهد شد.

File.Copy: با استفاده از این دستور می‌توانید فایل را به محل دیگر کپی کنید .ورودی‌های تابع، مسیر فایل یا فایل‌هایی که می‌خواهید آن‌ها را کپی کنید، (در صورت نیاز به کپی چندین فایل می‌توانید از * استفاده کنید.)، مسیر جدید برای قرارگیری فایل‌ها، متغیری بولین برای تعیین Resource (در صورتی که از * برای کپی چند فایل استفاده کرده باشید با true کردن این قسمت، فایل‌های داخل فولدرهای درونی هم کپی می‌شوند)، مقداری بولین برای جایگزینی فایل‌های همنام یا عدم جایگزینی، مقدار بولین برای قطع یا ادامه دادن در صورت بروز مشکل و خطا و مقدار بولینی دیگر برای فعال یا غیرفعال کردن فایل‌های مخفی، نام تابع بازگشتی است.

مقدار بازگشتی **ندارد** .

File.Delete: پاک کردن فایل‌های مورد نیاز با استفاده از این دستور امکان پذیر است. ورودی‌های تابع، مسیر فایل یا فایل‌هایی که می‌خواهید آن‌ها را حذف نمایید، متغیری بولین برای فعال و غیرفعال کردن پاک کردن زیر شاخه‌ها، ، مقدار بولین برای قطع یا ادامه دادن در صورت بروز مشکل و خطا و نام تابع بازگشتی است.

مقدار بازگشتی **ندارد**.

File.DeleteOnReboot: با این دستور می‌توانید فایلی را انتخاب کنید تا در هنگام راه اندازی مجدد سیستم پاک شوند .ورودی، مسیر فایل مورد نظر است.

مقدار بازگشتی **ندارد**.

File.DoesExist: با این دستور هم می‌توانید وجود یک فایل را کنترل نمایید .ورودی مسیر فایل است.

مقدار بازگشتی عبارتی بولین نمایش دهنده وجود یا عدم وجود فایل مورد نظر است.

File.ExploreFolder: با این دستور این امکان به شما داده می‌شود که شاخه مورد نظرتان را به حالت Explore باز کنید.ورودی دستور، مسیر شاخه و نوع باز شدن پنجره (معمولی، بزرگ‌ترین حالت، مینی مایز) است.

مقدار بازگشتی **ندارد**.

File.Find: با این دستور می‌توانید فایل خاصی را جستجو نمایید.ورودی‌های دستور، شاخه ای که در آن باید جستجو انجام شود، نام فایل (با * می‌توانید چندین نام را جستجو نمایید)، متغیری بولین برای فعال یا غیرفعال کردن جستجو در زیرشاخه‌ها، متغیری بولین برای شامل شدن فولدرها در مسیر دهی، نام تابع بازگشتی و تابع بازگشتی پیدا شدن فایل است.

مقدار بازگشتی آرایه ای متشکل از مسیر فایل‌های یافت شده است.در صورتی که فایل یافت نشود یا خطایی صورت گیرد، مقدار nil بازگردانده می‌شود.

File.GetAttributes: با این دستور می‌توانید خصوصیات فایل خاص را که باید نام آن را به عنوان ورودی قرار دهید را بدست آورید.

مقدار بازگشتی آرایه ای است متشکل از :

CreationDate: زمان ایجاد فایل

CreationDateISO: زمان ایجاد فایل با فرمت ISO

AccessDate: زمان آخرین دسترسی به فایل

AccessDateISO: زمان آخرین دسترسی به فایل با فرمت ISO

WriteDate: زمان آخرین نوشتن در فایل

WriteDateISO: زمان آخرین نوشتن در فایل با فرمت ISO

Directory: True در صورتی که خصوصیت Directory فعال باشد

True:Archived در صورتی که خصوصیت Archived فعال باشد

True:ReadOnly در صورتی که خصوصیت ReadOnly فعال باشد

True:Compressed در صورتی که خصوصیت Compressed فعال باشد

True:System در صورتی که خصوصیت System فعال باشد

True:Hidden در صورتی که خصوصیت Hidden فعال باشد

True:Temporary در صورتی که خصوصیت Temporary فعال باشد

True:Normal در صورتی که خصوصیت Normal فعال باشد

File.GetCRC: مقدار CRC فایل با این دستور به دست می‌آید .مسیر فایل به عنوان ورودی استفاده می‌شود.

مقدار بازگشتی عدد مربوط به CRC فایل است.

File.GetDefaultViewer: این دستور برنامه بازکننده فرمت مورد نظر را نشان می‌دهد .ورودی فرمت مورد نظر (مثلاً jpg.) است.

مقدار بازگشتی مسیر برنامه اجرا کننده فرمت مورد نظر است.

File.GetShortName: این دستور مسیر کوتاه شده را باز می‌گرداند .برای مثال در صورت ورود مسیر به صورت:

```
C:\\Program Files\\AutoPlay Media Studio 8\\AutoPlayDesign.exe
```

مسیر کوتاه شده به صورت زیر بازگردانده می‌شود:

```
C:\PROGRA~1\AUTOPL~1\AutoPlayDesign.exe
```

مقدار بازگشتی رشته ای است حاوی مسیر کوتاه شده .

File.GetSize: با ورود مسیر فایل، می‌توانید حجم آن را بر حسب بایت به دست آورید.

مقدار بازگشتی حجم فایل به صورت بایت است. (با تقسیم این حجم به 1024، حجم به کیلو بایت، با تقسیم مجدد به مگابایت و ... تبدیل می‌شود.)

File.GetVersionInfo: با این دستور می‌توانید اطلاعات مربوط به ورژن فایل را به دست آورید. مسیر فایل به عنوان ورودی دستور است.

مقدار بازگشتی آرایه ای متشکل از موارد زیر می‌باشد :

FileVersion: نسخه فایل (مثلاً 10.2.0.3)

ProductVersion: نسخه محصول (1.020.0)

CompanyName: نام شرکت سازنده

FileDescription: شرحی کوتاه مربوط به فایل

InternalName: نام داخلی فایل

ProductName: نام محصول

LegalCopyright: قانون کپی رایت

LegalTrademarks: قانون علامت تجاری

Comments: توضیحات فایل

OriginalFilename: نام اصلی فایل (بدون مسیر)

PrivateBuild: اطلاعات مربوط به نسخه محرمانه

SpecialBuild: تفاوت نسخه مخصوص با نسخه های استاندارد

هرکدام از مقادیر بالا در دسترس نباشد، رشته مربوط به آن خالی بازگشت داده می‌شود.

File.Install: با این دستور می‌توانید فایلی را نصب نمایید.ورودی‌ها، مسیر فایل برای نصب، محل نصب و نوع جایگذاری فایل‌ها،متغیر بولین برای گرفتن پشتیبان (فایل‌های تکراری)، متغیر بولین برای فعال یا غیرفعال کردن به اشتراک گذاشتن فایل‌ها، مقدار بازگشتی تابع Progress، مقدار بازگشتی تابع جایگذاری فایل‌ها است.

مقدار بازگشتی متغیر بولین برای تایید تکمیل نصب نرم افزار است.

File.IsInUse: با استفاده از این دستور می‌توانید وضعیت در حال اجرا بودن یا نبودن یک فایل خاص را کنترل کنید.ورودی مسیر فایل است.

مقدار بازگشتی متغیر بولین است که True نشان دهنده در حال اجرا بودن فایل می‌باشد.

در صورتی که فایل در حال اجرا باشد، فایل حذف نمی‌شود و فایلی هم جایگزین آن نمی‌شود.

File.Move: با استفاده از این دستور می‌توانید فایل را به محل دیگری جابجا کنید. ورودی‌های تابع، مسیر فایل یا فایل‌هایی که می‌خواهید آن‌ها را کپی کنید، (در صورت نیاز به کپی چندین فایل می‌توانید از * استفاده کنید.)، مسیر جدید برای قرارگیری فایل‌ها، متغیری بولین برای فعال کردن جابجایی فایل‌های درون زیر شاخه‌ها، مقداری بولین برای جایگزینی فایل‌های همنام یا عدم جایگزینی، مقدار بولین برای قطع یا ادامه دادن در صورت بروز مشکل و خطا و مقدار بولینی دیگر برای فعال یا غیرفعال کردن فایل‌های مخفی، نام تابع بازگشتی است.

مقدار بازگشتی **ندارد** .

File.MoveOnReboot: با این دستور می‌توانید فایل را در هنگام راه اندازی سیستم، منتقل کنید.ورودی دستور مسیر فایل و مسیر جدید برای قرارگیری می‌باشد.

مقدار بازگشتی **ندارد** .

File.Open: با این دستور می‌توانید فایلی را باز کنید .این فایل با برنامه پیش فرض باز می‌شود .ورودی‌ها، مسیر فایل، مسیر اجرایی فایل، نوع باز شدن پنجره می‌باشد.

مسیر اجرایی فایل زمانی تعیین می‌شود که فایل اجرایی نیاز به استفاده از فایل‌های کنار خود دارد.

مقدار بازگشتی **ندارد** .

File.OpenEmail: با این دستور، برنامه پیش فرض ارسال ایمیل بازشده و آدرس ایمیل مورد نظرتان در قسمت to قرار می‌گیرد. ورودی‌های دستور، نام ایمیل و نوع باز شدن پنجره (استاندارد، بزرگ‌ترین حالت ممکن و مینی مایز) می‌باشد.

لازم به ذکر است در صورتی که بخواهید موضوع را هم اضافه کنید می‌توانید پس از آدرس ایمیل با فرمت ?subject=X موضوع دلخواهتان را به جای X قرار دهید. برای مثال:

```
File.OpenEmail("hamed@vasva3.com?subject=AMS 8 Learning book";)
```

File.OpenURL: با این دستور هم می‌توانید آدرس خاصی را در مرورگر پیش فرض باز نمایید.آدرس و نوع باز شدن پنجره از ورودی‌های دستور هستند.

مقدار بازگشتی **ندارد**.

File.Print: با استفاده از این دستور می‌توانید سند مورد نظرتان را چاپ کنید .ورودی دستور مسیر فایل است.

مقدار بازگشتی **ندارد** .

File.Rename: این دستور به شما این امکان تغییر نام (و حتی فرمت) فایل را به شما می‌دهد .کافی است در فیلد اول ورودی مسیر فایل و در فیلد دوم مسیر فایل به همراه نام جدید و فرمت جدید را وارد نمایید.

مقدار بازگشتی **ندارد** .

مثال:

```
File.Rename("C:\\Book\\T1.zip", "C:\\Book\\T2.vdb;("
```

File.Run: اجرای یک فایل هم با این دستور اماکن پذیر است. ورودی‌های دستور، نام فایل، آرگومانی که باید به فایل فرستاده شود، محل کار فایل (مسیر اجرایی)، نوع باز شدن پنجره و مقداری بولین برای مشخص کردن انتظار یا عدم انتظار برای بسته شدن برنامه است. (این قابلیت به شما امکان غیرفعال شدن نرم افزار تا انتهای اجرای فایل را می‌دهد)

مقدار بازگشتی عدد 0 یا 1 نمایش دهنده انتظار یا عدم انتظار است .

File.RunAs: این دستور هم مانند دستور بالا عمل می‌کند با تفاوت اینکه دستور اجازه انتخاب User دیگری را برای اجرای فایل می‌دهد.این دستور هم 5 ورودی بالا را دارد. به علاوه آن‌ها، نام کاربری، پسورد، دامینی که User از روی آن اجرا می‌شود، پرچم لاگین، تنظیمات ساخت و اطلاعات پیغام خطا می‌باشد.

مقدار بازگشتی عدد 0 یا 1 نمایش دهنده انتظار یا عدم انتظار است .

File.RunOnReboot: این دستور اجازه اجرا کردن یک فایل در هنگام راه اندازی مجدد سیستم را به شما می‌دهد .مسیر فایل و آرگومان به عنوان ورودی‌های دستور خواهند بود.

مقدار بازگشتی **ندارد** .

File.SetAttributes: با این دستور می‌توانید خصوصیات فایل را تغییر دهید .خصوصیات قابل تغییر عبارتند از Archived , ReadOnly , Compressed , System , Hidden , Temporary , Normal , که در یک آرایه به عنوان ورودی بعد از مسیر فایل قرار می‌گیرد.

مقدار بازگشتی **ندارد** .

File.SetPermissions: با این دستور می‌توانید دسترسی به فایل را تغییر دهید.مسیر فایل، گروه های دسترسی، نوع دسترسی، مقدار دسترسی و وراثت دسترسی می‌باشد.

مقدار بازگشتی یک عدد است که 0 نشان دهنده موفقیت آمیز بودن آن و غیر صفر شماره خطای اتفاق افتاده است.

Flash

این تابع برای کار با شی فلش(Flash) به کار می‌رود و دستورات زیر مجموعه‌ی آن به شرح زیر است.

Flash.CallFunction: با استفاده از این دستور می‌توانید توابع فایل فلش را فراخوانی کنید .ورودی‌های تابع، مسیر فایل و رشته با فرمت و آرایش XML است.

مقدار بازگشتی رشته ای با فرمت XML می‌باشد.

مثال:

```
strXMLResult = Flash.CallFunction("Flash1", "<invoke
name=\"loadVideo\"
returntype=\"xml\"><arguments><string>C:\\Book\\AutoPlay.flv</stri
ng></arguments></invoke>");
```

Flash.GetFilename: با ورود نام شی می‌توانید مسیر فایل بار گذاری شده روی شی را به دست آورید.

مقدار بازگشتی رشته ای حاوی مسیر فایل فلش می‌باشد.

Flash.GetFlashVariable: با این دستور می‌توانید مقدار متغیر موجود در فلش را بدست آورید. نام شی فلش و نام متغیر ورودی‌های دستور هستند.

مقدار بازگشتی رشته ای است حاوی مقدار متغیر مورد نظر.

Flash.GetPos: با استفاده از این دستور می‌توانید موقعیت قرارگیری شی فلش را بدست آورید .این موقعیت نسبت به گوشه‌ی بالا و سمت چپ محاسبه می‌شود.ورودی این تابع نام شی فلش است.

مقدار بازگشتی یک آرایه با متغیرهای X و Y می‌باشد.

Flash.GetProperties: با این دستور خصوصیات شی فلشی که نام آن را به عنوان ورودی قرار می‌دهید را خواهید داشت.

مقدار بازگشتی آرایه ای است حاوی :

ObjectName: نام شی

FlashFile: نام فایل استفاده شده

Alignment: نوع قرارگیری فلش در شی (چپ، بالا، وسط، ....)

Menu: نوع منو (0: استاندارد، 1: کامل)

Quality: کیفیت نمایش

ScalingMode: استیل قرارگیری فلش در شی

OverrideBackground: فعال یا غیرفعال بودن OverrideColor (بولین)

BGOverrideColor: رنگ برای جایگزین کردن تغییر رنگ پس زمینه در صورت True بودن گزینه قبل

DeviceFont: نوع نمایش متن فلش

AutoStart: نمایش خودکار فلش با باز شدن صفحه یا دیالوگ

Loop: اجرای مجدد و عدم اجرای آن در صورت اتمام (بولین)

Enabled: فعال یا عدم فعال بودن شی (بولین)

Visible: دیده شدن یا عدم دیده شدن شی (بولین)

X: فاصله شی از سمت چپ صفحه

Y: فاصله شی از سمت راست صفحه

Width: عرض شی (پیکسل)

Height: طول شی (پیکسل)

TooltipText: متن نمایش دهنده زمانی که نشانگر ماوس روی شی فلش قرار می‌گیرد.

ResizeLeft: قابلیت تغییر اندازه از سمت چپ در زمان تغییر اندازه پروژه (بولین)

ResizeRight: قابلیت تغییر اندازه از سمت راست در زمان تغییر اندازه پروژه (بولین)

ResizeTop: قابلیت تغییر اندازه از بالا در زمان تغییر اندازه پروژه (بولین)

ResizeBottom: قابلیت تغییر اندازه از پایین در زمان تغییر اندازه پروژه (بولین)

WindowHandle: شماره هندل پنجره شی

Flash.GetSize: نمایش اندازه شی فلش بر حسب پیکسل با این دستور امکان پذیر است.ورودی دستور نام شی می‌باشد.

مقدار بازگشتی دستور، آرایه ای 2 عضوی با اعضای "Width" و "Height" می‌باشد.

Flash.GetState: وضعیت لحظه ای فلش را با این کد می‌توانید به دست آورید. نم شی فلش ورودی دستور خواهد بود.

مقدار بازگشتی عدد مربوط به وضعیت فلش می‌باشد:

LOADING(0) , UNINITIALIZED (1) , LOADED(2) , INTERACTIVE(3) , COMPLETE(4)

Flash.IsEnabled: با استفاده از این دستور می‌توانید وضعیت فعال یا غیرفعال بودن شی فلش را بدست آورید. مقدار ورودی نام شی فلش است.

مقدار بازگشتی Trueبه معنای فعال بودن و False به معنای غیرفعال بودن فلش می‌باشد.

Flash.IsVisible: با استفاده از این دستور می‌توانید وضعیت نمایش یا عدم نمایش فلش را بدست آورید. مقدار ورودی، نام شی فلش است

مقدار بازگشتی Trueبه نمایش یا False به معنای عدم نمایش فلش می‌باشد.

Flash.Load: بارگذاری فایل فلش در شی فلش با این دستور امکان پذیر است. نام شی، مسیر فایل، مقداری بولین برای مشخص کردن نمایش خودکار یا عدم نمایش و مقداری بولین برای تکرار نمایش فلش در صورت اتمام، ورودی‌های دستور هستند.

مقدار بازگشتی **ندارد** .

Flash.Play: با قرار دادن نام شی فلش در این دستور قادر به پخش فلش خواهید بود.

مقدار بازگشتی **ندارد** .

Flash.Seek: پرش به موقعیت دلخواه صوت در فلش با این دستور امکان پذیر است. پارامتر ورودی اول نام شی فلش می‌باشد، پارامتر ورودی دوم نوع پرش است:

SEEK_BEGINNING: پرش به اول صوت

SEEK_END: پرش به انتهای صوت

SEEK_FORWARD: پرش به جلو

SEEK_BACKWARD: پرش به عقب

SEEK_SPECIFIC: پرش به زمان خاص

ورودی سوم هم زمان مناسب برای پرش (برای موارد سوم تا پنجم) استفاده می‌شود.

مقدار بازگشتی **ندارد** .

Flash.SetEnabled: با این دستور می‌توانید فلش را فعال یا غیر فعال کنید.ورودی دستور نام شی فلش مورد نظر و همچنین عبارت بولین برای فعال یا غیرفعال کردن آن می‌باشد.

مقدار بازگشتی **ندارد** .

Flash.SetFlashVariable: با این دستور می‌توانید مقدار متغیر داخل فلش را تغییر دهید .نام شی، نام متغیر و مقدار جدید ورودی‌های دستور می‌باشند.

مقدار بازگشتی **ندارد** .

Flash.SetPos: با این کد می‌توانید شی فلش را به محل دلخواه در صفحه منتقل نمایید.ورودی‌های این دستور نام شی، عددی برای جای گیری در قسمت X (فاصله از چپ) و عدد دیگری برای جای گیری در Y (فاصله از بالا می‌باشد).

مقدار بازگشتی **ندارد** .

Flash.SetProperties: این دستور هم با دریافت نام شی فلش و آرایه خصوصیات، خصوصیات شی فلش را تغییر می‌دهد. (در قسمت Flash.GetProperties این آرایه توضیح داده شده است.)

مقدار بازگشتی ندارد.

Flash.SetReturnValue: با این دستور می‌توانید نتیجه بازگشتی فلش را تغییر دهید. نام شی و رشته با فرمت XML ورودی‌های تابع اند.

مقدار بازگشتی **ندارد** .

Flash.SetSize: با این دستور می‌توانید اندازه شی فلش را به اندازه دلخواه خود تغییر دهید.ورودی‌های این کد نام شی، عددی برای عرض و عددی دیگر برای طول شی می‌باشد.

مقدار بازگشتی **ندارد** .

Flash.SetVisible: کنترل نمایش یا عدم نمایش فلش در نرم افزار با این دستور امکان پذیر است. ورودی‌های دستور، نام شی و عبارتی بولین برای فعال یا غیر فعال کردن نمایش فلش می‌باشد.

مقدار بازگشتی **ندارد** .

Flash.Stop: با این دستور می‌توانید نمایش فلش را متوقف نمایید. نام شی فلش ورودی دستور است.

مقدار بازگشتی **ندارد** .

Folder

این تابع برای کار با پوشه‌ها مورد استفاده قرار می‌گیرد و دستورات زیر مجموعه‌ی آن به شرح زیر است:

Folder.Create: با استفاده از این دستور می‌توانید شاخه جدید ایجاد نمایید .مسیر شاخه جدید ورودی دستور می‌باشد.

مقدار بازگشتی **ندارد** .

Folder.Delete: با این دستور قادر به حذف کردن شاخه خواهید بود .دقت داشته باشید که برای حذف شاخه، باید محتویات داخل آن قبلاً حذف شده باشد .برای این کار می‌توانید از File.Delete استفاده نمایید.

مقدار بازگشتی **ندارد** .

Folder.DeleteTree: با این دستور می‌توانید یک شاخه با تمام محتویات داخل آن (فایل‌ها و زیرشاخه‌ها) حذف نمایید. مسیر شاخه و نام تابع بازگشتی ورودی‌های دستورند.

مقدار بازگشتی **ندارد** .

Folder.DoesExist: برای کنترل وجود یا عدم وجود یک شاخه می‌توانید از این کد استفاده نمایید.مسیر شاخه ورودی دستور است.

مقدار بازگشتی متغیر بولین می‌باشد که True نشان دهنده وجود شاخه و False نشان دهنده عدم وجود آن است .

Folder.Find: با این دستور می‌توانید با ورود نام شاخه، به جستجوی آن شاخه بپردازید.ورودی‌های دستور، مسیر برای جستجو در آن، نام شاخه مورد نیاز، مقدار بولین برای فعال یا غیرفعال کردن جستجو در زیرشاخه‌ها و نام تابع بازگشتی است.

مقدار بازگشتی آرایه ای است به صورت رشته که مسیر تمام شاخه های یافت شده با نام دلخواهتان در آن قرار دارد.

Folder.GetCurrent: شاخه ای که برنامه از روی آن اجرا شده است با این دستور به دست می‌آید.

مقدار بازگشتی، مسیر به صورت رشته است.

Folder.Rename: با استفاده از این دستور می‌توانید نام یک شاخه را تغییر دهید .فقط توجه داشته باشد برنامه ای از روی تابع نباید در حال اجرا باشد.مسیر شاخه برای تغییر نام و مسیر و نام جدید، 2 ورودی دستور هستند.

مقدار بازگشتی ندارد.

Folder.SetCurrent: این دستور با دریافت مسیر به عنوان ورودی، شاخه اصلی برنامه را تغییر می‌دهد.

مقدار بازگشتی **ندارد** .

Grid

این تابع برای کار با شی Grid به کار می‌رود که دستورات زیر مجموعه‌ی آن به شرح زیر است:

Grid.AutoSize: از این دستور برای تنظیم خودکار اندازه ردیف‌ها و ستون‌های شی Grid استفاده می‌کنیم. این دستور دارای سه پارامتر ورودی می‌باشد:

ObjectName: نام شی Grid

ResizeMode: مدل تغییر اندازه دادن که شامل 4 مدل می‌باشد:

مدل‌ها	مقدار	توضیحات
GVS_DEFAULT	0	استفاده از تغییر اندازه‌ی پیش‌فرض
GVS_HEADER	1	تغییر اندازه تمام ردیف‌ها و ستون‌ها با توجه به اندازه بزرگ‌ترین متن در هر سلول از ردیف یا ستون
GVS_DATA	2	تغییر اندازه تمام ردیف‌ها و ستون‌ها با توجه به اندازه بزرگ‌ترین متن در هر سلول از ردیف یا ستون
GVS_BOTH	3	ترکیبی از تغییر اندازه با توجه به استاندارد GVS_HEADER و GVS_DATA

Redraw: این خاصیت دارای مقدار بولین می‌باشد و اگر مقدار برابر با true باشد اقدام به رسم مجدد شی Grid و محتویات آن می‌نماید و اگر مقدارش false باشد شی Grid از نوع رسم نمی‌شود.

Grid.AutoSizeColumn: از این دستور برای تنظیم خودکار اندازه‌ی یک ستون خاص از شی Grid استفاده می‌کنیم. این دستور دارای پنج پارامتر ورودی می‌باشد:

ObjectName: نام شی Grid

Column: شماره‌ی ستونی که می‌خواهیم اندازه‌اش را تنظیم کنیم.

ResizeMode: مدل تغییر اندازه دادن که شامل 4 مدل می‌باشد:

مدل‌ها	مقدار	توضیحات
GVS_DEFAULT	0	استفاده از تغییر اندازه‌ی پیش‌فرض
GVS_HEADER	1	تغییر اندازه ستون با توجه به اندازه متن موجود در سربرگ ستون
GVS_DATA	2	تغیر اندازه ستون با توجه بزرگ‌ترین متن موجود در ستون به غیر از متن سربرگ
GVS_BOTH	3	ترکیبی از تغییر اندازه با توجه به استاندارد GVS_HEADER و GVS_DATA

ResetScroll: این خاصیت موجب رسم مجدد میله لغزنده در شی Grid می‌گردد که اگر مقدار آن برابر با true باشد این عمل انجام خواهد شد در غیر این صورت (false) عمل مورد نظر انجام نخواهد شد.

Redraw: ر ک Grid.AutoSize

Grid.AutoSizeColumns: از این دستور برای تنظیم خودکار اندازه‌ی ستون‌های شی Grid استفاده می‌کنیم. این دستور دارای سه پارامتر ورودی می‌باشد:

ObjectName: نام شی Grid

ResizeMode: مدل تغییر اندازه دادن که شامل 4 مدل می‌باشد:

مدل‌ها	مقدار	توضیحات
GVS_DEFAULT	0	استفاده از تغییر اندازه‌ی پیش‌فرض
GVS_HEADER	1	تغییر اندازه ستون‌ها با توجه به اندازه متن موجود در سربرگ ستون
GVS_DATA	2	تغیر اندازه ستون‌ها توجه بزرگ‌ترین متن موجود در ستون به غیر از متن سربرگ
GVS_BOTH	3	ترکیبی از تغییر اندازه با توجه به استاندارد GVS_HEADER و GVS_DATA

Redraw: ر ک Grid.AutoSize

Grid.AutoSizeRow: از این دستور برای تنظیم خودکار اندازه‌ی یک ردیف خاص از شی Grid استفاده می‌کنیم. این دستور دارای سه پارامتر ورودی می‌باشد:

ObjectName: نام شی Grid

Row: شماره‌ی ردیفی که می‌خواهیم اندازه‌اش را تنظیم کنیم؛

ResetScroll: این خاصیت موجب رسم مجدد میله لغزنده در شی Grid می‌گردد که اگر مقدار آن برابر با true باشد این عمل انجام خواهد شد در غیر این صورت (false) عمل مورد نظر انجام نخواهد شد؛

Redraw: ر ک Grid.AutoSize

Grid.AutoSizeRows: از این دستور برای تنظیم خودکار اندازه‌ی ردیف‌های شی Grid استفاده می‌کنیم. این دستور دارای دو پارامتر ورودی می‌باشد:

ObjectName: نام شی Grid

Redraw: ر ک Grid.AutoSize.

Grid.DeleteAllItems: از این دستور برای حذف تمام گزینه های موجود در شی Grid استفاده می‌کنیم. این دستور دارای یک پارامتر ورودی می‌باشد:

ObjectName: نام شی Grid.

Grid.DeleteColumn: از این دستور برای حذف یک ستون خاص از شی Grid استفاده می‌کنیم. این دستور دارای سه پارامتر ورودی می‌باشد:ObjectName

: نام شی Grid، Column: شماره‌ی ستون

Redraw: ر ک Grid.AutoSize.

Grid.DeleteNonFixedRows: از این دستور برای حذف کردن ردیف‌هایی به کار می‌رود که در حالت عادی قرار دارند یعنی حالت Fix ندارند. حالت Fix حالتی است که در آن نمی‌توان روی ردیف یا ستون مورد نظر فعالیتی را انجام داد و فقط قابل دیدن می‌باشد. این دستور دارای دو پارامتر ورودی می‌باشد:

ObjectName: نام شی Grid

Redraw: ر ک Grid.AutoSize.

Grid.DeleteRow: از این دستور برای حذف یک ردیف خاص از شی Grid استفاده می‌کنیم. این دستور دارای سه پارامتر ورودی می‌باشد:

ObjectName: نام شی Grid

Row: شماره‌ی سطر

Redraw: ر ک Grid.AutoSize.

Grid.EditCopy: از این دستور برای رونوشت (کپی) متن‌های موجود در سلول‌های انتخاب شده استفاده می‌کنیم. این دستور دارای یک پارامتر ورودی می‌باشد:

ObjectName: نام شی Grid.

Grid.EditCut: از این دستور برای برش (Cut) متن‌های موجود در سلول‌های انتخاب شده استفاده می‌کنیم. این دستور دارای یک پارامتر ورودی می‌باشد:

ObjectName: نام شی Grid.

Grid.EditPaste: از این دستور برای جایگذاری (Paste) متن‌های موجود از حافظه‌ی Clipboard به شی Grid استفاده می‌کنیم. این دستور دارای یک پارامتر ورودی می‌باشد:

ObjectName: نام شی Grid.

Grid.EnsureVisible: تضمین می‌کند که سلول‌های مشخص شده در یک شی Grid قابل رویت است. و در صورت لزوم سلول مشخص شده را با حرکت میله لغزان(Scroll Bar) ظاهر می‌سازد. این دستور دارای سه پارامتر ورودی می‌باشد:

ObjectName: نام شی Grid

Row: شماره‌ی سطر

Column: شماره‌ی ستون.

Grid.ExpandColumnsToFit: پهن کننده‌ی تمام ستون‌های موجود در شی Grid متناسب با اندازه‌ی شی. این دستور دارای سه پارامتر ورودی می‌باشد:

ObjectName: نام شی Grid

ExpandFixed: تعیین می‌کند که آیا ستون‌های Fix شده نیز پهن شوند یا نه؟ و دارای دو مقدار true(پهن شوند) و false(پهن نشوند) می‌باشد

Redraw: ر ک Grid.AutoSize.

Grid.ExpandLastColumn: پهن کننده‌ی آخرین ستون موجود در شی Grid متناسب با اندازه‌ی شی. این دستور دارای دو پارامتر ورودی می‌باشد:

ObjectName: نام شی Grid

Redraw: ر ک Grid.AutoSize.

Grid.ExpandRowsToFit: بسط دهنده‌ی تمام ردیف‌های موجود در شی Grid متناسب با اندازه‌ی شی. این دستور دارای سه پارامتر ورودی می‌باشد:

ObjectName: نام شی Grid

ExpandFixed: تعیین می‌کند که آیا ردیف‌های Fix شده نیز منبسط شوند یا نه؟ و دارای دو مقدار true(منبسط شوند) و false(منبسط نشوند) می‌باشد

Redraw: ر ک Grid.AutoSize.

Grid.ExpandToFit: پهن کننده‌ی تمام ردیف‌ها و ستون‌های موجود در شی Grid متناسب با اندازه‌ی شی. این دستور دارای سه پارامتر ورودی می‌باشد:

ObjectName: نام شی Grid

ExpandFixed: تعیین می‌کند که آیا ستون‌های Fix شده نیز پهن شوند یا نه؟ و دارای دو مقدار true(پهن شوند) و false(پهن نشوند) می‌باشد.

Redraw: ر ک Grid.AutoSize.

Grid.GetCellColors: از این دستور برای گرفتن رنگ پس زمینه و رنگ متن سلول مشخص شده در شی Grid استفاده می‌کنیم. این دستور دارای سه پارامتر ورودی می‌باشد:

ObjectName: نام شی Grid

Row : شماره‌ی سطر

Column: شماره‌ی ستون.

مقدار بازگشتی این دستور یک متغیر آرایه ای می‌باشد:

Background: شماره رنگ زمینه‌ی سلول را بر می‌گرداند.

Text: شماره‌ی رنگ متن سلول را بر می‌گرداند.

Grid.GetCellState: از این دستور برای گرفتن خصوصیات سلول مشخص شده در شی Grid استفاده می‌کنیم. این دستور دارای سه پارامتر ورودی می‌باشد:

ObjectName: نام شی Grid

Row : شماره‌ی سطر

Column: شماره‌ی ستون.

مقدار بازگشتی این دستور یک متغیر آرایه ای می‌باشد که صورت زیر می‌باشد:

نام ویژگی	نوع	توضیحات
Focused	boolean	اگر سلول کانون توجه برنامه نیز باشد true برگشت داده می‌شود و در غیر این صورت false برگردانده خواهد شد.
Selected	boolean	اگر سلول انتخاب شده باشد true برگشت داده می‌شود و در غیر این صورت false برگردانده خواهد شد.
DropHighlighted	boolean	اگر سلول برجسته (های لایت) باشد true برگشت داده می‌شود و در غیر این صورت false برگردانده خواهد شد.
ReadOnly	boolean	اگر سلول دارای خاصیت فقط خواندنی باشد true برگشت داده می‌شود و در غیر این صورت false برگردانده خواهد شد.
Fixed	boolean	اگر سلول دارای خاصیت Fix باشد true برگشت داده می‌شود و در غیر این صورت false برگردانده خواهد شد.
FixedRow	boolean	اگر سلول قسمتی از یک ردیف Fix شده باشد true برگشت داده می‌شود و در غیر این صورت false برگردانده خواهد شد.
FixedCol	boolean	اگر سلول قسمتی از یک ستون Fix شده باشد true برگشت داده می‌شود و در غیر این صورت false برگردانده خواهد شد.
Modified	boolean	اگر سلول اصلاح شده باشد true برگشت داده می‌شود و در غیر این صورت false برگردانده خواهد شد.

Grid.GetCellText: از این دستور برای گرفتن متن سلول مشخص شده در شی Grid استفاده می‌کنیم. این دستور دارای سه پارامتر ورودی می‌باشد:

ObjectName: نام شی Grid

Row : شماره‌ی سطر

Column: شماره‌ی ستون.

مقدار بازگشتی این دستور یک متغیر رشته ای می‌باشد.

Grid.GetColumnCount: از این دستور برای گرفتن تعداد ستون‌های شی Grid استفاده می‌کنیم. این دستور دارای یک پارامتر ورودی می‌باشد:

ObjectName: نام شی Grid.

مقدار بازگشتی این دستور یک متغیر عددی می‌باشد.

Grid.GetColumnHiding: از این دستور برای به دست آوردن این که آیا ستون‌های شی Gird مورد نظر قابل مخفی کردن توسط کاربر می‌باشند یا نه؟ استفاده می‌کنیم. این دستور دارای یک پارامتر ورودی می‌باشد:

ObjectName: نام شی Grid.

مقدار بازگشتی این دستور یک متغیر بولین می‌باشد true یعنی ستون‌های شی Gird مورد نظر قابل مخفی کردن توسط کاربر می‌باشند و false یعنی ستون‌ها قابل مخفی کردن نیستند.

Grid.GetColumnResize: از این دستور برای به دست آوردن این که آیا ستون‌های شی Gird مورد نظر قابل تغییر اندازه توسط کاربر می‌باشند یا نه؟ استفاده می‌کنیم. این دستور دارای یک پارامتر ورودی می‌باشد:

ObjectName: نام شی Grid.

مقدار بازگشتی این دستور یک متغیر بولین می‌باشد true یعنی ستون‌های شی Gird مورد نظر تغییر اندازه توسط کاربر می‌باشند و false یعنی ستون‌ها غیر قابل تغییر اندازه می‌باشند.

Grid.GetColumnWidth: از این دستور برای گرفتن پهنای ستون مشخص شده شی Grid استفاده می‌کنیم. این دستور دارای دو پارامتر ورودی می‌باشد:

ObjectName: نام شی Grid

Column: شماره‌ی ستون مورد نظر.

مقدار بازگشتی این دستور یک متغیر عددی می‌باشد.

Grid.GetDragAndDrop: از این دستور برای به دست آوردن این که آیا سلول‌های شی Gird مورد نظر قابلیت کشیدن و رها کردن توسط ماوس را دارا می‌باشند یا نه؟ استفاده می‌کنیم. این دستور دارای یک پارامتر ورودی می‌باشد:

ObjectName: نام شی Grid.

مقدار بازگشتی این دستور یک متغیر بولین می‌باشد true یعنی سلول‌های شی Gird مورد نظر قابل کشیدن و رها کردن توسط ماوس می‌باشند و false یعنی سلول‌ها این قابلیت را ندارند.

Grid.GetEditable: از این دستور برای به دست آوردن این که آیا شی Gird مورد نظر قابلیت ویرایش توسط کاربر را دارا می‌باشند یا نه؟ استفاده می‌کنیم. این دستور دارای یک پارامتر ورودی می‌باشد:

ObjectName: نام شی Grid.

مقدار بازگشتی این دستور یک متغیر بولین می‌باشد. true یعنی شی Gird مورد نظر قابل ویرایش می‌باشد و false یعنی این قابلیت را ندارد.

Grid.GetFixedColumnCount: از این دستور برای به دست آوردن تعداد ستون‌هایی که حالت Fix دارند استفاده می‌کنیم. این دستور دارای یک پارامتر ورودی می‌باشد:

ObjectName: نام شی Grid.

مقدار بازگشتی این دستور یک متغیر عددی می‌باشد.

Grid.GetFixedColumnSelection: از این دستور برای به دست آوردن این که آیا سلول‌های زیرین ستون Fix شده با کلیک روی آن انتخاب می‌شوند یا نه ؟ استفاده می‌کنیم. این دستور دارای یک پارامتر ورودی می‌باشد:

ObjectName: نام شی Grid.

مقدار بازگشتی این دستور یک متغیر بولین می‌باشد. true یعنی شی Gird مورد نظر این قابلیت را دارا می‌باشد و false یعنی این قابلیت را ندارد.

Grid.GetFixedRowCount: از این دستور برای به دست آوردن تعداد ردیف‌هایی که حالت Fix دارند استفاده می‌کنیم. این دستور دارای یک پارامتر ورودی می‌باشد:

ObjectName: نام شی Grid.

مقدار بازگشتی این دستور یک متغیر عددی می‌باشد.

Grid.GetFixedRowSelection: از این دستور برای به دست آوردن این که آیا سلول‌های کنار سلول Fix شده با کلیک روی آن انتخاب می‌شوند یا نه ؟ استفاده می‌کنیم. این دستور دارای یک پارامتر ورودی می‌باشد:

ObjectName: نام شی Grid.

مقدار بازگشتی این دستور یک متغیر بولین می‌باشد. true یعنی شی Gird مورد نظر این قابلیت را دارا می‌باشد و false یعنی این قابلیت را ندارد.

Grid.GetFocusCell:: از این دستور برای به دست آوردن شماره ردیف و ستون سلولی که در حالت مرکز توجه برنامه قرار دارد استفاده می‌کنیم. این دستور دارای یک پارامتر ورودی می‌باشد:

ObjectName: نام شی Grid.

مقدار بازگشتی این دستور یک متغیر آرایه ای می‌باشد:

نام ویژگی	نوع	توضیح
Row	number	شماره ردیف سلول را بر می‌گرداند
Column	number	شماره ستون سلول را بر می‌گرداند

مثال:

```
get_focus_cell = Grid.GetFocusCell("Grid1");
Dialog.Message("Notice", get_focus_cell.Row);
```

Grid.GetFrameFocusCell: از این دستور برای به دست آوردن این که آیا سلول کانون توجه برنامه با یک کادر کناری برجسته شده است یا نه؟ استفاده می‌کنیم. این دستور دارای یک پارامتر ورودی می‌باشد:

ObjectName: نام شی Grid.

مقدار بازگشتی این دستور یک متغیر بولین می‌باشد. true یعنی شی Gird مورد نظر این قابلیت را دارا می‌باشد و false یعنی این قابلیت را ندارد.

Grid.GetGridColors: از این دستور برای گرفتن رنگ پس زمینه شی Grid، رنگ خطها، رنگ پس زمینه‌ی متن توضیحی و رنگ متن توضیحی استفاده می‌کنیم. این دستور دارای یک پارامتر ورودی می‌باشد:

ObjectName: نام شی Grid.

مقدار بازگشتی این دستور به صورت آرایه ای می‌باشد که در زیر آمده است:

نام ویژگی	نوع	توضیحات
GridBackground	number	رنگ پس زمینه شی Grid را بر می‌گرداند.
GridLines	number	رنگ خط‌های شی Grid را بر می‌گرداند.
TooltipBackground	number	رنگ پس زمینه‌ی متن توضیحی شی Grid را بر می‌گرداند.
TooltipText	number	رنگ متن توضیحی شی Grid را بر می‌گرداند.

مثال:

```
tblGridColors = Grid.GetGridColors("Grid1");
Dialog.Message("Notice", tblGridColors.GridBackground);
```

Grid.GetGridLines: از این دستور برای گرفتن نوع خط‌های شی Grid (افقی، عمودی یا هر دو)، استفاده می‌کنیم. این دستور دارای یک پارامتر ورودی می‌باشد:

ObjectName: نام شی Grid.

مقدار بازگشتی این دستور به صورت عددی می‌باشد که در زیر آمده است:

نام ثابت	مقدار	توضیح
GVL_NONE	0	بدون سطر
GVL_HORZ	1	سطر افقی
GVL_VERT	2	عمودی
GVL_BOTH	3	هر دو

Grid.GetHeaderSort: از این دستور برای به دست آوردن این که سلول‌ها با کلیک روی سربرگ ستون مرتب می‌شوند یا نه؟ استفاده می‌کنیم. این دستور دارای یک پارامتر ورودی می‌باشد:

ObjectName: نام شی Grid.

مقدار بازگشتی این دستور یک متغیر بولین می‌باشد. true یعنی شی Gird مورد نظر این قابلیت را دارا می‌باشد و false یعنی این قابلیت را ندارد.

Grid.GetModified: از این دستور برای به دست آوردن این که آیا سلول مشخص اصلاح شده است یا نه؟ استفاده می‌کنیم. این دستور دارای سه پارامتر ورودی می‌باشد:

ObjectName: نام شی Grid

Row: شماره سطر یا ردیف

Column: شماره‌ی ستون.

مقدار بازگشتی این دستور یک متغیر بولین می‌باشد. true یعنی شی Gird مورد نظر این قابلیت را دارا می‌باشد و false یعنی این قابلیت را ندارد.

Grid.GetNextItem: از این دستور برای جستجوی شماره ردیف و ستون سلول مورد نظر با توجه به ویژگی‌های مشخص شده استفاده می‌کنیم. این دستور دارای پنج پارامتر ورودی می‌باشد:

ObjectName: نام شی Grid

Row: شماره‌ی سطری که جستجو از آن آغاز می‌شود.

Column: شماره‌ی ستونی که جستجو از آن آغاز می‌شود.

SearchType : نوع جستجو را تعیین می‌کند که انواع آن در جدول زیر آمده است:

نام ثابت	مقدار	توضیحات
GVNI_FOCUSED	1	جستجو برای سلول‌های کانون توجه واقع شده
GVNI_SELECTED	2	جستجو برای سلول‌های انتخاب شده
GVNI_DROPHILITED	4	جستجو برای سلول‌های برجسته شده
GVNI_READONLY	8	جستجو برای سلول‌های فقط خواندنی
GVNI_FIXED	16	جستجو برای سلول‌های Fix شده
GVNI_MODIFIED	32	جستجو برای سلول‌های اصلاح شده

SearchDirection: محل شروع جستجو را مشخص می‌کند که به ترتیب زیر است:

نام ثابت	مقدار	توضیحات
GVNI_ABOVE	256	جستجو در بالای سلول آغازگر جستجو
GVNI_BELOW	512	جستجو در پایین سلول آغازگر جستجو
GVNI_TOLEFT	1024	جستجوی تمام سلول به سمت چپ سلول آغازگر جستجو

جستجوی تمام سلول به سمت راست سلول آغازگر جستجو	2048	GVNI_TORIGHT
جستجوی تمام سلول‌ها از سلول آغازگر جستجو	3584	GVNI_ALL
جستجوی تمام سلول‌ها از پایین و به راست سلول آغازگر جستجو	2560	GVNI_AREA

مقدار بازگشتی این دستور به صورت آرایه ای می‌باشد:

Row: شماره ردیف سلول یافت شده

Column: شماره‌ی ستون سلول یافت شده

مثال:

```
tblresult = Grid.GetNextItem("Grid1", 1, 1, GVNI_SELECTED,
GVNI_ALL);
if tblresult == nil then
    Dialog.Message("Warning", "No Cells have focus");
else
    Dialog.Message("Focus Found", "Focus Cell =
"..tblresult.Column.."/"..tblresult.Row);
end
```

Grid.GetPos: از این دستور برای به دست آوردن مختصات شی Grid بر روی صفحه استفاده می‌کنیم. این دستور دارای یک پارامتر ورودی می‌باشد:

ObjectName: نام شی Grid.

مقدار بازگشتی این دستور یک متغیر آرایه ای می‌باشد. X: مختصات افقی شی و Y: مختصات عمودی شی.

Grid.GetProperties: از این دستور برای به دست آوردن ویژگی‌های شی Grid استفاده می‌کنیم. این دستور دارای یک پارامتر ورودی می‌باشد:

ObjectName: نام شی Grid.

مقدار بازگشتی این دستور به صورت آرایه ای می‌باشد که در جدول زیر آمده است:

ویژگی‌ها	نوع	توضیحات
ObjectName	string	نام شی
Rows	number	تعداد سطرها یا ردیف‌ها
Columns	number	تعداد ستون‌ها

FixedRows	number	تعداد سطرهای ثابت (Fix)
FixedColumns	number	تعداد ستون‌ها ثابت (Fix)
TextColor	number	رنگ متن
TextBackgroundColor	number	رنگ پس زمینه‌ی متن
FixedTextColor	number	رنگ متن ثابت
FixedBackgroundColor	number	رنگ پس زمینه‌ی متن ثابت
GridLineColor	number	رنگ خط‌های جدا کننده
GridBackgroundColor	number	رنگ پس زمینه‌ی شی Grid
TitleTipBackgroundColor	number	رنگ پس زمینه‌ی متن توضیحی
TitleTipTextColor	number	رنگ متن توضیحی
Editable	boolean	true: قابل ویرایش، false: غیر قابل ویرایش
Selectable	boolean	true: قابل انتخاب false: غیر قابل انتخاب
CellDragAndDrop	boolean	true: قابل کشیدن و رها کردن false: غیر قابل کشیدن و رها کردن
ResizableRows	boolean	true: قابل تغییر اندازه بودن سطرها false: غیر قابل تغییر اندازه
ResizableColumns	boolean	true: قابل تغییر اندازه بودن ستون‌ها false: غیر قابل تغییر اندازه
GridLines	number	یک متغیر عددی که نوع سطر را بر می‌گرداند

نام ثابت	مقدار	توضیح
GVL_NONE	0	بدون سطر
GVL_HORZ	1	سطر افقی
GVL_VERT	2	عمودی
GVL_BOTH	3	هر دو

FontName	string	نام فونت یا قلم
FontSize	number	اندازه‌ی قلم

<table>
<tr><td>FontStrikeout</td><td>boolean</td><td>true: فعال بودن ویژگی FontStrikeout، false: غیر فعال بودن آن</td></tr>
<tr><td>FontUnderline</td><td>boolean</td><td>true: فعال بودن ویژگی خط زیرین برای فونت، false: غیر فعال بودن آن</td></tr>
<tr><td>FontAntiAlias</td><td>boolean</td><td>true: فعال بودن ویژگی AntiAlias برای فونت، false: غیر فعال بودن آن</td></tr>
<tr><td>FontItalic</td><td>boolean</td><td>true: فعال بودن ویژگی خمیدگی برای فونت، false: غیر فعال بودن آن</td></tr>
<tr><td>FontWeight</td><td>number</td><td>نوع نوشته (تیرگی متن)
<table>
<tr><th>نام ثابت</th><th>مقدار</th><th>توضیح</th></tr>
<tr><td>FW_DONTCARE</td><td>0</td><td>بدون تغییر</td></tr>
<tr><td>FW_THIN</td><td>100</td><td>نازک</td></tr>
<tr><td>FW_EXTRALIGHT</td><td>200</td><td>نور بیشتر</td></tr>
<tr><td>FW_LIGHT</td><td>300</td><td>روشن</td></tr>
<tr><td>FW_NORMAL</td><td>400</td><td>معمولی</td></tr>
<tr><td>FW_MEDIUM</td><td>500</td><td>متوسط</td></tr>
<tr><td>FW_SEMIBOLD</td><td>600</td><td>نیمه پر</td></tr>
<tr><td>FW_BOLD</td><td>700</td><td>توپر</td></tr>
<tr><td>FW_EXTRABOLD</td><td>800</td><td>توپر اضافی</td></tr>
<tr><td>FW_HEAVY</td><td>900</td><td>سنگین</td></tr>
</table></td></tr>
<tr><td>FontScript</td><td>number</td><td>اسکریپت قلم برای نمایش نوشته‌ها
<table>
<tr><th>نام ثابت</th><th>مقدار</th><th>توضیح</th></tr>
<tr><td>ANSI_CHARSET</td><td>0</td><td>ANSI</td></tr>
<tr><td>BALTIC_CHARSET</td><td>186</td><td>Baltic.</td></tr>
<tr><td>CHINESEBIG5_CHARSET</td><td>136</td><td>Chinese</td></tr>
</table></td></tr>
</table>

		DEFAULT_CHARSET: 1 (Default) EASTEUROPE_CHARSET: 238 (Eastern European) GB2312_CHARSET: 134 (GB2312) GREEK_CHARSET: 161 (Greek) HANGUL_CHARSET: 129 (Hangul) MAC_CHARSET: 77 (MAC) OEM_CHARSET: 255 (OEM) RUSSIAN_CHARSET: 204 (Russian.) SHIFTJIS_CHARSET: 128 (Shiftjis) SYMBOL_CHARSET: 2 (Symbol) TURKISH_CHARSET: 162 (Turkish) -- : 178 (عربی و فارسی)
Enabled	boolean	true: فعال بودن شی، false: غیر فعال بودن آن
Visible	boolean	true: قابل مشاهده بودن شی، false: غیر قابل مشاهده بودن آن
X	number	مختصات افقی شی Grid
Y	number	مختصات عمودی شی Grid
Width	number	پهنای شی Grid
Height	number	درازای شی Grid
TooltipText	string	متن توضیحی در شی Grid که با بردن نشانگر ماوس روی آن فعال می‌گردد.
ResizeLeft	boolean	true: تغییر اندازه‌ی شی به سمت چپ زمانی که پروژه

تغییر اندازه می‌یابد، false: اندازه‌ی شی تغییر نمی‌کند.		
true: تغییر اندازه‌ی شی به سمت راست زمانی که پروژه تغییر اندازه می‌یابد، false: اندازه‌ی شی تغییر نمی‌کند.	boolean	ResizeRight
true: تغییر اندازه‌ی شی به سمت بالا زمانی که پروژه تغییر اندازه می‌یابد، false: اندازه‌ی شی تغییر نمی‌کند.	boolean	ResizeTop
true: تغییر اندازه‌ی شی به سمت پایین زمانی که پروژه تغییر اندازه می‌یابد، false: اندازه‌ی شی تغییر نمی‌کند.	boolean	ResizeBottom
یک عدد صحیح که نمایش دهنده هندل پنجره شی Grid می‌باشد.	number	WindowHandle

مثال:

```
tProperties = Grid.GetProperties("Grid1");
a = tProperties .X;
```

Grid.GetRowCount: با استفاده از این دستور می‌توانیم تعداد سطرها یا ردیف‌های شی Grid را به دست آوریم. این دستور دارای یک پارامتر ورودی می‌باشد:ObjectName: نام شی Grid.

مقدار بازگشتی این دستور به صورت عددی می‌باشد.

Grid.GetRowHeight: با استفاده از این دستور می‌توانیم درازای سطری را از شی Grid را به دست آوریم. این دستور دارای دو پارامتر ورودی می‌باشد:

ObjectName: نام شی Grid

Row: شماره‌ی سطر مورد نظر.

مقدار بازگشتی این دستور به صورت عددی می‌باشد.

Grid.GetRowHiding: از این دستور برای به دست آوردن این که آیا سطرهای شی Gird مورد نظر قابل مخفی کردن توسط کاربر می‌باشند یا نه؟ استفاده می‌کنیم. این دستور دارای یک پارامتر ورودی می‌باشد:

ObjectName: نام شی Grid.

مقدار بازگشتی این دستور یک متغیر بولین می‌باشد true یعنی سطر های شی Gird مورد نظر قابل مخفی کردن توسط کاربر می‌باشند و false یعنی قابل مخفی کردن نیستند.

Grid.GetRowResize: از این دستور برای به دست آوردن این که آیا سطر های شی Gird مورد نظر قابل تغییر اندازه توسط کاربر می‌باشند یا نه؟ استفاده می‌کنیم. این دستور دارای یک پارامتر ورودی می‌باشد:

ObjectName: نام شی Grid.

مقدار بازگشتی این دستور یک متغیر بولین می‌باشد true یعنی سطر های شی Gird مورد نظر تغییر اندازه توسط کاربر می‌باشند و false یعنی سطرها غیر قابل تغییر اندازه می‌باشند.

Grid.GetSelectable: از این دستور برای به دست آوردن این که آیا سلول‌های شی Gird مورد نظر قابل انتخاب کردن توسط می‌باشند یا نه؟ استفاده می‌کنیم. این دستور دارای یک پارامتر ورودی می‌باشد:

ObjectName: نام شی Grid.

مقدار بازگشتی این دستور یک متغیر بولین می‌باشد true یعنی سلول‌های شی Gird مورد نظر قابل انتخاب کردن توسط کاربر می‌باشند و false یعنی سلول‌ها غیر قابل انتخاب کردن می‌باشند.

Grid.GetSelectedCount: با استفاده از این دستور می‌توانیم تعداد گزینه های انتخاب شده را از شی Grid را به دست آوریم. این دستور دارای یک پارامتر ورودی می‌باشد:

ObjectName: نام شی Grid.

مقدار بازگشتی این دستور به صورت عددی می‌باشد.

Grid.GetSingleColumnSelection: از این دستور برای به دست آوردن این که آیا ستون‌های شی Gird مورد نظر به صورت تکی قابل انتخاب می‌باشند یا نه؟ استفاده می‌کنیم. این دستور دارای یک پارامتر ورودی می‌باشد:

ObjectName: نام شی Grid.

مقدار بازگشتی این دستور یک متغیر بولین می‌باشد true یعنی ستون‌های شی Gird مورد نظر به صورت تکی قابل انتخاب می‌باشند و false یعنی ستون‌ها به صورت تکی قابل انتخاب نمی‌باشند.

Grid.GetSingleRowSelection: از این دستور برای به دست آوردن این که آیا سطرهای شی Gird مورد نظر به صورت تکی قابل انتخاب می‌باشند یا نه؟ استفاده می‌کنیم. این دستور دارای یک پارامتر ورودی می‌باشد:

ObjectName: نام شی Grid.

مقدار بازگشتی این دستور یک متغیر بولین می‌باشد true یعنی سطرهای شی Gird مورد نظر به صورت تکی قابل انتخاب می‌باشند و false یعنی سطرها به صورت تکی قابل انتخاب نمی‌باشند.

Grid.GetSize: با استفاده از این دستور می‌توانیم اندازه‌ی شی Grid را به دست آوریم. این دستور دارای یک پارامتر ورودی می‌باشد:

ObjectName: نام شی Grid.

مقدار بازگشتی این دستور به صورت آرایه می‌باشد.Width : پهنای شی را به صورت عددی بر می‌گرداند. Height: درازای شی را به صورت عددی بر می‌گرداند.

مثال:

```
scores_size = Grid.GetSize("Grid1");
width = scores_size. Width
```

Grid.GetSortAscending: از این دستور برای به دست آوردن این که آیا ستون مرتب شده از شی Gird مورد نظر به صورت صعودی می‌باشد یا نه؟ استفاده می‌کنیم. این دستور دارای یک پارامتر ورودی می‌باشد:

ObjectName: نام شی Grid.

مقدار بازگشتی این دستور یک متغیر بولین می‌باشد true یعنی ستون مرتب شده از شی Gird مورد نظر به صورت صعودی می‌باشد و false یعنی ستون مرتب شده از شی Gird مورد نظر به صورت صعودی نمی‌باشد.

Grid.GetSortColumn: با استفاده از این دستور می‌توانیم شماره‌ی ستون مرتب شده‌ی کنونی را به دست آوریم. این دستور دارای یک پارامتر ورودی می‌باشد

:ObjectName: نام شی Grid.

مقدار بازگشتی این دستور به صورت عددی می‌باشد.

Grid.GetTabEnabled: با استفاده از این دستور می‌توان تشخیص داد که آیا کلید Tab برای پرش در بین سلول‌ها فعال می‌باشد یا نه؟ این دستور دارای یک پارامتر ورودی می‌باشد:

ObjectName: نام شی Grid.

مقدار بازگشتی این دستور یک متغیر بولین می‌باشد true یعنی کلید Tab فعال است و false یعنی کلید Tab فعال نیست.

Grid.GetToolTipsEnabled: با استفاده از این دستور می‌توان تشخیص داد که آیا قسمت متن توضیحی برای سلول‌های شی Grid فعال می‌باشد یا نه؟ این دستور دارای یک پارامتر ورودی می‌باشد:

ObjectName: نام شی Grid.

مقدار بازگشتی این دستور یک متغیر بولین می‌باشد true یعنی متن توضیحی برای سلول‌های شی Grid فعال می‌باشد و false یعنی متن توضیحی فعال نیست.

Grid.GetTrackFocusCell: با استفاده از این دستور می‌توان تشخیص داد که سطر و ستون ثابت در همان سطر یا ستون که به عنوان مرکز توجه برنامه هستند دارای کادر مخصوصی می‌باشد یا نه؟ این دستور دارای یک پارامتر ورودی می‌باشد:

ObjectName: نام شی Grid.

مقدار بازگشتی این دستور یک متغیر بولین می‌باشد true یعنی سلول‌های ثابت در همان سطر یا ستون به عنوان مرکز توجه برنامه هستند و false یعنی این طور نیست.

Grid.GetUnhideColumn: با استفاده از این دستور می‌توان تشخیص داد که آیا ستون‌های شی Grid می‌توانند مخفی باشند یا نه؟ این دستور دارای یک پارامتر ورودی می‌باشد:

ObjectName: نام شی Grid.

مقدار بازگشتی این دستور یک متغیر بولین می‌باشد true یعنی ستون‌های شی Grid می‌توانند مخفی باشند و false یعنی نمی‌توانند.

Grid.GetUnhideRow: با استفاده از این دستور می‌توان تشخیص داد که آیا سطرهای شی Grid می‌توانند مخفی باشند یا نه؟ این دستور دارای یک پارامتر ورودی می‌باشد:

ObjectName: نام شی Grid.

مقدار بازگشتی این دستور یک متغیر بولین می‌باشد true یعنی سطرهای شی Grid می‌توانند مخفی باشند و false یعنی نمی‌توانند.

Grid.InsertColumn: این دستور برای افزودن یک ستون جدید به شی Grid به کار می‌رود. این دستور دارای سه پارامتر ورودی می‌باشد:

ObjectName: نام شی Grid

InsertPosition: محل اضافه شدن ستون جدید. عدد 1- برای افزودن ستون به انتهای Grid به کار می‌رود.

Redraw: ر ک Grid.AutoSize.

مقدار بازگشتی این دستور یک متغیر عددی است که همان شماره‌ی ستون جدید است.

Grid.InsertRow: این دستور برای افزودن یک سطر جدید به شی Grid به کار می‌رود و دارای سه پارامتر ورودی می‌باشد:

ObjectName: نام شی Grid

InsertPosition: محل اضافه شدن سطر جدید. عدد 1- برای افزودن ستون به انتهای Grid به کار می‌رود.

Redraw: ر ک Grid.AutoSize.

مقدار بازگشتی این دستور یک متغیر عددی می‌باشد که همان شماره‌ی سطر جدید است.

Grid.IsCellEditable: این دستور برای تشخیص قابلیت ویرایش سلول مورد نظر به کار می‌رود و دارای سه پارامتر ورودی می‌باشد:

ObjectName: نام شی Grid

Row: شماره‌ی سطر مورد نظر

Column: شماره‌ی ستون مورد نظر.

مقدار بازگشتی این دستور یک متغیر بولین می‌باشد true یعنی سلول مورد نظر قابل ویرایش است و false یعنی غیر قابل ویرایش.

Grid.IsCellFixed: این دستور برای تشخیص این که آیا سلول مورد نظر ثابت (Fix) شده است یا نه؟ به کار می‌رود و دارای سه پارامتر ورودی می‌باشد:

ObjectName: نام شی Grid

Row: شماره‌ی سطر مورد نظر

Column: شماره‌ی ستون مورد نظر.

مقدار بازگشتی این دستور یک متغیر بولین می‌باشد true یعنی سلول مورد نظر ثابت (Fix) شده است و false یعنی سلول مورد نظر ثابت (Fix) نشده است.

Grid.IsCellSelected: این دستور برای تشخیص این که آیا سلول مورد نظر انتخاب شده است یا نه؟ به کار می‌رود و دارای سه پارامتر ورودی می‌باشد:

ObjectName: نام شی Grid

Row: شماره‌ی سطر مورد نظر

Column: شماره‌ی ستون مورد نظر.

مقدار بازگشتی این دستور یک متغیر بولین می‌باشد true یعنی سلول مورد نظر انتخاب شده است و false یعنی سلول مورد نظر انتخاب نشده است.

Grid.IsCellValid: این دستور برای تشخیص این که آیا سلول مورد نظر معتبر است یا نه؟ به کار می‌رود و دارای سه پارامتر ورودی می‌باشد:

ObjectName: نام شی Grid

Row: شماره‌ی سطر مورد نظر

Column: شماره‌ی ستون مورد نظر.

مقدار بازگشتی این دستور یک متغیر بولین می‌باشد true یعنی سلول مورد نظر معتبر است و false یعنی سلول مورد نظر معتبر نیست.

Grid.IsCellVisible: این دستور برای تشخیص این که آیا سلول مورد نظر قابل رویت است یا نه؟ به کار می‌رود و دارای سه پارامتر ورودی می‌باشد:

ObjectName: نام شی Grid

Row: شماره‌ی سطر مورد نظر

Column: شماره‌ی ستون مورد نظر.

مقدار بازگشتی این دستور یک متغیر بولین می‌باشد true یعنی سلول مورد نظر قابل رویت است و false یعنی سلول مورد نظر قابل رویت نیست.

Grid.IsEnabled: این دستور برای تشخیص این که آیا شی Grid مورد نظر فعال است یا نه؟ به کار می‌رود و دارای یک پارامتر ورودی می‌باشد:

ObjectName: نام شی Grid.

مقدار بازگشتی این دستور یک متغیر بولین می‌باشد true یعنی شی Grid مورد نظر فعال است و false یعنی شی Grid مورد نظر فعال نیست.

Grid.IsVisible: این دستور برای تشخیص این که آیا شی Grid مورد نظر قابل رویت است یا نه؟ به کار می‌رود و دارای یک پارامتر ورودی می‌باشد:

ObjectName: نام شی Grid.

مقدار بازگشتی این دستور یک متغیر بولین می‌باشد true یعنی شی Grid مورد نظر قابل رویت است و false یعنی شی Grid مورد نظر قابل رویت نیست.

Grid.LoadFromFile: این دستور برای فراخوانی محتویات شی Grid از یک فایل با فرمت csv به کار می‌رود که می‌توان برنامه‌هایی نظیر Microsoft Excel آن را طراحی نمود و دارای چهار پارامتر ورودی می‌باشد:

ObjectName: نام شی Grid

FullPath: مسیر کامل فایل

SeparatorCharacter: کاراکتر جداکننده

AutoSize: تنظیم کننده‌ی خاصیت تغییر اندازه‌ی خودکار سلول‌ها که دو مقدار true (تنظیم شود) و false(تنظیم نشود) می‌باشد.

مقدار بازگشتی این دستور یک متغیر بولین می‌باشد true فایل با موفقیت فراخوانی شده است و false یعنی فراخوانی فایل موفقیت آمیز نبوده است.

Grid.MakeColorRGB: این دستور برای ساخت رنگ از ترکیب رنگ‌های قرمز، سبز و آبی به کار می‌رود و دارای چهار پارامتر ورودی می‌باشد:

ObjectName: نام شی Grid

Red: عدد 0 تا 255 برای رنگ قرمز

Green: عدد 0 تا 255 برای رنگ سبز، Blue: عدد 0 تا 255 برای رنگ آبی

مقدار بازگشتی این دستور یک متغیر عددی می‌باشد که نتیجه‌ی رنگ درست شده را بر می‌گرداند.

Grid.Print: با استفاده از این دستور می‌توانیم محتویات شی Grid مورد نظر را چاپ کنیم. این دستور دارای چهار پارامتر ورودی می‌باشد:

ObjectName: نام شی Grid

WYSIWYG: یک مدل برای چاپ که اگر مقدار آن برابر با true باشد عمل خواهد کرد و اگر false باشد عمل نخواهد کرد، ShadedPrintOut: اگر مقدار آن برابر با true باشد سلول‌ها سایه دار چاپ می‌شوند.

MarginInfo: این ورودی پارامتری را به صورت ارایه ای دریافت می‌کند که شامل موارد زیر می‌باشد:

ویژگی‌ها	نوع	توضیحات
HeaderHeight	number	اندازه‌ی سربرگ
FooterHeight	number	اندازه‌ی پا برگ
LeftMargin	number	حاشیه چپ
RightMargin	number	حاشیه راست
TopMargin	number	حاشیه بالا
BottomMargin	number	حاشیه پایین
Gap	number	فاصله‌ی بین عنوان ستون‌ها و سربرگ.

Grid.RedrawCell: این دستور برای رسم مجدد سلولی خاص در شی Grid به کار می‌رود و دارای سه پارامتر می‌باشد: ObjectName: نام شی Grid

Row: شماره‌ی سطر مورد نظر

Column: شماره‌ی ستون مورد نظر.

Grid.RedrawColumn: این دستور برای رسم مجدد ستونی خاص در شی Grid به کار می‌رود و دارای دو پارامتر می‌باشد: ObjectName: نام شی Grid

Column: شماره‌ی ستون مورد نظر.

Grid.RedrawRow: این دستور برای رسم مجدد ردیف یا سطر خاصی در شی Grid به کار می‌رود و دارای دو پارامتر می‌باشد: ObjectName: نام شی Grid

Row: شماره‌ی سطر مورد نظر.

Grid.Refresh: این دستور موجب رسم شی Grid می‌شود و دارای یک پارامتر می‌باشد: ObjectName: نام شی Grid.

Grid.SaveToFile: این دستور برای ذخیره محتویات شی Grid در حافظه‌ی دیسک سخت به کار می‌رود و دارای سه پارامتر می‌باشد:

ObjectName: نام شی Grid

FullPath: مسیر مورد نظر برای ذخیره‌ی محتویات شی Grid

SeparatorCharacter: کاراکتر جداکننده

Grid.SelectAll: این دستور برای انتخاب تمام سلول‌های شی Grid به غیر از سلول‌های ثابت شده (Fix) به کار می‌رود و دارای یک پارامتر می‌باشد:

ObjectName: نام شی Grid.

Grid.SetCellColors: این دستور برای تنظیم کردن رنگ پس زمینه و متن شی Grid به کار می‌رود و دارای پنج پارامتر می‌باشد:

ObjectName: نام شی Grid

Row: شماره‌ی سطر مورد نظر

Column: شماره‌ی ستون مورد نظر

Colors: آرایه ای از رنگ‌ها را دریافت می‌کند

Redraw: ر ک Grid.AutoSize.

مثال:

```
tbColors = {Background=16777215,Text=0};
Grid.SetCellColors("Grid1",1,1,tbColors);
```

Grid.SetCellFont: این دستور برای تنظیم کردن قلم (فونت) سلول خاصی در شی Grid به کار می‌رود و دارای پنج پارامتر ورودی می‌باشد:

ObjectName : نام شی Grid

Row: شماره‌ی سطر مورد نظر

Column: شماره‌ی ستون مورد نظر

FontData: آرایه ای از اطلاعات فونت‌ها به صورت جدول زیر:

ویژگی	نوع	توضیحات
FaceName	string	نام فونت
Height	number	اندازه‌ی فونت(درازا)
Weight	number	پهنای فونت

نام ثابت	مقدار	توضیح
FW_DONTCARE	0	بدون تغییر
FW_THIN	100	نازک

			FW_EXTRALIGHT	200	نور بیشتر
			FW_LIGHT	300	روشن
			FW_NORMAL	400	معمولی
			FW_MEDIUM	500	متوسط
			FW_SEMIBOLD	600	نیمه پر
			FW_BOLD	700	توپر
			FW_EXTRABOLD	800	توپر اضافی
			FW_HEAVY	900	سنگین
Italic	boolean	true: فعال بودن ویژگی خمیدگی برای فونت، false: غیر فعال بودن آن			
Underline	boolean	true: فعال بودن ویژگی خط زیرین برای فونت، false: غیر فعال بودن آن			

Grid.SetCellText: این دستور برای وارد کردن متن در یک سلول خاص از شی Grid به کار می‌رود و دارای پنج پارامتر می‌باشد:

ObjectName: نام شی Grid

Row: شماره‌ی سطر مورد نظر

Column: شماره‌ی ستون مورد نظر

Text: متن مورد نظر را دریافت می‌کند

Redraw: ر ک Grid.AutoSize.

Grid.SetColumnCount: این دستور برای تعیین تعداد ستون‌های شی Grid به کار می‌رود و دارای پنج پارامتر می‌باشد: ObjectName: نام شی Grid،

ColumnCount : تعداد ستون‌های مورد نظر.

Grid.SetColumnHiding: این دستور برای تنظیم کردن قابلیت مخفی شدن ستون‌ها به کار می‌رود و دارای دو پارامتر ورودی می‌باشد:

ObjectName: نام شی Grid

AllowHiding:true مخفی کردن مجاز است، false: مخفی کردن غیر مجاز است.

Grid.SetColumnResize: این دستور برای تنظیم کردن قابلیت تغییر اندازه‌ی ستون‌های شی Grid به کار می‌رود و دارای دو پارامتر ورودی می‌باشد:

ObjectName: نام شی Grid

Resizable:true تغییر اندازه دادن مجاز می‌باشد، false: تغییر اندازه دادن مجاز نمی‌باشد.

Grid.SetColumnWidth: این دستور برای تنظیم کردن اندازه‌ی پهنای ستون‌ها به کار می‌رود و دارای چهار پارامتر ورودی می‌باشد:

ObjectName: نام شی Grid

Column: شماره ستون

Width: پهنای شی ستون

Redraw: ر ک Grid.AutoSize.

Grid.SetDragAndDrop: این دستور برای تنظیم کردن ویژگی کشیدن و رها کردن ماوس به وسیله‌ی کاربر به کار می‌رود و دارای دو پارامتر ورودی می‌باشد:

ObjectName: نام شی Grid

Enable: true فعال کردن ویژگی مربوطه، false: غیر فعال کردن آن.

Grid.SetEditable: این دستور برای تنظیم کردن ویژگی ویرایش شی Grid به کار می‌رود و دارای دو پارامتر ورودی می‌باشد:

ObjectName: نام شی Grid

Enable: true فعال کردن ویژگی مربوطه، false: غیر فعال کردن آن.

Grid.SetFixedColumnCount: این دستور برای ثابت کردن ستون‌های مشخصی از شی Grid به کار می‌رود و دارای دو پارامتر ورودی می‌باشد:

ObjectName: نام شی Grid

ColumnCount: تعداد ستون‌هایی که می‌بایست ثابت گردند.

Grid.SetFixedColumnSelection: این دستور برای فعال و غیر فعال کردن ویژگی انتخاب سلول‌های زیرین ستون Fix شده با کلیک روی آن به کار می‌رود و دارای دو پارامتر ورودی می‌باشد:

ObjectName: نام شی Grid

Enable: true فعال کردن ویژگی مربوطه، false: غیر فعال کردن آن.

Grid.SetFixedRowCount: این دستور برای ثابت (Fix) کردن ردیف‌های مشخصی از شی Grid به کار می‌رود و دارای دو پارامتر ورودی می‌باشد:

ObjectName: نام شی Grid

RowCount: تعداد سطرهایی که می‌بایست ثابت گردند.

Grid.SetFixedRowSelection: این دستور برای فعال و غیر فعال کردن ویژگی انتخاب ردیف‌های کناری ردیف Fix شده با کلیک روی آن به کار می‌رود و دارای دو پارامتر ورودی می‌باشد:

ObjectName: نام شی Grid

Enable: true فعال کردن ویژگی مربوطه، false: غیر فعال کردن آن.

Grid.SetFocusCell: این دستور برای مرکز توجه قرار دادن سلول مورد نظر توسط برنامه به کار می‌رود و دارای سه پارامتر ورودی می‌باشد:

ObjectName: نام شی Grid

Row: شماره‌ی سطر، Column: شماره‌ی ستون.

Grid.SetFrameFocusCell: این دستور برای برجسته کردن سلول مرکز توجه قرار داده شده به کار می‌رود و دارای سه پارامتر ورودی می‌باشد:

ObjectName: نام شی Grid

Enable: true فعال کردن ویژگی مربوطه، false: غیر فعال کردن آن

Redraw: ر ک Grid.AutoSize.

Grid.SetGridColors: این دستور برای تنظیم کردن رنگ پس زمینه، رنگ خطهای جدا کننده و رنگ متن توضیحی به کار می‌رود و دارای دو پارامتر ورودی می‌باشد:

ObjectName: نام شی Grid

Colors: که به صورت آرایه از جدول زیر می‌باشد:

نام ویژگی	نوع	توضیحات
GridBackground	number	رنگ پس زمینه شی Grid را بر می‌گرداند.
GridLines	number	رنگ خطهای شی Grid را بر می‌گرداند.
TooltipBackground	number	رنگ پس زمینه‌ی متن توضیحی شی Grid را بر می‌گرداند.
TooltipText	number	رنگ متن توضیحی شی Grid را بر می‌گرداند.

Grid.SetGridLines: این دستور برای نمایش خط جدا کننده در شی Grid به کار می‌رود و دارای دو پارامتر ورودی می‌باشد:

ObjectName: نام شی Grid

GridLines: که به صورت آرایه از جدول زیر می‌باشد:

نام ثابت	مقدار	توضیح
GVL_NONE	0	بدون سطر
GVL_HORZ	1	سطر افقی
GVL_VERT	2	عمودی
GVL_BOTH	3	هر دو

Grid.SetHeaderSort: این دستور برای تنظیم کردن ویژگی مرتب سازی به وسیله کلیک بر روی سربرگ ستون ثابت شده به کار می‌رود و دارای دو پارامتر ورودی می‌باشد:

ObjectName: نام شی Grid

SortOnClick: true فعال کردن ویژگی مربوطه، false: غیر فعال کردن آن.

Grid.SetListMode: زمانی که از ویژگی ListMode استفاده می‌کنیم دو ویژگی دیگر یعنی مرتب سازی بر اساس سربرگ ستون و انتخاب گزینه های زیرین آن فعال می‌گردند. با این دستور Grid.SetListMode می‌توان ویژگی ListMode را فعال یا غیر فعال نمود. این دستور دارای دو پارامتر ورودی می‌باشد:

ObjectName: نام شی Grid

ListMode: true فعال کردن ویژگی مربوطه، false: غیر فعال کردن آن.

Grid.SetModified: این دستور برای تنظیم ویژگی اصلاح شده برای سلولی از شی Grid به کار می‌رود و دارای چهار پارامتر ورودی می‌باشد:

ObjectName: نام شی Grid

Modified: true فعال کردن ویژگی مربوطه، false: غیر فعال کردن آن

Row: شماره‌ی سطر

Column: شماره‌ی ستون.

Grid.SetPos: این دستور برای تنظیم مختصات قرار گیری شی Grid بر روی صفحه‌ی پروژه به کار می‌رود و دارای سه پارامتر ورودی می‌باشد:

ObjectName: نام شی Grid

Y: مختصات عمودی

X: مختصات افقی.

Grid.SetProperties: این دستور برای تنظیم ویژگی‌های شی Grid به کار می‌رود و دارای دو پارامتر ورودی می‌باشد: ObjectName: نام شی Grid

Properties: مشخصات که به صورت آرایه ای می‌باشد و توضیح آن در قسمت Grid.GetProperties بیان شد.

مثال:

```
tblGridProps = {};
tblGridProps.Width = 250;
tblGridProps.Height = 250;
tblGridProps.Y = 0;
tblGridProps.X = 0;
tblGridProps.Enabled = true;
tblGridProps.Visible = true;
Grid.SetProperties("Grid1", tblGridProps);
```

Grid.SetRedraw: این دستور برای فعال و غیرفعال ویژگی رسم مجدد شی Grid به کار می‌رود و دارای سه پارامتر ورودی می‌باشد:

ObjectName: نام شی Grid

Redraw: ر ک Grid.AutoSize

RedrawScrollbars: رسم مجدد میله‌ی لغزان true فعال کردن ویژگی مربوطه، false: غیر فعال کردن آن.

Grid.SetRowCount: این دستور برای تعیین تعداد سطرهای شی Grid به کار می‌رود و دارای دو پارامتر ورودی می‌باشد: ObjectName: نام شی Grid

RowCount: تعداد سطرها.

Grid.SetRowHeight: این دستور برای تعیین درازای سطر خاصی از شی Grid به کار می‌رود و دارای چهار پارامتر ورودی می‌باشد:

ObjectName: نام شی Grid

Row: شماره‌ی سطر

Height: درازای سطر مورد نظر

Redraw: ر ک Grid.AutoSize.

Grid.SetRowHiding: این دستور برای تنظیم کردن قابلیت مخفی شدن سطرها به کار می‌رود و دارای دو پارامتر ورودی می‌باشد:

ObjectName: نام شی Grid

AllowHiding:true مخفی کردن مجاز است، false: مخفی کردن غیر مجاز است.

Grid.SetRowResize: این دستور برای تنظیم کردن قابلیت تغییر اندازه‌ی سطرهای شی Grid به کار می‌رود و دارای دو پارامتر ورودی می‌باشد:

ObjectName: نام شی Grid

Resizable:true تغییر اندازه دادن مجاز می‌باشد، false: تغییر اندازه دادن مجاز نمی‌باشد.

Grid.SetSelectable: این دستور برای تنظیم کردن قابلیت انتخاب سلول‌های شی Grid به کار می‌رود و دارای دو پارامتر ورودی می‌باشد:

ObjectName: نام شی Grid

Selectable:true انتخاب کردن مجاز می‌باشد، false: انتخاب کردن مجاز نمی‌باشد.

Grid.SetSelectedRange: این دستور برای انتخاب دسته ای از سلول‌ها به کار می‌رود و و دارای شش پارامتر ورودی می‌باشد:

ObjectName: نام شی Grid

MinRow: شماره کوچک‌ترین سطری که انتخاب از آن آغاز می‌شود، MinColumn: شماره کوچک‌ترین ستونی که انتخاب از آن آغاز می‌شود.

MaxRow: شماره بزرگ‌ترین سطری که انتخاب در آن تمام می‌شود، MaxColumn: شماره بزرگ‌ترین ستونی که انتخاب در آن تمام می‌شود.

SelectCells: true انتخاب کردن بیش از یک مورد مجاز است، false: انتخاب کردن بیش از یک مورد غیر مجاز است.

Grid.SetSingleColumnSelection: این دستور برای تنظیم کردن قابلیت انتخاب ستون‌ها به صورت تکی به کار می‌رود و دارای دو پارامتر ورودی می‌باشد:

ObjectName: نام شی Grid

SingleColumnSelection: true انتخاب کردن به صورت تکی مجاز می‌گردد، false: انتخاب کردن به صورت تکی غیرمجاز خواهد بود.

Grid.SetSingleRowSelection: این دستور برای تنظیم کردن قابلیت انتخاب سطرها به صورت تکی به کار می‌رود و دارای دو پارامتر ورودی می‌باشد:

ObjectName: نام شی Grid

SingleRowSelection: true انتخاب کردن به صورت تکی مجاز خواهد بود، false: انتخاب کردن به صورت تکی غیرمجاز خواهد بود.

Grid.SetSize: این دستور برای تنظیم اندازه‌ی شی Grid به کار می‌رود و دارای سه پارامتر ورودی می‌باشد: ObjectName: نام شی Grid

Width: پهنای شی

Height: درازای شی.

Grid.SetSortAscending: این دستور برای تنظیم کردن مرتب سازی شی Grid به صورت صعودی به کار می‌رود و دارای دو پارامتر ورودی می‌باشد:

ObjectName: نام شی Grid

Ascending: true : مرتب سازی صعودی و false مرتب سازی نزولی.

Grid.SetTabEnabled: این دستور برای فعال و غیر فعال کردن کلید Tab برای جا به جایی بین سلول‌ها به کار می‌رود و دارای دو پارامتر ورودی می‌باشد:

ObjectName: نام شی Grid

UseTabKey: true : فعال بودن ویژگی مورد نظر، false غیرفعال بودن ویژگی مورد نظر.

Grid.SetToolTipsEnabled: این دستور برای فعال و غیر فعال کردن متن توضیحی به کار می‌رود و دارای دو پارامتر ورودی می‌باشد:

ObjectName: نام شی Grid

Tooltips: true : فعال بودن ویژگی مورد نظر، false غیرفعال بودن ویژگی مورد نظر.

Grid.SetTrackFocusCell: این دستور برای فعال سازی و غیر فعال سازی کادر مخصوص سطر و ستون ثابت در همان سطر یا ستون که به عنوان مرکز توجه برنامه هستند به کار می‌رود این دستور دارای سه پارامتر ورودی می‌باشد:

ObjectName: نام شی Grid

TrackFocus: true : فعال بودن ویژگی مورد نظر، false غیرفعال بودن ویژگی مورد نظر

Redraw: ر ک Grid.AutoSize.

Grid.SetUnhideColumn: این دستور برای فعال سازی و غیر فعال سازی ویژگی تغییر اندازه‌ی ستون‌های مخفی شده با کشیدن ماوس به کار می‌رود. این دستور دارای دو پارامتر ورودی می‌باشد:

ObjectName: نام شی Grid

AllowUnhide: true : ستون با تغییر اندازه می‌تواند از حالت مخفی در بیاید، false ستون با تغییر اندازه نمی‌تواند از حالت مخفی در بیاید.

Grid.SetUnhideRow: این دستور برای فعال سازی و غیر فعال سازی ویژگی تغییر اندازه‌ی سطرهای مخفی شده با کشیدن ماوس به کار می‌رود. این دستور دارای دو پارامتر ورودی می‌باشد:

ObjectName: نام شی Grid

AllowUnhide: true : سطر با تغییر اندازه می‌تواند از حالت مخفی در بیاید، false سطر با تغییر اندازه نمی‌تواند از حالت مخفی در بیاید.

Grid.SetVisible: این دستور برای مخفی کردن و ظاهر کردن شی Grid به کار می‌رود. این دستور دارای دو پارامتر ورودی می‌باشد:

ObjectName: نام شی Grid

Visible: true : ظاهر کردن شی، false: مخفی کردن شی.

Grid.SortTextItems: این دستور برای مرتب سازی داده های یک ستون از شی Grid به کار می‌رود. . این دستور دارای دو پارامتر ورودی می‌باشد:

ObjectName: نام شی Grid

Column: شماره‌ی ستون مورد نظر

Ascending: true: مرتب سازی صعودی و false مرتب سازی نزولی

Redraw: ر ک Grid.AutoSize.

نکته: شماره گذاری ستون‌ها و ردیف‌های شی Grid از 0 شروع می‌شود، در این شماره گذاری سطرها و ستون‌های ثابت(Fix) نیز شمرده می‌شوند.

Hotspot

این تابع برای کار با شی Hotspot که در هنگام اجرای برنامه به چشم نمی‌آید به کار می‌رود که دارای دستورات زیر مجموعه‌ی زیر می‌باشد:

Hotspot.GetPos: با استفاده از این دستور می‌توانید موقعیت قرارگیری شی HotSpot را بدست آورید .این موقعیت نسبت به گوشه‌ی بالا و سمت چپ محاسبه می‌شود.ورودی این تابع نام شی HotSpot است.

مقدار بازگشتی یک آرایه با متغیرهایی برای طول و عرض می‌باشد.

Hotspot.GetProperties: این دستور خصوصیات شی HotSpot را در یک آرایه ذخیره و در اختیار شما قرار می‌دهد.نام شی ورودی دستور است.

مقدار بازگشتی آرایه ای با خانه های زیر است :

ObjectName: نام شی

Enabled: فعال یا عدم فعال بودن HotSpot (بولین)

X÷: فاصله HotSpot از سمت چپ صفحه

Y: فاصله HotSpot از سمت راست صفحه

Width: عرض HotSpot (پیکسل)

Height: طول HotSpot (پیکسل)

TooltipText: متن نمایش دهنده زمانی که نشانگر ماوس روی آن HotSpot قرار می‌گیرد.

Cursor: شکل نشانگر ماوس. آرایه ای با اعضای: ARROW , HAND , BLACK_ARROW , CROSSHAIR , EXPLORE , HELP , MAGNIFY , MEDIA , MONEY , NOTEPAD , PENCIL , PRINTER , SPEAKER یا UP_ARROW

ResizeLeft: قابلیت تغییر اندازه از سمت چپ در زمان تغییر اندازه پروژه (بولین)

ResizeRight: قابلیت تغییر اندازه از سمت راست در زمان تغییر اندازه پروژه (بولین)

ResizeTop: قابلیت تغییر اندازه از بالا در زمان تغییر اندازه پروژه (بولین)

ResizeBottom: قابلیت تغییر اندازه از پایین در زمان تغییر اندازه پروژه (بولین)

HighlightSound: نوع صوت در هنگام قرارگیری نشانگر ماوس بر روی HotSpot (NONE, STANDARD یا CUSTOM)

HighlightSoundFile: مسیر فایل صوتی پخش شونده در هنگام قرارگیری نشانگر ماوس بر روی HotSpot

ClickSound: نوع صوت در هنگام کلیک بر روی HotSpot (NONE, STANDARD یا CUSTOM)

ClickSoundFile: مسیر فایل صوتی پخش شونده در هنگام کلیک بر روی HotSpot

Hotspot.GetSize: نمایش اندازه شی HotSpot به پیکسل با این دستور امکان پذیر است.ورودی دستور نام شی می‌باشد.

مقدار بازگشتی دستور، آرایه ای 2 عضوی با اعضای "Width" و "Height" می‌باشد.

Hotspot.IsEnabled: با استفاده از این دستور می‌توانید فعال یا غیرفعال بودن HotSpot را بدست آورید. مقدار ورودی نام شی است.

مقدار بازگشتی Trueبه معنای فعال بودن یا False به معنای غیرفعال بودن HotSpot می‌باشد.

Hotspot.SetEnabled: با این دستور می‌توانید HotSpot را فعال یا غیر فعال کنید.ورودی دستور نام شی و همچنین عبارت بولین برای فعال یا غیرفعال کردن آن می‌باشد.

مقدار بازگشتی **ندارد** .

Hotspot.SetPos: با این کد می‌توانید HotSpot را به محل دلخواه در صفحه منتقل نمایید.ورودی‌های این کد نام شی، اعدادی برای فاصله از چپ و بالا می‌باشد.

Hotspot.SetProperties:با این دستور می‌توانید خواص مورد دلخواه خود را بر روی HotSpot اعمال کنید .ورودی‌های دستور، نام شی و آرایه ای حاوی خصوصیات می‌باشد که این آرایه در Hotspot.GetProperties توضیح داده شده است.

مقدار بازگشتی **ندارد** .

Hotspot.SetSize: با این دستور می‌توانید اندازه HotSpot را به اندازه دلخواه خود تغییر دهید.ورودی‌های این دستور نام شی، عددی برای عرض و عددی دیگر برای طول می‌باشد.

مقدار بازگشتی **ندارد** .

HTTP

این تابع برای انجام کارهایی اینترنتی نظر دریافت فایل، بررسی وضعیت اتصال به اینترنت، ثبت اطلاعات در یک سایت و ... به کار می‌رود که دستورات زیر مجموعه‌ی آن به شرح زیر است:

HTTP.Download: با استفاده از این دستور می‌توانید فایل را از اینترنت دانلود کنید.ورودی‌های دستور عبارتند از: مسیر فایل بر روی اینترنت که شامل نام سایت سرویس دهنده و نام فایل است (http://autoplay.ir/Book.rar)، محل ذخیره فایل، نوع ذخیره (معمولی یا متنی)، میزان وقفه (حال معمولی 20 می‌باشد)، پورت اتصال (در حالت معمولی 80)، نام کاربری و کلمه عبور برای تایید (بعضی از سایت‌ها نیاز دارند، در صورتی که سایت مورد نظر شما نیاز به این قسمت ندارد، nil جای این پارامتر قرار دهید)، اطلاعات پروکسی و نام تابع بازگشتی .

مقدار بازگشتی **ندارد** .

HTTP.DownloadSecure: با این دستور فایل‌ها را با امنیت بالا (https) دانلود کنید. ورودی‌های این دستور مانند ورودی‌های دستور قبلی می‌باشند.

مقدار بازگشتی ندارد.

HTTP.GetConnectionState: وضعیت اتصال به اینترنت با استفاده از این دستور قابل کنترل می‌باشد.

مقدار بازگشتی آرایه ای است حاوی 7 متغیر بولین :

Connected: متصل بودن یا نبودن به اینترنت

Modem: استفاده از مودم برای اتصال

LAN: استفاده از Lan (شبکه محلی) برای اتصال

Proxy: استفاده از پروکسی سرور برای اتصال به اینترنت

RASInstalled: فعال بودن یا نبودن Remote Access Service

ConnectionOffline: فعال یا غیرفعال بودن حالت Offline

ConnectionConfigured: وجود یا عدم وجود اتصال معتبر برای اتصال به اینترنت

HTTP.GetFileSize:با این دستور می‌توانید حجم یک فایل در یک سایت را بر حسب بایت به دست آورید. ورودی‌های دستور با ورودی‌های HTTP.Download تفاوتی ندارد.

مقدار بازگشتی میزان حجم فایل بر حسب بایت است.

HTTP.GetFileSizeSecure: این دستور هم حجم فایل را در پروتکل https نمایش می‌دهد. ورودی‌ها با دستور قبل یکی می‌باشد.

مقدار بازگشتی هم مانند دستور قبل میزان حجم فایل بر حسب بایت است.

HTTP.GetHTTPErrorInfo: اطلاعات آخرین خطای دستورهای HTTP را باز می‌گرداند.

مقدار بازگشتی آرایه ای است شامل :

Number: شماره خطا یا رویداد

Message: متن خطا یا رویداد

Status: وضعیت خطا

HTTP.Submit: با این دستور می‌توانید اطلاعاتی را در یک وب سایت ثبت نمایید.ورودی‌ها، مسیر فایل در وب سایت مورد نظر برای ثبت اطلاعات در آن، متغیر (ها)، نوع قرارگیری اطلاعات، پورت برای اتصال، اطلاعات برای تایید و پروکسی می‌باشند.

مقدار بازگشتی متن بازگشت داده شده توسط فایل می‌باشد.

HTTP.SubmitSecure: این دستور اطلاعات را در سایت‌های پیاده سازی شده بر روی https ثبت می‌کند. ورودی‌ها مانند کد بالا است.

مقدار بازگشتی متن بازگشت داده شده توسط فایل می‌باشد.

HTTP.TestConnection: چک کردن وضعیت و صحت اتصال به اینترنت .ورودی‌ها URL برای کنترل اتصال، میزان وقفه، پورت، اطلاعات تأییدیه و پروکسی است.

مقدار بازگشتی متغیر بولین می‌باشد که True نشان دهنده متصل بودن به اینترنت است.

Image

تابع Image برای کار با شی Image(عکس) به کار می‌رود که دارای دستورات زیر مجموعه‌ی زیر می‌باشد:

Image.GetFileInfo: با این کد می‌توانید طول و عرض و کیفیت تصویر را به دست آورید.ورودی مسیر فایل تصویری است.

مقدار بازگشتی آرایه ای متشکل از Width , Height و BitDepth می‌باشد.

Image.GetFilename: با این دستور می‌توانید مسیر تصویر بارگذاری شده در شی Image را به دست آورید . ورودی دستور نام شی می‌باشد.

مقدار بازگشتی مسیر تصویر می‌باشد. در صورت بروز خطا یا پیدا نکردن تصویر رشته خالی بازگردانده می‌شود.

Image.GetOpacity: میزان شفافیت تصویر با استفاده از این دستور به دست می‌آید . ورودی دستور نام شی Image می‌باشد.

مقدار بازگشتی عددی بین 0 تا 100 می‌باشد که نشان دهنده شفافیت تصویر است.

Image.GetPos: با استفاده از این دستور می‌توانید موقعیت قرارگیری شی Image را بدست آورید .این موقعیت نسبت به گوشه‌ی بالا و سمت چپ محاسبه می‌شود.ورودی این تابع نام شی Image است.

مقدار بازگشتی یک آرایه با متغیرهایی برای طول و عرض می‌باشد.

Image.GetProperties:این دستور خصوصیات شی Image را در یک آرایه ذخیره می‌کند .ورودی نام شی می‌باشد.

مقدار بازگشتی آرایه ای است حاوی :

ImageFile: نام تصویر بارگذاری شده در شی

UseTransColor:True در صورت استفاده از transparent

TransparentColor: شماره رنگ استفاده شده برای transparent

Tolerance: عدد transparent بین 0 تا 100

Opacity: میزان شفافیت تصویر

HitTest: حالت HITTEST به صورت STANDARD یا ALPHA

ObjectName: نام شی

Enabled: فعال یا عدم فعال بودن دکمه (بولین)

Visible:دیده شدن یا عدم دیده شدن دکمه (بولین)

X: فاصله دکمه از سمت چپ صفحه

Y: فاصله دکمه از سمت راست صفحه

Width: عرض دکمه (پیکسل)

Height:طول دکمه (پیکسل)

TooltipText: متن نمایش دهنده زمانی که نشانگر ماوس روی آن دکمه قرار می‌گیرد.

Cursor:شکل نشانگر ماوس. آرایه ای با اعضای: ARROW , HAND , BLACK_ARROW , CROSSHAIR , EXPLORE , HELP , MAGNIFY , MEDIA , MONEY , NOTEPAD , PENCIL , PRINTER , SPEAKER یا UP_ARROW

ResizeLeft:قابلیت تغییر اندازه از سمت چپ در زمان تغییر اندازه پروژه (بولین)

ResizeRight: قابلیت تغییر اندازه از سمت راست در زمان تغییر اندازه پروژه (بولین)

ResizeTop: قابلیت تغییر اندازه از بالا در زمان تغییر اندازه پروژه (بولین)

ResizeBottom: قابلیت تغییر اندازه از پایین در زمان تغییر اندازه پروژه (بولین)

HighlightSound: نوع صوت در هنگام قرارگیری نشانگر ماوس بر روی دکمه (NONE , STANDARD یا CUSTOM)

HighlightSoundFile: مسیر فایل صوتی پخش شونده در هنگام قرارگیری نشانگر ماوس بر روی دکمه

ClickSound: نوع صوت در هنگام کلیک بر روی دکمه(NONE , STANDARD یا CUSTOM)

ClickSoundFile: مسیر فایل صوتی پخش شونده در هنگام کلیک بر روی دکمه

Image.GetSize: نمایش اندازه شی Image به پیکسل با این دستور امکان پذیر است.ورودی دستور نام شی Image می‌باشد.

مقدار بازگشتی دستور، آرایه ای 2 عضوی با اعضای "Width" و "Height" می‌باشد.

Image.IsEnabled: با استفاده از این دستور می‌توانید فعال یا غیرفعال بودن شی Image را بدست آورید. مقدار ورودی نام شی Image است.

مقدار بازگشتی Trueبه معنای فعال بودن یا False به معنای غیرفعال بودن است.

Image.IsVisible: با استفاده از این دستور می‌توانید وضعیت نمایش یا عدم نمایش شی Image را بدست آورید. مقدار ورودی نام شی است.

مقدار بازگشتی Trueبه نمایش یا False به معنای عدم نمایش شی می‌باشد.

Image.Load:با ورود مسیر یک تصویر، می‌توانید آن را در شی Image بارگذاری نمایید.

فرمت های پشتیبانی شده :

JPG, BMP, TIFF, GIF, PhotoCD, Photoshop, WBMP, PNG, PCX, PAX, TLA, WMF, EMF, APM و TGA

مقدار بازگشتی **ندارد** .

Image.SetEnabled: با این دستور می‌توانید Image را فعال یا غیر فعال کنید.ورودی دستور نام شی Image و همچنین عبارت بولین برای فعال یا غیرفعال کردن شی می‌باشد.

مقدار بازگشتی **ندارد** .

Image.SetOpacity:با استفاده از این کد قادر به تغییر میزان شفافیت تصویر می‌باشید .ورودی‌ها، نام شی Image و عددی برای میزان شفافیت (0 بیشترین شفافیت و 100 بیشترین نمایش تصویر) می‌باشد.

مقدار بازگشتی **ندارد**.

Image.SetPos: با این دستور می‌توانید شی Image را به محل دلخواه در صفحه منتقل نمایید.ورودی‌های این کد نام شی، عددی برای X (فاصله از چپ) و عدد دیگری برای Y (فاصله از بالا) می‌باشد.

مقدار بازگشتی **ندارد** .

Image.SetProperties:می‌توانید با ورود نام شی و آرایه‌ی خصوصیات دلخواه، خصوصیات شی Image را تغییر دهید.در قسمت Image.GetProperties این آرایه توضیح داده شده است.

مقدار بازگشتی **ندارد**.

Image.SetSize: با این دستور می‌توانید اندازه شی را به اندازه دلخواه خود تغییر دهید.ورودی‌های این کد نام شی Image، عددی برای عرض و عددی دیگر برای طول دکمه می‌باشد.

مقدار بازگشتی **ندارد** .

Image.SetVisible:نمایش یا عدم نمایش دادن تصویر با این دستور امکان پذیر است .نام شی و مقدار بولین برای تعیین نمایش ورودی‌های دستورند .

مقدار بازگشتی **ندارد** .

INIFile

دستور INIFile برای کار با فایل‌هایی با فرمت ini به کار می‌رود و دستورات زیر مجموعه‌ی آن به شرح زیر است:

INIFile.DeleteSection: با این دستور می‌توانید یک Section خاص را در فایل ini حذف نمایید. ورودی‌های دستور مسیر فایل INI و نام Section فایل مورد نظر است.

مقدار بازگشتی **ندارد** .

INIFile.DeleteValue: مقدار خاص در فایل INI را با این دستور می‌توان حذف کرد. مسیر فایل، نام Section و مقدار (Value) مورد نظر ورودی‌های دستور هستند.

مقدار بازگشتی **ندارد** .

INIFile.GetSectionNames: با این دستور می‌توانید نام Section های موجود در یک فایل INI را بدست آورید. ورودی دستور مسیر فایل INI می‌باشد.

مقدار بازگشتی آرایه ای حاوی نام Section های داخل فایل می‌باشد.

INIFile.GetValue: مقدار موجود در یک فایل INI را با استفاده از این دستور می‌توان به دست آورد. مسیر فایل، نام Section و Value، ورودی‌های دستورند.

مقدار بازگشتی رشته ای حاوی Value درخواستی می‌باشد.

INIFile.GetValueNames: این دستور نام Value های موجود در یک Section را باز می‌گرداند. مسیر فایل و نام Section مورد نظر از ورودی‌های دستور هستند.

مقدار بازگشتی آرایه ای شامل نام Valueهای موجود در Section است.

INIFile.SetValue: با استفاده از این دستور می‌توانید یک مقدار خاص را داخل یک Section قرار دهید .مسیر فایل، نام Section، Value و Data ورودی‌های دستورند.

مقدار بازگشتی **ندارد**.

Input

تابع **Input برای کار با شی Input به کار می‌رود که دستورات زیر مجموعه ان به شرح زیر است:**

Input.CanUndo: با این دستور می‌توانید وضعیت فعال بودن "بازگشت" در شی Input را به دست آورید. نام شی ورودی دستور است.

مقدار بازگشتی بولین می‌باشد که True نشان دهنده فعال بودن و False نشان دهنده غیر فعال بودن این قابلیت است.

Input.Copy: این دستور متن داخل Input را در "کلیپ بورد" کپی می‌کند. نام شی ورودی دستور است.

مقدار بازگشتی **ندارد** .

Input.Cut: این دستور متن داخل Input را Cut کرده و در "کلیپ بورد" نگه می‌دارد. نام شی ورودی دستور است.

مقدار بازگشتی **ندارد** .

Input.Delete: با این دستور می‌توانید هر متنی داخل Input است را حذف نمایید. ورودی نام شی Input است.

مقدار بازگشتی **ندارد** .

Input.GetPos: با استفاده از این دستور می‌توانید موقعیت قرارگیری شی Input را بدست آورید .این موقعیت نسبت به گوشه‌ی بالا و سمت چپ محاسبه می‌شود.ورودی این تابع نام شی می‌باشد.

مقدار بازگشتی یک آرایه با متغیرهای X و Y می‌باشد.

Input.GetProperties: با استفاده از این دستور می‌توانید، خصوصیات شی Input را در اختیار داشته باشید. نام شی ورودی دستور است.

مقدار بازگشتی آرایه ای است حاوی :

Text: متن نمایش داده شده در Input

ObjectName: نام شی

FontName: نام فونت استفاده شده

FontSize: اندازه فونت

FontStrikeout: فعال یا غیر فعال بودن strikeout در تنظیمات فونت (بولین)

FontUnderline: فعال یا غیر فعال بودن underline در تنظیمات فونت (بولین)

FontAntiAlias: فعال یا غیر فعال بودن anti alias در تنظیمات فونت (بولین)

FontItalic: فعال یا غیر فعال بودن italic در تنظیمات فونت (بولین)

FontWeight: نوع نوشته (تیرگی متن) به صورت آرایه ای متشکل از DONTCARE, THIN, EXTRALIGHT, LIGHT, NORMAL, MEDIUM, SEMIBOLD, BOLD, EXTRABOLD و HEAVY.

FontScript: نوع اسکریپت فونت به صورت آرایه ای متشکل از ANSI , BALTIC , CHINESEBIG5 , DEFAULT , EASTEUROPE , GB2312 , GREEK , HANGUL , MAC , OEM , RUSSIAN , SHIFTJIS ,SYMBOL و TURKISH .

FontColor: شماره رنگ فونت

Multiline: وضعیت پشتیبانی از نوشتن چند خط (بولین)

VScrollbar: وضعیت فعال بودن اسکرول افقی

HScrollbar: وضعیت فعال بودن اسکرول عمودی

InputStyle: نوع نوشته داخل متن (معمولی، پسورد، ماسک)

MaskText: نوع ماسک استفاده شده در صورتی که InputStyle برابر 2 (ماسک) باشد.

MaskReturnMode: نوع داده بازگشتی (FORMATTED یا AS_TYPED)

Placeholder: کاراکتر PlaceHolder

Border: نوع Border (بدون Border، FLAT یا SUNKEN)

ReadOrder: نوع نمایش متن (استاندارد یا راست به چپ)

BackgroundColor: رنگ زمینه شی

ReadOnly: وضعیت خواندنی بودن شی (بولین)

Alignment: نوع چینش متن در شی (چپ، وسط و راست چین)

Enabled: فعال یا عدم فعال بودن دکمه (بولین)

Visible: دیده شدن یا عدم دیده شدن دکمه (بولین)

X: فاصله دکمه از سمت چپ صفحه

Y: فاصله دکمه از سمت راست صفحه

Width: عرض دکمه (پیکسل)

Height: طول دکمه (پیکسل)

TooltipText: متن نمایش دهنده زمانی که نشانگر ماوس روی کمبوباکس قرار می‌گیرد.

ResizeLeft: قابلیت تغییر اندازه از سمت چپ در زمان تغییر اندازه پروژه (بولین)

ResizeRight: قابلیت تغییر اندازه از سمت راست در زمان تغییر اندازه پروژه (بولین)

ResizeTop: قابلیت تغییر اندازه از بالا در زمان تغییر اندازه پروژه (بولین)

ResizeBottom: قابلیت تغییر اندازه از پایین در زمان تغییر اندازه پروژه (بولین)

WindowHandle: شماره هندل شی

Input.GetSelection: محل ابتدا و انتهای متن انتخاب شده در شی Input را با استفاده از این دستور می‌توانید به دست آورید .ورودی نام شی می‌باشد.

مقدار بازگشتی آرایه ای است با اعضای Start، End وLineNum .در صورتی که start و end یکی باشند، هیچ متنی انتخاب نشده است.

Input.GetSize: نمایش اندازه شی Input به پیکسل با این دستور امکان پذیر است.ورودی دستور نام شی می‌باشد.

مقدار بازگشتی دستور، آرایه ای 2 عضوی با اعضای "Width" و "Height" می‌باشد.

Input.GetText: با استفاده از این دستور می‌توانید متن موجود در Input را به دست آورید.نام شی ورودی دستور است.

مقدار بازگشتی رشته ای حاوی متن داخل Input می‌باشد.

Input.IsEnabled: با استفاده از این دستور می‌توانید وضعیت فعال یا غیرفعال بودن Input را بدست آورید. مقدار ورودی نام شی می‌باشد.

مقدار بازگشتی Trueبه معنای فعال بودن و False به معنای غیرفعال بودن شی Input می‌باشد.

Input.IsVisible: با استفاده از این دستور می‌توانید نمایش یا عدم نمایش Input را بدست آورید. مقدار ورودی نام شی است.

مقدار بازگشتی Trueبه معنای نمایش یا False به معنای عدم نمایش Input می‌باشد.

Input.Paste: این دستور متن داخل کلیپ بورد را در Input قرار می‌دهد. نام شی ورودی است.

مقدار بازگشتی **ندارد** .

Input.ScrollLines: با این دستور می‌توانید اسکرول تا محل دلخواه به حرکت در آورید .این قابلیت زمانی قابل استفاده است که MultiLine در شی فعال باشد. نام شی و خطی که اسکرول تا آنجا فعال باشد ورودی‌های دستور هستند.

مقدار بازگشتی **ندارد**.

Input.ScrollToLine: با این دستور می‌توانید اسکرول تا به محل مشخص ببرید .این قابلیت هم زمانی قابل استفاده است که MultiLine در شی فعال باشد. نام شی و خطی که اسکرول در آنجا قرار می‌گیرد، ورودی‌های دستور هستند.

مقدار بازگشتی **ندارد**.

Input.SetEnabled: با این دستور می‌توانید Input مورد نظر را فعال یا غیر فعال کنید.ورودی دستور نام شی و همچنین عبارت بولین برای فعال یا غیرفعال کردن آن می‌باشد.

مقدار بازگشتی **ندارد** .

Input.SetPos: با این کد می‌توانید Input را به محل دلخواه در صفحه منتقل نمایید.ورودی‌های این کد نام شی، عددی برای جای گیری در قسمت X (فاصله از چپ) و عدد دیگری برای جای گیری در Y (فاصله از بالا می‌باشد).

مقدار بازگشتی **ندارد** .

Input.SetProperties:با استفاده از این دستور می‌توانید خصوصیات جدیدی بر روی Input اعمال نمایید. نام شی و آرایه خصوصیات (در قسمت Input.GetProperties توضیح داده شده) ورودی‌ها هستند.

مقدار بازگشتی **ندارد** .

Input.SetSelection:با این دستور می‌توانید قسمتی از متن موجود در Input را انتخاب کنید. نام شی، مقدار ابتدا و مقدار انتهای محل انتخاب ورودی‌های دستور هستند .در صورت استفاده از 1- بجای ابتدا تمام متن از حالت انتخاب خارج شده و در صورت استفاده از 1- بجای انتها، تمام متن به حالت انتخاب در می‌آید.

مقدار بازگشتی **ندارد** .

Input.SetSize: با این دستور می‌توانید طول و عرض Input را به اندازه دلخواه خود تغییر دهید.ورودی‌های این کد نام شی، عددی برای عرض و عددی دیگر برای طول دکمه می‌باشد.

مقدار بازگشتی **ندارد** .

Input.SetText: با استفاده از این دستور می‌توانید متن دلخواه را در Input به نمایش در آورید. نام شی و متن ورودی‌های دستورند.

مقدار بازگشتی **ندارد** .

Input.SetVisible: کنترل نمایش یا عدم نمایش Input با این دستور امکان پذیر است. ورودی‌های کد، نام شی و عبارتی بولین برای فعال یا غیر فعال کردن نمایش شی می‌باشد.

مقدار بازگشتی **ندارد** .

Input.Undo: با ورود نام شی می‌توانید عملیات انجام شده بر روی Input را به یک مرحله قبل بازگردانید.

مقدار بازگشتی **ندارد** .

Label

دستور Label برای کار با شی Label(برچسب) به کار می‌رود که دستورات زیر مجموعه‌ی آن به شرح زیر است:

Label.GetPos: با استفاده از این دستور می‌توانید موقعیت قرارگیری شی Label را بدست آورید .این موقعیت نسبت به گوشه‌ی بالا و سمت چپ محاسبه می‌شود.ورودی این تابع نام شی "برچسب" می‌باشد .

مقدار بازگشتی یک آرایه با متغیرهای X و Y می‌باشد.

Label.GetProperties: با استفاده از این دستور می‌توانید خصوصیات شی Label را به دست آورید. نام شی ورودی دستور است .

مقدار بازگشتی آرایه ای با متغیرهای زیر است :

Text: متن نمایش داده شده

ObjectName: نام شی

FontName: نام فونت استفاده شده

FontSize: اندازه فونت

FontStrikeout: فعال یا غیر فعال بودن strikeout در تنظیمات فونت (بولین)

FontUnderline: فعال یا غیر فعال بودن underline در تنظیمات فونت (بولین)

FontAntiAlias: فعال یا غیر فعال بودن anti alias در تنظیمات فونت (بولین)

FontItalic: فعال یا غیر فعال بودن italic در تنظیمات فونت (بولین)

FontWeight: نوع نوشته (تیرگی متن) به صورت آرایه ای متشکل از DONTCARE ,THIN ,EXTRALIGHT ,LIGHT ,NORMAL ,MEDIUM, SEMIBOLD, BOLD, EXTRABOLD و HEAVY .

FontScript: نوع اسکریپت فونت به صورت آرایه ای متشکل از , ANSI , BALTIC , CHINESEBIG5 , DEFAULT , EASTEUROPE , GB2312 , GREEK , HANGUL , MAC , OEM , RUSSIAN , SHIFTJIS ,SYMBOL و TURKISH .

Alignment:نوع چینش متن در برچسب (چپ، وسط یا راست چین)

Orientation:چرخش نسبت به صفحه (0، 90، 180 یا 270 درجه)

ColorNormal:رنگ متن در حالت Normal

ColorHighlight: رنگ متن در حالت Highlight (وقتی ماوس روی آن قرار دارد)

ColorDisabled: رنگ متن در حالت غیرفعال

ColorDown: رنگ متن در حالت Down (وقتی بر روی آن کلیک می‌شود)

Enabled: فعال یا عدم فعال بودن (بولین)

Visible: دیده شدن یا عدم دیده شدن (بولین)

X: فاصله از سمت چپ صفحه

Y: فاصله از سمت راست صفحه

Width: عرض (پیکسل)

Height: طول (پیکسل)

TooltipText: متن نمایش دهنده در زمان قرارگیری نشانگر ماوس روی شی.

Cursor:عدد شکل نشانگر ماوس : ARROW , HAND , BLACK_ARROW , CROSSHAIR , EXPLORE , HELP , MAGNIFY , MEDIA , MONEY , NOTEPAD , PENCIL , PRINTER , SPEAKER , یا UP_ARROW

ResizeLeft: قابلیت تغییر اندازه از سمت چپ در زمان تغییر اندازه پروژه (بولین)

ResizeRight: قابلیت تغییر اندازه از سمت راست در زمان تغییر اندازه پروژه (بولین)

ResizeTop: قابلیت تغییر اندازه از بالا در زمان تغییر اندازه پروژه (بولین)

ResizeBottom: قابلیت تغییر اندازه از پایین در زمان تغییر اندازه پروژه (بولین)

HighlightSound: نوع صوت در هنگام قرارگیری نشانگر ماوس بر روی شی (NONE, STANDARD یا CUSTOM)

HighlightSoundFile: مسیر فایل صوتی پخش شونده در هنگام قرارگیری نشانگر ماوس بر روی شی

ClickSound: نوع صوت در هنگام کلیک بر روی شی (NONE, STANDARD یا CUSTOM)

ClickSoundFile: مسیر فایل صوتی پخش شونده در هنگام کلیک بر روی شی

Label.GetSize: نمایش اندازه شی برچسب بر حسب پیکسل با این دستور امکان پذیر است.ورودی دستور نام شی می‌باشد.

مقدار بازگشتی دستور، آرایه ای 2 عضوی با اعضای "Width" و "Height" می‌باشد.

Label.GetText: با استفاده از این دستور می‌توان متن موجود در Label را به دست آورد.نام شی ورودی دستور است.

مقدار بازگشتی رشته ای حاوی متن داخل برچسب می‌باشد.

Label.IsEnabled: با استفاده از این دستور می‌توانید وضعیت فعال یا غیرفعال بودن برچسب را بدست آورید. مقدار ورودی نام شی می‌باشد.

مقدار بازگشتی Trueبه معنای فعال بودن و False به معنای غیرفعال بودن شی Label می‌باشد.

Label.IsVisible: با استفاده از این دستور می‌توانید نمایش یا عدم نمایش Label را بدست آورید. مقدار ورودی نام شی است.

مقدار بازگشتی Trueبه معنای نمایش یا False به معنای عدم نمایش Label می‌باشد.

Label.SetEnabled: با این دستور می‌توانید برچسب را فعال یا غیر فعال کنید.ورودی دستور نام شی و همچنین عبارت بولین برای فعال یا غیرفعال کردن آن می‌باشد.

مقدار بازگشتی **ندارد** .

Label.SetPos: با این کد می‌توانید برچسب را به محل دلخواه در صفحه منتقل نمایید.ورودی‌های این کد نام شی، عددی برای جای گیری در قسمت X (فاصله از چپ) و عدد دیگری برای جای گیری در Y (فاصله از بالا می‌باشد).

مقدار بازگشتی **ندارد** .

Label.SetProperties: با استفاده از این دستور می‌توانید خصوصیات جدید برای شی Label تعیین نمایید. نام شی و آرایه خصوصیات (که در Label.GetProperties توضیح داده شده) ورودی‌های دستور هستند.

مقدار بازگشتی **ندارد** .

Label.SetSize: با این دستور می‌توانید طول و عرض برچسب را به اندازه دلخواه خود تغییر دهید.ورودی‌های این کد نام شی، عددی برای عرض و عددی دیگر برای طول برچسب می‌باشد.

مقدار بازگشتی **ندارد** .

Label.SetText: تغییر متن داخل برچسب به متن دلخواه با این دستور امکان پذیر است .نام شی و متن مورد نظر ورودی‌های دستورند.

مقدار بازگشتی **ندارد** .

Label.SetVisible: کنترل نمایش یا عدم نمایش برچسب با این دستور امکان پذیر است. ورودی‌های کد، نام شی و عبارتی بولین برای فعال یا غیر فعال کردن نمایش شی می‌باشد.

مقدار بازگشتی **ندارد** .

ListBox

دستور ListBox برای کار با شی ListBox به کار می‌رود و دارای دستورات زیر مجموعه ای به شرح زیر است:

ListBox.AddItem: با استفاده از این دستور می‌توان یک آیتم جدید به لیست باکس اضافه کرد. یکی از ورودی‌های تابع نام لیست باکس می‌باشد. ورودی‌های دیگر 2 رشته به عنوان متن و دیتا خواهند بود

مقدار بازگشتی عدد مربوط به ردیفی است که آیتم در آنجا قرار گرفته است.

ListBox.DeleteItem: این دستور امکان حذف یک آیتم را در لیست باکس به شما می‌دهد.نام شی و ردیف آیتم مورد نظر ورودی‌های دستورند .در صورت ورود 1- به جای ردیف، تمام آیتم‌های لیست باکس حذف می‌شود.

مقدار بازگشتی **ندارد** .

ListBox.DeselectItem: با این دستور می‌توانید آیتمی را از حالت انتخاب در آورید .نام شی و ردیف مورد نظر ورودی‌های دستورند.در صورت ورود 1- له جای ردیف، تمام آیتم‌ها از حالت انتخاب خارج می‌شوند.

ListBox.FindItem: این دستور امکان جستجو در میان آیتم‌ها را برای شما فراهم می‌کند.نام شی، ردیف برای شروع جستجو، نوع جستجو (Text, Dataیا هر دو) و متن برای جستجو ورودی‌های دستور هستند.

مقدار بازگشتی، عدد اولین ردیفی است که متن در آن وارد یافت شده است. در صورتی که متن یافت نشود یا خطایی صورت گیرد 1- بازگردانده می‌شود.

ListBox.GetChecked: با استفاده از این دستور می‌توانید نوع علامت دار بودن آیتم‌ها را به دست آورید .نام شی و نوع علامت (UNCHECKED, CHECKED یا INDETERMINATE) ورودی‌های دستور هستند.

مقدار بازگشتی جدولی شامل ردیف‌های مطابق با نوع علامت است.

ListBox.GetCheckedCount: این دستور مانند دستور بالا عمل می‌کند با تفاوت اینکه تعداد این ردیف‌ها را باز می‌گرداند.

مقدار بازگشتی عددی نمایش دهنده تعداد ردیف‌ها است.

ListBox.GetCount: تعداد آیتم‌های لیست باکس با استفاده از این دستور به دست می‌آید.نام شی ورودی دستور است .

مقدار بازگشتی تعداد آیتم‌ها می‌باشد.

ListBox.GetItemCheck: نوع علامت یک آیتم خاص با این دستور نمایش داده می‌شود .ورودی‌های دستور، نام شی و شماره ردیف می‌باشد .

مقدار بازگشتی عدد نوع علامت (CHECKED , UNCHECKED یا INDETERMINATE) می‌باشد.

ListBox.GetItemData: با این دستور می‌توانید Data یک آیتم خاص را به دست آورید. ورودی‌های دستور، نام شی و شماره ردیف می‌باشد .

مقدار بازگشتی رشته حاوی Data مورد نظر است.

ListBox.GetItemText: با این دستور می‌توانید متن یک آیتم را به دست آورید. ورودی‌های دستور، نام شی و شماره ردیف می‌باشد .

مقدار بازگشتی رشته حاوی متن آیتم مورد نظر است.

ListBox.GetPos: با استفاده از این دستور می‌توانید موقعیت قرارگیری شی لیست باکس را بدست آورید .این موقعیت نسبت به گوشه‌ی بالا و سمت چپ محاسبه می‌شود.ورودی این تابع نام شی می‌باشد.

مقدار بازگشتی یک آرایه با متغیرهای X و Y می‌باشد.

ListBox.GetProperties: این دستور خصوصیات شی ListBox را در اختیار شما قرار می‌دهد. نام شی ورودی تابع است.

مقدار بازگشتی آرایه ای شامل موارد زیر می‌باشد :

ObjectName: نام شی

ListBoxType: نوع لیست باکس (LISTBOX یا CHECKLISTBOX)

MultipleSelection: وضعیت چند خطه بودن (بولین)

Sort: وضعیت مرتب سازی (بولین)

VScrollbar: فعال یا غیرفعال بودن اسکرول افقی (بولین)

HScrollbar: فعال یا غیرفعال بودن اسکرول عمودی (بولین)

FontName: نام فونت استفاده شده

FontSize: اندازه فونت

FontStrikeout: فعال یا غیر فعال بودن strikeout در تنظیمات فونت (بولین)

FontUnderline: فعال یا غیر فعال بودن underline در تنظیمات فونت (بولین)

FontAntiAlias: فعال یا غیر فعال بودن anti alias در تنظیمات فونت (بولین)

FontItalic: فعال یا غیر فعال بودن italic در تنظیمات فونت (بولین)

FontWeight: نوع نوشته (تیرگی متن) به صورت آرایه ای متشکل از DONTCARE ,THIN ,EXTRALIGHT ,LIGHT ,NORMAL ,MEDIUM, SEMIBOLD, BOLD, EXTRABOLD و HEAVY .

FontScript: نوع اسکریپت فونت به صورت آرایه ای متشکل از , ANSI , BALTIC , CHINESEBIG5 , DEFAULT , EASTEUROPE , GB2312 , GREEK , HANGUL , MAC , OEM , RUSSIAN , SHIFTJIS ,SYMBOL و TURKISH .

TextColor: شماره رنگ متن

BackgroundColor: شماره رنگ پس زمینه

Border: نوع Border (بدون Border، FLAT یا SUNKEN)

ReadOrder: نوع نمایش متن (استاندارد یا راست به چپ)

Enabled: فعال یا عدم فعال بودن (بولین)

Visible: دیده شدن یا عدم دیده شدن (بولین)

X: فاصله از سمت چپ صفحه

Y: فاصله از سمت راست صفحه

Width: عرض (پیکسل)

Height: طول (پیکسل)

TooltipText: متن نمایش دهنده در زمان قرارگیری نشانگر ماوس روی شی.

Cursor: عدد شکل نشانگر ماوس : , ARROW , HAND , BLACK_ARROW
CROSSHAIR , EXPLORE , HELP , MAGNIFY , MEDIA , MONEY
UP_ARROW یا , NOTEPAD , PENCIL , PRINTER , SPEAKER

ResizeLeft: قابلیت تغییر اندازه از سمت چپ در زمان تغییر اندازه پروژه (بولین)

ResizeRight: قابلیت تغییر اندازه از سمت راست در زمان تغییر اندازه پروژه (بولین)

ResizeTop: قابلیت تغییر اندازه از بالا در زمان تغییر اندازه پروژه (بولین)

ResizeBottom: قابلیت تغییر اندازه از پایین در زمان تغییر اندازه پروژه (بولین)

WindowHandle: شماره هندل شی

ListBox.GetSelected: با این دستور می‌توانید آیتم‌های انتخاب شده را به دست آورید .ورودی نام شی است .

مقدار بازگشتی جدولی شامل شماره ردیف‌های انتخاب شده است.

ListBox.GetSelectedCount: این دستور با دریافت نام شی تعداد آیتم‌های انتخاب شده را نمایش می‌دهد.

مقدار بازگشتی تعداد آیتم‌های انتخاب شده است.

ListBox.GetSize: نمایش اندازه شی لیست باکس بر حسب پیکسل با این دستور امکان پذیر است. ورودی دستور نام شی می‌باشد.
مقدار بازگشتی دستور، آرایه ای 2 عضوی با اعضای "Width" و "Height" می‌باشد.

ListBox.GetType: با ورود نام شی در این دستور می‌توانید نوع لیست باکس را به دست آورید.

مقدار بازگشتی 0 یا 1 (LISTBOX یا CHECKLISTBOX) می‌باشد .

ListBox.InsertItem: با این دستور می‌توانید یک آیتم به لیست باکس در محل دلخواه اضافه کنید .نام شی، ردیف مورد نظر، متن و Data ورودی‌های دستورند.

مقدار بازگشتی عدد ردیفی است که آیتم جدید در آن اضافه شده است .

ListBox.IsEnabled: با استفاده از این دستور می‌توانید وضعیت فعال یا غیرفعال بودن لیست باکس را بدست آورید. مقدار ورودی نام شی می‌باشد.

مقدار بازگشتی Trueبه معنای فعال بودن و False به معنای غیرفعال بودن شی ListBox می‌باشد.

ListBox.IsItemSelected:با این دستور وضعیت انتخاب بودن یک آیتم را می‌توان به دست آورد.نام شی و شماره ردیف برای کنترل، ورودی‌های دستور هستند.

مقدار بازگشتی عبارتی بولین برای نمایش انتخاب شده بودن یا عدم انتخاب آیتم می‌باشد.

ListBox.IsVisible: با استفاده از این دستور می‌توانید نمایش یا عدم نمایش لیست باکس را بدست آورید. مقدار ورودی نام شی است.

مقدار بازگشتی Trueبه معنای نمایش یا False به معنای عدم نمایش ListBox می‌باشد.

ListBox.SelectItem: با این دستور می‌توانید آیتم مورد دلخواهتان را به حالت انتخاب در آورید.مقدار ورودی‌ها، نام شی و شماره ردیف می‌باشند.در صورت قرارگیری 1- در شماره ردیف، تمام آیتم‌ها به حالت انتخاب در می‌آیند.

مقدار بازگشتی **ندارد** .

ListBox.SetEnabled: با این دستور می‌توانید لیست باکس مورد نظر را فعال یا غیر فعال کنید.ورودی دستور نام شی و همچنین عبارت بولین برای فعال یا غیرفعال کردن آن می‌باشد.

مقدار بازگشتی **ندارد** .

ListBox.SetItemCheck: با این دستور می‌توانید حالت دلخواه برای آیتم‌هایی که قابلیت علامت خوردن را دارند تعیین نمایید.نام شی، شماره ردیف و نوع حالت (UNCHECKED، CHECKED یا INDETERMINATE) ورودی‌های کد می‌باشند.

ListBox.SetItemData: با این دستور می‌توانید Data یک آیتم خاص را تغییر دهید. ورودی‌های دستور، نام شی و شماره ردیف و Data جدید می‌باشد .

مقدار بازگشتی **ندارد** .

ListBox.SetItemText: با این دستور می‌توانید متن یک آیتم خاص را تغییر دهید. ورودی‌های دستور، نام شی و شماره ردیف و رشته ای حاوی متن می‌باشد .

مقدار بازگشتی **ندارد** .

ListBox.SetPos: با این کد می‌توانید محل قرارگیری لیست باکس را تعیین نمایید.ورودی‌های این کد نام شی، عددی فاصله از چپ و عدد دیگری برای فاصله از بالا می‌باشد.

مقدار بازگشتی **ندارد** .

ListBox.SetProperties: با استفاده از این دستور می‌توانید خصوصیات مورد نظرتان را بر روی لیست باکس اعمال کنید .ورودی دستور، نام شی و آرایه خصوصیات جدید می‌باشد. (این آرایه در ListBox.GetProperties توضیح داده شده است.)

ListBox.SetSize: با این دستور می‌توانید طول و عرض لیست باکس را به اندازه دلخواه خود تغییر دهید.ورودی‌های این کد نام شی، عددی برای عرض و عددی دیگر برای طول دکمه می‌باشد.

مقدار بازگشتی **ندارد** .

ListBox.SetType: با ورود نام شی در این دستور می‌توانید نوع لیست باکس را تغییر دهید.نام شی و 0 یا 1 (LISTBOX یا CHECKLISTBOX) می‌باشد.

مقدار بازگشتی **ندارد** .

ListBox.SetUpdate: فعال و غیرفعال کردن آپدیت لحظه ای (در هنگام انجام عملیات بر روی لیست باکس) با این دستور امکان پذیر است.نام شی و مقدار بولین برای فعال و غیرفعال کردن این قابلیت، ورودی‌های دستور هستند.

مقدار بازگشتی **ندارد** .

ListBox.SetVisible: کنترل نمایش یا عدم نمایش لیست باکس با این دستور امکان پذیر است. ورودی‌های کد، نام شی و عبارتی بولین برای فعال یا غیر فعال کردن نمایش شی می‌باشد.

مقدار بازگشتی **ندارد** .

Math

تابع Math برای انجام عملیات ریاضی به کار می‌رود. این تابع دارای دستورات زیر مجموعه ای به شرح زیر می‌باشد:

Math.Abs: با استفاده از این دستور می‌توانید قدر مطلق یک عدد را به دست آورید .عدد اولیه ورودی دستور است.

مقدار بازگشتی، عددی برابر قدر مطلق عدد وارد شده می‌باشد.

Math.Acos: کسینوس معکوس (آرک کسینوس) یک عدد را با این دستور می‌توانید به دست آورید. ورودی عددی است بین 1- و 1 .

مقدار بازگشتی عدد بر حسب رادیان می‌باشد.

Math.Asin: سینوس معکوس (آرک سینوس) یک عدد هم با این دستور به دست می‌آید. ورودی عددی است بین 1- و 1 .

مقدار بازگشتی عدد بر حسب رادیان می‌باشد.

Math.Atan: با این دستور می‌توانید تانژانت معکوس یک عدد را به دست آورید. ورودی عددی است بین 1- و 1 .

مقدار بازگشتی عدد بر حسب رادیان می‌باشد.

Math.Atan2: این دستور هم تانژانت معکوس دو عدد وارد شده را باز می‌گرداند.

مقدار بازگشتی عدد بر حسب رادیان می‌باشد.

Math.Ceil: این دستور عدد وارد شده به عنوان ورودی را به نزدیک‌ترین عدد صحیح می‌رساند (عدد را رو به بالا گرد می‌کند.)

مقدار بازگشتی عدد جدید می‌باشد.

Math.Cos: این دستور امکان محاسبه کسینوس را فراهم می‌کند. ورودی عددی بر حسب رادیان می‌باشد.

مقدار بازگشتی عدد محاسبه شده می‌باشد.

Math.Deg: این دستور، عدد وارد شده بر حسب رادیان را به درجه تبدیل می‌کند.

مقدار بازگشتی عدد محاسبه شده بر حسب درجه می‌باشد.

Math.Exp: این دستور ارزش نمایی عدد را باز می‌گرداند .ورودی تابع عدد مورد نظر است.

مقدار بازگشتی عدد محاسبه شده می‌باشد.

Math.Floor: این دستور عدد وارد شده به عنوان ورودی را به نزدیک‌ترین عدد صحیح می‌رساند (عدد را رو به پایین گرد می‌کند.)

مقدار بازگشتی عدد جدید می‌باشد.

Math.Frexp:با استفاده از این دستور می‌توانید عددی مانند X را وارد کرده و مقدار M و N را از فرمول زیر بیابید.

$X= M * 2^N$

مقدار بازگشتی آرایه ای است با مقادیر: عددی بین 0.5 و 1 (Mantissa) و عددی صحیح (Exponent) .

Math.HexColorToNumber: این دستور شماره رنگ را در مبنای 16 گرفته و در مبنای 10 برمی‌گرداند.

مقدار بازگشتی عدد مربوط به رنگ می‌باشد.

Math.HexToNumber: با این دستور می‌توانید عدد را از مبنای 16(هگز) به مبنای 10 (دسیمال) تبدیل کنید.

مقدار بازگشتی عدد مربوط به رنگ می‌باشد.

Math.Ldexp: این دستور برعکس دستور Math.Frexp عمل می‌کند.یعنی مقادیر M و N را وارد کرده، مقدار X را بدست آورید.

مقدار بازگشتی عدد محاسبه شده می‌باشد.

Math.Log: این دستور به شما امکان محاسبه‌ی لگاریتم عدد را می‌دهد.ورودی عدد مورد نظر می‌باشد.

مقدار بازگشتی عدد محاسبه شده می‌باشد.

Math.Log10: با استفاده از این دستور می‌توانید لگاریتم عدد ورودی را در مبنای 10 به دست آورید.

مقدار بازگشتی عدد محاسبه شده می‌باشد.

Math.Max: با استفاده از این دستور می‌توانید دو عدد را مقایسه کنید و عدد بزرگ‌تر را به دست آورید.ورودی‌های دستور، 2 عدد می‌باشند.

مقدار بازگشتی عدد بزرگ‌تر است.

Math.Min: با استفاده از این دستور می‌توانید دو عدد را مقایسه کنید و عدد کوچک‌تر را به دست آورید.ورودی‌های دستور، 2 عدد می‌باشند.

مقدار بازگشتی عدد کوچک‌تر است.

Math.Mod: با استفاده از این دستور می‌توانید باقی مانده تقسیمی را به دست آورید .دو عدد صورت و مخرج ورودی‌های دستور هستند.

مقدار بازگشتی باقیمانده تقسیم می‌باشد.

Math.Pow:با ورود عدد پایه و توان به عنوان ورودی، عدد را به توان دلخواه می‌رساند.

مقدار بازگشتی عدد محاسبه شده می‌باشد.

Math.Rad: این دستور، عدد وارد شده بر حسب درجه را به رادیان تبدیل می‌کند.

مقدار بازگشتی عدد محاسبه شده بر حسب رادیان می‌باشد.

Math.Random: این دستور با ورود عدد شروع و پایان، عددی تصادفی بین این دو عدد تولید می‌کند.

مقدار بازگشتی عدد تصادفی می‌باشد.

Math.RandomSeed: مقدار مورد استفاده برای تولید عدد تصادفی با استفاده از این دستور تولید می‌شود.

مقدار بازگشتی **ندارد** .

Math.RGBToNumber: با ورود میزان رنگ‌های قرمز، آبی، سبز (اعدادی بین 0 و 255) عدد رنگ را محاسبه کنید.

مقدار بازگشتی عدد رنگ مورد نظر است.

Math.Round: با استفاده از این دستور می‌توانید عدد را به نزدیک‌ترین عدد گرد می‌کند. عدد و تعداد اعشار مورد نیاز برای محاسبه ورودی‌های دستور هستند.

مقدار بازگشتی عددی محاسبه شده می‌باشد.

Math.Sin: این دستور امکان محاسبه سینوس را فراهم می‌کند. ورودی عددی بر حسب رادیان می‌باشد.

مقدار بازگشتی عدد محاسبه شده می‌باشد.

Math.Sqrt: با استفاده از این دستور می‌توانید جذر عدد مورد نیازتان را به دست آورید.

مقدار بازگشتی جذر عدد وارد شده است.

Math.Tan: این دستور تانژانت عدد وارد شده بر حسب رادیان را محاسبه می‌کند.

مقدار بازگشتی عدد محاسبه شده می‌باشد.

MSI

این دستور برای کار با فایل‌های نصب ویندوزی که با پسوند msi مشخص می‌شوند به کار می‌رود. این دستور دارای شامل 36 زیر مجموعه می‌باشد که در این کتاب به دلیل خارج بودن از موضوع به آن‌ها نمی‌پردازیم اما چنانچه خوانندگان محترم نیاز به یادگیری این دستور و زیر مجموعه های آن داشتند می‌توانند به انجمن تخصصی AMS یعنی www.vasva3.com مراجعه نمایند و در انجمن Autoplay Media Studio سؤالات خود را راجع به این دستور مطرح کنند. شایان ذکر است که در سریع‌ترین زمان به سؤالات پاسخ داده می‌شود.

Page

دستور Page برای کار با صفحه های ایجاد شده در پروژه به کار می‌رود و دارای دستوراتِ زیر مجموعه ای با شرح زیر است:

Page.ClickObject: این دستور با گرفتن نام شی مورد نظر در صفحه، کدهای On Click آن شی را اجرا می‌کند.

مقدار بازگشتی **ندارد** .

Page.CreateObject: با استفاده از این دستور می‌توانید شی خاصی را در صفحه بسازید. ورودی‌های این دستور، عددی برای مشخص کردن نوع شی درخواستی، نام شی جدید (دقت داشته باشید نام شی با اشیاء داخل دیالوگ مشابه نباشد)، جدولی متشکل از خصوصیات شی است.(جدول خصوصیات هر شی در دستورات مربوط به همان شی نوشته شده است.)

مقدار بازگشتی **ندارد** .

Page.DeleteObject: برای حذف شی مورد نظر از صفحه می‌توانید از این دستور استفاده کنید .ورودی کد، نام شی می‌باشد.

مقدار بازگشتی **ندارد** .

Page.EnumerateObjects: برای بدست آوردن نام تمام اشیاء موجود در صفحه، از این دستور باید استفاده نمایید.

مقدار بازگشتی، آرایه ای متشکل از نام اشیاء می‌باشد. در صورت عدم وجود شی در دیالوگ یا بروز مشکل مقدار nil بازگردانده می‌شود.

Page.GetFocus: با استفاده از این دستور می‌توانید شی که بر روی آن Focus شده را بدست آورید.

مقدار بازگشتی نام شی که بر روی آن Focus شده خواهد بود.

Page.GetObjectScript: با استفاده از این دستور می‌توانید دستورات رویداد مورد نظرتان در شی خاص در صفحه حاضر را بدست آورید.ورودی‌های دستور، نام شی و رویداد مورد نظر می‌باشد.

مقدار بازگشتی رشته ای حاوی دستورات بدست آمده می‌باشد.

Page.GetObjectType: این دستور با گرفتن نام شی به عنوان ورودی، نوع شی را بازمی‌گرداند.

مقدار بازگشتی عددی نمایش دهنده نوع شی است.

BUTTON(0) , LABEL(1) , PARAGRAPH(2) , IMAGE(3) , FLASH(4) , VIDEO(5) , WEB(6) , INPUT(7) , HOTSPOT(8) , LISTBOX(9) , COMBOBOX(10) , PROGRESS(11) , TREE(12) , RADIOBUTTON(13) , RICHTEXT(14) , CHECKBOX(15) , SLIDESHOW(16) , GRID(17) , PDF(18) , QUICKTIME(19) , XBUTTON(20) , PLUGIN(40)

Page.GetRadioValue: با استفاده از این دستور می‌توانید اطلاعات مربوط به دکمه رادیویی انتخاب شده در صفحه را بدست آوردید.ورودی دستور، کد گروه دکمه رادیویی و نوع داده مورد نیاز (Value، نام شی یا متن آن) می‌باشد.

مقدار بازگشتی رشته ای حاوی نوع داده خواسته شده است .در صورتی که دکمه رادیویی انتخاب نشده باشد یا خطایی صورت گرفته باشد رشته ای خالی بازگردانده می‌شود.

Page.GetSize: با این دستور می‌توانید اندازه صفحه حاضر را به پیکسل به دست آورید.

مقدار بازگشتی آرایه ای متشکل از خانه های Width و Height می‌باشد.

Page.Jump: با استفاده از این دستور می‌توانید به صفحه دلخواه بروید.نام صفحه ای که نیازمند نمایش آن هستید ورودی دستور است.

مقدار بازگشتی **ندارد** .

Page.Navigate: با این دستور می‌توانید صفحه را به صورت ترتیبی باز کنید .ورودی، عددی است بین 0 تا 5 نمایش دهنده نوع پرش صفحه (اولین، آخرین، بعدی، قبلی، بازگشت به عقب و بازگشت به جلو).

مقدار بازگشتی **ندارد** .

Page.Print: با استفاده از این دستور می‌توانید از صفحه حاضر پرینت بگیرید .شی ویدئو در پرینت نمایش داده نمی‌شود.ورودی‌ها، دو متغیر بولین است .یکی نمایش و عدم نمایش دیالوگ پرینت قبل از عمل پرینت، دیگری یکی کردن اندازه تصویر صفحه با صفحه پرینت شده است.

مقدار بازگشتی **ندارد** .

Page.Redraw: با این دستور می‌توانید اشیاء را دوباره رسم کنید (صفحه را به روز رسانی کنید)

مقدار بازگشتی **ندارد** .

Page.SetFocus: با استفاده از این دستور می‌توانید بر روی شی خاص Focus نمایید. فوکوس تنها بر روی اشیاء Flash, input, listbox, combobox, tree و Web انجام می‌شود .ورودی دستور نام شی مورد نظر است.

مقدار بازگشتی **ندارد** .

Page.SetObjectScript: با استفاده از این دستور می‌توانید دستورات خاصی را در رویداد مورد نظر شی موجود در صفحه قرار دهید.ورودی‌های تابع، نام شی، رویداد مورد نظر، و دستورات مورد نیاز می‌باشد.

مقدار بازگشتی **ندارد** .

Page.SetObjectZOrder: با استفاده از این دستور می‌توانید نوع چینش شی را تعیین کنید، نام شی، نوع چینش (FRONT(0) , BACK(1) , FORWARD(2) , BACKWARD(3) , INSERT_BEFORE(4) یا BEHIND(5))

مقدار بازگشتی **ندارد** .

Page.SetRadioValue: با این دستور می‌توانید دکمه رادیویی خاصی را در صفحه به انتخاب درآورید.ورودی‌ها عبارتند از: متن داده انتخاب شده، کد گروه دکمه رادیویی و نوع داده (VALUE, OBJECTNAME, TEXT که متن آن در ورودی اول وارد شده).

مقدار بازگشتی **ندارد** .

Page.StartTimer: با اجرای این کد، تایمر (شمارنده) مربوط به دیالوگ فعال می‌شود .ورودی دستور زمان به میلی ثانیه و کد (ID) تایمر می‌باشد.

مقدار بازگشتی **ندارد**.

Page.StopTimer: با استفاده از این دستور می‌توانید تایمر فعال شده با کد قبلی را غیرفعال نمایید.برای این کار کافی است کد تایمر را به عنوان ورودی دستور وارد نمایید.

مقدار بازگشتی **ندارد**.

Paragraph

این دستور برای کار با شی Paragraph به کار می‌رود که دستورات زیر مجموعه‌ی آن به شرح زیر هستند:

Paragraph.GetPos: با استفاده از این دستور می‌توانید موقعیت قرارگیری شی "پاراگراف" را بدست آورید .این موقعیت نسبت به گوشه‌ی بالا و سمت چپ محاسبه می‌شود.ورودی این تابع نام شی می‌باشد .

مقدار بازگشتی یک آرایه با متغیرهای X و Y می‌باشد.

Paragraph.GetProperties: این دستور خصوصیات شی پاراگراف را در اختیار شما قرار می‌دهد.نام شی ورودی تابع است.

مقدار بازگشتی آرایه ای شامل موارد زیر می‌باشد :

Text: متن نمایش داده شده

ObjectName:نام شی

FontSize: اندازه فونت

FontStrikeout: فعال یا غیر فعال بودن strikeout در تنظیمات فونت (بولین)

FontUnderline: فعال یا غیر فعال بودن underline در تنظیمات فونت (بولین)

FontAntiAlias: فعال یا غیر فعال بودن anti alias در تنظیمات فونت (بولین)

FontItalic: فعال یا غیر فعال بودن italic در تنظیمات فونت (بولین)

FontWeight: نوع نوشته (تیرگی متن) به صورت آرایه ای متشکل از DONTCARE ,THIN ,EXTRALIGHT ,LIGHT ,NORMAL ,MEDIUM, SEMIBOLD, BOLD, EXTRABOLD و HEAVY .

FontScript: نوع اسکریپت فونت به صورت آرایه ای متشکل از , ANSI , BALTIC , CHINESEBIG5 , DEFAULT , EASTEUROPE , GB2312 , GREEK , HANGUL , MAC , OEM , RUSSIAN , SHIFTJIS ,SYMBOL و TURKISH .

BGStyle: 0 یا 1 (نمایش دهنده حالت SOLID یا TRANSPARENT)

BGColor:رنگ پس زمینه در صورتی که BGStyle برابر 1 باشد

BorderStyle: 0 یا 1 (بدون Border یا Solid Border)

BorderColor:عدد رنگ Border

ScrollStyle:0 یا 1 (نوع اسکرول – ساده یا سفارشی)

SkinFile:مسیر اسکرول استفاده شده در صورتی که ScrollStyle برابر 1 باشد.

ScrollHorizontal:عدد بین 0 تا 2 نشان دهنده نوع اسکرول افقی (خودکار، فعال، غیرفعال)

ScrollVertical: عدد بین 0 تا 2 نشان دهنده نوع اسکرول عمودی(خودکار، فعال، غیرفعال
)

Alignment:نوع چینش متن (چپ، وسط یا راست چین)

ColorNormal:رنگ متن در حالت Normal

ColorHighlight: رنگ متن در حالت Highlight (وقتی ماوس روی آن قرار دارد)

ColorDisabled: رنگ متن در حالت غیرفعال

ColorDown: رنگ متن در حالت Down (وقتی بر روی آن کلیک می‌شود)

Enabled: فعال یا عدم فعال بودن (بولین)

Visible: دیده شدن یا عدم دیده شدن (بولین)

X: فاصله از سمت چپ صفحه

Y: فاصله از سمت راست صفحه

Width: عرض (پیکسل)

Height: طول (پیکسل)

TooltipText: متن نمایش دهنده در زمان قرارگیری نشانگر ماوس روی شی.

Cursor:عدد شکل نشانگر ماوس : , ARROW , HAND , BLACK_ARROW
CROSSHAIR , EXPLORE , HELP , MAGNIFY , MEDIA , MONEY
UP_ARROW یا , NOTEPAD , PENCIL , PRINTER , SPEAKER

ResizeLeft: قابلیت تغییر اندازه از سمت چپ در زمان تغییر اندازه پروژه (بولین)

ResizeRight: قابلیت تغییر اندازه از سمت راست در زمان تغییر اندازه پروژه (بولین)

ResizeTop: قابلیت تغییر اندازه از بالا در زمان تغییر اندازه پروژه (بولین)

ResizeBottom: قابلیت تغییر اندازه از پایین در زمان تغییر اندازه پروژه (بولین)

HighlightSound: نوع صوت در هنگام قرارگیری نشانگر ماوس بر روی شی (, NONE
STANDARD یا CUSTOM)

HighlightSoundFile: مسیر فایل صوتی پخش شونده در هنگام قرارگیری نشانگر ماوس بر روی شی

ClickSound: نوع صوت در هنگام کلیک بر روی شی(NONE , STANDARD یا CUSTOM)

ClickSoundFile: مسیر فایل صوتی پخش شونده در هنگام کلیک بر روی شی

Paragraph.GetScrollPos:با این دستور می‌توانید موقعیت اسکرول را به دست آورید.نام شی و مقداری بولین ورودی‌های دستورند .با True بودن این محل اسکرول عمودی و با False بودن آن اسکرول افقی نمایش داده می‌شود.

مقدار بازگشتی محل اسکرول می‌باشد.

Paragraph.GetScrollRange:کمترین و بیشترین مقدار اسکرول با استفاده از این دستور به دست می‌آید. نام شی و نوع اسکرول (بولین) ورودی‌های دستورند .

مقدار بازگشتی آرایه ای دو عضوی با اعضای Max و Min است.

Paragraph.GetSize: نمایش اندازه شی پاراگراف بر حسب پیکسل با این دستور امکان پذیر است.ورودی دستور نام شی می‌باشد.

مقدار بازگشتی دستور، آرایه ای 2 عضوی با اعضای "Width" و "Height" می‌باشد.

Paragraph.GetText: با استفاده از این دستور می‌توان متن موجود در شی پاراگراف را به دست آورد.نام شی ورودی دستور است.

مقدار بازگشتی رشته ای حاوی متن داخل می‌باشد.

Paragraph.IsEnabled: با استفاده از این دستور می‌توانید وضعیت فعال یا غیرفعال بودن شی پاراگراف را بدست آورید. مقدار ورودی نام شی می‌باشد.

مقدار بازگشتی Trueبه معنای فعال بودن و False به معنای غیرفعال بودن پاراگراف می‌باشد.

Paragraph.IsVisible: با استفاده از این دستور می‌توانید نمایش یا عدم نمایش شی پاراگراف را بدست آورید. مقدار ورودی نام شی است.

مقدار بازگشتی Trueبه معنای نمایش یا False به معنای عدم نمایش پاراگراف می‌باشد.

Paragraph.SetEnabled: با این دستور می‌توانید پاراگراف را فعال یا غیر فعال کنید.ورودی دستور نام شی و همچنین عبارت بولین برای فعال یا غیرفعال کردن آن می‌باشد.

مقدار بازگشتی **ندارد** .

Paragraph.SetPos: با این کد می‌توانید پاراگراف را به محل دلخواه در صفحه منتقل نمایید. ورودی‌های این کد نام شی، عددی برای جای گیری در قسمت X (فاصله از چپ) و عدد دیگری برای جای گیری در Y (فاصله از بالا می‌باشد).

مقدار بازگشتی **ندارد** .

Paragraph.SetProperties: با استفاده از این دستور می‌توانید خصوصیات جدید برای شی پاراگراف تعیین نمایید. نام شی و آرایه خصوصیات (که در Paragraph.GetProperties توضیح داده شده) ورودی‌های دستور هستند.

مقدار بازگشتی **ندارد** .

Paragraph.SetScrollPos:با استفاده از این دستور می‌توانید اسکرول را به محل مورد نظر منتقل نمایید.نام شی، عدد جایگاه مورد نظر و نوع اسکرول (true برای عمودی و false برای افقی) ورودی‌های دستورند.

مقدار بازگشتی **ندارد** .

Paragraph.SetSize: با این دستور می‌توانید طول و عرض شی پاراگراف را به اندازه دلخواه خود تغییر دهید.ورودی‌های این کد نام شی، عددی برای عرض و عددی دیگر برای طول پاراگراف می‌باشد.

مقدار بازگشتی **ندارد** .

Paragraph.SetText: تغییر متن پاراگراف به متن دلخواه با این دستور امکان پذیر است .نام شی و متن مورد نظر ورودی‌های دستورند.

مقدار بازگشتی **ندارد** .

Paragraph.SetVisible: کنترل نمایش یا عدم نمایش پاراگراف با این دستور امکان پذیر است. ورودی‌های کد، نام شی و عبارتی بولین برای فعال یا غیر فعال کردن نمایش شی می‌باشد.

مقدار بازگشتی **ندارد** .

PDF

این تابع برای کار با شی PDF به کار می‌رود که زیر مجموعه‌ی دستورات آن به شرح زیر می‌باشد:

PDF.GetFile: با استفاده از این دستور می‌توانید مسیر فایل بارگذاری شده در شی را به دست آورید. نام شی PDF ورودی دستور است.

مقدار بازگشتی مسیر یا آدرس اینترنتی فایل بارگذاری شده می‌باشد.

PDF.GetPos: با استفاده از این دستور می‌توانید موقعیت قرارگیری شیPDF را بدست آورید .ورودی این تابع نام شی می‌باشد .

مقدار بازگشتی یک آرایه با متغیرهای X و Y می‌باشد.

PDF.GetProperties: با استفاده از این دستور می‌توانید با ورود نام شی PDF خصوصیات آن را در اختیار داشته باشید. ورودی دستور نام شی است.

مقدار بازگشتی آرایه ای است با اعضای زیر :

ObjectName: نام شی

File: فایل استفاده شده در شی

ShowScrollbars: وضعیت نمایش اسکرول (بولین)

ShowToolbar: وضعیت نمایش تولبار (بولین)

ShowBorder: وضعیت نمایش Border (بولین)

Layout: نوع نمایش فایل (پیش فرض، تک صفحه، تک ستون، دو ستون سمت راست و دو ستون سمت چپ)

Page: حالت صفحات (معمولی، بوک مارک یا بند انگشتی)

View: نوع نمایش صفحات

ViewOffset: عدد ViewOffset تنظیم شده در خصوصیات شی

Enabled: فعال یا عدم فعال بودن (بولین)

Visible: دیده شدن یا عدم دیده شدن (بولین)

X: فاصله از سمت چپ صفحه

Y: فاصله از سمت راست صفحه

Width: عرض (پیکسل)

Height: طول (پیکسل)

ResizeLeft: قابلیت تغییر اندازه از سمت چپ در زمان تغییر اندازه پروژه (بولین)

ResizeRight: قابلیت تغییر اندازه از سمت راست در زمان تغییر اندازه پروژه (بولین)

ResizeTop: قابلیت تغییر اندازه از بالا در زمان تغییر اندازه پروژه (بولین)

ResizeBottom: قابلیت تغییر اندازه از پایین در زمان تغییر اندازه پروژه (بولین)

HighlightSound: نوع صوت در هنگام قرارگیری نشانگر ماوس بر روی شی (NONE , STANDARD یا CUSTOM)

HighlightSoundFile: مسیر فایل صوتی پخش شونده در هنگام قرارگیری نشانگر ماوس بر روی شی

WindowHandle: شماره هندل شی

PDF.GetSize: نمایش اندازه شی PDF بر حسب پیکسل با این دستور امکان پذیر است.ورودی دستور نام شی می‌باشد.

مقدار بازگشتی دستور، آرایه ای 2 عضوی با اعضای "Width" و "Height" می‌باشد.

PDF.GoToPage: با ورود نام شی و عدد مربوط به صفحه مورد نظر می‌توانید آن صفحه را به نمایش در آورید.

مقدار بازگشتی **ندارد** .

PDF.IsEnabled: با استفاده از این دستور می‌توانید وضعیت فعال یا غیرفعال بودن شی PDF را بدست آورید. مقدار ورودی نام شی می‌باشد.

مقدار بازگشتی Trueبه معنای فعال بودن و False به معنای غیرفعال بودن شی می‌باشد.

PDF.IsVisible: با استفاده از این دستور می‌توانید وضعیت نمایش یا عدم نمایش شی PDF را بدست آورید. مقدار ورودی نام شی است.

مقدار بازگشتی Trueبه معنای نمایش یا False به معنای عدم نمایش می‌باشد.

PDF.LoadFile: با استفاده از این دستور می‌توانید فایل جدیدی را در شی بارگذاری نمایید.نام شی و مسیر فایل ورودی‌های دستورند.

مقدار بازگشتی **ندارد** .

PDF.Navigate: با استفاده از این دستور می‌توانید به صفحات دلخواه پرش داشته باشد.نام شی و شماره نوع پرش (اولین صفحه، آخرین صفحه،صفحه بعد، صفحه قبل، صفحه قبلی، صفحه بعدی) ورودی‌های دستور هستند.

مقدار بازگشتی **ندارد** .

PDF.Print:با استفاده از این دستور می‌توانید فایل PDF را چاپ کنید .ورودی‌های دستور نام شی، صفحه شروع پرینت، صفحه اتمام آن و مقداری بولین برای مشخص کردن وضعیت همسان سازی اندازه با صفحه است .

مقدار بازگشتی **ندارد** .

PDF.PrintWithDialog: این دستور هم امکان چاپ PDF را می‌دهد با تفاوت اینکه ورودی دستور فقط نام شی است و با اجرای این دستور پنجره تنظیمات پرینت نمایش داده می‌شود.

مقدار بازگشتی **ندارد** .

PDF.SetEnabled: با این دستور می‌توانید شی PDF را فعال یا غیر فعال کنید. ورودی دستور نام شی و همچنین عبارت بولین برای فعال یا غیرفعال کردن آن می‌باشد.

مقدار بازگشتی **ندارد** .

PDF.SetNamedDest: با وارد کردن نام محل مورد نظر می‌توانید آن قسمت را به نمایش در آورید .نام شی و رشته مورد نظر ورودی‌های دستورند.مقدار بازگشتی ندارد.

PDF.SetPos: با این کد می‌توانید شیPDF را به محل دلخواه در صفحه منتقل نمایید.ورودی‌های این کد نام شی، اعدادی برای مشخص کردن میزان فاصله از چپ و بالا می‌باشد.

مقدار بازگشتی **ندارد** .

PDF.SetProperties: با استفاده از این دستور می‌توانید خصوصیات جدیدی برای شی PDF اعمال کنید. نام شی و آرایه خصوصیات (که در PDF.GetProperties توضیح داده شده) ورودی‌های دستورند.

مقدار بازگشتی **ندارد** .

PDF.SetSize: با این دستور می‌توانید طول و عرض شی PDF را به اندازه دلخواه خود تغییر دهید.ورودی‌های این کد نام شی، عددی برای عرض و عددی دیگر برای طول شی می‌باشد.

مقدار بازگشتی **ندارد** .

PDF.SetVisible: کنترل نمایش یا عدم نمایش شی PDF با این دستور امکان پذیر است. ورودی‌های کد، نام شی و عبارتی بولین برای فعال یا غیر فعال کردن نمایش شی می‌باشد.

مقدار بازگشتی **ندارد** .

PDF.SetZoomScroll:با استفاده از این دستور می‌توانید قسمتی از شی را بزرگ نمایی کنید .نام شی، میزان بزرگ نمایی، مکان افقی و عمودی ورودی‌های دستورند.

مقدار بازگشتی **ندارد** .

Plugin

این تابع برای کار با اشیای افزونه ای به کار می‌روند که دستورات زیر مجموعه‌ی آن به شرح زیر می‌باشند:

Plugin.GetPos: با استفاده از این دستور می‌توانید موقعیت قرارگیری شی پلاگین را بدست آورید .این موقعیت نسبت به گوشه‌ی بالا و سمت چپ محاسبه می‌شود.ورودی این دستور نام شی می‌باشد.

مقدار بازگشتی یک آرایه با متغیرهای X و Y می‌باشد.

Plugin.GetSize: نمایش اندازه شی پلاگین با این دستور امکان پذیر است.ورودی دستور نام شی می‌باشد.

مقدار بازگشتی دستور، آرایه ای 2 عضوی با اعضای "Width" و "Height" می‌باشد.

Plugin.IsEnabled: با استفاده از این دستور می‌توانید وضعیت فعال یا غیرفعال بودن شی پلاگین را بدست آورید. مقدار ورودی نام شی می‌باشد.

مقدار بازگشتی Trueبه معنای فعال بودن و False به معنای غیرفعال بودن شی می‌باشد.

Plugin.IsVisible: با استفاده از این دستور می‌توانید وضعیت نمایش یا عدم نمایش شی پلاگین را بدست آورید. مقدار ورودی نام شی است.

مقدار بازگشتی True به معنای نمایش یا False به معنای عدم نمایش شی می‌باشد.

Plugin.SetEnabled: با این دستور می‌توانید شی پلاگین را فعال یا غیر فعال کنید.ورودی دستور نام شی و همچنین عبارت بولین برای فعال یا غیرفعال کردن آن می‌باشد.

مقدار بازگشتی **ندارد** .

Plugin.SetPos: با این کد می‌توانید شی پلاگین را به محل دلخواه در صفحه منتقل نمایید.ورودی‌های این کد نام شی، عددی برای فاصله از چپ و عدد دیگری برای فاصله از بالا می‌باشد.

مقدار بازگشتی **ندارد** .

Plugin.SetSize: با این دستور می‌توانید طول و عرض شی پلاگین را به اندازه دلخواه خود تغییر دهید. ورودی‌های این کد نام شی، عددی برای عرض و عددی دیگر برای طول شی می‌باشد.

مقدار بازگشتی **ندارد** .

Plugin.SetVisible: کنترل نمایش یا عدم نمایش شی پلاگین با این دستور امکان پذیر است. ورودی‌های کد، نام شی و عبارتی بولین برای فعال یا غیر فعال کردن نمایش شی می‌باشد.

مقدار بازگشتی **ندارد** .

Progress

این دستور برای کار با شی Progress به کار می‌رود که دستورات زیر مجموعه‌ی آن به شرح زیر می‌باشند:

Progress.GetCurrentPos: با استفاده از این دستور می‌توانید موقعیت فعلی (میزان پیشرفت) پروگرس را به دست آورید.نام شی ورودی دستور است.

مقدار بازگشتی عددی است نمایش دهنده میزان پیشرفت.

Progress.GetPos: با استفاده از این دستور می‌توانید موقعیت قرارگیری شی Progress را بدست آورید .این موقعیت نسبت به گوشه‌ی بالا و سمت چپ محاسبه می‌شود.ورودی این دستور نام شی می‌باشد.

مقدار بازگشتی یک آرایه با متغیرهای X و Y می‌باشد.

Progress.GetProperties: این دستور امکان به دست آوردن خصوصیات یک شی Progress را به شما می‌دهد .نام شی ورودی دستور است.

مقدار بازگشتی آرایه ای با متغیرهای زیر است :

ObjectName: نام شی

Text: متن نمایش داده شده

FontName: نام فونت

FontSize: اندازه فونت

FontStrikeout: فعال یا غیر فعال بودن strikeout در تنظیمات فونت (بولین)

FontUnderline: فعال یا غیر فعال بودن underline در تنظیمات فونت (بولین)

FontAntiAlias: فعال یا غیر فعال بودن anti alias در تنظیمات فونت (بولین)

FontItalic: فعال یا غیر فعال بودن italic در تنظیمات فونت (بولین)

FontWeight: نوع نوشته (تیرگی متن) به صورت آرایه ای متشکل از DONTCARE ,THIN ,EXTRALIGHT ,LIGHT ,NORMAL ,MEDIUM, SEMIBOLD, BOLD, EXTRABOLD و HEAVY .

FontScript: نوع اسکریپت فونت به صورت آرایه ای متشکل از , ANSI , BALTIC , CHINESEBIG5 , DEFAULT , EASTEUROPE , GB2312 , GREEK , HANGUL , MAC , OEM , RUSSIAN , SHIFTJIS ,SYMBOL و TURKISH .

Style: استیل (ظاهر) شی (BARS یا SMOOTH)

Orientation: نوع پر شدن شی (افقی یا عمودی)

MinRange: موقعیت ابتدایی

MaxRange: موقعیت انتهایی

Step: میزان پرش (برای استفاده توسط Progress.StepIt)

UseCustomColors: وضعیت استفاده از رنگ اختصاصی (بولین)

BarColor: شماره رنگ قسمت پیش رونده

BackgroundColor: شماره رنگ پس زمینه

TextColor: شماره رنگ متن

XPStyle: وضعیت استیل XP (بولین)

Enabled: فعال یا عدم فعال بودن (بولین)

Visible: دیده شدن یا عدم دیده شدن (بولین)

X: فاصله از سمت چپ صفحه

Y: فاصله از سمت راست صفحه

Width: عرض (پیکسل)

Height: طول (پیکسل)

ResizeLeft: قابلیت تغییر اندازه از سمت چپ در زمان تغییر اندازه پروژه (بولین)

ResizeRight: قابلیت تغییر اندازه از سمت راست در زمان تغییر اندازه پروژه (بولین)

ResizeTop: قابلیت تغییر اندازه از بالا در زمان تغییر اندازه پروژه (بولین)

ResizeBottom: قابلیت تغییر اندازه از پایین در زمان تغییر اندازه پروژه (بولین)

WindowHandle: شماره هندل شی

Progress.GetRange: این دستور محدوده عملکرد Progress را باز می‌گرداند .ورودی دستور نام شی می‌باشد.

مقدار بازگشتی آرایه ای است با دو عضو Begin و End .

Progress.GetSize: نمایش اندازه شی Progress بر حسب پیکسل با این دستور امکان پذیر است.ورودی دستور نام شی می‌باشد.

مقدار بازگشتی دستور، آرایه ای 2 عضوی با اعضای "Width" و "Height" می‌باشد.

Progress.GetText: با استفاده از این دستور می‌توان متن موجود در Progress را به دست آورد.نام شی ورودی دستور است.

مقدار بازگشتی رشته ای حاوی متن داخل نوار می‌باشد.

Progress.IsEnabled: با استفاده از این دستور می‌توانید وضعیت فعال یا غیرفعال بودن شی Progress را بدست آورید. مقدار ورودی نام شی می‌باشد.

مقدار بازگشتی Trueبه معنای فعال بودن و False به معنای غیرفعال بودن شی Progress می‌باشد.

Progress.IsVisible: با استفاده از این دستور می‌توانید نمایش یا عدم شیProgress را بدست آورید. مقدار ورودی نام شی است.

مقدار بازگشتی Trueبه معنای نمایش یا False به معنای عدم نمایش شی Progress می‌باشد.

Progress.SetCurrentPos: با استفاده از این دستور می‌توانید محل جدیدی برای نوار پیش رونده تعیین نمایید. نام شی و عدد محل مورد نظر ورودی‌های دستور هستند.

مقدار بازگشتی **ندارد** .

Progress.SetEnabled: با این دستور می‌توانید Progress را فعال یا غیر فعال کنید.ورودی دستور نام شی و همچنین عبارت بولین برای فعال یا غیرفعال کردن آن می‌باشد.

مقدار بازگشتی **ندارد** .

Progress.SetPos: با این کد می‌توانید برچسب را به محل دلخواه در صفحه منتقل نمایید.ورودی‌های این کد نام شی، عددی برای جای گیری در قسمت X (فاصله از چپ) و عدد دیگری برای جای گیری در Y (فاصله از بالا می‌باشد).

مقدار بازگشتی **ندارد** .

Progress.SetProperties: با استفاده از این دستور می‌توانید خصوصیات جدید برای شی Label تعیین نمایید. نام شی و آرایه خصوصیات (که در Progress.GetProperties توضیح داده شده) ورودی‌های دستور هستند.

مقدار بازگشتی **ندارد** .

Progress.SetRange: با این دستور می‌توانید محدوده عملکرد Progress را تعیین نمایید. نام شی، عدد ابتدا و عدد انتها ورودی‌های دستور هستند.

مقدار بازگشتی **ندارد** .

Progress.SetSize: با این دستور می‌توانید طول و عرضProgress را به اندازه دلخواه خود تغییر دهید.ورودی‌های این کد نام شی، عددی برای عرض و عددی دیگر برای طول شی می‌باشد.

مقدار بازگشتی **ندارد** .

Progress.SetStep: با استفاده از این دستور می‌توانید گام‌های (اندازه پیشرفت در) نوار پیش‌رونده را تعیین نمایید.نام شی و مقدار پرش ورودی‌های دستور هستند.

مقدار بازگشتی **ندارد** .

Progress.SetText: تغییر متن داخل Progress به متن دلخواه با این دستور امکان پذیر است .نام شی و متن مورد نظر ورودی‌های دستورند.

مقدار بازگشتی **ندارد** .

Progress.SetVisible: کنترل نمایش یا عدم نمایش پروگرس با این دستور امکان پذیر است. ورودی‌های کد، نام شی و عبارتی بولین برای فعال یا غیر فعال کردن نمایش شی می‌باشد.

مقدار بازگشتی **ندارد** .

Progress.StepIt: با این دستور می‌توانید نوار را به اندازه یک گام (گام‌ها را به وسیله Progress.SetStep می‌توانید تنظیم نمایید.) پیش ببرید.نام شی ورودی دستور است.

مقدار بازگشتی **ندارد** .

RadioButton

این تابع برای کار با شی RadioButton به کار گرفته می‌شود و دارای دستورات زیر مجموعه ای به شرح زیر می‌باشد:

RadioButton.GetChecked: با استفاده از این دستور می‌توانید علامت دار یا علامت دار نبودن شی رادیویی را به دست آورید.ورودی این دستور نام شی می‌باشد.

مقدار بازگشتی عبارتی است بولین؛ true به معنای علامت دار و false به معنای عدم علامت دار بودن شی می‌باشد.

RadioButton.GetPos: با استفاده از این دستور می‌توانید موقعیت قرارگیری شی رادیویی را بدست آورید .این موقعیت نسبت به گوشه‌ی بالا و سمت چپ محاسبه می‌شود.ورودی این تابع نام شی می‌باشد.

مقدار بازگشتی یک آرایه با متغیرهای X و Y می‌باشد.

RadioButton.GetProperties: با استفاده از این دستور می‌توانید خصوصیات شی Label را به دست آورید. نام شی ورودی دستور است .

مقدار بازگشتی آرایه ای با متغیرهای زیر است :

ObjectName: نام شی

GroupID: شماره ID

Value: مقدار Value

Checked: نوع علامت (علامت دار بودن یا نبودن)

Text: متن نمایش داده شده

FontName: نام فونت استفاده شده

FontSize: اندازه فونت

FontStrikeout: فعال یا غیر فعال بودن strikeout در تنظیمات فونت (بولین)

FontUnderline: فعال یا غیر فعال بودن underline در تنظیمات فونت (بولین)

FontAntiAlias: فعال یا غیر فعال بودن anti alias در تنظیمات فونت (بولین)

FontItalic: فعال یا غیر فعال بودن italic در تنظیمات فونت (بولین)

FontWeight: نوع نوشته (تیرگی متن) به صورت آرایه ای متشکل از DONTCARE, THIN, EXTRALIGHT, LIGHT, NORMAL, MEDIUM, SEMIBOLD, BOLD, EXTRABOLD و HEAVY.

FontScript: نوع اسکریپت فونت به صورت آرایه ای متشکل از ANSI , BALTIC , CHINESEBIG5 , DEFAULT , EASTEUROPE , GB2312 , GREEK , HANGUL , MAC , OEM , RUSSIAN , SHIFTJIS ,SYMBOL و TURKISH .

TextAlignment: چینش متن

ButtonAlignment: محل قرار گیری دکمه رادیویی

ReadOrder: نوع نمایش متن

ColorNormal: رنگ متن در حالت Normal

ColorHighlight: رنگ متن در حالت Highlight (وقتی ماوس روی آن قرار دارد)

ColorDisabled: رنگ متن در حالت غیرفعال

ColorDown: رنگ متن در حالت Down (وقتی بر روی آن کلیک می‌شود)

Enabled: فعال یا عدم فعال بودن (بولین)

Visible: دیده شدن یا عدم دیده شدن (بولین)

X: فاصله از سمت چپ صفحه

Y: فاصله از سمت راست صفحه

Width: عرض (پیکسل)

Height: طول (پیکسل)

TooltipText: متن نمایش دهنده در زمان قرارگیری نشانگر ماوس روی شی.

Cursor:عدد شکل نشانگر ماوس : ARROW , HAND , BLACK_ARROW , CROSSHAIR , EXPLORE , HELP , MAGNIFY , MEDIA , MONEY , NOTEPAD , PENCIL , PRINTER , SPEAKER یا UP_ARROW

ResizeLeft: قابلیت تغییر اندازه از سمت چپ در زمان تغییر اندازه پروژه (بولین)

ResizeRight: قابلیت تغییر اندازه از سمت راست در زمان تغییر اندازه پروژه (بولین)

ResizeTop: قابلیت تغییر اندازه از بالا در زمان تغییر اندازه پروژه (بولین)

ResizeBottom: قابلیت تغییر اندازه از پایین در زمان تغییر اندازه پروژه (بولین)

HighlightSound: نوع صوت در هنگام قرارگیری نشانگر ماوس بر روی شی (NONE, STANDARD یا CUSTOM)

HighlightSoundFile: مسیر فایل صوتی پخش شونده در هنگام قرارگیری نشانگر ماوس بر روی شی

ClickSound: نوع صوت در هنگام کلیک بر روی شی (NONE, STANDARD یا CUSTOM)

ClickSoundFile: مسیر فایل صوتی پخش شونده در هنگام کلیک بر روی شی

RadioButton.GetSize: نمایش اندازه شی بر حسب پیکسل با این دستور امکان پذیر است. ورودی دستور نام شی رادیویی است.

مقدار بازگشتی دستور، آرایه ای 2 عضوی با اعضای "Width" و "Height" می‌باشد.

RadioButton.GetText: با استفاده از این دستور می‌توانید متن موجود در شی را به دست آورید.نام شی، ورودی دستور است.

مقدار بازگشتی رشته ای حاوی متن داخل شی می‌باشد.

RadioButton.IsEnabled: با استفاده از این دستور می‌توانید وضعیت فعال یا غیرفعال بودن شی رادیویی را بدست آورید. مقدار ورودی نام شی می‌باشد.

مقدار بازگشتی Trueبه معنای فعال بودن و False به معنای غیرفعال بودن شی رادیویی می‌باشد.

RadioButton.IsVisible: با استفاده از این دستور می‌توانید نمایش یا عدم نمایش شی رادیویی در صفحه را بدست آورید. مقدار ورودی نام شی است.

مقدار بازگشتی Trueبه معنای نمایش یا False به معنای عدم نمایش می‌باشد.

RadioButton.SetChecked: با استفاده از این دستور می‌توانید یک دکمه رادیویی را علامت دار کنید یا علامت از حالت علامت دار بودن خارج نمایید. مقدار وردی نام شی و متغیری بولین (True برای علامت دار کردن و False برای از برداشتن علامت است)

مقدار بازگشتی **ندارد** .

RadioButton.SetEnabled: با این دستور می‌توانید دکمه رادیویی را فعال یا غیر فعال کنید.ورودی دستور نام شی و همچنین عبارت بولین برای فعال یا غیرفعال کردن آن می‌باشد.

مقدار بازگشتی **ندارد** .

RadioButton.SetPos: با این کد می‌توانید برچسب را به محل دلخواه در صفحه منتقل نمایید.ورودی‌های این کد نام شی، عددی برای جای گیری در قسمت X (فاصله از سمت چپ صفحه) و عدد دیگری برای جای گیری در Y (فاصله از بالای صفحه می‌باشد).

مقدار بازگشتی **ندارد** .

RadioButton.SetProperties: با استفاده از این دستور می‌توانید خصوصیات جدید برای شی تعیین نمایید. نام شی و آرایه خصوصیات (که در RadioButton.GetProperties توضیح داده شده) ورودی‌های دستور هستند.

مقدار بازگشتی **ندارد** .

RadioButton.SetSize: با این دستور می‌توانید طول و عرض شی رادیویی را به اندازه دلخواه خود تغییر دهید.ورودی‌های این کد نام شی، عددی برای عرض و عددی دیگر برای طول شی می‌باشد.

مقدار بازگشتی **ندارد** .

RadioButton.SetText: تغییر متن شی رادیویی با این دستور امکان پذیر است .نام شی و متن مورد نظر ورودی‌های دستور هستند.

مقدار بازگشتی **ندارد** .

RadioButton.SetVisible: کنترل نمایش یا عدم نمایش دکمه رادیویی با این دستور امکان پذیر است. ورودی‌های کد، نام شی و عبارتی بولین برای فعال یا غیر فعال کردن نمایش شی می‌باشد.

مقدار بازگشتی **ندارد** .

Registry

تابع Registry برای کار با رجیستری ویندوز به کار می‌رود و دارای زیر مجموعه‌ی دستوراتی به شرح زیر می‌باشد:

Registry.CreateKey: با استفاده از این دستور می‌توانید یک کلید ("key") در رجیستری ایجاد نمایید. مقدار ورودی شماره کلید اصلی برای ایجاد کلید در آن و نام کلید می‌باشد.

REG_NONE , REG_SZ , REG_EXPAND_SZ , REG_BINARY , REG_DWORD , REG_DWORD_LITTLE_ENDIAN , REG_DWORD_BIG_ENDIAN , REG_LINK , REG_MULTI_SZ , REG_RESOURCE_LIST , REG_FULL_RESOURCE_DESCRIPTOR , REG_RESOURCE_REQUIREMENTS_LIST

مقدار بازگشتی **ندارد** .

مثال:

```
Registry.CreateKey(HKEY_LOCAL_MACHINE, "Software\\My Application");
```

Registry.DeleteKey: با استفاده از این دستور هم می‌توانید یک کلید ("key") را از رجیستری حذف نمایید. مقدار ورودی، شماره کلید و نام کلید می‌باشد.

مقدار بازگشتی **ندارد** .

Registry.DeleteValue: با استفاده از این دستور می‌توانید یک مقدار را از داخل یک کلید حذف نمایید. شماره (یا نام) کلید اصلی، نام کلید و مقدار مورد دلخواه ورودی‌های دستور هستند .

مقدار بازگشتی **ندارد** .

Registry.DoesKeyExist: این دستور امکان کنترل وجود یا عدم وجود یک کلید را به شما می‌دهد. شماره کلید اصلی و نام کلید مورد نظر ورودی‌های دستورند.

مقدار بازگشتی متغیری بولین است.

Registry.GetAccess: با استفاده از این دستور می‌توانید میزان دسترسی به رجیستری را کنترل نمایید. ورودی‌های دستور، نام کلید اصلی، نام کلید و نوع دسترسی است:

ACCESS_READ , ACCESS_WRITE , ACCESS_ENUMERATE,ACCESS_ALL

مقدار بازگشتی متغیری بولین است.

Registry.GetKeyNames: این دستور نام کلید های زیرشاخه را بر می‌گرداند. ورودی‌های دستور نام کلید اصلی و نام کلیدی که نیاز به، به دست آوردن کلید های آن دارید هستند.

مقدار بازگشتی آرایه ای است. در صورت عدم وجود کلید یا بروز خطا nil بازگردانده می‌شود.

Registry.GetValue: با استفاده از این دستور می‌توانید مقدار خاصی را از کلید مورد نظر به دست آورید .نام کلید اصلی، نام کلید، مقدار مورد نظر ورودی‌های مورد نظر هستند.

مقدار بازگشتی رشته می‌باشد.

Registry.GetValueNames: این دستور مقادیر موجود در یک کلید را باز می‌گرداند.ورودی‌های دستور شماره کلید اصلی و نام کلید است.

مقدار بازگشتی آرایه ای است متشکل از نام مقادیر موجود در کلید.در صورت عدم وجود مقدار یا بروز خطا nil بازگردانده می‌شود.

Registry.GetValueType: نوع مقادیر موجود با این دستور به دست می‌آید.ورودی‌های دستور شماره کلید اصلی، نام کلید و مقدار مورد نظر است.

مقدار بازگشتی عددی است نمایش دهنده نوع مقدار:

REG_NONE , REG_SZ , REG_EXPAND_SZ , REG_BINARY , REG_DWORD , REG_DWORD_LITTLE_ENDIAN , REG_DWORD_BIG_ENDIAN , REG_LINK , REG_MULTI_SZ , REG_RESOURCE_LIST , REG_FULL_RESOURCE_DESCRIPTOR , REG_RESOURCE_REQUIREMENTS_LIST

Registry.SetValue: با استفاده از این دستور می‌توانید مقداری را در کلید مورد نظر ایجاد کرده یا تغییر دهید.شماره کلید اصلی، نام کلید(ها)، نام مقدار، عبارت مورد نظر برای ذخیره و نوع مقدار ورودی‌های دستور هستند.

مقدار بازگشتی **ندارد** .

مثال:

```
Registry.SetValue(HKEY_LOCAL_MACHINE, "Software\\My Application",
"MyValue", "My Data", REG_SZ);
```

RichText

برای کار با شی RichText از این تابع استفاده می‌کنیم. این تابع دارای مجموعه‌ی دستورات زیر است.

RichText.CanPaste: این دستور وضعیت فعال بودن قابلیت چسباندن (Paste) کردن متن در شی RichText را نمایش می‌دهد. نام شی ورودی دستور است.

مقدار بازگشتی متغیری است بولین .

RichText.CanUndo: این دستور وضعیت فعال بودن بازگشت تغییرات به مرحله قبل را نشان می‌دهد.ورودی دستور نام شی است.

مقدار بازگشتی متغیری است بولین .

RichText.Copy: این دستور امکان کپی متن انتخاب شده به "کلیپ بورد" را مهیا می‌کند. ورودی دستور نام شی است .

مقدار بازگشتی رشته است.

RichText.Cut: این دستور امکان بریدن (Cut) متن انتخاب شده به "کلیپ بورد" را مهیا می‌کند. ورودی دستور نام شی است .

مقدار بازگشتی رشته است.

RichText.Delete: این دستور متن انتخاب شده در شی را حذف می‌کند. ورودی دستور نام شی است .

مقدار بازگشتی **ندارد** .

RichText.EmptyUndoBuffer: با استفاده از این دستور می‌توانید حافظه مربوط به عملیات Undo را خالی کنید .نام شی ورودی دستور است.

مقدار بازگشتی **ندارد** .

RichText.FindText:این دستور جستجو در متن RichText را برای شما امکان پذیر می‌کند.نام شی، رشته برای جستجو در شی، ابتدای محل جستجو (1 به معنای ابتدای متن)

انتهای جستجو (1- به معنای انتهای متن)، وضعیت حساس بودن به بزرگی و کوچکی حروف و وضعیت جستجو(جستجوی عین عبارت یا عبارات همسان) ورودی‌های دستور هستند.

مقدار بازگشتی آرایه ای است با متغیرهای Start و End (ابتدای عبارت یافت شده و انتهای آن .) در صورت پیدا نشدن متن nil بازگردانده می‌شود.

RichText.GetLine: این دستور با گرفتن نام شی و شماره خط به عنوان ورودی، متن آن را باز می‌گرداند.

مقدار بازگشتی رشته ای است حاوی متن خطا مورد نظر.

RichText.GetParagraphFormat: این دستور فرمت پاراگراف انتخاب شده را باز می‌گرداند. نام شی RichText و وضعیت فعال بودن نمایش فرمت قسمت انتخاب شده یا کل متن ورودی‌های دستور هستند.

مقدار بازگشتی آرایه ای است با متغیرهای زیر :

Bulleted, StartIndent, RightIndent, Offset, Alignment, Tabs, SpaceBefore, SpaceAfter, LineSpacing, LineSpacingRule, BorderSpace ,BorderWidth , Borders

RichText.GetPos: با استفاده از این دستور می‌توانید موقعیت قرارگیری شی RichText را بدست آورید .ورودی این تابع نام شی می‌باشد.

مقدار بازگشتی یک آرایه با متغیرهای X و Y می‌باشد.

RichText.GetProperties: این دستور خصوصیات شی Rich Text را در اختیار شما قرار می‌دهد.نام شی ورودی تابع است.

مقدار بازگشتی آرایه ای شامل موارد زیر می‌باشد :

ObjectName: نام شی

Text: متن نمایش داده شده

RawRTF:Raw متن (فرمت داخلی)

VScrollbar: وضعیت فعال بودن اسکرول افقی (بولین)

HScrollbar: وضعیت فعال بودن اسکرول عمودی (بولین)

Transparent: وضعیت شفاف بودن پس زمینه شی

BackgroundColor: شماره رنگ پس زمینه

Border: نوع Border (بدون Border یا SUNKEN)

ReadOrder: نوع نمایش متن (استاندارد یا راست به چپ)

ReadOnly: وضعیت قابل نوشتن روی متن (true فقط خواندن متن)

AutoDetectURL: تشخیص خودکار آدرس وب

ShowContextMenu: وضعیت نمایش منو راست کلیک

Enabled: فعال یا عدم فعال بودن (بولین)

Visible: دیده شدن یا عدم دیده شدن (بولین)

X: فاصله از سمت چپ صفحه

Y: فاصله از سمت راست صفحه

Width: عرض (پیکسل)

Height: طول (پیکسل)

TooltipText: متن نمایش دهنده در زمان قرارگیری نشانگر ماوس روی شی.

Cursor: عدد شکل نشانگر ماوس و یا نام ثابت آن:

ARROW , HAND , BLACK_ARROW , CROSSHAIR , EXPLORE , HELP , MAGNIFY , MEDIA , MONEY , NOTEPAD , PENCIL , PRINTER , SPEAKER یا UP_ARROW

ResizeLeft: قابلیت تغییر اندازه از سمت چپ در زمان تغییر اندازه پروژه (بولین)

ResizeRight: قابلیت تغییر اندازه از سمت راست در زمان تغییر اندازه پروژه (بولین)

ResizeTop: قابلیت تغییر اندازه از بالا در زمان تغییر اندازه پروژه (بولین)

ResizeBottom: قابلیت تغییر اندازه از پایین در زمان تغییر اندازه پروژه (بولین)

WindowHandle: شماره هندل شی

RichText.GetSelection: این دستور امکان به دست آوردن محل متن انتخاب شده را به شما می‌دهد.نام شی ورودی دستور است.

مقدار بازگشتی آرایه ای است با دو متغیر Start و End (محل ابتدا و انتهای متن انتخاب شده)

RichText.GetSelectionFormat: این دستور فرمت عبارت انتخاب شده را باز می‌گرداند.نام شی ورودی دستور است.

مقدار بازگشتی آرایه ای است با متغیرهای بولین، عددی و رشته ای.

بولین :

AllCaps , AutoBackColor , AutoColor , Bold , Disabled , Emboss , Hidden , Imprint , Italic , Link , Outline , Protected , Revised , Shadow , SmallCaps , StrikeOut, SubScript, SuperScript

عددی:

Underline, Height , YOffset , TextColor , CharacterSet , PitchAndFamily , Weight , Spacing , BackColor , LCID , Kerning , Style , UnderlineType , Animation , RevAuthor

رشته

FaceName:

RichText.GetSize: نمایش اندازه شی RichText بر حسب پیکسل با این دستور امکان پذیر است.ورودی دستور نام شی می‌باشد.

مقدار بازگشتی دستور، آرایه ای 2 عضوی با اعضای "Width" و "Height" می‌باشد.

RichText.GetText: با استفاده از این دستور می‌توانید متن موجود در RichText را به دست آورید. نام شی و وضعیت RAW ورودی‌های دستور هستند.

مقدار بازگشتی رشته ای حاوی متن داخل شی می‌باشد.

RichText.GetTextLength: با ورود نام شی، طول متن داخل آن را به شما می‌دهد.

مقدار بازگشتی تعداد کاراکترهای موجود در شی است.

RichText.IsEnabled: با استفاده از این دستور می‌توانید وضعیت فعال بودن شی را بدست آورید. مقدار ورودی نام شی می‌باشد.

مقدار بازگشتی Trueبه معنای فعال بودن و False به معنای غیرفعال بودن شی RichText می‌باشد.

RichText.IsVisible: با استفاده از این دستور می‌توانید نمایش یا عدم نمایش RichText را بدست آورید. مقدار ورودی نام شی است.

مقدار بازگشتی Trueبه معنای نمایش یا false به معنای عدم نمایش RichText است.

RichText.LoadFromFile: این دستور فایل با مسیر داده شده را در شی ورودی بارگذاری می‌کند.

مقدار بازگشتی **ندارد** .

RichText.Paste:با استفاده از این دستور می‌توانید متن موجود در کلیپ بورد را در شی قرار دهید.

مقدار بازگشتی **ندارد** .

RichText.SaveToFile: این دستور امکان ذخیره متن داخل شی را در یک فایل RTF می‌دهد. نام شی و مسیر فایل برای ذخیره ورودی دستور است .

مقدار بازگشتی **ندارد** .

RichText.ScrollLines: با این دستور می‌توانید به چند خط بعد یا قبل بروید .نام شی و تعداد خطوط (منفی به معنای حرکت به سوی بالا) ورودی‌های دستور هستند.

مقدار بازگشتی **ندارد** .

RichText.ScrollToLine: این دستور امکان رفتن به خط خاص را در اختیار می‌گذارد.نام شی و خط مورد نظر ورودی‌های دستور هستند .

مقدار بازگشتی **ندارد** .

RichText.SetEnabled: با این دستور می‌توانید شی RichText را فعال یا غیر فعال کنید.ورودی دستور نام شی و همچنین عبارت بولین برای فعال یا غیرفعال کردن آن می‌باشد.

مقدار بازگشتی **ندارد** .

RichText.SetParagraphFormat: با استفاده از این دستور می‌توانید فرمت پاراگراف متن انتخاب شده در شی را تغییر دهید. نام شی و آرایه مربوط (با متغیرهای زیر) ورودی دستور هستند.

متغیرهای آرایه:

Bulleted, StartIndent, RightIndent, Offset, Alignment,Tabs, SpaceBefore , SpaceAfter , LineSpacing , LineSpacingRule , BorderSpace, BorderWidth, Borders

مقدار بازگشتی **ندارد** .

RichText.SetPos: با این کد می‌توانید شی را به محل دلخواه در صفحه منتقل نمایید.ورودی‌های این کد نام شی، عددی برای فاصله از چپ و عدد دیگری برای فاصله از بالا می‌باشد.

مقدار بازگشتی **ندارد** .

RichText.SetProperties: با استفاده از این دستور می‌توانید خصوصیات جدید برای شی پاراگراف تعیین نمایید. نام شی و آرایه خصوصیات (که در RichText.GetProperties توضیح داده شده) ورودی‌های دستور هستند.

مقدار بازگشتی **ندارد** .

RichText.SetSelection: با استفاده از این دستور می‌توانید قسمتی از متن داخل شی را به انتخاب در آورید.نام شی، ابتدای محل انتخاب و انتهای آن ورودی‌ها هستند.

مقدار بازگشتی **ندارد** .

RichText.SetSelectionFormat: با استفاده از این دستور می‌توانید فرمت متن انتخاب شده را تغییر دهید.نام شی، آرایه (با متغیرهای زیر) و وضعیت اعمال تغییرات (کل کلمات در ناحیه انتخاب شده یا فقط کاراکترهای انتخاب شده) ورودی دستور هستند.

بولین :

AllCaps , AutoBackColor , AutoColor , Bold , Disabled , Emboss , Hidden , Imprint , Italic , Link , Outline , Protected , Revised , Shadow , SmallCaps , StrikeOut, SubScript, SuperScript

عددی:

Underline, Height , YOffset , TextColor , CharacterSet , PitchAndFamily , Weight , Spacing , BackColor , LCID , Kerning , Style , UnderlineType , Animation , RevAuthor

رشته:

FaceName

مقدار بازگشتی **ندارد** .

RichText.SetSize: با این دستور می‌توانید طول و عرض شی RichText را به اندازه دلخواه خود تغییر دهید. ورودی‌های این کد نام شی، عددی برای عرض و عددی دیگر برای طول شی می‌باشد.

مقدار بازگشتی **ندارد** .

RichText.SetText: تغییر متن شی RichText به متن دلخواه با این دستور امکان پذیر است .نام شی، متن مورد نظر و وضعیت RawRTF ورودی‌های دستورند.

مقدار بازگشتی **ندارد** .

RichText.SetVisible: کنترل نمایش یا عدم نمایش RichText با این دستور امکان پذیر است. ورودی‌های کد، نام شی و عبارتی بولین برای فعال یا غیر فعال کردن نمایش شی می‌باشد.

مقدار بازگشتی **ندارد** .

RichText.Undo: با استفاده از این دستور می‌توانید تغییرات را به یک مرحله قبل بازگردانید.ورودی نام شی می‌باشد.

مقدار بازگشتی **ندارد** .

Service

با استفاده از تابع Service می‌توانید با کار سرویس‌های ویندوز بپردازید توجه داشته باشید که کار با این دستورات به آگاهی از سرویس‌های ویندوز و عملکرد آن‌ها نیاز دارد و حتی ممکن است بی اطلاعی شما از این موضوع موجب صدمه زدن به سیستم عامل شود پس در استفاده از

این تابع با احتیاط عمل کنید. این دستور دارای زیر مجموعه ای از دستورات می‌باشد که شرح آن‌ها در زیر آمده است:

Service.Continue: با استفاده از دستور می‌توانید سرویسی را که قبلاً متوقف کرده بودید، به راه بیندازید (سرویس‌ها از طریق Windows Task Manager و سربرگ Services قابل مشاهده‌اند.) نام نمایش داده شده برای سرویس و کلید آن ورودی‌های دستور هستند.

مقدار بازگشتی **ندارد** .

Service.Create: با استفاده از این دستور می‌توانید یک سرویس جدید را ایجاد و اجرا نمایید. ورودی‌های دستور عبارتند از: مسیر فایل مربوط به سرویس، نام برای نمایش، نام کلید (حداکثر 256 کاراکتر)، نوع سرویس، مقدار بولین برای اجازه اجرا در اکانت LocalSystem ، نوع شروع به کار سرویس، نوع خطا در صورت عدم اجرا، نام گروه، تگ گروه، نام سرویس‌هایی که نیاز است قبل از سرویس ایجاد شده اجرا شوند، نام اکانتی که سرویس باید در آن اجرا شود و کلمه عبور .

مقدار بازگشتی **ندارد** .

نوع سرویس:

WIN32_OWN_PROCESS, WIN32_SHARE_PROCESS, KERNEL_DRIVER یا LE_SYSTEM_DRIVER

نوع شروع به کار سرویس:

BOOT_START, SYSTEM_START, AUTO_START, DEMAND_START یا DISABLED

نوع خطا:

ERROR_IGNORE, ERROR_NORMAL, ERROR_SEVERE یا ERROR_CRITICAL

Service.Delete: با استفاده از این دستور می‌توانید یک سرویس را حذف نمایید. نام نمایش داده شده برای سرویس و کلید آن ورودی‌های دستور هستند.

مقدار بازگشتی **ندارد** .

Service.Pause: این دستور امکان متوقف کردن سرویس را به شما می‌دهد. نام نمایش داده شده برای سرویس و کلید آن ورودی‌های دستور هستند.

مقدار بازگشتی **ندارد** .

Service.Query: نوع فعالیت سرویس در سیستم با این دستور به دست می‌آید. نام نمایش داده شده برای سرویس و کلید ورودی‌های دستورند.

مقدار بازگشتی عددی است نمایش دهنده نوع فعالیت سرویس

SERVICE_NOT_FOUND, SERVICE_STOPPED,
SERVICE_START_PENDING, SERVICE_STOP_PENDING,
SERVICE_RUNNING, SERVICE_CONTINUE_PENDING,
SERVICE_PAUSE_PENDING, SERVICE_PAUSED,
SERVICE_ERROR

Service.Start: شروع به کار یک سرویس با این دستور ممکن است .ورودی‌های دستور، نام نمایش داده شده برای سرویس، کلید آن و آرگومان‌های مورد نیاز برای شروع به کار هستند.

مقدار بازگشتی **ندارد** .

Service.Stop: با استفاده از این می‌توانید یک سرویس را به طور کامل متوقف کنید. .) نام نمایش داده شده برای سرویس، کلید آن و بیشترین زمان برای انتظار توقف سرویس ورودی‌ها هستند.

مقدار بازگشتی **ندارد**.

Shell

از این تابع برای انجام کارهایی نظر ایجاد میانبر(Shortcut) برای یک فایل اجرایی، حذف میانبر، اجرای فایل با روشی خاص و ... استفاده می‌شود. دستورات زیر مجموعه‌ی ان تابع به شرح زیر است:

Shell.CreateShortcut: با استفاده از این دستور می‌توانید میانبر از فایل دلخواه خود ایجاد نمایید.مقادیر ورودی شامل موارد زیر هستند.

محل قرارگیری میانبر، توضیحات، مسیر کامل فایل مورد نظر، آرگومان‌های فایل، شاخه ای که فایل باید روی آن کار کند، مسیر آیکون، شماره آیکون موجود در فایل، نوع نمایش صفحه در هنگام اجرای فایل، و آرایه ای حاوی کلیدهای میانبر برای اجرا .

مقدار بازگشتی **ندارد** .

Shell.DeleteShortcut: حذف میانبر با این دستور امکان پذیر است .شاخه و توضیحات میانبر ورودی‌های دستور هستند.

مقدار بازگشتی **ندارد** .

Shell.Execute: با استفاده از این دستور عملیات خاصی بر روی فایل یا فولدر انجام می‌شود. مسیر فایل (یا فولدر)، نوع عملیات (باز کردن معمولی، باز کردن در Windows Explorer، باز کردن با ویرایشگر و پرینت)، آرگومان مورد نیاز ، شاخه ای که فایل باید روی آن کار کند، نوع نمایش صفحه در هنگام اجرا، و مقدار بولین برای انتظار / عدم انتظار برای اتمام کار ورودی‌های دستور هستند.

مقدار بازگشتی عدد صفر می‌باشد در صورتی که WaitForReturn برابر false باشد و عدد بازگشتی انجام عملیات در صورت True بودن آن.

Shell.GetFolder: مسیر شاخه مورد نظرتان را با این دستور به دست آورید. مقدار ورودی نام شاخه می‌باشد :

FONTS , MYMUSIC , MYMUSIC_COMMON , MYPICTURES , MYPICTURES_COMMON , MYVIDEOS , MYVIDEOS_COMMON , DESKTOP , DESKTOP_COMMON , STARTMENU , STARTMENU_COMMON , STARTMENUPROGRAMS , STARTMENUPROGRAMS_COMMON , STARTUP , STARTUP_COMMON , COMMONFILES , PROGRAMFILES , MYDOCUMENTS , COMMON_DOCUMENTS , APPLICATIONDATA , APPLICATIONDATA_LOCAL , APPLICATIONDATA_COMMON

مقدار بازگشتی رشته ای حاوی مسیر شاخه است.

SlideShow

این تابع برای کار با شی SlideShow به کار می‌رود که دستورات زیر مجموعه آن به شرح زیر است:

SlideShow.AddSlide: با این دستور می‌توانید یک اسلاید به شی "اسلاید شو" اضافه نمایید .نام شی، مسیر تصویر و محل قرار گیری در میان اسلایدها (-1 آخرین اسلاید) ورودی‌های دستور هستند.

مقدار بازگشتی عدد مربوط به جایگاه اسلاید اضافه شده است.

SlideShow.DeleteSlide: با وارد کردن نام شی و شما ره اسلاید می‌توانید اسلاید مورد نظرتان را حذف نمایید.

مقدار بازگشتی **ندارد** .

SlideShow.FillFromFolder: با استفاده از این دستور می‌توانید تصاویر یک شاخه را به داخل "اسلاید شو" اضافه نمایید. نام شی، مسیر شاخه، مقدار بولین برای خالی کردن یا نکردن شی قبل از اضافه کردن تصاویر و مقداری بولین برای فعال یا غیرفعال کردن افزودن تصاویر از زیر شاخه‌ها ورودی‌های دستور هستند.

مقدار بازگشتی **ندارد** .

SlideShow.GetCurrentSlide: شماره اسلاید حاضر با این دستور به دست می‌آید. ورودی دستور نام شی است .

مقدار بازگشتی شماره اسلاید می‌باشد .

SlideShow.GetImagePath: این دستور مسیر تصویر یک اسلاید را باز می‌گرداند. نام شی و شماره اسلاید ورودی‌های دستورند.

مقدار بازگشتی رشته ای است حاوی مسیر تصویر .

SlideShow.GetPos: با استفاده از این دستور می‌توانید موقعیت قرارگیری شی را بدست آورید .ورودی این دستور نام شی "اسلاید شو" می‌باشد.

مقدار بازگشتی یک آرایه با متغیرهای X و Y می‌باشد.

SlideShow.GetProperties: با استفاده از این دستور می‌توانید خصوصیات شی l را به دست آورید. نام شی ورودی دستور است .

مقدار بازگشتی آرایه ای با متغیرهای زیر است :

ObjectName: نام شی

ImageFiles: آرایه ای شامل مسیر تصاویر

BackgroundStyle: نوع نمایش پس زمینه (SOLID یا TRANSPARENT)

BackgroundColor: شماره رنگ پس زمینه

BorderStyle: نوع نمایش Border

BorderColor: رنگ Border

Interval: مدت زمان نمایش هر اسلاید (میلی ثانیه)

AutoStart: نوع شروع اسلایدها (true اتوماتیک، false دستی)

ResizeMode: نوع تغییر اندازه تصاویر برای قرارگیری در شی

NoEnlarge: بولین برای نوع نمایش تصویر (وسط یا پر کردن کل شی)

Loop: تکرار اسلایدها (بولین)

Shuffle: نوع نمایش اسلایدها (true تصادفی، false متوالی)

Transitions: نمایش افکت (بولین)

Enabled: فعال یا عدم فعال بودن (بولین)

Visible: دیده شدن یا عدم دیده شدن (بولین)

X: فاصله از سمت چپ صفحه

Y: فاصله از سمت راست صفحه

Width: عرض (پیکسل)

Height: طول (پیکسل)

TooltipText: متن نمایش دهنده در زمان قرارگیری نشانگر ماوس روی شی.

Cursor:عدد شکل نشانگر ماوس : ARROW , HAND , BLACK_ARROW , CROSSHAIR , EXPLORE , HELP , MAGNIFY , MEDIA , MONEY , NOTEPAD , PENCIL , PRINTER , SPEAKER , یا UP_ARROW

ResizeLeft: قابلیت تغییر اندازه از سمت چپ در زمان تغییر اندازه پروژه (بولین)

ResizeRight: قابلیت تغییر اندازه از سمت راست در زمان تغییر اندازه پروژه (بولین)

ResizeTop: قابلیت تغییر اندازه از بالا در زمان تغییر اندازه پروژه (بولین)

ResizeBottom: قابلیت تغییر اندازه از پایین در زمان تغییر اندازه پروژه (بولین)

HighlightSound: نوع صوت در هنگام قرارگیری نشانگر ماوس بر روی شی (NONE, STANDARD یا CUSTOM)

HighlightSoundFile: مسیر فایل صوتی پخش شونده در هنگام قرارگیری نشانگر ماوس بر روی شی

ClickSound: نوع صوت در هنگام کلیک بر روی شی (NONE, STANDARD یا CUSTOM)

ClickSoundFile: مسیر فایل صوتی پخش شونده در هنگام کلیک بر روی شی

SlideShow.GetSize: نمایش اندازه شی بر حسب پیکسل با این دستور امکان پذیر است.ورودی دستور نام شی می‌باشد.

مقدار بازگشتی دستور، آرایه ای 2 عضوی با اعضای "Width" و "Height" می‌باشد.

SlideShow.GetSlideCount: این دستور تعداد اسلایدهای موجود را باز می‌گرداند. نام شی ورودی دستور است.

مقدار بازگشتی تعداد اسلایدها می‌باشد .

SlideShow.GoToSlide: با این دستور می‌توانید اسلاید خاص را به نمایش در آورید .نام شی و شماره اسلاید ورودی‌های دستور است.

مقدار بازگشتی **ندارد** .

SlideShow.IsEnabled: با استفاده از این دستور می‌توانید وضعیت فعال یا غیرفعال بودن شی را بدست آورید. مقدار ورودی نام شی می‌باشد.

مقدار بازگشتی true به معنای فعال بودن و false به معنای غیرفعال بودن شی می‌باشد.

SlideShow.IsVisible: با استفاده از این دستور می‌توانید وضعیت نمایش یا عدم نمایش شی را بدست آورید. مقدار ورودی نام شی است.

مقدار بازگشتی true به معنای نمایش یا false به معنای عدم نمایش می‌باشد.

SlideShow.Navigate: با استفاده از این دستور می‌توانید به اسلاید دلخواه پرش داشته باشد.نام شی و شماره نوع پرش (اولین صفحه، آخرین صفحه،صفحه بعد، صفحه قبل، صفحه قبلی، صفحه بعدی) ورودی‌های دستور هستند.

مقدار بازگشتی **ندارد** .

SlideShow.Pause: مکث در نمایش اسلایدها با این دستور امکان پذیر است. نام شی ورودی دستور می‌باشد.

مقدار بازگشتی **ندارد** .

SlideShow.Play: این دستور اسلایدها را نمایش می‌دهد. نام شی ورودی دستور می‌باشد.

مقدار بازگشتی **ندارد** .

SlideShow.SetEnabled: با این دستور می‌توانید شی "اسلاید شو" را فعال یا غیر فعال کنید.ورودی دستور نام شی و همچنین عبارت بولین برای فعال یا غیرفعال کردن آن می‌باشد.

مقدار بازگشتی **ندارد** .

SlideShow.SetPos: با این کد می‌توانید شی را به محل دلخواه در صفحه منتقل نمایید.ورودی‌های این کد نام شی، اعدادی برای مشخص کردن میزان فاصله از چپ و بالا می‌باشد.

مقدار بازگشتی **ندارد** .

SlideShow.SetProperties: با استفاده از این دستور می‌توانید خصوصیات جدیدی برای شی PDF اعمال کنید. نام شی و آرایه خصوصیات (که در SlideShow.GetProperties توضیح داده شده) ورودی‌های دستورند.

SlideShow.SetSize: با این دستور می‌توانید طول و عرض شی را به اندازه دلخواه خود تغییر دهید.ورودی‌های این کد نام شی، عددی برای عرض و عددی دیگر برای طول شی می‌باشد.

مقدار بازگشتی **ندارد** .

SlideShow.SetVisible: کنترل نمایش یا عدم نمایش شی با این دستور امکان پذیر است. ورودی‌های کد، نام شی و عبارتی بولین برای فعال یا غیر فعال کردن نمایش شی می‌باشد.

مقدار بازگشتی **ندارد** .

StatusDlg

این تابع برای نمایش یک کادر وضعیت برای انجام کارهایی همانند کپی، استخراج فایل فشرده و ... به کار می‌رود. دستورات زیر مجموعه‌ی این تابع به شرح زیر است:

StatusDlg.GetAutoSize: اندازه‌ی خودکار کادر StatusDlg (اتوماتیک یا غیر اتوماتیک) با این دستور به دست می‌آید.

مقدار بازگشتی، عبارتی است بولین. True در صورتی که بر روی اتوماتیک تنظیم شده باشد و بالعکس.

StatusDlg.GetMeterPos: با این دستور، میزان پیشرفت نوار StatusDlg را به دست آورید.

مقدار بازگشتی محل فعلی نوار می‌باشد. در صورت بروز خطا -1 بازگردانده می‌شود.

StatusDlg.Hide: این دستور StatusDlg را محو می‌کند.

مقدار بازگشتی **ندارد** .

StatusDlg.IsCancelled: این دستور وضعیت لغو شدن StatusDlg را کنترل می‌کند.

مقدار بازگشتی عبارتی است بولین نمایش دهنده وضعیت .

StatusDlg.SetAutoSize: اندازه‌ی خودکار کادر StatusDlg با این دستور تعیین می‌شود. ورودی مقداری است بولین برای تعیین اندازه.

مقدار بازگشتی **ندارد** .

StatusDlg.SetCancelled: این دستور امکان تعیین وضعیت دکمه لغو (کنسل) را در StatusDlg را می‌دهد. ورودی مقداری است بولین برای تعیین وضعیت.

مقدار بازگشتی **ندارد** .

StatusDlg.SetMessage: با این دستور می‌توانید پیغام خود را بر روی StatusDlg نمایش دهید. ورودی دستور رشته حاوی پیغام است.

مقدار بازگشتی **ندارد** .

StatusDlg.SetMeterPos: با این دستور می‌توانید موقعیت نوار StatusDlg را تعیین کنید .عدد محل مورد نظر ورودی دستور است.

مقدار بازگشتی **ندارد** .

StatusDlg.SetMeterRange:با این دستور می‌توانید محدوده نوار StatusDlg را تعیین کنید . ورودی‌ها عددی برای ابتدا (بزرگ‌تر یا مساوی 0)و عددی برای انتهای این محدوده (کوچک‌تر از 65534) می‌باشند.

مقدار بازگشتی **ندارد** .

StatusDlg.SetStatusText: می‌توانید متن وضعیت StatusDlg را با این دستور تغییر دهید. رشته حاوی متن ورودی دستور است.

مقدار بازگشتی **ندارد** .

StatusDlg.SetTitle: با این دستور هم می‌توانید عنوان StatusDlg را تغییر دهید. رشته حاوی عنوان ورودی دستور است.

مقدار بازگشتی **ندارد** .

StatusDlg.Show: این دستور StatusDlg را نمایش می‌دهد.

مقدار بازگشتی **ندارد** .

StatusDlg.ShowCancelButton: با استفاده از این دستور می‌توانید دکمه لغو را به نمایش در آورده یا حذف نمایید.مقدار بولین برای نمایش یا عدم نمایش دکمه ورودی دستور است.

مقدار بازگشتی **ندارد** .

StatusDlg.ShowProgressMeter: این دستور اجازه کنترل نمایش یا عدم نمایش نوار را توسط ورودی بولین به شما می‌دهد.

مقدار بازگشتی **ندارد** .

String

این تابع برای کار با متغیرهای رشته ای مورد استفاده قرار می‌گیرد. دستورات زیر مجموعه این تابع به شرح زیر است:

String.AbbreviateFilePath: این دستور امکان خلاصه کردن مسیر را می‌دهد .مسیر کامل و حداکثر تعداد کاراکترها پس از خلاصه کردن ورودی‌ها هستند.

مقدار بازگشتی رشته ای حاوی مسیر کوتاه شده است.

String.Asc: با استفاده از این دستور می‌توانید کد اسکی کاراکتر مورد نظر را به دست آورید .کاراکتر ورودی دستور است.

مقدار بازگشتی کد اسکی (عدد) می‌باشد.

String.Char: این دستور برعکس دستور بالا عمل می‌کند .یعنی کد اسکی را گرفته و کاراکتر را بازمی‌گرداند.

مقدار بازگشتی رشته حاوی کاراکتر مورد نظر است.

String.Compare: این دستور دو رشته را به عنوان ورودی گرفته و آن‌ها را با هم مقایسه می‌کند.

مقدار بازگشتی یکی از اعداد -1، 0 یا 1 است که به ترتیب نشان دهنده "کوچک‌تر بودن متن اول، برابر بودن متون و بزرگ‌تر بودن متن " می‌باشند.

قابل ذکر است که در مقایسه متون فارسی با انگلیسی، متون فارسی همیشه بزرگ‌ترند .

String.CompareFileVersions: این دستور دو نسخه (ورژن) ورودی را با هم مقایسه می‌کند.

مقدار بازگشتی یکی از اعداد -1، 0 یا 1 است که به ترتیب نشان دهنده "کوچک‌تر بودن ورژن اول، برابر بودن ورژن‌ها و کوچک‌تر بودن ورژن دوم " می‌باشند.

String.CompareNoCase: این دستور هم مانند String.Compare عمل می‌کند با تفاوت اینکه بزرگ یا کوچک بودن حروف در این دستور تأثیری ندارد.

String.Concat: این دستور دو رشته ورودی را با هم ادغام می‌کند. (رشته دومی به انتهای رشته اول متصل می‌شود)

مقدار بازگشتی رشته ای است حاصل از ادغام دو رشته ورودی .

String.Find: با استفاده از این دستور می‌توانید درون یک رشته کلمه ای را جستجو نمایید. رشته اصلی، رشته (کلمه) ای که نیاز به پیدا کردن آن در متن می‌باشد، محل شروع جستجو در رشته و حساسیت به حروف کوچک و بزرگ ورودی‌های دستور هستند.

مقدار بازگشتی موقعیت ابتدای کلمه در متن است .در صورت عدم وجود کلمه در متن مقدار -1 بازگردانده می‌شود .

String.GetFormattedSize: با استفاده از این دستور می‌توانید عدد حجم را بر حسب بایت وارد نموده و به کیلو بایت، مگابایت یا گیگابایت تبدیل نمایید.

مقدار بازگشتی، حجم بر حسب نوع درخواست شما می‌باشد.

String.Left: با استفاده از این دستور می‌توانید به مقدار مورد نیاز از کاراکترهای سمت چپ یک رشته جدا کنید. رشته و تعداد مورد نیاز شما ورودی‌های دستور هستند.

مقدار بازگشتی رشته ای جدید است.

String.Length: این دستور رشته ای را به عنوان ورودی گرفته و طول آن را باز می‌گرداند.

مقدار بازگشتی تعداد کاراکترهای موجود در رشته است.

String.Lower: این دستور تمام حروف رشته ورودی را به حروف کوچک تبدیل می‌کند.

مقدار بازگشتی رشته ای است حاوی متن با حروف کوچک.

String.MakePath: این دستور با گرفتن آرایه حاوی متغیرهای (Drive, Folder, Filename و Extension) مسیر فایل را درست می‌کند.

مقدار بازگشتی مسیر فایل بر اساس مقادیر ورودی است.

String.Mid:با استفاده از این دستور می‌توانید از قسمتی از متن به تعداد مورد دلخواه کاراکتر جدا کنید .ورودی‌ها رشته، محل شروع و تعداد کاراکتر مورد نیاز می‌باشند.

مقدار بازگشتی متن جداشده می‌باشد.

String.Repeat:این دستور امکان تکرار یک متن را به تعداد مورد نیاز به شما می‌دهد .رشته و تعداد تکرار ورودی‌های دستور هستند.

مقدار بازگشتی رشته ای است حاوی متن اولیه به تعداد مورد نیاز .

String.Replace:با استفاده از این دستور می‌توانید کلمه مورد نظرتان را جایگزین کلمه دیگری در متن کنید.رشته اصلی، کلمه(رشته) برای جستجو، کلمه برای جایگزینی و مقداری بولین برای تعیین حساسیت به حروف کوچک و بزرگ ورودی‌های دستور هستند.

مقدار بازگشتی رشته حاوی متن جدید می‌باشد .

String.ReverseFind:این دستور مشابه String.Find عمل می‌کند با تفاوت اینکه جستجو را از انتها انجام می‌دهد .

مقدار بازگشتی موقعیت ابتدای کلمه در متن از سمت راست است

String.Right: با استفاده از این دستور می‌توانید به مقدار مورد نیاز از کاراکترهای سمت راست (انتهای)یک رشته جدا کنید. رشته و تعداد مورد نیاز شما ورودی‌های دستور هستند.

مقدار بازگشتی رشته ای جدید است.

String.SplitPath:این دستور نیز برعکس دستور String.MakePath عمل می‌کند . به این صورت که مسیر را به عنوان ورودی دریافت و آرایه ای حاوی آدرس‌های مختلف باز می‌گرداند

مقدار بازگشتی آرایه ای است با متغیرهای Drive , Folder , Filename و Extension

String.ToNumber:این دستور امکان تبدیل یک رشته به عدد را مهیا می‌کند.ورودی رشته می‌باشد.

مقدار بازگشتی عدد می‌باشد . در صورتی که کاراکتر غیر عددی در ورودی باشد، مقدار 0 بازگردانده می‌شود.

String.TrimLeft: این دستور از سمت چپ، حرف، کلمه یا فضای خالی را از رشته حذف می‌کند. رشته اصلی و عبارت مورد نظر ورودی‌های دستور هستند.

مقدار بازگشتی رشته ای حاوی متن جدید است .

String.TrimRight: این دستور از سمت راست (انتها)، حرف، کلمه یا فضای خالی را از رشته حذف می‌کند. رشته اصلی و عبارت مورد نظر ورودی‌های دستور هستند.

مقدار بازگشتی رشته ای حاوی متن جدید است .

String.Upper: این دستور تمام حروف رشته ورودی را به حروف بزرگ تبدیل می‌کند.

مقدار بازگشتی رشته ای است حاوی متن با حروف بزرگ.

System

از این تابع برای کارهایی نظر به دست آوردن ساعت و تاریخ، زبان پیش فرض، اطلاعات کاربر، اطلاعات سیستم، انجام تنظیماتی بر روی سیستم عامل و ... استفاده می‌شود. مجموعه دستورات موجود در این تابع به شرح زیر است:

System.EnumerateProcesses: با این دستور می‌توان پروسه در حال اجرا در سیستم عامل را به دست آورد.

مقدار بازگشتی آرایه ای است حاوی پروسه های در حال اجرا .

System.GetDate: با استفاده از این دستور می‌توانید تاریخ سیستم را به دست آورید.مقدار ورودی عددی نشان دهنده نوع تاریخ درخواستی (آمریکا، اروپا، روز، ماه، سال و ...)

مقدار بازگشتی رشته حاوی تاریخ است .

System.GetDefaultLangID: این دستور کد زبان پیش فرض و زبان دوم ویندوز را باز می‌گرداند.

مقدار بازگشتی آرایه ای است با متغیرهای Primary و Secondary

مثال:

```
tLangID = System.GetDefaultLangID();

LangID = tLangID.Primary;
```

نکته: کد زبان فارسی 41 و کد زبان انگلیسی 9 می‌باشد.

System.GetDisplayInfo: این دستور اطلاعات صفحه نمایش را در اختیار قرار می‌دهد.

مقدار بازگشتی آرایه ای است با متغیرهای Width، Height و ColorDepth .

System.GetLANInfo: با استفاده از این دستور می‌توانید اطلاعات مربوط به Lan(Local Area Network) را به دست آورید .

مقدار بازگشتی آرایه ای است با متغیرهایHost ، Domain، User، IP، NIC.

System.GetMemoryInfo: با استفاده از این دستور می‌توانید اطلاعات مربوط به حافظه را به دست آورید.

مقدار بازگشتی آرایه ای است با متغیرهای AvailablePageFile، AvailableRAM، AvailableVirtual، MemoryLoad، TotalPageFile، TotalRAM، TotalVirtual (بر حسب مگابایت)

System.GetMousePosition: این دستور موقعیت نشان گر ماوس را باز می‌گرداند. مقدار ورودی متغیری بولین است برای تنظیم محاسبه فاصله از گوشه پنجره (True) یا گوشه صفحه نمایش (False)

مقدار بازگشتی آرایه ای است با متغیرهای X و Y.

System.GetOSName: با استفاده از این دستور می‌توانید نام ویندوز (ویندوز 7، ویندوز XP و ...) را به دست آورید.

مقدار بازگشتی رشته ای است حاوی نام ویندوز

System.GetOSProductInfo: نوع Product ویندوز با این دستور به دست می‌آید. لازم به ذکر است این دستور فقط در ویندوز Vista و بعد از آن قابل اجراست

مقدار بازگشتی عددی است نمایش دهنده نوع نسخه‌ی ویندوز:

ULTIMATE, HOME_BASIC, STARTER

System.GetOSVersionInfo: این دستور نیز نسخه ویندوز را باز می‌گرداند.

مقدار بازگشتی آرایه ای است با متغیرهای زیر:

MinorVersion, BuildNumber, PlatformId, MajorVersion
CSDVersion,

System.GetTime: با استفاده از این دستور می‌توانید ساعت سیستم را به دست آورید.مقدار ورودی عددی است نشان دهنده نوع ساعت درخواستی (24 ساعته، روز و شب، ثانیه، دقیقه و ساعت)

مقدار بازگشتی رشته حاوی زمان است .

System.GetUserInfo: این دستور اطلاعات کاربر را باز می‌گرداند.

مقدار بازگشتی جدولی است شامل متغیرهای:

RegOwner, RegOrganization, IsAdmin ,IsVistaAdminLimitedToken

System.Is64BitOS: این دستور 64 بیت بودن ویندوز را کنترل می‌کند.

مقدار بازگشتی عبارتی بولین است که True به معنای 64 بیت بودن ویندوز می‌باشد .

System.IsSystemRestoreAvailable: این گزینه فعال یا غیرفعال بودن System Restore را نشان می‌دهد.

مقدار بازگشتی عبارتی بولین است .

System.IsKeyDown: این دستور فشرده شدن یک دکمه را بررسی می‌کند.شماره دکمه (کلید) ورودی دستور است.

مقدار بازگشتی عبارتی بولین است.

System.Reboot: این دستور سیستم کاربر را راه اندازی مجدد ("ری استارت") می‌کند.

مقدار بازگشتی **ندارد** .

System.RegisterActiveX: این دستور یک ActiveX را در سیستم ثبت می‌کند. ورودی دستور مسیر فایل می‌باشد.

مقدار بازگشتی **ندارد** .

System.RegisterFont: با استفاده از این دستور می‌توانید یک فونت را در سیستم ثبت نمایید. مسیر فونت (با پسوند ttf)، نام فونت برای نمایش، متغیر بولین برای تعیین نوع رجیستر (لحظه یا زمان راه اندازی سیستم) ورودی‌های دستور هستند.

مقدار بازگشتی **ندارد** .

System.RegisterTypeLib: ثبت فایل type library با پسوند tpl با این دستور امکان پذیر است. ورودی، مسیر کامل فایل است.

مقدار بازگشتی **ندارد** .

System.RemoveRestorePoint: این دستور امکان پاک کردن Restore Point را با گرفتن شماره آن به عنوان ورودی، به شما می‌دهد.

مقدار بازگشتی **ندارد** .

System.SetRestorePoint: با استفاده از این دستور می‌توانید یک Restore Point جدید بسازید .نوع رویداد، نوع نشانه (APP_INSTALL, APP_UNINSTALL, APP_DDRIVER_INSTALL, MODIFY_SETTINGS, CANCELLED_OPERATION)، شماره و توضیحات ورودی‌های دستور هستند.

مقدار بازگشتی عدد مربوط به Restore Point ساخته شده می‌باشد.

System.TerminateProcess: این دستور با گرفتن شناسه فایل در حال پردازش، آن را متوقف می‌کند.

مقدار بازگشتی عبارتی بولین است که موفق یا ناموفق بودن متوقف کردن را نشان می‌دهد.

System.UnregisterActiveX: با استفاده از این دستور می‌توانید یک Active X را از حالت ثبت در سیستم خارج نمایید .مسیر کامل فایل، ورودی دستور است.

مقدار بازگشتی **ندارد** .

System.UnregisterFont: با استفاده از این دستور هم می‌توانید یک فونت را از حالت ثبت در سیستم خارج نمایید .مسیر کامل فونت، نام نمایش فونت و نوع خارج کردن، ورودی‌های دستور هستند.

مقدار بازگشتی **ندارد** .

Table

این تابع برای کار با **آرایه‌ها** مورد استفاده قرار می‌گیرد. دستورات زیر مجموعه‌ی این تابع به شرح زیر است:

Table.Concat: با استفاده از این دستور می‌توانید تمام یا قسمتی از خانه های آرایه را به هم متصل کنید. نام آرایه، جدا کننده خانه‌ها، ابتدا و انتهای مکانی که می‌خواهید در کنار هم قرار گیرند، ورودی‌ها هستند.

مقدار بازگشتی رشته ای است حاوی متن خانه‌ها .

Table.Count: این دستور با گرفتن نام آرایه تعداد خانه های آرایه را محاسبه می‌کند.

مقدار بازگشتی تعداد خانه های آرایه می‌باشد.

Table.Insert: با استفاده از این دستور می‌توانید خانه ای در محل مورد نظرتان به آرایه اضافه نمایید.ورودی‌ها نام آرایه، شماره سطر مورد نظر و مقدار برای قرارگیری در خانه جدید می‌باشند.

مقدار بازگشتی **ندارد** .

Table.Remove: با استفاده از این دستور می‌توانید خانه ای از آرایه را حذف نمایید. نام آرایه و شماره سطر ورودی‌ها هستند.

مقدار بازگشتی عبارت موجود در خانه آرایه است .

Table.Sort: این دستور عمل مرتب سازی جدول را انجام می‌دهد .نام آرایه و تابع مقایسه (در حالت عادی nil است.)

مقدار بازگشتی **ندارد** .

TextFile

این تابع برای کار با فایل‌های متنی با فرمت txt به کار می‌رود. دستورات زیر مجموعه این تابع به شرح زیر می‌باشند:

TextFile.ReadToString: با استفاده از این دستور می‌توانید محتویات یک فایل متنی به صورت رشته ای فراخوانی کنید. مسیر فایل را هم در پارامتر ورودی دستور وارد می‌کنید.

مقدار بازگشتی رشته ای است حاوی متن داخل فایل .

TextFile.ReadToTable: با استفاده از این دستور می‌توانید محتویات یک فایل متنی به صورت آرایه فراخوانی کنید. مسیر فایل را هم در پارامتر ورودی دستور وارد می‌کنید.

مقدار بازگشتی آرایه ای است که هر خانه‌ی آن حاوی اطلاعات و متن یک خط فایل متنی می‌باشد.

TextFile.WriteFromString: با استفاده از این دستور هم می‌توانید یک رشته را داخل یک فایل متنی ذخیره نمایید .پارامترهای این دستور نیز شامل مسیر فایل برای ذخیره، متن و یک متغیر بولین که تعیین کننده نوع ذخیره سازی است (true برای نگه داشتن متن قبلی و false برای نوشتن روی فایل قبلی) می‌باشد.

مقدار بازگشتی **ندارد** .

TextFile.WriteFromTable: با استفاده از این دستور هم می‌توانید یک آرایه را داخل یک فایل متنی ذخیره نمایید. پارامترهای این دستور نیز شامل مسیر فایل برای ذخیره، نام آرایه و متغیر بولین(true برای نگه داشتن متن قبلی و false برای نوشتن روی فایل قبلی) ورودی دستور است.

مقدار بازگشتی **ندارد** .

Tree

از این دستور برای کار با شی Tree(درخت‌واره) استفاده می‌کنیم. دستورات زیر مجموعه این تابع به شرح زیر است:

Tree.CollapseNode: این دستور زیر شاخه های یک گره را از حالت نمایش خارج می‌کند (گره را می‌بندد.)نام شی درخت و شماره گره ورودی‌ها هستند.

مقدار بازگشتی **ندارد** .

Tree.EnsureVisible: این دستور درخت را تا گره مورد نظر به نمایش در می‌آورد (باز می‌کند). ورودی‌ها نام شی و شماره گره می‌باشد.

مقدار بازگشتی **ندارد** .

Tree.ExpandNode: این دستور تمام زیر شاخه های یک گره را باز می‌کند. ورودی‌ها نام شی و شماره گره می‌باشد.

مقدار بازگشتی **ندارد** .

Tree.FindNodeByData: با استفاده از این دستور می‌توانید گره با Data خاص را جستجو نمایید. نام شی، محل شروع برای جستجو(0 به معنای شروع از بالاترین سطح) و رشته برای جستجو بین Data ی گره‌ها ورودی‌های دستورند.

مقدار بازگشتی رشته ای حاوی شماره گره است. در صورت پیدا نکردن یا بروز خطا رشته خالی "" بازگردانده می‌شود.

Tree.FindNodeByText: با استفاده از این دستور می‌توانید گره با متن خاص را جستجو نمایید. نام شی، محل شروع برای جستجو (0 به معنای شروع از بالاترین سطح) و رشته برای جستجو بین متون، ورودی‌های دستور هستند.

مقدار بازگشتی رشته ای حاوی شماره گره است. در صورت پیدا نکردن یا بروز خطا رشته خالی "" بازگردانده می‌شود.

Tree.GetChildCount: این دستور زیر شاخه های یک گره در سطح پایینی را می شماره. ورودی‌ها نام شی و شماره گره برای شمارش زیر مجموعه‌ها هستند.

مقدار بازگشتی تعداد زیر شاخه‌ها می‌باشد.

Tree.GetChildren: با استفاده از این دستور می‌توانید اطلاعات مربوط به زیر شاخه یک گره را به دست آورید. نام شی و شماره ردیف ورودی‌های دستور هستند.

مقدار بازگشتی آرایه ای است عددی با متغیرهای زیر :

Text, Data, Selected, Expanded, NodeIndex, ImageIndex, SelectedImageIndex

در صورتی که اطلاعات در متغیری مثل t قرار بگیرد، متن زیر گره اول به این صورت به دست می‌آید :

T[1].Text

Tree.GetNode: این دستور اطلاعات یک گره را باز می‌گرداند. نام شی و شماره ردیف ورودی‌های دستور هستند.

مقدار بازگشتی آرایه ای است با متغیرهای Text , Data , Selected , Expanded , Checked , NodeIndex , ImageIndex , SelectedImageIndex .

Tree.GetPos: با استفاده از این دستور می‌توانید موقعیت قرارگیری شی درخت را بدست آورید .این موقعیت نسبت به گوشه‌ی بالا و سمت چپ محاسبه می‌شود.ورودی این دستور نام شی می‌باشد .

مقدار بازگشتی یک آرایه با متغیرهای X و Y می‌باشد.

Tree.GetProperties: با استفاده از این دستور می‌توانید، خصوصیات شی Input را در اختیار داشته باشید. نام شی ورودی دستور است.

مقدار بازگشتی آرایه ای است حاوی :

ObjectName: نام شی

ShowCheckBoxes: وضعیت فعال بودن Check Box (بولین)

HasLines: وضعیت فعال بودن خطوط (بولین)

LinesAtRoot: وضعیت فعال بودن خط در گره های اولین سطح (بولین)

HasButtons: وضعیت فعال بودن دکمه باز و بسته کردن زیر شاخه‌ها (بولین)

EditLabels: وضعیت فعال بودن ویرایش متن گره‌ها (بولین)

AlwaysShowSelection: وضعیت فعال بودن نمایش آیتم انتخاب شده در هنگام انتخاب نبودن شی درخت (بولین)

FontName: نام فونت استفاده شده

FontSize: اندازه فونت

FontStrikeout: فعال یا غیر فعال بودن strikeout در تنظیمات فونت (بولین)

FontUnderline: فعال یا غیر فعال بودن underline در تنظیمات فونت (بولین)

FontAntiAlias: فعال یا غیر فعال بودن شیalias در تنظیمات فونت (بولین)

FontItalic: فعال یا غیر فعال بودن italic در تنظیمات فونت (بولین)

FontWeight: نوع نوشته (تیرگی متن) به صورت آرایه ای متشکل از DONTCARE, THIN, EXTRALIGHT, LIGHT, NORMAL, MEDIUM, SEMIBOLD, BOLD, EXTRABOLD و HEAVY.

FontScript: نوع اسکریپت فونت به صورت آرایه ای متشکل از ANSI , BALTIC , CHINESEBIG5 , DEFAULT , EASTEUROPE , GB2312 , GREEK , HANGUL , MAC , OEM , RUSSIAN , SHIFTJIS ,SYMBOL و TURKISH .

BackgroundColor: شماره رنگ پس زمینه

TextColor: رنگ متن

Border: نوع Border (بدون Border، FLAT یا SUNKEN)

UseImageList: وضعیت استفاده از لیست تصاویر (بولین)

ImageList:

ImageListTransColor:

ReadOrder: نوع نمایش متن (استاندارد یا راست به چپ)

Enabled: فعال یا عدم فعال بودن (بولین)

Visible: دیده شدن یا عدم دیده شدن (بولین)

X: فاصله از سمت چپ صفحه

Y: فاصله از سمت راست صفحه

Width: عرض (پیکسل)

Height: طول (پیکسل)

TooltipText: متن نمایش دهنده در زمان قرارگیری نشانگر ماوس روی شی.

Cursor: عدد شکل نشانگر ماوس یا نام ثابت آن:

ARROW , HAND , BLACK_ARROW, CROSSHAIR, EXPLORE, HEL ,
یا MAGNIFY, MEDIA, MONEY, NOTEPAD, PENCIL, PRINTER , SPEAKER
UP_ARROW

ResizeLeft: قابلیت تغییر اندازه از سمت چپ در زمان تغییر اندازه پروژه (بولین)

ResizeRight: قابلیت تغییر اندازه از سمت راست در زمان تغییر اندازه پروژه (بولین)

ResizeTop: قابلیت تغییر اندازه از بالا در زمان تغییر اندازه پروژه (بولین)

ResizeBottom: قابلیت تغییر اندازه از پایین در زمان تغییر اندازه پروژه (بولین)

WindowHandle: شماره هندل شی

Tree.GetSelectedNode:این دستور گره انتخاب شده را مشخص می‌کند. نام شی درخت ورودی دستور است .

مقدار بازگشتی رشته ای حاوی شماره گره است

Tree.GetSize: نمایش اندازه شی درخت بر حسب پیکسل با این دستور امکان پذیر است.ورودی دستور نام شی می‌باشد.

مقدار بازگشتی آرایه ای 2 عضوی با اعضای "Width" و "Height" می‌باشد.

Tree.InsertNode: با استفاده از این دستور می‌توانید یک گره به درخت اضافه نمایید.ورودی‌ها عبارتند از: نام شی درخت، شماره گره و آرایه اطلاعات گره با متغیرهای زیر:

Text, Data, Selected, Expanded, Checked, NodeIndex, ImageIndex, SelectedImageIndex

Tree.IsEnabled: با استفاده از این دستور می‌توانید وضعیت فعال یا غیرفعال بودن درخت را بدست آورید. مقدار ورودی نام شی می‌باشد.

مقدار بازگشتی Trueبه معنای فعال بودن و False به معنای غیرفعال بودن شی درخت می‌باشد.

Tree.IsVisible: با استفاده از این دستور می‌توانید نمایش یا عدم نمایش شی درخت را بدست آورید. مقدار ورودی نام شی است.

مقدار بازگشتی Trueبه معنای نمایش یا False به معنای عدم نمایش لیست باکس می‌باشد.

Tree.RemoveNode:این دستور با گرفتن نام شی و شماره گره به عنوان ورودی گره (0 به معنای تمام گره‌ها) مورد نظر را حذف می‌کند .

مقدار بازگشتی **ندارد** .

Tree.SetEnabled:: با این دستور می‌توانید درخت مورد نظر را فعال یا غیر فعال کنید.ورودی دستور نام شی و همچنین عبارت بولین برای فعال یا غیرفعال کردن آن می‌باشد.

مقدار بازگشتی **ندارد** .

Tree.SetNode:این دستور خصوصیات را بر روی گره اعمال می‌کند.نام شی، شماره گره و آرایه خصوصیات گره ورودی‌های دستور هستند

مقدار بازگشتی **ندارد** .

Tree.SetPos: با این کد می‌توانید محل قرارگیری لیست باکس را تعیین نمایید.ورودی‌های این کد نام شی، عدد فاصله از چپ و عدد دیگری برای فاصله از بالا می‌باشد.

مقدار بازگشتی **ندارد** .

Tree.SetProperties: با استفاده از این دستور می‌توانید خصوصیات مورد نظرتان را بر روی شی اعمال کنید .ورودی دستور، نام شی درخت و آرایه خصوصیات جدید می‌باشد.(این آرایه در Tree.GetProperties توضیح داده شده است.)

Tree.SetSelectedNode: با استفاده از این دستور می‌توانید گره مورد نظر را به حالت انتخاب در آورید . نام شی و شماره گره ورودی هستند.

مقدار بازگشتی **ندارد** .

Tree.SetSize: با این دستور می‌توانید طول و عرض شی را به اندازه دلخواه خود تغییر دهید.ورودی‌های این کد نام شی، عددی برای عرض و عددی دیگر برای طول دکمه می‌باشد.

مقدار بازگشتی **ندارد** .

Tree.SetVisible: کنترل نمایش یا عدم نمایش درخت با این دستور امکان پذیر است. ورودی‌های کد، نام شی و عبارتی بولین برای فعال یا غیر فعال کردن نمایش شی می‌باشد.

مقدار بازگشتی **ندارد**.

Video

این تابع برای کار با شی Video به کار می‌رود. دستورات زیر مجموعه‌ی این تابع به شرح زیر است:

Video.GetCurrentPos: این دستور امکان به دست آوردن زمان فعلی ویدئو را به شما می‌دهد .نام شی ورودی دستور است.

مقدار بازگشتی میزان زمان سپری شده از فیلم بر حسب ثانیه می‌باشد.

Video.GetFilename: با استفاده از این دستور می‌توانید مسیر فایل تصویری را به دست آورید. نام شی ورودی دستور می‌باشد.

مقدار بازگشتی رشته ای است حاوی مسیر فایل .

Video.GetLength: این دستور زمان کل ویدئو را باز می‌گرداند. نام شی ورودی دستور می‌باشد.

مقدار بازگشتی زمان کل فیلم بر حسب ثانیه است .

Video.GetPos: با استفاده از این دستور می‌توانید موقعیت قرارگیری شی ویدئو را بدست آورید .ورودی این دستور نام شی می‌باشد.

مقدار بازگشتی یک آرایه با متغیرهای X و Y می‌باشد.

Video.GetProperties: این دستور خصوصیات شی ListBox را در اختیار شما قرار می‌دهد. نام شی ورودی تابع است.

مقدار بازگشتی آرایه ای شامل موارد زیر می‌باشد :

ObjectName: نام شی

VideoFile: فایل تصویری بارگذاری شده

MaskFile: مسیر فایل ماسک

ApplyCustomMask: وضعیت استفاده از ماسک (بولین)

ScalingMode: وضعیت قرارگیری ویدئو در شی (STRETCH_MODE یا MAINTAIN_ASPECT)

CPStyle: نوع نمایش پنل شی (بدون پنل، معمولی یا سفارشی)

SkinFile: مسیر فایل پوسته در صورت سفارشی بودن CPStyle

CPTime: نوع نمایش زمان در پنل

PanelColor: رنگ پنل

TextColor: رنگ متن

ControlButtons: وضعیت نمایش دکمه های کنترلی در پنل (بولین)

Slider: وضعیت نمایش اسلایدر (بولین)

AutoStart: وضعیت پخش اتوماتیک ویدئو (بولین)

Loop: وضعیت فعال بودن تکرار مجدد فایل

Border: نوع Border (بدون Border، FLAT یا SUNKEN)

Enabled: فعال یا عدم فعال بودن (بولین)

Visible: دیده شدن یا عدم دیده شدن (بولین)

X: فاصله از سمت چپ صفحه

Y: فاصله از سمت راست صفحه

Width: عرض (پیکسل)

Height: طول (پیکسل)

TooltipText: متن نمایش دهنده در زمان قرارگیری نشانگر ماوس روی شی.

Cursor: عدد شکل نشانگر ماوس:

ARROW, HAND, BLACK_ARROW, CROSSHAIR, EXPLORE, HELP, MAGNIFY, MEDIA, MONEY, NOTEPAD, PENCIL, PRINTER, SPEAKER یا UP_ARROW

ResizeLeft: قابلیت تغییر اندازه از سمت چپ در زمان تغییر اندازه پروژه (بولین)

ResizeRight: قابلیت تغییر اندازه از سمت راست در زمان تغییر اندازه پروژه (بولین)

ResizeTop: قابلیت تغییر اندازه از بالا در زمان تغییر اندازه پروژه (بولین)

ResizeBottom: قابلیت تغییر اندازه از پایین در زمان تغییر اندازه پروژه (بولین)

WindowHandle: شماره هندل شی

Video.GetSize: نمایش اندازه شی ویدئو بر حسب پیکسل با این دستور امکان پذیر است.ورودی دستور نام شی می‌باشد.

مقدار بازگشتی دستور، آرایه ای 2 عضوی با اعضای "Width" و "Height" می‌باشد.

Video.GetState: این دستور وضعیت ویدئو را مشخص می‌کند .نام شی ورودی دستور است.

مقدار بازگشتی عددی بین 1- تا 2 است (Stop=0 , Pause = 1 , Playing = 2) در صورت بروز خطا 1- بازگردانده می‌شود .

Video.GetVolume: میزان صدای ویدئو با این دستور به دست می‌آید. نام شی ورودی دستور است .

مقدار بازگشتی عددی بین0 تا 100 نشان دهنده میزان صدای ویدئو می‌باشد .

Video.IsEnabled: با استفاده از این دستور می‌توانید وضعیت فعال بودن شی ویدئو را بدست آورید. مقدار ورودی نام شی می‌باشد.

مقدار بازگشتی Trueبه معنای فعال بودن و False به معنای غیرفعال بودن شی است.

Video.IsFullScreen: با استفاده از این دستور می‌توانید وضعیت تمام صفحه (فول اسکرین) بودن ویدئو را بدست آورید. مقدار ورودی نام شی می‌باشد.

مقدار بازگشتی Trueبه معنای تمام صفحه بودن و False به معنای نمایش در اندازه معمولی است.

Video.IsVisible: با استفاده از این دستور می‌توانید نمایش یا عدم نمایش ویدئو را بدست آورید. مقدار ورودی نام شی است.

مقدار بازگشتی Trueبه معنای نمایش یا False به معنای عدم نمایش شی می‌باشد.

Video.Load: با استفاده از این دستور می‌توانید فایل تصویری را در شی ویدئو بارگذاری نمایید. نام شی، مسیر فایل، وضعیت نمایش اتوماتیک (بولین) و وضعیت تکرار فیلم (بولین) ورودی‌های دستور هستند.

مقدار بازگشتی **ندارد**.

Video.Pause: این دستور عمل مکث در فیلم را انجام می‌دهد. نام شی ورودی دستور است.

مقدار بازگشتی **ندارد**.

Video.Play: این دستور عمل نمایش فیلم را انجام می‌دهد. نام شی ورودی دستور است.

مقدار بازگشتی **ندارد**.

Video.Seek: پرش به موقعیت دلخواه ویدئو در کانال مورد نظر. ورودی اول نام شی می‌باشد، ورودی دوم نوع پرش است:

SEEK_BEGINNING: پرش به اول صوت

_SEEK: پرش به انتهای صوت

SEEK_FORWARD: پرش به جلو

SEEK_BACKWARD: پرش به عقب

SEEK_SPECIFIC: پرش به زمان خاص

ورودی سوم هم زمان مناسب برای پرش (برای موارد سوم تا پنجم) استفاده می‌شود.

مقدار بازگشتی **ندارد** .

Video.SetEnabled: با این دستور می‌توانید شی مورد نظر را فعال یا غیر فعال کنید.ورودی دستور نام شی و همچنین عبارت بولین برای فعال یا غیرفعال کردن آن می‌باشد.

مقدار بازگشتی **ندارد** .

Video.SetFullScreen: با این دستور می‌توانید وضعیت تمام صفحه بودن ویدئو را کنترل کنید.ورودی‌های دستور نام شی و همچنین عبارت بولین می‌باشد.

مقدار بازگشتی **ندارد** .

Video.SetPos: با این کد می‌توانید محل قرارگیری لیست باکس را تعیین نمایید.ورودی‌های این کد نام شی، عددی فاصله از چپ و عدد دیگری برای فاصله از بالا می‌باشد.

مقدار بازگشتی **ندارد** .

Video.SetProperties: با استفاده از این دستور می‌توانید خصوصیات مورد نظرتان را بر روی ویدئو اعمال کنید .ورودی دستور، نام شی و آرایه خصوصیات جدید می‌باشد. (این آرایه در Video.GetProperties توضیح داده شده است.)

Video.SetSize: با این دستور می‌توانید طول و عرض شی ویدئو را به اندازه دلخواه خود تغییر دهید.ورودی‌های این کد نام شی، عددی برای عرض و عددی دیگر برای طول شی می‌باشد.

مقدار بازگشتی **ندارد** .

Video.SetVisible: کنترل نمایش یا عدم نمایش شی ویدئو با این دستور امکان پذیر است. ورودی‌های کد، نام شی و عبارتی بولین برای فعال یا غیر فعال کردن نمایش شی می‌باشد.

مقدار بازگشتی **ندارد** .

Video.SetVolume:با استفاده از این کد می‌توانید میزان بلندی صدای ویدئو را تنظیم نمایید .نام شی و میزان صدا(0 برابر قطع صدا و 100 برابر بیشترین میزان صدا) ورودی‌ها هستند.

مقدار بازگشتی **ندارد** .

Video.Stop: این دستور عمل توقف فیلم را انجام می‌دهد. نام شی ورودی دستور است.

مقدار بازگشتی **ندارد**.

QuickTime

این تابع برای کار با فایل‌هایی که با نرم افزار QuickTime اجرا می‌شوند به کار می‌رود و آن‌ها را در شی QuickTime به نمایش در می‌آورد. انواع فرمت‌هایی که با شی QuickTime قابل نمایش هستند شرح زیر می‌باشند:

نوع فایل	فرمت فایل	نوع فایل	فرمت فایل	نوع فایل	فرمت فایل
سند	.qht	ویدئو	.m15	ویدئو	.386
سند	.qhtm	صوتی	.m1a	ویدئو	.3g2
ویدئو	.qt	ویدئو	.m1s	ویدئو	.3gp
عکس	.qti	ویدئو	.m1v	ویدئو	.3gp2
عکس	.qtif	صوتی	.m3u	ویدئو	.3gpp
ویدئو	.qtl	صوتی	.m3url	صوتی	.aac
عکس	.rgb	صوتی	.m4a	صوتی	.ac3
ویدئو	.rts	صوتی	.m4b	صوتی	.adts
ویدئو	.rtsp	صوتی	.m4p	صوتی	.aif
صوتی	.sd2	ویدئو	.m4v	صوتی	.aifc
ویدئو	.sdp	ویدئو	.m75	صوتی	.aiff
ویدئو	.sdv	عکس	.mac	ویدئو	.amc
عکس	.sgi	صوتی	.mid	ویدئو	.AMR
صوتی	.smf	صوتی	.midi	صوتی	.au
ویدئو	.smi	ویدئو	.mov	ویدئو	.avi

.bmp	عکس	.mp2	صوتی	.smil	ویدئو
.bwf	صوتی	.mp3	صوتی	.sml	ویدئو
.caf	صوتی	.mp4	ویدئو	.snd	صوتی
.cdda	صوتی	.mpa	ویدئو	.swa	صوتی
.cel	صوتی	.mpeg	ویدئو	.targa	عکس
.dib	عکس	.mpg	ویدئو	.tga	عکس
.dif	ویدئو	.mpm	ویدئو	.tif	عکس
.dv	ویدئو	.mpv	ویدئو	.tiff	عکس
.flc	ویدئو	.mqv	ویدئو	.ulw	صوتی
.fli	ویدئو	.pct	عکس	.vfw	ویدئو
.gif	ویدئو	.pic	عکس	.jpg	عکس
.gsm	صوتی	.pict	عکس	.kar	صوتی
.jp2	عکس	.png	عکس	.psd	عکس
.jpe	عکس	.pnt	عکس	.qcp	صوتی
.jpeg	عکس	.pntg	عکس		

این تابع شامل 23 زیر مجموعه از دستورات می‌باشد که به دلیل شباهت آن با دستورات شی Video از ذکر آن‌ها خود داری می‌کنیم. برای مثال دستور QuickTime.Play درست همان کاری را انجام می‌دهد که دستور Video.Play انجام می‌دهد یعنی پخش کردن و نمایش فایل در شی مورد نظر.

Web

این تابع برای کار با شی Web به کار می‌رود و دستورات زیر مجموعه‌ی آن به شرح زیر می‌باشند:

Web.Back: بازگشت به صفحه قبلی مشاهده شده در شی وب با این دستور امکان پذیر است. نام شی وب ورودی دستور است.

مقدار بازگشتی **ندارد** .

Web.Forward: پرش به صفحه جلو در شی وب (زمانی فعال می‌شود که از Back استفاده شده باشد) با این دستور امکان پذیر است. نام شی وب ورودی دستور است.

مقدار بازگشتی **ندارد** .

Web.GetPos: با استفاده از این دستور می‌توانید موقعیت قرارگیری شی وب را بدست آورید. ورودی این تابع نام شی می‌باشد .

مقدار بازگشتی یک آرایه با متغیرهای X و Y می‌باشد.

Web.GetProperties: این دستور خصوصیات شی وب را در اختیار شما قرار می‌دهد. نام شی ورودی تابع است.

مقدار بازگشتی آرایه ای شامل موارد زیر می‌باشد :

URL: آدرس صفحه بارگذاری شده در شی

ObjectName: نام شی

ShowBorder: وضعیت نمایش Border (بولین)

ShowScrollbars: وضعیت نمایش اسکرول (بولین)

Enabled: فعال یا عدم فعال بودن (بولین)

Visible: دیده شدن یا عدم دیده شدن (بولین)

X: فاصله از سمت چپ صفحه

Y: فاصله از سمت راست صفحه

Width: عرض (پیکسل)

Height: طول (پیکسل)

ResizeLeft: قابلیت تغییر اندازه از سمت چپ در زمان تغییر اندازه پروژه (بولین)

ResizeRight: قابلیت تغییر اندازه از سمت راست در زمان تغییر اندازه پروژه (بولین)

ResizeTop: قابلیت تغییر اندازه از بالا در زمان تغییر اندازه پروژه (بولین)

ResizeBottom: قابلیت تغییر اندازه از پایین در زمان تغییر اندازه پروژه (بولین)

WindowHandle: شماره هندل شی

Web.GetSize: نمایش اندازه شی وب بر حسب پیکسل با این دستور امکان پذیر است.ورودی دستور نام شی می‌باشد.

مقدار بازگشتی دستور، آرایه ای 2 عضوی با اعضای "Width" و "Height" می‌باشد.

Web.GetURL: این دستور درس صفحه بارگذاری شده در صفحه را باز می‌گرداند .نام شی ورودی دستور است .

مقدار بازگشتی رشته ای است حاوی مسیر فایل .

Web.IsEnabled: با استفاده از این دستور می‌توانید وضعیت فعال بودن شی وب را بدست آورید. مقدار ورودی نام شی می‌باشد.

مقدار بازگشتی true به معنای فعال بودن و false به معنای غیرفعال بودن آن می‌باشد.

Web.IsSilent: این دستور هم Silent بودن شی وب را کنترل می‌کند .نام شی ورودی دستور است

مقدار بازگشتی متغیری بولین است.

Web.IsVisible: با استفاده از این دستور می‌توانید نمایش یا عدم نمایش شی وب را بدست آورید. مقدار ورودی نام شی است.

مقدار بازگشتی true به معنای نمایش یا false به معنای عدم نمایش شی می‌باشد.

Web.LoadHTML:این دستور کد به زبان HTML را در شی وب بارگذاری می‌کند. نام شی و رشته حاوی کدها ورودی‌های دستورند.

مقدار بازگشتی متغیری بولین است نشان دهنده وضعیت موفق بودن بارگذاری.

Web.LoadURL:با استفاده از این دستور می‌توان یک آدرس را بارگذاری نمود. نام شی و آدرس مورد نظر ورودی‌های دستور هستند.

مقدار بازگشتی **ندارد** .

Web.Print: با استفاده از این دستور می‌توانید محتویات شی وب را چاپ کنید .ورودی‌های دستور نام شی و مقداری بولین برای مشخص کردن وضعیت همسان سازی اندازه با صفحه است.

مقدار بازگشتی **ندارد** .

Web.Refresh: عمل به روز رسانی صفحه (مانند دکمه Refresh در مرورگر) را انجام می‌دهد. ورودی دستور نام شی است.

مقدار بازگشتی **ندارد** .

Web.SetEnabled: با این دستور می‌توانید شی وب را فعال یا غیر فعال کنید.ورودی دستور نام شی و همچنین عبارت بولین برای فعال یا غیرفعال کردن آن می‌باشد.

مقدار بازگشتی **ندارد** .

Web.SetPos: با این کد می‌توانید شی وب را به محل دلخواه در صفحه منتقل نمایید.ورودی‌های این کد نام شی، عددی برای جای گیری در قسمت X (فاصله از چپ) و عدد دیگری برای جای گیری در Y (فاصله از بالا می‌باشد).

مقدار بازگشتی **ندارد** .

Web.SetProperties: با استفاده از این دستور می‌توانید خصوصیات جدید برای شی پاراگراف تعیین نمایید. نام شی و آرایه خصوصیات (که در Web.GetProperties توضیح داده شده) ورودی‌های دستور هستند.

Web.SetSilent: کنترل وضعیت Silent بودن شی با این دستور امکان پذیر است. نام شی و متغیر بولین برای فعال و غیرفعال کردن آن ورودی‌ها هستند

مقدار بازگشتی **ندارد** .

Web.SetSize: با این دستور می‌توانید طول و عرض شی وب را به اندازه دلخواه خود تغییر دهید.ورودی‌های این کد نام شی، عددی برای عرض و عددی دیگر برای طول شی می‌باشد.

مقدار بازگشتی **ندارد** .

Web.SetVisible: کنترل نمایش یا عدم نمایش شی وب با این دستور امکان پذیر است. ورودی‌های کد، نام شی و عبارتی بولین برای فعال یا غیر فعال کردن نمایش شی می‌باشد.

مقدار بازگشتی **ندارد** .

Web.Stop: با این دستور می‌توانید بارگذاری صفحه داخل شی وب را متوقف نمایید. نام شی ورودی دستور است.

مقدار بازگشتی **ندارد** .

Window

این تابع برای کار بر روی پنجره‌ی برنامه به کار می‌روند. دستورات زیر مجموعه‌ی این تابع به شرح زیر می‌باشند:

Window.Close: با استفاده از این دستور می‌توانید یک پنجره را ببندید. شماره هندل پنجره و نوع بسته شدن ورودی‌های دستور هستند.

مقدار بازگشتی **ندارد** .

Window.EnumerateProcesses: این دستور لیستی از برنامه‌های در حال پردازش و شماره هندل پنجره آن‌ها را باز می‌گرداند .مقدار بولین برای تعیین نوع پنجره های انتخابی (فقط پنجره های در حال نمایش یا تمام پنجره های فعال) ورودی دستور است.

مقدار بازگشتی جدولی است از برنامه‌های در حال پردازش و شماره هندل پنجره های آن‌ها .

Window.EnumerateTitles: این دستور لیستی از عناوین و شماره هندل پنجره‌ها را باز می‌گرداند .مقدار بولین برای تعیین نوع پنجره های انتخابی (فقط پنجره های در حال نمایش یا تمام پنجره های فعال) ورودی دستور است.

مقدار بازگشتی جدولی است از عناوین و شماره هندل پنجره‌ها.

Window.GetPos: با استفاده از این دستور می‌توانید موقعیت قرارگیری پنجره را بدست آورید. ورودی این دستور شماره هندل پنجره می‌باشد .

مقدار بازگشتی یک آرایه با متغیرهای X و Y می‌باشد. در صورتی که پنجره یافت نشود یا خطایی رخ دهد nil بازگردانده می‌شود.

Window.GetSize: نمایش اندازه پنجره با این دستور امکان پذیر است. ورودی این دستور شماره هندل پنجره می‌باشد.

مقدار بازگشتی دستور، آرایه ای 2 عضوی با اعضای "Width" و "Height" می‌باشد.

Window.Hide: این دستور امکان مخفی کردن پنجره را به شما می‌دهد. ورودی این دستور شماره هندل پنجره می‌باشد .

مقدار بازگشتی **ندارد** .

Window.Maximize: این دستور پنجره را به بزرگ‌ترین اندازه ممکن می‌برد. ورودی این دستور شماره هندل پنجره می‌باشد.

مقدار بازگشتی **ندارد**.

Window.Minimize: این دستور پنجره را کمینه (Minimize) می‌کند. ورودی این دستور شماره هندل پنجره می‌باشد.

مقدار بازگشتی **ندارد** .

Window.Restore: این دستور پنجره را به حالت اصلی باز می‌گرداند. ورودی این دستور شماره هندل پنجره می‌باشد.

مقدار بازگشتی **ندارد**.

Window.SetMask: با استفاده از این دستور می‌توانید ماسک بر روی پنجره قرار دهید. . ورودی‌های این دستور شماره هندل پنجره و مسیر تصویر برای ماسک پنجره، وضعیت تغییر اندازه فایل(بولین) و میزان شفافیت می‌باشد .

مقدار بازگشتی **ندارد** .

Window.SetOrder: با این دستور می‌توانید نوع قرارگیری پنجره نسبت به پنجره های دیگر را تعیین نمایید. شماره هندل پنجره نوع قرارگیری ورودی‌های دستور هستند .

مقدار بازگشتی **ندارد** .

Window.SetPos: با این کد می‌توانید پنجره را به محل دلخواه در صفحه منتقل نمایید.ورودی‌های این کد شماره هندل پنجره، فاصله از چپ و بالا ورودی‌های دستورند.

مقدار بازگشتی **ندارد** .

Window.SetSize: با این دستور می‌توانید طول و عرض پنجره را به اندازه دلخواه خود تغییر دهید.ورودی‌های این کد شماره هندل پنجره، عددی برای عرض و عددی دیگر برای طول شی می‌باشد.

مقدار بازگشتی **ندارد** .

Window.SetText:با ورود شماره هندل صفحه و متن، می‌توانید عنوان پنجره را تغییر دهید.

مقدار بازگشتی **ندارد** .

Window.Show: این دستور امکان نمایش پنجره را به شما می‌دهد. ورودی این دستور شماره هندل پنجره می‌باشد .

مقدار بازگشتی **ندارد** .

xButton

این تابع برای کار با شی xButton (دکمه‌ی پیش رفته) به کار می‌رود. دستورات زیر مجموعه این تابع به شرح زیر است:

xButton.GetImage: این دستور می‌توانید مسیر تصویر استفاده شده برای xButton را به دست آورید. نام شی ورودی دستور است.

مقدار بازگشتی رشته ای است حاوی مسیر فایل

xButton.GetPos: با استفاده از این دستور می‌توانید موقعیت قرارگیری دکمه در صفحه را بدست آورید .این موقعیت نسبت به گوشه‌ی بالا و سمت چپ محاسبه می‌شود.ورودی این تابع نام شی می‌باشد.

مقدار بازگشتی یک آرایه با متغیرهای X و Y می‌باشد.

xButton.GetProperties: با این دستور می‌توانید خصوصیات دکمه را در یک آرایه داشته باشید.ورودی این تابع نام دکمه می‌باشد.

مقدار بازگشتی این دستور، یک آرایه با متغیرهای زیر می‌باشد :

ObjectName: نام شی

Text: متن موجود در دکمه

ButtonImage: مسیر تصویر دکمه

TextColor: شماره رنگ متن

EnableMarkup: وضعیت Markup (بولین)

FontName: نام فونت استفاده شده

FontSize: اندازه فونت

FontStrikeout: فعال یا غیر فعال بودن strikeout در تنظیمات فونت (بولین)

FontUnderline: فعال یا غیر فعال بودن underline در تنظیمات فونت (بولین)

FontAntiAlias: فعال یا غیر فعال بودن anti alias در تنظیمات فونت (بولین)

FontItalic: فعال یا غیر فعال بودن italic در تنظیمات فونت (بولین)

FontWeight: نوع نوشته (تیرگی متن) به صورت آرایه ای متشکل از:

DONTCARE, THIN, EXTRALIGHT, LIGHT, NORMAL, MEDIUM, SEMIBOLD, BOLD, EXTRABOLD, HEAVY.

FontScript: نوع اسکریپت فونت به صورت آرایه ای متشکل از:

ANSI , BALTIC , CHINESEBIG5 , DEFAULT , EASTEUROPE , GB2312 , GREEK , HANGUL , MAC , OEM , RUSSIAN , SHIFTJIS ,SYMBOL و TURKISH.

TextAlignment: چینش متن در دکمه

ImageAlignment:نوع قرارگیری تصویر

ImageTextRelation: نوع قرارگیری متن و تصویر

Theme: تم استفاده شده در دکمه

Style: نوع استیل (STANDARD یا TOGGLE)

ToggleState: نوع TOGGLE (UP یا DOWN)

Enabled: فعال یا عدم فعال بودن دکمه (بولین)

Visible: دیده شدن یا عدم دیده شدن دکمه (بولین)

X: فاصله دکمه از سمت چپ صفحه

Y: فاصله دکمه از سمت راست صفحه

Width: عرض دکمه (پیکسل)

Height: طول دکمه (پیکسل)

TooltipText: متن نمایش دهنده زمانی که نشانگر ماوس روی آن دکمه قرار می‌گیرد.

Cursor: شکل نشانگر ماوس. آرایه ای با اعضای :

ARROW , HAND , BLACK_ARROW , CROSSHAIR , EXPLORE , HELP , MAGNIFY , MEDIA , MONEY , NOTEPAD , PENCIL , PRINTER , SPEAKER یا UP_ARROW

ResizeLeft: قابلیت تغییر اندازه از سمت چپ در زمان تغییر اندازه پروژه (بولین)

ResizeRight: قابلیت تغییر اندازه از سمت راست در زمان تغییر اندازه پروژه (بولین)

ResizeTop: قابلیت تغییر اندازه از بالا در زمان تغییر اندازه پروژه (بولین)

ResizeBottom: قابلیت تغییر اندازه از پایین در زمان تغییر اندازه پروژه (بولین)

HighlightSound: نوع صوت در هنگام قرارگیری نشانگر ماوس بر روی دکمه (NONE, STANDARD یا CUSTOM)

HighlightSoundFile: مسیر فایل صوتی پخش شونده در هنگام قرارگیری نشانگر ماوس بر روی دکمه

ClickSound: نوع صوت در هنگام کلیک بر روی دکمه (NONE, STANDARD یا CUSTOM)

ClickSoundFile: مسیر فایل صوتی پخش شونده در هنگام کلیک بر روی دکمه

WindowHandle: شماره هندل شی

xButton.GetSize: نمایش اندازه دکمه به پیکسل با این دستور امکان پذیر است.ورودی دستور نام دکمه می‌باشد.

مقدار بازگشتی دستور، آرایه ای 2 عضوی با اعضای "Width" و "Height" می‌باشد.

xButton.GetState: با این دستور می‌توانید حالت UP یا Down بودن دکمه در هنگام فعال بودن toggle را بدست آورید. ورودی این کد نام شی است.

مقدار بازگشتی کد، عدد0، 1 یا 1- می‌باشد که نمایش دهنده UP، Down یا خطا می‌باشد.

xButton.GetText: متن موجود در xButton با استفاده از این دستور بدست می‌آید. مقدار ورودی نام دکمه است.

مقدار بازگشتی رشته ای است حاوی متن دکمه . در صورت بروز خطا رشته خالی بازگردانده می‌شود.

xButton.IsEnabled: با استفاده از این دستور می‌توانید فعال یا غیرفعال بودن دکمه را بدست آورید. مقدار ورودی نام دکمه است.

مقدار بازگشتی Trueبه معنای فعال بودن یا False به معنای غیرفعال بودن دکمه می‌باشد.

xButton.IsVisible: با استفاده از این دستور می‌توانید نمایش یا عدم نمایش xButton را بدست آورید. مقدار ورودی نام دکمه است.

مقدار بازگشتی true به نمایش یا false به معنای عدم نمایش دکمه می‌باشد.

xButton.SetEnabled: با این دستور می‌توانید دکمه را فعال یا غیر فعال کنید.ورودی دستور نام دکمه مورد نظر و همچنین عبارت بولین برای فعال یا غیرفعال کردن دکمه می‌باشد

مقدار بازگشتی **ندارد** .

xButton.SetImage: این دستور، تصویر دکمه را تغییر می‌دهد. نام شی و مسیر فایل ورودی‌های دستور هستند.

مقدار بازگشتی **ندارد** .

xButton.SetPos: با این کد می‌توانید دکمه را به محل دلخواه در صفحه منتقل نمایید.ورودی‌های این کد نام دکمه، عددی برای جای گیری در قسمت X (فاصله از چپ) و عدد دیگری برای جای گیری در Y (فاصله از بالا می‌باشد).

xButton.SetProperties: با این دستور می‌توانید خصوصیات جدیدی بر روی دکمه خود قرار دهید.ورودی‌های این تابع یکی نام دکمه مورد نظر و دیگری آرایه ای متشکل از خصوصیات دکمه است که این خصوصیات در xButton.GetProperties توضیح داده شده.

مقدار بازگشتی **ندارد** .

xButton.SetSize: با این دستور می‌توانید دکمه را به اندازه دلخواه خود تغییر دهید.ورودی‌های این کد نام دکمه، عددی برای عرض و عددی دیگر برای طول دکمه می‌باشد.

مقدار بازگشتی **ندارد** .

xButton.SetState: تغییر حالت دکمه به Up یا Down (در صورت فعال بودن Toggle) با استفاده از این دستور امکان پذیر می‌باشد.ورودی‌های این دستور عبارتند از نام دکمه و عدد 0 یا 1 برای نمایش Up یا Down .

مقدار بازگشتی **ندارد** .

xButton.SetText: با این دستور می‌توانید متن نمایش داده شده بر روی دکمه را تغییر دهید. ورودی‌های این دستور عبارتند از نام دکمه و متن جدید به صورت رشته.

مقدار بازگشتی **ندارد** .

xButton.SetVisible: کنترل نمایش یا عدم نمایش دکمه در نرم افزار هم با این دستور امکان پذیر است. ورودی‌های کد، نام دکمه و عبارتی بولین برای فعال یا غیر فعال کردن نمایش دکمه می‌باشد.

مقدار بازگشتی **ندارد** .

XML

این تابع برای کار با فایل‌های XML به کار می‌رود. دستورات زیر مجموعه این تابع به شرح زیر است:

XML.Count: این دستور برای شمارش تعداد عنصرهای موجود در یک مسیر از داده های XML به کار می‌رود و دارای دو پارامتر ورودی است که یکی برای دریافت مسیر داده های XML به کار می‌رود(XMLPath) و دیگری برای دریافت نام خانه ای که می‌خواهید عناصر آن شمرده شوند به کار می‌رود، در این قسمت می‌توانید از علامت * برای شمارش کل عناصر یک خانه استفاده نمایید.(ElementName)

مقدار بازگشتی این دستور به صورت یک متغیر عددی است .

مثال:

```
count = XML.Count("vasva3", "New_3");
```

XML.GetAttribute: این دستور برای فراخوانی مقدار اختصاص یافته به ویژگی به یک عنصر از داده های XML به کار می‌رود و دارای دو پارامتر ورودی است که یکی برای دریافت مسیر داده‌هایی که می‌خواهیم خاصیت آن را به دست آوریم (XMLPath) و دیگری برای دریافت نام ویژگی‌ای که می‌خواهید مقدار آن را فراخوانی کنید به کار می‌رود. (AttributeName)

مقدار بازگشتی این دستور به صورت متغیر رشته ای است .

مثال:

```
XML_Attribute = XML.GetAttribute("vasva3", "Forum");
```

XML.GetAttributeNames: این دستور برای فراخوانی نام‌های اختصاص یافته به ویژگی یک عنصر از داده های XML به کار می‌رود و دارای یک پارامتر ورودی است که برای دریافت مسیر داده‌هایی که می‌خواهیم نام خاصیت آن‌ها را به دست آوریم (XMLPath) به کار می‌رود.

مقدار بازگشتی این دستور به صورت متغیر آرایه ای می‌باشد .

مثال:

```
XML_Attribute_name = XML.GetAttributeNames("vasva3");
```

XML.GetElementNames: این دستور برای فراخوانی نام‌های اختصاص یافته به عناصر یک مسیر از داده های XML به کار می‌رود و دارای سه پارامتر ورودی است که اولی برای دریافت مسیر داده‌هایی که می‌خواهیم نام عناصر آن‌ها را به دست آوریم (XMLPath) به کار می‌رود؛ دومی دارای ویژگی بولین می‌باشد true: فراخوانی اسم و مسیر عناصر به صورت کامل، false: فقط اسم عناصر فراخوانی می‌شود؛ سومی هم دارای ویژگی بولین می‌باشد true: فراخوانی شماره و علامت جدا کننده‌ی عناصر false: فقط اسم عناصر فراخوانی می‌شود.

مقدار بازگشتی این دستور به صورت متغیر آرایه ای می‌باشد .

XML.GetElementXML: این دستور برای فراخوانی داده های XML یک مسیر خاص به کار می‌رود و دارای یک پارامتر ورودی است که برای دریافت مسیر داده‌هایی که می‌خواهیم داده های آن را به دست آوریم (XMLPath) به کار می‌رود.

مقدار بازگشتی این دستور به صورت متغیر رشته ای است .

XML.GetValue: این دستور برای فراخوانی مقدار یک عنصر از داده های XML به کار می‌رود و دارای یک پارامتر ورودی است که برای دریافت مسیر داده‌هایی که می‌خواهیم مقدار آن را به دست آوریم (XMLPath) به کار می‌رود.

مقدار بازگشتی این دستور به صورت متغیر رشته ای است .

مثال:

```
XML_Value = XML.GetValue("vasva3/New");
```

XML.GetXML: این دستور برای فراخوانی داده های یک فایل XML فراخوانی شده به کار می‌رود و هیچ پارامتر ورودی ندارد.

مقدار بازگشتی این دستور به صورت متغیر رشته ای است .

مثال:

```
str_XML = XML.GetXML();
```

XML.InsertXML: این دستور برای افزودن یک عنصر جدید به داده های یک XML فراخوانی شده به کار می‌رود. پارامترهای ورودی این دستور عبارتند از: XMLPath: برای دریافت مسیر داده‌هایی که می‌خواهیم یک عنصر جدید به آن اضافه کنیم به کار می‌رود؛ Text: متنی از نوع XML که جهت افزودن عنصر جدید به داده های یک XML فراخوانی شده به کار می‌رود. InsertionMode: که نحوه‌ی افزودن شدن متن را تعیین می‌کند:
XML.INSERT_BEFORE: افزودن متن به محل قبل از عنصر تعیین شده،
XML.INSERT_AFTER: افزودن متن به محل بعد از عنصر تعیین شده، XML.REPLACE: جایگزین نمودن عنصر جدید به جای عناصر قبلی.

این دستور مقدار بازگشتی ندارد.

XML.Load: این دستور برای فراخوانی محتویات یک فایل XML به حافظه‌ی موقت کامپیوتر به کار می‌رود.

این دستور مقدار بازگشتی ندارد.

XML.RemoveAttribute: این دستور برای حذف کردن یک ویژگی از یک عنصر از داده های XML فراخوانی شده به کار می‌رود. و دارای دو پارامتر ورودی می‌باشد که یکی برای دریافت مسیر داده‌هایی که می‌خواهیم خاصیت آن را حذف کنیم (XMLPath) و دیگری برای دریافت نام آن ویژگی که می‌خواهیم حذفش کنیم (AttributeName) به کار می‌رود.

این دستور مقدار بازگشتی ندارد.

XML.RemoveElement: این دستور برای حذف کردن یک عنصر از داده های XML فراخوانی شده به کار می‌رود. و دارای یک پارامتر ورودی می‌باشد که یکی برای دریافت مسیر داده‌هایی که می‌خواهیم عنصر آن را حذف کنیم (XMLPath) به کار می‌رود.

XML.Save: این دستور برای ذخیره کردن اعمال انجام شده بر روی داده های XML فراخوانی شده با حافظه‌ی موقت کامپیوتر به کار می‌رود و دارای یک پارامتر ورودی می‌باشد که برای دریافت محل ذخیره فایل XML به کار می‌رود (Filename).

این دستور مقدار بازگشتی ندارد.

XML.SetAttribute: از این دستور برای اختصاص دادن یک ویژگی به یک عنصر از داده های XML فراخوانی شده استفاده می‌کنیم. این دستور دارای سه پارامتر ورودی می‌باشد XMLPath: برای دریافت مسیر داده‌هایی که می‌خواهیم دادن یک ویژگی به یک عنصر از آن اضافه کنیم. AttributeName: برای دریافت نام ویژگی جدید برای عنصر مورد نظر AttributeValue: برای دریافت مقداری که می‌خواهیم به ویژگی جدید اختصاص دهیم.

این دستور مقدار بازگشتی ندارد.

XML.SetValue: از این دستور برای اختصاص دادن مقدار به یک عنصر از داده های XML فراخوانی شده به حافظه‌ی موقت کامپیوتر استفاده می‌نماییم. این دستور دارای سه پارامتر

ورودی می‌باشد: XMLPath: برای دریافت مسیر داده‌هایی که می‌خواهیم مقداری را به یک عنصر از آن اضافه کنیم. Value: مقدار مورد نظر جهت افزودن به عنصر WriteCDATA: دارای دو مقدار بولین می‌باشد: true: ذخیره کردن مقدار به صورت CDATA، false ذخیره کردن مقدار به صورت معمولی.

مثال:

```
XML.SetValue("vasva3/New_3", "javad", false);
```

این دستور مقدار بازگشتی ندارد.

XML.SetXML: از این دستور برای تنظیم کردن سند XML فراخوانی شده به حافظه‌ی موقت کامپیوتر استفاده می‌نماییم که دارای یک پارامتر ورودی برای درافت متن XML می‌باشد (Text).

این دستور مقدار بازگشتی ندارد.

Zip

این تابع برای کار با فایل‌های zip(فشرده) به کار می‌رود. دستورات زیر مجموعه این تابع به شرح زیر است:

Zip.Add: با استفاده از این دستور می‌توانید فایل را با فرمت zip فشرده کرده یا به فایل فشرده قبلی اضافه نمایید. ورودی‌های دستور عبارتند از :مسیر فایل zip، آرایه ای حاوی مسیر فایل‌هایی که قصد اضافه کردن به فایل فشرده را دارید (با * می‌توانید چند فایل را انتخاب کنید)، وضعیت شامل شدن شاخه‌ها (در صورت استفاده از *)، رمز عبور برای فایل، میزان فشرده سازی، تابع بازگشتی در هنگام اضافه کردن فایل‌ها و مقداری بولین برای وضعیت زیر شاخه‌ها در صورت استفاده از Wildcard (*)

مقدار بازگشتی **ندارد**.

Zip.Extract: این دستور برعکس دستور بالا عمل می‌کند. به این صورت که فایل(های) مورد نیاز را از حالت فشرده خارج می‌کند. مسیر فایل، آرایه نام فایل‌های مورد نیاز (*.* به معنای تمام فایل‌ها)، مسیر شاخه ای که فایل‌ها پس از استخراج در آنجا قرار گیرند، وضعیت قرارگیری شاخه‌ها (در صورت True بودن شاخه‌ها ساخته می‌شوند و در صورت False بودن تمام فایل‌ها در یک شاخه قرار می‌گیرند)، رمز عبور فایل، نوع نوشتن فایل‌ها در صورت وجود نام فایل در آن شاخه، تابع بازگشتی ورودی‌های دستور هستند.

مقدار بازگشتی **ندارد**.

Zip.GetContents: این دستور لیستی از فایل‌های داخل فایل فشرده را تهیه می‌کند .مسیر فایل zip و نوع نمایش شاخه‌ها ورودی‌های دستور هستند.

مقدار بازگشتی آرایه ای است حاوی نام فایل‌های موجود در فایل فشرده.

پیوست‌ها:

تنظیمات مربوط به زبان فارسی

برای آنکه بتوانید در محیط AMS فارسی تایپ کنید و هم چنین بتوانید هنگام اجرای پروژه خود به درستی نوشته های فارسی را مشاهده کنید می‌توانید از دو روش استفاده کنید.

روش اول: با استفاده از یک نرم افزار گرافیکی، متن مورد نظر تایپ و به صورت تصویر به Auto play وارد گردد.

روش دوم، مرحله 1:

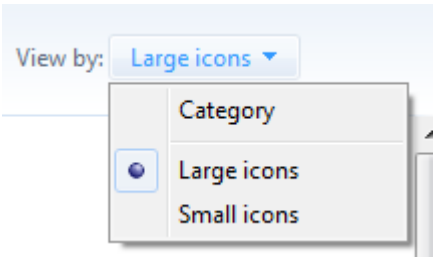
اعمال تنظیمات زبان فارسی بر روی سیستم عامل ویندوز:

ویندوز 7

1. بر روی منوی Start کلیک کنید.

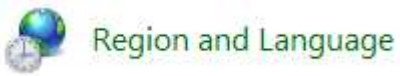
تصویر ضمیمه1

2. از منوی Start به Control Panel بروید.
3. از قسمت View by گزینه‌ی Large Icons را انتخاب کنید. (تصویر 0- 2)



تصویر ضمیمه 2

4. سپس روی گزینه‌ی Regional and Language کلیک کنید. (تصویر0- 3)



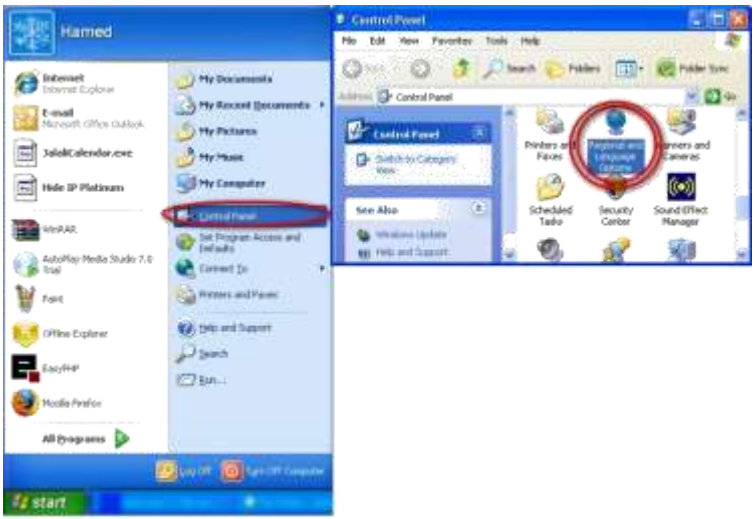
تصویر ضمیمه 3

5. در پنجره‌ی ظاهر شده از سربرگ Format و از منوی کشویی Format گزینه‌ی Persian را انتخاب کنید و روی دکمه‌ی Applay کلیک کنید.
6. سپس به سربرگ Administrative بروید و روی دکمه‌ی Change system locale… کلیک کنید و از منوی کشویی در پنجره‌ی جدید Persian را انتخاب کنید و روی دکمه‌ی Ok کلیک کنید و سپس با راه اندازی مجدد ویندوز تنظیمات خود را کامل نمایید.

ویندوز XP :

برای این کار کافی است CD ویندوز XP که از روزی آن ویندوز خود را نصب کرده‌اید را در درایو CD-ROM خود قرار دهید و دستورات زیر را طبق شکل دنبال نمایید:

ابتدا مطابق شکل از منوی Start گزینه Control Panel و سپس Regional and language Options را انتخاب نمایید.



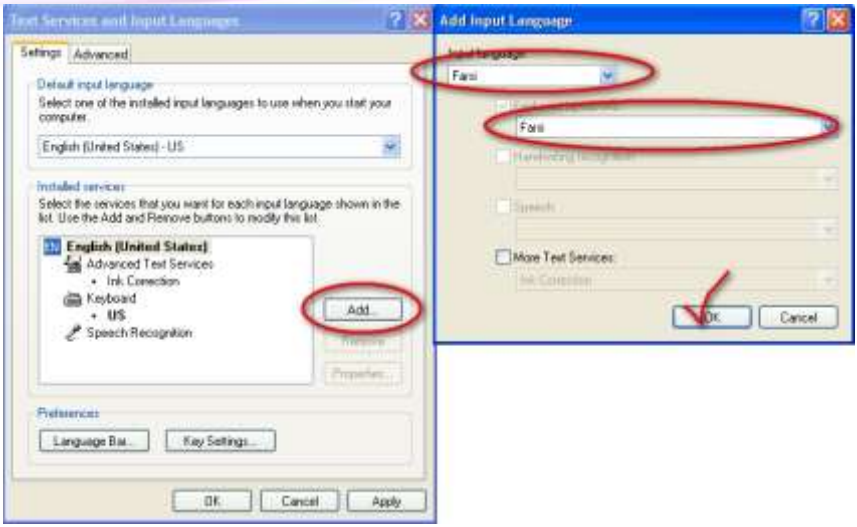
تصویر ضمیمه 4

اگر Panel Control شما به صورت منو نباشد می‌توانید به راحتی از گزینه Date , Time ... استفاده نمایید و در نهایت گزینه Add other languages را انتخاب نمایید.

تصویر ضمیمه 5

حال از پنجره ای که باز می‌شود ابتدا سربرگ Languages را انتخاب کنید. سپس دکمه Details را انتخاب کنید. حال دکمه Add را فشار دهید تا لیست زبان‌های جدیدی که می‌خواهید اضافه کنید ظاهر شوند. از لیست ظاهر شده گزینه Farsi را انتخاب کنید.

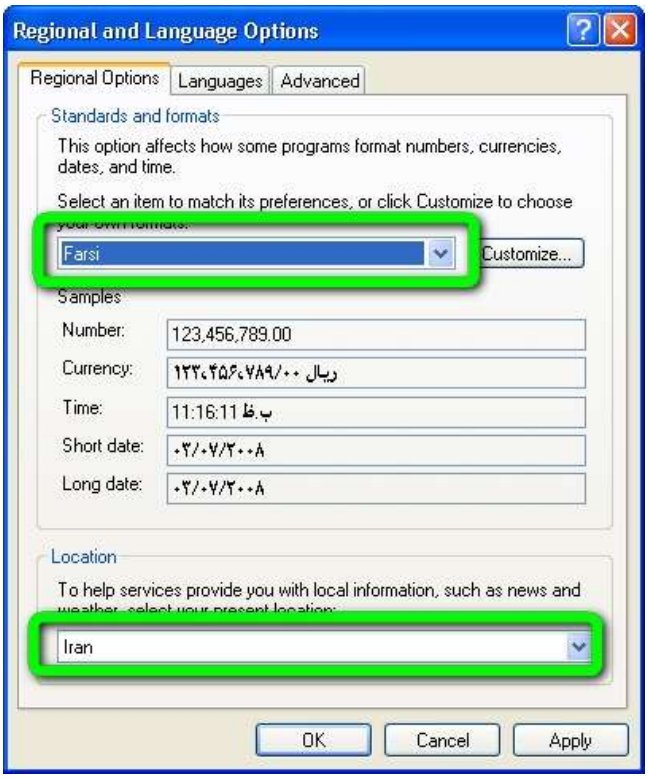
پس از این اعمال و تائید آن‌ها توسط دکمه OK ویندوز به صورت خودکار کیبورد فارسی استاندارد مایکروسافت را برای شما نصب خواهد کرد.



تصویر ضمیمه 6

برای انجام این کار شما یک راه دیگری نیز دارید. در پنجره Regional and Language Options گزینه Install Files for Complex Script and Right to Left Language (including Thai) را علامت بزنید در این هنگام از شما سؤالی پرسیده می‌شود که باید به آن جواب مثبت بدهید. حالا یک تعداد فایل از روی سی دی ویندوز کپی می‌شوند.

پس از اتمام کار به گزینه Regional Option رفته و از لیست می‌توانید زبان فارسی را انتخاب کنید.

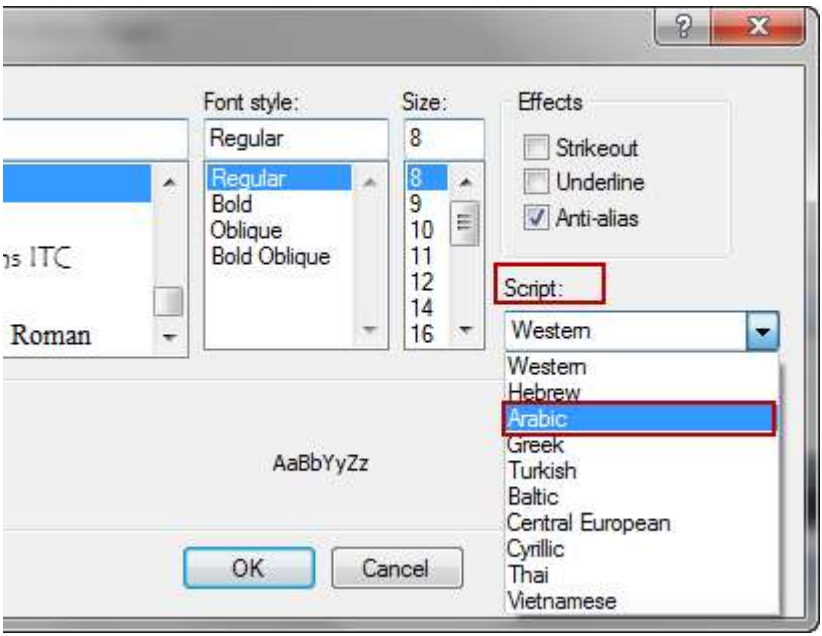


تصویر ضمیمه 7

پس از این اعمال و تائید آن‌ها توسط دکمه OK ویندوز به صورت خودکار کیبورد فارسی استاندارد مایکروسافت را برای شما نصب خواهد کرد.

روش دوم، مرحله‌ی 2:

پس از آن که تنظیمات زبان فارسی بر روی سیستم عامل ویندوز شما اعمال شد برای آن که بتوانید به راحتی در محیط AMS فارسی تایپ کنید می‌بایست هنگام انتخاب فونت دقت داشته باشید که فونت مورد نظر از الفبای زبان عربی نیز پشتیبانی کند. برای این کار می‌بایست هنگام انتخاب فونت مطابق تصویر زیر عمل نمایید:



تصویر ضمیمه 8

کد کلیدها

همان طور که در طول کتاب مطالعه کردیم هر کلید از صفحه کلید یک شماره‌ی منحصر به فرد دارد که به وسیله آن شماره می‌توان فهمید کاربر کدام کلید را فشرده است. جدول کد کلیدها به شرح زیر است:

نکته: کدهای ستون Decimal مد نظر می‌باشند.

Decimal کد دسیمال	Hex کد هگزا	Character کاراکترها
8	8	Backspace
9	9	Tab
13	D	Enter
16	10	Shift (both)
17	11	Ctrl (both)
18	12	Alt (both)
19	13	Pause
20	14	Caps Lock
27	1B	Esc
32	20	Spacebar
33	21	Page Up
34	22	Page Down
35	23	End
36	24	Home
37	25	(left arrow)
38	26	(up arrow)
39	27	(right arrow)
40	28	(down arrow)
45	2D	Insert
46	2E	Delete

48	30	0
49	31	1
50	32	2
51	33	3
52	34	4
53	35	5
54	36	6
55	37	7
56	38	8
57	39	9
65	41	A or a
66	42	B or b
67	43	C or c
68	44	D or d
69	45	E or e
70	46	F or f
71	47	G or g
72	48	H or h
73	49	I or i
74	4A	J or j
75	4B	K or k
76	4C	L or l
77	4D	M or m
78	4E	N or n
79	4F	O or o
80	50	P or p
81	51	Q or q
82	52	R or r

83	53	S or s
84	54	T or t
85	55	U or u
86	56	V or v
87	57	W or w
88	58	X or x
89	59	Y or y
90	5A	Z or z
91	5B	(left Windows key)
92	5C	(right Windows key)
93	5D	(application key - located between the right Windows and Ctrl keys on most keyboards)
96	60	0 (numpad with Num Lock on)
97	61	1 (numpad with Num Lock on)
98	62	2 (numpad with Num Lock on)
99	63	3 (numpad with Num Lock on)
100	64	4 (numpad with Num Lock on)
101	65	5 (numpad with Num Lock on)
102	66	6 (numpad with Num Lock on)
103	67	7 (numpad with Num Lock on)

104	68	8 (numpad with Num Lock on)
105	69	9 (numpad with Num Lock on)
106	6A	* (numpad)
107	6B	+ (numpad)
109	6D	- (numpad)
110	6E	. (numpad)
111	6F	/ (numpad)
112	70	F1
113	71	F2
114	72	F3
115	73	F4
116	74	F5
117	75	F6
118	76	F7
119	77	F8
120	78	F9
121	79	F10
122	7A	F11
123	7B	F12
144	90	Num Lock
145	91	Scroll Lock
186	BA	;
187	BB	=
188	BC	,
189	BD	-
190	BE	.

191	BF	/
192	C0	`
219	DB	[
220	DC	\
221	DD	]
222	DE	'

علایم

علایم و عبارت‌هایی که می‌توانیم در AMS مورد استفاده قرار دهیم عبارتند از :

or and
< > <= >= ~= ==
..
+ -
* / %
not # - (unary)
^

مثال برای استفاده از علایم:

```
a = 10;
a = (5 * 1) * 2;
a = 100 / 10;
a = 100 / (2 * 5);
a = 5 + 2;
b = a * 100;
twentythreepercent = 23 / 100;
neg = -29;
pos = -neg;
(3 * 200) > 500;
if "Peyman" ~= "Mohsen" then
....
end
if  "Ali" == " Ali " then
...
end
```

```
a = true;
b = false;
e = not b;
nLength = #"Hey";→ علامت # تعداد کاراکترها را بر می‌گرداند.
name = "javad".." hamed";
```

توابع بازگشتی

در هنگام کار با دستورات AMS به دستوراتی برخورد می‌کنیم که در خصوصیات خود خاصیتی به نام Callback Function دارند که در اصطلاح تابع بازگشتی نامیده می‌شوند یعنی این دستور می‌تواند یک تابع دیگر را در خود فراخوانی نماید و نتیجه‌ی کارهای خود را به صورت پارامترهایی به آن تابع بفرستد تا تابع مورد نظر با توجه به آن پارامترها دستورات داده شده را به اجرا در آورد.

دستوراتی که این قابلیت را دارا هستند عبارتند از :

1. File.Copy
2. File.Delete
3. File.Find
4. File.Install
5. File.Move
6. Folder.DeleteTree
7. Folder.Find
8. HTTP.Download
9. HTTP.DownloadSecure
10. HTTP.GetFileSize
11. HTTP.GetFileSizeSecure
12. Zip.Add
13. Zip.Extract

مثال برای توابع بازگشتی: کدی می‌نویسیم که مقدار کپی شده فایل را بر حسب کیلوبایت در یک شی Label نمایش دهد.

این کدها را در رویداد On Click یک xButton بنویسید:

```
function
Copy_file(Source,Destination,Copied,Total,FileCopied,FileTotal)
kb = Copied/1024;
Label.SetText("Label1", kb.." kb");
end
File.Copy(_SourceFolder.."\\*.*", "C:\\", true, true, false, true,
Copy_file);
```

همان طور که در مثال مشاهده می‌کنید دستور File.Copy می‌تواند 6 پارامتر را به تابع Copy_file ارسال نماید و تابع مورد نظر نیز می‌تواند از تمام آن‌ها یا یکی از آن‌ها در داخل خود استفاده نماید.

دقت داشته باشید که این پارامترها باید طبق مثال پشت سر هم نوشته شوند یعنی ردیف آن‌ها باید رعایت شود:

Source,Destination,Copied,Total,FileCopied,FileTotal

حال که با نحوه کار توابع بازگشتی آشنا شده‌اید پارامترهای تمامی دستوراتی که می‌توانند توابع بازگشتی را قبول کنند به ترتیب ذکر می‌کنیم:

- ✓ File.Copy

پارامترها: Source,Destination,Copied,Total,FileCopied,FileTotal

Source: مسیر فایلی که در حال کپی شدن است

Destination: مسیری که فایل دارد به آن جا کپی می‌شود

Copied: مقدار کپی شده‌ی کل را به صورت بایت بر می‌گرداند.

Total: حجم کل فایل‌هایی که باید کپی شوند را بر می‌گرداند.

FileCopied: مقدار کپی شده‌ی فایلی را که در حال کپی شدن است بر می‌گرداند.

FileTotal: حجم کل فایلی که در حال کپی شدن است را بر می‌گرداند.

- ✓ File.Delete

پارامترها: Source, Deleted, Total

Source: مسیر فایلی که در حال حذف شدن است.

Deleted: حجم فایل‌هایی که تاکنون حذف شده‌اند.

Total: حجم کل فایل‌هایی که باید حذف شوند را بر می‌گرداند.

✓ File.Find

این دستور در دو قسمت تابع بازگشتی قبول می‌کند:

- CallbackFunction با پارامتر CurrentPath

CurrentPath: مسیری که هم اکنون دارد جستجو می‌شود.

- FileFoundCallbackFunction با پارامتر FoundPath

FoundPath: مسیری فایلی که در جستجو یافت شده است.

✓ File.Install

این دستور نیز در دو قسمت تابع بازگشتی قبول می‌کند:

- ProgressCallbackFunction با پارامتر های :

Source, Destination, Copied, Total

Source: مسیر فایلی که در حال نصب شدن است.

Destination: مسیری که فایل دارد به آن جا نصب می‌شود.

Copied: مقدار نصب شده‌ی کل را به صورت بایت بر می‌گرداند.

Total: حجم کل فایل‌هایی که باید نصب شوند را بر می‌گرداند.

- OverwriteCallbackFunction با پارامترهای :

Source, Destination

Source: مسیر فایلی که در حال نصب شدن است.

Destination: مسیری که فایل دارد به آن جا نصب می‌شود.

✓ File.Move

پارامترها: Source, Destination, Copied, Total, FileCopied, FileTotal

Source: مسیر فایلی که در حال کپی شدن است.

Destination: مسیری که فایل دارد به آن جا کپی می‌شود

Copied: مقدار کپی شده‌ی کل را به صورت بایت بر می‌گرداند.

Total: حجم کل فایل‌هایی که باید کپی شوند را بر می‌گرداند.

FileCopied: مقدار منتقل شده‌ی فایلی را که در حال کپی شدن است بر می‌گرداند.

FileTotal: حجم کل فایلی که در حال منتقل شدن است را بر می‌گرداند.

✓ Folder.DeleteTree

پارامترها: Source, Deleted, Total

Source: مسیر فایلی که در حال حذف شدن است.

Deleted: تعداد فایل‌هایی که تا کنون حذف شده‌اند.

Total: حجم کل فایل‌هایی که باید حذف شوند را بر می‌گرداند.

✓ Folder.Find

پارامترها: CurrentPath

CurrentPath: مسیری که دارد جستجو می‌شود.

✓ HTTP.Download

پارامترها: BytesRead, FileSize, TransferRate, SecondsLeft, SecondsLeftFormat, Message

BytesRead: حجم دانلود شده فایل را به صورت بایت(عدد) برمی‌گرداند.

FileSize: حجم فایل را به صورت بایت(عدد) برمی‌گرداند.

TransferRate: سرعت دریافت بر حسب کیلوبایت بر ثانیه بر می‌گرداند.

SecondsLeft: مدت زمان (احتمالی) مورد نیاز برای دریافت فایل بر حسب ثانیه

SecondsLeftFormat: مدت زمانی احتمالی مورد نیاز برای دریافت فایل را به صورت MM:SS نمایش می‌دهد.

Message: یک رشته خالی و یا اطلاعاتی در مورد ارسال و دریافت اطلاعات را برمی‌گرداند.

✓ HTTP.DownloadSecure

پارامترها: BytesRead, FileSize, TransferRate, SecondsLeft, SecondsLeftFormat, Message

BytesRead: حجم دانلود شده فایل را به صورت بایت(عدد) برمی‌گرداند.

FileSize: حجم فایل را به صورت بایت(عدد) برمی‌گرداند.

TransferRate: سرعت دریافت بر حسب کیلوبایت بر ثانیه بر می‌گرداند.

SecondsLeft: مدت زمان (احتمالی) مورد نیاز برای دریافت فایل بر حسب ثانیه

SecondsLeftFormat: مدت زمانی احتمالی مورد نیاز برای دریافت فایل را به صورت MM:SS نمایش می‌دهد.

Message: یک رشته خالی و یا اطلاعاتی در مورد ارسال و دریافت اطلاعات را برمی‌گرداند.

✓ HTTP.GetFileSize

پارامترها: Message

Message: یک رشته خالی و یا اطلاعاتی در مورد ارسال و دریافت اطلاعات را برمی‌گرداند.

✓ HTTP.GetFileSizeSecure

پارامترها: Message

Message: یک رشته خالی و یا اطلاعاتی در مورد ارسال و دریافت اطلاعات را برمی‌گرداند.

✓ Zip.Add

پارامترها: String, Percent, Status

String: مسیر و نام فایلی که در حال فشرده شدن می‌باشد.

Percent: مقدار پیشرفت فشرده شدن فایل را بر می‌گرداند. .(عدد)

Status: وضعیت فشرده شدن فایل‌ها را بر می‌گرداند. .(عدد)

- ✓ Zip.Extract

پارامترها: String, Percent, Status

String: مسیر و نام فایلی که در حال خارج کردن از حالت فشردگی می‌باشد.

Percent: مقدار پیشرفت خارج کردن از حالت فشردگی را بر می‌گرداند.(عدد)

Status: وضعیت خارج شدن فایل‌ها از حالت فشردگی را بر می‌گرداند.(عدد)

نصب افزونه‌ها(Plugins)

برنامه AMS طوری طراحی شده است که قابلیت ارتقا دارد و می‌توان برای آن افزونه های جدیدی را برنامه نویسی کرد و سپس به آن اضافه کرد. این افزونه‌ها هم می‌توانند از نوع شی باشند و هم می‌توانند از نوع دستوری باشند.

افزونه های دستوری (Action Plugins)

افزونه‌هایی که از نوع دستوری هستند در مسیری مشابه مسیر زیر کپی می‌شوند و با اجرای دوباره‌ی برنامه AMS فعال می‌گردند:

C:\Program Files\AutoPlay Media Studio 8\Plugins\Actions

البته باید توجه داشته باشید که در هر پروژه ای که شما قصد دارید از افزونه های دستوری استفاده کنید می‌بایست از منوی Project>Plugins... افزونه‌ی مورد نظر را علامت گذاری کنید تا دستورات و توابع آن در برنامه فعال گردد.

اشیای افزونه ای (Objects>Plugins)

این افزونه‌ها وقتی به برنامه افزوده می‌شود از منوی Objects>Plugins قابل دسترسی هستند و مثل سایر اشیا قابل مشاهده می‌باشند.

اشیای افزونه ای در مسیری مشابه مسیر زیر کپی می‌شوند و با اجرای دوباره برنامه AMS فعال می‌گردند:

C:\Program Files\AutoPlay Media Studio 8\Plugins\Objects

متغیر محلی(local):

متغیر محلی متغیری است که با پایان یافتن یک دستور یا رویداد، از بین می‌رود و دیگر قابل استفاده نمی‌باشد.

```
local a= 10;
```

توجه: تمامی مثال‌های کتاب به صورت پروژه سورس باز در CD همراه کتاب ارایه شده‌اند.

کدهای نمونه:

در این قسمت از کتاب کدهای نمونه ای را برایتان آماده کرده‌ایم تا با استفاده از آن‌ها بتوانید پروژه های مفیدی را طراحی و تولید کنید و هم چنین با تأمل بر روی این کدها بتوانید دانش برنامه نویسی تان را بالا ببرید.

Extract فایل با نمایش میزان پیشرفت:

یک برچسب با نام Label1 برای نمایش مسیر فایل Zip و یک Progress با نام Progress1 برای نمایش میزان پیشرفت بسازید. کد زیر فایل را از حالت فشرده خارج می‌کند.

قرارگیری در Global Function:

```
function Zip_File(String, Percent, Status)
    Label.SetText("Label1", String);
   if (Status == ZIP_STATUS_MINOR) then
    Progress.SetCurrentPos("Progress1", Percent);
    Progress.SetText("Progress1", Percent.."%");
   end
end
```

قرارگیری در محل مورد نیاز برای Extract:

```
Zip.Extract("C:\\Test_File.Zip", {"*.*"}, "C:\\Learn AMS", true, true,
"", ZIP_OVERWRITE_NEVER, Zip_File);
```

*****

استفاده از Progress برای کنترل زمان آهنگ:

یک Progress با نام Progress1 بسازید

در قسمت On Timer:

```
Lenght = Audio.GetLength(CHANNEL_NARRATION);
CurPos = Audio.GetCurrentPos(CHANNEL_NARRATION);
```

```
Progress.SetCurrentPos("Progress1", CurPos / Lenght * 1000);
```

در قسمت On Click شی پروگرس :

```
MOUSE = System.GetMousePosition(true);
Lenght = Audio.GetLength(CHANNEL_NARRATION);
slider_pos = Progress.GetPos("Progress1");
slider_size = Progress.GetSize("Progress1");
SeekPosition = Lenght / slider_size.Width * (MOUSE.X -
(Disp.Width/2)+400- slider_pos.X);
Audio.Seek(CHANNEL_NARRATION, SEEK_SPECIFIC,
SeekPosition);
CurPos = Audio.GetCurrentPos(CHANNEL_NARRATION);
Progress.SetCurrentPos("Progress1", CurPos / Lenght * 1000);
```

یک شی دکمه به پروژه‌ی خود اضافه کنید و با استفاده از کد Audio.Load یک آهنگ در کانال CHANNEL_NARRATION فراخوانی نمایید و صفحه را هم فعال نمایید. سپس برنامه را اجرا کنید.

*****

کپی یک فایل (مثلاً با مسیر c:\test.jpg) به مسیر مورد نظر کاربر

```
result = Dialog.FolderBrowse("Please select a folder:", "c:\\")
if result ~= "CANCEL" and result ~="" then
File.Copy("C:\test.jpg", result, true, true, false, true, nil);
end
```

*****

استفاده از کلیدهای ترکیبی:

در قسمت On Key دستور زیر را وارد نمایید تا با زدن کلیدهای Ctrl + D برنامه بسته شود.

```
if (e_Key == 68 and e_Modifiers.ctrl ==true) then
Application.Exit(0);
end
```

*****

فرض کنید یک کمبوباکس داریم که در حالت اول متن "یک آیتم را انتخاب کنید" در آن قرار دارد.

گزینه های بعدی نام نرم افزار یا هر نام دیگری است که نیاز به دسترسی به فایل خاص دارد(اینجا نام فایل متنی) .پس گزینه اول نباید دستوری را اجرا کند.

در قسمت Text کمبوباکس نام مورد نیاز و در قسمت Data نام فایل مورد نظر را بنویسید.

از کد زیر در رویداد On Select شی کمبوباکس استفاده کنید تا متن فایل‌ها را در لیست باکس اضافه کند.

مثال برای مقادیر کمبوباکس :

	Item Text	Item Data
1	یک آیتم را انتخاب کنید	0
2	جمله شماره یک	S_1
3	جمله شماره دو	S_3
4	جمله شماره سه	S_4
5		
6		

تصویر ضمیمه 9

```
if e_Selection ~= 1 then
        ListBox.SetVisible("ListBox1", true);
        Name = ComboBox.GetItemData("ComboBox1",
e_Selection);
        ListBox.DeleteItem("ListBox1", -1);
```

```
        db_1 =
TextFile.ReadToTable(_SourceFolder.."\\AutoPlay\\Docs\\"..Name..".
txt");
        for i=1,Table.Count(db_1) do
                result = ListBox.AddItem("ListBox1", db_1[i],"");
        end
end
```

*****

ورود به برنامه با رمز عبور (پس از اولین ورود، نیاز به وارد کردن مجدد رمز عبور در دفعات بعد نیست) :

```
pass = Registry.GetValue(HKEY_CURRENT_USER,
"Software\\Load", "pass", true);
real_password = "Hamed_VS_Javad";
if pass ~= real_password then
   user_password =  Dialog.PasswordInput("Enter Data", "Your
answer:", MB_ICONQUESTION);
        if real_password ~= user_password then
        Application.Exit();
   else
    Registry.SetValue(HKEY_CURRENT_USER, "Software\\Load",
"pass", real_password, REG_SZ);
   end
end
```

*****

شروع مجدد برنامه از صفحه ای که از آن خارج شده :

On StartUp:

```
pagename = Registry.GetValue(HKEY_LOCAL_MACHINE,
"Software\\vasva3.com", "page", true);
```

```
if pagename then
Page.Jump(pagename);
end
```

On ShutDown:

```
result = Application.GetCurrentPage();
Registry.SetValue(HKEY_LOCAL_MACHINE,
"Software\\vasva3.com", "page", result, REG_SZ);
```

*****

نمایش پروسه های در حال اجرا در لیست باکس :

```
processes = System.EnumerateProcesses();
for j, file_path in pairs(processes) do
   file = String.SplitPath(file_path);
   result = ListBox.AddItem("ListBox1",file_path , "");
end
```

*****

نمایش تعداد هر عدد

این قطعه کد اعداد داخل Input را می‌شمارد. برای مثال عدد 2335 را گرفته و تعداد 2، تعداد 3 و تعداد 5 را در عدد نمایش می‌دهد.در انتها به لیست باکس اضافه می‌کند.

```
ListBox.DeleteItem("ListBox1", -1);
Table_A={};
for M = 0,9 do
Table_A[M] = 0
end
TEXTs = Input.GetText("Input1");
Nums = String.Length(TEXTs);
```

```
for G = 1,Nums  do
Rigs = String.Right(TEXTs, G);
RES = String.Left(Rigs, 1);
RES = String.ToNumber(RES);

Table_A[RES]=Table_A[RES]+1
end

for Count = 1, 9 do
        if Table_A[Count] ~= 0 then
        result = ListBox.AddItem("ListBox1", Count .." -> ".. Table_A[Count], "");
        end
end
```

*****

تبدیل ثانیه به ساعت کامل (ساعت، دقیقه و ثانیه)

Global Function :

```
function sec2Min(secs)
        local myMinutes = 0;
        local mySeconds = 0;
        local myTime =0;
        local myHour=0;
        if secs > 59 then
                myMinutes = Math.Floor(secs/60);
                mySeconds = secs-(myMinutes*60); -- Changes here.
                        if mySeconds < 10 then
                                mySeconds = "0"..mySeconds;
                        end
```

```
                myTime = myMinutes..":"..mySeconds;
        else
secs = String.ToNumber(secs);
                        if secs < 10 then
                                secs = "0"..secs;
                        end
                myTime = "0:"..secs;
        end
if myMinutes > 59 then
                myHour = Math.Floor(myMinutes/60);
                myMinutes = myMinutes-(myHour*60); -- Changes
here.
mySeconds = String.ToNumber(mySeconds);
                        if mySeconds < 10 then
                                mySeconds = "0"..mySeconds;
                        end
myMinutes = String.ToNumber(myMinutes);
                        if myMinutes < 10 then
                                myMinutes = "0"..myMinutes;
                        end
                myTime = myHour..":" ..myMinutes..":"..mySeconds;
end
        return myTime;
end
```

استفاده به صورت :

```
a=sec2Min(3752)
```

****

جابجا کردن دکمه در صفحه :

On Enter دکمه :

```
_OBJECT = this
```

On Leave دکمه :

```
_OBJECT = nil;
```

On Mouse Button صفحه:

```
if e_Type == 0 then
        _LeftClickDown = true
        if _OBJECT and not difference then
                difference = {X = e_X -
xButton.GetPos(_OBJECT).X,Y = e_Y -
xButton.GetPos(_OBJECT).Y}
        end
else
        _LeftClickDown = false
        difference = nil;
end
```

On Mouse Move:

```
if _LeftClickDown then
        if _OBJECT and difference then
                xButton.SetPos(_OBJECT, e_X-(difference.X), e_Y-
(difference.Y));
                Page.Redraw();
        end
end
```

*****

کنترل نصب بودن فلش با ورژن مورد نظر با پیغام سفارشی :

Dependencie را از منوی Project انتخاب کنید .تیک مربوط به Adobe Flash ... ActiveX را زده و Minimum Versionرا برابر 0 قرار دهید

حال در On StartUp بنویسید :

```
if _FlashVer <10 then
        نسخه فلش کمتر از ده می‌باشد . دستور مورد نیاز را جای این خط بنویسید --
else
        نسخه فلش صحیح است .در صورت نیاز دستور خود را بنویسید--
end
```

*****

نمایش تمام فایل‌های یک شاخه در لیست باکس :

```
folder = Dialog.FolderBrowse("Select the folder to display",
_SourceFolder);
if (folder ~= "CANCEL") then
        ListBox.DeleteItem("ListBox1", -1);
        files = File.Find(folder, "*.*", false, true);
        if (files ~= nil) then
                for i, filename in pairs(files) do
                        if (Folder.DoesExist(filename)) then
                                ListBox.AddItem("ListBox1",
"["..filename.."]", "");
                        else
                                ListBox.AddItem("ListBox1", filename,
"");
                        end
                end
        end
end
```

*****

یافتن لینک‌ها در یک فایل html :

```
webpage = TextFile.ReadToString("AutoPlay\\Docs\\Test.htm");
if webpage ~= "" then
	while start ~= -1 do
		start= String.Find(webpage, "href=\"", old, false);
		if start ~= -1 then
			start=start + 7;
			End = String.Find(webpage, "\"", start, false);
			link = String.Mid(webpage, start, End-start);
			result = ListBox.AddItem("ListBox1", link, "");
			old=End;
		else
			result = ListBox.AddItem("ListBox1",
"_____", "");
			old=1;
		end
	end
	start=0;
else
result = Dialog.Message("Notice", "Your WebPage Is Empty",
MB_OK, MB_ICONEXCLAMATION, MB_DEFBUTTON1);
end
```

*****

جستجو در لیست باکس به وسیله Input:

OnKey شی Input :

```
text_tosearch = Input.GetText("Input1");
```

```
idx = 1;
finded_text_tbl = {};
finded_data_tbl = {};
for i,v in pairs(item_text_tbl) do
	finded_result = String.Find(v, text_tosearch, 1, false);
	if finded_result ~= -1 then
		finded_text_tbl[idx] = v;
		finded_data_tbl[idx] = item_data_tbl[i];
		idx = idx+1;
	end
end
ListBox.DeleteItem("ListBox1", -1);
for i,v in pairs(finded_text_tbl) do
	ListBox.AddItem("ListBox1", finded_text_tbl[i],"");
end
```

: On Show

```
all_items = ListBox.GetCount("ListBox1");
item_text_tbl = {};
item_data_tbl = {};
for a=1, all_items do
	item_text = ListBox.GetItemText("ListBox1", a);
	item_data = ListBox.GetItemData("ListBox1", a);

	item_text_tbl[a] = item_text;
	item_data_tbl[a] = item_data;
end
```

*****

قرارگیری متن در پاراگراف با افکت تایپ :

دکمه:

```
str_txt = "Sample Text"
all_chars = String.Length(str_txt);
Page.StartTimer(25)
```

On Timer :

```
if num_char <= all_chars then

 str_char = String.Mid(str_txt, num_char, 1);

 str_par = Paragraph.GetText("Paragraph1");
 Paragraph.SetText("Paragraph1", str_par..str_char);
 num_char = num_char+1;
else
 Page.StopTimer()
end
```

*****

کنترل صحت لوح فشرده قرارگرفته در سیستم با کنترل یک فایل داخل آن:

```
drives = Drive.Enumerate();
for index , D in pairs(drives) do
D_Type = Drive.GetType(D);
  if D_Type == 5  then
   Find_MyFile = File.Find(D, "Project_.dll", false, false, nil, nil);
      if Find_MyFile then
    dr_name=D;
    break;
    else
```

```
   dr_name="NOT"
 end
 end
end
if dr_name == "NOT" then
result = Dialog.Message("Notice", "Please Insert Disk", MB_OK,
MB_ICONINFORMATION, MB_DEFBUTTON1);
Application.Exit(0);
end
```

*****

اعداد تصادفی در بازه‌ی مشخص بدون تکرار.

```
End=25
rnd={};
        rnd[1] = Math.Random(1,End);
        ag=1;
        while Table.Count(rnd)~=20 do
        t = Math.Random(1, End);
                        for i=1,Table.Count(rnd) do
                                if t==rnd[i] then
                                        ag=2;
                                end
                        end
                if ag ==1 then
                rnd[Table.Count(rnd)+1]=t;
                else
                ag=1 ;
```

```
                end
        end
for i=1,20 do
result = ListBox.AddItem("ListBox1", rnd[i], "");
end
```

*****

انتخاب آیتم بعدی در لیست باکس :

```
s = ListBox.GetSelected("ListBox1");
c = ListBox.GetCount("ListBox1");
if (s) then
        ListBox.SelectItem("ListBox1", s[1]+1);
        if s[1] == c then
                ListBox.SelectItem("ListBox1",1);
        end
else
        ListBox.SelectItem("ListBox1",1);
end
```

*****

فعال کردن صفحه زیر دیالوگ :

```
hwnd = Application.GetWndHandle();
DLL.CallFunction("User32.dll", "EnableWindow", hwnd..", "..1,
DLL_RETURN_TYPE_INTEGER, DLL_CALL_STDCALL);
```

*****

شفاف کردن صفحه :

```
function SetWindowLong(hWnd, nIndex, dwNewLong)
```

```
   return tonumber(DLL.CallFunction("User32.dll",
"SetWindowLongA", hWnd..", "..nIndex..", "..dwNewLong,
DLL_RETURN_TYPE_LONG, DLL_CALL_STDCALL))
end

function SetLayeredWindowAttributes(hWnd, crKey, bAlpha,
dwFlags)
   return tonumber(DLL.CallFunction("User32.dll",
"SetLayeredWindowAttributes", hWnd..", "..crKey..", "..bAlpha..",
"..dwFlags, DLL_RETURN_TYPE_LONG,
DLL_CALL_STDCALL))
end

function SetTrance(hWnd, opacity)
   SetWindowLong(hWnd, -20, 524288)
   SetLayeredWindowAttributes(hWnd, 0, opacity, 2)
end
```

روش استفاده :

```
Opacity = 75;
handle = Application.GetWndHandle();
SetTrance(handle, Opacity)
```

*****

راست چین کردن اشیای استاندارد :

کد در Global Function :

```
function GetWindowLong(hWnd, nIndex)
   return tonumber(DLL.CallFunction("user32.dll",
"GetWindowLongA", hWnd..", "..nIndex,
DLL_RETURN_TYPE_LONG, DLL_CALL_STDCALL))
end

function SetWindowLong(hWnd, nIndex, dwNewLong)
   return tonumber(DLL.CallFunction("user32.dll",
"SetWindowLongA", hWnd..", "..nIndex..", "..dwNewLong,
```

```
DLL_RETURN_TYPE_LONG, DLL_CALL_STDCALL))
end

function Right2Left()
   local Object_hwnd
   local ExStyle
   local Type
   local WS_EX_LAYOUTRTL = 4194304


   -- For Window
   Object_hwnd = Application.GetWndHandle()
   ExStyle = GetWindowLong(Object_hwnd ,-20)
   SetWindowLong(Object_hwnd, -20, ExStyle +
WS_EX_LAYOUTRTL)
objects = Page.EnumerateObjects( );
   if objects ~= nil then
   Application.SetRedraw(false)
   for index, object in pairs(objects) do
   Type = Page.GetObjectType(object)
   -- For ComboBox
   if Type == OBJECT_COMBOBOX then
     Object_hwnd = ComboBox.GetProperties(object).WindowHandle
   ExStyle = GetWindowLong(Object_hwnd ,-20)
   SetWindowLong(Object_hwnd, -20, ExStyle +
WS_EX_LAYOUTRTL)
   ComboBox.SetProperties(object, {ReadOrder =
READ_RIGHT_TO_LEFT})
   -- For Input
   elseif Type == OBJECT_INPUT then

   Input.SetProperties(object, {ReadOrder =
READ_RIGHT_TO_LEFT, Alignment = ALIGN_RIGHT})
   -- For ListBox
   elseif Type == OBJECT_LISTBOX then

   Object_hwnd = ListBox.GetProperties(object).WindowHandle
   ExStyle = GetWindowLong(Object_hwnd ,-20)
```

```
   SetWindowLong(Object_hwnd, -20, ExStyle +
WS_EX_LAYOUTRTL)
   ListBox.SetProperties(object, {ReadOrder =
READ_RIGHT_TO_LEFT})
   -- For RadioButton
   elseif Type == OBJECT_RADIOBUTTON then
      RadioButton.SetProperties(object, {TextAlignment =
ALIGN_RIGHT, ReadOrder = READ_RIGHT_TO_LEFT,
ButtonAlignment = BTN_ALIGN_RIGHT})
   -- For Tree
   elseif Type == OBJECT_TREE then
   Object_hwnd = Tree.GetProperties(object).WindowHandle
   ExStyle = GetWindowLong(Object_hwnd ,-20)
   SetWindowLong(Object_hwnd, -20, ExStyle +
WS_EX_LAYOUTRTL)
   Tree.SetProperties(object, {ReadOrder =
READ_RIGHT_TO_LEFT})
   end
   end
   Application.SetRedraw(true)
   end

end
```

کد در on preload صفحه :

```
Right2Left()
```

*****

فرمت اختصاصی:

```
Registry.SetValue(HKEY_CLASSES_ROOT,".vasva3","","VaSvA3_
File",REG_SZ);
Registry.SetValue(HKEY_CLASSES_ROOT," VaSvA3_File
","InfoTip","This File Crated By vasva3.com",REG_SZ);
Registry.SetValue(HKEY_CLASSES_ROOT,"VaSvA3_File","Never
ShowExt","",REG_SZ);
Registry.SetValue(HKEY_CLASSES_ROOT,"VaSvA3_File
```

```
","IsShortCut","",REG_SZ);
exe_path =_SourceFolder.."\\"..SourceFilename
Registry.SetValue(HKEY_CLASSES_ROOT,"VaSvA3_File\\DefaultIcon", "",exe_path..",0", REG_SZ);
Registry.SetValue(HKEY_CLASSES_ROOT,
"VaSvA3_File\\shell\\open\\command", "",""..exe_path.."".." ".."%1"",
REG_SZ);
```

exe_path = مسیر و اسم فایل اجرایی مورد نظر برای اجرای فایل‌هایvasva3

قسمت دوم یعنی برنامه از کجا تشخیص بدهد که شما چه فایلی از باز کرده‌اید و آن را اجرا کنید و یا ...

این تابع هم برای گرفتن آدرس فایل کلیک شده یا open شده

```
_CommandLineArgs[1]
```

*****

جلوگیری از کوچک شدن پنجره(Minimize) :

در GlobalFunctions

```
function GetWindowLong(hWnd,Index)
  return tonumber(DLL.CallFunction("User32.dll",
"GetWindowLongA", hWnd..", "..Index,
DLL_RETURN_TYPE_LONG, DLL_CALL_STDCALL))
end

function SetWindowLong(hWnd,Index,dwNewLong)
  return tonumber(DLL.CallFunction("User32.dll",
"SetWindowLongA", hWnd..", "..Index..", "..dwNewLong,
DLL_RETURN_TYPE_LONG, DLL_CALL_STDCALL))
end
```

در OnShow

```
SetWindowLong(Application.GetWndHandle(),-
16,GetWindowLong(Application.GetWndHandle(),-16)-131072)
```

*****

قرار دادنUninstall برای برنامه

برای قرار دادن Uninstall در Add/Remove Program در مسیر زیر:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall\Print File List Pro
```

متغیرهای زیر رو با مقدار مورد نظر برای برنامه ایجاد کنید:

```
"UninstallString"="C:\\Windows\\vasva3_Uninstaller.exe"
"DisplayName"="Esme narm Afzar"
"DisplayVersion"="1.2"
"Publisher"="VaSvA3.CoM"
"NoModify"=dword:00000001
"NoRepair"=dword:00000001
```

*****

بسته‌ی نرم افزاری(Package):

در این قسمت قصد داریم یک پروژه‌ی بسته‌ی نرم افزاری بسازیم که دارای امکانات زیر باشد:

- گروه بندی نرم افزارها
- لیست نرم افزارهای موجود در هر گروه
- توضیح در مورد نرم افزارها
- عکس برای هر نرم افزار و دیالوگ مخصوص برای نمایش عکس
- جستجو در لیست نرم افزارها
- جستجو در بین توضیحات
- دکمه های نصب و سریال و ...

روش ساخت:

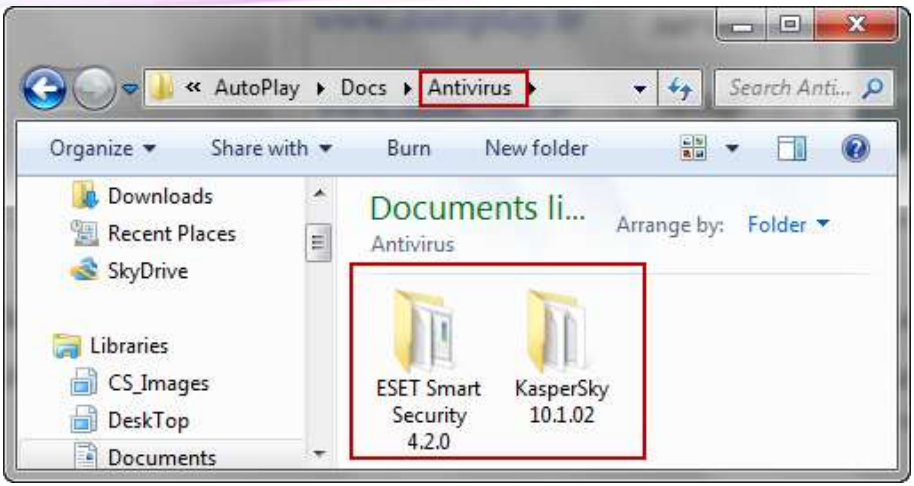
1. پروژه جدیدی با نام Package و با مشخصات زیر ایجاد کنید:

اندازه: 545 *790، یک عدد دیالوگ از نوع DialogEx و در آن یک شی Input قرار دهید به طوری که کل فضای دیالوگ را بپوشاند، یک عدد شی ComboBox، دو عدد شی Input، یک عدد شی ListBox، یک عدد شی Image، یک عدد شی Web، سه عدد دکمه از نوع xbutton، دو عدد شی Label و سه شی Shape برای کادر بندی کردن پروژه

پروژه شما می‌بایست مشابه تصویر زیر باشد:

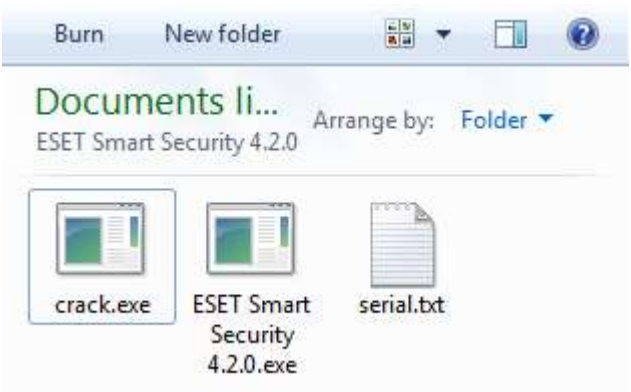
تصویر ضمیمه 10

2. نرم افزارهای خود را گروه بنده کنید به طوری که هر نرم افزار در پوشه‌ی گروه خود قرار بگیرد برای مثال: نرم افزارهای ضد ویروس در گروهی به نام Antivirus قرار داده شوند. توجه داشته باشید نرم افزارهای درون گروه می‌بایست در پوشه خاص خود قرار داده شوند.(مطابق با) پس از گروه بندی گروه‌ها را به پوشه‌ی Docs در مسیر پروژه‌ی خود منتقل کنید.



تصویر ضمیمه 11

فایل‌های کرک و سریال باید در کنار فایل اجرای گذاشته شوند برای مثال به تصویر زیر دقت کنید:



تصویر ضمیمه 12

نکته: فایل سریال نرم افزار باید از نوع متنی (txt) باشد.

نکته: فایل توضیحات برنامه باید از نوع فایل وب(htm) باشد، نامش باید help باشد و می‌بایستی طوری طراحی شود که فایل‌های عکس استفاده شده آن در پوشه ای به نام 1 قرار گیرند.

نکته: فایل عکس باید از نوع jpg باشد و نام آن‌ها می‌بایست image باشد.

نکته: فایل کرک باید از نوع exe باشد و نام آن‌ها می‌بایست crack باشد.

نکته: فایل نصب باید از نوع exe باشد و نام آن‌ها می‌بایست هم نام با پوشه خود باشد.

فایل‌های توضیحات و عکس برای آن که حجم کمتری را شغال کنند آن‌ها پس از ایجاد به روش زیر فشرده سازی کنید:

ابتدا پوشه ای به نام نرم افزار مورد نظر ایجاد کنید و سپس درون آن پوشه فایل عکس و توضیحات مربوط به آن نرم افزار را قرار دهید. برای مثال به تصاویر زیر دقت کنید:

تصویر ضمیمه 13

بعد از انجام این کارها فایل data.zip را در پوشه‌ی Docs موجود در پروژه‌ی خود ذخیره کنید.

نکته: دقت کنید که فایل عکس(image) در کنار فایل help قرار بگیرد

3. این مرحله که از مهم‌ترین مراحل کار می‌باشد ساخت یک بانک اطلاعاتی از نوع ini می‌باشد که شامل اطلاعات مربوط به نرم افزارها از جمله تعداد گروه‌ها، نام آن‌ها، نام نرم افزارهای موجود در هر گروه و کلمات کلیدی- توضیحی نرم افزار می‌باشد برای ساخت این فایل برنامه Notepad ویندوز را باز کنید و پس از وارد کردن اطلاعات طبق توضیحاتی که می‌آید آن را با نام و پسوند data.ini در پوشه‌ی Docs موجود در پروژه‌ی خود ذخیره کنید.

توضیح محتویات فایل ini:

متن فایل ini می‌بایستی مشابه با متن زیر باشد:

```
[Groups]
number=3
------------
[group_1]
name=All
number=4
------------
[group_2]
name=Antivirus
num_1=ESET Smart Security 4.2.0
search_1=ویروس کش، امنیت، نود32، حملات ویروسي، مخرب
num_2=KasperSky 10.1.02
search_2=ویروس کش، امنیت، کاسپرسکي حملات ویروسي، مخرب
number=2
------------
[group_3]
name=Multimedia
num_1=Multimedia Builder
search_1=اتوران، اتوپلي، چند رسانه اي، مالتي مديا، سي دي
num_2=AuotoPlay Media Studio
search_2=اتوران، مالتي مديا بيلدر، چند رسانه اي، مالتي مديا، سي دي
number=2
```

قسمت [Groups] نشان دهنده تعداد کل گروه‌ها می‌باشد. شامل یک متغیر به نام number است که مقدار آن برابر با 3 می‌باشد یعنی ما 3 گروه داریم.

قسمت [group_1] نشان دهنده اولین گروه می‌باشد که شامل دو متغیر به نام name و number می‌باشد. متغیر name نام گروه می‌باشد که ما آن را All قرار داده‌ایم یعنی شامل کل نرم افزارهای بسته‌ی نرم افزاری ما می‌باشد و متغیر number که نشان دهنده تعداد نرم افزارهای این گروه می‌باشد.

قسمت [group_2] نشان دهنده‌ی دومین گروه می‌باشد که متغیر name آن را برابر با Antivirus قرار داده‌ایم یعنی این گروه شامل نرم نام، توضیح و تعداد نرم افزارهای ضد ویروس می‌باشد. متغیر num_1 نشان دهنده‌ی نام اولین نرم افزار در این گروه می‌باشد در این قسمت می‌بایستی نام فایل به طور دقیق نوشته شود هم چنین توجه داشته باشید که نام فایل و نام پوشه‌ی محتوی آن می‌بایست یکسان باشند و متغیر search_1 نیز نمایش دهنده کلمات کلیدی – توضیحی نرم افزار موجود در قسمت num_1 به زبان فارسی می‌باشد و برای جستجوی کلمات کلیدی به درد می‌خورد. دو متغیر num_1 و search_1 در هر گروهی می‌توانند از عدد 1 تا هر شماره ای که نیاز داشتید افزایش یابند برای مثال: num_2، num_3، num_4 ... متغیر number نیز در این گروه نمایشگر تعداد نرم افزارهای این گروه می‌باشد.

قسمت [group_2] نیز می‌تواند تا هر اندازه که نیاز داشتید افزایش یابد برای مثال [group_3]، [group_4]، [group_5] ...

4. به پروژه خود برگردید و کدهای زیر را در جاهای مشخص شده وارد کنید:

Global Functions از منوی Project:

```
MB_RTL=1048576; -- راست چین کردن پنجره
MB_RIGHT=524288; --راست چین کردن متن
```

گرفتن حجم نرم افزار

```
function _GetSize(_Size)
local strFolder = File.GetSize(_Size);
Label.SetText("size", String.GetFormattedSize(strFolder,
FMTSIZE_AUTOMATIC));
end
```

جستجو بر اساس نام نرم افزار

```
function search_1()
txt = Input.GetText("Input1");
        if Selection == 1 then
        number = INIFile.GetValue(_SourceFolder.."\\AutoPlay\\Docs\\data.ini",
"Groups", "number");
                for i = 2 ,String.ToNumber(number) do
                 _number =
INIFile.GetValue(_SourceFolder.."\\AutoPlay\\Docs\\data.ini", "group_"..i,
"number");
                        for j = 1 ,String.ToNumber(_number) do
                          name =
INIFile.GetValue(_SourceFolder.."\\AutoPlay\\Docs\\data.ini", "group_"..i,
"num_"..j);
                          name_gp =
INIFile.GetValue(_SourceFolder.."\\AutoPlay\\Docs\\data.ini", "group_"..i,
"name");
                          find = String.Find(name, txt, 1, false);
                                if find ~= -1 then
                                  add = ListBox.AddItem("ListBox1",name,
name_gp);
                                end
                        end
                end
        else
number = INIFile.GetValue(_SourceFolder.."\\AutoPlay\\Docs\\data.ini",
"group_"..Selection, "number");
for i = 1 ,String.ToNumber(number) do
name = INIFile.GetValue(_SourceFolder.."\\AutoPlay\\Docs\\data.ini",
"group_"..Selection, "num_"..i);
name_gp = INIFile.GetValue(_SourceFolder.."\\AutoPlay\\Docs\\data.ini",
"group_"..Selection, "name");
find = String.Find(name, txt, 1, false);
if find ~= -1 then
add = ListBox.AddItem("ListBox1",name,  name_gp);
end
end
end
end
```

جستجو بر اساس نام کلمات کلیدی – توضیحی نرم افزار

```
function search_2()
txt = Input.GetText("Input2");
        if Selection == 1 then
        number = INIFile.GetValue(_SourceFolder.."\\AutoPlay\\Docs\\data.ini",
"Groups", "number");
                for i = 2 ,String.ToNumber(number) do
         _number = INIFile.GetValue(_SourceFolder.."\\AutoPlay\\Docs\\data.ini",
"group_"..i, "number");
                        for j = 1 ,String.ToNumber(_number) do
                search =
INIFile.GetValue(_SourceFolder.."\\AutoPlay\\Docs\\data.ini", "group_"..i,
"search_"..j);
                name =
INIFile.GetValue(_SourceFolder.."\\AutoPlay\\Docs\\data.ini", "group_"..i,
"num_"..j);
                name_gp =
INIFile.GetValue(_SourceFolder.."\\AutoPlay\\Docs\\data.ini", "group_"..i,
"name");
                find = String.Find(search, txt, 1, false);
                        if find ~= -1 then
                        add = ListBox.AddItem("ListBox1",name,
name_gp);
                        end
                end
        end
        else
        number = INIFile.GetValue(_SourceFolder.."\\AutoPlay\\Docs\\data.ini",
"group_"..Selection, "number");
                for i = 1 ,String.ToNumber(number) do
                search =
INIFile.GetValue(_SourceFolder.."\\AutoPlay\\Docs\\data.ini", "group_"..Selection,
"search_"..i);
                 name =
INIFile.GetValue(_SourceFolder.."\\AutoPlay\\Docs\\data.ini", "group_"..Selection,
"num_"..i);
                 name_gp =
INIFile.GetValue(_SourceFolder.."\\AutoPlay\\Docs\\data.ini", "group_"..Selection,
"name");
                find = String.Find(search, txt, 1, false);
                        if find ~= -1 then
```

```
                    add = ListBox.AddItem("ListBox1",name,  name_gp);
                    end
                end
            end
end
```

On Show از Page1:

گرفتن تعداد گروه‌ها

```
_Number = INIFile.GetValue(_SourceFolder.."\\AutoPlay\\Docs\\data.ini",
"Groups", "number");
```

خطایابی

```
error = Application.GetLastError();
if (error ~= 0) then
        Dialog.Message("Error", _tblErrorMessages[error], MB_OK,
MB_ICONEXCLAMATION);
        Application.Exit(0);
end
```

کد زیر هم خاصیت به روز رسانی شی ComboBox1 را غیر فعال می‌کند و باعث افزایش سرعت برنامه می‌گردد

```
ComboBox.SetUpdate("ComboBox1", false);
```

تعریف حلقه‌ی تکرار for برای گرفتن نام گروه‌ها

```
for i = 1 , String.ToNumber(_Number) do
```

گرفتن نام گروه‌ها بر اساس شماره‌ی شمارنده‌ی i

```
gp_name = INIFile.GetValue(_SourceFolder.."\\AutoPlay\\Docs\\data.ini",
"group_"..i, "name");
add = ComboBox.AddItem("ComboBox1", gp_name, "");
end
```

کد زیر هم خاصیت به روز رسانی شی ComboBox1 را فعال می‌کند تا محتویات جدید آن نمایش داده شوند

```
ComboBox.SetUpdate("ComboBox1", true);
```

انتخاب اولین گزینه از کمبوباکس

```
ComboBox.SetSelected("ComboBox1", 1);
```

On Select از شی ComboBox1:

حذف محتویات شی ListBox

```
ListBox.DeleteItem("ListBox1", -1);
```

بدست آوردن گزینه انتخاب شده و بررسی اینکه اگر گزینه‌ی 1 بود نام کل نرم افزارها را در لیست فراخوانی کند.

```
if e_Selection == 1 then
```

به دست آوردن تعداد گروه‌ها

```
number = INIFile.GetValue(_SourceFolder.."\\AutoPlay\\Docs\\data.ini", "Groups",
"number");
```

تعریف یک حلقه‌ی تو در تو برای فراخوانی نام گروه‌ها و فراخوانی نام نرم افزارها

```
        for i = 2 ,String.ToNumber(number) do
        _number = INIFile.GetValue(_SourceFolder.."\\AutoPlay\\Docs\\data.ini",
"group_"..i, "number");
                    for j = 1 ,String.ToNumber(_number) do
                    name =
INIFile.GetValue(_SourceFolder.."\\AutoPlay\\Docs\\data.ini", "group_"..i,
"num_"..j);
                    name_gp =
INIFile.GetValue(_SourceFolder.."\\AutoPlay\\Docs\\data.ini", "group_"..i,
"name");
```

افزودن نام نرم‌افزار به پارامتر ItemText و افزودن نام گروه هر نرم‌افزار به پارامتر ItemData از شی ListBox

```
                    add = ListBox.AddItem("ListBox1",name,  name_gp);
                   end
        end
else
number = INIFile.GetValue(_SourceFolder.."\\AutoPlay\\Docs\\data.ini",
"group_"..e_Selection, "number");
        for i = 1 ,String.ToNumber(number) do
         name = INIFile.GetValue(_SourceFolder.."\\AutoPlay\\Docs\\data.ini",
"group_"..e_Selection, "num_"..i);
         name_gp = INIFile.GetValue(_SourceFolder.."\\AutoPlay\\Docs\\data.ini",
"group_"..e_Selection, "name");
         add = ListBox.AddItem("ListBox1",name,  name_gp);
        end
end
ListBox.SelectItem("ListBox1", 1);
```

قرار دادن متغیرSelection برابر با مقدار متغیر رویدادی e_Selection برای استفاده در برنامه

```
Selection = e_Selection
```

On Key از شی Input1: جهت جستجو در نام نرم افزارها

```
ListBox.DeleteItem("ListBox1", -1)
ListBox.SetUpdate("ListBox1", false);
```

فراخوانی تابع جستجو در نام نرم افزار---

```
search_1()
ListBox.SetUpdate("ListBox1", true);
count = ListBox.GetCount("ListBox1");
```

با کد زیر بررسی می‌کنیم که آیا جستجوی ما نتیجه‌اش 0 است؟ اگر چنین بود کادر پیامی نشان داده می‌شود و سپس مقدار input1 برابر با جای خالی قرار می‌گیرد و دوباره تابع جستجوی نام فراخوانی می‌گردد تا تمام نام‌ها موجود در گروه حاضر فراخوانی شوند.

```
if count == 0 then
Dialog.Message("عبارت مورد نظر یافت نشد " ,"توجه: "..txt,
MB_OK+MB_RTL+MB_RIGHT, MB_ICONINFORMATION,
MB_DEFBUTTON1);
Page.SetFocus("Input1");
Label.SetText("Label1", "");
Input.SetText("Input1", "")
ListBox.SetUpdate("ListBox1", false);
search_1()
ListBox.SetUpdate("ListBox1", true);
ListBox.SelectItem("ListBox1", 1)
else
```

اگر نتیجه جستجو 0 نباشد یک گزینه از شی ListBox فراخوانی می‌شود.

```
ListBox.SelectItem("ListBox1", 1)
end
```

On Key از شی Input2: جهت جستجو در کلمات کلیدی نرم افزارها

```
ListBox.DeleteItem("ListBox1", -1)
ListBox.SetUpdate("ListBox1", false);
search_2()
ListBox.SetUpdate("ListBox1", true);
count = ListBox.GetCount("ListBox1");
if count == 0 then
Dialog.Message("عبارت مورد نظر یافت نشد " ,"توجه: "..txt,
MB_OK+MB_RTL+MB_RIGHT, MB_ICONINFORMATION,
MB_DEFBUTTON1);
Page.SetFocus("Input2");
Label.SetText("Label1", "");
Input.SetText("Input2", "")
ListBox.SetUpdate("ListBox1", false);
search_2()
ListBox.SetUpdate("ListBox1", true);
ListBox.SelectItem("ListBox1", 1)
```

```
else
ListBox.SelectItem("ListBox1", 1)
end
```

On Select از شی ListBox1:

بدست آوردن شماره انتخاب شده از لیست

```
sq = ListBox.GetSelected("ListBox1");
```

بدست آوردن متن ItemData گزینه‌ی انتخاب شده از لیست

```
gp_n = ListBox.GetItemData("ListBox1", sq[1]);
Label.SetText("Label2", gp_n);
```

بدست آوردن متن ItemText گزینه‌ی انتخاب شده از لیست

```
sa = ListBox.GetItemText("ListBox1", sq[1]);
name_g = Label.GetText("Label2");
size =
File.GetSize(_SourceFolder.."\\AutoPlay\\Docs\\"..name_g.."\\"..sa.."\\"..sa..".txt");
```

گرفتن حجم نرم افزار انتخاب شده با استفاده از تابع _GetSize

```
_GetSize(_SourceFolder.."\\AutoPlay\\Docs\\"..name_g.."\\"..sa.."\\"..sa..".txt")
```

استخراج کردن محتویات فایل فشرده شده مربوط به نرم افزار انتخاب شده در پوشه تمپ

```
Zip.Extract(_SourceFolder.."\\AutoPlay\\Docs\\data.zip", {sa.."\\*.*"},
_TempFolder.."\\hamseda04_04_04", false, false, "",
ZIP_OVERWRITE_ALWAYS, nil)
Zip.Extract(_SourceFolder.."\\AutoPlay\\Docs\\data.zip", {sa.."\\1\\*.*"},
_TempFolder.."\\hamseda04_04_04\\1", false, false, "",
ZIP_OVERWRITE_ALWAYS, nil)
```

فراخوانی توضیحات و عکس از پوشه تمپ

```
Web.LoadURL("Web1", _TempFolder.."\\hamseda04_04_04\\help.htm");
Image.Load("Image1", _TempFolder.."\\hamseda04_04_04\\image.jpg");
```

جستجو برای فایل کرک نرم افزار

```
filefind = File.Find(_SourceFolder.."\\AutoPlay\\Docs\\"..name_g.."\\"..sa,
"crack.exe", false, false, nil, nil);
```

اگر موجود بود دکمه‌ی مربوط به کرک فعال و در غیر این صورت غیر فعال می‌شود

```
if (filefind) then
xButton.SetEnabled("xButton3", true);
else
xButton.SetEnabled("xButton3", false);
end
```

جستجو برای فایل سریال نرم افزار

```
filefinds = File.Find(_SourceFolder.."\\AutoPlay\\Docs\\"..name_g.."\\"..sa,
"serial.txt", false, false, nil, nil);
```

اگر موجود بود دکمه‌ی مربوط به سریال فعال و در غیر این صورت غیر فعال می‌شود

```
if (filefinds) then
xButton.SetEnabled("xButton1", true);
else
xButton.SetEnabled("xButton1", false);
end
```

On Click از xButton1 برای فراخوانی سریال نرم افزار در متغیر:

```
ser =
TextFile.ReadToString((_SourceFolder.."\\AutoPlay\\Docs\\"..name_g.."\\"..sa.."\\se
rial.txt");
DialogEx.Show("Dialog1", true, nil, nil);
```

On Show از DialogEx برای نمایش سریال در آن:

```
Input.SetText("Input1", ser);
```

On Click از xButton2 برای اجرای فایل نصب نرم افزار انتخاب شده:

```
Shell.Execute(_SourceFolder.."\\AutoPlay\\Docs\\"..name_g.."\\"..sa.."\\"..sa..".exe",
"open", "", "", SW_SHOWNORMAL, false);
```

On Click از xButton3 برای اجرای فایل کرک نرم افزار انتخاب شده:

```
Shell.Execute(_SourceFolder.."\\AutoPlay\\Docs\\"..name_g.."\\"..sa.."\\crack.exe",
"open", "", "", SW_SHOWNORMAL, false);
```

On Click از Image1 برای نمایش عکس نرم افزار انتخاب شده:

```
Dialog.SplashImage(_TempFolder.."\\hamseda04_04_04\\image.jpg", 5, true);
```

5. حال پروژه را با فشردن کلید F5 اجرا کنید تا نتیجه‌ی کار خود را مشاهده نمایید. شما هم اکنون یک بسته نرم افزاری تولید کرده‌اید که می‌توانید با کلیک کردن روی دکمه‌ی Setup، نرم افزار انتخاب شده را نصب کنید، با کلیک روی دکمه‌ی Serial شماره سریال نرم افزار مورد نظر را مشاهده نمایید و با کلیک روی دکمه‌ی Crack، کرک نرم افزار را نصب کنید. شما می‌توانید با کار کردن روی گرافیک این پروژه جذابیت بسیار زیبایی به ان ببخشید و از مزایای آن برخوردار شوید.

خطاها:

هر خطایی که در پروژه های ساخته شده با AMS رخ می‌دهد دارای شماره خاصی است که با توجه به آن شماره‌ها می‌توان به نوع خطای مورد نظر پی برد. در این قسمت به خطاهای مهم با ذکر توضیح آن‌ها می‌پردازیم:

خطای شماره (9999)اشاره به خطاهایی دارند که در برنامه تعریف نشده‌اند.

خطاهای شماره 1000 تا 1099 خطاهای مربوط به فایل (File) می‌باشند

خطاهای شماره1100 تا 1199 خطاهای مربوط به اشیاء عمومی برنامه(Generic Object) می‌باشند.

خطاهای شماره 1200 تا 1299خطاهای مربوط به صوت (Audio) می‌باشند.

خطاهای شماره 1300 تا 1399 مربوط به صفحه می‌باشند(Page).

خطاهای شماره1400 تا 1499 خطاهای مربوط به کادر محاوره ای (Status Dialog) می‌باشند.

خطاهای شماره 1500 تا 1599 خطاهای مربوط به فایل‌های INI می‌باشند.

خطاهای شماره 1600 تا 1699 خطاهای مربوط به رجیستری(Registry) می‌باشند.

خطاهای شماره 1700 تا 1799 خطاهای مربوط به لیست باکس (ListBox) می‌باشند.

خطاهای شماره 1800 تا 1899 خطاهای مربوط به سیستم (System)هستند.

خطاهای شماره 1900 تا 1999 خطاهای مربوط به فایل‌های متنی (Text File) می‌باشند.

خطاهای شماره 2000 تا 2099 خطاهای مربوط به پنجره‌ها(Window) هستند.

خطاهای شماره 2100 تا 2199 خطاهای مربوط به درایو (Drive) می‌باشند.

خطاهای شماره 2200 تا 2299 خطاهای مربوط به عملیات بر روی پوشه‌ها(Folder Action) می‌باشند.

خطاهای شماره 2300 تا 2399 خطاهای مربوط به عملیات بر روی شل (Shell Action)می‌باشند.

خطاهای شماره 2400 تا 2499 خطاهای مربوط به فایل‌های کتابخانه ای (DLL) می‌باشند.

خطاهای شماره 2500 تا 2599خطاهای مربوط به فایل‌های اینترنتی (HTTP) هستند.

خطاهای شماره2600 تا 2699 خطاهای مربوط به فایل‌های فشرده (ZIP File) می‌باشند.

خطاهای شماره 2700 تا 2799 خطاهای مربوط به کادر محاوره ای (Dialog) می‌باشند.

خطاهای شماره 2800 تا 2899 خطاهای مربوط به دستورات کاربردی(Application) می‌باشند

خطاهای شماره 2900 تا 2999خطاهای مربوط به توابع درونی برنامه(Internal Function) هستند.

خطاهای شماره 3000 تا 3199 خطاهای مربوط به متغیرهای رشته ای(String) می‌باشند.

خطاهای شماره 3200 تا 3299 خطاهای مربوط به عملیات ریاضی و منطقی(Math) هستند.

خطاهای شماره 3300 تا 3399 خطاهای مربوط به اشیاء درختی (Tree) می‌باشند.

خطاهای شماره 4200 تا 4299 خطاهای مربوط به فایل‌های نصبی(MSI) می‌باشند.

خطاهای شماره 4300 تا 4399 خطاهای مربوط به فایل‌های متنی(RichText) می‌باشند.

خطاهای شماره 4400 تا 4499 خطاهای مربوط به نمایش اسلاید (SlideShow) می‌باشند.

خطاهای شماره 5000 تا 5499 خطاهای مربوط به صفحات از نوع محاوره ای(DialogEx) هستند.

خطاهای شماره 6000 تا 6099 خطاهای مربوط سرویس‌ها(Service) هستند.

خطاهای شماره 7000 تا 7099 خطاهای مربوط به فایل‌های تصویری(QuickTime) هستند.

خطاهای شماره 8000 تا 8099 خطاهای مربوط به فایل‌های متنی پی دی اف(PDF) هستند.

خطاهای شماره 12400 تا 12499 خطاهای مربوط به گرید یا جدول(Grid) می‌باشند.

خطاهای شماره 34000 تا 34099 خطاهای مربوط به رمز نگاری(Crypto) می‌باشند.

خطاهای شماره 37000 تا 37099 خطاهای مربوط به فایل‌های نشانه گذاری متن (XML) می‌باشند.

خطاهای شماره 77000 تا 77099 خطاهای مربوط به کمبوباکس (ComboBox) می‌باشند.

فهرست

www.SoftGozar.Com