FLASH5 ACTIONSCRIPT SÖZLÜĞÜ

A

- add
- _alpha
- and
- Array (object)
- Array.concat
- Array.join
- Array.length
- Array.pop
- Array.push
- Array.reverse
- Array.shift
- Array.slice
- Array.sort
- Array.splice
- Array.toString
- Array.unshift

B

- Boolean (function)
- Boolean (object)
- Boolean.toString
- Boolean.valueOf
- break

C

- call
- chr
- Color (object)
- Color.getRGB
- Color.getTransform
- Color.setRGB
- Color.setTransform
- continue
- _currentframe

D

- Date (object)
- Date.getDate
- Date.getDay
- Date.getFullYear
- Date.getHours
- Date.getMilliseconds
- Date.getMinutes
- Date.getMonth
- Date.getSeconds
- Date.getTime
- Date.getTimezoneOffset
- Date.getUTCDate
- Date.getUTCDay
- Date.getUTCFullYear
- Date.getUTCHours
- Date.getUTCMilliseconds
- Date.getUTCMinutes
- Date.getUTCMonth
- Date.getUTCSeconds
- Date.getYear
- Date.setDate
- Date.setFullYear
- Date.setHours
- Date.setMilliseconds
- Date.setMinutes
- Date.setMonth
- Date.setSeconds
- Date.setTime
- Date.setUTCDate
- Date.setUTCFullYear
- Date.setUTCHours
- Date.setUTCMilliseconds
- Date.setUTCMinutes
- Date.setUTCMonth
- Date.setUTCSeconds
- Date.setYear
- Date.toString
- Date.UTC
- delete
- do... while
- _droptarget
- duplicateMovieClip

E

- else
- eq (equal—string specific)
- escape
- eval
- evaluate

F

- _focusrect
- for
- for..in
- _framesloaded
- fscommand
- function

G

- ge (greater than or equal to)
- getProperty
- getTimer
- getURL
- getVersion
- gotoAndPlay
- gotoAndStop
- gt (greater than —string specific)

H

- _height
- _highquality

I

- if
- ifFrameLoaded
- #include
- Infinity
- int
- isFinite
- isNaN

K

- Key (object)
- Key.BACKSPACE
- Key.CAPSLOCK
- Key.CONTROL
- Key.DELETEKEY
- Key.DOWN
- Key.END
- Key.ENTER
- Key.ESCAPE
- Key.getAscii
- Key.getCode
- Key.HOME
- Key.INSERT
- Key.isDown
- Key.isToggled
- Key.LEFT
- Key.PGDN
- Key.PGUP
- Key.RIGHT
- Key.SHIFT
- Key.SPACE
- Key.TAB
- Key.UP

L

- le (less than or equal to — string specific)
- length
- _level
- loadMovie
- loadVariables
- lt (less than — string specific)

M

- Math (object)
- Math.abs
- Math.acos
- Math.asin
- Math.atan
- Math.atan2
- Math.ceil
- Math.cos
- Math.E
- Math.exp
- Math.floor
- Math.log
- Math.LOG2E
- Math.LOG10E
- Math.LN2
- Math.LN10
- Math.max
- Math.min
- Math.PI
- Math.pow
- Math.random
- Math.round
- Math.sin
- Math.sqrt
- Math.SQRT1_2
- Math.SQRT2
- Math.tan
- maxscroll
- mbchr
- mblength
- mbord
- mbsubstring
- Mouse (object)
- Mouse.hide
- Mouse.show
- MovieClip (object)
- MovieClip.attachMovie
- MovieClip.duplicateMovieClip
- MovieClip.getBounds
- MovieClip.getBytesLoaded
- MovieClip.getBytesTotal
- MovieClip.getURL
- MovieClip.globalToLocal
- MovieClip.gotoAndPlay
- MovieClip.gotoAndStop
- MovieClip.hitTest
- MovieClip.loadMovie
- MovieClip.loadVariables
- MovieClip.localToGlobal
- MovieClip.nextFrame
- MovieClip.play
- MovieClip.prevFrame
- MovieClip.removeMovieClip
- MovieClip.startDrag
- MovieClip.stop
- MovieClip.stopDrag
- MovieClip.swapDepths
- MovieClip.unloadMovie

N

- _name
- NaN
- ne (not equal — string specific)
- new
- newline
- nextFrame
- nextScene
- not
- null
- Number (function)
- Number (object)
- Number.MAX_VALUE
- Number.MIN_VALUE
- Number.NaN
- Number.NEGATIVE_INFINITY
- Number.POSITIVE_INFINITY
- Number.toString
- Number.valueOf

O

- Object (object)
- Object.toString
- Object.valueOf
- onClipEvent
- on(mouseEvent)
- or
- ord

P

- _parent
- parseFloat
- parseInt
- play
- prevFrame
- prevScene
- print
- printAsBitmap

Q

- _quality

R

- random
- removeMovieClip
- return
- _root
- _rotation

S

- scroll
- Selection.getBeginIndex
- Selection.getCaretIndex
- Selection.getEndIndex
- Selection.getFocus
- Selection.setFocus
- Selection.setSelection
- set
- setProperty
- Sound (object)
- Sound.attachSound
- Sound.getPan
- Sound.getTransform
- Sound.getVolume
- Sound.setPan
- Sound.setTransform
- Sound.setVolume
- Sound.start
- Sound.stop
- _soundbuftime
- startDrag
- stop
- stopAllSounds
- stopDrag
- String (function)
- String (object)
- String.charAt
- String.charCodeAt
- String.concat
- String.fromCharCode
- String.indexOf
- String.lastIndexOf
- String.length
- String.slice
- String.split
- String.substr
- String.substring
- String.toLowerCase
- String.toUpperCase
- substring

T

- _target
- targetPath
- tellTarget
- this
- toggleHighQuality
- _totalframes
- trace
- typeof

U

- unescape
- unloadMovie
- updateAfterEvent
- _url

V

- var
- _visible
- void

W

- while
- _width
- with

X

- _x
- _xmouse
- _xscale

Y

- _y
- _ymouse
- _yscale

ACTIONSCRIPT SÖZLÜĞÜ

add : İki veya daha fazla string'i ard arda yazdırır. Bu ifade yerine & veya + komutları da kullanılabilir. Flash 4'de kullandığınız & ifadesi, otomatik olarak add ifadesine dönüştürülür. Yazılımı;

```
String1 add String2
```

Örnek :

```
Mesaj = "Merhaba" add "Dünya";

// MerhabaDünya
```

Player : Flash 4 ve sonrası.

_alpha : Bir movie clip'e alpha özelliği, yani görünürlük değeri vermek için kullanılır. Bu değer 0 olursa, nesne tam olarak görünmez, 100 olursa, tam olarak görünür olur. Herhangi bir movie clip'in, alpha değerinin 0 olması demek kullanılmaması demek değildir. Alpha değeri 0 olan bir movie clip içindeki düğme, hala çalışır durumdadır. Yazılımı;

```
movieClip._alpha = değer
```

Örnek : Aşağıdaki örnekte, düğmeye basıldığı zaman, masa adlı movie clip'in görünürlüğü % 50 olacaktır.

```
on(press){
      setProperty("masa",_alpha, 50;
}
```

veya

```
on(press){
      masa._alpha = 50;
}
```

Player : Flash 4 ve sonrası

and : Flash 4 playerda, mantıksal sınama işlemcisidir. Eğer iki ifade de doğru (true) ise, sonuç doğru olur. Herhangi bir ifade false olursa, sonuç da yanlış (false) olur. Yazılımı;

```
kosul1 and kosul2
```

Örnek :

```
if(x = 10 and y < 100){
      gotoAndPlay(25);
}
```

Player : Flash 4 ve sonrası. Bu işlemci, Flash 5'te, yerini && ifadesine bırakmıştır.

Array (object) : Birden fazla değişken tanımlamak için kullanılan dizi değişkenlerdir. Array ile tutulan değişkenler, sıralı bir şekilde numaralandırılırlar. Dizi değişken tanımlandığında, ilk elementin numarası 0'dır. İkinci element 1 olur. Mesela, aşağıda günler dizi değişken içinde tanımlanmıştır.

```
gunler[0] = "Pazartesi"
gunler[1] = "Sali"
gunler[2] = "Carsamba"
gunler[3] = "Persembe"
gunler[4] = "Cuma"
gunler[5] = "Cumartesi"
gunler[6] = "Pazar"
```

Bir dizi değişken tanımlamak için, new array komutu kullanılır. Dizi değişkenlerin numaraları köşeli parantez ([ ]) içine alınır. Yazılımı;

```
new Array();
new Array(boyut);
new Array(element1, element2...elementN);
```

Burada kullanılan boyut ifadesi, dizi değişken elementlerinin sayısıdır. Elementler ise, dizi içindeki her bir öğedir. Eğer boyut değeri olarak bir şey yazılmamış ise, hiç bir öğesi olmayan, değişken dizisi oluşturulmuş olur.

Örnek :

```
bizimTakim = new Array();
bizimTakim = new Array("Osman", "Mustafa", "Hakkı");
```

buna göre, değişken dizisi şöyledir;

```
bizimTakım[0] = "Osman";
bizimTakim[1] = "Mustafa";
bizimTakim[2] = "Hakı";
```

Player : Flash 5 ve sonrası

Array.concat : İki veya daha fazla sayıdaki dizi değişkenlerinin, ard arda yazılmasıyla oluşturulan yeni bir dizi değişken metodudur. Dizi değişkenlerinin tuttuğu veri tipleri önemli değildir. String ve sayısal veri tiplerini bir arada toplayabilir. Yazılımı,

```
isim.contac(isim2, isim3,...isimN)
```

Örnek :

```
harfler = new Array("a", "b", "c");
rakamlar = new Array(1, 2, 3);

yeniDizi = harfler.concat(rakamlar);

buna göre, yeniDizi şöyle oluşur.

yeniDizi = ["a", "b", "c", 1, 2, 3]
```

Oluşturulan `yeniDizi`'deki eleman sırası, "`.concat`" ifadesinden öncekiler ve "`.concat`" ifadesinden sonra, parantez içinde yazılanlar diye gider. Başka bir örnek,

```
sayilar1 = [1, 3, 5];
sayilar2 = [2, 4, 6];
sayilar3 = [7, 8, 9];

toplam = sayilar1.concat(sayilar2, sayilar3);
```

Bu örnekteki toplam dizi değişkeni listesi şöyle oluşur,

```
toplam = [1, 3, 5, 2, 4, 6, 7, 8, 9]
```

Player : Flash 5 ve sonrası

Array.join : Dizi değişken içindeki tüm elemanları, aralarına belirlediğiniz bir karakter koyarak, birleştirip tek bir string yapma metodudur. Eğer, karakter belirlemezseniz, virgül işareti varsayılan değer olarak, otomatik koyulur. Yazılımı,

```
isim.join();
isim.join(karakter);
```

Örnek :

```
meyvelar = new Array("elma", "armut", "muz");
sepet_1 = meyveler.join();

      // sepet_1 = "elma,armut,muz" olur.

sepet_2 =meyveler.join(, );

      // sepet_2 = "elma, armut, muz" olur.

sepet_3 = meyveler.join( / );

      //sepet_3 = "elma / armut / muz" olur.
```

Örneklerde görüldüğü gibi, ".join" ifadesinden sonra gelen, parantez işareti içine ne yazarsanız, dizi değişken içindeki elemanlar arasına, o karakterleri koyarak, birleştirir ve tek bir string yapar.

Player : Flash 5 ve sonrası

Array.length : Dizi değişken içindeki elemanların sayısını verir. Herhangi bir zamanda eleman sayısı değişirse, `array.length` de ototmatik olarak değişmiş olur. Dizi değişkenlerinin eleman sayısını, direkt olarak array(boyut) şeklinde de belirtebilirdik. Burada kullanılan boyut değişkeni, herhangi bir sebeple değişirse, `array.length` de değişmiş olur. Yazılımı,

```
isimarray.length;
```

Örnek :

```
dizi = new Array();

      // dizi.length = 0 olacaktır.

dizi[1] = "serkan";

      // dizi.length = 2 olarak değişir.
```

Player : Flash 5 ve sonrası

Array.pop : Bir dizi değişken içindeki elemanların, en sonuncusunu çıkararak bulur. Yazılımı,

```
isim = isimarray.pop();
```

Örnek :

```
urunler = new Array("monitor", "kablo", "klavye", "kasa");
sonuncu = urunler.pop();

      // sonuncu = "kasa" olacaktır.
      // urunler = ["monitor", "kablo", "klavye"] olarak kalır.
```

Player : Flash 5 ve sonrası.

Array.push : Bir dizi değişken içindeki elemanlara, bir veya daha fazla sayıda eleman eklemek için kullanılan metoddur. Yazılımı,

```
isim = isimarray.push(eleman1, eleman2...elemanN);
```

Örnek :

```
hobiler = new Array("internet", "muzik");
yenihobiler = hobiler.push("kitap", "sinema");
```

Örnekte, `hobiler` adlı değişken ilk önce 2 elemanı varken, ikinci satırda, 2 yeni eleman daha eklenmiş oldu. İkinci satırda kullandığımız `yenihobiler` adlı değişken, `hobiler` değişkeninin eleman sayısını verir. Yani,

```
hobiler= ["internet", muzik", "kitap", "sinema"];
yenihobiler = 4
```

olacaktır.

Player : Flash 5 ve sonrası.

Array.reverse : Dizi değişken içindeki elemanların yerini, ters çeviren metoddur. Yazılımı,

```
isim = isimarray.reverse();
```

Örnek :

```
sayilar = new Array(1, 2, 3, 4, 5);
tersi = sayilar.reverse();

      // tersi = [5, 4, 3, 2, 1] olacaktır.
```

Player : Flash 5 ve sonrası.

Array.shift : Dizi değişken elemanları içindeki, ilk elemanı çıkartarak, alan metoddur. `Pop` metodunun tam tersidir. Yazılımı,

```
isim = isimarray.shift();
```

Örnek :

```
iller = new Array("adana", "urfa", "mersin", "sinop");
birinci_il = iller.shift();

      // birinci_il = "adana" olacaktır.
      // iller = ["urfa", "mersin", "sinop"] olarak kalır.
```

Player : Flash 5 ve sonrası.

Array.slice : Dizi değişken elemanları içinde belirttiğiniz iki sayı arasında olanları alan metoddur. Belirttiğiniz iki sayı parantez içinde yazılır. Başlangıç sayısı olarak 0 verirseniz, ilk eleman dahil, bitiş olarak belirttiğiniz rakama kadar olan elemanları alır. Eğer, başlangıç olarak negatif bir değer girilirse (-1), o zaman dizi değişkenin içini boşaltır. Yani negatif değer, en son eleman olarak tanımlanmış olur. Yazılımı,

```
isim = isimarray.slice(başlangıç, bitiş);
```

Örnek :

```
plakalar = new Array(05, 12, 45, 27, 63, 71, 35);
kesim1 = plakalar.slice(0,-1);

      // kesim1 = [05, 12, 45, 27, 63, 71] olur.

kesim2 = plakalar.slice(-1,5);

      // kesim2= "", yani hiç bir elemanı olmaz.

kesim3 = plakalar.slice(2,5);

      // kesim3 = [45, 27, 63] olacaktır.
```

Player : Flash 5 ve sonrası.

Array.sort : Dizi değişken içindeki elemanları, harf veya küçükten büyüğe göre dizen metoddur. Liste içindeki elemanlar, string ise, harf sırasına göre, sayı ise, küçükten büyüğe doğru dizer. Eğer bu dizilişi, ters şeklinde yapmak isterseniz, dizme işleminden sonra, array.reverse komutunu kullanarak yapabilirsiniz. Diziliş sırasını kendinize göre belirlemek isterseniz, sıralamayı belirleyen özel bir fonksiyon hazırlamalısınız. Hazırladığınız bu fonksiyonu, `sort` ifadesinden sonraki parantez içine yazın. Eğer parantez içine bir şey yazılmamışsa, harf sırası ve küçükten-büyüğe, varsayılan değer olarak işlem yapılır. Yazılışı,

```
isim = isimarray.sort();
isim = isimarray.sort(sıralama fonksiyonu);
```

Örnek :

```
uyeler = new Array("ozger", "ersoy", "genco", "erhan", "serkan");
sirala1 = uyeler.sort();

// sirala1 = ["erhan", "ersoy", "genco", "ozger", "serkan"]
// şeklinde olacaktır.

// Sayısal sıralama,

kodNumaralari = new Array(23, 85, 62, 38, 110, 8);
```

```
sirala2 = kodNumaralari.sort();
sirala3 = sirala2.reverse();

// sirala2 = [8, 23, 38, 62, 85, 110] olur.
// sirala3 = [110, 85, 62, 38, 23, 8] olur.
```

Player : Flash 5 ve sonrası.

Array.splice : Dizi değişken içinden, belirlenmiş bir sayıda ve belirli bir yerden başlayarak, eleman çıkartma ve istenirse, çıkarılan yerden itibaren, araya eleman koyma metodudur. Yazılımı,

```
isim = isimarray.splice(başlangıç, silinecek eleman sayısı,
       eklenecek eleman1, eleman2...elemanN);
```

Örnek :

```
renkler = ["sari", "mavi", "yesil", "siyah", "bej"];
cikanrenkler = renkler.splice(2,2);

// cikanrenkler = ["yesil", "siyah"] olur.
// renkler = ["sari", "siyah", "bej"] olur.
// Yani 2. elemandan itibaren 2 eleman çıkart demek.

renkler = ["sari", "mavi", "yesil", "siyah", "bej"];
cikanrenkler = renkler.splice(1,3, "turuncu", "beyaz");

// cikanrenkler = ["mavi", "yesil", "siyah"] olur.
// renkler = ["sari", "turuncu", "beyaz", "bej"] olur.
// Bu ise, 1. elemandan itibaren, 3 eleman çıkart ve
// araya, belirlediğim elemanları koy demektir.
```

Player : Flash 5 ve sonrası.

Array.toString : Bir değişken dizisi içindeki tüm elemanları ard arda toplayıp, aralarına virgül koyarak, tek bir string haline getirir. Yazılımı,

```
isim = isimarray.toString();
```

Örnek :

```
kullaniciAdlari = ["zoque", "frenky", "malina"]
hepsi = kullaniciAdlari.toString();
// hepsi = "zoque,frenky,malina" olacaktır.
```

Player : Flash 5 ve sonrası.

Array.unshift : Belirlediğiniz sayıları, değişkenleri veya stringleri, bir dizi değişken listesinin en başından itibaren, içine yerleştiren metoddur. Yazılımı,

```
isim = isimarray.unshift(elemnt1, element2...elementN)
```

Örnek :

```
defter = [2, 35, "burak", "kare"];
ekle = defter.unshift("elips", "bordo", 87);
```

```
// ekle = 7 olacaktır.
// defter = ["elips", "bordo", 87, 2, 35, "burak", "kare"]
// olacaktır.
```

Örnekte, `ekle` adlı değişken, en son eklenenlerle birlikte, `defter` adlı değişkenin eleman sayısını verir.

Player : Flash 5 ve sonrası.

Boolean(function) : Bir ifadeyi Boolean fonksiyonuna çevirir. Yani, ifade ya true (doğru), yada false (yanlış) olur. Eğer ifade, sayısal bir değer içeriyorsa, true, alfanümerik bir değer içeriyorsa, false olur. Yazılımı,

```
isim = Boolean(ifade);
```

Örnek :

```
bugun = "Pazartesi";
bu_ay = 12;
gun = Boolean(bugun);
ay = Boolean(ay);

// gun = false olacaktır.
// ay = true olacaktır.
```

Player : Flash 5 ve sonrası.

Boolean(object) : Boolean nesnesinin bir durumunu oluşturmak için kullanılır. Yazılımı,

```
new Boolean();
new Boolean(ifade);
```

Örnek :

```
new Boolean();
new Boolean(yaziKutusu);
```

Eğer oluşturduğunuz Boolean nesnesi, bir ifade içermiyorsa, değeri false olur. Herhangi bir ifade içeriyorsa, aşağıdaki kurallara göre değer alır.

- İfade bir sayı ise; Sayı 0 ise, fonksiyon false, diğer sayılarda, true olur.
- İfade bir Boolean ise; Fonksiyon değeri aynen kalır.
- İfade bir nesne veya movie clip ise; İfade değeri boş (null) ise false, değilse true olur.
- İfade bir string ise; String değeri boş ise false, değilse true olur.

Player : Flash 5 ve sonrası.

Boolean.toString : Boolean nesnesine çevirdiğiniz bir ifadeyi, string yapar. Yazılımı,

```
isim.toString();
```

Örnek :

```
yaziKutusuDenetle = yaziKutusu.toString();
```

Player : Flash 5 ve sonrası.

Boolean.valueOf : Basit tip verileri, Boolean nesnelerine, Boolean nesnelerini de basit veri tiplerine çevirir. Yazılımı,

```
isim.valueOf();
```

Örnek :

```
gorunurluk = _root.oyun._alpha;
dogruMu = gorunurluk.valueOf();
```

Player : Flash 5 ve sonrası.

break : Bir döngü içinde kullanılır (`for, for....in, do...while, while`). Program bu komutu gördüğü zaman, döngüyü durdurur ve döngüden sonra gelen ilk satırdan itibaren, programı yürütmeye devam eder. Yazılımı,

```
break;
```

Örnek :

```
i = 0;
while (true) {
      if (i >= 100) {
            break;
      }
      i++;
}
// i 100'den büyük veya eşit olduğunda döngü duracak,
// ve programa döngü kısmından sonra devam edecektir.
```

Player : Flash 4 ve sonrası.

call : ActionScript çalışırken, arada başka bir frame'e ait değerleri kullanmak için çağırma metodudur. Frame'i çağırırken, ya ismi ile (label), ya da numarası ile çağırılır. Yazılımı,

```
call(frame);
```

Örnek :

```
on (press){
      call(mesaj);
}
// düğmeye basıldığı zaman, mesaj adlı frame gidip gelecek.
```

Player : Flash 4 ve sonrası. Bu komut, Flash 5'te fazla önemsenmemiş. Bunun yerine, `function` komutu, kullanılması tavsiye ediliyor.

chr : Sayıyı, ASCII kodlarına dönüştüren bir komuttur. Veri, string olarak yazılır. Yazılımı,

```
chr(sayı);
```

Örnek :

```
karakter = chr(75);
// karakter = "K" olacaktır.
```

Player : Flash 4 ve sonrası. Bu komut, Flash 5'te önemsenmemiştir. Bunun yerine, `string.fromCharCode` komutu, kullanılması tavsiye ediliyor.

Color(object) : Adını belirttiğiniz bir movie clip'in renk özelliklerini ayarlamak için kullanılır. `new Color` komutu ile oluşturulan nesne, `setRGB` ve `setTransform` komutları ile renk ayarları yapılır. Bu ayarlama ile ilgili bilgileri de `getRGB` ve `getTransform` komutları ile alırız. Yazılımı,

```
isim = new Color(movie clip adı);
```

Örnek :

```
// spot adlı bir movie clip'imiz var.

renkSpot = new Color(spot);

// artık renkSpot nesnesi ile renk ayarlamasını yapabiliriz.
```

Player : Flash 5 ve sonrası.

Color.getRGB : Script içinde, `setRGB` ile renk ayarlaması yapılmış bir movie clip'in, nümerik değer olarak renk bilgisini verir. Yazılımı,

```
renkNesnesi.getRGB();
```

Örnek :

```
// setRGB ile renk bilgisi tanımlanmış kabul ediyoruz.

renkBilgi_1 = spot.getRGB();
renkBilgi_2 = (spot.getRGB()).toString(16);

// renkBilgi_1 ile nümerik değer elde edilir.
// renkBilgi_2 ile hexadecimal cinsten bilgi alınır.
```

Player : Flash 5 ve sonrası.

Color.getTransform : Script içinde, `setTransform` ile renk ve görünürlük ayarı yapılmış bir movie clip'in, hexadecimal cinsinden, bilgisini verir. Yazılımı,

```
renkNesnesi.getTransform(movie clip adı);
```

Örnek :

```
// setTransform ile renk bilgisi tanımlanmış kabul ediyoruz.

renkBilgi = spot.getTransform();
```

Player : Flash 5 ve sonrası.

Color.setRGB : Movie clip için oluşturulmuş renk nesnesinin, RGB (Red, Green, Blue, 'kırmızı, yeşil, mavi') veya hexadecimal (0xRRGGBB) olarak düzenlenmesini sağlar. Yazılımı,

```
isim = renkNesnesi.setRGB(0xRRGGBB);
```

Örnek :

```
renkNesnesi = new Color(spot);
bilgi = renkNesnesi.setRGB(0x993366);
```

Player : Flash 5 ve sonrası.

Color.setTransform : Bir movie clip için oluşturulmuş renk nesnesinin, renk ve görünürlük ayarlarını belirler. Bu renk nesnesine, bu bilgileri atıfta bulunabilmek için, özel parametreler belirtilmesi ve bu ayarlamayı yapabileceğimiz, jenerik bir nesne oluşturmamız lazımdır. Bu parametreler,

- ra ; Red, yani kırmızı bileşenini, yüzde cinsinden ayarlar. (-100, 100)
- rb ; Kırmızı bileşeninin, ofset değerini ayarlar. (-255, 255)
- ga ; Green, yani, yeşil bileşenini, yüzde cinsinden ayarlar. (-100, 100)
- gb ; Yeşil bileşeninin, ofset değerini ayarlar. (-255, 255)
- ba ; Blue, yani, mavi bileşenini, yüzde cinsinden ayarlar. (-100, 100)
- bb ; Mavi bileşeninin, ofset değerini ayarlar. (-255, 255)
- aa ; Alpha, yani, görünürlük bileşenini, yüzde cinsinden ayarlar. (-100, 100)
- ab ; Görünürlük bileşeninin, ofset değerini ayarlar. (-255, 255)

Bu değerleri, tek tek kullanabileceğiniz gibi, tümünü tek bir satırda da belirtebilirsiniz. Yazılımı,

```
// tek tek yazılım.

renkTransform = new Object();
renkTransform.ra = 45;
renkTransform.rb = 122;
renkTransform.ga = 33;
renkTransform.gb = -12;
renkTransform.ba = 82;
renkTransform.bb = 91;
renkTransform.aa = 54;
renkTransform.ab = -70;

// bir satırdaki yazılım.

renkTransform = { ra: ‘45’, rb: ‘122’, ga: ‘33’, gb: ‘-12’,
                                       ba: ‘82’, bb: ‘91, aa: ‘54’, ab: ‘-70’}
```

Örnek :

```
// spot adlı movie clip için renk nesnesi oluşturuluyor.

renkNesne = new Color(spot);

// color transform nesnesi çağırıldığı zaman, buna cevap
// verebilecek, jenerik nesne oluşturuluyor.

renkTransform = new Object;

// renkTransform ayarları belirleniyor.

renkTransform = { ra: '50', rb: '244', ga: '40', gb: '112', ba: '12', bb:
'90', aa: '40', ab: '70'}

// ve nihayet renk nesnesi ile jenerik nesne kullanılarak
// değerler, movie clip’e gönderiliyor.

renkNesne.setTransform(renkTransform);
```

Player : Flash 5 ve sonrası

continue : Bütün döngü ifadelerinde kullanılabilir. Genel olarak manası, devam et demektir. `while` döngüsü içinde herhangi bir şartın durumuna göre, döngünün başına dönerek devam eder. `do...while` döngüsü içinde şartın durumuna göre, döngünün sonuna giderek, döngüye devam eder. `for` döngüsü içinde kullanılan `continue`, döngünün devam etmesini sağlar. `for...in` döngüsü içinde kullanılan `continue`, döngünün başına döner ve bir sonraki değere göre işleme devam eder. Yazılımı,

```
continue;
```

Örnek :

```
skor = metinKutusuSkor;
for (skor=0; skor>50; skor++){
      if (skor>30){
            gotoAndPlay(15);
      } else{
            continue;
      }
}
```

Player : Flash 4 ve sonrası

_currentframe : Zaman ekseni üzerindeki oynatma başlığının, o andaki frame numarasını verir. Bu özelliğe herhangi bir atama yapılamaz. Sadece frame numarası alınabilir. Yazılımı,

```
movieclipadı._currentframe;
```

Örnek :

```
gotoAndStop(kutu._currentframe +1);
```

Player : Flash 4 ve sonrası

Date (object) : Bir tarih nesnesi oluşturarak, bu nesneye tarih ve zaman bilgileri atama yapılabilir. Oluşturulan bu nesne yardımı ile çeşitli tarih bilgileri elde edilir. Tarih nesnesinin çeşitli metodlarından faydalanabilmek için, ilk önce bu tarih nesnesi oluşturulur. Tarih nesnesi oluştururken, argüman yazılmadığı takdirde, nesne bilgisi sistem tarih bilgisini alır. Argümanlar yazılımı aşağıda belirtildiği sırada verilerek yazılmalıdır. Ancak argüman verilirken sadece yıl bilgisi zorunludur. Diğerleri opsiyoneldir.

```
tarihNesnesi = new Date();
```

veya,

```
new Date(yıl,[ay,[gün,[saat,[dakika,[saniye,[milisaniye]]]]]]);
```

Örnek :

```
dogumGunu = new Date (1970, 1, 5);
// buna göre dogumGunu, 5 Ocak 1970 olacaktır.
```

Player : Flash 5 ve sonrası

Date.getDate : Daha önce oluşturulmuş olan tarih nesnesinde, ayın gün bilgisini verir (1 ile 31 arasında rakamlar). Yazılımı,

```
tarihNesnesi.getDate();
```

Örnek :

```
tarihNesnesi = new Date();
gun = tarihNesnesi.getDate();
// gun değişkeni o günün değerini verir (12,24,30 gibi).
```

Player : Flash 5 ve sonrası

Date.getDay : Önceden oluşturulmuş olan tarih nesnesinin, haftanın kaçıncı günü olduğunu verir. Verilen değer, 0 ile 6 arasında olacaktır. Buna göre 0 – Sunday (Pazar), 6 – Saturday (Cumartesi) olacaktır. Yazılımı,

```
tarihNesnesi.getDate();
```

Örnek :

```
tarihNesnesi = new Date();
gun = tarihNesnesi.getDay();
// gun değişkeni haftanın gün değerini verir (1,4,5 gibi).
```

Player : Flash 5 ve sonrası

Date.getFullYear : Önceden oluşturulan tarih nesnesinin, yıl değerini dört basamaklı sayı olarak verir. Tarih nesnesine herhangi bir argüman verilmemişse, sistem bilgisinden aldığı tarih bilgisine göre çalışır. Yazılımı,

```
tarihNesnesi.getFullYear();
```

Örnek :

```
tarihNesnesi = new Date();
yil = tarihNesnesi.getFullYear();
// yil değişkeni sistemin yıl değerini verir (2001 gibi).
```

Player : Flash 5 ve sonrası

Date.getHours : Önceden oluşturulmuş tarih nesnesinin saat bilgisini verir. Verilen değer 0 ile 23 arasında tamsayı bir değerdidir. Tarih nesnesine herhangi bir argüman verilmemişse, sistem bilgisinden aldığı değeri verir. Yazılımı,

```
tarihNesnesi.getHours();
```

Örnek :

```
tarihNesnesi = new Date();
saat = tarihNesnesi.getHours();
// saat değişkeni sistemin saat değerini verir (12, 20 gibi).
```

Player : Flash 5 ve sonrası

Date.getMilliseconds : Önceden oluşturulmuş olan tarih nesnesinin, milisaniye değerini verir. Verilen değer 0 ile 999 arasında tamsayı bir değerdir. Tarih nesnesine herhangi bir argüman verilmemişse, sistem bilgisinden aldığı değeri verir. Yazılımı,

```
tarihNesnesi.getMilliseconds();
```

Örnek :

```
tarihNesnesi = new Date();
miliSaniye = tarihNesnesi.getMilliseconds();
// miliSaniye değişkeni sistemin milsaniye değerini verir.
// (12, 200, 755 gibi).
```

Player : Flash 5 ve sonrası

Date.getMinutes : Önceden oluşturulan tarih nesnesinin dakika değerini verir. Verilen değer 0 ile 59 arasında tamsayı bir değerdir. Tarih nesnesine herhangi bir argüman verilmemişse, sistem bilgisinden aldığı değeri verir. Yazılımı,

```
tarihNesnesi.getMinutes();
```

Örnek :

```
tarihNesnesi = new Date();
dakika = tarihNesnesi.getMinutes();
// dakika değişkeni sistemin dakika değerini verir.
// (2, 20, 55 gibi).
```

Player : Flash 5 ve sonrası

Date.getMonth : Önceden oluşturulmuş olan tarih nesnesinin, ay değerini verir. Verilen değer 0 ile 11 arasında tamsayı bir değerdir. Buna göre, 0 – January (Ocak), 11 – December (Aralık) olacaktır. Tarih nesnesine herhangi bir değer verilmemişse, sistem bilgisinden aldığı değeri verecektir. Yazılımı,

```
tarihNesnesi.getMonth();
```

Örnek :

```
tarihNesnesi = new Date();
ay = tarihNesnesi.getMonth();
// ay değişkeni sistemin ay değerini verir. (0, 5, 10 gibi).
```

Player : Flash 5 ve sonrası

Date.getSeconds : Önceden oluşturulan tarih nesnesinin, saniye değerini verir. Verilen değer 0 ile 59 arasında tamsayı bir değerdir. Tarih nesnesine herhangi bir argüman verilmemişse, sistem bilgisinden aldığı değeri okur. Yazılımı,

```
tarihNesnesi.getSeconds();
```

Örnek :

```
tarihNesnesi = new Date();
saniye = tarihNesnesi.getSeconds();
// saniye değişkeni sistemin saniye değerini verir.
// (0, 15, 49 gibi).
```

Player : Flash 5 ve sonrası

Date.getTime : Önceden oluşturulan tarih nesnesine, eğer herhangi bir argüman verilmemişse, 1 Ocak 1970 tarihinin gece yarısından itibaren, sistemin o anki saatine kadar olan farkı, milisaniye cinsinden verir. Eğer tarih nesnesine bir argüman verilmişse, verilen argüman tarihinden itibaren sistemin o anki saatine kadar geçen süreyi verir. Bu metod, daha çok iki tarih arasındaki farkı bulmak için kullanılır. Yazılımı,

```
tarihNesnesi.getTime();
```

Örnek :

```
tarih_1 = new Date(85,4,8);
tarih_2 = new Date(72,1,5);
gecenSure = tarih_1.getTime() - tarih_2.getTime();
// gecenSure değişkeni tarih_1 ile tarih_2 arasındaki
// farkı milisaniye olarak verir.
// (418258800000 olacaktır).
```

Player : Flash 5 ve sonrası

Date.getTimezoneOffset : Önceden oluşturulan tarih nesnesinin, uluslarası saate (Greenwich) göre farkını dakika cinsinden verir. Bu metodu doğru olarak kullanabilmek için, tarih nesnesine herhangi bir argüman verilmemesi gerekir. Yazılımı,

```
tarihNesnesi.getTimezoneOffset();
```

Örnek :

```
fark = new Date().getTimezoneOffset();

trace(fark);
// Türkiye şartlarında bu değer, -120 olarak verilir.
// Yani greenwich'e göre -2 saat fark vardır.
// Ancak sistemdeki saat dilimi bilgisi farklı
// ise değer değişir.
```

Player : Flash 5 ve sonrası

Date.getUTCDate : UTC, Universal Time Coordinated yani Uluslararası Zaman Koordinatı demektir. Bir diğer deyişle, Greenwich zamanıdır. Bu metodla, sistemin zaman bilgisi alınıp, UTC'ye göre, ayın gün bilgisi belirlenir. Yani saat farkı çıkarılarak gün bulunur. Yazılımı,

```
tarihNesnesi.getUTCDate();
```

Örnek :

```
tarihNesnesi = new Date();
gunUTC = tarihNesnesi.getUTCDate();

// eğer sistem saati 01:00 ise, Türkiye şartlarına göre
// sistem gün bilgisinden bir gün önceki değeri verir.
// mesela sistem günü 25 ise gunUTC 24 olur.
```

Player : Flash 5 ve sonrası

Date.getUTCDay : Önceden oluşturulan tarih nesnesinin, UTC şartlarına göre haftanın gün bilgisi alınır. Saat farkına göre bulunan değer, greenwich saatine göre düzenlenmiş olur. Bulunan değer, 0 ile 6 arasındadır. 0 Pazar'ı, 6 ise Cumartesi'yi simgeler. Yazılımı,

```
tarihNesnesi.getUTCDay();
```

Örnek :

```
tarihNesnesi = new Date();
gunUTC = tarihNesnesi.getUTCDay();

// eğer sistem saati 01:00 ise, Türkiye şartlarına göre
// sistem gün bilgisinden bir gün önceki değeri verir.
// mesela sistem günü 2 (Salı) ise gunUTC 1 (Pazartesi) olur.
```

Player : Flash 5 ve sonrası

Date.getUTCFullYear : Önceden belirlenmiş olan tarih nesnesinin, UTC şartlarına göre yıl bilgisi alınır. Alınan bilgi dört basamaklı bir tamsayıdır. Türkiye şartlarında, sadece 1 Ocak tarihinin, saat 00:00 ile saat 02:00 'ye kadar olan diliminde bir önceki yılı verir. Diğer zamanlarda aynı yıl bilgisi alınır. Bu fark UTC'ye göre –2 saatlik zaman fakından doğar. Yazılımı,

```
tarihNesnesi.getUTCFullYear();
```

Örnek :

```
tarihNesnesi = new Date();
yilUTC = tarihNesnesi.getUTCFullYear();

// eğer sistem zamanı 1 Ocak 01:00 ise, Türkiye şartlarına göre
// sistemdeki yıl bilgisinden bir önceki yıl değeri verir.
// mesela sistem yılı 2001 ise yilUTC 2000 olur.
```

Player : Flash 5 ve sonrası

Date.getUTCHours : Önceden belirlenmiş olan tarih nesnesinin, UTC şartlarına göre saat bilgisi alınır. Alınan bilgi 0 ile 23 arasında tamsayı bir değerdir. Greenwich'e göre oluşan saat farkı çıkarılarak bulunur. Türkiye şartlarında, bu metodu kullandığınız zaman, iki saat önceki değeri alırsınız. Yazılımı,

```
tarihNesnesi.getUTCHours();
```

Örnek :

```
tarihNesnesi = new Date();
saatUTC = tarihNesnesi.getUTCHours();

// sistem saatinden, sisteme kayıtlı olan saat dilimi farkını
// çıkartarak saat değerini bulur.
// mesela sistem saati 13:00 iken,
// Türkiye şartlarında, 11:00 olur.
```

Player : Flash 5 ve sonrası

Date.getUTCMilliseconds : Önceden belirlenmiş olan tarih nesnesinin, UTC şartlarına göre milisaniye bilgisi alınır. Alınan bilgi 0 ile 999 arasından tamsayı bir değerdir. Ancak sistemdeki saat dilimi farkı olarak net saat bilgisi verildiğinden, bu değer `getMilliseconds` değeri ile aynıdır. Yazılımı,

```
tarihNesnesi.getUTCMilliseconds();
```

Örnek :

```
tarihNesnesi = new Date();
miliSaniyeUTC = tarihNesnesi.getUTCMilliseconds();
miliSaniye = tarihNesnesi.getMilliseconds();

// her iki değişkende aynı değeri verecektir.
```

Player : Flash 5 ve sonrası

Date.getUTCMinutes : Önceden belirlenmiş olan tarih nesnesinin, UTC şartlarına göre, dakika değeri alınır. Alınan değer 0 ile 59 arasında tamsayı değerdir. Ancak sistemdeki saat dilimi farkı olarak net saat bilgisi verildiğinden, bu değer `getMinutes` ile aynı olacaktır. Yazılımı,

```
tarihNesnesi.getUTCMinutes();
```

Örnek :

```
tarihNesnesi = new Date();
dakikaUTC = tarihNesnesi.getUTCMinutes();
dakika = tarihNesnesi.getMinutes();

// her iki değişkende aynı değeri verecektir.
```

Player : Flash 5 ve sonrası

Date.getUTCMonth : Önceden belirlenmiş olan, tarih nesnesinin, UTC şartlarına göre ay bilgisi alnır. Alıan bilgi 0 ile 11 arasında tamsayı bir değerdir. 0 Ocak ayını simgelerken, 11 Aralık ayını simgeler. Türkiye şartlarına göre, sadece her ayın ilk gününde saat 00:00 ile 02:00 arasında bir önceki ayı gösterir. Bunun dışındaki zamanlarda sitem bilgisindeki ay değeri verilir. Bu fark, greenwich'e göre –2 saatlik farktan oluşur. Yazılımı,

```
tarihNesnesi.getUTCMonth();
```

Örnek :

```
tarihNesnesi = new Date();
ayUTC = tarihNesnesi.getUTCMonth();

// sistem zamanı, 1 Haziran 01:00 ise,
// ayUTC değişkeni, Mayıs olacaktır.
```

Player : Flash 5 ve sonrası

Date.getUTCSeconds : Önceden belirlenmiş olan tarih nesnesinin, UTC şartlarına göre saniye bilgisi alınır. Alınan bilgi 0 ile 59 arasından tamsayı bir değerdir. Ancak sistemdeki saat dilimi farkı olarak net saat bilgisi verildiğinden, bu değer `getSeconds` değeri ile aynıdır. Yazılımı,

```
tarihNesnesi.getUTCSeconds();
```

Örnek :

```
tarihNesnesi = new Date();
saniyeUTC = tarihNesnesi.getUTCSeconds();
saniye = tarihNesnesi.getSeconds();

// her iki değişkende aynı değeri verecektir.
```

Player : Flash 5 ve sonrası

Date.getYear : Önceden belirlenmiş olan tarih nesnesinin, yıl bilgisini alır. Ancak alınan bilgi, eğer argüman olarak bir değer girilmemişse, sistem zaman bilgisinden aldığı değerden 1900 çıkartarak verir. Eğer argüman olarak sadece yıl girilmişse, 70 değerini verir (yıl bilgisini 1970 olarak algılar). Argüman olarak yıldan başka değerlerde girilmişse, yine 1900 çıkartarak bulur. Yazılımı,

```
tarihNesnesi.getYear();
```

Örnek :

```
tarihNesnesi = new Date();
yil = tarihNesnesi.getYear();

// Sistem zaman bilgisine göre yıl 2001 ise,
// yil değişkeni 101 (2001-1900) olacaktır.
```

Player : Flash 5 ve sonrası

Date.setDate : Önceden belirlenmiş olan tarih nesnesinin, ayın gün bilgisi ayarlanır. Ayarlanacak bilgi 1 ile 31 arasında bir tamsayı değeridir. Tarih nesnesine herhangi bir argüman verilmemişse, sistem zaman bilgisine göre ayarlama yapılır. Ancak sistemdeki zaman bilgisi değişmez. Sadece tarih nesnesinin tutuğu gün bilgisi değişmiş olur. Bu metodun argümanı, ayarlanacak olan gün bilgisidir. Yazılımı,

```
tarihNesnesi.setDate(gün);
```

Örnek :

```
tarihNesnesi = new Date(1985,2,23);
gun = tarihNesnesi.setDate(15);

// gun değişkeni 15 olarak ayarlandı.
// buna göre tarihNesnesi, 15 Mart 1985 olur.
```

Player : Flash 5 ve sonrası

Date.setFullYear : Önceden belirlenmiş olan tarih nesnesine, yıl, ay ve gün bilgisi ayarlanır. Bu metodun argümanları bu bilgilerdir. Argüman olarak sadece yıl bilgisi girilebileceği gibi her üç bilgiyide girilebilir. Bu argümanlardan yıl bilgisi olarak dört basamaklı sayı girilmelidir. Aksi halde girilen rakamı gerçek yıl olarak algılar. Mesela, 75 olarak girilen değer, M.S. 75 yılını simgeler. Bunu engellemek için 1975 yazılmalıdır. Ay olarak girilen değer 0 ile 11 arasında tamsayı değerdir. 0, Ocak ayını, 11 ise Aralık ayını simgeler. Gün bilgisi olarak 1 ile 31 arasında tamsayı bir değerdir. Yazılımı,

```
tarihNesnesi.setFullYear(yıl,[ay,[gün]]);
```

Örnek :

```
tarihNesnesi = new Date();
tarihNesnesi.setFullYear(1998,6,19);

// tarihNesnesi, 19 Temmuz 1998 olcaktır.
```

Player : Flash 5 ve sonrası

Date.setHours : Önceden belirlenmiş olan tarih nesnesine, saat bilgisi ayarlanır. Argüman olarak girilecek değer saat bilgisidir. Bu bilgi 0 (geceyarısı) ile 23 arasında tamsayı bir değerdir. Bu ayarlamayı yaptıktan sonra, sistem saatinde herhangi bir değişiklik olmaz. Sadece tarih nesnesinin tuttuğu saat bilgisi değişir. Yazılımı,

```
tarihNesnesi.setHours(saat);
```

Örnek :

```
tarihNesnesi = new Date();
tarihNesnesi.setHours(13);

// tarihNesnesinin tuttuğu değer, öğleden sonra 01:00 olacaktır
```

Player : Flash 5 ve sonrası

Date.setMilliseconds : Önceden belirlenmiş olan tarih nesnesinin, milisaniye bilgisi ayarlanır. Argüman olarak milisaniye cinsinden, 0 ile 999 arasında bir tamsayı değer girilir. Yazılımı,

```
tarihNesnesi.setMilliseconds(milisaniye);
```

Örnek :

```
tarihNesnesi = new Date();
tarihNesnesi.setMilliseconds(550);

// tarih nesnesinin tuttuğu milisaniye değeri 550 olacaktır.
```

Player : Flash 5 ve sonrası

Date.setMinutes : Önceden belirlenmiş olan tarih nesnesinin, dakika bilgisi ayarlanır. Argüman olarak dakika cinsinden, 0 ile 59 arasında bir tamsayı değer girilir. Yazılımı,

```
tarihNesnesi.setMinutes(dakika);
```

Örnek :

```
tarihNesnesi = new Date();
tarihNesnesi.setMinutes(42);

// tarihNesnesinin tuttuğu dakika değeri 42 olacaktır.
```

Player : Flash 5 ve sonrası

Date.setMonth : Önceden belirlenmiş olan tarih nesnesinin, ay ve gün bilgisi ayarlanır. Gün bilgisi opsiyoneldir ve 1 ile 31 arasında, tamsayı bir değerdir. Ay bilgisi ise, 0 ile 11 arasında, tamsayı bir değerdir. 0, Ocak ayını simgelerken, 11 Aralık ayını simgeler. Yazılımı,

```
tarihNesnesi.setMonth(ay,[gün]);
```

Örnek :

```
tarihNesnesi = new Date();
tarihNesnesi.setMonth(0,18);

// tarihNesnesinin ay bilgisi Ocak, gün bilgisi ise 18'dir.
```

Player : Flash 5 ve sonrası

Date.setSeconds : Önceden belirlenmiş olan tarih nesnesinin, saniye bilgisi ayarlanır. Argüman olarak verilen bu bilgi 0 ile 59 arasında, tamsayı bir değerdir. Yazılımı,

```
tarihNesnesi.setSeconds(saniye);
```

Örnek :

```
tarihNesnesi = new Date();
tarihNesnesi.setSeconds(35);

// tarihNesnesinin saniye bilgisi 35 olacaktır.
```

Player : Flash 5 ve sonrası

Date.setTime : Önceden belirlenmiş olan tarih nesnesinin, zaman bilgisi, milisaniye cinsinden ayarlanır. Argüman olarak verilen bu bilgi 0 ile 999 arasında, tamsayı bir değerdir. Yazılımı,

```
tarihNesnesi.setTime(milisaniye);
```

Örnek :

```
tarihNesnesi = new Date();
tarihNesnesi.setTime(435);

// tarihNesnesinin milisaniye bilgisi 435 olacaktır.
```

Player : Flash 5 ve sonrası

Date.setUTCDate : Önceden belirlenmiş olan tarih nesnesinin, gün bilgisini, UTC şartlarına göre ayarlar. Yani bu ayarla yapılan bir tarih nesnesi, Türkiye şartlarında, her gün 00:00 ile 02:00 saatleri arasında bir sonraki günü belirtir. Argüman olarak verilen bilgi, 1 ile 31 arasında tamsayı bir değerdir. Yazılımı,

```
tarihNesnesi.setUTCDate(gün);
```

Örnek :

```
tarihNesnesi = new Date();
tarihNesnesi.setUTCDate(30);
```

```
// tarihNesnesi, Türkiye şartlarında 01:00 ise
// gün bilgisi bir sonraki günü verir.
```

Player : Flash 5 ve sonrası

Date.setUTCFullYear : Önceden belirlenmiş olan tarih nesnesinin, yıl, ay ve gün bilgisi, UTC şartlarına göre ayarlanır. Argüman olarak girilen yıl bilgisi dört basamaklı, tamsayı bir değerdir. Ay bilgisi 0 ile 11 arasında, tamsayı bir değerdir. 0, Ocak ayını simgelerken, 11 Aralık ayını simgeler. Gün bilgisi, 1 ile 31 arasında tamsayı bir değerdir. Bu bilgiler UTC şartlarına göre ayarlandığından, Türkiye şartlarına göre, 00:00 ile 02:00 saatleri arasında bir sonraki günü, ayı veya yılı belirtir. Argümanlardan ay ve gün bilgisi opsiyonledir. Yazılımı,

```
tarihNesnesi.setUTCFullYear(yıl,[ay,[gün]]);
```

Örnek :

```
tarihNesnesi = new Date();
tarihNesnesi.setUTCFullYear(2000,11,31);

// tarihNesnesi, Türkiye şartlarına göre
// 31 Aralık 2000, 00:00 - 02:00 saatleri
// arasında ise, steUTCFullYear'dan sonra
// 2001,11,31 olacaktır.
```

Player : Flash 5 ve sonrası

Date.setUTCHours : Önceden belirlenmiş olan tarih nesnesinin, UTC şartlarına göre, saat, dakika, saniye ve milisaniye bilgisi ayarlanır. Argüman olarak verilen saat bilgisi, 0 ile 59 arasında tamsayı bir değerdir. Dakika bilgisi, 0 ile 59 arasında tamsayı bir değerdir. Saniye bilgisi, 0 ile 59 arasında tamsayı bir değerdir. Milisaniye ise, 0 ile 999 arasında tamsayı bir değerdir. Bu argümanlardan, dakika, saniye ve milisaniye opsiyoneldir. UTC şartlarına göre bu ayarlama yapılırsa, sadece saat bilgisi değerine saat farkı eklenir. Yani Türkiye için, 2 saat eklenir. Yazılımı,

```
tarihNesnesi.setUTCHours(saat,[dakika,[saniye,[milisaniye]]]);
```

Örnek :

```
tarihNesnesi = new Date();
tarihNesnesi.setUTCHours(20,30,45);

// tarihNesnesi, Türkiye şartlarına göre
// sistem saati, 20:30:45 ise, ayardan sonra, 22:30:45 olur.
```

Player : Flash 5 ve sonrası

Date.setUTCMilliseconds : Önceden belirlenmiş olan tarih nesnesinin, UTC şartlarına göre, milisaniye bilgisi ayarlanır. Argüman olarak verilen bu değer, 0 ile 999 arasında, tamsayı bir değerdir. Saat dilimi farkı sadece saat olarak verildiğinden, tarih nesnesinin tuttuğu milisaniye bilgisi değişir. Yazılımı,

```
tarihNesnesi.setUTCMilliseconds(milisaniye);
```

Örnek :

```
tarihNesnesi = new Date();
```

```
tarihNesnesi.setUTCMilliseconds(745);

// tarihNesnesi değişkeni, ilk önce 250 ise
// ayarlamadan sonra 745 olacaktır.
// Ancak diğer zaman birimleri değişmez.
```

Player : Flash 5 ve sonrası

Date.setUTCMinutes : Önceden belirlenmiş olan tarih nesnesinin, UTC şartlarına göre, dakika, saniye ve milisaniye bilgisi ayarlanır. Kullanımı, `setUTCHours` ile aynıdır. Yazılımı,

```
tarihNesnesi.setUTCMinutes(dakika,[saniye,[milisaniye]]);
```

Örnek :

```
tarihNesnesi = new Date();
tarihNesnesi.setUTCMinutes(12,10,632);

// sadece tarihNesnesi değişkeninin, tuttuğu
// değerler değişir.
```

Player : Flash 5 ve sonrası

Date.setUTCMonth : Önceden belirlenmiş olan tarih nesnesinin, UTC şartlarına göre, ay ve gün bilgisi ayarlanır. Argüman olarak verilen ay bilgisi, 0 ile 11 arasında tamsayıdır. 0, Ocak ayını simgelerken, 11 Aralık ayını simgeler. Gün bilgisi ise, 1 ile 31 arasında tamsayı bir değerdir. Ancak gün bilgisi opsiyoneldir. Saat dilimi farkından oluşan farkı ekleyerek zamanı bulur. Türkiye şartlarına göre, 00:00 ile 02:00 saatleri arasında, bir sonraki günü, eğer ayın son günü ise, bir sonraki ayı verir. Yazılımı,

```
tarihNesnesi.setUTCMonth(ay,[gün]);
```

Örnek :

```
tarihNesnesi = new Date();
tarihNesnesi.setUTCMonth(0,31);

// tarihNesnesi, Türkiye şartlarına göre
// 31 Ocak, 00:00 - 02:00 saatleri arasında ise
// 1 Şubat olarak belirlenir.
```

Player : Flash 5 ve sonrası

Date.setUTCSeconds : Önceden belirlenmiş olan tarih nesnesinin, UTC şartlarına göre, saniye ve milisaniye bilgisi ayarlanır. Kullanımı, `setUTCHours` ile aynıdır. Yazılımı,

```
tarihNesnesi.setUTCSeconds(saniye,[milisaniye]);
```

Örnek :

```
tarihNesnesi = new Date();
tarihNesnesi.setUTCSeconds(15,815);

// sadece tarihNesnesi değişkeninin, tuttuğu
// değerler değişir.
```

Player : Flash 5 ve sonrası

Date.setYear : Önceden belirlenmiş olan tarih nesnesinin, yıl bilgisi ayarlanır. Argüman olarak verilen bu bilgi dört basamaklı tamsayı bir değerdir. Eğer bir ve iki basamaklı sayı girilirse, 1900 tarihine ekleyerek bulur. Ancak üç basamaklı sayı girilirse, bu sayıyı olduğu gibi kabul eder ve yıl olarak belirler. Yazılımı,

```
tarihNesnesi.setYear(yıl);
```

Örnek :

```
tarihNesnesi = new Date();
tarihNesnesi.setYear(2032);

// tarihNesnesi değişkeninin tuttuğu yıl bilgisi 2032 olacaktır
```

Player : Flash 5 ve sonrası

Date.toString : Tarih nesnesi olarak belirlenen nesneyi, string olarak yorumlar. Yazılımı,

```
tarihNesnesi.toString();
```

Örnek :

```
tarihNesnesi = new Date();
trace(tarihNesnesi.toString());

// pencerede, Wed Feb 14 13:27:52 GMT+0200 2001
// gibi bir çıktı alınır.
```

Player : Flash 5 ve sonrası

Date.UTC : Başlı başına bir tarih nesnesidir. Argüman olarak verilen tarih ile 1 Ocak 1970 tarihinin gece yarısına kadar olan farkı, milisaniye cinsinden verir. Yeni oluşturulan tarih nesnesi içine argüman olarak yazılırsa, UTC şartlarına göre tarih tanımlanmış olur. Yazılımı,

```
Date.UTC(yıl,[ay,[gün,[saat,[dakika,[saniye,[milisaniye]]]]]]);
```

Örnek :

```
Date.UTC(1985,1,25);

// 1 Ocak 1970 tarihine kadar, 478137600000 milisaniye fark var

imzaTarihi = new Date(Date.UTC(2001,0,30));

// imzaTarihi değişkenini, UTC şartlarına göre tanımlamış.
```

Player : Flash 5 ve sonrası

delete : Türkçe karşılığı, silmek olan bu komut, bir değişkeni veya nesneyi siler. Bilgisayarın hafızasını boşalttığından, scriptlerin çalışmasını hızlandırır. ActionScript ile kodları oluştururken, işe yaramayan tüm değişken ve nesneler, bu yolla silinebilir. Silme işlemi gerçekleştiği zaman, sonuç true olarak geri döner. Eğer silme işlemi başarısız olmuşsa, sonuç false olarak geri döner. Önceden

tanımlanmış olan değişkenler, nesneler ve özellikler var ile oluşturulduğundan, bu metod ile silinmeyebilir (`Math.E` gibi..). Yazılımı,

```
delete referans;
```

Örnek :

```
dizi = new Array();
dizi[0] = "abc";
dizi[1] = "def";
dizi[2] = "ghi";

// dizi.length =3 olacaktır.

delete dizi[2];

// dizi'nin 3. elemanı silinmiştir. Ancak dizi.length değişmez
```

veya ;

```
hesap = new Object();
hesap.isim = 'ibrahim'
delete hesap.isim

// hesap adlı nesnenin isim özelliği silinmiştir.
```

Player : Flash 5 ve sonrası

do...while : En az bir kere de olsa, arada kalan işlerin bir döngü içinde yapılmasıdır. `While` ifadesinden sonra gelen koşul yerine getirildiği müddetçe, süslü parantez içinde kalan durumlar gerçekleşir. Yani, `while` ifadesinden sonra yazılan argüman değeri false olunca, döngü durur. Eğer ilk döngüde bu koşul gerçekleşirse, döngü durur ve sonraki ifadeler uygulanır. Diğer döngülerden farkı en az bir kere dönmüş olmasıdır. Yazılımı,

```
do {
      durumlar;
} while (koşul);
```

Örnek :

```
i = 4
do {
      klip.duplicateMovieClip("klip" +i, i );
      i --;
} while (i > 0);

// while ifadesinden sonraki i>0 koşulu false oluncaya kadar
// döngü devam edecek. Her döngüde i bir eksileceğinden
// 0 > 0 olunca false olacak ve döngü duracaktır.
// Eğer başlangıçta i=0 olsaydı bile, döngü bir kere dönecekti.
```

Player : Flash 4 ve sonrası

_droptarget : Sürükleme yöntemi ile alınan bir movie clip'in bırakıldığı yeri belirtir. `startDrag` ile bir movie clip'i fare işaretçisine yapıştırabiliriz. Bu movie clip'i, sahnenin herhangi bir

yerine `stopDrag` ile fare işaretçisinden bırakabiliriz. Fare işaretçisine yapışan bu movie clip'in merkez noktası, bıraktığımız yerde başka bir movie clip' e temas ediyorsa, ters bölü işareti ile bu movie clip'in durum adını verir. Baş tarafına koyulan bu ters bölü işaretini kaldırıp `level` ifadesi koymak için, `eval` fonksiyonu kullanılır. Yazılımı,

```
movieClipDurumAdı._droptarget;
```

Örnek : İki adet movie clip yapın. Bunlardan birine (Durum adı : klip_1) ikinci keyframe'i yerleştirin ve o frame'de rengini değiştirin. İlk frame'e ise `stop();` aksiyonunu verin. Diğer movie clip'i sağ tıklayıp, edit'i seçin ve gelen yeni sahnedeki şekli seçip F8'e basın. Bu şekli ise Button olarak tanımlayın ve aşağıdaki kodları yazın.

```
on (press) {
      startDrag (this, true);
}
on (release) {
      stopDrag ();
      if (this._droptarget == "/klip_1") {
      _root.klip_1.gotoAndStop(2);
      }
}

// Programı çalıştırın ve düğmeyi diğer movie clip'in
// üstüne sürükleyin. Bıraktığınızda, üzerine bırakılan
// movie clip renk değiştirecektir.
```

Bu örnekte, `this._droptarget` ifadesi, sürüklemekte olduğunuz movie clip'in bırakıldığı yeri belirtecektir. Eğer başka bir movie clip'e temas ediyorsa "`/`" işareti ile, üzerine bırakılan movie clip'in durum adını alır. Yani, "`/klip_1`" olacaktır. Bu durumda ise, if koşulu gerçekleşecek ve klip_1 adlı movie clip'in ikinci frame'ine gidecektir.

Player : Flash 4 ve sonrası

duplicateMovieClip : Sahnede oynamakta olan bir movie clip'i kopyalayıp yeni bir movie clip oluşturur. Herhangi bir movie clip, bu yöntem ile çoğaltığında, ilk frame'den itibaren oynamaya başlar. Üç adet argümanı vardır. Birincisi kopyalanacak olan movie clip'in durum adıdır. İkincisi, yapıştırılacak olan yeni movie clip'in adıdır. Sonuncusu ise, yeni movie clip'in derinlik seviyesidir. Derinlik seviyesi (Depth level), movie clip'i tanımlamaya yarar. Bir movie clip'i ilk kez oluşturduğunuzda, derinlik seviyesi 0'dır. Bundan sonra kopyalanacak olanlar farklı derinlik seviyesinde olmalıdır. Yazılımı,

```
hedefMovieClip.duplicateMovieClip("yeni isim", seviye);
duplicateMovieCLip("hedef", "yeni isim", seviye);
```

Örnek : Sahneye, "daire" durum isimli bir movie clip çizin. Başka bir yere bir düğme yapın ve aşağıdaki kodları yazın.

```
on(release) {
      sayac = 10;
      while(sayac>0) {
            duplicateMovieClip (_root.daire, "yeniMC" + i, i);
            setProperty("yeniMC" + i, _x, random(275));
            setProperty("yeniMC" + i, _y, random(275));
            setProperty("yeniMC" + i, _alpha, random(275));
            setProperty("yeniMC" + i, _xscale, random(50));
            setProperty("yeniMC" + i, _yscale, random(50));
            i = i + 1;
            sayac = sayac-1;
      }
```

```
}

// Düğmeye her bastığınızda,
// farklı yer, görünürlük ve ölçülerde 10 adet yeni
// movie clip belirecektir.
```

Player : Flash 4 ve sonrası

else : Herhangi bir `if` koşulu veya kendisinden önceki `else if` koşulları gerçekleşmediği zaman, ne olursa olsun, bundan sonraki ifadeler uygulanır. Yani kendisinden önce gelen tüm ifadeler false olarak dönmüşse, `else` ifadesinden sonra gelen durumlar uygulanır. Yazılımı,

```
if (koşul){
      durum(lar);
} else{
      durum(lar);
}
```

Örnek :

```
if (i=100){
      gotoAndPlay(12);
} else{
      gotoAndPlay(1);
}

// i, 100 ise 12. frame'e gidip oynamaya devam edecektir.
// değilse, başa dönecektir.
```

Player : Flash 4 ve sonrası

else if : Kendisinden önce gelen `if` koşulu veya `else if` koşulları gerçekleşmiyorsa, bu komuttan sonra gelen koşula bakılır. Eğer gerçekleşmişse, bu komuta ait durumlar uygulanır. Eğer gerçekleşmemişse, bir sonraki satıra geçer. Bir önceki `else` komutundan farkı, kendisine ait bir koşulun olmasıdır. Yani `else` ifadesi ne olursa olsun uygulanırken, `else if` ifadesi kendisine ait koşulun gerçekleşme durumuna bakar. Yazılımı,

```
if (koşul){
      durum(lar);
} else if (koşul){
      durum(lar);
}
```

Örnek :

```
if (i=100){
      gotoAndPlay(12);
} else if (i=50){
      gotoAndPlay(6);
} else{
      gotoAndPlay(1);
}

// i, 100 ise 12. frame'e gidip oynamaya devam edecektir.
// i, 50 ise 6. frame'e gidip oynamaya devam edecek.
// değilse, başa dönecektir.
```

Player : Flash 5 ve sonrası

eq : İngilizce equal (eşit) kelimesinin kısaltılarak yazılışıdır. İki ifadenin birbirine olan eşitliğini sınamak için kullanılır. Bu ifadeler rakam, string veya değişken olabilir. Normal eşit işareti (=) ile karıştırılmamalıdır. Eşittir (=) işareti, atama işlemi için kullanılır. Sınamanın sonucu doğru ise, true değerini verir. Aksi halde false değeri verir. Yazılımı,

```
ifade_1 eq ifade_2
```

Örnek :

```
if (_root.isim eq "Osman"){
      mesaj = "Hoşgeldiniz";
} else{
      mesaj = "İsim yanlış";
}
```

Player : Flash 1 ve sonrası. Bu ifade, Flash 5'te yerini "==" işaretine bırakmıştır.

escape : Argüman olarak verilen bir ifadeyi, URL encoded formatında bir stringe çevirir. Yazılımı,

```
escape(ifade);
```

Örnek :

```
mesaj = escape("Merhaba Dünya");
trace(mesaj);

// programı çalıştırdığınız zaman, OutPut penceresinde
// "Merhaba%20D%Fcnya" ibaresi görünür.
```

Player : Flash 5 ve sonrası

eval : Argüman olarak verilen bir ifadeyi string olarak yeniden değerlendir. Bu argüman, eğer bir değişken adı veya bir özellik ise, değişkenin değeri veya özelliği verir. Argüman, eğer bir nesne veya movie clip ise, nesne veya movie clip'in referansı ile birlikte verir. Eğer argüman, bir element olarak adlandırılmış ise, `eval` fonksiyonu boş olarak geri döner. Flash 4'te, `eval` fonksiyonu, dizi değişkenleri taklit etmek için kullanılırdı. JavaScript'te kullanılan `eval` fonksiyonu ile aynı değildir. ActionScript'in başka bir komutu olan `evaluate` ile karıştırılmamalıdır. Yazılımı,

```
eval (ifade);
```

Örnek :

```
metin = "Bilgi, Güçtür";
yeniMetin = eval("metin");
trace(yeniMetin);

// yeniMetin, metin adlı değişkenin değerini alır.

klipAdi = eval(_root.daire);

// klipAdi değişkeni, _level0.daire olacaktır.
```

Player : Flash 5 ve sonrasında tüm özellikleri ile kullanılabilir. Ancak Flash 4 Player olarak da dışarı çıkarabilirsiniz. Flash 4 olarak dışarı çıkarıldığında, ters bölü işareti kullanılmalıdır. Bu halde iken, sadece değişkenler için kullanılabilir.

evaluate : ActionScipt editörünü, normal modda kullananlar için boş bir satır açar ve arkasına noktalı virgül (;), yani satırın bittiğini belirten işareti koyar. Bir değişken tanımlama veya atama işlemleri için kullanma ihtiyacı duyulduğunda, editörün solundaki menüden, `evaluate` seçeneği çift tıklanırsa bu boş satırı açar. Expert modu kullananlar için böyle bir komut yoktur.

Player : Flash 5 ve sonrası

_focusrect : Animasyonda bir metin kutusu ve bu metin kutusuna bağlı olarak çalışan bir düğme varsa ve kullanıcı klavyeden tab tuşu ile geçiş yapıyorsa, son metin kutusundan çıktıktan sonra, düğmenin etrafında sarı bir dörtgen belirir. Yani metin kutularını tek tek doldurduktan sonra, tab tuşuna basıldığı an düğmenin etrafında sarı bir dörtgen belirecektir. Varsayılan değer olarak true, yani herhangi bir değişiklik yapılmamışsa, sarı dörtgen her seferinde çıkacaktır. Ancak zaman eksenindeki bir frame'e false değeri verilirse veya düğmenin `rollOver` durumuna false değeri verilirse, bu sarı dörtgen çıkmaz. Yazılımı,

```
_focusrect = Boolean değeri;
```

Örnek : Bir metin kutusu ve bir düğme yapın. Hiç bir değişiklik yapmadan çalıştırın ve metin kutusuna tıkladıktan sonra, klavyeden tab tuşuna basın. Sarı dörtgeni göreceksiniz. Bunu engellemek için düğmeye şu aksiyonları yazın.

```
on (rollOver){
      _focusrect = false;
}

// sarı dörtgen, artık görünmeyecektir.
```

Player : Flash 4 ve sonrası

for : Çoğu programlama dilinde olduğu gibi, bir döngü ifadesidir. Argüman olarak döngü şartları verilir. Her bir argüman, noktalı virgül ile ayrılır. İlk argüman, döngünün başlangıç durumunu belirler. İkinci argüman, döngünün hangi koşulda duracağını belirtir. Koşul true olduğu müddetçe döngü devam eder. Koşul false değerini verirse döngü durur. son argüman ise, sonraki döngü değerini belirtir. Argümanlardan sonra yazılan ve süslü parantez arasında kalan ifadeleri, her döngüde yürütür. Yazılımı,

```
for (başlangıç; koşul; sonraki döngü değeri) {
      ifadeler;
}
```

Örnek :

```
dizi = new Array("abc", "cde", "fgh", "ijk");
for (i=0; i<4; i++){
      trace(dizi[i]);
}

// çıktı penceresinde, tüm dizi elemanları tek tek yazılır.
// i değişkeni her döngüde bir arttırılıyor (i++).
// i=4 olunca koşul false değeri verecek ve döngü duracaktır.
```

Player : Flash 5 ve sonrası

for...in : Bir nesnenin özelliklerini veya dizi içindeki elemanları döndürerek, sonraki satırlarda belirtilen ifadeleri yürütür. İki adet argümanı vardır. Bu argümanların arasına in ifadesi yazılır. Argümanlardan ilki, bir nesnenin her bir özelliğini belirten referansı veya dizi elemanlarını belirtir. Diğer argüman ise, nesnenin kendisi veya dizi ismidir. Yazılımı,

```
for (referans in nesne){
      ifadeler;
}
```

Örnek :

```
nesne = { adi:'Mustafa', yas:26, sehir:'Ordu' };
for (name in nesne) {
      trace ("nesne." + name + " = " + nesne[name]);
}

// çıktı penceresinde,
// nesne.adi = Mustafa
// nesne.yas = 26
// nesne.sehir = Ordu
// görünecektir.
```

Player : Flash 5 ve sonrası

_framesloaded : Flash animasyonu içeren bir web sayfasını ziyaret eden kişi, bütün frame'lerin tek tek yüklenmesini bekler (veya tek tek izler). Bu komut, yüklenmiş olan frame'lerin boyutunu verir. Eğer önüne bir movie clip'in durum adı yazılırsa, sadece bu movie clip'e ait yüklenen frame'lerinin boyutunu verir. Tek başına kullanılırsa, tüm animasyonun yüklenen frame boyutunu verir. Daha çok preloader (önyükleme) tekniğinde kullanılır. Yazılımı,

```
movie clip durum adı._framesloaded;
```

Örnek :

```
if (_framesloaded >= _totalframes) {
      gotoAndPlay (3);
} else {
      setProperty ("_root.bar", _xscale,
      (_framesloaded/_totalframes)*100);
}

// yüklenmiş olan frame boyutu ile toplam frame boyutu
// eşit ise, 3. frame gidip oynamaya devam edecek.
// değilse sahnedeki bar adlı movie clip'in uzunluğu
// yüklenmiş boyuta göre artacak.
```

Player : Flash 4 ve sonrası

fscommand : Flash ile yapılan animasyonlar, Flash Player eklentisi ile çalışırlar (swf). Flash Player'dan diğer programlara geçiş yapabilmek için JavaScript komutları kullanılır. Bir HTML sayfasında Flash ile JavaScript'i ilişkilendirebilmek için, fscommand komutu kullanılır. Standart olarak Flash'a etki edebilecek beş komut bulunurken, `exec` komutu ile diğer programlara geçiş yapılabilir. İki adet argümanı vardır. Birincisi, icra edilecek komut. Diğeri ise buna ait kodlardır. Yazılımı,

```
fscommand (komut, argüman);
```

Örnek :

```
// animasyonun ilk frame'ine şu kodları yazın

fscommand ("fullscreen", "true");

// dışarı çıkarıp çalıştırdığınızda, tüm ekranı kaplar.
// bir düğme yapın ve şu aksiyonları verin.

on (release) {
      fscommand("quit", "true");
}
// dışarı çıkarıp çalıştırın ve düğmeye basın.
// program kapanacaktır.

// bir düğme daha yapın ve şu kodları yazın

on (release) {
    fscommand ("exec", "start\tcalc.exe");
    fscommand ("exec", "cmd\t/c\tcalc.exe");
}

// exec komutu ile, windows'un hesap makinasını çalıştırır
```

Player : Flash 3 ve sonrası

function : Sürekli olarak aynı işler, başka değerlere göre yapılıyorsa, bu işleri ayrı ayrı yazmaktansa bir kez yazıp, her seferinde bu işin yapılacağı yeri çağırmak daha kolay olur. Belirttiğiniz işlerin, farklı değerlere göre yapılıp size başka değerler gönderen, kodların tümüne fonksiyon denir. Fonksiyon, sınırsız sayıda argüman içerebilir. Yapılmasını istediğiniz işleri bir fonksiyona tanımlayıp değişkenleri istediğiniz yerde gönderdiğiniz zaman, bu alınan değişken değerlerine göre, yapılması gereken işleri yapar ve size istediğiniz değerleri geri gönderir. Yazılımı,

```
function fonksiyon adı(argüman1,argüman2...argümanN) {
      yapılacak işler;
}
```

Örnek :

```
function mesaj (saat) {
      if (saat < 11){
            metin = "Günaydın";
      } else if (saat < 15){
            metin = "İyi günler";
      } else if (saat < 20){
            metin = "İyi akşamlar";
      } else{
            metin = "İyi geceler";
      }
return metin;
}

tarih = new Date();
saat = tarih.getHours();
trace(mesaj(saat));
```

Boş bir çalışma sahnesinin ilk frame'ine bu kodları yazın ve çalıştırın. Sistem saatine göre, OutPut penceresinde, mesaj görünecektir. `Mesaj` adlı fonksiyonu herhangi bir yerde çağırabilirsiniz. Bu fonksiyonun yaptığı iş şöyledir. Kendisine, bir `saat` adlı değişken bilgisi gelecektir. Bu değişken değerine göre `metin` değişkeni tanımlanacak. Ve `metin` adlı değişkeni geri gönderecektir. Bu örnekte, `saat` adlı değişken, bir tarih nesnesinin `getHours` metodu ile alınan saat bilgisinden geliyor. Daha sonra, `trace` komutu içinde fonksiyonu çağırıyoruz. Fonksiyonun argümanı `saat` adlı değişken olacaktır.

Player : Flash 5 ve sonrası

ge : İngilizce **g**reater than or **e**qual (büyük veya eşit) kelimelerinin kısaltılmış halidir. Birinci ifadenin, ikinciden büyük olup olmadığını veya birbirine eşit olup olmadıklarını sınar. Bu ifadeler sayı, string veya değişken olabilirler. Sınamanın sonucu doğru ise, true değerini verir. Aksi halde false değeri verir. Yazılımı,

```
ifade_1 ge ifade_2;
```

Örnek :

```
if (Number(_framesloaded) ge Number(_totalframes)){
      gotoAndStop(2);
}
```

Player : Flash 4 ve sonrası. Flash 5'te bu komutun yerine "<=" işareti kullanılır.

getProperty : Bir movie clip'in, argüman olarak belirtilen özelliğini verir. Argüman olarak ilk önce, özelliği alınacak movie clip'in durum adı yazılır. Bir movie clip'in bir çok özelliği alınabilir. Sahnedeki yeri, görünürlük oranı, ölçüleri gibi. Yazılımı,

```
getProperty (movie clip durum adı, özellik);
```

Örnek :

```
yer = getProperty ("klip1", _x);
setProperty ("klip2", _x, yer+10);

// yer adlı değişkene, klip1 adlı movie clip'in x eksenindeki
// yeri atanıyor. Bu değişkene 10 eklenip, klip2'nin yeri
// belirleniyor
```

Player : Flash 4 ve sonrası

getTimer : Animasyon çalıştırıldığında, programa ait bir kronometre de çalışmaya başlar. Bu kronometre bilgisini almak için, `getTimer` metodu kullanılır. Bu komutun çalıştığı yerde, animasyonun başlangıcından, o ana kadar geçen süreyi, milisaniye cinsinden verir. Yazılımı,

```
getTimer();
```

Örnek :

```
on (press){
      sure = getTimer();
      trace(sure);
}
```

```
// Sahneye bir düğme çizin ve yukarıdaki kodları yazın
// ikinci bir keyframe yerleştirip çalıştırın
// düğmeye her bastığınızda geçen süreyi görürsünüz
```

Player : Flash 4 ve sonrası

getURL : Argüman olarak belirtilen bir URL adresine gider veya bu adrese bilgi gönderir. Üç adet argümanı vardır. İlk argüman, gidilecek olan URL adresidir. İkinci argüman, ilgili URL adresini, hangi pencerede açacağıdır.

- _self : Mevcut pencerenin, mevcut frame'inde açar.
- _blank : Yeni bir pencerede açar.
- _parent : Mevcut açık pencerede açar.
- _top : Mevcut pencere içinde, üst seviyede açar.

Son argüman ise, gidilecek olan URL adresine bilgi gönderileceği zaman kullanılır. Eğer bu bilgiler sınırlı sayıda ise, GET metodu ile bir yumak halinde gider. Gönderilecek bilgiler, POST metodu ile sınırsız sayıda ve HTTP başlıklarına göre ayrılmış olarak gider. Yazılımı,

```
getURL(URL adresi, [pencere, [metod]]);
```

Örnek :

```
on (release) {
      getURL("http://www.webteknikleri.com", "_blank");
}

// düğmeye basılıp bırakıldığında, ilgili adresi
// başka bir pencerede açacaktır.
```

Player : Flash 2 ve sonrası. Metod kısmı, Flash 4'ten sonrası için geçerlidir.

getVersion : Animasyonun çalışmakta olduğu işletim sistemi ve Flash Player'in versiyonu hakkında bilgi verir. Yazılımı,

```
getVersion();
```

Örnek :

```
trace(getVersion());

// çıktı olarak WIN 5,0,30,0 değerleri görünür
// WIN; Windows işletim sistemini
// 5; major versiyonu
// 0,30,0; minör versiyonu temsil eder.
```

Player : Flash 5 ve sonrası

gotoAndPlay : Zaman ekseni üzerindeki oynatma başlığını, argüman olarak verilen scene ve frame'e gönderir ve oradan oynamaya devam eder. Scene, opsiyoneldir. Eğer aynı scene'de bir yere gönderilecekse, bu argüman yazılmaz. Yazılımı,

```
gotoAndPlay(scene, frame);
```

Örnek :

```
on (press){
      gotoAndPlay("scene2", "basla");
}

// düğmeye basıldığı an, scene2 adlı sahnenin
// basla etiket isimli (label) frame'ine gidecek
```

Player : Flash 2 ve sonrası

gotoAndStop : Zaman ekseni üzerindeki oynatma başlığını, argüman olarak verilen scene ve frame'e gönderir ve durdurur. Scene opsiyoneldir. Eğer aynı scene'de bir yere gidecekse, yazılmaz. Yazılımı,

```
gotoAndStop(scene, frame);
```

Örnek :

```
if (skor > 100){
      gotoAndStop(20);
}

// skor 100 ise, aynı scene'de 20 nolu frame'e gidip duracak
```

Player : Flash 2 ve sonrası

gt : İngilizce, **g**reater **t**han (büyüktür) ifadesinin kısaltılmış halidir. Birinci ifadenin, ikinci ifadeden büyük olup olmadığını sınar. Bu ifadeler sayı, string veya değişken olabilir. Sınamanın sonucu doğru ise, true değerini verir. Aksi halde false değeri verir. Yazılımı,

```
ifade_1 gt ifade_2;
```

Örnek :

```
if (deger_1 gt deger_2){
      trace("deger_1 buyuk");
} else{
      trace("deger_2 buyuk");
}

// sınamanın sonucuna göre, çıktı alınır
```

Player : Flash 4 ve sonrası. Flash 5'te, bu ifadenin yerine ">" işareti kullanılır.

_height : Bir movie clip'in yükseklikle ilgili bilgisinin, ayarlanıp veya alındığı bir özelliktir. Flash'ın bundan önceki sürümlerinde, bu özellik ile sadece yükseklik bilgisi alınabilirdi. Flash 5 ile, bu özellik ayarlanabilir hale getirilmiş. Yazılımı,

```
değer = movie clip durum adı._height;
// movie clip'in yükseklik bilgisi alınması

movie clip durum adı._height = değer;
// movie clip'in yükseklik bilgisinin ayarlanması
```

Örnek :

```
onClipEvent(mouseDown) {
      _width=200;
      _height=200;
}

// bir movie clip'e yukarıdaki aksiyonları yazın
// programı çalıştırın ve farenin sol tuşuna basın
// yükseklik ve genişlik değişecektir.
```

Player : Flash 4 ve sonrası

_highquality : Oynamakta olan bir movie clip'in, anti-aliasing seviyesini ayarlayan bir özelliktir. Bu ayar, üç değerle yapılabilir. Bir bitmap resmin pürüzsüzlüğü kalitesinde anti-aliasing değeri için, 2 verilmelidir. Orta kalitede bir anti-aliasing değeri için, 1 verilir. En düşük kalitede anti-aliasing için, 0 verilir. Yazılımı,

```
_highquality = değer;
```

Örnek :

```
on (press){
      _root.klip._highquality = 0;
}

// düğmeye basıldığı an, düşük kalite anti-aliasing
// değeri verilmiş olur.
```

Player : Flash 4 ve sonrası

if : Tüm programlama dillerinde olduğu gibi, programın akışını etkileyen, bir şarta veya şartlara bağlı olarak bir takım işlerin yapılması sağlanır. Bu komut tek başına çalışabildiği gibi, birden fazla şartı ard arda sınamak mümkündür (`else if`). Argüman olarak, koşul verilir. Eğer koşul gerçekleşmişse, true olarak döner ve kendisine ait olan işler icra edilir. Koşul false olarak dönerse, kendisine ait işler icra edilmez ve başka bir şartın olup olmadığına bakar. Eğer varsa tek tek bu şatları kontrol eder. Eğer böyle bir şart yoksa, sonraki satırlara geçerek devam eder. Yazılımı,

```
if (koşul) {
      icra edilecek işler;
}
```

Örnek :

```
if (isim == "Cüneyd") {
      getURL("http://www.ocal.net/kisiler.asp?isim"+isim);
}
// eğer isim "Cüneyd" ise, ilgili web sayfasına
// isim adlı değişkeni gönderir.
```

Player : Flash 4 ve sonrası

ifFrameLoaded : Animasyonun yer aldığı sayfayı ziyaret edenler, animasyonu ön yükleme komutları eklenmemişse, frame frame yüklenmesini izlerler. Bu durum özellikle, büyük boyutlu animasyonlarda, akıcı bir şekilde izlenmesine engel olur. Animasyonu akıcı bir şekilde izleyebilmek için, bütün framelerin önceden yüklenmiş olması gerekir. Bunu yapabilmek için, `ifFrameLoaded` komutu

kullanılır. Argüman olarak belirttiğiniz scene ve frame'ler yüklenmişse, kendisine ait olan işleri icra eder. Bu komut Flash 3 ile birlikte gelmiştir ve ancak bir tek koşul gerçekleşirse, işler yapılır. Flash 5 ile `_framesloaded` komutu gelmiştir. Böylece `if, else if` ve `else` komutlarını ayrı ayrı kullanabilme özelliği kazandırılmıştır. Yazılımı,

```
ifFrameLoaded (scene, frame);
ifFrameLoaded (frame);
```

Örnek :

```
ifFrameLoaded (65){
      gotoAndPlay(3);
}

// ilk frame'e bu kodları yazdıktan sonra, ikinci keyframe'e

gotoAndPlay(1);

// yazılır. Böylece animasyon tamamı ile yüklenince oynar.
```

Player : Flash 3 ve sonrası

#include : Dışarıdan script dosyası eklemek için kullanılır. Bazı değişkenleri dışarıda bir dosyada tutarak, animasyonun gerçek boyutunu azaltabilir. İhtiyaç durumunda bu dosyayı çağırabilirsiniz. Dışarıdan eklenecek olan dosyanın uzantısı *.as olmalıdır. Yazılımı,

```
#include "dosya adı.as"
```

Örnek :

```
// Bir movie clip (durum adı;kutu) ve bir düğme yapın.
// İlk frame'e şu kodu yazın

x = "1";

// Düğmeye aşağıdaki aksiyonları verin.

on (press) {
      #include "dosya.as"
}

// Metin editörünü açın ve aşağıdaki kodları yazıp
// dosya.as olarak kaydedin.

Ykutu = "kutu" add x;
duplicateMovieClip ("kutu", Ykutu, x);
setProperty          (Ykutu,          _x,          Number(getproperty(Ykutu,
_x))+Number(x*(getproperty(Ykutu, _width))));
x = Number(x)+1;
set (Ykutu add ":x", x);

// export movie ile animasyonu, dosya.as ile aynı yere çıkartın
// swf dosyasını açın ve düğmeye basın.
```

Aslında include dosyasındaki kodları, direkt olarak düğmenin aksiyonlarına da verebiliriz. Söz konusu kodlar karmaşık ve büyük boyutlarda ise, dışarıda tutabiliriz. Örnekte, düğmeye basıldığı anda, dosya.as adlı dosyayı çağıracak ve içindeki kodları icra edecektir.

Player: YOK

Infinity : Türkçe karşılığı, sonsuzluk olan bu kod aslında bir değerdir. Çok büyük sayılar ile işlem yapılıyorsa, bu değeri bir değişkene atayarak, karşılaştırma yapılabilir. Bu değerin negatif veya pozitif olması önemli değildir. Eğer, negatif sonzuluğa ihtiyaç duyarsanız, `Number.NEGATIVE_INFINITY` değerini veya pozitif sonsuzluk için, `Number.POSITIVE_INFINITY` değerini kullanabilirsiniz. Yazılımı,

```
Infinity
```

Örnek :

```
a = Math.pow(211, 132);
b = Math.pow(42, 63);
c = Math.pow(22, 213);
d = Math.pow(35, 111);
e = Math.pow(97, 75);

dizi = new Array(a, b, c, d, e);
diziAD = new Array("a", "b", "c", "d", "e");

x = Infinity;
for (i=0; i<5; i++) {
      if (dizi[i] < x) {
            x = dizi[i];
            y = i;
      }
}

trace("En küçük sayı: "+ diziAD[y]+" = "+ x);

// x değişkenine en büyük değer olan sonsuzluk veilir.
// eğer dizi içindeki elemanlar bu değerden küçükse,
//  x değişkeninin değeri olur.
// En küçük sayı: b = 1.83952330826244e+102
```

Player : Flash 5 ve sonrası

int : Argüman olarak verilen bir sayıyı tamsayıya çevirir. Yani ondalıklı kısmını atar ve virgülden önceki kısmı verir. Yazılımı,

```
int(sayı);
```

Örnek :

```
x = int(12.873463);

// x = 12 olacaktır.
```

Player : Flash 4 ve sonrası. Flash 5'te bu komutun yerine "`Math.floor`" kullanılır.

isFinite : Bir sayının sonlu olup olmadığını kontrol eder ve sonuca göre bir Boolean değeri verir. Eğer argüman olarak verilen sayı sonlu ise, true olur. Eğer sonsuz ise, false olur. Bu komut bir matematik işleminin kontrolü için yararlıdır. Mesela bir sayıyı sıfıra böldüğünüz zaman, sonuç sonsuza gidecektir. Yazılımı,

```
isFinite(sayı);
```

Örnek :

```
isFinite(45);
// sonuç true olur.

isFinite(Number.POSITIVE_INFINITY);
// sonuç false olur

isFinite(8763/0);
// sonuç false olur.
```

Player : Flash 5 ve sonrası

isNaN : Argüman olarak verilen ifadenin, sayı olup olmadığı kontrol eder ve bir Boolean değeri olarak geri döner. Bu ifade bir değişken, Boolean değeri veya değerlendirlebilir başka ifadeler olabilir. Daha çok matematik işlemlerindeki yanlışları kontrol etmek için kullanılır. Argüman olarak verilen ifade, bir sayı ise, false olarak geri döner. Sayı değilse, true olarak geri döner. Yazılımı,

```
isNan(ifade);
```

Örnek :

```
isNaN("Merhaba");
// sonuç, true olur.

isNaN(90);
// sonuç, false olur.

isNaN(Number.NEGATIVE_INFINITY);
// sonuç, false olur.
```

Player : Flash 5 ve sonrası

Key (Object) : Kullanıcı ile Program arasında etkileşimi sağlamak için en özel nesnedir. Standart bir klavye ile, kullanıcının programla arasında bir arayüz oluşturur. En çok, oyunlar için kullanılır. Belirtilen bir tuşa basıldığı an, tanımlamış olan işler icra edilir. Dört adet metodu vardır. Yazılımı,

```
Key.metod(tuş kodu);
```

Örnek :

```
onClipEvent (enterFrame) {
      if(Key.isDown(Key.RIGHT)) {
            setProperty ("", _x, _x+10);
      }
}

// bir movie clip'e yukarıdaki aksiyonları verin ve
// animasyonu çalıştırın.
// Klavyeden, sağ ok tuşuna her basışınızda clip ilerler.
```

Programa standart olarak, 18 adet tuş özelliği tanıtılmış. Ancak bu tuşların dışındaki tuşlar için, kod yazmanız gerekir. Bu kodlara klavye tuş kodları bölümünden ulaşabilirsiniz. Standart tuşlar aşağıdaki gibidir.

- Key.BACKSPACE : Klavyedeki geri al tuşunu simgeler.
- Key.CAPSLOCK : Klavyedeki büyük/küçük harf kilit tuşunu simgeler.
- Key.CONTROL : Klavyedeki CTRL tuşunu simgeler.
- Key.DELETEKEY: Klavyedeki DEL tuşunu simgeler.
- Key.DOWN : Klavyedeki aşağı ok tuşunu simgeler.
- Key.END : Klavyedeki END tuşunu simgeler.
- Key.ENTER : Klavyedeki ENTER/RETURN geri tuşunu simgeler.
- Key.ESCAPE : Klavyedeki ESC tuşunu simgeler.
- Key.HOME : Klavyedeki HOME tuşunu simgeler.
- Key.INSERT : Klavyedeki INSERT/INS tuşunu simgeler.
- Key.LEFT : Klavyedeki sol ok tuşunu simgeler.
- Key.PGDN : Klavyedeki PAGE DOWN tuşunu simgeler.
- Key.PGUP : Klavyedeki PAGE UP tuşunu simgeler.
- Key.RIGHT : Klavyedeki sağ ok tuşunu simgeler.
- Key.SHIFT : Klavyedeki SHIFT tuşunu simgeler.
- Key.SPACE : Klavyedeki boşluk tuşunu simgeler.
- Key.TAB : Klavyedeki TAB tuşunu simgeler.
- Key.UP : Klavyedeki yukarı ok tuşunu simgeler.

Player : Flash 5 ve sonrası

Key.getAscii : Klavyede basılan en son tuşun, ASCII kodunu verir. Yazılımı,

```
Key.getAscii();
```

Örnek :

```
onClipEvent (enterFrame) {
      if(Key.isDown(Key.getCode())) {
            trace (Key.getAscii());
      }
}

// Bir movie clip yapın ve yukarıdaki aksiyonları yazın.
// Programı çalıştırıp, herhangi bir tuşa basın.
// Her bastığınız tuşun ASCII kodunu göreceksiniz.
```

Player : Flash 5 ve sonrası

Key.getCode : Klavyeden basılan en son tuşun, ActionScript'in tanıdığı şekilde kodunu verir. Yazılımı,

```
Key.getCode();
```

Örnek :

```
onClipEvent (enterFrame) {
      if(Key.isDown(Key.getCode())) {
            trace (Key.getCode());
      }
}
```

```
// Bir movie clip yapın ve yukarıdaki aksiyonları yazın.
// Programı çalıştırıp, herhangi bir tuşa basın.
// Her bastığınız tuşun ActionScript'in tanıdığı
// kodu göreceksiniz.
```

Player : Flash 5 ve sonrası

Key.isDown : Argüman olarak belirtilen tuşun basılıp basılmadığını kontrol eder. Eğer basılmışsa, true olarak geri döner. Basılmamışsa, false olarak geri döner. Argüman olarak, Key nesnesinin 18 özelliğinden biri girilebileceği gibi, ActionScript'in tanıdığı kodları da girebilirsiniz. Machintosh'taki CAPSLOCK ve NUMLOCK tuşları aynıdır. Yazılımı,

```
Key.isDown(tuş kodu);
```

Örnek :

```
onClipEvent (enterFrame) {
      if(Key.isDown(39)) {
            setProperty("", _alpha, _alpha-10);
      }
      if(Key.isDown(37)) {
            setProperty("", _alpha, _alpha+10);
      }
}

// Bir movie clip yapın ve yukarıdaki aksiyonları yazın.
// Programı çalıştırıp, sağ ve sol oklarına basın.
// Her bastığınızda, movie clip'in, alpha değeri değişir.
```

Player : Flash 5 ve sonrası

Key.isToggled : Klavyedeki CAPSLOCK ve NUMLOCK tuşlarının basılı olup olmadığını kontrol eder. Eğer aktif ise, yani kilitlenmiş ise, true olarak geri döner. Aksi halde false olarak geri döner. Argüman olarak, iki kod girilebilir. CAPSLOCK için "20", NUMLOCK için ise "144" girilebilir. Yazılımı,

```
Key.isToggled(20 veya 144);
```

Örnek :

```
if (Key.isToggled(20)){
      trace("CAPLOCK aktif halde");
} else{
      trace("CAPLOCK aktif değil");
}
if (Key.isToggled(144)){
      trace("NUMLOCK aktif halde");
} else{
      trace("NUMLOCK aktif değil");
}

// İlk frame'e yukarıdaki kodları yazın ve çalıştırın.
// Her iki anahtarın, aktif olup olmadığını göreceksiniz.
```

Player : Flash 5 ve sonrası

le : İngilizce **l**ess than or **e**qual to (küçük veya eşit) kelimelerinin kısaltılmışıdır. Birinci ifadenin, ikinci ifadeden küçük olup olmadığını veya birbirine eşit olup olmadıklarını sınar. Bu ifadeler sayı, string veya değişken olabilirler. Sınamanın sonucu doğru ise, true değerini verir. Aksi halde false değeri verir. Yazılımı,

```
ifade_1 le ifade_2;
```

Örnek :

```
if (deger_1 le deger_2){
      trace("deger_2 buyuk");
} else{
      trace("deger_1 buyuk");
}

// sınamanın sonucuna göre, çıktı alınır
```

Player : Flash 4 ve sonrası. Flash 5'te bu komutun yerine "<=" işareti kullanılır.

length : Bir string fonksiyonudur. Argüman olarak verilen string veya değişken adının kaç karakterden oluştuğunu belirler. Flash 5 ile, her nesneye ait bir özellik olarak eklendiğinden ayrıca kullanılmasına gerek yoktur. Yazılımı,

```
length(string veya değişken adı);
```

Örnek :

```
length("Merhaba Dünya");

// sonuc, 13 olacaktır.
```

Player : Flash 4 ve sonrası

_level : Birden fazla animasyon iç içe kullanıldığı zaman, her biri değişik seviyelerde yer alırlar. Normal şartlarda, bir animasyonu oluşturduğunuz zaman, seviyesi 0 olacaktır (`_level0`). Bu temel animasyondur. Daha sonra `loadMovie` ile birden fazla animasyonu, temel animasyonun üstüne ekleyebilirsiniz. Eklenen bu animasyonların seviyeleri, eklenme sırasına göre artacaktır (`_level1`, `_level2....levelN`). Ancak sonradan eklenen tüm animasyonlar, temel animasyonun arkaplan rengini, saniyedeki frame gösterme hızını ve frame ölçülerini alacaktır. Herhangi bir animayona atıfta bulunacağınız zaman, bu animasyona ait seviye ile birlikte yazmanız gerekir. Yazılımı,

```
_levelN;
```

Örnek :

```
_level0.gotoAndPlay(10);

// temel animasyondaki ana zaman ekseni üzerinde 10. frame
// gidip oynanması emredilmiş.

_level3.stop();

// Eklenen 3. animasyonun durması emredilmiş.
```

Player : Flash 4 ve sonrası

loadMovie : Normalde, Flash Player tek bir animasyonu oynatır ve kapanır. Bu komut ile, oynamakta olan animasyona birden fazla swf dosyası eklenebilir. Böylece aynı animasyon üzerinde, bir kaç tane animasyon oynatılabilir. Eklenen her animasyonun farklı bir `level`'i vardır. Eğer mevcut olan bir `level`'in üzerine yeni bir animasyon yüklenirse, eski animasyonun silinir ve yeni eklenen bu animasyonun yerine geçer. Eğer yeni eklenen animasyon, `level0` olarak tanımlanırsa, diğer `level`'de bulunan tüm animasyonlar silinmiş olur ve yeni eklenen animasyon temel animasyon olur. Üç adet argümanı vardır. İlk argüman, yüklenecek olan swf dosyasının, URL adresidir. URL adresi, aynı veya buna bağlı bir klasör içinde kalan, klasör içinde olmalıdır. İkinci argüman opsiyoneldir ve yüklenecek olan animasyonun `level`'ini belirtir. Eğer mevcut olan bir animasyonun `level`'ine yüklenirse, eski animasyon silinir, yerine yenisi geçer. Son argüman ise, yüklenecek olan animasyona bilgi gönderildiği zaman kullanılır. GET veya POST metodlarından biri ile, bilgiler gönderilir. Yazılımı,

```
loadMovie(URL, [hedef,[metod]]);
```

Örnek : Önceden oluşturduğunuz bir animasyonu hedef olarak gösterip, yeni animasyona yükleyin.

```
on (press){
      loadMovie ( "testMovie2.swf", _level1);
}

// Animasyonu, testMovie2 ile aynı yere export edin.
// Buradaki swf dosyasını çalıştırın ve düğmeye basın.
```

Player : Flash 3 ve sonrası

loadVariables : Flash dışındaki bir dosyaya veya programa ulaşmak için kullanılır. Bu komut kullanıldığı zaman, hem Flash içindeki değişkenler, hem de argüman olarak belirtilen dosyadaki değişkenler, ortak olarak kullanılabilir. Flash'ın bu değerleri kullanabilmesi için, ASCII formatında olması gerekir. Diğer programlar ise Flash'taki değişkenleri, form nesnesinden veya QueryString nesnesinden alabilirler. Üç adet argümanı vardır. İlk argüman, atıfta bulunulan adres ve dosya ismidir. Bu adres, bilgisayarınızda çalışacaksa, bilgisayarınızdaki tam adres olmalıdır. Eğer internet üzerinde çalışacaksa, sayfanıza ev sahipliği yapan server'ın adresi olmalıdır. İkinci argüman, bilgilerin nereye gönderileceğini belirten hedefi tanımlar. Eğer animasyona başka bir swf dosyası eklenmişse, derinlik seviyesini belirten adres yazılmalıdır (`_level2` gibi). Son argüman ise, bilgilerin hangi metodla gönderileceğini belirler. Bu metodlar "GET" veya "POST" olmalıdır. Yazılımı,

```
loadVariables(adres, hedef, metod);
```

Örnek :

```
loadVariables("http://www.webteknikleri.com/metin.txt", "");

// Bu sitedeki, metin.txt dosyası içindeki tüm değerler alınır.

loadVariables("yazdir.php?sonuc="+sonuc, "_level3", "POST");

// Flash içindeki sonuc adlı değişken, _level3'te duran
// movie clip'e, yazdir.php dosyası ile POST metoduyla
// aktarılır.
```

Player : Flash 4 ve sonrası

lt : İngilizce **l**ess **t**han (küçüktür) kelimelerinin kısaltılmış halidir. Birinci ifadenin, ikinci ifadeden küçük olup olmadığını sınar. Bu ifadeler sayı, string veya değişken olabilir. Sınamanın sonucu doğru ise, true değerini verir. Aksi halde false değeri verir. Yazılımı,

```
ifade_1 lt ifade_2;
```

Örnek :

```
if (deger_1 lt deger_2){
      trace("deger_1 küçük");
} else{
      trace("deger_2 küçük");
}

// sınamanın sonucuna göre, çıktı alınır
```

Player : Flash 4 ve sonrası. Flash 5'te, bu ifadenin yerine "<" işareti kullanılır.

Math (Object) : Matematik işlemleri için kulanılan nesnedir. 18 adet metodu ve 8 adet sabiti vardır. Metod yada sabitleri kullanırken, `Math` ifadesi, yazılmak zorundadır. `Math` nesnesinin metodlarından birini kullanırken, argüman olarak sayı girebileceğimiz gibi, denklem de girebiliriz. `Math` nesnesinin çoğu metodu radyan üzerinden işlem yapar. Radyan olarak bulunan sonucu dereceye çevirmek için, sonucu `180/Math.PI` ile çarpamamız gerekir. Yazılımı,

```
Math.metod(sayı veya denklem);
Math.sabit;
```

Örnek :

```
Math.SIN(Math.PI/90*60)+Math.random();

// 1.02211664781709 gibi bir sonuc olacaktır.
```

Player : Flash 5 ve sonrası. Ancak Flash 4'de, yaklaşık olarak hesaplama yapar.

Math.abs : Argüman olarak verilen denklem veya sayının, mutlak değerini bulur. Sonuç negatif olsa bile, pozitif olarak verir. Yazılımı,

```
Math.abs(sayı veya denklem);
```

Örnek :

```
gemiX = getProperty("gemi", _x);
hedefX = getProperty("hedef", _x);
mesafe = Math.abs(Number(gemiX)-Number(hedefX));

// sonuç ne olursa olsun, pozitif değer verir.
```

Player : Flash 5 ve sonrası

Math.acos : Argüman olarak verilen sayının (-1.0 ile 1.0 arasında) ters cosinisünü, radyan olarak bulur. Yazılımı,

```
Math.acos(sayı);
```

Örnek : kenarları 6, 8 ve 10 olan bir üçgenin açısını bulmak için, komşu kenarın hipotenüse oranı alınır (cos X = 8/10). X açısı ters cosininüs ile bulunur.

```
aci = Math.acos(0.8);

// acı = 0.643501108793284 olur. Bu değer radyan cinsindedir.

aciDerece = (180*aci)/Math.PI

// aciDerece = 36.869897645844 olacaktır.
```

Player : Flash 5 ve sonrası

Math.asin : Argüman olarak verilen sayının (-1.0 ile 1.0 arasında) ters sinisünü, radyan olarak bulur. Yazılımı,

```
Math.asin(sayı);
```

Örnek : kenarları 6, 8 ve 10 olan bir üçgenin açısını bulmak için, karşı kenarın hipotenüse oranı alınır (sin X = 6/10). X açısı ters sinüs ile bulunur.

```
aci = Math.asin(0.6);

// acı = 0.643501108793284 olur. Bu değer radyan cinsindedir.

aciDerece = (180*aci)/Math.PI

// aciDerece = 36.869897645844 olacaktır.
```

Player : Flash 5 ve sonrası

Math.atan : Argüman olarak verilen sayının veya denklemin ters tanjantını, radyan olarak bulur. Yazılımı,

```
Math.atan(sayı veya denklem);
```

Örnek : kenarları 6, 8 ve 10 olan bir üçgenin açısını bulmak için, karşı kenarın komşu kenara oranı alınır (cos X = 6/8). X açısı ters tanjant ile bulunur.

```
aci = Math.atan(0.75);

// acı = 0.643501108793284 olur. Bu değer radyan cinsindedir.

aciDerece = (180*aci)/Math.PI

// aciDerece = 36.869897645844 olacaktır.
```

Player : Flash 5 ve sonrası

Math.atan2 : Argüman olarak verilen koordinat noktasının x eksenine göre açısını, radyan olarak bulur. Argüman olarak verilen koordinat noktası, x ve y koordinatlarındaki herhangi bir noktadır. Yazılımı,

```
Math.atan2(x koordinatı, y koordinatı);
```

Örnek : Dynamic text ile bir yazı kutusu (adı, aci olsun) yapın ve klavyeden F8 tuşuna basarak, movie clip olarak tanımlayın. Bu movie clip'e aşağıdaki kodları yazın.

```
onClipEvent(mouseMove){
      x=Math.atan2(_xmouse,_ymouse);
      aci=(180*x)/Math.PI
}

// programı çalıştırın ve fare işaretçisini hareket ettirin
// fare, her kımıldadığında, açı değeri değişecektir.
```

Player : Flash 5 ve sonrası

Math.ceil : Argüman olarak verilen sayıyı veya denklemi, yukarıya doğru yuvarlatır ve tamsayı olarak sonucu verir. Yazılımı,

```
Math.ceil(sayı veya denklem);
```

Örnek :

```
trace(Math.ceil(0.215);
// sonuc, 1 olacaktır.
```

Player : Flash 5 ve sonrası

Math.cos : Argüman olarak verilen sayının veya denklemin cosinisünü, radyan olarak bulur. Yazılımı,

```
Math.cos(sayı veya denklem);
```

Örnek :

```
Math.cos(Math.random());

// 0.826143573258944 gibi sonuçlar verir.
// tüm sonuçlar -1.0 ile 1.0 arasında olacaktır.
```

Player : Flash 5 ve sonrası

Math.E : Matematikte kullanılan, doğal logaritmanın tabanı olan Euler sabitidir. Yaklaşık değeri 2.71828182845905'dir. Yazılımı,

```
Math.E;
```

Örnek :

```
Math.E*Math.random();

// 0.942162371441323 gibi sonuçlar verir.
```

Player : Flash 5 ve sonrası

Math.exp : Matematikte kullanılan, doğal logaritmanın tabanı olan Euler sabitinin, argüman olarak verilen sayı kadar üstünü alır. Yani $2.71828^{sayı}$ olur. Yazılımı,

```
Math.exp(sayı veya denklem);
```

Örnek :

```
Math.exp(2);

// 7.38905609893065 sonucunu verir (2.71828^2).
```

Player : Flash 5 ve sonrası

Math.floor : Argüman olarak verilen sayıyı veya denklemi, aşağıya doğru yuvarlatır ve tamsayı olarak sonucu verir. Yazılımı,

```
Math.floor(sayı veya denklem);
```

Örnek :

```
trace(Math.floor(25.915);
// sonuc, 25 olacaktır.
```

Player : Flash 5 ve sonrası

Math.log : Matematikte kullanılan, doğal logaritma işlemlerini yapar. Logaritmanın tabanı 2.71828182845905 sayısı yani Euler sabitidir. Yazılımı,

```
Math.log(sayı veya denklem);
```

Örnek :

```
Math.log(10);

// 2.30258509299405 sonucunu verir.
```

Player : Flash 5 ve sonrası

Math.LOG2E : Matematikte kullanılan, logaritmanın, 2 tabanına göre Euler sabiti değerini verir. Yaklaşık değeri 0.693147180559945'dir. Yazılımı,

```
Math.LOG2E;
```

Örnek :

```
Math.LOG2E*Math.random();

// 0.0869678429907105 gibi sonuçlar verir.
```

Player : Flash 5 ve sonrası

Math.LOG10E : Matematikte kullanılan, logaritmanın, 10 tabanına göre Euler sabiti değerini verir. Yaklaşık değeri 0.434294481903252'dir. Yazılımı,

```
Math.LOG10E;
```

Örnek :

```
Math.LOG10E*Math.random();

// 0.33055295969478 gibi sonuçlar verir.
```

Player : Flash 5 ve sonrası

Math.LN2 : Matematikte kullanılan, logaritmanın, E tabanına göre 2 sayısının değerini verir (Math.log(2);). Yaklaşık değeri 0.693147180559945'dir. Yazılımı,

```
Math.LN2;
```

Örnek :

```
Math.LN2*Math.random();

// 0.39437395853729 gibi sonuçlar verir.
```

Player : Flash 5 ve sonrası

Math.LN10 : Matematikte kullanılan, logaritmanın, E tabanına göre 10 sayısının değerini verir (Math.log(10);). Yaklaşık değeri 2.30258509299405'dir. Yazılımı,

```
Math.LN10;
```

Örnek :

```
Math.LN10*Math.random();

// 0.429895573798239 gibi sonuçlar verir.
```

Player : Flash 5 ve sonrası

Math.max : Argüman olarak verilen iki sayıdan veya denklemden, büyük olanını verir. Yazılımı,

```
Math.max(sayı_1, sayı_2);
```

Örnek :

```
sayi_1 = 255;
sayi_2 = 300;

buyukSayi = Math.max(sayi_1, sayi_2);

// buyukSayi = 300 olacaktır.
```

Player : Flash 5 ve sonrası

Math.min : Argüman olarak verilen iki sayıdan veya denklemden, küçük olanını verir. Yazılımı,

```
Math.min(sayı_1, sayı_2);
```

Örnek :

```
sayi_1 = 255;
```

```
sayi_2 = 300;

kucukSayi = Math.min(sayi_1, sayi_2);

// kucukSayi = 255 olacaktır.
```

Player : Flash 5 ve sonrası

Math.PI : Matematikte kullanılan, pi sayısının değerini verir. Yaklaşık değeri 3.14159265358979'dur. Yazılımı,

```
Math.PI;
```

Örnek :

```
cap = 100;
cevre = Math.PI*2*cap;

// cevre = 628.318530717959 olacaktır.
```

Player : Flash 5 ve sonrası

Math.pow : Argüman olarak verilen iki sayı verilir. İlk sayı üzeri ikinci sayı olarak işlem yapar ve sonucu verir. Yani x üzeri y sayısını hesaplar. Yazılımı,

```
Math.pow(sayı_1, sayı_2);
```

Örnek :

```
sayi_1 = 5;
sayi_2 = 3;

sayi_3 = Math.pow(sayi_1, sayi_2);

// sayi = 125 olacaktır (3^5 = 125).
```

Player : Flash 5 ve sonrası

Math.random : 0 ile 1 arasında rastgele bir sayı üretir. Bu sayıyı, bilgisayarın o anki saatini karmaşık formüllerden geçirerek bulur. Yazılımı,

```
Math.random();
```

Örnek :

```
on (press){
      x = Math.random()*10;
      trace(x);
}

// düğmeye her basışınızda değişik sayılar görünecektir.
```

Player : Flash 5 ve sonrası

Math.round : Argüman olarak verilen sayıya veya denklemin sonucuna göre, ondalık kısımdaki sayıya bağlı olarak aşağıya veya yukarıya doğru yuvarlatır ve tamsayı olarak sonucu verir. Yazılımı,

```
Math.round(sayı veya denklem);
```

Örnek :

```
trace(Math.round(25.915);
// sonuc, 25 olacaktır.

trace(Math.round(25.4915);
// sonuc, 24 olacaktır.

trace(Math.round(25.5);
// sonuc, 25 olacaktır.
```

Player : Flash 5 ve sonrası

Math.sin : Argüman olarak verilen sayının veya denklemin sinisünü, radyan olarak bulur. Yazılımı,

```
Math.sin(sayı veya denklem);
```

Örnek :

```
Math.sin(Math.random());

// 0.56213325842 gibi sonuçlar verir.
```

Player : Flash 5 ve sonrası

Math.sqrt : Argüman olarak verilen sayının veya denklemin karekökünü bulur. Yazılımı,

```
Math.sqrt(sayı veya denklem);
```

Örnek :

```
Math.sqrt(121);

// sonuç 11 olacaktır.
```

Player : Flash 5 ve sonrası

Math.SQRT1_2 : ½ yani 0.5 sayısının karekök değerini verir. Yaklaşık değeri 0.707106781186548'dir. Yazılımı,

```
Math.SQRT1_2;
```

Örnek :

```
x = Math.SQRT1_2*Math.random();

// x = 0.256214847959 gibi sonuçlar verir.
```

Player : Flash 5 ve sonrası

Math.SQRT2 : 2 sayısının karekök değerini verir. Yaklaşık değeri 1.4142135623731'dir. Yazılımı,

```
Math.SQRT2;
```

Örnek :

```
x = Math.SQRT2*Math.random();

// x = 1.17616419743478 gibi sonuçlar verir.
```

Player : Flash 5 ve sonrası

Math.tan : Argüman olarak verilen sayının veya denklemin tanjantını, radyan olarak bulur. Yazılımı,

```
Math.tan(sayı veya denklem);
```

Örnek :

```
Math.tan(Math.random());

// 0.166644070268393 gibi sonuçlar verir.
```

Player : Flash 5 ve sonrası

maxscroll : Çok satırlı bir metin kutusunda, aşağı veya yukarı kaydırabileceğiniz maksimum satırı, metin kutusunun yüksekliğine bağlı olarak verir. Yani metin kutusunun dışında kalan satırların adedini verir. Daha çok `scroll` metodu için kullanılır. Bu metod ile alınacak değer daima normal değerden 1 fazladır. Çünkü, `scroll` metodunun varsayılan değeri 1'dir. Sadece okunabilir bir değer verir. Bu komutu kullanarak herhangi bir ayarlama yapılmaz. Yazılımı,

```
metin_Kutusu_adı.maxscroll
```

Örnek :

```
// sahneye bir dynamic textbox yapın.
// İlk frame şu kodları yazın ("\r" ifadesi satır başıdır).

metin = "Hakkı\rMustafa\rOsman\rCüneyd\rMemduh"

// bir düğme yapın ve şu kodları yazın.

On (press){
trace(_root.metin.maxscroll);
}

// OutPut penceresinde, 5 rakamı görünecektir.
// Aslında 4 adet satır dışarıda kalmıştır.
```

Player : Flash 4 ve sonrası

mbchr : Argüman olarak verilen sayıyı ASCII kodlarına çevirir. Yazılımı,

```
mbchr(sayı);
```

Örnek :

```
for (i=50; i<100; i++){
      trace(mbchr(i));
}

// 50'den 100'e kadar olan ASCII kodlarını yazdırır.
```

Player : Flash 4 ve sonrası. Flash 5'te bu komutun yerine "`String.fromCharCode`" kullanılır.

mblength : Argüman olarak verilen string'in karakter sayısını verir. Yazılımı,

```
mblength(string);
```

Örnek :

```
sayi = mblength("Merhaba Yaşlı Dünya");

// sayi = 19 olacaktır.
```

Player : Flash 4 ve sonrası. Flash 5'te bu komutun yerine String fonksiyonları kullanılır.

mbord : Argüman olarak verilen karakeri, ASCII kod numarasına çevirir. Yazılımı,

```
mbord(karakter);
```

Örnek :

```
dizi = new Array("a","b","c","d","e","f","g","h","i");
for (i=0; i<dizi.length; i++){
      trace(mbord(dizi[i]));
}

// 97,98,99,100,101,102,103,104,105 yazdırır.
```

Player : Flash 4 ve sonrası. Flash 5'te bu komutun yerine "`String.charCodeAt`" kullanılır.

mbsubstring : Argüman olarak verilen bir string içinden, yine argüman olarak verilen yerden başlayıp belirtilen sayı kadar olan karakeri alıp yeni bir string oluşturur. Yazılımı,

```
mbsubstring(string, başlangıç, karakter sayısı);
```

Örnek :

```
metin = "Sana da merhaba insanoğlu");
yeniMetin = mbsubstring(metin,9,7)

// yeniMetin, "merhaba" olacaktır.
```

Player : Flash 4 ve sonrası. Flash 5'te bu komutun yerine "`String.substr`" kullanılır.

Mouse (Object) : Fare işaretçisi saklayıp göstermek için kullanılan iki adet metodu vardır. Değişik fare işaretçisi kullanmak için sıkça kullanılır.

Player : Flash 5 ve sonrası

Mouse.hide : İşletim sisteminde, tanımlı olan fare işaretçisini saklamak için kullanılır. Bu komut nerede kullanılırsa kullanılsın, o anda fare işaretçisi görünmez. Yazılımı,

```
Mouse.hide();
```

Örnek :

```
onClipEvent (enterFrame){
      Mouse.hide();
}
```

Player : Flash 5 ve sonrası

Mouse.show : İşletim sistemine ait olan fare işaretçisi saklanmış ise, tekrar gösterilmesine izin verir. Dikkat edilmesi gereken bir nokta vardır. Başka bir nesneyi fare işaretçisi olarak tanımlanmış ise, bu komut ile, aynı zamanda tanımlanmış olan nesneyi de kaldırmak lazımdır. Aksi takdirde, iki işaretçi üst üste görünecektir. Yazılımı,

```
Mouse.show();
```

Örnek :

```
on (press){
      stopDrag();
      setProperty("fare", _visible, false);
      Mouse.show();
}

// ik önce, belirlemiş olduğumuz fare işaretçisini kaldırıyoruz
// sonra sistemin fare işaretçisini gösteriyoruz
```

Player : Flash 5 ve sonrası

MovieClip (Object) : Sahnedeki herhangi bir nesneye, davranış biçimi olarak movie clip verildiği zaman, kendisine ait 22 metoddan yararlanma imkanı doğar. Bir movie clip'e ait üç adet isim vardır. Bunlardan birincisi, kütüphanedeki ismidir. İkincisi, durum adıdır (Instance name). Metodların çoğunda, durum adı kullanılır. Bir movie clip'e durum adı vermek için, Instance Panel'deki Name kutusuna ismini yazmak gerekir. Son olarak, kimlik adı vardır. Bu isim de, `attachMovie` metodundan yararlanmak için kullanılır. Bu adı tanımlamak için, kütüphanedeki movie clip'in üzerinde sağ tıklayıp, Linkage'i seçin. Açılan pencerede, Linkage kısmındaki, Export this symbol'u seçili yapın ve üstündeki Identifier kutusuna kimlik ismini yazın. Genel olarak yazılımı,

```
movieClip.metod(argümanlar);
```

Örnek :

```
takvimFilmi.play();
```

Player : Flash 5 ve sonrası

MovieClip.attachMovie : Kütüphanede duran bir movie clip'ten bir kopya almak için kullanılır. Bunun için kütüphanedeki movie clip'e kimlik ismi verilmesi gerekir. Üç adet argümanı vardır. İlk argüman, kimlik adıdır (bknz. MovieClip Object). İkinci argüman, kopyalanacak olan movie clip'in yeni ismidir. Son agüman ise, derinlik seviyesidir. Kopyalanacak olan her movie clip, farklı derinlik seviyelerinde olmalıdır. Aynı derinlik seviyesi verilirse, eskisini silip yenisini kopyalar. Yazılımı,

```
movieClip.attachMovie(kimlik_adı, yeni_isim, seviye);
```

Örnek :

```
on (press) {
    _root.klip_1.attachMovie("klip_2", "klip_3" , 1);
}

// Düğmeye bastığınız an, kütüphanedeki kimlik adı, klip_2 olan
// movie clip'ten bir kopya alacak ve klip_1 adlı
// movie clip'e kopyalacak.
// Buna göre, yeni movie clip'in adı, klip_3 ve seviyesi 1'dir
```

Player : Flash 5 ve sonrası

MovieClip.duplicateMovieClip : Sahnede oynamakta olan bir movie clip'ten kopya alır ve sahneye yapıştırır. Bu yöntem ile kopyalanan movie clip'ler daima 1. frame'den itibaren oynamaya başlarlar. Kopyasını aldığınız movie clip'teki değişkenler, yeni movie clip'lere aktarılmaz. İki adet argümanı vardır. İlki, yeni kopyanın ismidir. İkincisi ise, derinlik seviyesidir. Yazılımı,

```
movieClip.duplicateMovieClip(yeni_isim, derinlik);
```

Örnek :

```
on (press){
      _root.klip.duplicateMovieClip("klipYeni",2);
}

// Oynamakta olan bir movie clip'ten kopya alır.
```

Player : Flash 5 ve sonrası

MovieClip.getBounds : Bir nesnenin şekli nasıl olursa olsun, onu çevreleyen bir dörtgen içindedir. Yuvarlak şekilli bir movie clip oluşturduğunuz zaman, bu movie clip'e bir kez tıkladığınız zaman, çevresinde mavi ince bir dörtgen gürürsünüz. Bu dörgen şekil, nesnenin, x-eksenine ve y-eksenine göre referans noktaları verir. ActionScript'te bu dörtgene bounds adı verilir. Bu metod ile, bir movie clip'in başka bir movie clip'e göre olan referans noktaları alınır. Alınan bilgi, başka bir nesneye aktarılır. Bu nesnenin, `xMax`, `xMin`, `yMax` ve `yMin` özellikleri ile, ilgili referans noktalarına ulaşılabilir. Bir adet argümanı vardır. Bu argümana sadece "`-root`" yazarsanız, sahnenin sol üst noktasına göre referans noktaları verilir. Bir movie clip'e göre refrans noktası almak için, seviyesi ile birlikte, movie clip'in durum adı yazılmalıdır. Yazılımı,

```
movieClip.getBounds(referans_nokta);
```

Örnek :

```
// İlk önce bir movie clip oluşturun. Sonra
// bir düğme yapın ve düğmeye aşağıdaki aksiyonları yazın

on(release)
```

```
{
jNesne = new Object();
jNesne = this.getBounds(_root.klip);
trace("xmin:"+ jNesne.xMin);
trace("xmax:"+ jNesne.xMax);
trace("ymin:"+ jNesne.yMin);
trace("ymax:"+ jNesne.yMax);
}

// Düğme ile movie clip'in merkez noktası arasındaki değerler
// aşağıdaki gibi olabilir.
// xmin:-63.75
// xmax:15.15
// ymin:-63.85
// ymax:15.1
```

Örneğe göre, düğmenin en sol noktası ile movie clip'in merkez noktası arasındaki yatay mesafe : -63.75'dir. Yani, düğmenin sol kenarı, movie clip'in merkezinden –63.75 piksel (birim olarak ne seçili ise) solundadır.

Player : Flash 5 ve sonrası

MovieClip.getBytesLoaded : Bir movie clip'in yüklenmiş olan byte miktarını verir. Dışarıdan eklenen tüm movie clip'ler için geçerlidir. Animasyonun kendi içindeki movie clip'ler otomatik olarak yüklenir. Diğer bir motod olan `getBytesTotal` ile karşılaştırılarak, movie clip'in byte miktarını yüzde cinsinden bulabiliriz. Kendi bilgisayarınızda test etmeye kalkıştığınızda, daima toplam miktarı alırsınız. Bu, bilgisayarınızın, internet ortamına göre çok hızlı çalışmasından kaynaklanır. Yazılımı,

```
movieClip.getBytesLoaded();
```

Örnek :

```
yuklenen = int(_root.klip.getBytesLoaded());

// klip adlı movie clip'in yüklenmiş olan byte miktarını verir
```

Player : Flash 5 ve sonrası

MovieClip.getBytesTotal : Dışarıdan eklemek istediğiniz bir movie clip'in toplam byte miktarını verir. Diğer bir metod olan `getBytesLoaded` ile birlikte kullanılırsa, yüklenen byte miktarını, yüzde cinsinden bulabilirsiniz. Yazılımı,

```
movieClip.getBytesTotal();
```

Örnek :

```
onClipEvent (enterFrame) {
    yuklenen = int(_root.getBytesLoaded()/1000)+" kB";
    kalan = int(_root.getBytesTotal()/1000-
      _root.getBytesLoaded()/1000)+" kB";
    yuzde = int(100* root.getBytesLoaded()/
      _root.getBytesTotal())+"%";
    _width = int(100*_root.getBytesLoaded()/
      _root.getBytesTotal());
}

// Basit bir preloader örneği.
```

```
// Bir dörtgen çizin ve movie clip olarak tanımlayın.
// Yukarıdaki kodları yazın.
// Değişkenleri bir text kutusnda yazdırabilirsiniz.
```

Player : Flash 5 ve sonrası

MovieClip.getURL : Argüman olarak belirtilen bir URL adresine gider veya bu adrese bilgi gönderir. Üç adet argümanı vardır. İlk argüman, gidilecek olan URL adresidir. İkinci argüman, ilgili URL adresini, hangi pencerede açacağıdır.

- _self : Mevcut pencerenin, mevcut frame'inde açar.
- _blank : Yeni bir pencerede açar.
- _parent : Mevcut açık pencerede açar.
- _top : Mevcut pencere içinde, üst seviyede açar.

Son argüman ise, gidilecek olan URL adresine bilgi gönderileceği zaman kullanılır. Eğer bu bilgiler sınırlı sayıda ise, GET metodu ile bir yumak halinde gider. Gönderelecek bilgiler, POST metodu ile sınırsız sayıda ve HTTP başlıklarına göre ayrılmış olarak gider. Yazılımı,

```
movieClip.getURL(URL adresi, [pencere, [metod]]);
```

Örnek :

```
on (press) {
      klip.getURL("http://www.webteknikleri.com", "_blank");
}

// movie clip yüklendiği anda, ilgili adresi
// başka bir pencerede açacaktır.
```

Player : Flash 5 ve sonrası.

MovieClip.globalToLocal : Jenerik olarak oluşturulan `point` nesnesine atanan, koordinat bilgilerini, içinde çalışan movie clip'in koordinat bilgisi olarak belirler. Diğer bir metod olan `getBounds` ile koordinat bilgileri, jenerik bir nesneye atanır. Bu nesne, x ve y eksenindeki noktaları alıp belirtilen movie clip'in x ve y koordinat bilgisi olarak belirler. Yazılımı,

```
movieClip.globalToLocal(referans nesnesi);
```

Örnek :

```
onClipEvent (enterFrame) {
      point1 = new object();
      point1.x = _root._xmouse;
      point1.y = _root._ymouse;
      _root.out13 = point1.x+" === "+point1.y;
      _root.out1 = _root._xmouse+" === "+_root._ymouse;
      localToGlobal(point1);
      _root.out12 = point1.x+" === "+point1.y;
      updateAfterEvent();
}

// Yukarıdaki kodları bir movie clip'e yazın.

onClipEvent (enterFrame) {
    globalToLocal(point1,point2);
```

```
      _root.cizgi._x = _root.kutu.point1.x;
      _root.cizgi._y = _root.kutu.point1.y;
      updateAfterEvent();
}

// Bir çizgi çizip, movie clip olarak tanımladıktan sonra
// yukarıdaki kodları yazın ve çalıştırın.
// Fare işaretçisini hareketi, çizginin yerini belirleyecek.
```

Player : Flash 5 ve sonrası

MovieClip.gotoAndPlay : Durum adını önüne yazdığınız movie clip'i kendi içindeki beliretilen yere gönderir ve oynatamaya devam eder. Argüman olarak frame numarası yada, etiket ismi (label) yazabilirsiniz. Yazılımı,

```
movieClip.gotoAndPlay(frame);
```

Örnek :

```
on (keypress "1"){
      klip.gotoAndPlay(10);
}
on (keypress "2"){
      klip.gotoAndPlay(20);
}
on (keypress "3"){
      klip.gotoAndPlay(30);
}
on (keypress "4"){
      klip.gotoAndPlay(40);
}

// hangi numaraya basılırsa, ilgili frame gidip oynayacaktır
```

Player : Flash 5 ve sonrası

MovieClip.gotoAndStop : Durum adını önüne yazdığınız movie clip'i kendi içindeki beliretilen yere gönderir ve durdurur. Argüman olarak frame numarası yada, etiket ismi (label) yazabilirsiniz. Yazılımı,

```
movieClip.gotoAndStop(frame);
```

Örnek :

```
on (press){
      klip.gotoAndPlay(10);
}

// düğmeye basıldığı an, ilgili frame gidip, duracaktır.
```

Player : Flash 5 ve sonrası

MovieClip.hitTest : Bir nesnenin, belirtilen bir movie clip ile temasının olup olmadığını kontrol eder. Flash'ta, fare işaretçisinin de bir nesne olduğu düşünülürse, fare işaretçisinin de, bir movie clip'le olan teması kontrol edilebilir. Bu temas iki şekilde yapılabilir. Movie clip'in şekli nasıl olursa olsun, onu

çevreleyen bir dörtgen vardır. Teması bu dörtgene göre (false) veya sadece şeklin kendisi (true) ile kontrol ettirilebilir. Argümanları iki şekilde belirtebiliriz. Birincisi, x ve y koordinatları ve temas durumunu belirterek ki bu, daha çok fare işaretçisi için kullanılır. İkincisi ise, sadece hedef belirterek. Yazılımı,

```
movieClip.hitTest(x,y,durum);
movieClip.hitTest(hedef);
```

Örnek :

```
// Bir fare işaretçisinin, bir movie clip'le olan teması için,
// Daire şeklinde bir movie clip oluşturun
// ve aşağıdaki kodları yazın.

onClipEvent (mouseMove){
      if (hitTest( _root._xmouse, _root._ymouse, false)){
      trace("Temas Sağlandı");
}
}

// Şimdi, if satırındaki false olan ifadeyi,
// true olarak değiştirin ve farkı inceleyin.

// Hedef belirterek çarpışma testi için iki movie clip
// yapın ve birine şu kodları yazın.

onClipEvent (mouseMove){
      if(_root.klip_1, hittest(_root.klip_2)){
      trace("Temas Var");
}
}

// Diğerine ise şunları yazın.

onClipEvent(enterFrame){
      startDrag ( "", true );
}
```

Player : Flash 5 ve sonrası

MovieClip.loadMovie : Normalde, Flash Player tek bir animasyonu oynatır ve kapanır. Bu komut ile, oynamakta olan animasyona birden fazla swf dosyası eklenebilir. Böylece aynı animasyon üzerinde, bir kaç tane animasyon oynatılabilir. Eklenen her animasyonun farklı bir `level`'i vardır. Eğer mevcut olan bir `level`'in üzerine yeni bir animasyon yüklenirse, eski animasyonun silinir ve yeni eklenen bu animasyonun yerine geçer. Eğer yeni eklenen animasyon, `level0` olarak tanımlanırsa, diğer `level`'de bulunan tüm animasyonlar silinmiş olur ve yeni eklenen animasyon temel animasyon olur. Üç adet argümanı vardır. İlk argüman, yüklenecek olan swf dosyasının, URL adresidir. URL adresi, aynı veya buna bağlı bir klasör içinde kalan, klasör içinde olmalıdır. İkinci argüman opsiyoneldir ve yüklenecek olan animasyonun `level`'ini belirtir. Eğer mevcut olan bir animasyonun `level`'ine yüklenirse, eski animasyon silinir, yerine yenisi geçer. Son argüman ise, yüklenecek olan animasyona bilgi gönderildiği zaman kullanılır. GET veya POST metodlarından biri ile, bilgiler gönderilir. Yazılımı,

```
MovieClip.loadMovie(URL, [hedef,[metod]]);
```

Örnek : Önceden oluşturduğunuz bir animasyonu hedef olarak gösterip, yeni animasyona yükleyin.

```
on (press){
      klip.loadMovie ( "testMovie2.swf", _level1);
}
```

```
// Animasyonu, testMovie2 ile aynı yere export edin.
// Buradaki swf dosyasını çalıştırın ve düğmeye basın.
```

Player : Flash 3 ve sonrası

MovieClip.loadVariables : Flash dışındaki bir dosyaya veya programa ulaşmak için kullanılır. Bu komut kullanıldığı zaman, hem Flash içindeki değişkenler, hem de argüman olarak belirtilen dosyadaki değişkenler, ortak olarak kullanılabilir. Flash'ın bu değerleri kullanabilmesi için, ASCII formatında olması gerekir. Diğer programlar ise Flash'taki değişkenleri, form nesnesinden veya QueryString nesnesinden alabilirler. Üç adet argümanı vardır. İlk argüman, atıfta bulunulan adres ve dosya ismidir. Bu adres, bilgisayarınızda çalışacaksa, bilgisayarınızdaki tam adres olmalıdır. Eğer internet üzerinde çalışacaksa, sayfanıza ev sahipliği yapan server'ın adresi olmalıdır. İkinci argüman, bilgilerin nereye gönderileceğini belirten hedefi tanımlar. Eğer animasyona başka bir swf dosyası eklenmişse, derinlik seviyesini belirten adres yazılmalıdır (`_level2` gibi). Son argüman ise, bilgilerin hangi metodla gönderileceğini belirler. Bu metodlar "GET" veya "POST" olmalıdır. Yazılımı,

```
MovieClip.loadVariables(adres, hedef, metod);
```

Örnek :

```
klip.loadVariables("http://www.site.com/metin.txt", "");

// Bu sitedeki, metin.txt dosyası içindeki tüm değerler alınır.

klip.loadVariables("deger.asp?sonuc="+sonuc, "POST");

// Flash içindeki sonuc adlı değişken, movie clip'e,
// deger.asp dosyası ile POST metoduyla aktarılır.
```

Player : Flash 4 ve sonrası

MovieClip.localToGlobal : Jenerik olarak oluşturulan `point` nesnesine atanan, koordinat bilgilerini, koordinat bilgisi olarak belirler. Diğer bir metod olan `getBounds` ile koordinat bilgileri, jenerik bir nesneye atanır. Bu nesne, movie clip'in x ve y koordinat noktalarını alıp, x ve y eksenindeki koordinat bilgisi olarak belirler. Yazılımı,

```
movieClip.localToGlobal(referans nesnesi);
```

Örnek :

```
onClipEvent (enterFrame) {
      point1 = new object();
      point1.x = _root._xmouse;
      point1.y = _root._ymouse;
      _root.out13 = point1.x+" === "+point1.y;
      _root.out1 = _root._xmouse+" === "+_root._ymouse;
      localToGlobal(point1);
      _root.out12 = point1.x+" === "+point1.y;
      updateAfterEvent();
}

// Yukarıdaki kodları bir movie clip'e yazın.

onClipEvent (enterFrame) {
    globalToLocal(point1,point2);
      _root.cizgi._x = _root.kutu.point1.x;
```

```
        _root.cizgi._y = _root.kutu.point1.y;
        updateAfterEvent();
}

// Bir çizgi çizip, movie clip olarak tanımladıktan sonra
// yukarıdaki kodları yazın ve çalıştırın.
// Fare işaretçisini hareketi, çizginin yerini belirleyecek.
```

Player : Flash 5 ve sonrası

MovieClip.nextFrame : Belirtilen movie clip'in, kendi içindeki animasyonunda, oynatma başlığını bir sonraki frame'e gönderir. Yazılımı,

```
movieClip.nextFrame();
```

Örnek :

```
on (press){
        _root.klip.nextFrame();
}
```

Player : Flash 5 ve sonrası

MovieClip.play : Belirtilen movie clip'i oynatmak için kullanılır. Yazılımı,

```
movieClip.play();
```

Örnek :

```
if (skor = 100){
        _root.klip.play();
}
```

Player : Flash 5 ve sonrası

MovieClip.prevFrame : Belirtilen movie clip'in, kendi içindeki animasyonunda, oynatma başlığını bir önceki frame'e gönderir. Yazılımı,

```
movieClip.prevFrame();
```

Örnek :

```
on (press){
        _root.klip.prevFrame();
}
```

Player : Flash 5 ve sonrası

MovieClip.removeMovieClip : Bir movie clip, `duplicateMovieClip` veya `attachMovieClip` yöntemlerinden biri ile kopyalanarak yüklenmişse, bu movie clip'i sahneden kaldırmak için kullanılır. Yazılımı,

```
movieClip.removeMovieClip();
```

Örnek :

```
on (keypress "<Backspace>") {
      _root.klip.removeMovieClip();
}

// Klavyeden geri alma tuşuna basıldığı an, klip adlı
// movie clip'i sahneden kaldırır.
```

Player : Flash 5 ve sonrası

MovieClip.startDrag : Bir movie clip'i fare işaretçisine yapıştırmak için kullanılır. Daha çok değişik fare işaretçileri oluşturmak ve bir nesnenin yerini değiştirmek için kullanılır. Fare işaretçisine yapışan bu movie clip'i ister tüm sahnede, isterseniz sadece belirttiğiniz bir dörtgen alanda aktif yapabilirsiniz. Argüman olarak sadece merkeze yapışması için gerekli Boolean değerini yazarsanız, tüm sahnede aktif olur. Bu değerin arkasına, x ve y eksenindeki minimum ve maksimum sınırları yazarsanız, bu alanlarda aktif olacaktır. Yazılımı,

```
movieClip.startDrag(Boolean, sol, üst, sağ, alt);
```

Örnek :

```
klip.startDrag(true, 100, 100, 250, 250);

// İlk frame'e yukarıdaki kodları yazın.
// Sonra bir movie clip oluşturun (klip) ve çalıştırın.

// Argümanları değiştirin veya sayıları kaldırın tekrar deneyin
```

Player : Flash 5 ve sonrası

MovieClip.stop : Oynamakta olan bir movie clip'i durdurmak için kullanılır. Yazılımı,

```
movieClip.stop();
```

Örnek :

```
if (oyun.oyuncu_1.skor > 10){
      _root.oyun.demo.stop();
}
```

Player : Flash 5 ve sonrası

MovieClip.stopDrag : Önceden fare işaretçisine yapışan bir movie clip'in, fare işaretçisine yapışma işlemini durdurur. Yazılımı,

```
movieClip.stopDrag();
```

Örnek :

```
_root.klip.stopDrag();

// Daha önce fare işaretçisine yapışan klip adlı movie clip'in
// yapışma işlemi durdurulmuştur.
```

Player : Flash 5 ve sonrası

MovieClip.swapDepths : İki movie clip'in derinlik seviyelerini değiştirmek için kullanılır. Böylece, üstte olan movie clip, diğer movie clip'in altına geçer. Bu metodun doğru olarak çalışabilmesi için, iki movie clip'inde aynı yerde olması gerekir. Yani, biri bir başka movie clip'in içinde yeralırsa, çalışmaz. Bu metodun uygulandığı yerde, oynamakta olan movie clip'ler duracaktır. Argüman olarak, iki farklı değer girilebilir. Birincisi, derinlik seviyesidir. Böylece bu derinlik seviyesinde hangi movie clip varsa, onun yerine geçecektir. Kendisine ait olan derinlik seviyesi de, atıfta bulunduğu movie clip'e geçecektir. Dier argüman değeri ise, movie clip'in durum adıdır. Yazılımı,

```
MovieClip.swapDepths(seviye);
MovieClip.swapDepths(movieClip);
```

Örnek :

```
on (press){
      klip1.swapDepths(klip2);
}

// Düğmeye basıldığı an, klip2'nin derinlik seviyesi
// klip2'le yer değişecektir.
```

Player : Flash 5 ve sonrası

MovieClip.unloadMovie : Bir movie clip, `loadMovie` veya `attachMovieClip` yöntemlerinden biri ile kopyalanarak yüklenmişse, bu movie clip'i sahneden kaldırmak için kullanılır. Yazılımı,

```
movieClip.unloadMovie();
```

Örnek :

```
on (keypress "<Enter>") {
      _root.klip.unloadMovie();
}

// Klavyeden enter tuşuna basıldığı an, klip adlı
// movie clip'i sahneden kaldırır.
```

Player : Flash 4 ve sonrası

_name : Bir movie clip'in durum adını almak veya değiştirmek için kullanılır. Durum adı, Instance Panel'deki Name kısmına yazılan isimdir. Yazılımı,

```
durumAdı._name;
durumAdı._name = yeni isim;
```

Örnek :

```
on (press){
      klip._name = "yeniKlip";
      MC = yeniKlip._name;
      trace(isim);
}
```

```
// klip, adlı movie clip'in ismi yeniKlip olarak değiştirilmiş,
// değiştirilen bu isim, MC değişkenine atanmış.
```

Player : Flash 5 ve sonrası

NaN : İngilizcesi **N**ot **A** **N**umber (sayı değil) olan kelimelerin kısaltılmış halidir. Önceden tanımlanmış bir değişkendir. Bir değişkene, sayı olmadığını tanımlamak için kullanılır. Yazılımı,

```
NaN;
```

Örnek :

```
metin = NaN;
```

Player : Flash 5 ve sonrası

ne : İngilizce, **n**ot **e**qual (eşit değil) kelimelerinin kısaltılmış halidir. Bir değişkenin, diğer değişkene eşit olmama durumunu kontrol eder. Belirtilen değişkenler eşit ise, sonuç false olarak, eşit değilse, true olarak geri döner. Yazılımı,

```
ifade_1 ne ifade_2;
```

Örnek :

```
on (release){
      isim = isimKutu;
      parola = parolaKutu;
            if (isim ne "Murat"){
                  trace("İsim hatalı")
            }
            if (parola ne "1234"){
                  trace("Parola hatalı")
            }
}

// isim veya parola değişkeneleri yanlış yazıldı ise
// OutPut penceresinde hatalı olan kısım görünecektir.
```

Player : Flash 4 ve sonrası. Flash 5'de bu ifadenin yerine "!=" işareti kullanılır.

new : Yeni bir nesne oluşturmak için kullanılır. Flash ile önceden tanımlanmış nesnelerin dışında, özel nesneler ve bu nesnelere ait özellikler oluşturabilirsiniz. Bu özelliklere birden fazla değer atamak için bir fonksiyon tanımlanır. Her yeni bir değer için bu fonksiyon çağırılır. Yazılımı,

```
new isim();
```

Örnek :

```
function Kayit(isim, no) {
      this.isim = isim;
      this.no = no;
}

// Bu fonksiyon çağırılarak, yeni değerler oluşturulabilir.
```

```
kayit_1 = new Kayit("Batuhan", 1001);
kayit_2 = new Kayit("Ceren", 1002);

// Her iki kayıt için, aynı fonksiyon çağırılıyor.
// Buna göre, yeni nesnelerin özellikleri aşağıdaki gibidir

kayit_1.isim = "Batuhan";
kayit_1.no = 1001;
kayit_2.isim = "Ceren";
kayit_2.no = 1002;
```

Player : Flash 5 ve sonrası

newline : ActionScript kodlarında metinlerin arasında yeni bir satır açmaya yarar. Bu satırı, "/r" ifadesi ile de açabilirsiniz. Yazılımı,

```
newline;
```

Örnek :

```
// Çok satırlı bir text-box açın ve bir düğme yapın.
// düğmeye aşağıdaki kodları verin ve çalıştırın.

on (press){
      dizi = new Array("aaa", "bbb", "ccc");
      metinKutusu =dizi[0]+newline+dizi[1]+newline+dizi[2]
}
```

Player : Flash 4 ve sonrası

nextFrame : Zaman ekseni üzerindeki oynatma başlığını, bir sonraki frame'in üzerine gönderir ve durdurur. Yazılımı,

```
nextFrame();
```

Örnek :

```
on (release) {
      nextFrame();
}

// düğmeye basılıp bırakıldığı anda bir sonraki frame'e gider
```

Player : Flash 2 ve sonrası

nextScene : Zaman ekseni üzerindeki oynatma başlığını, bir sonraki sahnenin (scene), 1.frame'ine gönderir ve durdurur. Yazılımı,

```
nextScane();
```

Örnek :

```
on (release) {
      nextScane();
}
```

```
// düğmeye basılıp bırakıldığı anda bir sonraki sahnenin
// 1.frame'e gider
```

Player : Flash 2 ve sonrası

not : İngilizcedeki karşılığı, değil manasına gelir. Bir değişkenin, tuttuğu Boolean değerini, tersine çevirir. Belirtilen değişken true ise , sonuç false olarak, false ise, true olarak geri döner. Yazılımı,

```
not ifade;
```

Örnek :

```
on (release){
      no = noKutu;
            if (not no){
                  trace("Sayısal değer girilmelidir")
            }
}

// noKutu adlı yazı kutusuna sayısal bilgi girilmedi ise
// OutPut penceresinde hata mesajı görünecektir.
```

Player : Flash 4 ve sonrası. Flash 5'de bu ifadenin yerine "!" işareti kullanılır.

null : Bir değişkenenin içeriğini boşaltmak veya içeriğinin boş olup olmadığını kontrol etmek için kullanılır. Değişkenin değeri ne olursa olsun, null değeri atanırsa, değişkenin değeri silinecektir. Yazılımı,

```
null;
```

Örnek :

```
if (jNesne.isim == null){
      jNesne.isim = new Jnesne("belirtilmedi");
}

// Özel nesnenin özelliklerinden biri olan jNesne.isim,
// değeri boş ise, "belirtilmedi" diye bir değer atanacak.
```

Player : Flash 5 ve sonrası

Number (function) : Argüman olarak verilen bir ifadenin sayı olma durumuna göre değerlendirme yaparak, bir değer gönderir. Alınan bu değer, argümanın veri tipine göre değişir. Argüman sayı ise, sayıyı olduğu gibi verir. Argüman, Boolean değeri ise, true için 1, false için 0 değerini verir. Argüman alfanümerik ise, `NaN` değerini verir. Argüman belirtilmemiş bir veri tipi ise, 0 olarak geri döner. Yazılımı,

```
Number(ifade);
```

Örnek :

```
sonuc = Number(yaziKutusu)*100
      trace(sonuc);
```

Player : Flash 4 ve sonrası

Number (object) : Sayısal işlemlerde, bazı hazır değerlerden ve özelliklerden faydalanmak için kullanılır. Bu nesnenin 5 adet sabiti ve iki özelliği vardır. Ayrıca, yeni bir sayı üretmek için de, bu nesne kullanılır. Yazılımı,

```
Number.metod();
Number.sabit();
```

Örnek :

```
sayi = new Number(12345);

// sayi adlı değişkenin değeri 12345 olacaktır.
```

Player : Flash 5 ve sonrası

Number.MAX_VALUE : IEEE tarafından tespit edilmiş en büyük sayıyı tutar. Yaklaşık değeri 1.79769313486231e+308'dir. Bu değer, daha çok karşılaştırma yapmak için kullanılır. Yazılımı,

```
Number.MAX_VALUE;
```

Örnek :

```
dizi = new Array(25136, 365412, 3652148, 365112);

var x = Number.MAX_VALUE;
for (i=0; i<dizi.length; i++) {
      if (dizi[i]<x) {
            x = dizi[i];
      }
}
trace("En küçük sayı :" + x);
```

Player : Flash 5 ve sonrası

Number.MIN_VALUE : IEEE-754 tarafından tespit edilmiş en küçük sayıyı tutar. Yaklaşık değeri 5e-324'tür. Bu değer, daha çok karşılaştırma yapmak için kullanılır. Yazılımı,

```
Number.MIN_VALUE;
```

Örnek :

```
dizi = new Array(25136, 365412, 3652148, 365112);

var x = Number.MIN_VALUE;
for (i=0; i<dizi.length; i++) {
      if (dizi[i]<x) {
            x = dizi[i];
      }
}
trace("En büyük sayı :" + x);
```

Player : Flash 5 ve sonrası

Number.NaN : IEEE-754 tarafından tespit edilmiş, sayı olmayan değeri tutar. Bu değer, daha çok atama yapmak için kullanılır. Yazılımı,

```
Number.NaN;
```

Örnek :

```
dizi = new Array(2516, "abc", 365, 612);

var x = null;
for (i=0; i<dizi.length; i++) {
      if (not dizi[i]) {
            dizi[i] = Number.NaN;
            x=dizi[i];
      }
}
```

Player : Flash 5 ve sonrası

Number.NEGATIVE_INFINITY : IEEE-754 tarafından tespit edilmiş eksi sonsuzluk ifadesidir. Diğer bir değer olan `INFINTY` ile aynı işi görür. Bu değer, daha çok karşılaştırma yapmak için kullanılır. Yazılımı,

```
Number.NEGATIVE_INFINITY;
```

Örnek :

```
a = Math.pow(214, 132);
b = Math.pow(482, 63);
c = Math.pow(224, 42);
d = Math.pow(312, 111);
e = Math.pow(974, 44);

dizi = new Array(a, b, c, d, e);
diziAD = new Array("a", "b", "c", "d", "e");

x = Number.NEGATIVE_INFINITY;
for (i=0; i<5; i++) {
      if (dizi[i] > x) {
            x = dizi[i];
            y = i;
      }
}

trace("En büyük sayı: "+ diziAD[y]+" = "+ x);
```

Player : Flash 5 ve sonrası

Number.POSITIVE_INFINITY : IEEE-754 tarafından tespit edilmiş artı sonsuzluk ifadesidir. Diğer bir değer olan `INFINTY` ile aynı işi görür. Bu değer, daha çok karşılaştırma yapmak için kullanılır. Yazılımı,

```
Number.NEGATIVE_INFINITY;
```

Örnek :

```
a = Math.pow(214, 132);
b = Math.pow(482, 63);
c = Math.pow(224, 42);
d = Math.pow(312, 111);
e = Math.pow(974, 44);

dizi = new Array(a, b, c, d, e);
diziAD = new Array("a", "b", "c", "d", "e");

x = Number.POSITIVE_INFINITY;
for (i=0; i<5; i++) {
      if (dizi[i] < x) {
            x = dizi[i];
            y = i;
      }
}

trace("En küçük sayı: "+ diziAD[y]+" = "+ x);
```

Player : Flash 5 ve sonrası

Number.toString : Bir sayıyı, argüman olarak belirtilen tabana göre string'e çevirir. Sayı ilk önce `Number(function)` ile belirlenmiş olmalıdır. Yazılımı,

```
sayı.Number.toString(taban);
```

Örnek :

```
sayi = new Number(2125);
string = (2125).toString(2);

trace(string);

// OutPut penceresinde, 100001001101 değeri görünür.
// Bu sayı, 2125 sayısının, 2'lik tabana göre yazılışıdır.
```

Player : Flash 5 ve sonrası

Number.valueOf : Bir sayıyı, basit veri tipine dönüştürür. Sayı ilk önce `Number(function)` ile belirlenmiş olmalıdır. Yazılımı,

```
sayı.Number.valueOf();
```

Örnek :

```
sayi = new Number(2125);
deger = (2125).valueOf();

trace(deger);

// OutPut penceresinde, 2125 değeri görünür.
// Bu sayı, 2125 sayısının, basit veri tipine dönüşmüş halidir.
```

Player : Flash 5 ve sonrası

Object (Object) : Jenerik bir nesne oluşturmak için kullanılır. Bu nesne, `Sound` ve `Color` nesneleri için oluşturulur. Jenerik nesne oluşturmak için, başına `new` ifadesi koyulmalıdır. Yazılımı,

```
new Object();
```

Örnek :

```
// klip adlı movie clip için renk nesnesi oluşturuluyor.

renkNesne = new Color(klip);

// color transform nesnesi çağırıldığı zaman, buna cevap
// verebilecek, jenerik nesne oluşturuluyor.

renkTransform = new Object;

// renkTransform, jenerik nesnesinin ayarları belirleniyor.

renkTransform = { ra: '50', rb: '244', ga: '40', gb: '112', ba: '12', bb:
'90', aa: '40', ab: '70'}

// ve renk nesnesi ile jenerik nesne kullanılarak
// değerler, movie clip'e gönderiliyor.

renkNesne.setTransform(renkTransform);
```

Player : Flash 5 ve sonrası

Object.toString : Jenerik olarak oluşturulan nesnenin değerini string'e çevirir. Yazılımı,

```
jNesne.toString();
```

Player : Flash 5 ve sonrası

Object.valueOf : Jenerik olarak oluşturulan nesnenin değerini basit veri tipine dönüştürür. Yazılımı,

```
jNesne.valueOf();
```

Player : Flash 5 ve sonrası

onClipEvent : Bir movie clip'in hangi durumda yönetimi devralacağını belirler. Kendisine verilen komutları, argüman olarak belirtilen durumda yürütür. Bu durumlar aşağıdaki gibidir.

- `load` : Movie clip, sahnede göründüğü, yani tam olarak yüklendiği zaman.
- `unload` : Movie clip, sahneden kaldırıldığı zaman. Daha doğrusu, bir movie clip'in sahneden kaldırılması için bir komut geldiği zaman, kalkmadan hemen önce komutları yürütür.
- `enterFrame` : Her frame'in oynadığında.
- `mouseMove` : Fare işaretçisi, hareket ettiği zamanlarda.
- `mouseDown` : Fare üzerindeki sol tuşa basıldığında.
- `mouseUp` : Fare üzerindeki sol tuşa basılıp, bırakıldığında.
- `keyDown` : Klavyeden bir tuşa basıldığında. Ancak bu tuş, yürütülecek işlerin başında belirtilmeli.
- `keyUp` : Klavyeden bir tuşa basılıp, bırakıldığında. Bu tuş, yürütülecek işlerin başında belirtilmeli.

- `data` : `loadVariables` veya `loadMovie` ile komutları ile bilgilerin yüklenmesinden sonra işleri yürütür.

Yazılımı,

```
onClipEvent (durum) {
      yürütülecek işler;
}
```

Örnek :

```
onClipEvent(mouseMove) {
      _x=_root._xmouse;
      _y=_root._ymouse;
}

// startDrag benzeri kod. Fare işaretçisinin hareketi,
// movie clip'in yerini belirliyor.

onClipEvent(keyDown) {
      if (Key.getCode() == Key.RIGHT) {
            _x=getProperty("", _x)+10;
      } else if (Key.getCode() == Key.LEFT){
           _x=getProperty("", _x)-10;
      }
}

// Klavyeden, sol ve sağ ok işaretlerine basınca,
// movie clip sağa veya sola hareket edecek.
```

Player : Flash 5 ve sonrası

on (mouseEvent) : Bir düğmenin fare işaretçisi ile hangi durumda etkileşeceğini belirler. Bir nesneyi düğme (Button) olarak belirlediğiniz zaman, fare işaretçisi bu düğmeye, bir şekilde atıfta bulunmalıdır. Bu durumu, argüman olarak belirtmelisiniz. Bir yada birkaç durum belirtebilirsiniz. Her bir durum için farklı işleri icra etmesini de söylebilirsiniz. Bu durumlar aşağıdaki gibidir.

- `press` : Düğmenin üstünde, fare üzerindeki sol tuşa basıldığı an.
- `release` : Düğmenin üstünde, fare üzerindeki sol tuşa basılıp, bırakıldığı an.
- `releaseOutside` : Fare üzerindeki sol tuşa, düğme üzerinde basıp, düğmenin dışında bir yerde bırakıldığı zaman.
- `rollOver` : Fare işaretçisi, düğmenin üzerine geldiği an.
- `rollOut` : Fare işaretçisi, düğmenin üzerine gelip, düğmenin dışına çıktığı an.
- `dragOver` : Fare üzerindeki sol tuşa, düğme üzerinde basılıp, düğmenin dışına çıkılıp, tekrar düğme üzerine gelindiği an (sol tuş basılı kalacak).
- `dragOut` : Fare üzerindeki sol tuşa, düğme üzerinde basılıp, düğmenin dışına çıkıldığı an.
- `keyPress` : Klavyeden bir tuşa basıldığı an (bu tuş, argüman olarak belirtilmeli).

Yazılımı,

```
on (durum) {
      yürütülecek işler;
}
```

Örnek :

```
on (press) {
```

```
      startDrag("klip", true);
}
on (release) {
      stopDrag();
}

// Aynı zamanda, movie clip olan düğmeye basıldığı an,
// düğme fare işaretçisine yapışacak.
// Bırakıldığı an, yapışam işlemi duracak.
```

Player : Flash 2 ve sonrası

or : İngilizce manası "veya" demektir. İki adet koşulun durumunu kontrol eder. Her iki koşuldan en az birinin durumu, true ise true olarak geri döner. Eğer her iki koşulda false olursa, false olarak geri döner. Yazılımı,

```
koşul_1 or koşul_2;
```

Örnek :

```
if (x=100 or y>50){
      z = x + y;
}

// x, 100'e eşit ise veya y, 50'den büyük ise,
// z, x ve y'nin toplamıdır.
```

Player : Flash 4 ve sonrası. Flash 5'de bu ifadenin yerine "||" işareti kullanılır.

ord : Argüman olarak belirtilen bir karakteri, ASCII kod numarasına çevirir. Yazılımı,

```
ord(karakter);
```

Örnek :

```
trace(ord("m");

// OutPut penceresinde 109 sayısını göreceksiniz.
```

Player : Flash 4 ve sonrası. Flash 5'de bu ifadenin yerine String metodları kullanılır.

_parent : İç içe koyulmuş olan movie clip'lerde, tam olarak yolu belirtmek için kullanılır. En içte yer alan bir movie clip'e atıfta bulunmak için, sırası ile bütün movie clip'lerin durum adlarını ard arda yazarız (`_root.klip_1.klip_2.klip_3.play()` gibi). Tersi durumda ise `_parent` komutu kullanılır. Yani en içteki movie clip'ten en dıştaki movie clip'e atıfta bulunmak için kullanılır (`_parent._parent.stop()` gibi). Yazılımı,

```
_parent.özellik() = değer;
```

Örnek : Sahneye bir movie clip yapın. Movie clip'i sağ tıklayıp edit'i seçin ve başka bir movie clip yerleştirin. Aynı şekilde bu movie clip'in içine başka bir movie clip daha yerleştirin. Bunun da içine bir düğme yerleştirin. Buna göre, en dıştan en içe doğru şöyle sıralanacak;

MovieClip_1 > MovieClip_2 > MovieClip_3 > Button

MovieClip_3'e atıfta bulunmak için,

```
_root.MovieClip_1.MovieClip_2.MovieClip_3.play();
```

yazmak lazımdır. En içteki düğmeden, MovieClip_1'e atıfta bulunmak içinse,

```
_parent._parent.play();
```

yazmak gerekir.

Player : Flash 4 ve sonrası

parseFloat : Argüman olarak verilen bir stringi, sayıya dönüştürür. Ancak stringin, sayısal değeri olmalıdır. Eğer harf ile başlıyorsa, `NaN` (sayı değil) olarak geri döner. İlk karekter sayı ise, harf'e kadar olan sayıları verir. Yazılımı,

```
parseFloat(string);
```

Örnek :

```
ornek_1 = parseFloat("125");
ornek_2 = parseFloat("12abc34");
ornek_3 = parseFloat(-2.5");
ornek_4 = parseFloat("abc123");

// ornek_1, 125 olarak geri döner.
// ornek_1, 12 olarak geri döner.
// ornek_1, -2.5 olarak geri döner.
// ornek_1, NaN olarak geri döner.
```

Player : Flash 5 ve sonrası

parseInt : Argüman olarak verilen bir stringi, sayıya dönüştürür. Ancak stringin, sayısal değeri olmalıdır. Eğer harf ile başlıyorsa, `NaN` (sayı değil) olarak geri döner. İlk karekter sayı ise, harf'e kadar olan sayıları verir. Buraya kadar olan kısım, `parseFloat` ile aynıdır. Ancak `parseInt`, ikinci argüma sahiptir. Bu argüman, sayısal değerin, tabanını belirler. Ayrıca, hexadecimal cinsinden verilen bir string'i, belirtilen tabana göre dönüştürebilir. Yazılımı,

```
parseInt(string, taban);
```

Örnek :

```
ornek_1 = parseInt("123abc");
ornek_2 = parseInt("-12.3abc45");
ornek_3 = parseInt("abc");
ornek_4 = parseInt("2E4", 16);
ornek_5 = parseInt("0x12AF5");

// ornek_1, 123 olarak geri döner.
// ornek_2, -12.3 olarak geri döner.
// ornek_3, NaN olarak geri döner.
// ornek_4, 740 olarak geri döner.
// ornek_5, 76533 olarak geri döner.
```

Player : Flash 5 ve sonrası

play : Zaman ekseni üzerindeki oynatma başlığını aktive ederek, animasyonun oynatılmasını sağlar. Yazılımı,

```
play();
```

Örnek :

```
stop();
if (parolaKutu == "1234") {
      play();
} else{
      trace("Parola yanlış");
}
```

Player : Flash 2 ve sonrası

prevFrame : Zaman ekseni üzerindeki oynatma başlığını, bir önceki frame'in üzerine gönderir ve durdurur. Yazılımı,

```
prevFrame();
```

Örnek :

```
on (press){
      prevFrame();
}
```

Player : Flash 2 ve sonrası

prevScene : Zaman ekseni üzerindeki oynatma başlığını bir önceki sahnenin (scene) ilk frame'ine gönderir ve durdurur. Yazılımı,

```
prevScene();
```

Örnek :

```
on (release){
      prevScene();
}
```

Player : Flash 2 ve sonrası

print : Argüman olarak belirtilen bir movie clip'te görünen tüm nesnelerin, yazıcıdan, dökümü alınması sağlanır. Varsayılan değer olarak, her bir frame'deki görüntüyü ayrı ayrı kağıtlara yazdırır. Eğer tek bir frame'e ait görüntü yazdırılmak istenirse, ilgili frame etiketine (label) `#P` ifadesi yazılmalıdır. Bu etiket değeri görülmediği zaman, hedef olarak belirtilen movie clip'teki bütün frame'ler ayrı ayrı basılacaktır. Dökümü alınacak olan görüntü, bazı kriterlere göre belirlenir. Bu kriterler, ikinci argümanın değeri olarak belirtilmelidir. Bu değerler şöyledir,

- `bmovie` : Her frame'deki görüntüyü, kağıda sığdırmadan olduğu gibi yazdırır.
- `bmax` : Her frame'deki görüntüyü, kağıda sığdırarak yazdırır. Bir nesnenin boyutları değiştiren tween işlemi söz konusu ise, her sayfada aynı görüntü alınabilir.
- `bframe` : Her frame'deki görüntüyü, kağıda sığdırarak yazdırır. Ölçeklerin değiştiği bir tween için bu yöntemi kullanın. Her frame'deki ölçüyü koruyarak basar.

Bu komut icra edildiğinde, o ana kadar yüklenmiş olan nesneler yazdırılır. Yazılımı,

```
print(hedef, tip);
```

Örnek :

```
print("klip", "bmovie");

// klip adlı movie clip'in, tüm frame'leri (#P etiketi yoksa)
// kağıda sığdırılmadan yazdırılır.
```

Player : Flash 5 ve sonrası

printAsBitmap : `print` komutu gibidir. Ancak hedef olarak belirttiğiniz movie clip'te, şeffalık ve renk efektleri varsa, bu komut kullanılmalıdır. Bu komut ile görüntü, yüksek kalitede basılır. Yazılımı,

```
printAsBitmap(hedef, tip);
```

Örnek :

```
on (press){
      printAsBitmap("klip", "bmax");
}
```

Player : Flash 5 ve sonrası

_quality : Animasyondaki genel görüntünün kalitesini belirler. Dört adet değer verilebilir. LOW, en düşük kalite ve antialiasing kullanılmaz. MEDIUM, orta kalitede, antialiasing kullanılır ancak bitmap resimler pürüssüz değildir. HIGH, varsayılan değerdir. Bitmap resimler, durağan halde pürüssüzdür. BEST, en kaliteli görüntüdür. Yazılımı,

```
_quality = değer;
```

Örnek :

```
on (press){
      _quality = "BEST";
}

// Düğmeye basıldığı an, en iyi görüntü kalitesi, elde edilir.
```

Player : Flash 5 ve sonrası

random : Argüman olarak belirtilen sayı ile, 0 arasında rastgele bir tamsayı üretir. Yazılımı,

```
random(sayı);
```

Örnek :

```
random(10);

// 0,1,2,3,4,5,6,7,8,9, sayılarından biri gelir.
```

Player : Flash 4 ve sonrası. Flash 5'de bu ifadenin yerine, "Math.random" kullanılır.

removeMovieClip : Argüman olarak belirtilen bir movie clip'i sahneden silmek için kullanılır. Bu komut ile silebileceğiniz movie clip'ler, `duplicateMovieClip` veya `attachMovieClip` yöntemleri ile sahneye getirilmiş olmalıdır. Yazılımı,

```
removeMovieClip(hedef);
```

Örnek :

```
on (press){
      removeMovieClip("_root.klip");
}
```

Player : Flash 4 ve sonrası

return : Bir fonksiyonda, belirli işler icra edildikten sonra, bir değeri geri göndermek için kullanılır. Fonksiyon herhangi bir nedenden dolayı belirtilen işleri icra edemez ise, fonksiyondan dönen değer boş (null) olacaktır. Yazılımı,

```
return ifade;
```

Örnek :

```
function toplama(a,b,c) {
      return a+b+c;
}

// Bu fonksiyona üç adet sayı gönderdiğimiz zaman,
// üçünün toplamını, geri gönderecektir.
```

Player : Flash 5 ve sonrası

_root : Esas zaman eksenini belirtmek için kullanılır. Boş bir çalışma sahnesi açtığınızda, bu sahne `_root` olarak adlandırılır. Bu sahneye birden fazla swf dosyası eklenmiş ise, her biri farklı seviyelerde olacaktır. Ancak ilk sahne, `_root` olarak kalacak. Eğer başka bir seviyedeki animasyonun esas sahnesine atıfta bulunursak, `_root` ifadesi `level` olarak yorumlanacaktır (`_level2`, `_level4`...gibi). Bu ifade, bölme işareti gibi (/), hedef yolu belirtmede kullanılır. Yazılımı,

```
_root.hedef;
_root.metod;
```

Örnek :

```
on (press){
      _root.klip.play();
}

// Düğmeye basıldığı an, esas zaman ekseni üzerindeki,
// klip adlı movie clip'i oynatır.
```

Player : Flash 4 ve sonrası

_rotation : Bir movie clip'in, derece cinsinden döndürülme değerini belirler veya bu değer alınır. Yazılımı,

```
movieClip._rotation = değer;
değer = movieClip.rotation;
```

Örnek :

```
on (press){
      _root.klip._rotation = 100;
trace(_root.klip._rotation);
}

// Düğmeye basıldığı anda, klip adlı movie clip, 100 derece
// döndürülecek ve bu değeri OutPut penceresinde yazdıracaktır.
```

Player : Flash 4 ve sonrası

scroll : Bir yazı kutusu içindeki metinlerin satır kontrolünü yapar. Birden fazla satır içeren yazı kutularında, satırları aşağı veya yukarı kaydırmak için verilmiş bir özelliktir. Başlangıçta, `scroll` değeri 1'dir. Bu değer, daha çok `maxscroll` komutu için kullanılmakta. Yazılımı,

```
yazıKutusu.scroll = x;
```

Örnek : İki satırlık bir yazı kutusu yapın. Bu yazı kutusunun yanına, aşağı ve yukarıyı temsil eden, iki düğme yapın. İlk frame'e aşağıdaki kodları yazın.

```
yaziKutusu = "aaa\rbbb\rccc\rddd\reee\rfff";

// Yukarıyı temsil eden düğmeye şu kodları yazın.

on (press){
      root.yaziKutusu.scroll =_root.yaziKutusu.scroll - 1;
}

// Aşağıyı temsil eden düğmeye ise, şu kodları yazın.

on (press){
      _root.yaziKutusu.scroll =_root.yaziKutusu.scroll + 1;
}

// Programı çalıştırın ve düğmelere basın.
```

Player : Flash 4 ve sonrası

Selection.getBeginIndex : Argüman olarak belirtilen bir metin kutusu içinde, fare işaretçisi ile seçimin başladığı yeri verir. Eğer seçim yapılmamış ise, -1 olarak geri döner. Bu metod aynı frame içinde düzgün çalışmayabilir. Yazılımı,

```
Selection.getBeginIndex(metinKutusu);
```

Örnek :

```
// Sahneye iki keyframe yerleştirin bu frame'lerde
// iki adet metin kutusu olsun. Birine sonuc adı verin.
// 2. frame'e aşağıdaki kodları yazın.

sonuc = Selection.getBeginIndex("kutu");
```

```
// kutu adlı metin kutusuna birşeyler yazıp, fare ile
// herhangi bir yerden seçim yapın.
// sonuc kutusunda, seçim yapılan yeri verir.
```

Player : Flash 5 ve sonrası

Selection.getCaretIndex : Argüman olarak belirtilen bir metin kutusu içinde, imlecin yanıp söndüğü yeri verir. Eğer seçim yapılmamış ise, -1 olarak geri döner. Bu metod aynı frame içinde düzgün çalışmayabilir. Yazılımı,

```
Selection.getCaretIndex(metinKutusu);
```

Örnek :

```
// Sahneye iki keyframe yerleştirin bu frame'lerde
// iki adet metin kutusu olsun. Birine sonuc adı verin.
// 2. frame'e aşağıdaki kodları yazın.

sonuc = Selection.getCaretIndex("kutu");

// kutu adlı metin kutusuna birşeyler yazıp, fare ile
// herhangi bir yere tıklayın.
// sonuc kutusunda, imlecin olduğu yeri verir.
```

Player : Flash 5 ve sonrası

Selection.getEndIndex : Argüman olarak belirtilen bir metin kutusu içinde, fare işaretçisi ile seçimin bittiği yeri verir. Eğer seçim yapılmamış ise, -1 olarak geri döner. Bu metod aynı frame içinde düzgün çalışmayabilir. Yazılımı,

```
Selection.getEndIndex(metinKutusu);
```

Örnek :

```
// Sahneye iki keyframe yerleştirin bu frame'lerde
// iki adet metin kutusu olsun. Birine sonuc adı verin.
// 2. frame'e aşağıdaki kodları yazın.

sonuc = Selection.getEndIndex("kutu");

// kutu adlı metin kutusuna birşeyler yazıp, fare ile
// herhangi bir yerden seçim yapın.
// sonuc kutusunda, seçimin bittiği yeri verir.
```

Player : Flash 5 ve sonrası

Selection.getFocus : Sahnedeki birden fazla metin kutusunda fare ile nereye tıklandığını, yani hangi metin kutusunda çalışıldığını verir. Eğer seçim yapılmamış ise, null olarak geri döner. Bu metod aynı frame içinde düzgün çalışmayabilir. Yazılımı,

```
Selection.getFocus();
```

Örnek :

```
// Sahneye iki keyframe yerleştirin bu frame'lerde
```

```
// biden fazla metin kutusu olsun. Herbirine bir isim verin.
// 2. frame'e aşağıdaki kodları yazın.

trace(Selection.getFocus());

// OutPut penceresinde başlangıçta "null" ifadesi görünü
// Metin kutularından herhangi birine tıklarsanız,
// "_level0.metinkutusu" diye bir ifade gelir.
```

Player : Flash 5 ve sonrası

Selection.setFocus : Argüman olarak belirtilen bir metin kutusuna, imlecin gitmesi sağlanır. Böylece, bir metin kutusuna dikkat çekilmiş olur. Yazılımı,

```
Selection.setFocus(metinKutusu);
```

Örnek :

```
// Sahneye bir kaç tane metin kutusu yapın.
// Her birine bir isim verin.
// ilk frame'e aşağıdaki kodları yazın.

Selection.setFocus("kutu");

// Programı çalıştırdığınızda kutu adlı metin kutusunda
// imlecin yanıp söndüğünü göreceksiniz.
```

Player : Flash 5 ve sonrası

Selection.setSelection : Önceden dikkati çekilmiş bir metin kutusu içinde, başlangıç ve bitiş değerleri, argüman olarak verilen yeri, seçili hale getirir. Yani, bir metin kutusuna, önce `setFocus` ile imlecin oraya gitmesi sağlanır. Ardından `setSelection` ile, bu metin kutusu içinde, belirli bir yerin seçili hale gelmesi sağlanır. Bu metod aynı frame içinde düzgün çalışmayabilir. Yazılımı,

```
Selection.setSelection(başlangıç, bitiş);
```

Örnek :

```
// Sahneye bir metin kutusu yerleştirin. Adı "kutu" olsun.
// İkinci keyframe'i koyup, birincisine aşağıdaki kodları yazın

kutu="merhaba";
Selection.setFocus("kutu");

// İkinci frame'e ise aşağıdaki kodları yazın

Selection.setSelection(2,4);
stop();

// Programı çalıştırdığınızda, "rh" harflerinin seçili
// olduğunu göreceksiniz (merhaba).
```

Player : Flash 5 ve sonrası

set : Bir değişkene, bir değeri atamak için kullanılır. Normal atama işleminden farksızdır. Yazılımı,

```
set (değişken, değer);
```

Örnek :

```
on (release){
      set (x_pozisyon = getProperty("klip", _x));
}

// Yukarıdaki ifade, şu şekilde de yazılabilir.

on (release){
      x_pozisyon = getProperty("klip", _x);
}
```

Player : Flash 4 ve sonrası

setProperty : Animasyon oynarken, bir movie clip'in bir özelliğinin ayarlanması için kullanılır. Üç adet argümanı vardır. Birincisi, özelliği ayarlanacak olan hedef movie clip'in durum adıdır. İkincisi, movie clip'in ayarlanacak özelliğidir. Üçüncüsü ise, bu özelliğe atanacak olan değerdir. Yazılımı,

```
setProperty(movieClip, özellik, değer);
```

Örnek :

```
on (press){
      setProperty("klip", _alpha, 50);
}

// Düğmeye basıldığı an, klip adlı movie clip,
// %50 oranında görünür olacaktır.
```

Player : Flash 5 ve sonrası

Sound (object) : Jenerik olarak bir ses nesnesi oluşturarak, movie clip'in veya tüm animasyonun, ses kontrolünün sağlanması için kullanılır. Bir ses kontrolü sağlanmak isteniyorsa, başlangıçta bu jenerik nesneyi oluşturmak lazımdır. Oluşturulan bu nesneye herhangi bir argüman yazılmamışsa, bütün animasyondaki sesleri kontrol eder. Eğer bir movie clip'in durum adı yazılmışsa, sadece ilgili movie clip'e ait sesleri kontrol eder. Yazılımı,

```
sesNesne = new Sound();            // Bütün animasyon için,
sesNesne = new Sound(movieClip);   // hedef movie clip için.
```

Örnek :

```
on (press){
      ses = new Sound();
      ses.setVolume = 0;
}

// Animasyondaki sesleri kapatır.
```

Player : Flash 5 ve sonrası

Sound.attachSound : Kütüphanede yer alan bir sesi, sahneye taşımak ve bu sesi kullanılabilir hale getirmek için kullanılır. Kütüphanedeki sesin alınabilmesi için, dışarı çıkarılabilir olması gerekir. Bu özelliği vermek için de, kütüphanedeki ses dosyasının üstüne sağ tıklayın ve "Linkage" seçeneğini seçin. Gelen pencerede, "Export this symbol" seçeneğini seçili duruma getirin. Bunu yaptıktan sonra, yukarıdaki "Identifier" kutusu aktif olacaktır. Bu kutuya bir isim yazın. Yazdığınız bu isim, `attachSound` metodunu kullanırken vereceğiniz isimdir. Yazılımı,

```
sesNesnesi.attachSound("identifier");
```

Örnek :

```
sesNesnesi = new Sound();
sesNesnesi.attachSound("5.senfoni");
```

Player : Flash 5 ve sonrası

Sound.getPan : Bir ses nesnesinin, `setPan` ile denge ayarı yapıldıktan sonra, bu denge miktarının alınması için kullanılır. Alınan değer –100 (sol) ile 100 (sağ) arasında, tamsayı bir değerdir. Yazılımı,

```
sesNesnesi.getPan();
```

Örnek :

```
sesNesnesi = new Sound();
sesNesnesi.setVolume(50);
sesNesnesi.setPan(-100);

trace(sesNesnesi.getPan());

// Sesin hacmi %50 oranında ayarlanır.
// Tüm ses, sol kolona verilir.
// OutPut penceresinde, ayarlanan denge oranı alınır.
```

Player : Flash 5 ve sonrası

Sound.getTransform : Önceden oluşturulmuş olan ses nesnesinin, transform değeri alınır. Alınan bu değer, `setTransform` ile, en son ayarlanan değerdir. Ancak bu değer okunabilir bir değer değildir. Okunabilir olması için, özellik isminin de yazılması gerekir. Yazılımı,

```
sesNesnesi.getTransform();
```

Örnek :

```
sesNesnesi = new Sound();
sesTransformNesnesi = new Object();
sesTransformNesnesi.ll = 100;
sesTransformNesnesi.lr = 100;
sesTransformNesnesi.rr = 0;
sesTransformNesnesi.rl = 0;
sesNesnesi.setTransform(sesTransformNesnesi);

transform = sesNesnesi.getTransform();
trace(transform.ll);
trace(transform.lr);
```

```
trace(transform.rr);
trace(transform.rl);

// OutPut penceresinde, 100,100,0,0 değerleri görünecek.
```

Player : Flash 5 ve sonrası

Sound.getVolume : Önceden oluşturulmuş olan ses nesnesinin, ses hacmi alınır. Alınan bu değer, `setVolume` ile, en son ayarlanan değerdir. Yazılımı,

```
sesNesnesi.getVolume();
```

Örnek :

```
sesNesnesi = new Sound();
sesNesnesi.setVolume(75);

trace(sesNesnesi.getVolume());

// Sesin hacmi %75 oranında ayarlanır.
// OutPut penceresinde, ayarlanan ses hacim oranı alınır.
```

Player : Flash 5 ve sonrası

Sound.setPan : Oluşturulan bir ses nesnesinin, denge ayarı yapılır. Yapılan bu denge ayarına göre sesin, sağ ya da sol kolonlardaki hacmi belirlenir. Argüman olarak belirtilen bu denge ayarı, -100 ile 100 arasındadır. –100, tüm sesi sol kolonda çaldırır. 100 ise, sesin tamamını, sağ kolonda çaldırır. 0 değeri ise, sesi her iki kolona eşit olarak yayar. Bu metodu çağırdığınız zaman, daha önce oluşturduğunuz `setPan` ve `setTransform` ayarları silinir. Yazılımı,

```
sesNesnesi.setPan(değer);
```

Örnek :

```
sesNesnesi = new Sound();
sesNesnesi.setVolume(100);
sesNesnesi.start();
sesNesnesi.setPan(100);

// Sesin hacmi %100 oranında ayarlanır.
// Tüm ses, sağ kolona verilir.
```

Player : Flash 5 ve sonrası

Sound.setTransform : Ses nesnesininde çeşitli efektler yapmak için kullanılır. Bu metodu kullanabilmek için, ilk önce jenerik bir nesne oluşturmak gerekir. Oluşturulan bu jenerik nesneye dört özellik verilerek, sese çeşitli biçimler verilebilir. Bu metod çağırıldığı zaman, önceden oluşturulan `setPan` ve `setTransform` ayarları silinir. Jenerik nesnenin dört özelliği aşağıdaki gibidir.

- ll : Sol kolonda çalan sesin, sol giriş miktarıdır.
- lr : Sol kolonda çalan sesin, sağ giriş miktarıdır.
- rr : Sağ kolonda çalan sesin, sağ giriş miktarıdır.
- rl : Sağ kolonda çalan sesin, sol giriş miktarıdır.

Buna göre toplam sol kolondaki ses, ll + lr ve sağ kolondaki ses ise, rr + rl olacaktır. Bütün özellikler için verilebilecek değerler, -100 ile 100 arasındadır. Yazılımı,

```
jSesTransform = new Object();
jSesTransform.ll = değer;
jSesTransform.lr = değer;
jSesTransform.rr = değer;
jSesTransform.rl = değer;
```

Örnek :

```
sesNesnesi = new Sound();
sesNesnesi.attachSound("ses");
sesNesnesi.start(0,200);
jSesTransform = new Object();
jSesTransform.ll = -100;
jSesTransform.lr = 0;
jSesTransform.rr = -100;
jSesTransform.rl = 0;
sesNesnesi.setTransform(jSesTransform);
```

Player : Flash 5 ve sonrası

Sound.setVolume : Bir ses nesnesinin ses haciminin belirlenmesi için kullanılır. Varsayılan değer 100, yani maksimumdur. Argüman olarak belirleyebileceğiniz değerler, 0 ile 100 arasındadır. Yazılımı,

```
sesNesnesi.setVolume(değer);
```

Örnek :

```
onClipEvent (load) {
      sesNesnesi = new sound();
      sesNesnesi.setVolume(50);
}
```

Player : Flash 5 ve sonrası

Sound.start : Bir ses nesinin çalınmasını, yani aktive edilmesini sağlar. Argüman olarak bir değer girilmediği zaman, sesi bir kez çaldırır ve durdurur. Opsiyonel olarak iki argümanı vardır. Bu argümanlardan ilki, ses dosyasının kaç saniyelik bölümünün çalınacağını belirler. Yani 30 saniyelik bir ses dosyası için, ilk argüman olarak 15 girilmişse, daima 15. saniyeden itibaren çalacaktır. Eğer bu değeri 0 girerseniz, ses dosyasını olduğu gibi çalar. İkinci argüman ise, sese döngü vermek için kullanılır. Bir ses dosyasını, sürekli olarak çaldırmak isterseniz, yüksek bir değer girebilirsiniz. Bu değer animasyonun boyutuna etki etmeyecektir. Yazılımı,

```
sesNesnesi.start();
sesNesnesi.start(başlangıç, döngü);
```

Örnek :

```
sesNesnesi = new Sound();
sesNesnesi.attachSound("Türk_marşı");
sesNesnesi.start();

// Türk_marşı'nı bir kez çaldıracak ve duracaktır.
```

```
sesNesnesi = new Sound();
sesNesnesi.attachSound("Türk_marşı");
sesNesnesi.start(10,200);

// Türk_marşı'nı, 200 kez çaldıracaktır. Ancak her çalışında
// 10. saniyeden itibaren başlayacak.
```

Player : Flash 5 ve sonrası

Sound.stop : Çalınmakta olan ses dosyalarını durdurur. Eğer argüman olarak bir ses dosyası belirtilirse, çalmakta olan bu sesi durdurur. Ancak bu argümanda yer alan isim, `attachSound` metodu ile çağırılmış olmalıdır. Yazılımı,

```
sesNesnesi.stop();
sesNesnesi.stop("identifier");
```

Örnek :

```
sesNesnesi = new Sound();
sesNesnesi.attachSound("Cimbom_marşı");
sesNesnesi.start(0,100);

// Bir düğmeye şu kodları verin.

on (press){
      sesNesnesi.stop("Cimbom_marşı");
}

// Düğmeye basıldığı an, Cimbom_marşı duracaktır.
```

Player : Flash 5 ve sonrası

_sounbuftime : Animasyon içindeki ses dosyalarının belirtilen bir değer kadar, önceden hafızaya alınmasını sağlar. Böylece animasyon oynamadan önce belirtilen süre kadar, müzik hafızaya alınır ve kesintisiz olarak çalar. Varsayılan değeri 5 sn.'dir. Yazılımı,

```
_soundbuftime = değer;
```

Örnek :

```
_soundbuftime = 10;

// ses dosyalarının 10 sn önce yüklenmesi sağlanır.
```

Player : Flash 4 ve sonrası

startDrag : Argüman olarak belirtilen bir movie clip'i fare işaretçisine yapıştırmak için kullanılır. Daha çok değişik fare işaretçileri oluşturmak ve bir nesnenin yerini değiştirmek için kullanılır. Fare işaretçisine yapışan bu movie clip'i ister tüm sahnede, isterseniz sadece belirttiğiniz bir dörtgen alanda aktif yapabilirsiniz. Argüman olarak sadece merkeze yapışması için gerekli Boolean değerini yazarsanız, tüm sahnede aktif olur. Bu değerin arkasına, x ve y eksenindeki minimum ve maksimum sınırları yazarsanız, bu alanlarda aktif olacaktır. Yazılımı,

```
startDrag(hedef, Boolean, sol, üst, sağ, alt);
```

Örnek :

```
startDrag("klip", true, 100, 100, 250, 250);

// İlk frame'e yukarıdaki kodları yazın.
// Sonra bir movie clip oluşturun (klip) ve çalıştırın.

// Argümanları değiştirin veya sayıları kaldırın tekrar deneyin
```

Player : Flash 5 ve sonrası

stop : Oynamakta olan bir animasyonu, durdurmak için kullanılır. Yazılımı,

```
stop();
```

Örnek :

```
on (press){
      stop();
}

// Düğmeye basıldığı an, animasyon duracaktır.
```

Player : Flash 3 ve sonrası

stopAllSounds : Bir animasyon oynarken, çalmakta olan bütün sesleri durdurur. Ancak animasyon oynamaya devam eder. Yazılımı,

```
stopAllSounds();
```

Örnek :

```
on (release){
      stopAllSounds();
}

// düğmeye basılıp, bırakıldığı anda, tüm sesler duracaktır
```

Player : Flash 5 ve sonrası

stopDrag : Önceden oluşturulan `startDrag` komutunu pasive eder. Yani fare işaretçisine yapışmış olan bir movie clip'i serbest bırakır. Bırakılan movie clip'in yeri, komutun çalıştığı andaki yer olacaktır. Yazılımı,

```
stopDrag();
```

Örnek :

```
on (press){
      startDrag("klip");
}
on (release){
      stopDrag();
}
```

```
// Düğmeye basıldığı an, klip adlı movie clip,
// fare işaretçisine yapışacak.
// Bırakıldığı anda, yapışma işlemi duracak.
```

Player : Flash 4 ve sonrası

String (function) : Argüman olarak belirtilen bir ifadeyi değerlendirmek için kullanılır. Bu argüman bir string ise, aynen geri döner. Bir sayı ise, yine sayı olarak döner. Eğer argüman, bir Boolean değeri ise, Boolean değerini olduğu gibi verir. Bir movie clip ise, hedef yolu ile birlikte verir. Tanımlanmamış bir değer ise, boş olarak geri döner. Yazılımı,

```
String(ifade);
```

Örnek :

```
ornek_1 = String("abc");
ornek_2 = String(123);
ornek_3 = String(_root.klip);

// ornek_1, abc olarak geri döner.
// ornek_2, 123 olarak geri döner.
// ornek_3, _level0.klip olarak geri döner.
```

Player : Flash 3 ve sonrası

String (Object) : Kendisine ait 12 adet metod yardımı ile, bir string üzerinde çeşitli değişiklikler yapamak için kullanılır. Yeni bir string oluşturmak için, `new` ifadesi ile oluşturulabildiği gibi, bir değişkene çift tırnak işareti içinde atanan değerle olur. Ancak çift tırnak içine alınan bir string, nesne değildir. Bir string üzerinde bu metodlar uygulandığı zaman, program otomatikmen, bu string'i nesneye çevirir.Yazılımı,

```
yeniString = new String("Merhaba Dünya");
yeniString = "Merhaba Dünya";
```

Örnek :

```
yeniString = yaziKutusu;
uzunluk = yeniString.length;

// yazı kutusu içindeki metinin, kaç karakter olduğunu verir
```

Player : Flash 5 ve sonrası

String.charAt : String içinde, argüman olarak verilen bir sayıya karşılık gelen karakteri verir. String içindeki ilk karakter, 0'a tekabül eder. Eğer argüman olarak bir sayı girilmemişse, -1 olarak geri döner. Yazılımı,

```
string.charAt(sayı);
```

Örnek :

```
yeniString = new String("Merhaba Dünya");
x = yeniString.charAt(3);
```

```
// x = "h" olacaktır.
```

Player : Flash 5 ve sonrası

String.charCodeAt : String içinde, argüman olarak verilen bir sayıya karşılık gelen karakterin kodunu verir. String içindeki ilk karakter, 0'a tekabül eder. Yazılımı,

```
string.charCodeAt(sayı);
```

Örnek :

```
yeniString = new String("Merhaba Dünya");
x = yeniString.charAt(3);

// x = 104 olacaktır.
```

Player : Flash 5 ve sonrası

String.concat : Bir string'in arkasına başka karakterler eklemek için kullanılır. Argüman olarak verilen değer, string özelliği taşımalıdır. Yazılımı,

```
string.concat(değer_1, değer_2...değer_N);
```

Örnek :

```
on (press){
      ifade_1 = "Sayın ";
      ifade_2 = metinKutusu;
      ifade_3 = " hoşgeldiniz.";

      mesaj = ifade_1.concat(ifade_2,ifade_3);
}

// Düğmeye basıldığı anda, mesaj kutusunda,
// üç ifade birleşmiş şekilde görünecektir.
```

Player : Flash 5 ve sonrası

String.fromCharCode : Argüman olarak verilen kod numarasına karşılık gelen karakteri verir. String ifadesi ile birlikte yazılmalıdır. Yazılımı,

```
String.fromCharCode(kod_1,kod_2...kod_N);
```

Örnek :

```
for (i=48;i<65;i++){
      trace(String.fromCharCode(i));
}

// Pencerede, "0123456789:;<=>?@" karakterleri görünür.
```

Player : Flash 5 ve sonrası

String.indexOf : Argüman olarak verilen bir stringi, belirtilen string içinde, baştan sona doğru arar ve bulduğu ilk yeri, sayısal olarak verir. Eğer arama, belirli bir yerden sonra yaptırmak istenirse, ikinci argümana bu değer yazılmalıdır. Yazılımı,

```
string.indexOf(string);
string.indexOf(string, başlangıç);
```

Örnek :

```
mesaj = "Sayın ziyaretçimiz hoşgeldiniz";
x = mesaj.indexOf("z");

// x = 6 olacaktır. Bulduğu ilk "z"nin yerini verir.

y = mesaj.indexOf("z", 7);

// y = 17 olacaktır. 7. karakterden sonra arama yapar.
```

Player : Flash 5 ve sonrası

String.lastIndexOf : Argüman olarak verilen bir stringi, belirtilen string içinde sondan başa doğru arar ve bulduğu son yeri, sayısal olarak verir. Eğer arama, belirli bir yerden sonra yaptırmak istenirse, ikinci argümana bu değer yazılmalıdır. Yazılımı,

```
string.lastIndexOf(string);
string.lastIndexOf(string, başlangıç);
```

Örnek :

```
mesaj = "Sayın ziyaretçimiz hoşgeldiniz";
x = mesaj.lastIndexOf("z");

// x = 29 olacaktır. Bulduğu ilk "z"nin yerini verir.

y = mesaj.indexOf("z", 7);

// y = 6 olacaktır. 7. karakterden sonra arama yapar.
```

Player : Flash 5 ve sonrası

String.length : Bir string'in kaç karakterden oluştuğunu bulur. String içindeki son karakter, length-1 olacaktır. Yazılımı,

```
string.length;
```

Örnek :

```
mesaj = "Sayın ziyaretçimiz hoşgeldiniz";
x = mesaj.length;

// x = 30 olacaktır.
```

Player : Flash 5 ve sonrası

String.slice : Bir string içinde , belirtilen yerdeki karakterleri verir. İki adet argümanı vardır. Birincisi, başlangıç yeridir. İkincisi ise, bitiş yeridir. String içinde, başlangıç yeri dahil, bitiş yerine kadar (bitiş yeri hariç) olan karakterleri verir. Yazılımı,

```
string.slice(başlangıç, bitiş);
```

Örnek :

```
mesaj = "Sayın ziyaretçimiz hoşgeldiniz";
x = mesaj.slice(6,15);

// x = "ziyaretçi" olacaktır.
```

Player : Flash 5 ve sonrası

String.split : Bir string'i, argüman olarak verilen karaktere göre ayırır, bu karakterlerin yerine virgül işareti (,) koyar ve her ayrılan parçasını dizinin bir elemanı olarak yorumlar. Bu metodun uygulandığı yer, bir değişkene atanırsa, bu değişkeni dizi değişkene çevirir. Dizi değişkenin bütün özelliklerini taşır. Yazılımı,

```
string.split(karakter);
```

Örnek :

```
uyeler = "Hakkı+Osman+Mustafa+Memduh";

dizi = uyeler.split("+");

/*
dizi = "Hakkı,Osman,Mustafa,Memduh" olacaktır.
dizi.length = 4 olacaktır.
dizi[0] = "Hakkı",
dizi[1] = "Osman",
dizi[2] = "Mustafa",
dizi[3] = "Memduh", olacaktır.
*/
```

Player : Flash 5 ve sonrası

String.substr : Bir string içinde, belirtilen yerden başlayıp, belirtilen karakter kadar yeri alır. İki adet argümanı vardır. İlk argüman, yeni stringin nereden başlayacağını bildirir. İkinci argüman ise, kaç adet karakteri alacağını bildirir. Eğer ikinci argüman yazılmaz ise, tüm karakterleri alır. Yazılımı,

```
string.substr(başlangıç, karakter_sayısı);
```

Örnek :

```
mesaj = "Sayın ziyaretçimiz hoşgeldiniz";
x = mesaj.substr(6,9);

// x = "ziyaretçi" olacaktır.
```

Player : Flash 5 ve sonrası

String.substring : Bir string içinde, belirtilen yerden başlayıp, belirtilen yere kadar olan karakteri alır. İki adet argümanı vardır. İlk argüman, yeni stringin nereden başlayacağını bildirir. İkinci argüman ise, nerede biteceğini bildirir. Eğer ikinci argüman yazılmaz ise, tüm karakterleri alır. Yazılımı,

```
string.substring(başlangıç, bitiş);
```

Örnek :

```
mesaj = "Sayın ziyaretçimiz hoşgeldiniz";
x = mesaj.substring(6,9);

// x = "ziy" olacaktır.
```

Player : Flash 5 ve sonrası

String.toLowerCase : Bir string içindeki büyük harfleri küçük harfe dönüştür. Yazılımı,

```
string.toLowerCase();
```

Örnek :

```
mesaj = "YIL 2001, MERHABA DÜNYA";
mesaj.toLowerCase();

// mesaj = "yil 2001, merhaba dünya" olacaktır.
```

Player : Flash 5 ve sonrası

String.toUpperCase : Bir string içindeki küçük harfleri büyük harfe dönüştür. Yazılımı,

```
string.toUpperCase();
```

Örnek :

```
mesaj = "yil 2001, merhaba dünya";
mesaj.toUpperCase();

// mesaj = "YIL 2001, MERHABA DÜNYA" olacaktır.
```

Player : Flash 5 ve sonrası

substring : Argüman olarak verilen bir string içinden başka bir string oluşturmak için kullanılır. Üç argümandan ilki, hedef string'dir. İkincisi, kaçıncı karakterden başlayacağını bildirir. String nesnesinde olduğu gibi, ilk karakter 0 değildir. String'in ilk karakteri 1'dir. Üçüncü argüman ise, kaç karakterin alınacağını bildirir. Yazılımı,

```
substring(string, başlangıç, karakter_sayısı);
```

Örnek :

```
mesaj = "Merhaba Türkiye";
yeniMesaj = substring(mesaj, 1, 7);

// yeniMesaj = "Merhaba", olacaktır.
```

Player : Flash 4 ve sonrası. Flash 5'de bu fonksiyon kullanılmamaktadır.

_target : Bir movie clip'in tam yolunu verir. Verilen bu yol bölü işaretli şekilde olacaktır. Eğer bu ifadeyi, `_level` ifadesine dönüştürmek istenirse, `eval` fonksiyonu kullanılmalıdır. Yazılımı,

```
movieClip._target;
```

Örnek :

```
trace(klip_1.klip_2.klip_3._target);

// Pencerede, "/klip_1/klip_2/klip_3" olarak görünecektir.

trace(eval(klip_1.klip_2.klip_3._target));

// Pencerede, "level0.klip_1.klip_2.klip_3" olarak görünecek
```

Player : Flash 4 ve sonrası

targetPath : Argüman olarak belirtilen bir movie clip'e çeşitli işler yaptırmak için kullanılan bir fonksiyondur. Bu fonksiyona benzeyen diğer bir fonksiyon olan `tellTarget` ile aralarındaki tek fark, argümanların yazılış biçimidir. Bu komutla, noktalı şekilde yazılabilirken, `tellTarget` fonksiyonu bölü işareti ile yazılır. Yazılımı,

```
targetPath(movieClip){
      yürütülecek işler;
}
```

Örnek :

```
tagetPath(_root.klip){
      stop();
}

// fonksiyonun geçtiği yerde, movie clip duracaktır.
```

Player : Flash 5 ve sonrası

tellTarget : Hedef olarak belirtilen bir movie clip'e belirtilen işleri yaptırmak için kullanılır. Başlı başına bir fonksiyon gibi düşünülebilir. Hedef movie clip, argüman olarak yazılır. Bu movie clip'e atıfta bulunulacak işler ise, iki süslü parantez arasında yer alır. Yazılımı,

```
tellTarget (movieClip) {
      ifadeler;
}
```

Örnek :

```
on (press){
      tellTarget ("/klip") {
            gotoAndPlay(2);
      }
}

// Düğmeye basıldığı an, movie clip, 2. frame gidip oynayacak
```

Player : Flash 3 ve sonrası

this : İnglizce anlamı, "bu" demektir. ActionScript'te de kullanımı, anlamı gibidir. Atıfta bulunduğu nesne, yürütülen işlerin yer aldığı nesnenin kendisidir. Bir nesneye atıfta bulunurken, o nesnenin yolu ve ismi tam olarak belirtilir. Eğer kodların geçtiği yerde, this ifadesi varsa, bu kod veya kodlar, içinde geçtiği nesneye aittir. Yazılımı,

```
this
```

Örnek :

```
onClipEvent (load) {
      startDrag(this, true);
}

// Movie clip yüklendiği anda, fare işaretçisine yapışacak.
```

Player : Flash 5 ve sonrası

toggleHighQuality : Antialiasling'i açıp kapatmak için kullanılır. Bu komutu gördüğü yerde, antialiasing açıksa, kapatır. Eğer kapalı ise, bu kez açar. Yazılımı,

```
toggleHighQuality;
```

Örnek :

```
on (press) {
      toggleHighQuality;
}

// Düğmeye her basıldığında, antialiasin açılıp, kapanır.
```

Player : Flash 2 ve sonrası

_totalframes : Önüne yazılan bir movie clip'in toplam frame sayısını verir. Eğer önüne bir şey yazılmamışsa, o sahnedeki bütün animasyonun frame sayısını verir. Yazılımı,

```
_totalframes;
movieClip._totalframes;
```

Örnek :

```
x = _root.klip_1.klip_2._totalframes;

// x, klip_2'nin toplam frame sayısını verir.
```

Player : Flash 4 ve sonrası

trace : Javascript'teki alert komutu gibidir. Program bu komutu gördüğü yerde, bir pencere açar ve komuta verilen argümanları bu pencerede yazdırır. Daha çok programın akışını kontrol etmek için kullanılır. Tüm kontroller yapıldıktan sonra, bu komutlar silinmelidir veya Publish Settings'in Flash sekmesindeki, Omit Trace actions seçeneği işaretlenmelidir. Yazılımı,

```
trace(ifadeler);
```

Örnek :

```
dizi = new Array("a", "b", "c", "d");

for (i=0; i<dizi.length; i++){
      trace(i+ ".eleman :" + dizi[i]);
}
```

Player : Flash 4 ve sonrası

typeof : Argüman olarak verilen bir ifadenin veri tipini belirler. İfade bir değişken ise, program içinde kullanılan değerine göre veri tipini belirler. Yazılımı,

```
typeof(ifade);
```

Örnek :

```
trace(typeof("abc"));

// Pencerede, string olarak görünür.

trace(typeof(123));

// Pencerede, number olarak görünür.

trace(typeof(_root.klip));

// Pencerede, movieClip olarak görünür.
```

Player : Flash 5 ve sonrası

unescape : Argüman olarak verilen Hexadecimal bir ifadeyi, ASCII formatında bir stringe çevirir. Yazılımı,

```
escape(ifade);
```

Örnek :

```
mesaj = escape("Merhaba Dünya");
trace(mesaj);

// programı çalıştırdığınız zaman, OutPut penceresinde
// "Merhaba%20D%Fcnya" ibaresi görünür.

yeniMesaj = unescape(mesaj);
trace(yeniMesaj);

// Bu kez, "Merhaba Dünya" olacaktır.
```

Player : Flash 5 ve sonrası

unloadMovie : Önceden, `loadMovie` ile sahneye eklene bir movie clip'i kaldırmak için kullanılır. Argüman olarak ilgili movie clip'in derinlik seviyesi yazılır. Yazılımı,

```
unloadMovie (seviye);
```

Örnek :

```
on (press) {
      unloadMovie(_level20);
}

// Düğmeye basıldığı an, _level20'deki movie clip silinecek
```

Player : Flash 3 ve sonrası

updateAfterEvent : Bir movie clip'i, yönetici olarak belirledikten sonra, bazı işler yaptırılır. Movie clip, bu işleri yaptıktan sonra, yöneticiliği aldığı pozisyona göre, işleri durdurur. Bazı durumlarda, bu işlerin güncellenmesini isteyebiliriz. Bu gibi durumlarda, `updateAfterEvent` ifadesi kullanılır. Ancak bu komut, `mouseMove`, `mouseDown`, mouseUp, `keyDown` ve `keyUp` durumları için geçerlidir. Yazılımı,

```
updateAfterEvent;
```

Örnek :

```
onClipEvent (mouseMove) {
      _x = _root._xmouse;
      _y = _root._ymouse;
            updateAfterEvent;
}
```

Player : Flash 5 ve sonrası

_url : Bir movie clip'in bulunduğu yere göre, tam adresini verir. Komutu bilgisayarınızda çalıştırdığınızda, bilgisayarınızdaki yeri alırsınız. Web üzerinde çalıştırdığınızda ise, sitedeki yerini verir. Yazılımı,

```
_url;
```

Örnek :

```
trace(_root.klip1._url);

// Bilgisayarınızda, file:///C|/WINDOWS/TEMP/klip1.swf
// şeklinde olurken,
// Web'de, http://www.webteknikleri.com/klip1.swf
// şeklinde olur.
```

Player : Flash 4 ve sonrası

var : Bir değişken tanımlamak için kullanılır. Tanımlanan bu değişken, bir fonksiyon içinde yer alırsa (local variable), fonksiyon işlemi bitince, değişkende silinir. Fonksiyon içindeki değişken, geri gönderiliyorsa yerel değişken olmaktan çıkar ve silinmez (global variable). Yazılımı,

```
var değişken_ismi;
```

Örnek :

```
function (x,y){
	var z
	return z = x+y;
}

// z değişkeni global değişkendir.

for (var i=0; i<5;i++){
	trace(dizi[i]);
}

// i değişkeni local değişkendir. Fonksiyon bitince silinir.
```

Player : Flash 5 ve sonrası

_visible : Bir movie clip'in aktive edilip edilmeyeceğini belirler. Bir movie clip oluşturulduğu zaman, varsayılan değer olarak `_visible=true`, yani aktif olacaktır. Aktif olan bir movie clip ise, tüm fonksiyonları çalışır durumda ve tam olarak görünürdür. Eğer false olursa, movie clip görünmeycek ve tüm fonksiyonları kullanılmaz hale gelecektir. Diğer bir özellik olan `_alpha` ile karıştırlmamalıdır. Alpha, sadece şeffaflık değerini belirler. Yani, `_visible=false` olan bir movie clip'in içindeki düğmeler çalışmaz. Yazılımı,

```
movieClip._visible = Boolean;
```

Örnek :

```
onClipEvent (load){
	_visible=false;
}

// movie clip yüklendiği zaman, aktif değildir.
```

Player : Flash 4 ve sonrası

void : Argüman olarak verilen bir ifadenin tanımlanmış bir veri tipi olup olmadığını kontrol eder. Daha çok karşılaştırma yapmak için kullanılır. Yazılımı,

```
void(ifade);
```

Örnek :

```
if (ifade == void(ifade)) {
	// ifade'nin değeri boş.
	gotoAndStop(2);
} else{
	// ifade'nin tanımlı bir değeri var.
	gotoAndStop(3);
}
```

Player : Flash 5 ve sonrası

while : Bir döngü fonksiyonudur. Argüman olarak verilen koşul, true olduğu sürece, süslü parantez içindeki ifadeler yürütülür. Koşul false olarak geri dönerse, döngüden çıkılır. Diğer bir döngü

fonksiyonu olan `do..while`'dan farkı, döngünün işlemesi için koşulun gerçekleşmesi gerekmektedir. `do...while`'de ise, döngü en az bir kere döner. Yazılımı,

```
while (koşul) {
      ifadeler;
}
```

Örnek :

```
sayac = 0;
while (sayac < 10) {
      duplicateMovieClip("_root.klip", "klip"+sayac, sayac);
      setProperty("klip"+sayac, _x, random(200));
      setProperty("klip"+sayac, _y, random(200));
      setProperty("klip"+sayac, _xscale, random(200));
      setProperty("klip"+sayac, _yscale, random(200));
      setProperty("klip"+sayac, _alpha, random(100));
      sayac = sayac + 1;
}

/*
sayac değişkeni başlangıçta 0'dır. duplicateMovieClip yöntemi ile, klip
adlı movie clip'ten kopya alınıyor. Ve değişik özellikler atanıyor. Her
döngünün sonunda, sayac değişkeni 1 arttırılıyor. En son sayac değişkeni
10'a ulaştığı zaman, koşul false olarak geri dönecek ve döngü duracaktır.
*/
```

Player : Flash 4 ve sonrası

_width : Bir movie clip'in genişlikle ilgili bilgisinin, ayarlanıp veya alındığı bir özelliktir. Flash'ın bundan önceki sürümlerinde, bu özellik ile sadece genişlik bilgisi alınabilirdi. Flash 5 ile, bu özellik ayarlanabilir hale getirilmiş. Yazılımı,

```
değer = movieClip._width;
// movie clip'in yükseklik bilgisinin alınması

movieClip._width = değer;
// movie clip'in yükseklik bilgisinin ayarlanması
```

Örnek :

```
onClipEvent(mouseDown) {
      _width = 200;
      _height = 200;
}

// bir movie clip'e yukarıdaki aksiyonları yazın
// programı çalıştırın ve farenin sol tuşuna basın
// yükseklik ve genişlik değişecektir.
```

Player : Flash 4 ve sonrası

with : İngilizce anlamı, "ile" demektir. Bir nesneye atıfta bulunmak için kullanılır. Bu nesne bir movie clip, `Math` veya başka bir tanımalanmış nesne olabilir. Bir nesneye atıfta bulunamanın üç değişik yöntemi vardır. Nesnenin tam yolunu yazıp arkasına özelliği yazabilirsiniz veya `tellTarget` yönetimi ile atıfta bulunabilirsiniz. Ama en kullanışlısı, `with` komutudur. Bu komut yardımı ile, atıfta bulunacağınız nesnelerin içerdiği alt nesnelere de atıfta bulunabilirsiniz. Ayrıca nesne tipinin movie clip olması

gerekmiyor. Atıfta bulunacağınız nesneyi argüman olarak yazmalısınız. Süslü parantez içinde ise, yürütülecek işler yazılmalıdır. Bir `with` komutunun içinde başka `with` komutları yazılabilir. Yazılımı,

```
with (nesne) {
      yürütülecek işler;
}
```

Örnek :

```
// Nesneye atıfta bulunmak için, 1. yöntem;

klip._xscale = 75;

// 2. yöntem;

tellTarget (klip) {
      _xscale = 75;
}

// 3. yöntem;

with (klip) {
      _xscale = 75;
}

// Başka nesnelerin kullanımı;

function bilgi(ycap) {
      var x,y,a
      with (Math) {
            a = PI*ycap*ycap;
            x = r* cos(PI);
            y = r*sin(PI/2);
      }
trace("Alan :" + a);
trace("X :" + x);
trace("Y :" + y);
}

// İç içe kullanımı;

with (klip_1) {
      with (klip_2) {
            play();
      }
      with (klip_3) {
            stop();
      }
}

// klip_1 adlı movie clip'in içinde iki adet movie clip vardır
// bunlardan, klip_2 oynayacak, klip_3 duracaktır.
```

Player : Flash 5 ve sonrası

_x : Bir movie clip'in, x – eksenindeki yer bilgisi için kullanılır. Bu bilgiye bir değer atayarak, movie clip'in x – eksenindeki yerini belirleyebilir veya bu bilgiyi alabilirsiniz. Verilen bu bilgi, sahnenin sol üst noktasını referans alarak belirlenir. Sahnenin sol üst noktası (0,0) noktasıdır. Yazılımı,

```
movieClip._x = değer;
değer = movieClip._x;
```

Örnek :

```
on (press){
      _root.klip._x = 100;
}

// movie clip'in x – eskenindeki yeri, 100 olarak belirlenmiş
```

Player : Flash 3 ve sonrası

_xmouse : Fare işaretçisinin sahnede, x – eksenindeki yerini belirler. Bu bilgiye bir atama yapılamaz. Bu komutun önüne bir movie clip durum adı yazıldığı zaman, ilgili movie clip'in kendi alanı içindeki yeri verir. Yazılımı,

```
_xmouse;
movieClip._xmouse;
```

Örnek :

```
on (press) {
      trace(_xmouse);
}

// düğmeye basıldığı an, fare işaretçisin yerini yazdırır.
```

Player : Flash 5 ve sonrası

_xscale : Bir movie clip'in genişliğini yüzde olarak belirler veya bu bilgi alınır. Bir movie clip'in genişliği, varsayılan değer olarak %100'dür. Yazılımı,

```
movieClip._xscale;
```

Örnek :

```
onClipEvent (load) {
      _xscale = 50;
}

// mocie clip sahnede göründüğü zaman, genişliği %50 olacaktır
```

Player : Flash 5 ve sonrası

_y : Bir movie clip'in, y – eksenindeki yer bilgisi için kullanılır. Bu bilgiye bir değer atayarak, movie clip'in y – eksenindeki yerini belirleyebilir veya bu bilgiyi alabilirsiniz. Verilen bu bilgi, sahnenin sol üst noktasını referans alarak belirlenir. Sahnenin sol üst noktası (0,0) noktasıdır. Yazılımı,

```
movieClip._y = değer;
değer = movieClip._y;
```

Örnek :

```
on (press){
```

```
      _root.klip._y = -250;
}

// movie clip'in x - eskenindeki yeri, -250 olarak belirlenmiş
// yani sahnenin görünmeyen kısmındadır.
```

Player : Flash 3 ve sonrası

_ymouse : Fare işaretçisinin sahnede, y – eksenindeki yerini belirler. Bu bilgiye bir atama yapılamaz. Bu komutun önüne bir movie clip durum adı yazıldığı zaman, ilgili movie clip'in kendi alanı içindeki yeri verir. Yazılımı,

```
_ymouse;
movieClip._ymouse;
```

Örnek :

```
on (press) {
      trace(_ymouse);
}

// düğmeye basıldığı an, fare işaretçisin yerini yazdırır.
```

Player : Flash 5 ve sonrası

_yscale : Bir movie clip'in yüklekliğini yüzde olarak belirler veya bu bilgi alınır. Bir movie clip'in yüksekliği, varsayılan değer olarak %100'dür. Yazılımı,

```
movieClip._yscale;
```

Örnek :

```
onClipEvent (load) {
      _yscale = 50;
}

// mocie clip sahnede göründüğü zaman, genişliği %50 olacaktır
```

Player : Flash 5 ve sonrası