آموزش اکشن اسکریپت ۳ از سطح صفر

Learning ActionScript 3.0

گردآوری:
سید محمد جواد نیکوکار

Adobe

مقدمه:

اکشن اسکریپت به صورت قوی از نسخه ۴ فلش به بعد مطرح شد. از اون موقع تا حالا مطالب بسیار زیادی در مورد اکشن نوشته شده و نوشته خواهد شد. در مورد اکشن اسکریپت ۲ در ایران کارهای خوبی صورت گرفت و برنامه نویسای خوبی در ایران پیدا شدن که به این زبان تسلط کافی پیدا کردن و نمونه کارهای خوبی هم از اونا ارائه شد. ولی باز هم برنامه فلش با اینکه بین ایرانی ها برنامه معروف و محبوبیه با قابلیتهای واقعی خودش شناسانده نشد و از نظر من هنوز هم نشده.

بنده از فلش ۵ با این برنامه آشنا شدم و از اون موقع جسته و گریخته با این نرم افزار درگیر شدم و مثل خیلی های دیگه اوایل تنها چیزی که تمام حواسم بهش معطوف بود تایم لاین فلش بود و تنها کاری که بلد بودم ساخت کلیپ های انیمیشنی بود ولی کم کم که بیشتر باهاش کار کردم دیدم کارای خیلی بیشتری میشه با این نرم افزار استفاده کرد. از اون موقع بود که آروم آروم شروع کردم به یادگیری action script. از همون موقع یکی از دغدغه های من منبعی برای یادگیری کامل و اصولی این بخش از فلش بود ولی متاسفانه مطالب حول این زمینه نایاب بودن و برای هر کار کوچیکی که داشتم مجبور بودم ساعتها و یا حتی روزها وقت بذاریم و در اکثر موراد با شکست مواجه می شدم. تا اینکه بحث forum ها توی ایران پا گرفت و تا حدی مشکلات کمتر شد.

ولی متاسفانه به دلیل کمبود منابع من اکشن رو صحیح و اصولی یاد نگرفتم(مثل خیلی چیزای دیگه) و الزاما هر موقع هرجا سوالی برلم پیش میومد می رفتم دنبالش و فقط به فکر پیدا کردن جوابش و رفع مشکل بودم و برام مهم نبود که مشکلم رو چجوری دارم حل می کنم. فقط مشکل حل بشه بقیش با خدا.

با پیدایش as3 کلا مجبور شدم بزنم گاراژ و فهمیدم که باید آستینم رو بالا بزنم یه کاری بکنم. این بود که راه افتادم تا اکشن و کلا اصول برنامه نویسی رو از پایه شروع کنم و به وضعیت اطلاعات به هم ریختم یه سر وسامونی بدم این شد که شروع کردم به یادگیری اکشن اسکریپت ۳ از سطح صفر. اما هرچی سعی کردم یه مرجع ایرانی پیدا کنم که درست و حسابی اون رو آمورش داده باشه پیداش نکردم حتی تو فروم هام جایی نبود که به صورت اصولی و پایه مفاهیم یاد داده شده باشن . فرم ها محدود شده بودن به پاسخ های جزیی به سوالات و اگرم آموزش بود آموزهاش پراکنده مثلا آموزش ساخت mp3 player که برا یه کاربر صفر مناسب نبود. انگار کسایی که بلد بودن نمی خواستن به بقیه یاد بدن . شایدم وقت نداشتن که یاد بدن. این بود که تصمیم گرفتم خودم دست به کار بشم و یادگیری رو شروع کنم. پس یه کتاب درست و درمون گرفتم و شروع کردم به خوندن و یاد گرفتن.

دیدم حالا که مجبورم کتاب رو بخونم و ترجمش کنم بهتره درس به درس بزارم اینجا که یه جورایی سایت مرجعه و خیلی هم بهش مدیونم تا علاوه بر اینکه زکات علمم رو به جا آورده باشم انگیزه ای بشه برای این که کارم رو جدی تر و مقید تر انجام بدم.

توجه:

۱- پی دی افی که در اختیار دارین جمع آوری مطالبیه که تو سایت مجید آنلاین می نویسم. بالتبع از نظر ویرایشی پر از مشکله! فعلا فقط هدف جمع آوری مطالبه ، انشاءالله با تکمیل مباحث ویرایش هم انجام خواهد گرفت.

۲- با کامل شدن هر بخش ، آن بخش به پی دی اف اضافه خواهد شد.

۳- انتشار و استفاده از این مطالب در سایتهای دیگر به شرط ذکر نام نویسنده مانعی ندارد.

معرفی action script و تاریخچه آن:

برای شروع طبق روال همیشگی آغاز هر آموزش،بهتره که یه نگاه اجمالی به actionscript و تاریخچه اون بندازیم.

اولین نسخه برنامه فلش توسط شرکت ماکرومدیا در سال ۱۹۹۶ ارائه شد. فلش در نسخه های اولیه با هدف ایجاد برنامه ای برای ساخت انیمیشن توسط کامپیوتر شکل گرفت ولی بعدها به صورت حریصانه (و البته قدرتمندانه!) گامهای بعدی خودش رو در زمینه هایی هچون وب ، پروژه های مالتی مدیا ، موبایل و حتی برنامه های دسکتاپ هم برداشت. برنامه نویسی به شکل ساده از همون نسخه های اول با فلش همراه بود.

از نسخه های ۱ تا ۳ نام زبان برنامه نویسی فشل، "لینگو لایت" بود. ولی این زبان از نسخه ۴ به بعد به " action script " تغییر نام داد تا با طی روند رو به رشدش خودش در نسخه های بعدی به action script 3 رسید.

- ✓ خوب حالا ببینیم اصلا action script به چه دردی می خوره؟

با چند مثال ساده شروع می کنیم؟

۱) فرض کنید در حال ساخت یه انیمشین هستید و هنرپیشه نقش اول شما (مثلا فردین) تو سکانس پایانی توسط آدم بدای فیلم مورد حمله قرار می گیره و یه تیر بهش شلیک میشه. شما می خواین برا این که تاثیر بیشتری روی بینندتون بذارین صحنه ای رو که تیر به هنرپیشه برخورد می کنه رو چند بار به طور سریع دنبال هم تکرار کنید . اولین راهی که به ذهن می رسه اینه که اون صحنه رو یه بار درست کنید و فریم های مربوط به اون رو ۳ - ۴ بار دنبال هم کپی و paste کنین. شاید در نظر اول راه جالی باشه اما فکر کنید شما می خواید این فیلم رو روی یه سایت قرار بدین و بیننده های شما یوزرهای ایرانی با اینترنت dialup هستن. این کار شما باعث چی میشه؟ خوب معلومه افزایش حجم و بالتبع کاهش سرعت لود!!! تازه ممکنه شما کارگردان خوش ذوقی باشین و بخواین از این صحنه های تاثیر گذار زیاد تو فیلمتون داشته باشین! اون وقته که دیگه واویلا! خوب حالا اینجا می تونین از یه زبانی به نام action script استفاده کنین. به این صورت که تو اون صحنه ی مربوط به تیر خوردن اون فریم هایی که قراره تکرار بشن رو در انتهاشون بنویسین " اگه به این نقطه رسیدی برگرد به فریم اول مربوط به قسمت تیر خوردن و این کار رو

۳ بار انجام بده" به همین راحتی با یه دستور کوچیک شما کار چندصد یا حتی چندین هزار فریم رو انجام می دین!!!

۲) شما در حال ساخت یه سی دی مالتی مدیا برای معرفی محصولات یک شرکت هستید.

مثلا پوشک بچه !!! می خواید با کلیک بر روی یک محصول ویژگیهای اون رو روی صفحه نمایش بدین. اینجاست که باز هم باید از action script استمداد بطلبید و با اون به برنامه فلش حالی کنین که "اگر روی این دکمه کلیک شد این صفحه رو نشون بده"

۳) شما برای یه شرکت یه وبسایت با فلش طراحی کردین. مدیر شرکت از شما می خواد که در قسمت تماس با یه فرم ایجاد کنید که مشتریها بتونن سفارشات خودشون رو بصورت آنلاین از طریق فرم به شرکت بدن. باز هم کار شما به action script گیر می کنه و شما بوسیله این زبان باسد به برنامه فلش بگین که " اگر قسمت های سفارش فرم پر بود و کلید ارسال زده شد اطلاعات موجود در فرم رو بفرست به صندوق میل شرکت"

و خیلی چیزای دیگه که در ادامه بهشون خواهیم پرداخت. لپ کلام هر موجودی یه زبونی داره که بشه باهاش ارتباط برقرار کرد زبان صحبت کردن و دستور داده به برنامه فلش هم action script هست. همین!!!

اگر اشتباه نکنم، امتیاز فلش از نسخه ۷ (یا ۸) توسط شرکت adobe از شرکت macromedia خریده شد و روند صعودی فلش سرعت بیشتری گرفت و همچنین برنامه ها و تکنولوژی های وابسته جدیدی نظیر flex، air، flash lite هم معرفی شدند که به محبوبیت و کاربردی تر شدن این برنامه کمک کردند.

شاید بتوان مهمترین تحول ایجاد شده توسط adobe را برای فلش ارائه نسخه سوم action script دانست.

یکی از ویژگیهای زبان ac on script 3 اینه که بر خلاف دو نسخه قبلی اجازه ماست مالی رو به کاربر نمی ده!!! و برنامه نویس رو مجبور می کنه تا برنامه خودش رو کاملا اصولی و تو یه چارچوب معین بنویسه. اینکار باعث پدید اومدن ویژگیهای بسیار خوبی میشه که انشاء الله اگه خدا عمری بده تو جلسه های آینده باهاشون آشنا خواهیم شد.

syntax(گرامر) این زبان بسیار شبیه به گرامر زبانهای معروف برنامه نویسی نظیر java و c# هست. بنابر این کسایی که قبلا با این زبان ها آشنایی داشتن می تونن به راحتی با این زبان ارتباط برقرار کنن

ویژگی بعدی این زبان اینه که مثل همون دو تا زبانی که در بالا بهشون اشاره شد شی گرا است (object-oriented) انشاء الله در درس های آینده مفاهیم شی گرایی(oop --- مخفف object oriented programing) رو در موردشون بحث خواهیم کرد.

هسته زبان ac on script 3 بر پایه خصوصیات زبان ECMAScript(4th edi on) نوشته شده. زبان معروف اسکریپتینگ javascript هم از ECMAScript برای هسته زبان خودش استفاده می کنه. این استفاده مشترک باعث شده شرکت Adobe به شرکت موزیلا (توسعه دهندگان مرورگرفایرفاکس) پیشنهاد کنه در نسخه ۳ این مرورگر javascript 2 رو با کدهایی بر مبنای action script پیاده سازی کنه. والا دیگه خبر ندارم که موزیلا این کار رو کرد یا نه؟!!

action script می تونه روی سه نوع محیط کاربری استفاده بشه Adobe AIR, Flash Player,Flash Lite (البته فلش لایت ac on script 3 پشتیبانی نمی کنه. ولی action script نسخه ۲ رو پشتیبانی می کنه).

درس دوم:

خوب این درس رو با معرفی یه سری کلمات کلیدی و مفاهیم بنیادی برنامه نویسی شروع می کنیم و مابینش یه مقدار بیشتر از onscrip3 ac خواهیم گفت.

یک برنامه (program) به مجموعه ای از دستورات (instructions) گفته می شه که می تونه بوسیله یه نرم افزار یا کامپوتر می تونه اجرا بشه.

اون دستوراتی رو که توسط برنامه نویس نوشته می شه source code نامیده می شه. دستورات بوسیله syntax یا گرامر به کامپایلر (این کلمه در پایین توضیح داده شده) تفهیم می شن. syntax در حقیقت یه جور فرم دادن به دستوراته. مثل زبان فارسی که برای بیان جملاتش دستور زبان داریم زبانهای برنامه نویسی هم برای اینکه به کامپایلر تفهیم بشن به syntax متوسل می شن.

برای نوشتن onscrip3 ac شما فقط به جایی نیاز دارین که بتونین توش بنویسین حالا می خواد notepad باشه یا word یا هر ویرایشگر متن دیگه!!!

اما اکثر برنامه نویسا برای نوشتن onscrip3 ac از ابزارهای ویژه شرکت Adobe مثل ابزار ویژه محیط خود فلش یا flex builder استفاده می کنن. (flex builder هم از onscrip3 ac پشتیبانی می کنه و از MXML که یه زبان برمبنای xml برای توصیف محیط ارتباط کاربریه (intrface))

برنامه های نوشته شدخ توسط onscrip3 ac می تونن توسط سه نوع محیط نرم افزاری شناخته و اجرا بشن : Adobe Air - Flash liteوَ Adobe Flash Player. در ادامه درسها در مورد هر کدومشون توضیح خواهیم داد. ولی این رو بدونین که هر کدوم از این سه نوع محیط رو در اصطلاح flash client runtime environments مینامند و هر کدومشون یه نوع ماشین تو دلشون دارن که می تونن onscrip3 ac رو به وسیله اون اجرا کنن. (AVM- action script virtual machine)

قبل از اینکه action script توسط Flash client runtime environments (هر کدوم از اون سه محیطی که در بالا گفته شد) اجرا بشه باید از کدهای نوشته شده توسط برنامه نویس همون source code تبدیل بشه به کد دودویی (تنها زبانی که کامپیوتر حالیشه) این تبدیل توسط کامپایلر (compiler) انجام میشه. پس تا اینجا فهمیدیم که ما برای دستور دادن به کامپوتر ابتدا دستورات

خودمون رو توسط یه گرامر خاص یا syntax می نویسیم که تا حدی بـه زبـان خودمـون شـبیه و بعـد یـه موجودی به نام کامپایلر میاد و این نوشته های ما رو به زبان دودویی (صفر و یک) تبدیل می کنه تا کـامپوتر اون رو بفهمه.

action Script های کامپایل شده با پسوند swf شناخته می شن. پس swf حاوی کدهای کامپایـل شـده actionscript و مدیاهای مورد نیاز action script هست .(البته به صورت دودویی!!!)

پس اگه بخوایم به صورت خلاصه این درس رو تو چند جمله مـرور کنـیم مـی تـونیم بگـیم کـه یـه برنامـه Action script مجموعه ای از دستورالعمل هاست که می تونن به وسیله یکی از محـیط هـای : Adobe Air- Flash liet یا Flash Player اجرا بشن.

Action script می تونه بوسیله یه ویراشگر متن مثـل Notepad یـا ابزارهـای اسـتاندارد Adobe (کـه البته امکانات بسیار ویژه ای دارن) نوشته بشه.

برای اجرای کدهای Action script ابتدا باید اونا رو کامپایل کنیم تا بـرای ماشـین قابـل فهـم باشـن کـه کامپایلر ویژه اینکار همراه خود برنامه فلش هست و شما با یک کلیک می تـونین کـد خودتـون رو کامپایـل کنین. (یا به وسیله Flex Builder 2)

قاعدتا بحثای مقدماتی هر موضوعی برای کسایی که تازه شروع کردن یه کم خسته کننده و نامفهوم هسـتن. اگه زیاد ازشون سر در نمی آرید (البته تازه کارها) زیاد نگران نباشید. تو درسای بعـدی کـه بـا مثـال همـراه هستن مفهومشون رو به صورت عملی لمس می کنید و براتون روشن میشن.

تو درس بعدی مفاهیم مفاهیم بنیادی تـر و کـاربردی تـری رو توضـیح خـواهیم داد و انشـاء الله اولـین کـد خودمون رو با فلش خواهیم نوشت

درس سوم:

درس رو با معرفی یه سری مفاهیم و اصطلاحات کلیدی ادامه می دیم(البته این مفاهیم تقریبا تو همه زبانهای برنامه نویسی مشترک هستند) :

البته قبل از شروع مفاهیم یه چندتا نکته کوچیک رو بگم که تو مثالهایی که میارم دچار مشکل نشیم.

۱- دستورای اکشن اسکریپت از بالا به پایین و از چپ به راست اجرا میشن. یعنی ابتدا خط اول از چپ به راست تا آخر سپس خط دوم از چپ به راست تا آخر و بعد خط سوم...

۲- در حالت استاندارد در زبان اکشن اسکریپت هر دستور در هر خط به یک سیمی کالن (;) خطم میشه.

۳- دستور trace : یه دستوریه که وقتی کدمون رو می نویسیم (نه موقع اجرا!) بتونیم یه سری چیزا رو تست کنیم مثلا مقدار متغیرامون رو. اینا رو یادتون باشه که تا آخر درسا ازشون استفاده می کنیم.

۱- متغیرها و انواع داده (Variables and Data Types) :

به صورت ساده متغیرها جعبه هایی هستند که ما داده ها و یا اطلاعاتمون رو توی اونا ذخیره می کنیم تا بتونیم بعدا ازشون استفاده کنیم. برای مثال شما می خواین توی یه سایت اینترنتی username ها و password هاتون رو ذخیره کنین. شما میتونید اونا رو توی یه سری متغیر ذخیره کنید. برای ساختن یه متغیر فقط کافیه شما برای اون یه اسم منحصر به فرد در نظر بگیرید تا از سایر متغیرهای برنامه تون متمایز باشه. بعد از ایجاد اون شما می تونید بهش مقدار بدین برای مثال دستور زیر مقدار ۱ رو به myVariable اختصاص میده:

```
myVariable = 1 ;
```

نکته ۱ : برای نامگذاری متغییر بهتره که نامی متناسب با کارکردی که دارن بهشون اختصاص بدیم مثلا اگه می خوایم یه پسورد توشون ذخیره کنیم می تونیم براشون اسم pass یا password رو انتخاب کنیم.

نکته ۲: نامگذاری متغیرها در بعضی موارد محدودیت داره مثلا نمی تونیم نام یه متغیر رو با عدد آغاز کنیم یا نمی تونیم برای نام گذاری اونا از کلمات کلیدی خود زبان استفاده کنیم. اگه هر کدوم از این اشتباهات رو مرتکب بشین actionScript شما رو اگاه خواهد کرد و اجازه این کار رو بهتون نخواهد داد.

وقتی که ما می خوایم برای اولین بار از یک متغیر تو برنامومون استفاده کنیم باید اون رو با کلمه کلیدی var اعلان کنیم. همچنین برای اعلان متغیر در بار اول باید نوع دیتایی(اطلاعاتی) که می خوایم تو متغیر ذخیره بشه رو مشخص کنیم برای مثال دستور زیر برای اولین بار یک متغیر از نوع عددی (انواع متغیر رو در ادامه توضیح خواهیم داد) با اسم myVariable اعلان می کنه:

```
var myVariable:Number = 1;
```

! همونطور که می بینید ادیتور فلش کلمات کلیدی زبان اکشن اسکریپت رو به صورت رنگی در میاره.

بعضی از انواع داده در جدول زیر اومدن:

نوع داده	مثال	توضیحات
Number	4.5	هر عددی حتی عدد اعشاری دهدهی
int	-5	هر عدد صحیح
uint	1	اعداد صحیح بدون علامت
String	"hello"	متن یا رشته ای از کارکترها
Boolean	true	True or false
Array	[2, 9, 17]	بیشتر از یک مقدار در یک متغیر
Object	myObject	یک ساختار پایه برای هر واحد از زبان actionScript

همچنین انواع دیگه ای از متغیر ها هم وجود دارن که بوسیله کلاسی که می خوایم استفاده کنیم تعریف میشن که در ادمه توضیح خواهیم داد.

(مثلا دستور روبرو از کلاس MovieClip برای ساختن یه MovieClip در زمان اجرا استفاده می کنه)

```
var myMC:MovieClip = new MovieClip();
```

این که 'گفتم برای مثال بود. لازم نیست نگران باشید! در ادامه درسها بیشتر در موردش صحبت خواهیم کرد.

۲ - **عبارات شرطی(conditionals)** :

بعضی مواقع ممکنه که کد ما نیاز به تصمیم گیری داشته باشـیه. یعنـی در صـورت وجـود یـه شـرایطی یـه کارایی رو انجام بده.

عبارات شرطی بر دو نوع هستند if و switch .

if

به همراه این کلمه کلیدی یک جفت پرانتز میاد . در صورتیکه عبارت داخل این پرانتز true (صـحیح) باشـه دستوراتی که در حدفاصل دو آکولاد بعد از پرانتزها هستند اجرا میشن در غیر اینصورت این دسـتورات اجـرا نمیشن و برنامه اونا رو نادیده می گیره.

بهتره با یه مثال واضح تر توضیح بدیم:

```
var a:Number = 1;
var b:String = "hello";
var c:Boolean = false;

if (a == 1) {
    trace("option a");
}
```

خوب در دستورات بالا ابتدا یه متغیر از با نام a از نوع عدد در نظر می گیریم و مقـدار ۱ رو بهـش اختصـاص میدیم. بعد b رو در نظر میگیریم از نوع رشته (مقدار دهی به متغیرهای رشـته بـا دوتـا آکـولاد صـورت مـی گیره) و مقدار hello رو بهش اختصاص میدیم. و متغیر سومی رو هم از نوع بولی در نظر می گیریم و مقـدار true رو بهش اختصاص میدیم.

در خط بعد می گیم اگه مقدار a برابر با ۱ بود اونوقـت در پنجـره خروجـی (trace) بنـویس optional a. توجه داشته باشید اینجام چون می خواستیم رشته چاپ کنیم اون رو داخل " " گذاشتیم.

شاید در مورد عبارت داخل پرانتز براتون سوال پیش بیاد که چرا از == استفاده کردیم؟

جواب: هر موقعی که بخوایم دو عبارت رو از نظر تساوی نسبت به هم اعتبارسنجی کنیم از این عبارت استفاده می کنیم. واضح تر بگم اگه بخوایم یه مقداری رو به یه متغیر نسبت بدیم از = استفاده می کنیم مثلا a=1 یعنی ۱ رو بریز تو متغیر a. اما a==1 یعنی اینکه "آیا a مساوی ۱ است؟" پس عبارت دوم همیشه مقدار "بله هست" (true) و یا "خیر نیست"(false) رو بر می گردونه.

همچنین عبارات مقایسه ای دیگه هم با معنیشون در پایین اومدن:

"a>b" = آیا a بزرگتر از است؟

"a<b" = آیا a کوچکتر از است؟

"a>=b" = آیا a بزرگتر یا مساوی از است؟

"a<=b" = آیا a بزرگتر یا مساوی از است؟

علاوه بر عبارات مقایسه ای عبارات منطقی رو هم داریم که در پایین توضیح می دیم و یه مثال هم می زنیم:

عملگر "&&": معنی "و" تو زبان فارسی رو می ده و اسمش هست AND.

عملگر "||" : معنی "یا" تو زبان فارسی رو می ده و بهش میگن OR.

عملگر "!": معنی "نچ" . فارسی رو می ده و صداش می زنن NOT.

مثال زیر همه چیز رو مشخص می کنه:

```
if (a == 1 && b == "goodbye") {
    trace("options a and b");
}
```

توضیح : کد داره میگه که اگه مقدار متغیر a برابر ۱ بود **و** مقدار b برابر با goodbye بود. در پنجره trace چاپ کن options a and b.

با توجه به مقدار متغیر ها که تو کد قبلی داشتیم چون مقدار b برابر با hello بود اینجا هر دوتا شرط برقرار نیست پس عبارات داخل آکولاد اجرا نمیشه و چیزی در پنجره trace چاپ نمی شه. نکته: در عملگر AND **هر دو عبارت** چپ و راست اون باید برقرار باشن یا به عبارتی مقدار true داشته باشن.

مثال بعدی :

```
if (a == 1 || b == "goodbye") {
    trace("option a or b");
}
```

در این مثال در صورتی دستورات داخل آکولاد اجرا میشه که طرف راست **یا** طرف چپ عبارت برقرار باشه . به عبارت دیگه یکی از دوعبارت یا هر دو عبارت دو طرف عملگر || برقرار باشن. پس در این مثال عبارت options a and b در پجره trace چاپ میشه. چو اگرچه عبارت b برقرار نیست ولی همونکه a برقراره کافیه

مثال بعدی:

```
if (!c) {
    trace("not option c");
}
```

در این مثال میگه که اگر c –true **نبود** چاپ کن not option c . چون در بالا c، false اعلان شده بود شرط برقرار است این عبارت چاپ می شه.

فکر کنم برا این جلسه دیگه بسه! در جلسه بعدی در مورد if بیشتر توضیح می دیم و اگه مجالی بود switch رو هم توضیح می دیم.

پانوشت: اگه درس خسته کننده به نظر میاد کاملا یه امر عادیه. اینا مسائل مقدماتیه هر زبان برنامه نویسیه که باید خوب یاد گرفته بشن. چند تا درس دیگه که بگذره کاملا وارد کدنویسی میشیم و دیگه درسا خسته کننده نخواهند بود

درس چهارم:

جلسه قبل رو با بحث نیمه کاره if تموم کرده بودیم بنابر این با همین مبحث این جلسه رو شروع می کنیم.

خوب در جلسه قبل گفته بودیم که برای گذاشتن شرط رو دستور العمل هامون از کلمه کلیدی if استفاده می کنیم و چندتا مثال هم در رابطه با اون گفتیم. حالا فرض کنیم که می خوایم به کامپوتر بگیم "اگه این شرط برقرار بود این کار رو بکن **وگرنه** کار دوم رو انجام بده" . قسمت اول رو که در جلسه قبلی گفتیم با if انجام میشه قسمت دوم رو با کلمه کلیدی **else** انجام میدیم. else بلافاصله بعد از آکولاد بسته ی if میاد و بعد از خودش آکولاد رو باز می کنه و دستورات مربوط به خودش رو در محدوده دوتا آکولاد باز و بسته قرار میده. قطعه کد زیر همه چیز رو مشخص می کنه.

```
if (a != 1) {
    trace("a does not equal 1");
} else {
    trace("a does equal 1");
}
```

خط اول میگه که اگه a برابر با ۱ نبود چاپ کن "a doesnot equal 1" و خط سوم میگه در غیر اینصورت(یعنی در صورتیکه a برابر با ۱ بود) چاپ کن "a does equal 1".

تا اینجا تونستیم که توسط دستور if و متعلقات اون دو تا شرط رو چک کنیم و بنا به شرایط کار خاصی رو که مدنظرمون هست انجام بدیم. حالا اگه بخوایم بیش از ۲ تا شرط رو چک کنیم چی؟ مثلا بخوایم به

کامپیوتر بفهمونیم که اگه شرط ۱ برقرار بود کار یک رو انجام بده، اگه شرط ۲ برقرار بود کار ۲ رو انجام بده، اگه شرط ۳ برقرار بود کار ۳ رو انجام بده و .... اینجاست که از کلمه کلیدی **elseif** برای حل مشکلمون کمک می گیریم.

مثال زیر رو ببینید:

```
if (a == 1) {
    trace("option a");
} else if (b == "hello") {
    trace("option b");
} else {
    trace("option other");
}
```

کد بالا در خط اول و دوم به کامپوتر می گه که اگه a برابر با ۱ بود چاپ کن "option a" و اگر مقدار b برابر با hello بود چاپ کن "option b" و در غیر اینصورت(یعنی در صورتی که هر کدوم از دوتا حالت بالا برقرار نباشه) چاپ کن "Option other".

در مورد بحث if به مقدار مقدماتی فکر می کنم کافی باشه. در درس های آینده و مثالهای اونا کاملا با کاربردها و تکنیک های if و هیات همراه آشنا خواهیم شد.

switch

switch در حقیقت کار کل if و هیات همراهش(elseif و else) رو با هم انجام میده. به صورتی که یک متغیر رو در نظر می گیره و در داخل خودش تمامی شرط ها رو یک به یک چک می کنه و در صورت برقراری هر شرط کار مربوط به اون رو انجام میده.

switch داخل خودش سه تا کلمه کلیدی داره:

۱) کلمه کلیدی **case** : این کلمه در حقیقت کار چک کردن شرط ها رو انجام میده(کاربردی شبیه elseif)

۲) کلمه کلیدی **break** : این کلمه هم حد فاصل اجرای دستورات رو مشخص میکنه یعنی دستورات پتیین اون اجرا نمی شن!!!

۳) کلمه کلیدی **default**: کارایی شبیه به else داره. یعنی هر شرطی غیر از شروط بالایی برقرار باشه.

مثال زیر همه چیز رو روشن می کنه:

```
switch (a) {
    case 1 :
    trace("one");
    break;
    case 2 :
    trace("two");
    break;
    case 3 :
    trace("three");
    break;
    default :
    trace("other");
    break;
}
```

کد با کلمه کلیدی switch شروع میشه و a داخل آکولاد نشون می ده که می خوایم شرطای اتفاق افتاده رویa رو بررسی کنیم

در خط دوم داریم :case 1 یعنی اگر a باشه (توجه کنید که انتهای هر case باید : بیاد) خط بعد میگه چاپ کن "one" و break میگه که وقتی case 1 برقرار بود دستوراتی که تا بالای سر من نوشته شده رو اجرا کن. توجه: اگه این break رو نذاریم دستورات تا break بعدی اجرا می شن.

بقیه case ها هم همینطور ادامه دارن و به ترتیب گفته شده کار خودشون رو می کنن!

default هم میگه که اگه هیچ کدوم از case های بالایی برقرار نبودن بنویس "other" . و با آکولاد switch ما تموم میشه.

درس پنجم :

همونطور که قول داده بودیم این جلسه حلقه های تکرار (loop) ها رو توضیح می دیم.

با ذکر یه مثال بحث رو شروع می کنیم.فرض کنید شما تو یه پروژه برنامه نویسی مجبور میشین یه کار ثابت رو چندین بار پشت سر هم انجام بدین. مثلا کد شما باید یه لیست ۲۰ نفره رو بررسی کنه و یه اسم خاص رو تو اون پیدا کنه. برای اینکار کد شما باید ۲۰ بار عمل چک کردن اسم رو انجام بده. خوب اولین راهی که به ذهنمون میرسه اینه که به کامپوتر بگیم:

"اسم اول رو چک کن اگر برابر بود چاپ کن"

"اسم دوم رو چک کن اگر برابر بود چاپ کن"

"اسم سوم رو چک کن اگر برابر بود چاپ کن"

.

.

.

"اسم بیستم رو چک کن اگر برابر بود چاپ کن"


این دقیقا همون کاری بود که من در ابتدای آشنایی با دنیای کامپوتر موقع کدنویسی انجام میدادم.
البته با زبان qbasic یادش به خیر...

شاید برای ۲۰ بار کار ساده ای باشه ولی حالا فرض کنید همین عمل باید ۱۰۰۰۰ بار انجام بشه. پس با این حساب شما باید برای همین کار ساده ۱۰۰۰۰ خط کد بنویسید.

خوب اینجاست که می تونیم قبول کنیم دستوری برای انجام این کار باید وجود داشته باشه. اینچنین کارهایی رو حلقه های تکرار یا به عبارت کامپوتری loop ها انجام می دن.

حلقه های تکرار بر دو نوع هستند:

۱- حلقه تکرار for

۲- حلقه تکرار while

حلقه تکرار for :

حلقه for کارهایی رو که بهش مربوط میشه رو برای تعداد مشخصی دفعه انجام میده. یعنی برنامه نویس خودش برای for مشخص می کنه که چند بار کار مربوط به خودش رو انجام بده.

برای معرفی for و ساختار اون مثال زیر رو پیگیری می کنیم.

```
for (var i:Number = 0; i < 3; i++) {
    trace("hello");
}
```

در مثالی که بیان می کنیم قصد داریم عبارت hello رو ۳ بار چاپ کنیم. اولین کاری که می کنیم نوشتن کلمه کلیدی for هست. بعد از اون شرایط رو باید برای for مشخص کنیم. شرایط رو در داخل یک جفت پرانتز باز و بسته برای for بیان می کنیم . برای اینکه به for بفهمونیم که باید ۳ بار کارش رو انجام بده از یه متغیر استفاده می کنیم. در ابتدا به متغیر مقدار ۰ رو اختصاص میدیم(در اصلاح اون رو initialize) می کنیم(ما در این مثال متغیر i رو با مقدار صفر و از نوع Number در نظر گرفتیم). بعد مقدار انتهایی رو برای اون مشخص می کنیم(بعد از ;) در این مثال میگیم تا موقعی که i<3 باشه کارت رو تکرار کن. پس با این حساب کار ما ۳ بار انجام خواهد شد(به ازای i=0 ,i=1 ,i=2) . در قسمت بعدی (بعد از ;) داریم i++ این عبارت یعنی در هر مرحله یکی به مقدار i اضافه کن.

با این حساب در مرحله اول i=0 هست for میاد کارای خودش رو (کارای بین دو تا آکولاد باز و بسته) انجام میده. بعد از اون مقدار i رو بررسی میکنه میبینه i=0 هست پس هنور از ۳ کوچیکتره بنابراین کارش

رو ادامه میده یکی به مقدار i اضافه میکنه و مقدار i رو می کنه ۱. دوباره کارها رو انجام میده و مقدار i رو چک میکنه . مقدار i ،یک هست پس هنوز از ۳ کوچیکتره بنابراین بازم یکی به مقدار i اضافه میکنه و i رو برابر با ۲ میکنه. بازم کارها رو انجام میده و بازم میبینه که مقدار i، 2 هست پس هنوز از ۳ کوچیکتر هست پس بازم یکی به مقدار i اضافه می کنه و کارها رو انجام میده ، اینبار که مقدار i رو چک میکنه می بینه که i=3 شده پس دیگه i<3 برقرار نیست و کار for تموم میشه و خط اجرا به بعد از آکولاد بسته for منتقل میشه .

چند نکته:

۱- به جای i++ می تونیم بنویسیم i+=1 . اگر بخوایم مقدار i دوتا دوتا اضافه بشه می نویسیم i+=2 و...

۲- معموله که تو حلقه های شمارنده متغیر شمارنده رو i در نظر می گیرن.(اجبار نیست!!!)

۳- امکان برعکس شمردن هم وجود داره. یعنی به جای اینکه به i اضافه کنیم از اون کم کنیم.(i--)

```
for (var i:Number = 3; i > 0; i--) {
    trace("hello");
}
```

از حلقه for مطمئنا در درسهای آینده بیشتر استفاده خواهیم کرد. حلقه های تکرار از جزئهای اساسی برنامه های کامپوتری هستند.

پانوشت: راستی تا حالا نگفتیم این کدها رو کجا مینویسیم. چون از این به بعد در مثالها لازمه که خروجی رو خودمون چک کنیم لازمه که کدها رو تو ادیتور فلش وارد کنیم.

برای اینکار شما برنامه فلش رو که باز می کنین(فلش ۸ بعد) در قسمت create new گزینه اول(flash file (Ac on script3 . انتخاب و روی فریم اول کلیک راست کنید و گزینه action رو انتخاب کنید و کدها رو در ادیتوری که ظاهر میشه بنویسید.

برای دیدن نتیجه کدهاتون هم ctrl+Enter رو بزنید یا از control گزینه test Movie رو انتخاب کنید. در آینده در مورد اعمال کد به اجزای دیگه هم بحث خواهیم کرد.

در جلسه بعدی در مورد حلقه ی تکرار while خواهیم گفت.

درس ششم:

بحث جلسه قبل رو تا حلقه تکرار for ادامه دادیم. اگه یـه مـروری روی بحث قبل داشته باشیم متوجه می شیم که حلقه For یه حلقه ی کرانداره یعنی ما می دونیم کـه چنـد بـار این حلقه تکرار خواهد شد. یعنی این خودمون هستیم که براش انتها مشـخص مـی کنـیم. ولـی امکـان داره بعضی از حلقه ها وجود داشته باشن که ما ندونیم چندبار باید تکرار بشن. یعنی حلقه هایی که اونقـدر تکـرار می شن تا خواسته یا خواسته های یه شرطی رو برآورده کنن.

حلقه تکرار while :

برای مثال فرض کنید قصد دارید یه بازی شانس طراحی کنید که تو اون اعدادی به صورت تصـادفی توسـط کامپیوتر تولید می شن و به بازیکنا داده میشن ، اگه عددی کـه بـرای یـه بـازیکن انتخـاب مـی شـه از ۰.۵ بزرگتر باشه اون بازیکن بازندست.

اول از همه باید از کامپوتر بخوایم که برای ما یه عدد تصادفی تولید کنه. این کار تو اکشن اسکریپت بـا تـابع random انجام میشه (مثل خیلی از زبانای برنامه نویسی دیگه).

نکته: از این به بعد تو مثالهامون گهگاهی توابع زبان اکشن اسـکریپت رو هـم معرفـی مـی کنـیم. البتـه نـه همشون رو!!! تا جایی که امکان داشته باشه توضیحشون میدم ولی برای اطلاعات بیشتر می تونین از بهتـرین منبع یعنی help خود برنامه فلش استفاده کنید.

خوب، گفتیم از تابع random() (فعلا بدونین که جلوی توابع از پرانتز چه خالی، چه پـر، اسـتفاده خـواهیم کرد تا دو جلسه دیگه که تابع ها رو توضـیح بـدیم) اسـتفاده مـی کنـیم بـرای تولیـد اعـداد تصـادفی. تـابع random یه عدد تصادفی بین ۰ تا ۱ تولید می کنه.

همونطور که شرط کرده بودیم عدد ما باید از ۰.۵ کوچیکتر باشه . امکان داره کامپوتر اولین عددی که تولید می کنه تو محدوده ۰ تا نیم نباشه(یعنی از ۰.۵ تا ۱ باشه) ، یا اینکه اولی تو محدوه صفر تا ۰.۵ باشه ولی دومی نباشه یا شایدم نه ، اولی و دومی تو محدوه صفر تا ۰.۵ باشن ولی سومی نباشه و بازم شاید اولی و دومی و ... تو محدوه صفر تا ۰.۵ باشن ولی n امی نباشه. پس کامپوتر باید اونقدر عدد تولید کنه تا برسه به یه عددی بزگتر از ۰.۵. خوب ! اینجا می دونیم که باید پای یه حلقه تکرار وسط باشه ولی نمی دونیم این حلقه چند بار قراره دور بزنه! یه بار؟ دو بار ؟ یا خیلی هزار بار؟

اینجاست که حلقه تکرار while به دادمون می رسه.

به کد زیر توجه کنید.

```
var num:Number = 0;
while (num < .5) {
    num = Math.random();
}
```

خوب، خط اول که معلومه یه متغیر به نام num از جنس Number (عددی) تولید و عدد ۰ رو به عنوان مقدار اولیه به اون اختصاص میده (به اصلاح اون رو initialize) میکنه. در خط بعد به کامپیوتر میگیم تا موقعی که num کوچکتر از ۰.۵ هست کارای محدوده کروشه باز و بسته رو انجام بده. در بار اول که مقدار num صفره در دفعات بعدی هم که اعداد ما بوسیله تابع random تولید میشن و اگه کوچیکتر از نیم بودن بالتبع حلقه همینطور دور می زنه تا عدد تولیدی توسط random از ۰.۵ بزرگتر بشه.

نکته : تابع random() یکی از تابع های کلاس math هست. کلاس math حاوی یه سری توابع برای کارهای و عملیات ریاضی هست . در مورد کلاس ها در درس های آینده بحث خواهیم کرد. فعلا در همین حد بدونین که برای استفاده از تابع random باید قبلش یه .math بیاد تا بعدا ببینیم چی میشه.

در مورد while یه نکته خیلی مهم وجود داره و اون اینکه همیشه حواسمون باشه که اگه شرط داخل پرانتز همیشه برقرار باشه یعنی هیچ وقت نقض نشه برنامه ما تا ابدالدهر در حالت اجرا خواهد موند. و کامپوتر شما در هنگام اجرای برنامه لاجرم هنگ خواهد نمود مثل این بابا 

حالا اگه خامی کردید و این اتفاق براتون افتاد یه وقت دستپاچه نشید. خونسرد باشین ، خـود موتـور اجـرای فلش یا همون avm اونقدر باشعور هست که بعد از یه مدت که ببینه کارش داره سـنگین میشـه ، میفهمـه که شما سوتی دادین و کار رو خودش متوقف می کنه ولی طبیعیه که هدف شما مبنی بر اجـرای اون کـاری که مدنظرتون بوده یحتمل به ثمر نخواهد نشست. (البته معمولا زبانای برنامه نویسی دیگه در اینجـور مـوارد با ctrl+break متوقف میشن).

یه نمونه از این حلقه های بی پایان در زیر اومده:

```
var flag:Boolean = true;
while (flag) {
    trace ("infinite loop");
}
```

همونطور که میبینیم متغیر flag از جنسی بولی با مقدار اولیه true تعریف میشه و در خـط بعـد میگـه تـا موقعی که flag مقدارش true هست چاپ کن infinite loop . و چون مقدار flag هیچ جایی تغییر نمـی کنه و برای همیشه true خواهد بود حلقه تا بی نهایت تکرار میشه.

نکته: در قسمت شرط while فقط اسم متغیر آورده شده. اگر به تنهایی این اسم آورده بشه یعنی تا مـوقعی که متغیر true بود ولی اگه بخوایم عکس این قضیه اتفاق بیفته باید حتما اعلام کنـیم flag=false یعنـی برای false باید حتما و به طور صریح اعلام کنیم ولی برای true الزامی نیست. یه راه دیگم استفاده از صفر و ۱ هست. صفر به معنی false و۱ به معنای true.

بازم نکته: استفاده بیجا از حلقه (چه for چه while) ممنوع میباشد. چون کامپیوتر تمـام کـاراش رو رهـا میکنه تا کار حلقه تموم بشه و اگه از حلقه به طور بیجا و نامناسب اسـتفاده بشـه بیخـود برنامـه رو سـنگین کردیم و به عبارت فنی سربار زمان اجرا ایجاد کرده ایم.

جلسه بعد رو با بحث آرایه ها(Arrays) در اکشن اسکریپت ادامه خواهیم داد.

درس هفتم:

در ادمه بحثمون پیرامون برنامه نویسی با actionscript می رسیم به مبحث مهم و شیرین آرایه ها. که از مفاهیم پایه ای هر زبان برنامه نویسی هستند.

اگه یادتون باشه در جلسات قبلی (درس سوم) در مورد متغیر ها و انواع اونا صحبت کردیم. و گفتیم متغیرا می تونن مقداری رو داخل خودشون ذخیره کنن. ولی می دونیم که متغیرا فقط قابلیت ذخیره یک مقدار رو در خوشون دارن. یعنی یک قطعه داده در هر متغیر. خوب برای برنامه های ساده شاید شما به چیزی بیش از این هم احتیاجی نداشته باشین ولی کم کم و در برنامه های بزرگتر با انبوهی از داده ها مواجه می شین که امکان ذخیره اونا به صورت تک تک در متغیرها دشوار و در مواردی غیر ممکن خواهد شد. اینجاست که دست به دامن آرایه ها می شیم تا علاوه بر راحت شدن کارمون بتونیم قطعات داده رو هم دسته بندی کنیم.

فکر کنید شما می خواین ۵۰ تا داده رو ذخیره کنین. یه راهش اینه که ۵۰ تا متغیر رو تعریف کنین و داده ها رو تک تک تو اونا ذخیره کنین و راه دوم و عقلانی اون هم استفاده از آرایه هاست که به راحتی می تونیم روی عناصر اونا حرکت کنیم ، اونا رو بررسی کنیم ، به ابتدا و انتهای اونا عضو اضافه کنیم و خیلی کارای دیگه.

خوب اولین کاری که باید انجام بدیم اینه که آرایه رو تعریف کنیم و به برنامه بشناسونیم. به دو روش این کار امکانپذیره:

۱- روش عادی مثل تعریف متغیر که نوع رو Array انتخاب کنیم.(البته اینم یه نوع استفاده از کلاسه بعدا می فهمید)

۲- استفاده از کلاس array و ایجاد یک شی خالی از آرایه. (بازم یادآور می شم که مبحث کلاسها و مفاهیمش رو در آینده به طور کامل توضیح خواهیم داد.)

دو نوع تعریف رو در زیر میبینید. که خط اول در مورد روش اول و خط دوم هم برای روش دوم هست.

```
var myArray:Array = [1, 2, 3]
var yourArray:Array = new Array();
```

در خط اول آرایه با نام myArray تعریف شده و مقادیر اولیه ۱ و ۲ و ۳ به اون اختصاص داده میشن.(initialize میشن)

در خط دوم هم یه شی خالی با نام myArray از کلاس Array ساخته میشه.

نکته ۱: در مورد اسم گذاری متغیرها طبق یه قرارداد اگه اسم از دو کلمه تشکیل بشه به دو صورت رسمی انجام می گیره. یا حرف اول کلمه دوم بزرگ نوشته میشه (مثل اینجا که Array با حرف بزرگ شروع شده) و یا کلمه دوم با یه _(آندرلاین) از کلمه اول جدا میشه. یادتون باشه این کار بین برنامه نویسا مرسومه، الزامی نیست!!! در آینده خواهید دید که شکل اول در مورد کلمات کلیدی فلش رعایت شده. اینجوری شما راحتت تر کلمات کلیدی دوبخشی رو حفظ می کنید.

نکته ۲: همونطور که می بینید مقدار دادن به آرایه به اینصورته که یه براکت باز میشه و اعدادی که باید به عنوان محتوای آرایه در نظر گرفته بشن با ویرگول از هم جدا میشن.

نکته ۳: برچسب گذاری اعضای آرایه توسط خود زبان از صفر شروع میشه. یعنی برچسب اولین عضو ۰ و برچسب دومین عضو ۱ و به همین صورت ادامه داره. از برچسب ها برای دسترسی به اعضای آرایه استفاده می کنیم. مثلا مثال زیر عضو اول آرایه رو برامون چاپ می کنه یعنی عدد ۱ رو.

```
var myArray:Array = [1, 2, 3];
trace(myArray[0]);
```

خوب یه مثال هم از روش دوم می زنیم تا با اون هم آشنا باشیم. استفاده از روش دوم بهتر و اصولی تره. چون می تونیم از امکانات و توابع آماده کلاس Array استفاده کنیم

در مثال زیر از دوتابع ویژه این کلاس استفاده شده. (البته به اینا می گن متد که در مبحث کلاسا باهاشون آشنا خواهیم شد) . تابع () push یه عضو به انتهای آرایه اضافه می کنه و تابع () pop یه عضو از ته آرایه بر می داره.

نکته : اگه می خواین در مورد ()push و ()pop بیشتر بدونید کافیه کلمه stack رو تو گوگل یه سرچ کنین.

```
var myArray:Array = new Array();
myArray.push(1);
trace(myArray)
// یک چاپ می شود //
myArray.push(2);
// الان آرایه دو عضو دارد //
trace(myArray.pop());
// الان آرایه یک عضو دارد //
trace(myArray)
```

در خط اول یه شی به نام myArray از کلاس Array ساخته می شه. خط دوم عدد یک رو توسط push به انتهای آرایه (که الان خالیه و میشه خونه اول) اضافه می کنه و در خط هفتم هم عدد ۲ که آخرین عضو آرایه هست چاپ می شه. یادتون باشه اینجا علاوه بر چاپ شدن عنصر آخر برداشته هم میشه. در مورد بقیه خطها هم دیگه توضیح نمیدم . اگه آموزش ها رو تا اینجا دنبال کرده باشین مطمئنا خودتون متوجه می شین.

نکته : برای اینکه تو قطعه کدمون توضیحات بذاریم تا بعدا راحت تر متوجه بشیم یا بقیه راحت تر بفهمنش ابتدای خط دوتا اسلش(//) می زاریم و تو اون خط هر توضیحی که خواستیم می ذاریم تا کامپایلر(کامپایلر چیست؟ رجوع کنید به درس دوم) بهش کاری نداشته باشه و ترجمش نکنه. اگه بخوایم چند خط توضیح بذاریم در ابتدای خط اول (*/) و در انتهای خط آخر توضیحاتمون (/*) می ذاریم.

مثال زیر هم در مورد استفاده رشته ها در آرایه هست.

```
var myArray:Array = ["a", "b", "c", "d", "e"]
trace(myArray[4])
```

در این برنامه e را به عنوان خروجی خواهیم دید. در ضمن طول رشته ها هر مقداری می تونه باشه. مثلا عضو اول آرایه باش "amir" و عضو دوم باشه "mohammad javad" و ...

فکر می کنم برای آرایه ها فعلا همینقدر کافی باشه. اگه وقت کنم یه فایل اتچ هم قرار می دم تا متدهای مهمی مثل sort و length و ... رو توش معرفی کنم. تو درس نمی آرمشون چون هنوز به مبحث کلاسها نرسیدیم. همینقدر هم که گفتم فکر کنم زیاده روی کردم.

برای جلسه بعد تابع ها (functions) رو توضیح خواهم داد.

درس هشتم:

در ادامه درسهای اکشن اسکریپت به بحث مهم توابع می رسیم. فکر می کنم یک دو جلسه دیگه بحث مفاهیم بنیادی رو تموم کنیم و انشاء الله با کدنویسی و جزئیات actionscript بیشتر و دقیقتر درگیر بشیم

توابع (functions):

✓ اندر فواید توابع:

تا اینجا که برنامه نویسی رو شروع کردیم کدهامون رو خط به خط می نوشتیم و از بالا به پایین حرکت می کردیم. یعنی اگه فرض کنیم یه فلش جلوی خط اول برناممون باشه با اجرا شدن برنامه فلشه شروع به حرکت می کردو خط به خط به سمت پایین حرکت می کرد تا به انتهای برنامه برسه. ولی با مطرح شدن بحث توابع، این فلش یه کم باید تحرکش رو بیشتر کنه و به جای اینکه خط به خط به سمت پایین حرکت کنه گاهی اوقات از روی چندین خط پرش کنه و یه سری خط رو پشت سر هر پیمایش کنه و دوباره بپره سر جای خودش (یعنی همون چندین خط قبل) و همینطور ادامه بده و و ادامه بده و ادامه بده و... درسته که با این استراتژی آقا فلش ما مجبور میشه یه روند نامنظمی رو طی کنه ،ولی اینکارش چندین سود داره: یکی اینکه کدهای ما خواناتر میشن و دیگر اینکه ما از کدهامون می تونیم تو برنامه های دیگه هم استفاده کنیم، دیگری اینکه اشکال یابی (debuging) برنامه ما راحت تر میشه و مهم تر از همه تحرک بیشتر برای تناسب اندامش خیلی خوبه

توجه: اگه از متن بالا چیزی سر در نیاوردید پس از خوندن تمام مطالب این درس یه بار خطوط بالا رو مرور کنید. مطمئنا اینبار برای شما مفید فایده واقع خواهند شد

توابع بر دودسته اند:

۱- توابعی که ما خودمون می نویسیم

۲- توابع از پیش نوشته شده ای که خود زبان در اختیار ما قرار میده . مثل توابع پرکاربرد و عمومی مثل تابع محاسبه cos یا محاسبه sin یا توابع کار با رشته ها و خیلی توابع دیگه . هرچه برنامه نویس توابع یه

زبان رو بیشتر و بهتر بشناسه قاعدتا دستش برای نوشتن کدهای بهتر باز تـره. در طـول درسـهای آینـده بـا توابعی که برای مثالهامون بهشون نیاز خواهیم داشت آشنا خواهیم شد ولی یادگیری تمامی توابع ایـن زبـان فقط و فقط به کوشش و تحقیق بیشتر خودتون بستگی داره. یه منبع خیلـی خـوب بـرای یـادگیری و آشـنا شدن با اونا سیستم help خود برنامه فلاشه.

در این درس ما نوشتن دستی توابع توسط خود برنامه نویس رو یاد می گیریم و به مورد دوم کاری نداریم.

توابع در action script با کلمه کلیدی function معرفی می شن و می تونن ورودی یـا خروجـی داشـته باشن یا اصلا هیچ کدوم رو نداشته باشن.

توابع در حقیقت کدهای ما رو در بلاکهایی دسته بندی می کنن و هرموقع که بهشون نیاز داشته باشیم اونـا رو اجرا می کنیم.

برای مثال قطعه کد زیر تابعیه که عبارت hello رو در خروجی نشون می ده.

```
function showMsg(){
    trace("hello");
}
showMsg();
```

خط اول با عبارت کلیدی function شروع میشه و بعد از اون اسم تابع میاد و بعد از اون یـه جفـت پرانتـز که ورودی تابع بین اونا قرار می گیره. چون تابع ما ورودی نداره جفت پرانتز رو خالی می ذاریم. بعد از پرانتز دوم یه بلاک باز می کنیم و محتویات تابع رو بین دو جفت کروشه قرار می گیره.

در اینجا تنها کاری که تابع ما می کنه همانا چاپ عبارت hello است.

حالا کارمون که با تابع تموم شد ، نباید به امون خدا ولش کنیم ، بلکه باید در موقع نیاز اون رو صـدا بـزنیم. صدا زدن تابع هم فقط با اسمش انجام می گیره. چون در این مثال تابع ما ورودی نداره مثل خود تابع جفـت پرانتز صدازننده اون هم خالی از عبارت خواهد بود. در خط آخر این کد ، تابع رو صدا زدیم. ایـن صـدا زننـده هر جای برنامه می تونه باشه.

تمرین: جلوی خط آخر // بذارین تا این خط موقع کامپایل ترجمه نشه. چه نتیجه ای می گیرید؟

مثال بالا ساده ترین کابرد یه تابع بود. در مثال بعدی یه مرحله جلوتر می ریم تا تابعمون یه مقدار بیشتر به مخش فشار بیاره در مثال بعدی برای تابعمون ورودی تعریف می کنیم. مثال رو ببینید تا توضیحاتش رو ذکر کنیم.

```
function showMsg(msg:String) {
    trace(msg);
}
showMsg("goodbye");
```

مثال شبیه مثال قبلیه با این فرق که در خط اول تعریف تابع ما یه متغیر تعریف می کنیم به نام msg و از نوع string(رشته). هر ورودی که برای تابع می فرستیم در بدنه تابع با نام msg شناخته می شه. نکته مهم اینکه چون ما در تعریف تابع ورودی رو از جنس رشته تعریف کردیم موقع ارسال آرگومان(ورودی تابع) به تابع هم باید یه رشته برای تابع بفرستیم در غیر اینصورت موقع کامپایل با خطا مواجه خواهیم شد.

نکته: توابع می تونن بیش از یک ورودی داشته باشن. در ادامه با توابع چند ورودی هم آشنا خواهیم شد. در ضمن اگه تابع ما در تعریف یک ورودی داشته باشه باید یکی هم برای اون بفرستیم و اگه دو تا بود براش دقیقا دوتا و...

در خط دوم و در بدنه تابع میگیم که msg رو چاپ کن. حالا بستگی داره که ما چی برای ورودی تابع (msg) فرستاده باشیم. در اینجا ما در موقع صدا زدن تابع goodbye رو برای تابع فرستادیم. پس goodbye چاپ میشه.

حالا شما هرچی که دلتون می خواد جاش بذارین! اگه کار نکرد بزنین تو گوش من

در مثال سوم می خوایم یه مرحله جلوتر بریم و برای تابع خروجی هم قرار بدیم. یه موقع خروجی رو با خروجی trace که در مثالای قبلی داشتیم اشتباه نگیرین!!!

این خروجی یعنی اینکه تابع یه مقداری رو به برنامه برگردونه و برنامه با این خروجی هر کاری خواست بکنه. مثلا تابع cos به عنوان وردی ۹۰ درجه رو بگیره و به عنوان خروجی به برنامه ۰، (cos 90 =0) رو تحویل بده و برنامه هم از این نتیجه در محاسباتش استفاده کنه.

مثال زیر یه عدد از نوع number می گیره و بعد از انجام محاسباتش یه عدد از نوع number تحویل برنامه می ده.

```
function celToFar(cel:Number):Number {
    return (9/5)*cel + 32;
}
trace(celToFar(20));
```

در خط اول، بعد از پرانتز دوم (بسته) یه دونقطه میذاریم و نوع خروجی رو مشخص می کنیم(اینجا number).

خط بعد با عبارت کلیدی return شروع می شه. در توابعی که خروجی دارن با این عبارت مقداری که باید به برنامه تحویل داده بشه رو مشخص می کنیم. مثلا در اینجا چون ما در موقع صدا زدن تابع ۲۰ رو به عنوان ورودی به تابع دادیم مقدار برگشتی به برنامه ۶۸ خواهد بود. خودتون محاسبه کنید به همین نتیجه می رسید.

و بالاخره در برنامه آخر یه مقدار اصولی تر، از خروجی استفاده کردیم. یعنی اینکه یه متغیر تعریف کرده و مقدار خروجی رو توی اون می ریزیم و بعد از اون متغیر استفاده می کنیم.

```
function farToCel(far:Number):Number {
    return (5/9)*(far - 32);
}
var celDeg:Number = farToCel(68));
trace(celDeg);
```

نکته ۱: خروجی برنامه بالا ۲۰ هست. خودتون هم حساب کنید.

نکته ۲: در صورتی که برنامه خروجی نداشته باشه می تونیم بعد از دو نقطه ی بعد از پرانتز بسته عبارت void رو قرار بدیم که نشانه بدون خروجی بودن تابعه! اینکار برنامه رو خواناتر می کنه و استاندارد، وگرنه همونطور که در دو برنامه اول این درس دیدید نداشتن اون هم برای توابع بدون خروجی مشکلی ایجاد نمی کنه! برای ورودی ها اگه ورودی نداشتیم در قسمت ورودی تابع هیچی(مثل دو برنامه اول) نمی ذاریم و لازم نیست مثل خروجی void بذاریم.

نکته ۳: توابع بسیار پرکاربرد هستند. و در اکثر مواقع پیچیده تر از این ۴ مثال ما. در درسهای آینده کم کم با اونا آشنا شده و کاربردشون رو بیشتر و بهتر لمس خواهیم کرد.

نکته ۴: حالا دو مرتبه "اندر فواید تابع" رو که در پاراگراف دوم درس اومده بخـونین. فکـر کـنم حـال بهتـر بتونین باهاش ارتباط برقرار کنید. اگه بازم نتونستین، اشکالی نداره هنوز یه عالمه درس در آینـده داریـم کـه تو مثالای اونا از تابع استفاده خواهیم کرد.

در درس آینده custom object و اگر بتونم کلمه کلیدی this رو توضیح خواهم داد.

درس نهم:

به امید خدا امروز می خوام دیگه مباحث پایه ای رو تموم کنم. مفاهیمی که در همه زبانها مشترکه و در زبانهای مختلف با syntax متفاوت بیان می شن. تا اینجا سعی کردم که تمامی مفاهیم رو به صورت ساده بیان کنم و در آینده هم همین قصد رو دارم.

در هر صورت خودتون هم می دونین بیان کردن تمام ویژگیها و نکات مقدور نیست! من تا جایی که برام مقدور بود گفتمشون و به عبارتی سرنخ رو دادم دستتون. دیگه تمرین و تکمیل کردن اونا با خودتون.

برای این جلسه هم custom object و مفهوم this و شیوه های آدرس دهی رو بیان می کنم و بحث مفاهیم پایه رو تموم می کنم.

Custom Objects

یه مقداری که با actionscript کار بکنین می فهمین که با object ها خیلی سروکار دارین. اکثر هویت های مستقل مثل button ها (کلیدها) ، MovieClip ها و ... در حقیقت اشیا (object هایی) از کلاس های خودشون هستن. هر شی یه سری ویژگیها (properties) مثل عرض شی ، موقعیت اون تو صفحه نمایش و ... داره و همچنین هر شی یه سری خصیصه هایی (Method) داره. متدها کارایی هستن که یه شی می تونه انجام بده. مثلا ماشین می تونه حرکت کنه پس حرکت کردن یه متد برای ماشینه.

و هر شی می تونه به یه سری رخدادهایی (event) جواب بده مثل کلیک ماوس یا فشار یه کلید از صفحه کلید و ... مثلا اگه ماوس کلیک شد یه تابعی رو فراخوانی کنه.(تابع ها رو هم که جلسه قبل توضیح دادیم).

نکته: تمامی نکاتی رو که در مورد شی گفتیم و خواهیم گفت در بحث شی گرایی (oop) به صورت کامل توضیح خواهم داد. انشاءالله جلسات آینده.

ما می تونیم بنا به نیازمون خودمون هم یه شـی (object) بسـازیم و بـرای اون ویژگیهـا و خصیصـه هـایی تعریف کنیم.

مثلا در مثال زیر یه شی به نام plane می سازیم و یه سری ویژگـی مثـل pitch، roll، yaw بهـش مـی دیم و به اونا هم مقدار می دیم.

پس فعلا لازمه که بدونین ما می تونیم یه شی ای رو بسازیم و یه سری ویژگی براش تعریف کنیم در همـین حد فعلا کافیه!!!

```
var plane:Object = new Object();
plane.pitch = 0;
plane.roll = 5;
plane.yaw = 5;
trace(plane.pitch);
```

در واقع trace آخر عدد ۰ رو چاپ خواهد کرد.

بد نیست که بدونین شی ها می تونن ورودی برای توابع هم باشن و تابع بنا به نیاز از اونا استفاده کنه . مثـال زیر یه نمونه از این توابع هست که شی plane رو که در بالا ایجاد کردیم به عنوان ورودی بهش می دیم

```
function showPlaneStatus(obj:Object):void {
    trace(obj.pitch);
    trace(obj.roll);
    trace(obj.yaw);
};
showPlaneStatus(plane);
```

خروجی trace ها که به عهده خودتون! خط آخر رو هم که می دونین فراخوانی تابعه.

در همین حد فعلا از شی ها بدونین کافیه تا بعدا به صورت مفصل بررسیشون کنیم.

this

هر چی که بیشتر با actionscript کار کنین به این نتیجه خواهید رسید که this می تونه دوست خـوب و پرکابردی براتون باشه.

this در حقیقت یه شورتکات(shortcut) برای شی یا بردی (scope) هست که الان دارین باهاش کار می کنین.

scope یا برد حوزه ای هست که یه شی یا متغیر در اون معتبره. مثلا اگه در یک function یه شی یا متغیر تعریف کنیم. اون شی یا متغیر فقط در حوزه اون function معتبر هستن و به عبارتی زنده (live) هستن و با خروج از آکولاد بسته function اونا از بین می رن. یا اگه یه مووی کلیپ (که خودش شه هست) تو main تعریف بشه در تمام حوزه main برنامه scope داره.

دو مثال زیر همه چیز رو روشن می کنه:

در مثال اول فرض می کنیم که می خوایم به یه موی کلیپ که در timeline اصلی و یا به عبارتی main برنامه قرار داره دسترسی داشته باشیم. پس this در اینجا به timeline اشاره می کنه. یعنی شما در timeline هستید و می خواید به موی کلیپ دسترسی داشته باشید. پس اینجایی که الان هستید timeline هست و می تونید this رو timeline فرض کنید.

```
this.mc.width;
```

و در مثال زیر فرض کنین که الان در حوزه یا برد موی کلیپ mc هستین و می خواین به timeline دسترسی پیدا کنین . بنابراین الان اینجایی که هستین موی کلیپ هست پس this موی کلیپ می شه.

parent یه کلمه کلیدیه و میگه از اینجایی که هستم یه سطح برو بالاتر. پس چون mc داخل timeline هست و this الان موی کلیپمونه با parent یه سطح از موی کلیپ بالاتر می ریم و می رسیم به timeline.

مثال زیر رو ببینید:

```
this.parent.mc.width;
```

پس با یه کم دقت متجه می شیم که مثال بالا به width تایم لاینمون دسترسی داریم. خط آخر رو یه trace بگیرین متوجه می شین که مقدار پهنای timeline پروژتون رو نشان می ده.

نکته: این مفهوم رو هم در مثالهای درسهای بعد بیشتر باهاش کار خواهیم کرد.

شیوه های آدرس دهی در actionscript :

مانند سیستم عامل و وب سایت در actionscript هم دو نوع آدرس دهی داریم آدرس دهی مطلق (Absolute) و آدرس دهی نسبی (Relative).

در دنیای واقعی مثلا برای آدرس دهی یه بقالی همین دو روش رو داریم:

۱) **مطلق**:

-سلام داداش

- سلام علیکم

- ببخشید آدرس بقالی آقا کریم کجاست؟

- ببین برادر شما می رین بلوار شهید مطهری - کوچه لاله - ابتدای کوچه

۲) **نسبی**:

-سلام داداش

- سلام علیکم

- ببخشید آدرس بقالی آقا کریم کجاست؟

- آقا **از همینجا** (الان تو بلوار شهید مطهری وایسادن!) می ری کوچه لاله ، بقالی ابتدای کوچست!

همونطور که دیدن در آدرس دهی دوم گفت از همینجا که هستین . پس آدرس رو نسبی (نسبت به بلوار شهید مطهری) داد. در دنیای مجازی کامپوتر هم همینطوره.

دو روش آدرس دهی و معادل اونا رو تو ویندوز، مکینتاش و وبسایت می بینید. فقط اینو بدونین که:

۱) mc1 و mc2 دو تا موی کلیپ هستن که mc2 داخل mc1 هست.

۲) root در Ac onscript3 یعنی نقطه صفر پروژمون در فلش. یعنی هر چیزی که به پروژمون اضافه می کنیم. در هر صورت زیر شاخه ای از root خواهد بود. حالا چه مستقیم چه غیر مستقیم. مثلا یه موی کلیپ می سازید و در تایم لاین می ذاریم که اون زیر شاخه root هست. و یه موی کلیپ دیگه می سازیم و در موی کلیپ اولی می زاریم. موی کلیپ دومی زیر شاخه اولی و اون اولی خودش زیر شاخه root هست. پس موی کلیپ دومی نوه ی root هست

۳- ما با آدرس دهی می خوایم به mc2 دسترسی پیدا کنیم.

جدول اول در برای آدرس دهی مطلقه :

ActionScript	Windows OS	Mac OS	Web Site
root.mc1.mc2	*c:\folder1\folder2*	*Macintosh/folder1/folder2*	http://www.domain.com/dir/dir

و جدول دوم آدرس دهی نسبی از یه موی کلیپ سومی که خودش داخل timeline اصلیه:

ActionScript	Windows OS	Mac OS	Web Site
this.parent.mc1.mc2	..\folder1\folder2	../folder1/folder2	../dir/dir

خوب درس امروز و به همراه اون بحث مفاهیم پایه تموم شد فقط ذکر چند نکته رو لازم می دونم.

۱) مطالب این جلسه برای کسایی که تازه کار هستین یه کم ثقیله! ولی تازه کارا مطمئن باشن در ادامه و در مثالهای آینده کاملا به این مفاهیم مسلط خواهند شد. هنوز یه عالمه درس مونده با مثالای جور واجور

۲) از درس آینده که روی syntax و بحثای خودactionscript متمرکز می شیم در هر درس حتما مثال پیوستی به همراه سورس خواهیم داشت!

۳) اگه برسم می خوام یه وبلاگ در همین رابطه راه اندازی کنم که درس ها رو به صورت منظم و خصوصی تر اونجا هم بذارم ولی همچنان نوشتن در majidonline ادامه خواهد داشت.

در جلسه بعدی property های عمومی و شیوه به کار گیری اونا رو به صورت کلی توضیح خواهم داد.

درس دهم:

بسم الله الرحمن الرحیم

از این جلسه قراره که یه مقدار تخصصی تر روی actionscript متمرکز بشیم. بنابراین دیگه روی مفاهیم بنیادی که قبلا توضیحشون دادیم ایست نمی کنیم و فرض رو بر این می گیریم که شما مطالب قبلی رو به خوبی خوندید و فراگرفتید. بدون فوت وقت درس امروز رو شروع می کنیم.

برای درس امروز دو بحث مهم properties و events رو توضیح می دیم. مفاهیمی که اکشن اسکریپت به وسیله اونا به ورودی هایی واکنش می ده یا خروجی هایی رو تولید می کنه.

Properties

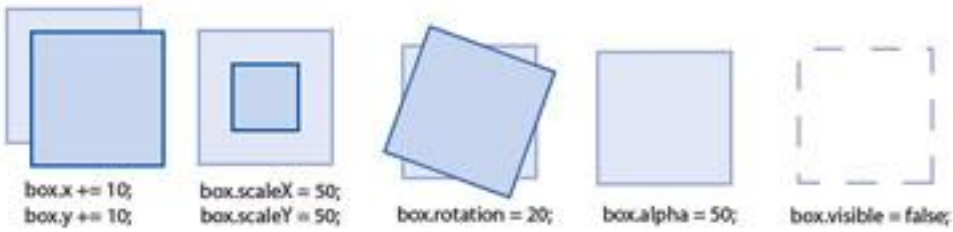
Properties یا ویژگیها چیزی هستند که به وسیله اونا می تونیم خصوصیات و ویژگیهایی رو برای یک شی تعریف , مدیریت کنیم. برای مثال شما می تونید خصوصیت width یک کلید (button) رو چک کنید یا تغییر(set) بدید.

اکثر ویژگیها خواندنی و نوشتنی (read-write) هستند به این معنی که شما هم می تونید اونا رو بگیرید(get) و تو برنامه استفاده کنید و یا خصوصیات یه شی رو set کنید. اما بعضی از ویژگیها هم هستند که فقط خواندنی(read only) هستند که به شما اجازه Set کردن رو نمی دن.

خوب ، بهتره بریم سر مثال. در ابتدا ما با syntax ایجاد ۶ نوع تغییر پرکاربرد توی یه مووی کلیپ آشنا می شیم تا در ادامه که با event handle ها آشنا شدیم اونا رو در کدنویسی هم به کار بگیریم و تستشون کنیم.

جدول زیر و شکلها گویای همه چیز هستند.

توضیحات	Property	Syntax	واحد سنجش
موقعیت	x, y	box.x = 100; box.y = 100;	pixels
تغییر اندازه نسبی	scaleX, scaleY	box.scaleX = .5; box.scaleY = .5;	percent / 0-1
تغییر اندازه مطلق	width, height	box.width = 72; box.height = 72;	pixels
چرخش	rotation	box.rotation = 45;	degrees / 0-360
شفافیت	alpha	box.alpha = .5;	percent / 0-1
قابلیت نمایش	visible	box.visible = false;	Boolean



اونایی که سابقه استفاده از ac onscript 2 رو دارن شاید متوجه یـه تغییـرات کـوچیکی بشـن. بعضـی از ویژگیها undescope (یا _) ندارن!

این تغییر توی ac onscript 3 اعمال شده تا همه ویژگیها یکنواخت بشن و بدون _ بیان نه اینکه بعضـی بدون _ و بعضی با _ بیان. بی خیال همچین مطلب مهمی هم نیست!

دستیابی به Properties هم خیلی راحته مثل کد زیر:

```
trace(box.alpha);
var bAlpha:Number = box.alpha;
```

حتی میشه به وسیله operator ها (عملگرای محاسباتی) هم اونا رو دستکاری کرد مثل کد زیر:

```
box.rotation += 20;
```

Events

Event ها یا رخدادها در حقیقت مثل کاتالیزگر عمل می کنن. با اتفاق افتادن یه رخداد می تونه یه method اجرا بشه یا یه ویژگی set و یا حتی get بشه.

برای مثال کلیک کردن ماوس یه رخداد (event) هست. یا فشار داده شدن یه کلید از صفحه کلید یا تکون خوردن ماوس و...

همه اینا می تونن توسط برنامه نویس معیاری برای اجرای یه قطعه کد قرار داده بشن.

Event ها انواع مختلفی دارن و هر کلاسی می تونه event های مخصوص خودش رو داشته باشه. مثلا کلاس video ، رخداد(event)هایی مربوط به پخش ویدئو داره و...

برای اینکه بفهمیم در طول اجرای برنامه چه event ای اتفاق افتاده از مفهومی تحت عنوان eventlistener استفاده می کنیم.

Eventlistener ها در حقیقت همون خاله زنکها یا فضول باشی های خودمون هستن  به این معنی که در طول اجرای برنامه یه گوشه ای می شینن و می پان که چه event ای اتفاق می افته تا سریعا اون رو به event handler اطلاع بدن Event listener ها بوسیله کلاسی به نام EventDispatcher پیاده سازی میشن. این کلاس امکانات جالبی رو در اختیار ما قرار میده که در آینده با چند تا از این امکانات (مثل حذف Event listener ها در مواقعی که کارشون نداریم) آشنا خواهیم شد.

برای ایجاد یه event listener ما نیاز به استفاده از متدی تحت عنوان addEventListener() داریم . مثال زیر رو ببینید.

```
1  rotate_right_btn.addEventListener(MouseEvent.MOUSE_UP,
   onRotateRight);
2  function onRotateRight(evt:MouseEvent):void {
3      box.rotation += 20;
4  }
```

این مثال یک eventlistener رو به کلیدی به نام rotate_right_btn و گفته اگر موس روی دکمه کلیک شد به محض بالا آمدن دکمه ماوس(Mouse_up) تابع مذکور اجرا بشه.

نکته: evt در ورودی تابع , از جنس کلاس MouseEvent تعریف شده اشاره داره به اون المنتی که به رخداد پاسخ داده مثلا در اینجا کلیدی که کلیک میشه المنت ماست.

در ضمن کلاس MouseEvent شامل همه رخدادهای ماوس مثل کلیک و حرکت و ... هست که در ادامه بیشتر باهاش آشنا خوایم شد.

مثال زیر دو تا رخداد رو برای یه موی کلیپ تعریف می کنه هر کدوم که اتفاق بیافتن یه تابعی رو اجرا می کنن. مثال زیر مثال معروف drag & drop هست. که با ماوس یه شی رو می گیریم و هر جا که خواستیم قرار می دیم. برنامه کامل رو در پیوست به اسم drag & drop آوردم.

```
1  myMovieClip.addEventListener(MouseEvent.MOUSE_DOWN, onStartDrag);
2  myMovieClip.addEventListener(MouseEvent.MOUSE_UP, onStopDrag);
3  function onStartDrag(evt:MouseEvent):void {
4      evt.target.startDrag();
5  }
6  function onStopDrag(evt:MouseEvent):void {
7      evt.target.stopDrag();
8  }
```

نکته: evt که در بالا توضیح داده بودم اینجا استفاده شده. اینجا evt.target.startDrag() یعنی ببین رخداد به چی وصله(اینجا به مووی کلیپ) به هر چی که وصل بود اون شی رو شروع کن به drag کردن یا کشیدن به همراه ماوس.

! توجه کنید که target یه نوع property هست.

برای امروز دیگه بسه در درس آینده کنترل property ها بوسیله event ها و اگه برسم method ها رو توضیح خواهم داد.

درس یازدهم:

بسم الله الرحمن الرحیم

برای این جلسه فقط می خوام چگونگی استفاده از رخدادها(events) برای کنترل ویژگیها(properties) رو توضیح بدم. کل بحث این جلسه یه مثاله ، یه مثال جالب و پروپیمون که بتونید کلی چیز ازش یاد بگیرید و خیلی از مفاهیمی که از اول تا حالا در موردش بحث کردیم رو در عمل تست کنید.

طرح مساله:

قراره که یه موی کلیپ داشته باشیم و به وسیله کلیدهای مختلف بتونیم ویژگیهای ذکر شده در جدول درس قبلی برای موی کلیپ رو تغییر بدیم از قبیل اندازه و موقیت و stop و play و alpha و...

در ابتدا باید یه تو پروژمون interface ای به صورت زیر درست کنیم

همونطور که در شکل می بینید یه موی کلیپ داریم و چندین کلید که به وسیله این کلیها موی کلیپ رو هدایت خواهیم کرد.

نام گذاری رو به صورت زیر انجام می دیم (برای نام گذاری ، کلید یا موی کلیپ رو انتخاب و از پنجره پایین به نام پنجره properties در قسمت instance name نام رو وارد می کنیم):

نام مووی کلیپ : mc

نام کلیدهای موقعیت: pd - pu - pr - pl

نام کلیدهای زوم: zd - zu

نام کلیدهای چرخش : rl - rr

نام کلیدهای وضوح: au - ad

نام کلید نمایش: vb

می رسیم به کدنویسی. به فریم اول کدهای زیر رو اعمال می کنیم. البته من برای توضیح میون کدها پریدم ، شما به بزرگی خودتون ببخشید

در اولین گام برای هرکدوم از کلیدهامون یه خاله زنک ببخشید eventlistenr قرار میدیم. به شکل زیر:

```
pu.addEventListener(MouseEvent.CLICK,positionu);
pd.addEventListener(MouseEvent.CLICK,positiond);
pl.addEventListener(MouseEvent.CLICK,positionl);
pr.addEventListener(MouseEvent.CLICK,positionr);
zu.addEventListener(MouseEvent.CLICK,zoomu);
zd.addEventListener(MouseEvent.CLICK,zoomd);
rl.addEventListener(MouseEvent.CLICK,Rotationr);
rr.addEventListener(MouseEvent.CLICK,Rotationl);
au.addEventListener(MouseEvent.CLICK,Alphau);
ad.addEventListener(MouseEvent.CLICK,Alphad);
vb.addEventListener(MouseEvent.CLICK,Visible);
```

هر کلیدی با رخداد کلیک ، تابع مربوط به خودش رو اجرا مـی کنـه. تـابع هـا هـم کـه بنـا بـه وظایفشـون دستکاری تو خصوصیات مووی کلیپ رو انجام میدن. فکر نکنم نیاز به توضیح بیشتر باشـه. ایـنم تـابع هـای مربوطه:

```
function positionu(evt:MouseEvent){
    mc.y-=5;
}

function positiond(evt:MouseEvent){
    mc.y+=5;
}

function positionl(evt:MouseEvent){
    mc.x-=5;
}

function positionr(evt:MouseEvent){
    mc.x+=5;
}

function zoomu(evt:MouseEvent){
    mc.scaleX+=.1;
    mc.scaleY+=.1;
}

function zoomd(evt:MouseEvent){
    mc.scaleX-=.1;
    mc.scaleY-=.1;
}

function Rotationr(evt:MouseEvent){
    mc.rotation+=15;
}

function Rotationl(evt:MouseEvent){
    mc.rotation-=15;
}

function Alphau(evt:MouseEvent){
    mc.alpha+=.1;
}

function Alphad(evt:MouseEvent){
    mc.alpha-=.1;
}

function Visible(evt:MouseEvent){
    mc.visible=!mc.visible
}
```

در صورتی که مطالب درسای قبلی رو خونده باشید تمامی مثال رو به راحتی میفمید و دیگه نیازی به توضیح من نیست.

در مورد بقیه event ها مثل eventهای صفحه کلید و ... در درسهای آینده مثالهایی خواهیم داشت.

در جلسه آینده به متدها methods خواهیم پرداخت.

درس دوازدهم:

بسم الله الرحمن الرحیم

همونطور که قرار بود درس امروز رو با معرفی method ها شروع می کنیم تا ببینیم در ادامه به کجا می رسیم.

Methods

متدها در واقع فعلهای اکشن اسکریپت هستن! همونطور که افعال در یک زبان(مثلا فارسی یا انگلیسی) بیان کننده انجام یه کاری هستند ، متدها هم در اکشن اسکریپت این کار رو انجام می دن و به آبجکت ها (مثلا کلیدها یا مووی کلیپ ها یا ...) می گن که چه کاری رو انجام بدن.

برای مثال شما می تونین به یه مووی کلیپ بگین که بایست! به وسیله متد stop()

دستور زیر به موی کلیپ box میگه که بایسته.

```
box.stop();
```

به بیان دیگه متدها function هایی هستن که خود اکشن اسکریپت به صورت آماده در اختیار ما قرار میده. البته این یه تعریف غیر رسمیه! بین خودمون بمونه!!!

متدها در اصل توابعی هستن که برای کلاس نوشته می شن. یعنی اینکه در آینده که شما بتونین کلاس بنویسین هر تابعی(function) که برای کلاستون تعریف کنین اون میشه یه متد برای کلاستون.

اینجاست که یه کشف بزرگ می کنیم. بله متدهای خود اکشن اسکریپت در حقیقت خودشون تابع هایی هستن در یه سری کلاس. یعنی شما موقعی که از یه متد استفاده می کنین در حقیقت دارین از یه کلاس کار می کشین. اینجاست که می فهمیم که as3 یه زبان شی گراست. سرتون رو درد نیارم همه این مباحث باشه برای فصل شی گرایی.

شاید مناسب ترین کار در این زمان یه مثال باشه!! که هم توش از یه سری متد ساده استفاده کنیم و هم کار با event صفحه کلید رو یاد بگیریم.

```
function onKeyPressed(evt:KeyboardEvent):void {
    switch (evt.keyCode) {
        case Keyboard.ENTER:
            box.play();
            break;
        case Keyboard.BACKSPACE:
            box.stop();
            break;
        case Keyboard.LEFT:
            box.prevFrame();
            break;
        case Keyboard.RIGHT:
            box.nextFrame();
            break;
        case Keyboard.SPACE:
            box.gotoAndStop(3);
            break;
        default:
            trace("keyCode:", evt.keyCode);
    }
};
```

در مورد مثال من انتظار دارم که اگه پیگیر درسا بوده باشین خودتون همش رو بفهمین ولی بازم چندتا نکته ظریفش رو توضیح می دم.

۱- در مورد دستور swish تو درسای قبلی توضیح دادم. در مورد آرگومان ورودی اون که اینجا keycode هست. کار این دستور اینه که کد اسکی کلیدی که فشار داده میشه رو بر می گردونه و در دستورات پایینی بوسیله case ها با کد اسکی کلیدهای enter و backspace و left و... مقایسه میشه و کار مربوطه رو انجام میده.

۲- کد اسکی برای صفحه کلید در حقیقت یه کد دو رقمیه. هر کلید از صفحه کلید یه کد داره که بشه اون رو تشخیص داد. مثلا کد enter، 13 هست. تو خط آخر switch ما مشخص کردیم که اگه هر کلیدی به غیر از کلیدایی که ما مشخص کردیم فشار داده شد کد اسکی اون رو چاپ کن.

۳- برای تست برنامه بالا موقعی که برنامه رو تست می کنین شاید کلیدای enter و backspace کار نکنن. برای فعال کردنشون از گزینه control< disable keyboard shortcut رو تیک بزنین.

۴- برنامه رو اتچ میکنم. که دیگه مشکلی نباشه.

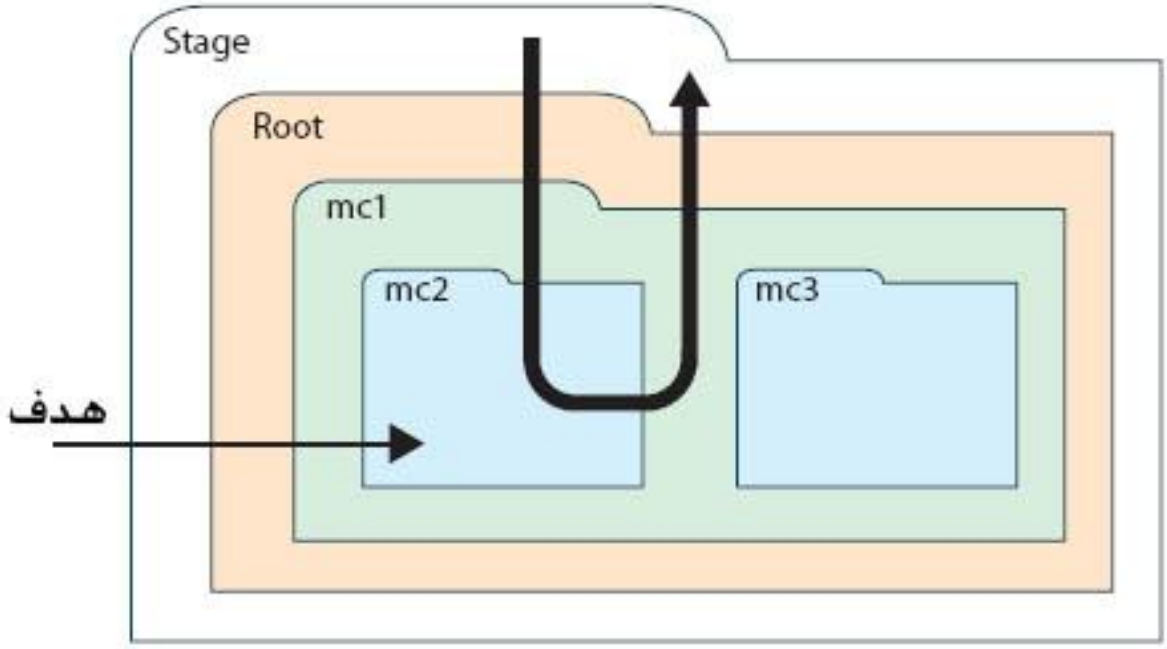
Event propagation (انتشار رخداد)

در مورد diplay list در دو سه درس آینده به صورت کامل توضیح خواهم داد. ولی محض اطلاع اینو بدونین که مفهوم displaylist با as3 معرفی شد.

Displaylist تمام آبجکت ها یی که در پروژمون وجود داره رو شامل میشه. Displaylist از stage گرفته تا swf های لود شده ،کلیدها(button) ، مووی کلیپها ، شیپ ها و... هر چی که داخل اینا و در کل هر آبجکتی که در پروژمون هست رو شامل میشه.

در حقیقت موقعی که ما یه event رو برای یه آبجکت مثلا مووی کلیپ می فرستیم اون رخداد از ابتدای displaylist شروع به حرکت می کنه و شاخه به شاخه میاد پایین و به مووی کلیپ می رسه. یعنی در حقیقت event قبل از رسیدن به مووی کلیپ از روی stage و root هم عبور میکنه و جواب هم از این مسیر به صورت برعکس برگشت داده میشه. چون اینا تو displaylist در ردیف بالاتری از مووی کلیپ هستن.

بزرارین یه مثال بزنیم. تصور کنین که یه مووی کلیپ داریم به نام mc1 که دو تا مووی کلیپ دیگه تو دل خودش داره به نام های mc2 و mc3 پس شکل رفت برگشت یه event به صورت دیاگرام زیر خواهد بود.



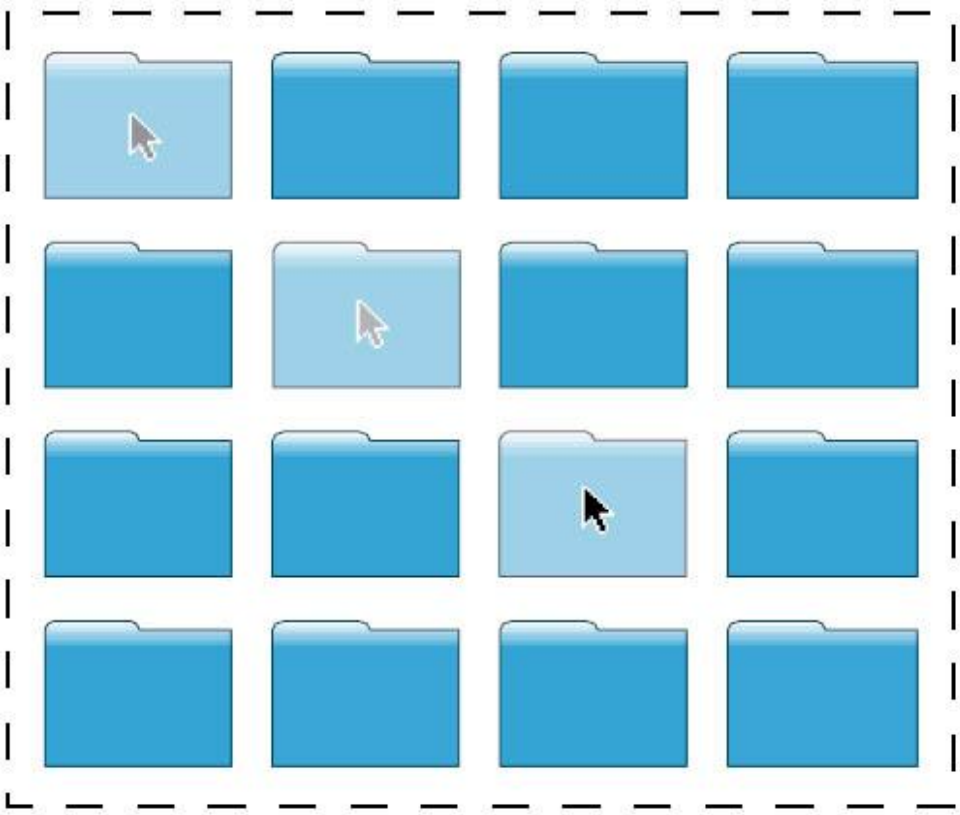
وقتی یه event رو می فرستیم به mc2 در حقیقت event مستقیما به mc2 رجوع نمیکنه بلکه ابتدا از روی stage بعد روی اون که swf جاری رو لود کرده (در اینجا root). و بعد به mc1 و سپس به mc2 میرسه و برای برگشت نتیجه مسیر زیر رو برعکس طی می کنه.

یه مثال کاربردی قشنگ از event propagation رو ببینید.

در ابتدا یه آبجکت به شکل فولدر طراحی کنید به این صورت و تبدیلش کنین به مووی کلیپ. یا همین عکس رو import کنین تو فلش.

بعد چند تا نمونه از اون بسازین و تک تک اسم گذاریشون کنین به اسم های folder0 و folder1 و...

حالا همشون رو انتخاب کنین و گروپشون کنین(ctrl+g). گروپ رو انتخاب کنین و اسمش رو بزارین folder_group. در مرحله بعد روی فریم اول کلیک راست کنین و کد زیر رو بنویسین.



```
folder_group.addEventListener(MouseEvent.MOUSE_OVER, onFolderOver);
folder_group.addEventListener(MouseEvent.MOUSE_OUT, onFolderOut);

function onFolderOver(evt:MouseEvent):void {
	evt.target.alpha = 0.5;
}

function onFolderOut(evt:MouseEvent):void {
	evt.target.alpha = 1;
}
```

همونطور که از کدها متوجه شدین ، با نوشتن event های mouse over و mouse out برای folder_group این event ها به زیر مجموعه folder_group که folder ها بودن انتشار پیدا کردن. فکرش رو بکنین اگه مجبور بودین برای هر folder دو تا رخداد رو بنیسین چقدر زحمت داشت! من برا دو تاش نوشتم شد این! حالا شما برا تمرین ۱۲۰ تاش بنویسین

```
ro.addEventListener(MouseEvent.MOUSE_OVER, onFolderOver);
ro.addEventListener(MouseEvent.MOUSE_OUT, onFolderOut);
r1.addEventListener(MouseEvent.MOUSE_OVER, onFolderOver);
r1.addEventListener(MouseEvent.MOUSE_OUT, onFolderOut);

ion onFolderOver(evt:MouseEvent):void {
 evt.target.alpha = 0.5;


ion onFolderOut(evt:MouseEvent):void {
 evt.target.alpha = 1;
```

نکته: همه event ها قابلیت انتشار ندارن، مثلا enter frame. برای اطلاعات بیشتر به help فلش مراجعه کنین.

درس بعد رو با بررسی timer events ادامه خواهیم داد.

درس سیزدهم:

بسم الله الرحمن الرحیم

تو درسای قبلی با event ها یا رخدادها آشنا شدیم و با دو تا از event های معروف (mouse و keyboard) آشنا شدیم. در این درس قصد داریم با دو رخداد مهم و جدید دیگه(frame event و mouse event) آشنا بشیم. البته اکشن اسکریپت ۳ رخدادهای زیادی داره که با بعضی های دیگه در درسای آینده آشنا خواهیم شد ولی بالتبع گفتن همشون تو درسا مقدور نیست.

Frame Events

این رخداد بر خلاف دو رخداد قبلی که در موردشون بحث کردیم بوسیله کاربر اتفاق نمی افته یا به اصطلاح trigger نمیشه بلکه رخداد اون بوسیله خود فلش پلیر انجام می گیره. رخداد enter frame با قرار

گرفتن playhead روی هر فریم اتفاق می افته. از این رخداد میشه برای آپدیت کردن ویژگی های یه آبجکت استفاده کرد. یه مثال کوچیک در این مورد می زنم تا کار با این event رو یاد بگیرید . مثال به قدری سادست که نیازی به توضیح نمی بینم.

```
stage.addEventListener(Event.ENTER_FRAME,onFrameLoop);

function onFrameLoop(evt:Event):void {
    cycle.x = mouseX;
    cycle.wheel.rotation = mouseX;
}
```

نکته جالب در مورد این مثال اینه که اکشن اسکریپت خودش زاویه از ۳۶۰ به بالا رو تجزیه تحلیل میکنه و نیازی نداره شما نگران باشین

Timer Events

مواقعی که دوست داریم یه اتفاقی در زمان مشخصی انجام بشه دست به دامن تایمر می شیم. تایمر هم معمولا برای آپدیت ویژگیهای آبجکت ها استفاده میشه ولی کاربردای دیگه ای هم داره.

در مورد استفاده از timer Event چند تا نکته ریز رو باید بدونیم .

اولا اینکه برای استفاده از این رخداد باید یه شی از جنس کلاس تایمر بسازیم. و مقدار دهی اولیه هم بکنیم. فعلا اینا رو بدونین تا تو درس شی گرایی دقیق بهشون بپردازیم. دو تا آرگومان می تونیم برای تابع سازنده

کلاس ارسال کنیم که اول محدوده زمانی رخداد ماست و دومی که اختیاری هم هست اینکه این رخداد چند بار تکرار بشه

```
var timer:Timer = new Timer(delay:Number, repeatCount:int);
```

دوما اینکه پارامتر اول به میلی ثانیه هست. یعنی برای اینکه رخدادمون هر یک ثانیه اتفاق بیفته باید پارامتر اول رو ۱۰۰۰ بدیم.

سوما اینکه برای راه افتادن listener تایمر باید اون رو start کنیم. حتی می تونیم هر زمان که بخوایم اون رو stop کنیم.

بریم سر مثال. تو مثالمون یه ثانیه شمار سادس که کدش هم کاملا واضحه

```
var timer:Timer = new Timer(1000);
timer.addEventListener(TimerEvent.TIMER, onTimer);
timer.start();

function onTimer(evt:TimerEvent):void {
    watch.hand.rotation +=5;
}
```

فعلا دیگه وقت ندارم اینا باشه تا تو درس آینده با چندتا مثال دیگه فصل رو تموم کنیم.

درس چهاردهم:

بسم الله الرحمن الرحیم

همونطوری که قرار بود فصل دوم رو با درس امروز تموم می کنیم. تو درس امروز یه مثال خیلی ساده از timer event که جلسه قبلی معرفیش کردیم داریم. از مطرح کردن این مثال سه تا هدف دارم:

۱. با timer event که خیلی مهمه بیشتر آشنا بشین

۲. یه بازی ساده رو بنویسین که ببینین بازی نوشتن تو فلش خیلی ساده و راحته

۳. در حد آشنایی با display list آشنا بشین که قراره فصل بعد رو کاملا به اون بپردازیم

نکته: کد برنامه خیلی سادس به این دلیل که نمی خواستم زیاد شلوغ بشه. ایشالله بعد از بحث display list کاملش می کنیم و نقایصش رو برطرف می کنیم.

بازی شلیک به خرگوش

طرح مساله:

می خوایم یه بازی رو طراحی کنیم که هر یک ثانیه یک خرگوش یه جا از صفحه ظاهر بشه. اگه کاربر روش کلیک کنه یه پیغام چاپ کنه

حالا یکی یکی موارد قرمز رنگ رو بررسی می کنیم.

هر یک ثانیه: پس باید یه timer بذاریم که هر یه ثانیه یه خرگوش (موی کلیپ) رو نمایش بده

یه جا از صفحه: چون می خوایم اون یه جا اتفاقی انتخاب بشه که کاربر نتونه از قبل پیش بینیش کنه باید به صورت اتفاقی x و y اون رو انتخاب کنیم. که این کار هم بوسیله تابع random از کلاس math قابل انجامه

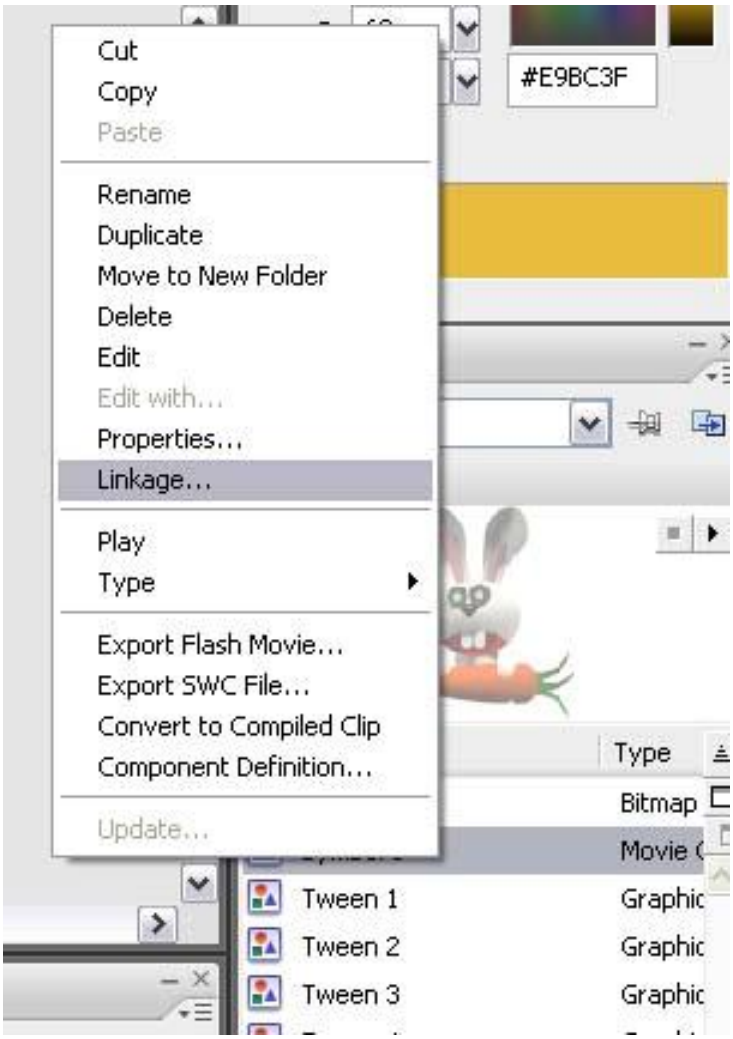
روش کلیک کنه : پس باید یه mouse event هم داشته باشیم.

خوب، اینم از کد برنامه:

```
var hadaf:MovieClip = new sibl();
addChild(hadaf);
hadaf.x=0;
hadaf.y=0;

var timer:Timer = new Timer(1000);
timer.addEventListener(TimerEvent.TIMER, onTimer);
timer.start();

function onTimer(evt:TimerEvent):void {
var a:int = Math.random()*500;
var b:int = Math.random()*350;
removeChildAt(0);
var hadaf:MovieClip = new sibl();
hadaf.x=a;
hadaf.y=b;
addChild(hadaf);
this.addEventListener(MouseEvent.CLICK,shot);

}

function shot(evt:MouseEvent):void {
trace("ok");
}
```

تو خط اول یه نمونه از نوع مووی کلیپ از کلاس sibl می سازیم و اسمش رو می ذاریم hadaf. حالا ببینیم اصلا sibl چی هست. sibl در حقیقت همون خرگوش (مووی کلیپیه) که ما می خوایم روی صفحه ظاهر بشه و کاربر بهش شلیک کنه.

موقعی که ما مووی کلیپمون (همون خرگوش) رو می سازیم روش کلیک راست می کنیم و گزینه linkage رو انتخاب و از منوی باز شده export for action script رو تیک می زنیم و اسمش رو می ذاریم sibl. این باعث می شه که یه کلاس به این نام از این مووی کلیپ ایجاد بشه و ما هر موقعی که تو طول برنامه نیاز ش داشتیم یه نمونه ازش بسازیم(مثل خط اول این کد)


Cut
Copy
Paste
Rename
Duplicate
Move to New Folder
Delete
Edit
Edit with...
Properties...
Linkage...
Play
Type
Export Flash Movie...
Export SWC File...
Convert to Compiled Clip
Component Definition...
Update...
#E9BC3F
Type
Bitmap
Tween 1
Tween 2
Tween 3


Linkage Properties
Identifier:
Class:
sibl
Base class:
flash.display.MovieClip
Linkage:
Export for ActionScript
Export for runtime sharing
Export in first frame
Import for runtime sharing
URL:
OK
Cancel

نکته: تو as3 تنها راه استفاده از مووی کلیپ به همین صورته!

خوب، تو خط دوم با استفاده از دستور addchild این نمونه از مووی کلیپ رو که ساختیم به stage(صفحه نمایش) اضافه می کنیم و اونو نمایش میدیم. اگه این کار رو نکنیم نمونه از مووی کلیپ تو حافظه ساخته می شه ولی نمایش داده نمیشه! addchild جزو مباحث displaylist هست که تو فصل بعد قراره بهش بپردازیم.

تو خطوط ۳ و ۴ هم که واضحه! x و y نمونه مووی که ساختیم رو صفر می ذاریم.

نکته: کاری به این نداشته باشین که چرا این ۴ خط رو نوشتم. بعد از فصل displaylist بهش می پردازیم. اما اگه کسی دلیلش رو بگه معلومه که بچه باهوشیه

خوب تو خطوط بعدی یه timer می سازیم و هر یه ثانیه فعالش می کنیم و تابع onTimer رو صدا می زنیم.

توی تابع onTimer ابتدا دو تا عدد تصادفی تولید می کنیم و می ریزیم توی دو ا متغیر به نام a و b.که به ترتیب برای x و y خرگوشمون استفاده خواهیم کرد.

نکته: اون عدد ۵۰۰ و ۳۵۰ جلوی تابع رندوم برای اینه که عدد تولید شدمون در محدوده ۰ تا ۵۰۰ و ۰ تا ۳۵۰ به ترتیب طول و عرض صفحمون هستن باشه.

در خط بعدی removeChildAt(0) باعث میشه مووی کلیپی که در مرحله قبل به stage به عنوان فرزند اضافه شده حذف بشه(این خط رو حذف کنین ببینین چی میشه). در حقیقت بچه های سطح صفر رو در displaylist حذف می کنه. این هم مربوط به همون بحث جلسه بعدمون که قبلا ذکر شد.

پس می بینیم که برای کدنویسی hot با as3 حتما باید از displaylist استفاده کنیم!

در خطوط بعدی یه نمونه از مووی می سازه و x و y اش رو اعداد تصادفی تولید شده توسط تابع رندوم میده. در حقیقت در هر مرحله اجرای تابع(هر یک ثانیه) یه نمونه از مووی ساخه میشه و یه eventlistenet از نوع کلیک ماوس بهش وصل میشه که میگه اگه روی این مووی کلیک شد چاپ کن ok.

نکته: هر کی که درسا رو پیگیری کرده و می تونه این نمونه بازی رو پرورش بده و کاملترش کنه لطفا تو پستای بعدی نمونه خودش رو بزاره تا هم منون خوشحال کنه و هم به بقیه کمک کنه. فقط خواهشا کد رو پیچیده نکنین که تازه کارا راحت بفهمن.

پاک کردن event listener ها:

موقعی که ما یه eventlistener رو به کار می گیریم در حقیقت داریم از کامپیوتر حافظه می گیریم. اگه بعد از اینکه کارمون با اون event listenr تموم شد حافظه اشغال شده رو آزاد نکنیم به مرور حافظه زیادی از کامپیوتر گرفته می شه که باعث کندی دستگاه میشه.

با دستور removeEventListener می تونیم این کار رو بکنیم. مثال زیر رو ببینین و اجراش کنین تا ببینین چی میشه. نیازی به توضیح نمی بینم. کاملا واضحه!!!

```
var timer:Timer = new Timer(1000);
timer.addEventListener(TimerEvent.TIMER, onTimer);
timer.start();

function onTimer(evt:TimerEvent):void {
    watch.hand.rotation +=5;
        if (watch.hand.rotation >= 25) {
                    timer.removeEventListener(TimerEvent.
TIMER, onTimer);
        }
}
```

نکته: اکشن اسکریپت خودش یه موتوری به نام gorbage collector هم داره که هر چند وقت یه بار حافظه هایی که در زمان اجرا از کامپیوتر گرفته شده و دیگه کاری بهشون نداریم به صورت اتوماتیک رو آزاد می کنه

در اینباره تو درسای بعدی با یه مثال توضیح خواهیم داد.