آموزش اکشن اسکریپت ۳ از سطح صفر

Learning ActionScript 3.0

گردآوری:
سید محمد جواد نیکوکار

Adobe

درس پانزدهم:

بسم الله الرحمن الرحیم

درس امروز فقط بهانه ایه برای شروع فصل جدیدمون که displaylist هست. یکی از مفاهیم مهم و کلیدی که تو AS3 معرفی شد و خیلی از مشکلات As2 رو برطرف کرد. چون مطلب آموزش As3 هست و فرض من از اول بر اون بوده که خواننده های این تاپیک صفر هستن اصلا AS3 رو با As2 مقایسه نمی کنم .

در این فصل

۱- ابتدا displaylist رو معررفی می کنیم

۲- ویژگیهای کلاسای مربوط به displaylist رو بیان می کنیم

۳- اضافه و حذف کردن فرزند(موی کلیپ-کلید-تصویر و ... هر آبجکتی) به displaylist

۴- مدیریت displaylist از قبیل تغییر نام آبجکتا و تغییر مکانشون و اندازه و ...

۵- مدیریت سلسله مراتب فرزندای لیست نمایش و تغییر اونا

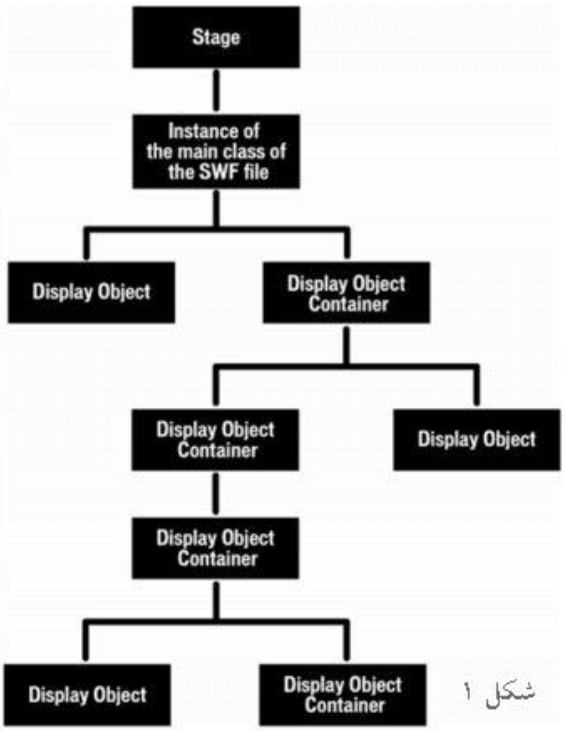
۶- و در نهایت سعی خواهم کرد یه مثال جون دار نون آب دار و مشتی و مشتری پسند بزنم که کلیه ۵ مورد بالا رو به صورت عملی تست کرده باشیم

معرفی displayllist

Displaylist در بیان ساده وسیله ای هست برای اینکه بتونیم آبجکتهایی مثل مووی کلیپ یا کلید یا تصویر(Bitmap) و یا text و... رو در زمان اجرا به پروژه که همون فایل swf امون هست رو اضافه کنیم. این اضافه سازی به صورت سلسله مراتبی صورت می گیره و آبجکتای اضافه شده تشکیل یه درخت رو می دن که هر برگش می تونه یه نوع آبجکت باشه.

خوب. حالا این displaylist برای خودش یه ساختاری داره که برای کار با اون باید این ساختار رو درست بشناسیم.

همونطور که در شکل زیر(شکل ۱) می بینیم بالاترین سطح در displaylist یه DisplayObjectContaner هست به نام stage. در پایین stage تایم لاین اصلی (فایل swf اصلی برنامه) وجود داره.



شکل ۱

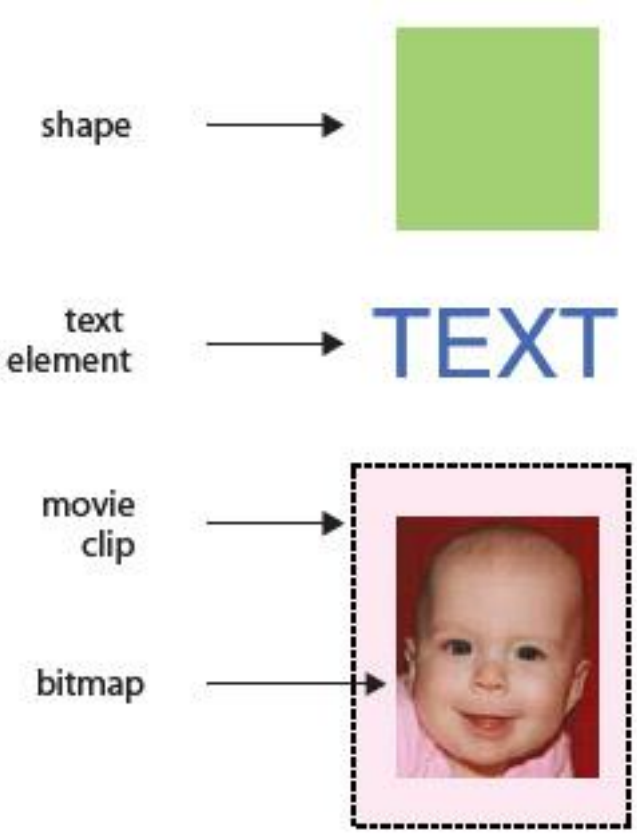
نکته: DisplayObjectContaner یه نوع DisplayObject هست که خودش می تونه شامل المنت های دیگه و یا displayobject های دیگه باشه. برای مثال در ادمه خواهید دید که تو displaylist کلاسهای bitmap , shape و ویدئو داریم که هر کدوم از اینا یه displayobject(شی نمایش) هستن ولی نمی تونن شامل چیز دیگه ای باشن. ولی movieclip علاوه بر اینکه یه diplayobject هست می تونه شامل displayobject های دیگه هم باشه که این موضوع باعث می شه که مووی کلیپ یه DisplayObjectContaner هم باشه.

خوب برگردیم به شکل.

همونطوری که از شکل پیداست تو displaylist ما همیشه یه شی stage داریم که داخل این stage یه شی از تایم لاین اصلی ما وجود داره. تا اینجاش همیشه وجود داره.

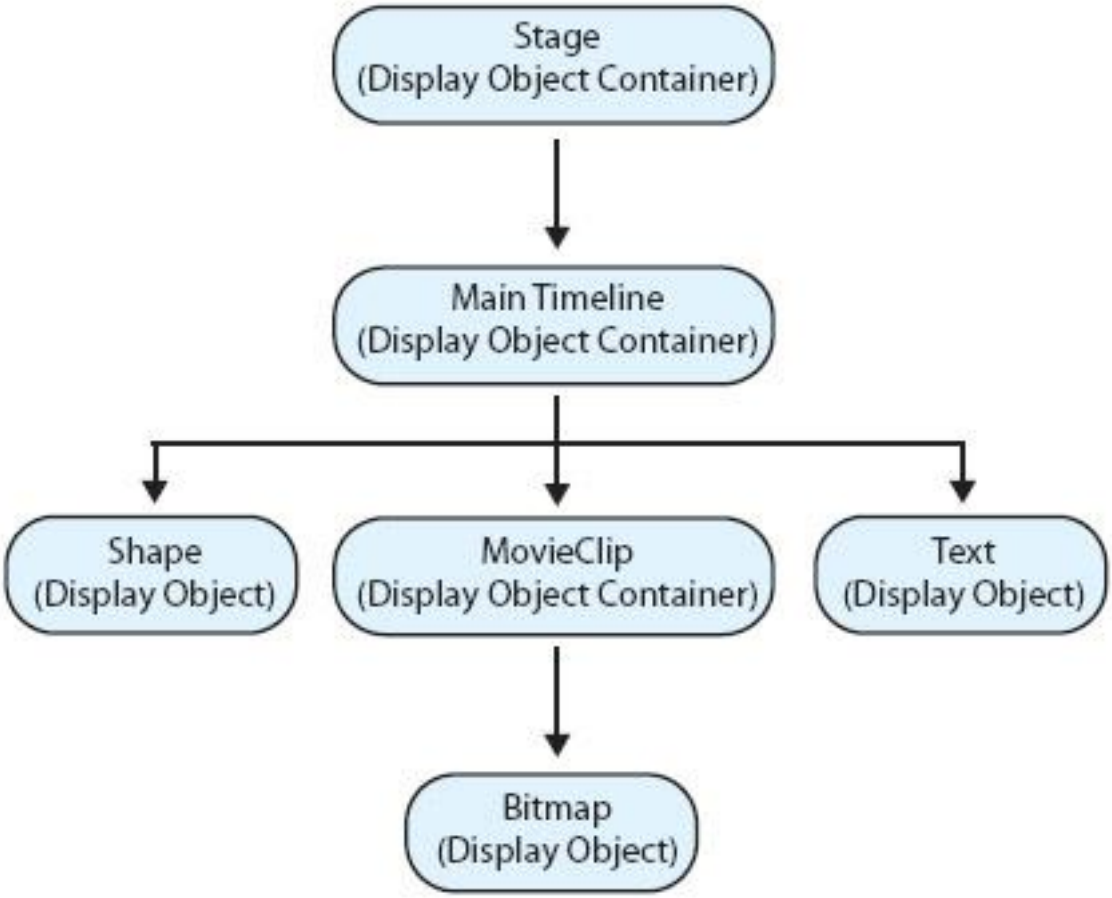
حالا در داخل این شی تایم لاین ما هر displayobject و DisplayObjectContaner ای می تونیم داشته باشیم. این که چند تا displayobject و از چه نوعی در داخل شی تایم لاین باشه دیگه به عهده خود ماست

- ✓ برای اینکه توضیحات بالا بیشتر بهمون بچسبه مثال زیر رو بررسی می کنیم.



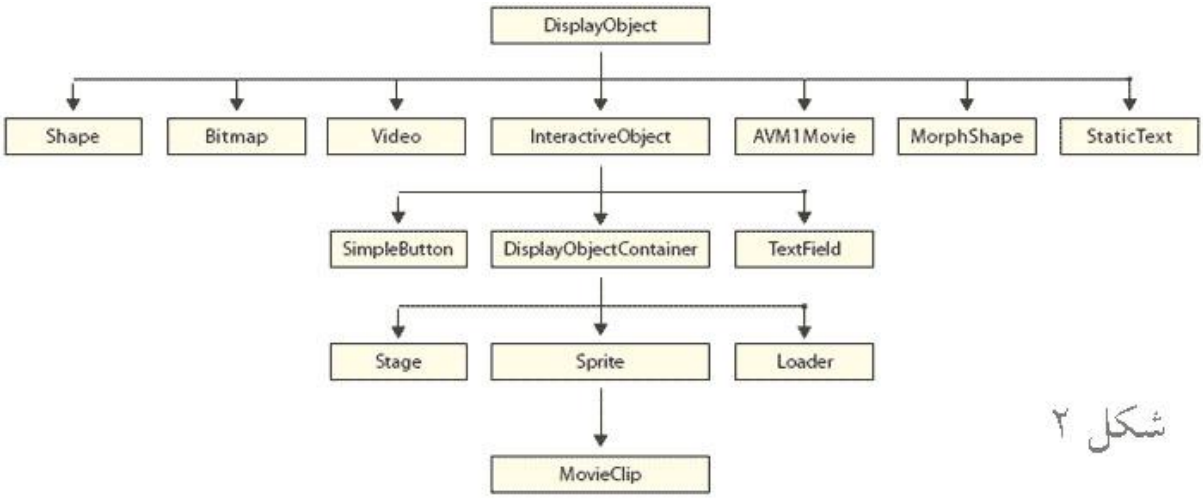
فرض کنین ما یه پروژه رو تو فلش باز می کنیم و بی درنگ یه shape تو پروژه مون رسم می کنیم. مثلا یه مربع. بعد از اون یه رو صفحه تایپ می کنیم. در ادامه عکس این آقا کوچولو رو که در حقیقت یه bitmap هست رو به داخل پروژمون import میکنیم و اون رو انتخاب کرده و اون رو تبدیل به مووی کلیپ می کنیم. پس یه مووی کلیپ حاوی bitmap این آقا کوچولو ایجاد می شه. پس مووی کلیپ اینجا یه کانتینر حاوی bitmap میشه.

تمام این مثال رو مطرح کردیم که بگیم آقاجووووا!(به قول ما کاشونیا) شکل پایین میشه displaylist این مثالمون.



فکر کنم دیگه با این مثال و شکل مفاهیم پایه ای display list رو کاملا فهمیده باشین.

شکل زیر کلاسهای مختلف displaylist رو نمایش میده.



شکل ۲

نکته: همونطور که تو شکل میبینیم آخرین سطح مووی کلیپه. ولی اشتباه نکنین!!! این به این معنی نیست که مووی کلیپ در سطح پایین تری نسبت به بقیه displayobject ها قرار داره. در حقیقت این نمودار داره نوع کلاسها و والدهای اونارو نشون می ده. مثلا در مورد مووی کلیپ می گه:

مووی کلیپ یه نوع Sprite هست که این Sprite خودش یه نوع DisplayObjectContainer هست که این DisplayObjectContainer خودش یه نوع InteractiveObject هست که این

InteractiveObject خودش یه displayobject هست

ولی shape فقط و فقط و فقط یک نوع displayobject هست و بس!!!!

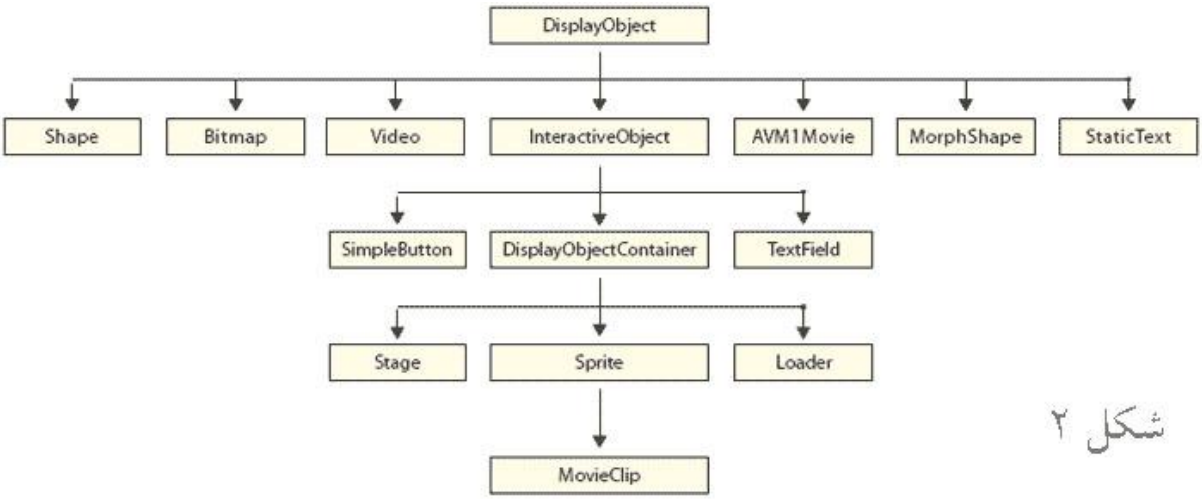
شکل رو خودتون دقیقتر بررسی کنین تا تو درس بعدی کلاسای اون و ویژگیهاشون رو یکی یکی بررسی کنیم.

jav1366@yahoo.com

درس شانزدهم:

بسم الله الرحمن الرحیم

برای این جلسه کلاسای شکل زیر رو که درس قبلی قولش رو داده بودم ، به صورت اجمالی توضیح می دیم.



شکل ۲

:DisplayObject

هر چیزی که می تونه تو لیست نمایش وجود داشته باشه یک شیء نمایشه(DisplayObject) . همونطور که تو شکل می بینید اکثر کلاس های تخصصی از DisplayObject مشتق شدن.

:Shape

shape در حقیقت یک مستطیل ، بیضی ، خط ، و یا هر چیزی هست که به وسیله ابزارهای رسم خود فلش ایجاد شده.

تو اکشن اسکریپت ۳.۰ ، این ویژگی اضافه شده که شما می تونید در زمان اجرا shape ایجاد کنید

:Bitmap

یک bitmap ایجاد شده در زمان اجرا بوسیله اکشن اسکریپت و با استفاده از کلاس BitmapData است. توجه داشته باشید که یک JPG که از بیرون import میکنیم این نوع از bitmap حساب نمیشه، بلکه یه نوع shape حساب میشه.

پس از ایجاد یک bitmap با این کلاس ،می تونیم یک JPG رو به اون وارد کنید و نمایشش بدیم

:Video

یه آبجکت نمایش ویدئو ، حداقل چیزیه که برای نمایش یه فیلم لازمه!! صرف نظر از هر ویدیو کامپوننتی!

:InteractiveObject

این کلاس شامل هر شی در صفحه نمایشه که کاربر می تونه با استفاده از موس و یا کیبورد باهاش تعامل داشته باشه. از این کلاس برای دستکاری در لیست نمایش داده به صورت مستقیم استفاده نمی کنیم. در عوض ، با فرزندان اون(که در شکل می بینید) کار می کنیم

:SimpleButton

این کلاس برای ایجاد کلیدی استفاده میشود که از نظر کارایی شبیه دکمه هایی است که احتمالا تجربه کار کردن با اونا رو در حالت طراحی عادی داشتین .(همون کلید عادی که تو کتابخونه فلش پیدا میشه یا می تونیم خودمون با button symble ایجادشون کنیم) .

تو در اکشن اسکریپت ۳.۰ ، می تونیم دکمه هایی در زمان اجرا (runtime) ایجاد کنیم و از اشیاء دیگر را برای حالت های up، down، over ، و hit کلید استفاده کنیم.

:TextField

این کلاس عناصر پویا و ورودی متن را شامل میشود، که قابل کنترل بااکشن اسکریپت هستند.

:DisplayObjectContainer

این کلاس شبیه به DisplayObject هست . تفاوت در اینجا اینه که این شیء می تونه شامل فرزندانی باشه.

تمام DisplayObjectContainer ها در حقیقت یه نوع DisplayObject هستند ولی تنها DisplayObject هایی که می تونن فرزند داشته باشن DisplayObjectContainer حساب

می شن.

بزارین یه مثال بزنیم که قضیه روشن تر بشه:

یه video همونطور که از شکل پیداست یه شی نمایشه(DisplayObject) ولی نمی تونه فرزندی داشته باشه اما یه movieclip یه نوع DisplayObject هست که می تونه فزندانی مثل bitmap، textو ... داشته باشه. پس movieclip یه نوع DisplayObjectContainer هم هست.

چهار نوع DisplayObjectContainer داریم که عبارتند از:

Stage:

stage ، به خودی خود بخشی از این لیست نمایشه. از این کلاس برای آدرس دهی هر شی تعاملی استفاده میشه ، همیشه اشیا داخل stage قرار دارن و برای آدرس دهی اونا می تونیم از stage استفاده کنیم. پس stage یه نوع DisplayObjectContainer هست.

Sprite:

sprite در حقیقت یه مووی کلیپ ساده ی بدون تایم لاینه! اکثر کارای حرفه ای که با اکشن اسکریپت انجام میشه حاوی مووی کلیپ هایی با یک فریم هستند. بنابراین دیگه مدیریت فریم ها و یا حرکت فریم به فریم برای این مووی کلیپها بی معنا هستن. خوب نتیجه می گیریم که چه کاریه؟!!! از sprite استفاده می کنیم

Loader:

این کلاس برای لود کردن آبجکتهای بیرونی جهت نمایش در display list استفاده میشه. مثل bitmap یا swf هایی که از بیرون لود می شن

MovieClip:

دقیقا همون چیزیه که تا حالا به عنوان مووی کلیپ می شناختیم .هیچ فرقی نداره. فقط اینجا با اکشن اسکریپت ایجاد و کنترل میشه.از اول درسا تا حالا باهاش کار کردیم.پس نیازی به توضیح نداره!!!.

خوب! اینم از DisplayObjectContainer ها!!!

حالا ادامه نمودار رو پیگیری می کنیم:

AVM1Movie:

swf هایی که با زبانهای اکشن اسکریپت ۱ و ۲ ساخته شدن بوسیله atction script virtual machin1 یا به قولی AVM1 اجرا میشن. در صورتی که زبان اکشن اسکریپت ۳ از AVM2 برای اجرا کردن کدها استفاده می کنه. این کلاس برای حل این مشکل تهیه شده. بله درست حدس زدین! این کلاس برای اجرای swf های لود شده از بیرون که بوسیله اکشن اسکریپت ۱ و ۲ ساخته شدن هست.

MorphShape و StaticText:

این دو کلاس به ترتیب برای ارائه motion shape و کار با متن های ثابت هستن. هردوشون در حقیقت یه نوع display object هستن و می تونن بوسیله اکشن اسکریپت کنترل بشن. مثلا یه متن به این صورت می تونه جابجا بشه یا چرخش داشته باشه.

آخیش تموم شد...!!! کلاسهایی رو که معرفی کردیم در درسهای آینده به صورت عملی به کار خواهیم برد. کافیه فقط یک بار هر کدوم رو مطالعه کنین تا فقط اگه جایی تو کدها دیدنشون بدونین چی هستن و چیکار می کنن . در همین حد کافیه.

از جلسه آینده به بعد دیگه به صورت عملی با display list کار خواهیم کرد. راستش خودمم از تئوری گفتن متنفرم  ، ولی یه وقتایی مباحث تئوری پیش نیاز کار عملی هستن و کاریش هم نمی شه کرد. می دونم مباحث این جلسه خسته کننده بوده، به همین خاطر این گل رو از طرف من داشته باشین شاید تلافی بشه 

درس هفدهم:

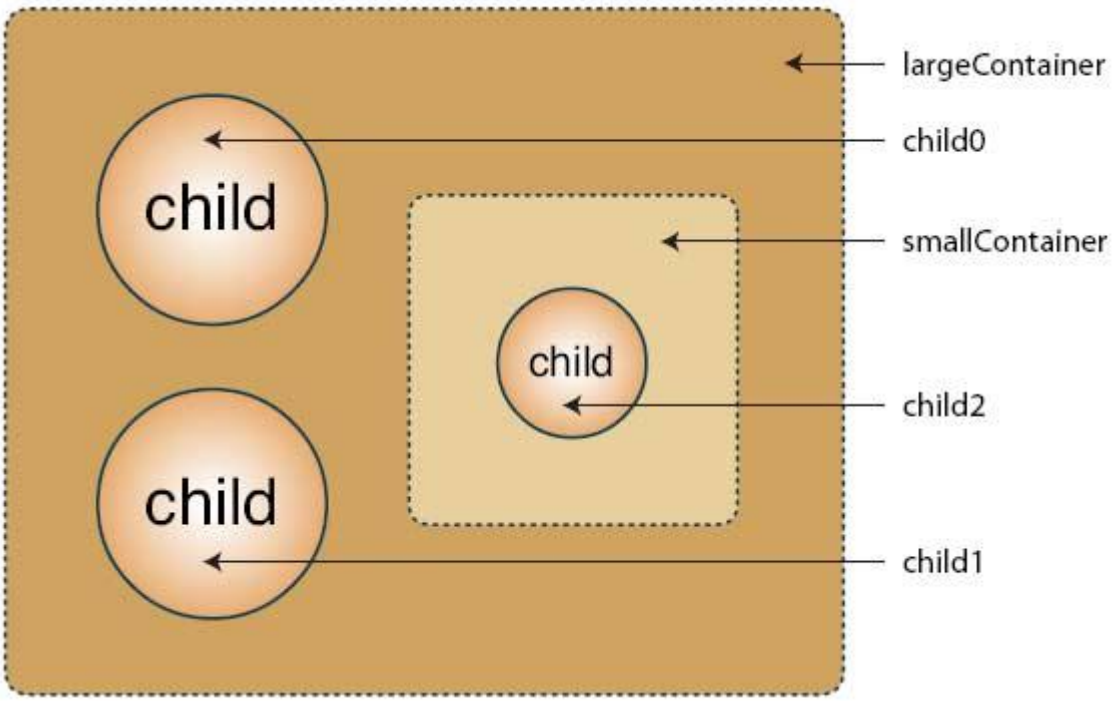
بسم الله الرحمن الرحیم

خوب همونطور که جلسه قبل قول داده بودم بریم سر مباحث عملی و استفاده از displaylist.

نمایش displaylist

بدون هیچ پیش توضیحی می ریم سر یه مثال. تو این مثال قصد داریم اعضا یا به عبارتی آبجکت های موجود در یک لیست نمایش رو نمایش بدیم. مثال رو در پیوست آوردم. کدش رو هم همینجا تحلیل می کنیم.ولی قبل خوندن کد بهتره که برنامه رو یه بار اجرا کنین تا راحت تر باهاش ارتباط برقرار کنین

شکل پایین در حقیقت محتویات برنامه ما یا به عبارتی محتویات displaylist ما رو نشون می ده.



---در ابتدا یه MovieClip به نام largeContainer

---داخلش سه عدد مووی کلیپ به نام های child0 و child1 و smallContainer

--- و داخل smallContainer یه مووی کلیپ به نام child2

✓ اینا رو داشته باشین تا بریم سراغ کد

```
function showChildren(dispObj:DisplayObject):void {
    for (var i:int = 0; i< dispObj.numChildren; i++) {
        var obj:DisplayObject = dispObj.getChildAt(i);
        if (obj is DisplayObjectContainer) {
            trace(obj.name, obj);
            showChildren(obj);
        } else {
            trace(obj);
        }
    }
}

showChildren(stage);
```

همونطور که تو خط اول میبینید تابع اصلی برنامه showChildren هست که کارش اینه که ایم آبجکت ها رو چاپ کنه. این تابع به عنوان ورودی یه DisplayObject می گیره و شناسه اون رو dispObj می ذاره.

تو خط دوم یه حلقه for داره که به تعداد فرزندان این DisplayObject عملیاتش تکرار میشه.

نکته: خط آخر رو نیگا کنین. تابع برای اولین بار stage رو به عنوان ورودی می گیره. بنابراین نتیجه می گیریم که عملیات تابع به تعداد فرزندان داخل stage که تمام آبجکت های موجود در برنامه هستن انجام خواهد شد.

تو خط سوم یه DisplayObject به نام obj می سازیم و فرزند سطح i ام dispObj(ورودی تابع) رو داخل اون قرار می دیم.

در خط بعدی بررسی می کنیم که آیا این فرزند از نوع DisplayObjectContainer هست یا نه؟

اگه نبود که فقط نوعش رو چاپ می کنیم. (قسمت else)

ولی اگه DisplayObjectContainer بود اسمش رو چاپ می کنیم و نوعش رو و دوباره خود همین DisplayObjectContainer رو به عنوان ورودی به تابع showChildren می فرستیم.(تابع بازگشتی که تو درسای قبلی توضیح دادم)

اگه پیگیر درسا بوده باشین مثال رو به راحتی میفهمین!!!!

دیدن کار کردن با displaylist چقدر راحته؟

اگه برنامه رو اجرا کرده باشین خروجی زیر رو در پنجره trace خواهید داشت.

```
root1 [object MainTimeline]
largeContainer [object largeContainer_1]
[object Shape]
smallContainer [object smallContainer_2]
[object Shape]
child2 [object MovieClip]
[object Shape]
[object StaticText]
child0 [object MovieClip]
[object Shape]
[object StaticText]
child1 [object MovieClip]
[object Shape]
[object StaticText]
```

بیاین یه مقدار کدمون رو دستکاری کنیم و خروجیمون رو خوشکلتر کنیم. یعنی با ایجاد تو رفتگی متناسب با سطح فرزندا ساختار درختی لیست نمایش رو به صورت واضح تری نمایش بدیم. وای چه جمله قلنبه ای!!! نگران نباشین مثال رو ببینین خودش باهاتون حرف می زنه و منظور منو می رسونه.

```
function showChildren(dispObj:DisplayObject, indentLevel:Number):
void {
    for (var i:int = 0; i < dispObj.numChildren; i++) {
        var obj:DisplayObject = dispObj.getChildAt(i);
        if (obj is DisplayObjectContainer) {
            trace(padIndent(indentLevel), obj.name, obj);
            showChildren(obj, indentLevel + 1);
        } else {
            trace(padIndent(indentLevel) + obj);
        }
    }
}

showChildren(stage, 0);

function padIndent(indents:int):String {
    var indent:String = "";
    for (var i:Number = 0; i < indents; i++) {
        indent += "    ";
    }
    return indent;
}
```

تابع showChildren رو به صورت زیر اصلاح می کنیم:

اول از همه تابع رو دو ورودی می کنیم. یه ورودی دیگه به نام indentLevel از نوع عددی به ورودی تابع اضافه می کنیم. indentLevel در حقیقت نشون دهنده سطح آبجکتمونه که الان ارسال شده به عنوان ورودی تابع.

نکته: بازم اگه توجه کنین می بینین که برای بار اول stage رو با صفر صدا زدیم یعنی این که سطح stage در لیست نمایش صفره.

خطوط دو سه و چهار هم که مثل برنامه قبلیه . تو خط پنجم موقعی که آبجکت از نوع DisplayObjectContainer باشه موقع اولا trace گرفتن سطح آبجکت جاری (indentLevel) رو برای تابع padIndent (که در ادامه بررسیش می کنیم) می فرستیم و سپس showChildren رو بصورت بازگشتی صدا می زنیم. البته برای showChildren سطح این آبجکت بعلاوه ۱ رو می فرستیم. کاملا واضحه که فرزند این آبجکت یه سطح از خودش پایین تره!!!!

تو خط هشتم هم که اگه آبجکت DisplayObjectContainer نبود سطحش رو برای padIndent می فرستیم هرچی که padIndent برگردوند به انتهاشم آبجکت رو می چسبونیم و چاپ میکنیم

بررسی تابع padIndent :

طرز کار این تابع بسیار بسیار سادست. به این صورت که به عنوان ورودی سطح آبجکت رو می گیره و شناسه indents رو بهش می ده.

در خط دومش متغیری به نام indent (توجه کنین این S نداره!!!) تعریف می کنه و مقدار رشته تهی رو بهش میده.

در خط بعدی یه حلقه for داره که به ازای عدد سطح آبجکت به indent چند کارکتر space اضافه می کنه.

در انتها همین مقدار indent که یه رشته هست رو برمیگردونه.

این مقدار برگردونده شده همونطور که تو تابع showChildren دیدن به ابتدای آبجکتی که قراره چاپ بشه اضافه میشه.

خروجی این برنامه رو ببینین که چه خوشکل و منظم شده!!!!

```
root1 [object MainTimeline]
     largeContainer [object largeContainer_1]
        [object Shape]
         smallContainer [object smallContainer_2]
            [object Shape]
             child2 [object MovieClip]
                [object Shape]
                [object StaticText]
         child0 [object MovieClip]
            [object Shape]
            [object StaticText]
         child1 [object MovieClip]
            [object Shape]
            [object StaticText]
```

در جلسه آینده شیوه های add کردن و remove کردن آبجکت به و از لیست نمایش رو توضیح خواهیم داد.

درس هجدهم:

بسم الله الرحمن الرحیم

اضافه سازی و حدف آبجکت ها:

تا الان فقط به توصیف ویژگی های displaylist آنالیز اون پرداختیم. ولی بحث مهمتری که باید یاد بگیریم چگونگی اضافه سازی و یا حذف فرزند به و از لیست نمایش در حالت اجرا(runtime) است.

نکته: هر عضوی که تو لیست نمایش وجود داره فرزند اون حساب میشه. هر عضوی هم که زیرشاخه یه عضو دیگه باشه فرزند اون حساب میشه. مثلا اگه داخل یه مووی کلیپ یه عکس داشته باشیم اون عکس فرزند اون مووی کلیپ حساب میشه

اضافه کردن آبجکت(فرزند) به لیست (addChild):

برای اضافه کردن آبجکت به لیست نمایش دو مرحله خیلی کوچیک رو پشت سر بذارین.

الف) ساختن یه آبجکت. مثلا ساختن یه آبجکت از نوع مووی کلیپ

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

ب) اضافه سازی آبجکت به لیست نمایش

```
addChild(mc);
```

نکته: در مرحله اول ما فقط آبجکت رو می سازیم بدون اینکه نمایشی از اون داشته باشیم. ولی با اجرای مرحله دوم آبجکت نمایش داده میشه.

jav1366@yahoo.com

حتی به راحتی این امکان وجود داره که ما آبجکتمون رو به عنوان فرزند یه آبجکت دیگه هم اضافه کنیم.
مثلا تو کد پایین ما همین mc رو به مووی کلیپی به نام navBar اضافه می کنیم.

```
navBar.addChild(mc);
```

مثل مووی کلیپ بقیه انواع آبجکت ها هم به همین سادگی به لیست اضافه می شن. مثلا

```
var sp:Sprite = new Sprite();
addChild(sp);

var sh:Shape = new Shape();
addChild(sh);
```

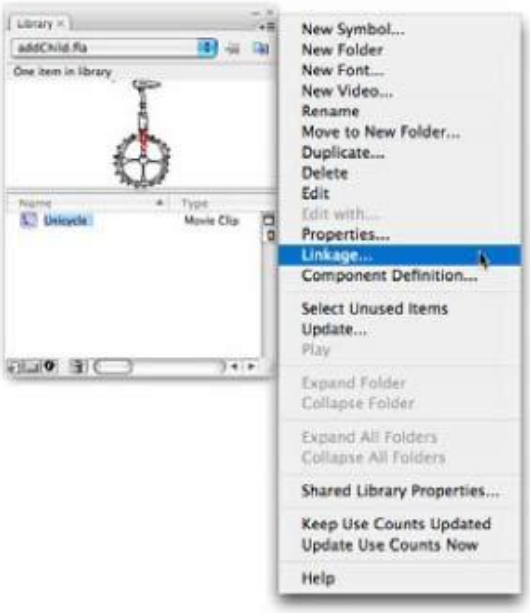
اضافه سازی سمبلهای موجود در کتابخانه به لیست نمایش:

تا حالا هر آبجکتی که به لیست اضافه کردیم آبجکت خالی بوده اما حالا می خوایم آبجکتایی که تو کتابخونه وجود داره رو به لیست اضافه کنیم.

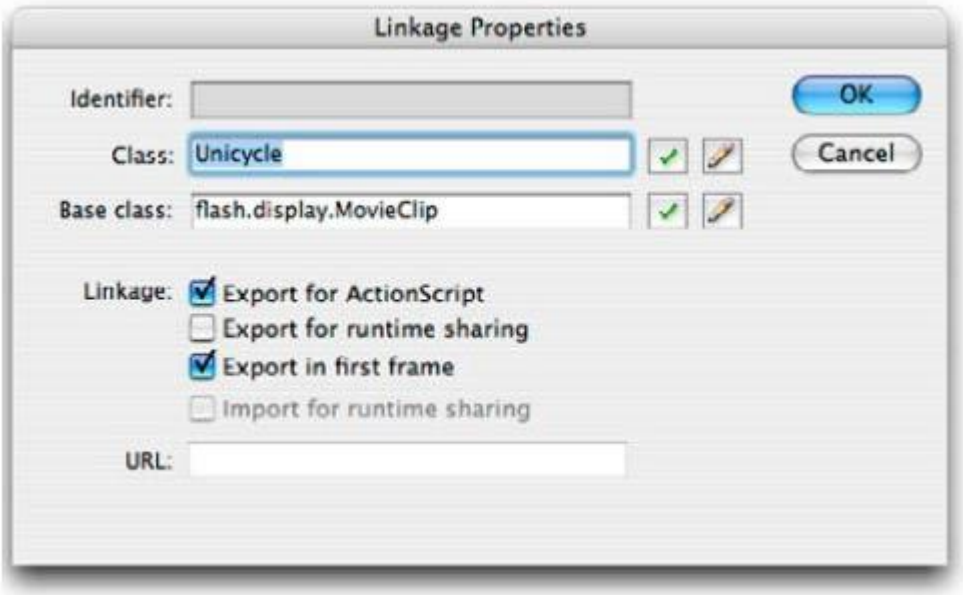
در ابتدا باید یه آبجکت بسازیم و اون رو به کتابخونه اضافه کنیم. ما در این مثال یه مووی کلیپ (یه چرخه) می سازیم.

در مرحله بعدی باید اون آبجکت رو به عنوان یه کلاس معرفی کنیم. مفهوم این جمله رو در فصل بعد (کلاسها) درک خواهید کرد.

برای ساختن کلاس از این آبجکت. ابتدا کتابخونه رو باز میکنیم و روی آبجکت(اینجا تو این مثال یه چرخه) کلیک راست می کنیم و گزینه linkage رو انتخاب می کنیم.



تو پنجره باز شده تیک export for actionScript رو میزنیم و نام کلاسمون رو توی تکست باکس مخصوص می نویسیم.(تو این مثال ما نام Unicycle برای کلاس در نظر میگیریم).



مرحله آخر کار ما ساختن یه نمونه از کلاسه. چون کلاس از جنس مووی کلیپه باید نمونه ای هم که می سازیم(نمونه خالی) از جنس مووی کلیپ باشه.

مرحله آخر هم اضافه سازی آبجکت به لیست نمایشه.

```
var cycle:MovieClip = new Unicycle();
addChild(cycle);
```

نمونه این برنامه در قسمت برنامه های اتچ شده موجوده.

اضافه سازی آبجکت به یک سطح خاص (addChildAt):

متد AddChild آبجکت رو به انتهای لیست نمایش اضافه می کنه ولی مطمئنا مواردی پیش میاد که ما نیاز داشته باشیم تا آبجکت رو به سطح مورد نظرمون اضافه کنیم. اینکار به وسیله متد addChildAt() قابل انجامه.

مثال زیر همه چیز رو روشن می کنه.

```
var inc:uint = 0;

stage.addEventListener(MouseEvent.CLICK, onClick, false, 0, true);

function onClick(evt:MouseEvent):void {
    var ball:MovieClip = new Ball();
    ball.x = ball.y = 100 + inc * 10;
    addChildAt(ball, 0);
    inc++;
}
```

این مثال خیلی ساده با هر کلیک یه توپ رو با نام کلاس Ball به لیست نمایش اضافه می کنه و در هر مرحله توپ رو به ۱۰ پیکسل پایین تر اضافه می کنه.

تو خط اول یه متغیر به نام inc ایجاد کرده و مقدار اوله اون رو صفر می ذاریم.

تو خط بعد یه eventlistener به stage اضافه می کنیم که با هر کلیلک تابع onClick فراخونی بشه.

نکته: true و false آخر eventlistener باعث میشه یه جورایی هر موقع که کار باهاش تموم شه eventlistener خود به خود از حافظه حذف بشه. (توضیحاتش فنیه و از حوصله جمع خارج). فقط اینو بدونین که اگه به انتهای eventlistener سه قسمت ۰ و false و true رو اضافه کنین بهتره. (مربوط میشه به بحث garbage collector که در انتهای درس توضیح داده شده)

تنها نکته ای که تو تابع onClick قابل ذکره همون متد AddChild هست که توپ (Ball) ایجاد شده در این مرحله رو به سطح صفر اضافه می کنه. یعنی همه توپها به سطح صفر displaylist اضافه میشن.

خوب ، اینم از بحث اضافه سازی به سطح خاص. حالا بریم سراغ حذف آبجکت از displaylist.

حذف کردن آبجکت ها از لیست نمایش و آزاد کردن حافظه:

بر قضیه حذف آبجکت ها هم همون قوانین اضافه سازی آبجکت به لیست نمایش حاکمه.

برای حذف یه آبجکت از لیست نمایش فقط نوشتن کد ساده زیر کفایت می کنه

```
removeChild(ball);
```

و برای حذف یه آبجکت از یه سطح خاص به صورت زیر هست

```
removeChildAt(0);
```

خوب ، همون مثال قبلی در مورد اضافه سازی آبجکت به سطح خاص رو اینجا میاریم به یه سری تغییرات.

```
for (var inc:uint = 0; inc < 20; inc++) {
    var ball:MovieClip = new Ball();
    ball.x = ball.y = 100 + inc * 10;
    addChild(ball);
}

stage.addEventListener(MouseEvent.CLICK, onClick, false, 0, true);

function onClick(evt:MouseEvent):void {
    removeChildAt(0);
}
```

تغییراتش رو می گم، خودتون تو کد بررسی کنید.

اولا اینکه به جای کلیک، اضافه سازی آبجکت ها رو به وسیله یه حلقه انجام میدیم. یه حلقه که ۲۰ بار اجرا میشه و ۲۰ تا Ball به لیست نمایش اضافه میکنه.

و بعد از اونا هم یه eventlistener که با هر کلیک تمام فرزندان موجود در سطح صفر رو پاک می کنه.

نکته : اگه این برنامه رو که نمونش هم تو برنامه های اتچ شده هست رو اجرا کرده باشین باید متوجه یه قضیه شده باشین. با کلیک ۲۰ تمام آبجکت ها حذف می شن و با کلیک ۲۱ چون دیگه آبجکتی تو سطح صفر وجود نداره با error ای با مضمون the supplied index is out of bounds مواجه میشین.

برای رفع این خطا کد زیر رو می نویسیم که برای حذف هر آبجکت در تابع onClick شرط وجود فرزند چک بشه.(بوسیله numChildren)

```
function onClick(evt:MouseEvent):void {
    if (numChildren > 0) {
        removeChildAt(0);
    }
}
```

حذف کردن آبجکت ها از حافظه:

اگه یادتون باشه تو بخش eventlistener ها گفتیم که هر موقع که کارمون با eventlistener تموم شد بهتره که اون رو از حافشه حذفش کنیم. خوب دلیلشم معلومه! با زیاد شده eventlistener ها تو حافظه شاید با کمبود حافظه مواجه بشیم و سرعت اجرا برناممون هم پایین بیاد.

در راستای همین موضوع باید همین کار رو برای آبجکتهایی که به لیست اضافه می کنیم هم انجام بدیم. یعنی وقتی که کارمون باهاشون تموم شد اونا رو از حافظه بندازیم بیرون.

نکته: خود ActionScript به تبعیت از زبان جاوا یه موتور به نام garbage Collector داره که هر چند وقت یه بار خود به خود فعال میشه و eventlistener ها و آبجکتهای اضافه و ناکارا رو حذف می کنه. ولی چه بهتر که ما خودمون این کار رو قبل از garbage Collector انجام بدیم!!!

خوب ، چی داشتیم می گفتیم؟ اهان! باید آبجکت هایی رو که کارمون باهاشون تموم شده حذف کنیم.

وسطای همین درس با دستورهای removeChild و removeChildAt آشنا شدیم و گفتیم این دستورا آبجکت رو از لیست نمایش حذف می کنن. اما باید توجه کنید که این دستورا آبجکت رو فقط از لیست نمایش حذف می کنن ولی اون آبجکت رو از حافظه موقت RAM حذف نمی کنن. برای اینکار باید خودمون دست به کار بشیم.

خیلی خیلی راحته فقط کافیه اون آبجکت رو مساوی با null قرار بدیم.

کد زیر رو ببینید تا همه چیز رو متوجه بشین

```
var ball:MovieClip = new Ball();
ball.x = ball.y = 100;
addChild(ball);

stage.addEventListener(MouseEvent.CLICK, onClick, false, 0, true);

function onClick(evt:MouseEvent):void {
    this.removeChild(ball);
    //ball removed from display list but still exists
    trace(ball)
    ball = null;
    //ball now entirely removed
    trace(ball)

    stage.removeEventListener(MouseEvent.CLICK, onClick);
}
```

هیچ نیازی به توضیح احساس نمی کنم. همه چیز روشنه عین روز!!!

دیگه فکر میکنم برای این جلسه کافیه. با این حساب یه جلسه دیگه تا اتمام فصل displaylist باقی می مونه که ایشالله تا یکشنبه اونم آماده می کنم تا هر چه زودتر بریم سراغ فصل مربوط به شی گرایی (opp).

درس نوزدهم:

بسم الله الرحمن الرحیم

مدیریت نام آبجکت ها و ویژگی های آنها:

هرچی که لیست نمایش ما بزرگتر میشه قاعدتا باید بتونیم مدیریت خوبی رو اشیا و آبجکتها داخل اون

داشته باشیم وگرنه بعد از یه مدتی لیست نمایشمون میشه بازار شام

پیرو همین موضوع مباحث زیر مطرح میشن.

پیدا کردن فرزندان بوسیله موقعیت آنها و اسم آنها:

پیدا کردن یه فرزند به وسیله موقعیت اون خیلی سادست. یه چیزی تو مایه های اضافه سازی و حذف فرزند به لیسته. مثال زیر رو ببینید . تو این مثال فرزند سطح صفر از container رو انتخاب میکنه

```
var do:DisplayObject = getChildAt(0);
```

همونطور که می بینید متغیر do از جنس DisplayObject رو تعریف میکنه. و داخل اون آبجکت سطح صفر لیست رو قرار میده.

خوب، ممکنه یکی بگه اگه ما موقعیت یه شی رو ندونیم باید چیکار کنیم؟

بوسیله کد زیر می تونیم بوسیله نام اون آبجکت رو انتخاب کنیم.

```
var do:DisplayObject = getChildByName("circle");
```


jav1366@yahoo.com

تو این مثال ما یه آبجکت به نام circle رو تو آبجکت do قرار میدیم.

حالا مساله رو یه مقدار قشنگتر می کنیم. یه موقعی ممکنه ما بخوایم بوسیله اسم یه آبجکت موقعیت اون در لیست رو بدست بیاریم. کد زیر چاره کار ماست

```
var do:DisplayObject = getChildByName("circle");
var doIndex:int = getChildIndex(do);
```

casting برای شی displayobject:

casting یعنی قالب ریزی کردن. حالا ببینیم چه دخلی به displayobject داره؟

بذارین یه مثال بزنیم تا قضیه روشن تر بشه.

فرض کنین ما یه مووی کلیپ داریم که در سطح صفر وجود داره. حالا از داخل این مووی کلیپ می خوایم به main بگیم برو به فریم ۲۰.

اولین چیزی که به ذهنمون می رسه کد زیره

```
parent.gotoAndStop(20);
```

ولی اینجاست که مشکل پیش میاد. error زیر ماحصل اونه.

Call to a possibly undefined method gotoAndStop through a reference

with static type flash.display:DisplayObjectContainer.

- ✓ مشکل از کجاست؟

مشکل اینه که خود فلش نمی تونه نوع رو تشخیص بده. خوب همونطور که تو نمودار درختی displaylist (جلسه دوم) دیدم parent در اینجا امی تونه stage هم باشه. ولی مگه stage میتونه بره به فریم ۲۰؟ اصلا تابع gotoAndStop مال stage نیست.

اینجاست که فلش گرخیجه میگیره و نمی تونه تصمیم بگیره.

و اینجاست که casting به درد میخوره. بوسیله کد زیر ما به فلش پلیر میگیم که نوع parent ما از نوع MovieClip هست. یعنی منظور مورد نظر ما main timeline هست.

```
MovieClip(parent).gotoAndStop(20);
```

نکته: یادم نیست که parent رو تو درسای قبل گفتم یا نه. parent میگه که یه سطح از سطح مووی کلیپ بر. بالاتر. یعنی اگه الان مووی کلیپ سطح پنجه بر سطح ۴.

مدیریت عمق (Depth Management):

موقعی که به وسیله addChild به یه لیست نمایش آبجکت اضافه می کنیم به صورت اتوماتیک اون آبجکت به انتهای لیست اضافه میشه .

مثال زیر رو ببینین.

```
var mc1:MovieClip = new MovieClip();
mc1.name = "clip1";
addChild(mc1);
var mc2:MovieClip = new MovieClip();
mc2.name = "clip2";
addChild(mc2);
trace(getChildAt(0).name);
trace(getChildAt(1).name);
```

در ابتدا mc1 رو به لیست اضافه می کنیم و بعدش mc2 رو موقعی که trace می گیریم، mc1 در سطح صفر و mc2 در سطح یک قرار داره.

حالا دومرتبه mc1 رو add می کنیم و trace می گیریم. می بینید که سطوح عوض میشه.

```
addChild(mc1);
trace(getChildAt(0).name);
trace(getChildAt(1).name);
```

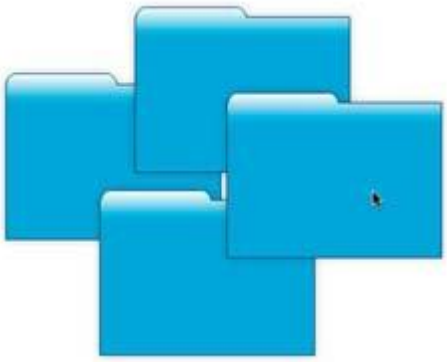
حالا اگه بخوایم جای دوتا آبجکت موجود تو لیست نمایشمون رو با هم عوض کنیم(عمقشون رو) قطعه کد زیر کفایت می کنه.

```
swapChildren(mc1, mc2);
```

عوض کردن عمق دو تا آبجکت با توجه به سطحشون هم به این صورته.

```
swapChildrenAt(0, 1);
```

یه روش دیگه هم برای جابجایی عمق آبجکتها وجود داره که از دوتای قبلی خیلی کاربردی تره. پس تو یه مثال این روش رو شرح می دیم.



چهار تا فولدر داریم که می خوایم موس رو هرکدومشون رفت بیاد روی همه قرار بگیره. این کار قاعدتا باید با مدیریت عمق صورت بگیره.

فایلش رو پیوست کردم. کدش رو هم اینجا می ذارم.

```
this.addEventListener(MouseEvent.MOUSE_OVER, onBringToTop, false, 0, true);

function onBringToTop(evt:MouseEvent):void {
    var folder:MovieClip = evt.target as MovieClip;
    setChildIndex(folder, numChildren - 1);

}
```

تو این مثال folder رو از جنس مووی کلیپ تعریف می کنیم و داخلش evt.target رو که همون فولدریه که روش کلیلک کردیم رو می ریزیم

- ✓ سوال: چرا جلوی evt.target صراحتا گفتیم که نوع اون مووی کلیپه؟
- ← جواب: همون عمل casting هست دیگه. دلیلشم که تو بخش casting گفتم.

تو خط آخر هم بوسیله setChildIndex میگیم که فولدری رو که روش کلیک می کنیم رو بیار روی همه فولدرا.

شیوه کارشم اینه که ، تعداد فرزندان رو بدست میاریم(در اینجا ۴ تا).بالاترین سطح ، یکی کمتر از تعداد فرزندانه(اینجا سطح ۳ بالاترینه). چون سطوح از صفر شروع میشن(مثل آرایه ها) . یعنی ۴ فرزند ما به ترتیب در سطوح ۰و۱و۲و ۳ خواهند بود.

بنابراین با کد خط آخر ما فولدر کلیک شده(folder) رو به بالاترین سطح(اینجا سطح۳) می فرستیم.

نکته: این بحث آخر (مدیریت عمق) خیلی مهمه! در ادمه که با اکشن بیشتر کار کنین یکی از مباحثیه که خیلی باهاش درگیر خواهید بود.

ایشالله فصل بعد رو با موضوع کار با تایم لاین در خدمتتون خواهم بود.