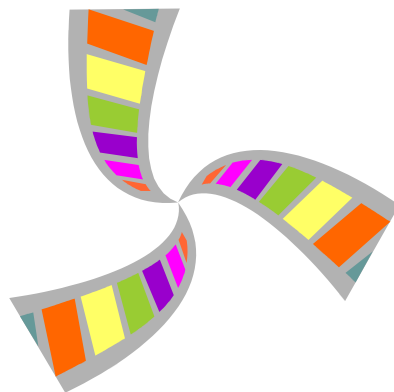




Programowanie w języku

ActionScript



Wprowadzenie do ActionScript

ActionScript to język opracowany do zastosowań w animacjach Flash. Pozwala on na kontrolę zawartości dokumentu Flash za pomocą komend wykonywanych w trakcie odtwarzania filmu.

Opisane jest tu programowanie w programie Alligator Flash Designer. Bezpłatną wersję testową można pobrać ze strony www.flashdesigner.pl

Minimalna wersja oprogramowania Alligator Flash Designer to 7.1 Wersję programu można sprawdzić po wybraniu komendy Pomoc > O programie.

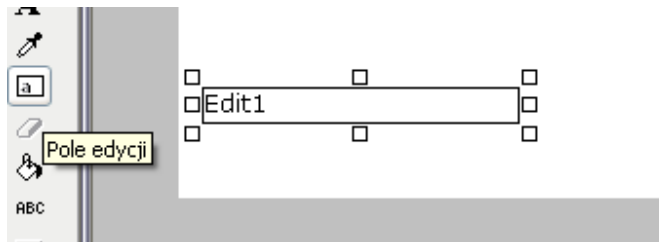
Kod ActionScript wpisujemy bezpośrednio w projekcie Flash. Rozróżniamy dwa typy skryptów, kod ramek oraz kod zdarzeń przycisków.

Skrypty ramek

ActionScript ramki jest wykonywany tuż przed jej wyświetleniem. Aby zdefiniować skrypt wybierz "Ramka" > "Kod ActionScript" i wpisz skrypt w polu dialogowym.

Najprostszy skrypt

Uruchom program Alligator Flash Designer, narysuj pole edycji narzędziem "Pole edycji". Pole pojawi się jako element "Edit1" (lub Edit2 itp.).



Następnie wybierz "Ramka" > "Kod ActionScript" i wklej poniższy kod:

```
Edit1 = "witaj!!!";
```

Wciśnij klawisz F9 aby uruchomić podgląd animacji i wykonać skrypt. W polu edycji pojawi się napis "witaj!!!".

Składnia języka

Skrypt programu ActionScript składa się z szeregu komend zakończonych znakiem średnika ; Dla przejrzystości skryptu najlepiej jest umieszczać każdą komendę w oddzielnej linii.

Teksty i liczby

Aby odróżnić teksty i liczby w języku ActionScript teksty ogranicza się znakami cudzysłowu a liczby wpisuje się bez takich znaków.

W tym przykładzie w pole Edit1 zamiast napisu "witaj!!!" zostanie wpisana liczba 100:

```
Edit1 = 100;
```

Zmienne

Zmienne dzielimy na tekstowe, przechowujące ciągi znaków oraz na liczbowe przechowujące liczby. Zmienne przechowują dane podczas trwania całej animacji Flash.

Dla przykładu obliczmy za pomocą komend ActionScript pole prostokąta o wymiarach 20 na 30:

Uruchom program Alligator Flash Designer, narysuj pole edycji narzędziem "Pole edycji". Pole pojawi się jako element "Edit1" (lub Edit2 itp.).

Następnie wybierz "Ramka" > "Kod ActionScript" i wklej poniższy kod:

```
szerokosc = 20;  
wysokosc = 30;  
wynik = szerokosc * wysokosc;  
Edit1 = wynik;
```

Wciśnij klawisz F9 aby uruchomić podgląd animacji i wykonać skrypt. W polu edycji pojawi się napis "600".

Pierwsze 2 komendy to przypisanie początkowych wartości zmiennym szerokosc i wysokosc, 3 komenda to przypisanie zmiennej pole wyniku mnożenia a komenda 4 wyświetla zmienną wynik w polu edycji o nazwie Edit1.

Na zmiennych liczbowych możemy wykonywać różne działania matematyczne, możemy także używać nawiasów () zgodnie z zasadami matematyki. Zamiast zmiennych można także używać liczb wpisanych bezpośrednio w działanie.

na przykład obliczmy pole trójkąta

```
wynik = 0.5 * szerokosc * wysokosc;
```

lub bardziej skomplikowane działanie

```
wynik = 1.45 + (szerokosc * wysokosc + 20) * 100;
```

Zmienne tekstowe

Na zmiennych tekstowych możemy wykonać tylko działanie dodawania. W wyniku uzyskamy teksty połączone.

Zmodyfikuj zatem program obliczania pola:

```
szerokosc = 20;  
wysokosc = 30;  
wynik = szerokosc * wysokosc;  
napis = "Pole: " + wynik + " m2";  
Edit1 = napis;
```

W wyniku uzyskamy napis "Pole: 600 m2".

Ciągi znaków które przypisujemy do zmiennych lub używamy do połączeń ciągów muszą być zawarte w znakach cudzysłowu.

Nazwa zmiennej musi zaczynać się od znaku alfanumerycznego czyli od a do z i może zawierać oprócz liter także cyfry o ile nie są na pierwszej pozycji oraz znak podkreślenia _ .

W nazwach zmiennych nie należy używać polskich znaków.

Przykłady zmiennych:

zmienna1, moja_zmienna

Przykłady błędnych zmiennych

1zmienna (zaczyna się od cyfry)
małazmienna (ma polskie znaki)

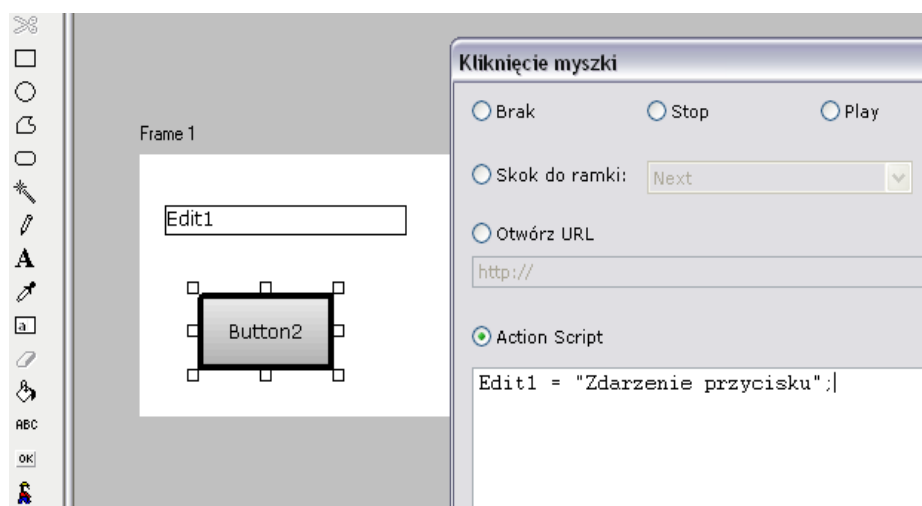
Skrypty przycisków

ActionScript może być wykonywany przy zdarzeniach myszki (kliknięcia, najechania, wyjechania, zwolnienia klawisza). Aby zdefiniować skrypt przycisku zaznacz obiekt i wybierz "Obiekt" > "Akcje" i jedno z poleceń związane ze zdarzeniami myszki: "Kliknięcia", "Najechania", "Wyjechania" lub "Zwolnienia klawisza". W okienku edycji akcji zaznacz opcję "ActionScript" i wstaw skrypt w poniższym polu edycji.

Aby zdefiniować prostą obsługę zdarzenia kliknięcia, otwórz nowy projekt Flash i stwórz 2 obiekty: pole tekstowe Edit1 oraz przycisk Button2. Następnie zaznacz obiekt Button2 i wybierz komendę "Obiekt" > "Akcje" > "Kliknięcia".

Wpisz poniższy kod i kliknij Ok

```
Edit1 = "Zdarzenie przycisku";
```



Uruchom podgląd. Na początku nic się nie dzieje, jednak po kliknięciu przycisku Button2 kod zostanie wykonany i napis "Zdarzenie przycisku" pojawi się w polu edycji.

Zdarzenie najechania i wyjechania.

Zrób to samo dla innych dostępnych zdarzeń. Dla najechania wpisz kod:

```
Edit1 = "Najechanie myszy";
```

a dla wyjechania

```
Edit1 = "";
```

co spowoduje skasowanie zawartości pola tekstowego.

Komendy warunkowe

Instrukcja ta sprawdza wartość zmiennej a następnie wykonuje część kodu tylko jeżeli warunek jest spełniony

Składnia tej komendy wygląda następująco:

```
if ( warunek)
{
  .. wykonanie kodu
}
```

Przykład:

```
szerokosc = 20;
wysokosc = 30;
wynik = szerokosc * wysokosc;
if (wynik > 500)
{
  napis = "Pole > 500";
}
Edit1 = napis;
```

Jeżeli obliczone pole będzie większe od 500 to w polu Edit1 pojawi się napis "Pole > 500".

Instrukcja else

Komenda else bezpośrednio po nawiasie zamykającym } powoduje wykonanie kodu jeżeli warunek nie jest spełniony:

```
szerokosc = 20;
wysokosc = 30;
wynik = szerokosc * wysokosc;
if (wynik > 500)
{
  napis = "Ale wielkie pole";
}
else
{
  napis = "Ale malutkie pole";
}
Edit1 = napis;
```

W powyższym przykładzie jeżeli wynik jest większy od 500 wykona się komenda napis = "Ale wielkie pole", w przeciwnym wypadku wykona się komenda napis = "Ale malutkie pole".

Pętle

Pętla powoduje wykonanie tego samego kodu kilka razy, za każdym razem pewna zmienna zostaje powiększona (lub zmniejszona) o dany krok, co pozwala na wykonanie tych samych obliczeń dla różnych wartości zmiennej.

Instrukcja for

Instrukcja ta ma następującą składnię

```
for( wartość początkowa ; warunek kontynuacji ; komenda zwiększająca )
{
    te instrukcje zostaną powtórzone w pętli
}
```

Dla przykładu napisz najprostszą pętlę

Narysuj pole tekstowe Edit1, wybierz "Ramka" > "Kod ActionScript" i wpisz skrypt w polu dialogowym:

```
napis = "Liczby parzyste: ";
for( i = 2 ; i < 10 ; i = i + 2 )
{
    napis = napis + i + " ";
}
Edit1 = napis;
```

Powyższy kod spowoduje wyświetlenie napisu Liczby parzyste: 2 4 6 8

Instrukcja while

Innym rodzajem pętli jest pętla while. Ma ona następującą składnię:

```
while( warunek )
{
    te instrukcje zostaną powtórzone w pętli
}
```

W tym przypadku sami musimy napisać komendę inicjującą zmienną oraz zwiększającą jej wartość tak aby pętla się kiedykolwiek skończyła.

Poniższy kod robi dokładnie to samo co pętla for.

```
napis = "Liczby parzyste: ";
i = 2;
while( i < 10 )
{
    napis = napis + i + " ";
    i = i + 2;
}
Edit1 = napis;
```

Funkcje

Funkcja to kod który może być zachowany w pamięci i wykonany jako jedna z komend programu ActionScript. Dodatkowo dla funkcji można przekazać parametry oraz odczytać wartość obliczoną z funkcji.

W poniższym przykładzie zdefiniujemy funkcję obliczającą pole prostokąta oraz wywołamy ją.

Składnia programu wygląda następująco:

```
function pole(szerokosc,wysokosc)
{
    wynik = szerokosc * wysokosc;
    return wynik;
}
```

```
Edit1 = pole(20,30);
```

Jak widać definicję funkcji poprzedzamy słowem function, następnie należy podać nazwę funkcji która podlega takim samym ograniczeniom jak nazwy zmiennych, tzn nie może zaczynać się od cyfry i może zawierać tylko litery lub cyfry oraz znak podkreślenia.

Następnym elementem jest lista parametrów funkcji rozdzielona przecinkami i zawarta w nawiasach () w tym przypadku nasze parametry to szerokość i wysokość prostokąta.

W nawiasach { } wpisujemy kod funkcji.

Ostatnią komendą jest `return` który powoduje zwrócenie obliczonej wartości.

Aby wywołać funkcję wpisujemy jej nazwę z tym że po nazwie podajemy parametry w nawiasach ().

Efektom wykonania tego programu będzie wpisanie w polu Edit1 wartości 600.

Tablice

Tablica to zmienna, która posiada kilka wartości do których możemy odnosić się za pomocą indeksu. Indeks tablicy wpisujemy w nawiasach []. Dla przykładu stwórzmy tablicę przechowującą imiona żeńskie.

```
imiona = new Array();  
  
imiona[0] = "Julia";  
imiona[1] = "Maria";  
imiona[2] = "Sandra";  
  
Edit1 = imiona[2];
```

Tablice w przeciwieństwie do zmiennych liczbowych i tekstowych wymagają wcześniejszego utworzenia. W pierwszej komendzie tworzymy obiekt typu Array czyli tablica instrukcją new.

Tablicę możemy także zainicjalizować bezpośrednio w komendzie new:

```
imiona = new Array("Julia","Maria","Sandra");  
Edit1 = imiona[2];
```

Obiekty

Obiekty zwane inaczej klasami są w pewnym sensie podobne do zmiennych, jednak ich struktura pozwala na przechowywanie zmiennych zwanych atrybutami oraz funkcji zwanych metodami, których można użyć do modyfikacji danego obiektu.

Przykładowo wszystkie obiekty Sprite są klasy MovieClip. Zawierają zatem między innymi atrybuty `_x` oraz `_y` które są pozycją lewego górnego rogu Sprite. Modyfikując te atrybuty możemy przemieszczać obiekt Sprite na ekranie.

Do atrybutów i metod klasy możemy odnosić się za pomocą kropki pomiędzy nazwą zmiennej reprezentującej dany obiekt a nazwą funkcji (zwanej metodą) lub zmiennej klasy.

Celem ćwiczenia utwórz nowy projekt Flash i narysuj obiekt Sprite1, w Spricie narysuj kółko. Wyjdź ze sprita i wpisz kod ramki:

```
Sprite1._x = 0;  
Sprite1._y = 0;
```

Wciśnij F2 aby ewentualnie zmienić nazwę sprite tak aby była identyczna do tej która jest użyta w kodzie, w tym przypadku Sprite musi nazywać się "Sprite1". Wciśnij F9 aby wykonać kod. Sprite przesunie się do lewego górnego rogu projektu.

Obiekt jeżeli nie został utworzony przez film lub przez funkcję musimy utworzyć sami. Służy do tego komenda **new**

Użycie komendy new wygląda następująco:

```
zmienna = new TypObiektu( parametry );
```

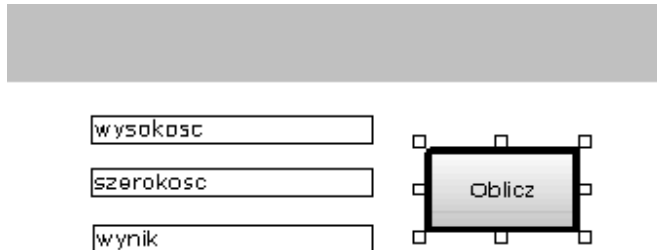
Pola edycji

Pola edycji które najczęściej występują pod nazwami Edit1, Edit2 itd mogą służyć zarówno do wyświetlania zmiennych, jak również do wczytywania danych.

W poniższym przykładzie napiszemy kod który oblicza pole prostokąta, tym razem jednak wysokość i szerokość zostaną wczytane z pól edycji.

Utwórz nowy projekt Flash i narysuj 3 pola: Edit1, Edit2 i Edit3 oraz przycisk Button4.

Zaznacz każde z pól i wciśnij Enter aby zmienić jego parametry. Nazwij pola odpowiednio wysokosc, szerokosc (bez polskich liter) i wynik a przycisk nazwij Oblicz:



Następnie w akcję kliknięcia przycisku Oblicz wpisz kod:

```
wynik = wysokosc * szerokosc;
```

Wciśnij F9 i spróbuj wykonać kilka obliczeń wpisując dane wejściowe i klikając przycisk Oblicz.

Pola wieloliniowe

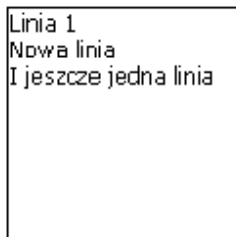
Domyślnie pola tekstowe są jednoliniowe. Aby zmodyfikować parametry pola zaznacz je i wciśnij Enter. Zaznacz opcje Wiele wierszy aby pole akceptowało znaki nowej linii oraz Zawijaj wiersze jeżeli nie chcesz aby tekst przesunął się jeżeli osiągnie krawędź pola ale wiersz przeskakiwał do nowej linii.

Znak nowej linii

Ciąg "\n" zawiera znak przeniesienia tekstu do nowej linii. Utwórz pole tekstowe i rozciągnij je w pionie tak aby mogło zawierać kilka linii, zaznacz opcję Wiele wierszy.

Wpisz następujący kod ramki:

```
Edit1 = "Linia 1\nNowa linia\nI jeszcze jedna linia";
```



Operatory

Operatory to komendy działań matematycznych lub logicznych na zmiennych i liczbach

+

dodaje wartości

jeżeli parametry nie są numeryczne należy przekonwertować je na liczbę przy pomocy funkcji Number

```
Edit1 = Number(Edit2) + Number(Edit3);
```

-

odejmuje wartości

/

dzieli wartości

*

mnoży wartości

%

modulo, reszta dzielenia całkowitego

na przykład

```
Edit1 = 10 % 3;
```

w wyniku uzyskamy liczbę 1

Operatory porównania

Operatory te są głównie używane w komendach warunkowych i zwracają wartość true (prawda) lub false (fałsz)

<

mniejsze, zwraca true jeżeli pierwszy parametr jest mniejszy od drugiego

>

większe, zwraca true jeżeli pierwszy parametr jest większy od drugiego

<=

mniejsze lub równe, zwraca true jeżeli pierwszy parametr jest mniejszy lub równy drugiemu

>=

większe lub równe, zwraca true jeżeli pierwszy parametr jest większy lub równy od drugiemu

==

równość, zwraca true jeżeli parametry są identyczne, false jeżeli różne

===

równość dokładna, zwraca true jeżeli parametry są identyczne oraz identycznego typu, false jeżeli różne

!=

nierówność, zwraca true jeżeli parametry są różne, false jeżeli identyczne

!

negacja logiczna, zwraca odwrotność parametru przed którym stoi

Operatory logiczne

Operatory te są głównie używane w komendach warunkowych i zwracają wartość true (prawda) lub false (fałsz)

&&

suma logiczna, zwraca true jeżeli obydwa warunki są spełnione, w przeciwnym razie false

||

zwraca true jeżeli jeden z warunków jest spełniony, jeżeli obydwa nie są to zwraca false

Operatory bitowe

Operatory te służą do działania na bitach liczb w postaci zerojedynkowej

Przykład liczb zapisanych w systemie dziesiętnym i zerojedynkowym

1 = 00000001
2 = 00000010
3 = 00000011
4 = 00000100
8 = 00001000
16 = 00010000
32 = 00100000

&

suma bitowa, jeżeli w obydwu parametrach na danej pozycji bit ma wartość 1 to w wyniku na tej pozycji bit także ma wartość 1

przykład

1 & 2 = 0
1 & 3 = 1

|

jeżeli w jednym z parametrów na danej pozycji bit ma wartość 1 to w wyniku na tej pozycji bit także ma wartość 1

przykład

1 | 2 = 3

^

Xor, jeżeli bity na danej pozycji są równe to wynikiem jest bit 0 jeżeli są różne to wynikiem jest 1

przykład

1 ^ 3 = 2

~

Negacja bitowa, odwraca wartości bitów na każdej pozycji

Operatory przypisania

Operator przypisania powoduje obliczenie wartości wyrażenia po prawej stronie znaku równości i zapamiętanie go w zmiennej podanej po lewej stronie znaku równości

Najprostszym przypisaniem jest zwykły znak równości

```
Edit1 = x + 1;
```

Oprócz zwykłego przypisania możemy wykonać dodatkowe działanie matematyczne na zmiennej stojącej po lewej stronie znaku równości.

Dostępne są następujące operatory:

+=

dodaje wartość po prawej stronie znaku równości do aktualnej zmiennej i przypisuje do niej wynik

```
x += 4;
```

jest równoważne operacji

```
x = x + 4;
```

lub

```
Edit1 += "dodatkowy tekst";
```

jest równoważne operacji

```
Edit1 = Edit1 + "dodatkowy tekst";
```

-=

odejmuje wartość po prawej stronie znaku równości od aktualnej zmiennej i przypisuje do niej wynik

```
x -= a + 2;
```

jest równoważne operacji

```
x = x - (a + 2);
```

***=**

mnoży wartość po prawej stronie znaku równości przez aktualną zmienną i przypisuje do niej wynik

```
x *= a + 2;
```

jest równoważne operacji

```
x = x * (a + 2);
```

/=

dzieli zmienną przez wartość po prawej stronie znaku równości i przypisuje do niej wynik

`x /= a + 2;`

jest równoważne operacji

`x = x / (a + 2);`

%=

oblicza modulo zmiennej i przypisuje do niej wynik

&=

dodaje bitowo wartość od aktualnej zmiennej i przypisuje do niej wynik

|=

wykonuje działanie OR wartości i aktualnej zmiennej i przypisuje do niej wynik

^=

wykonuje działanie XOR wartości i aktualnej zmiennej i przypisuje do niej wynik

>>=

przesuwa z prawo bity zmiennej i przypisuje do niej wynik

<<=

przesuwa w lewo bity zmiennej i przypisuje do niej wynik

Funkcje wbudowane

escape(expression:String) : String

Zamienia ciąg znaków na postać która może być przesłana jako argumenty w wywołaniu HTTP, tzn wszystkie znaki nie alfanumeryczne są zamieniane na kod szesnastkowy %

```
Edit1 = "sendmail.php?email=" + escape("user@domain.com");
```

unescape(x:String) : String

Zamienia ciąg znaków z postaci argumentów w wywołaniu HTTP na postać zwykłego tekstu

getTimer() : Number

Zwraca ilość milisekund od momentu uruchomienia filmu

getURL(url:String)

Otwiera link internetowy

getURL(url:String, window:String)

Otwiera link internetowy z podaniem parametru target

Przykład

```
getURL("http://www.selteco.com", "_blank");
```

otwiera adres www.selteco.com w nowym oknie

Parametry wywołania linku możemy podać po znaku ?

```
getURL("http://www.selteco.com?param1=value1&param2=value2", "_blank");
```

gotoAndPlay(scene:String)

skacze do ramki o nazwie scene aktualnego filmu lub sprita

Przykłady

```
gotoAndPlay("Frame 2");  
Sprite1.gotoAndPlay("Frame 2");  
_root.gotoAndPlay("Frame 2");
```

gotoAndPlay(frame:Number)

skacze do ramki o fizycznym indeksie frame, ilość ramek na scenie jest uzależniona od częstotliwości filmu, z reguły 20 ramek na sekundę filmu.

gotoAndStop(scene:String)

skacze do ramki o nazwie scene aktualnego filmu lub sprita i zatrzymuje się

Przykłady

```
gotoAndStop("Frame 2");  
Sprite1.gotoAndStop("Frame 2");  
_root.gotoAndStop("Frame 2");
```

gotoAndStop(frame:Number)

skacze do ramki o indeksie frame i zatrzymuje się

Skok do ramki nie następuje natychmiast. Wskaźnik skoku jest ustawiany w czasie wykonywania akcji i wykorzystywany po zakończeniu wykonywania akcji (np. kliknięcia myszką). Należy uważać aby nie zamazać wskaźnika skoku w komendach warunkowych. Na przykład dany kod jest poprawny:

```
if(warunek==1)
    gotoAndPlay("Frame 2");
else
    gotoAndPlay("Frame 3");
```

natomiast w przypadku takiego kodu

```
if(warunek==1) gotoAndPlay("Frame 2");
gotoAndPlay("Frame 3");
```

skok zostanie zawsze wykonany do ramki Frame 3 gdyż program ustawi wskaźnik skoku na Frame 3 niezależnie od warunku.

Można także przekazać adres skoku do ramki poprzez zmienną tekstową

```
nazwaramki = "Frame 3";
if(warunek==1) nazwaramki = "Frame 2";
gotoAndPlay(nazwaramki);
```

isNaN(expression:Object) : Boolean

Zwraca true jeżeli wartość jest nieliczbowa, false jeżeli jest liczbowa

```
i = "Ala ma kota";
Edit1 = isNaN(i);
```

Number(string:String) : Number

Zamienia ciąg znaków na postać numeryczną

Przykład

```
Edit3 = Number(Edit1) + Number(Edit2);
```

parseFloat(string:String) : Number

Zamienia ciąg znaków na liczbę

Przykład

```
Edit1 = parseFloat("3.5e6");
```

uzyskamy liczbę 3500000

parseInt(expression:String [, base:Number]) : Number

Zamienia liczbę całkowitą w danym systemie base np dwójkowym lub szesnastkowym

Przykłady

```
Edit1 = parseInt("101",2);
```

setInterval(functionName:Function, interval:Number) : Number

Tworzy powtarzalne wywołanie danej funkcji co daną ilość milisekund

Przykład, narysuj pole Edit1 i wklej kod ramki

```
Edit1 = 0;

function myInterval()
{
  Edit1 = Edit1 + 1;
}

setInterval(myInterval,100);
```

this

Zmienna ta użyta wewnątrz funkcji odnosi się do aktualnego obiektu którego metodą jest ta funkcja

```
function Constructor()
{
  this.attribut1 = "some text";
}

o = new Constructor();

Edit1 = o.attribut1;
```

typeof(expression) : String

Zwraca typ zmiennej

Ciąg znaków: string
Sprite: movieclip
Przycisk: object
Pole tekstowe: object
Liczba: number
Boolean: boolean
Obiekt: object
Funkcja: function
Wartość pusta: null
Wartość niezdefiniowana: undefined

Przykłady:

```
s = "12345";
Edit1 = typeof(s);
```

lub

```
s = 12345;
Edit1 = typeof(s);
```

lub

```
Edit1 = typeof(_root);
```

lub

Narysuj Sprite1 z kółkiem w środku i pole tekstowe Edit1, wklej kod ramki:

```
Sprite1.onPress = function ()
{
  Edit1 = "press";
}
Edit1 = typeof(Sprite1.onPress);
```

undefined

Wartość nieokreślona

```
if (someUndefinedVariable == undefined)
{
  Edit1 = "zmienna jeszcze nie istnieje";
}
```

unescape(string:String)

Odwraca działanie funkcji escape

```
Edit1 = "Your email is: " + unescape("user%40domain%2Ecom");
```

Przekazywanie zmiennych z kodu HTML

Parametry zdefiniowane w kodzie OBJECT oraz EMBED zostaną przekazane do ActionScript w postaci zmiennych o tej samej nazwie

Na przykład po wpisaniu zmiennej param1:

```
<OBJECT CLASSID="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000"
CODEBASE="http://download.macromedia.com/pub/shockwave/cabs/flash/swflash.cab#
version=8,0,0,0"
WIDTH="850" HEIGHT="134" >
<PARAM NAME="MOVIE" VALUE="Flash.swf?param1=Ala ma kota">
...
```

W czasie ładowania pliku Flash w ActionScript zostanie zdefiniowana zmienna param1, którą można wykorzystać np. w przypisaniu:

```
Edit1 = param1;
```

Pole tekstowe wyświetli napis "Ala ma kota"

Jeżeli używamy PHP możemy przekazać zmienną z PHP do Flash w ten sposób:

```
<?
$params = "Ala ma kota";
?>

<OBJECT>
...
<PARAM NAME="MOVIE" VALUE="Flash.swf?param1=<?echo $params;?>">
```

MovieClip

Jest to najczęściej używana klasa. Wszystkie obiekty typu Grupa lub Sprite są zdefiniowane jako obiekty MovieClip.

Główny film zdefiniowany jest jako obiekt o nazwie `_root`.

MovieClip._alpha : *Number*

Krycie obiektu od 0 do 100 %

MovieClip._currentframe : *Number*

Numer aktualnej ramki podczas odtwarzania

MovieClip._droptarget : *String*

Nazwa innego Sprita na który ten Sprite został przeciągnięty i upuszczony

MovieClip.enabled : *Boolean*

Wartość true jeżeli Sprite może odbierać zdarzenia myszy, w przeciwnym wypadku Sprite jest zablokowany

MovieClip.focusEnabled : *Boolean*

Wartość true jeżeli Sprite może odbierać zdarzenia klawiszy, w przeciwnym wypadku Sprite jest zablokowany

MovieClip._focusrect : *Boolean*

Jeżeli true Sprite posiada prostokąt naokoło oznaczający że przyjmuje zdarzenia klawiatury

MovieClip._framesloaded : *Number*

Ilość ramek Sprita dotychczas pobranych z internetu jeżeli jest ładowany z pliku zewnętrznego

MovieClip._height : *Number*

Wysokość Sprita w pikselach

MovieClip.hitArea : *MovieClip*

Wskaźnik do innego Sprita jeżeli Sprite ma inny obiekt służący jako pole aktywne przycisku

MovieClip._lockroot : *Boolean*

W przypadku gdy podfilm jest ładowany z pliku zewnętrznego `_lockroot = true` sprawia że odniesienia z podfilmu do obiektu `_root` odnoszą się do obiektu podfilmu a nie do filmu głównego który łąduje podfilm.

MovieClip.menu : *ContextMenu*

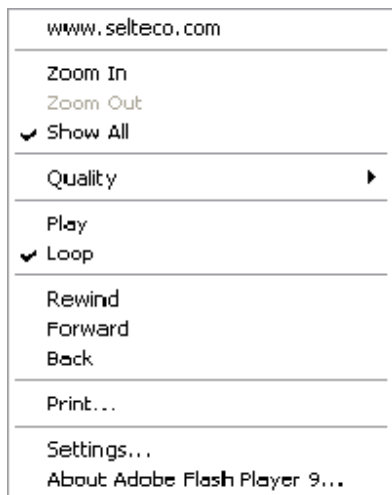
Obiekt menu kontekstowego (pod prawym klawiszem myszy) przypisanego do danego Sprita. Plik musi być wyeksportowany dla Flash Player 8 lub wyższej.

W tym przykładzie dodamy element menu dla głównego filmu

```
function goOnTheWeb()
{
    getURL("http://www.selteco.com", "_blank");
}

mymenu = new ContextMenu();
mymenu.customItems.push(new ContextMenuItem("www.selteco.com", goOnTheWeb));

_root.menu = mymenu;
```



MovieClip._name : *String*

Nazwa instancji Sprite

```
Edit1 = Sprite1._name;
```

MovieClip._parent : *MovieClip*

Wskaźnik do Sprita rodzica który zawiera tego sprite

MovieClip._quality : *String*

Jakość filmu:

"LOW" Niska jakość, szybkie wyświetlanie

"MEDIUM" Średnia jakość, bitmapy i tekst nie są optymalizowane

"HIGH" Domyślna jakość

"BEST" Najwyższa jakość, bitmapy są optymalizowane.

MovieClip._rotation : *Number*

Kąt obrotu Sprite

MovieClip._soundbuftime : *Number*

Opóźnienie w sekundach zanim buforowany dźwięk zacznie być odtwarzany

MovieClip.tabEnabled : *Boolean*

True jeżeli Sprite jest w łańcuch przełączania pomiędzy elementami klawiszem Tab

MovieClip.tabChildren : *Boolean*

True jeżeli dzieci Sprita mają także być włączone do cyklu przełączana klawiszem Tab

MovieClip.tabIndex : *Number*

Numer pozycji przełączania klawiszem Tab

MovieClip._target : *String*

Ścieżka absolutna do Sprita

```
Edit1 = Sprite1._target;
```

MovieClip._totalframes : *Number*

Całkowita ilość ramek Sprite

MovieClip.trackAsMenu : *Boolean*

Jeżeli true dany Sprite przejmie wszystkie zdarzenia zwolnienia klawisza myszy nawet spoza obszaru sprita

MovieClip._url : *String*

Adres internetowy z którego Sprite został załadowany

MovieClip.useHandCursor : *Boolean*

Jeżeli false Sprite będzie miał kursor strzałki zamiast kursora linku o ile jest dla niego zdefiniowana akcja myszy

```
Sprite1.useHandCursor = false;
```

MovieClip._visible : *Boolean*

Określa czy Sprite jest widoczny czy nie

Pokazanie sprite:

```
Sprite1._visible = true;
```

Ukrycie sprite:

```
Sprite1._visible = false;
```

MovieClip._width : *Number*

MovieClip._height : *Number*

Szerokość i wysokość Sprite w pikselach

MovieClip._x : *Number*

MovieClip._y : *Number*

Pozycja Sprite wewnątrz rodzica

MovieClip._xmouse : *Number*

MovieClip._ymouse : *Number*

Pozycja kursora myszy

```
function readmouse()
{
  Edit1 = _root._xmouse + ", " + _root._ymouse;
}
```

```
setInterval(readmouse,10);
```

MovieClip._xscale : *Number*

MovieClip._yscale : *Number*

Skala x i y Sprite w procentach, domyślnie 100

MovieClip.createEmptyMovieClip(instanceName:String, depth:Number) : *MovieClip*

Tworzy nowy i pusty obiekt Sprite o nazwie instanceName i na głębokości depth, większy depth chowa obiekt pod innymi obiektami

MovieClip.createTextField(instanceName:String, depth:Number, x:Number, y:Number, width:Number, height:Number)

Tworzy puste pole tekstowe o nazwie instanceName, na głębokości depth i o podanych wymiarach. Wymiary podane są w pikselach.

```
_root.CreateTextField("EditField1",10,20,20,500,40);  
EditField1.text = "Moje pole tekstowe";
```

MovieClip.duplicateMovieClip(newname:String, depth:Number) : MovieClip

Powiera Sprite i umieszcza go na nowej głębokości

```
Sprite1.duplicateMovieClip("Sprite2",100);  
Sprite2._x = Sprite1._x + 10;  
Sprite2._y = Sprite1._y + 10;
```

MovieClip.getBounds(targetCoordinateSpace:Sprite) : Object

Zwraca prostokąt zawierający elementy widoczne wewnątrz Sprite względem obiektu targetCoordinateSpace, lub względem siebie jeżeli parametr nie jest podany

```
rect = Sprite1.getBounds();  
Edit1 = rect.yMin + ", " + rect.yMax + ", " + rect.xMin + ", " + rect.xMax;
```

MovieClip.getBytesLoaded() : Number

Zwraca ilość bajtów załadowanych jeżeli plik pobierany jest z internetu

MovieClip.getBytesTotal() : Number

Zwraca całkowitą ilość bajtów Sprite

MovieClip.getDepth() : Number

Zwraca głębokość Sprite

MovieClip.getInstanceAtDepth(depth:Number) : MovieClip

Zwraca wskaźnik do sprita na danej głębokości

MovieClip.getNextHighestDepth() : Number

Zwraca wolną głębokość na której można umieścić nowy Sprite. Na każdej głębokości może być tylko jeden obiekt.

MovieClip.getSWFVersion() : Number

Zwraca numer wersji dla której dany sprite został przeznaczony jeżeli został załadowany z pliku zewnętrznego

MovieClip.getTextSnapshot() : String

Tworzy ciąg znaków z zawartości pól tekstowych w Spricie

MovieClip.getURL(URL:String [,window:String, method:String])

Otwiera dany link

URL: adres internetowy np <http://www.selteco.com>

window: _blank otwiera nowe okno przeglądarki, _self otwiera link w dotychczasowym oknie

method: ciąg POST lub GET jeżeli link zawiera parametry za znakiem ?, domyślną wartością jest GET

MovieClip.globalToLocal(point:Object)

Zamienia współrzędne punktu (x i y) globalnego na współrzędne wewnątrz Sprite

MovieClip.gotoAndPlay(framename:String)

Skacze do danej ramki o nazwie np "Frame 2" i rozpoczyna odtwarzanie

MovieClip.gotoAndStop(framename:String)

Skacze do danej ramki o nazwie np "Frame 2" i zatrzymuje film

MovieClip.hitTest(target:Object)

Zwraca true jeżeli Sprite pokrywa się (dotyka) Sprite podanego jako parametr target

MovieClip.hitTest(x:Number, y:Number, shapeFlag:Boolean)

Zwraca true jeżeli punkt x,y (dotyka) Sprite, parametr shapeFlag określa czy do obliczeń przyjąć elementy widoczne sprite czy cały prostokąt

MovieClip.loadMovie(url:String)

Ładuje plik SWF, FLV lub JPG

MovieClip.loadVariables(url:String)

Ładuje zmienne z pliku tekstowego, plik tekstowy musi zawierać zmienne w takiej postaci jak adresy URL

Przykład pliku .txt

```
var1="hello"&var2="goodbye"
```

MovieClip.localToGlobal(point:Object)

Zamienia współrzędne punktu (x i y) wewnątrz Sprite na współrzędne globalne

MovieClip.nextFrame()

Skacze do następnej ramki

MovieClip.play()

Rozpoczyna odtwarzanie filmu

MovieClip.prevFrame()

Skacze do poprzedniej rami

MovieClip.removeMovieClip()

Usuwa Sprite utworzonego wcześniej komendą `MovieClip.duplicateMovieClip()`

MovieClip.setMask(target:Sprite)

Ustala Sprite jako maskę dla innego Sprite

MovieClip.startDrag()

Rozpoczyna przesuwanie Sprite za pomocą myszy

MovieClip.startDrag([lock:Boolean, [left:Number, top:Number, right:Number, bottom:Number]])

Rozpoczyna przesuwanie Sprite za pomocą myszy ograniczając dostępny obszar za pomocą liczb left, top, right i bottom.

Parametr lock sprawia że środek Sprita pokrywa się ze środkiem kursora myszy

MovieClip.stop()

Zatrzymuje odtwarzanie Sprite

MovieClip.stopDrag()

Kończy przesuwanie Sprite za pomocą myszy

MovieClip.swapDepths(depth:Number)

MovieClip.swapDepths(target:String)

Zamienia 2 Sprite głębokościami na danej głębokości lub o danej nazwie

MovieClip.unloadMovie()

Usuwa z pamięci sprite załadowany dynamicznie z pliku zewnętrznego

Rysowanie w Spritach

MovieClip.beginFill(rgb:Number)

MovieClip.beginFill(rgb:Number, alpha:Number)

Określa kolor wypełnienia za pomocą kodu heksadecymalnego lub także krycia alpha

MovieClip.beginGradientFill(fillType:String, colors:Array, alphas:Array, ratios:Array, matrix:Object)

Określa wypełnienie przejściem tonalnym pomiędzy różnymi kolorami

fillType: "linear" lub "radial"

colors: tablica kolorów, maks 8 elementów

alphas: tablica kryć (od 0 przezroczysty do 255 całkowicie kryjący), maks 8 elementów

ratios: tablica pozycji kolorów przejścia tonalnego, wartości od 0 do 255, maks 8 elementów

MovieClip.clear()

Czyści to co zostało narysowane za pomocą komend rysujących

MovieClip.curveTo(cx:Number, cy:Number, x:Number, y:Number)

Rysuje krzywą Bezierra do punktu x,y i punkcie kontrolnym cx i cy

MovieClip.endFill()

Zamyka rozpoczęte linie i wypełnia krzywą kolorem określonym komendami MovieClip.beginFill() lub MovieClip.beginGradientFill().

MovieClip.lineStyle(thickness:Number, rgb:Number, alpha:Number)

Ustala nowy styl linii o grubości thickness, kolorze rgb i przezroczystości alpha

MovieClip.lineTo(x:Number, y:Number)

Rysuje linię prostą do punktu x, y

MovieClip.moveTo()

Ustala nową pozycję startową do rysowania

Obsługa zdarzeń w Spritach

Poniższe zdarzenia można przypisać do zdefiniowanych wcześniej funkcji.

MovieClip.onData : *Function*

Wywoływane w czasie pobrania danych

MovieClip.onDragOut : *Function*

Wywoływane gdy użytkownik wciśnie przycisk myszy wewnątrz sprite i przesunie mysz poza obszar

MovieClip.onDragOver : *Function*

Wywoływane gdy użytkownik wciśnie przycisk myszy na zewnątrz sprite i przesunie mysz na jego obszar

MovieClip.onEnterFrame : *Function*

Wywoływane przed wyświetleniem każdej ramki fizycznej

MovieClip.onKeyDown : *Function*

Wywoływane po naciśnięciu klawisza Użyj funkcji Key.getCode() i Key.getAscii() aby uzyskać kod klawisza

MovieClip.onKeyUp : *Function*

Wywoływane po zwolnieniu klawisza

MovieClip.onKillFocus : *Function*

Wywoływane gdy Sprite straci moliwaść przyjmowania zdarzeń klawiatury

MovieClip.onLoad : *Function*

Wywoływane zanim Sprite pojawi się pierwszy raz w filmie

MovieClip.onMouseDown : *Function*

Wywoływane po naciśnięciu lewego przycisku myszy

MovieClip.onMouseMove : *Function*

Wywoływane przy ruchu myszy

MovieClip.onMouseUp : *Function*

Wywoływane po zwolnieniu przycisku myszy

MovieClip.onPress : *Function*

Wywoływane po naciśnięciu lewego przycisku myszy

MovieClip.onRelease : *Function*

Wywoływane po zwolnieniu przycisku myszy

MovieClip.onReleaseOutside : *Function*

Wywoływane gdy użytkownik wciśnie przycisk myszy wewnątrz sprite, przesunie mysz poza obszar i zwolni przycisk myszy

MovieClip.onRollOut : *Function*

Wywoływane gdy wskaźnik myszy opuści obszar sprite

MovieClip.onRollOver : *Function*

Wywoływane gdy wskaźnik myszy wejdzie w obszar sprite

MovieClip.onSetFocus : *Function*

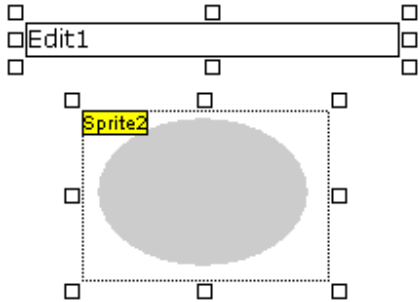
Wywoływane gdy sprite przejmie zdarzenia klawiatury

MovieClip.onUnload : *Function*

Wywoływane gdy sprite jest usunięty z filmu

Przykład:

Narysuj pole Edit1 i Sprite2, w sprite wstaw kółko:



wklej poniższy kod w ActionScript ramki:

```
Sprite2.onPress = function ()
{
    Edit1 = "onPress";
}

Sprite2.onRelease = function ()
{
    Edit1 = "onRelease";
}

Sprite2.onRollOut = function ()
{
    Edit1 = "onRollOut";
}

Sprite2.onRollOver = function ()
{
    Edit1 = "onRollOver";
}
```

MovieClipLoader

Klasa ta służy do obsługi Spritów (plików SWF lub JPEG) ładowanych z zewnętrznego pliku jeżeli niezbędna jest większa kontrola nad procesem ładowania pliku niż w funkcji `MovieClip.loadMovie()`. W tym przypadku należy użyć funkcji `MovieClipLoader.loadClip` zamiast funkcji `MovieClip.loadMovie`.

Klasa jest dostępna w przypadku eksportu pliku SWF w wersji Flash Player 8 lub wyższej. Wersję eksportu należy ustalić w projekcie komendą Film > Częstotliwość Ramki.

new MovieClipLoader()

Tworzy nowy obiekt tej klasy

MovieClipLoader.loadClip(url:String, target:MovieClip) : Boolean

Rozpoczyna pobieranie pliku url (w formacie SWF lub JPEG) celem umieszczenia go w obiekcie target

MovieClipLoader.getProgress(target:MovieClip) : Object

Zwraca obiekt zawierający ilość pobranych bajtów oraz ilość całkowitą bajtów w pobieranym pliku.

Zwrócony obiekt ma następujące atrybuty:

bytesLoaded : *Number*

Ilość bajtów pobranych

bytesTotal : *Number*

Całkowita ilość bajtów w pliku

MovieClipLoader.unloadClip(target:MovieClip)

Usuwa plik z pluginu i zwalnia pamięć

Zaawansowana obsługa zdarzeń

MovieClipLoader.addListener(listener:Object)

Dodaje listener do obsługi zdarzeń

new Object()

Tworzy nowy obiekt który może być użyty jako listener do obsługi zdarzeń związanych z ładowaniem filmu

Przykład, ustaw Flash Player 8 lub wyższy w Film > Częstotliwość Ramki, narysuj pole Edit1 oraz obiekt Sprite2, wklej kod ramki. W polu Edit1 pojawi się komunikat błędu:

```
loader = new MovieClipLoader();
listener = new Object();

listener.onLoadError = function(target, errorCode)
{
    Edit1 = "Error loading file: " + errorCode;
}

loader.addListener(listener);

loader.loadClip("http://www.non-existent-domain.com/error_file.jpeg",Sprite2);
```

MovieClipLoader.removeListener(listenerObject:Object)

Usuwa poprzednio dodany listener

Zdarzenia obiektu listener

Kolejność wywoływania zdarzeń w obiekcie MovieClipListener jest następująca:

MovieClipListener.onLoadStart : *function([target:MovieClip])*

W momencie nawiązania połączenia i rozpoczęcia przesyłu danych, opcjonalny parametr target wskazuje na Sprite użyty w funkcji loadClip.

MovieClipListener.onLoadProgress : *function (target: MovieClip, loadedBytes:Number, totalBytes:Number)*

W czasie przesyłu danych

MovieClipListener.onLoadComplete : *function([target:MovieClip])*

Po zakończeniu przesyłu

MovieClipListener.onLoadInit : *function([target:MovieClip])*

Po inicjalizacji obiektu Sprite ale przed jego wyświetleniem

MovieClipListener.onLoadError : *function(target:MovieClip, errorCode:String)*

W momencie wystąpienia błędu ładowania

ErrorCode ma następujące wartości

"URLNotFound" – nie można nawiązać połączenia, w tym przypadku onLoadError jest wywoływane zamiast onLoadStart

"LoadNeverCompleted" – pobieranie nie zakończyło się pomyślnie, w tym przypadku onLoadError jest wywoływane zamiast onLoadComplete

Przykład wyświetlania postępu przy pobieraniu dużego pliku JPEG. Narysuj pole Edit1 oraz obiekt Sprite2, wklej kod ramki, ustaw Flash Player 8 lub wyższy w Film > Częstotliwość Ramki.

```
loader = new MovieClipLoader();
listener = new Object();

listener.onLoadStart = function()
{
    Edit1 = "Connected";
}

listener.onLoadProgress = function(target, bytesLoaded, bytesTotal)
{
    Edit1 = "Loading: " + Math.round(bytesLoaded * 100 / bytesTotal) + "%";
}

listener.onLoadComplete = function()
{
    Edit1 = "Done!";
}

loader.addListener(listener);

loader.loadClip("http://www.selteco.com/flashdesigner/largebox.jpg",Sprite2);
```

Array

Klasa ta reprezentuje typ tablicy, zmiennych ustawionych w porządku do których można odwoływać się za pomocą indeksu w nawiasach [].

Array.concat(array1, array2, ...)

Łączy tablice

```
arr1 = new Array();
arr1[0] = "Julia";
arr1[1] = "Maria";

arr2 = new Array();
arr2[0] = "Sandra";
arr2[1] = "Pamela";

arr3 = arr1.concat(arr2);

Edit1 = arr3[3];
```

Array.join(separator)

Łączy elementy tablicy w ciąg tekstowy

```
arr1 = new Array();
arr1[0] = "Julia";
arr1[1] = "Maria";
arr1[2] = "Pamela";
Edit1 = arr1.join(", ");
```

Array.pop()

Usuwa ostatni element tablicy i zwraca jego wartość

```
arr1 = new Array();
arr1[0] = "Julia";
arr1[1] = "Maria";
arr1[2] = "Pamela";
Edit1 = arr1.pop();
```

Array.push()

Dodaje element do końca tablicy i zwraca nową ilość elementów

```
arr1 = new Array();
arr1[0] = "Julia";
arr1[1] = "Maria";
arr1.push("Pamela");
```

Array.reverse()

Odwraca kolejność elementów tablicy

```
arr1 = new Array();
arr1[0] = "Julia";
arr1[1] = "Maria";
arr1[2] = "Pamela";
arr1.reverse();
Edit1 = arr1.join(", ");
```

Array.shift()

Usuwa pierwszy element tablicy i zwraca jego wartość

```
arr1 = new Array();
arr1[0] = "Julia";
arr1[1] = "Maria";
arr1[2] = "Pamela";
Edit1 = arr1.shift();
```

Array.slice(start, end)

Wycina kawałek tablicy od elementu start do elementu end (nie włączając) i zwraca nową tablicę

```
arr1 = new Array();
arr1[0] = "Julia";
arr1[1] = "Maria";
arr1[2] = "Sandra";
arr1[3] = "Pamela";

arr2 = arr1.slice(1,3);

Edit1 = arr2.join(", ");
```

Array.sort()

Sortuje elementy tablicy

```
arr1 = new Array();
arr1[0] = "Maria";
arr1[1] = "Sandra";
arr1[2] = "Pamela";
arr1[3] = "Julia";

arr1.sort();

Edit1 = arr1.join(", ");
```

Array.sort(option)

Dostępne wartości parametru option

- 1 lub Array.CASEINSENSITIVE, wielkość liter bez różnicy
- 2 lub Array.DESENDING, kolejność odwrotna (malejąca)
- 4 lub Array.UNIQUESORT, błąd sortowania jeżeli są wartości identyczne w tablicy
- 8 lub Array.RETURNINDEXEDARRAY, zwraca tablicę posortowanych indeksów nie sortując oryginalnej tablicy
- 16 lub Array.NUMERIC, wartości cyfrowe w tablicy, inaczej algorytm posortuje liczbę 100 przed 99 gdyż 1 jest przed 9

Funkcja sortująca ma składnię

```
function sortuj(a, b)
{
    ... porównaj elementy a i b
    return 1 , 0 lub -1
}
```

Array.sort(compareFunction)

Funkcja sortująca compareFunction musi zwrócić 0 gdy elementy są identyczne, -1 gdy element a jest mniejszy od b, 1 gdy b jest mniejszy od a.

```
arr1 = new Array();
arr1[0] = 30;
arr1[1] = 4;
arr1[2] = 1;
arr1[3] = 16;

function sortuj(a,b)
{
    if(a<b) return -1;
    if(a>b) return 1;
    return 0
}

arr1.sort(sortuj);

Edit1 = arr1.join(", ");
```

Array.sortOn(fieldName)

Sortuje elementy względem pola w tablicy

Jeżeli tablica zawiera pola można użyć pola jako wartości do sortowania:

```
arr1 = new Array();
arr1[0] = new Object(); arr1[0].nazwa = "Maria"; arr1[0].wiek = 24;
arr1[1] = new Object(); arr1[1].nazwa = "Sandra"; arr1[1].wiek = 15;
arr1[2] = new Object(); arr1[2].nazwa = "Pamela"; arr1[2].wiek = 31;
arr1[3] = new Object(); arr1[3].nazwa = "Julia"; arr1[3].wiek = 22;

arr1.sortOn("wiek");

Edit1 = arr1[0].nazwa + ", " + arr1[1].nazwa + ", " + arr1[2].nazwa + ", " +
arr1[3].nazwa ;
```

Array.splice(start, count)

Usuwa elementy z tablicy.

```
arr1 = new Array();
arr1[0] = "Julia";
arr1[1] = "Maria";
arr1[2] = "Sandra";
arr1[3] = "Pamela";
arr1.splice(1,2);
```

```
Edit1 = arr1.join(", ");
```

Array.toString()

Konwertuje tablicę na ciąg tekstowy .

```
arr1 = new Array();
arr1[0] = 1;
arr1[1] = 10;
arr1[2] = 100;
arr1[3] = 1000;
```

```
Edit1 = arr1.toString();
```

Array.unshift()

Dodaje elementy na początku tablicy.

```
arr1 = new Array();
arr1[0] = "Julia";
arr1[1] = "Maria";
```

```
arr1.unshift("Sandra", "Pamela" );
```

```
Edit1 = arr1.join(", ");
```

Array.length

Zwraca ilość elementów w tablicy

```
imiona = new Array();
```

```
imiona[0] = "Julia";
imiona[1] = "Maria";
imiona[2] = "Sandra";
```

```
Edit1 = imiona.length;
```

Key

Klasa ta odpowiada za obsługę klawiatury.

Zanim zdarzenia klawiatury będą obsługiwane przez film, należy go najpierw uaktywnić w przeglądarce poprzez wciśnięcie przycisku myszy na obszarze filmu.

Key.addListener(newListener:Object)

Dodaje listener do obsługi zdarzeń przyciśnięcia i zwolnienia klawisza

Przykład

```
myListener = new Object();

myListener.onKeyDown = function ()
{
  Edit1 = "Key pressed";
}
myListener.onKeyUp = function ()
{
  Edit1 = "Key released.";
}

Key.addListener(myListener);
```

Key.getAscii() : Number

Zwraca kod ASCII ostatnio przyciśniętego klawisza

Key.getCode() : Number

Zwraca kod ostatnio przyciśniętego klawisza

Key.isDown(keycode:Number) : Boolean

Zwraca true jeżeli dany klawisz jest wciśnięty

Key.isToggled(keycode:Number) : Boolean

Zwraca true jeżeli Num Lock lub Caps Lock został przełączony.

Key.removeListener(listener:Object) : Boolean

Usuwa listener

Kody klawiszy

Dla uproszczenia klasa Key zawiera atrybuty odpowiadające kodom najczęściej używanych klawiszy

```
Key.BACKSPACE = 8
Key.CAPSLOCK = 20
Key.CONTROL = 17
Key.DELETEKEY = 46
Key.DOWN = 40
Key.END = 35
Key.ENTER = 13
Key.ESCAPE = 27
Key.HOME = 36
Key.INSERT = 45
Key.LEFT = 37
Key.PGUP = 33
Key.PGDN = 34
```

Przykład przesuwania sprita za pomocą klawiszy.

Narysuj Sprite1 i umieść w nim kółko, wyjdź ze sprita i wpisz kod ramki:

```
myListener = new Object();

myListener.onKeyDown = function ()
{
  if(Key.isDown(Key.LEFT)) Spritel._x = Spritel._x - 5;
  if(Key.isDown(Key.RIGHT)) Spritel._x = Spritel._x + 5;
  if(Key.isDown(Key.UP)) Spritel._y = Spritel._y - 5;
  if(Key.isDown(Key.DOWN)) Spritel._y = Spritel._y + 5;
}

Key.addListener(myListener);
```

Mouse

Mouse.addListener(newListener:Object)

Dodaje listener obsługujący zdarzenia myszy

Mouse.hide() : *Number*

Ukrywa kursor myszy, zwraca wartość true jeżeli kursor jest widoczny

Mouse.removeListener((listener:Object) : *Boolean*

Usuwa listener dodany przez addListener().

Mouse.show() : *Number*

Pokazuje kursor, zwraca czy kursor był widoczny przed wywołaniem tej funkcji

MouseListener.onMouseDown : *Function*

Funkcja wywoływana przy naciśnięciu przycisku myszy

MouseListener.onMouseMove : *Function*

Funkcja wywoływana przy ruchu myszy

MouseListener.onMouseUp : *Function*

Funkcja wywoływana przy zwolnieniu przycisku myszy

MouseListener.onMouseWheel : *Function*

Funkcja wywoływana przy obrocie scrollera myszy

Przykład wyświetlania aktualne pozycji myszy w polu Edit1

```
myListener = new Object();

myListener.onMouseMove = function ()
{
  Edit1 = _root._xmouse + ", " + _root._ymouse;
}

Mouse.addListener(myListener);
```

Button

Klasa Button odpowiada przyciskom utworzonym narzędziem "Przycisk"

Przyciski domyślnie mają nazwy ButtonObject1, ButtonObject2 itd. Aby wyświetlić nazwę przycisku należy go zaznaczyć i wcisnąć F2.

Aby przycisk został zdefiniowany dla kodu ActionScript opcja "Obiekt ActionScript" musi być zaznaczona.

Button._alpha : *Number*

Krycie przycisku w procentach od 0 do 100

Button.enabled : *Boolean*

Określa czy przycisk może przyjmować zdarzenia myszy i klawiatury

Przykład zablokowania przycisku przed kliknięciem, narysuj Button1 i wklej kod ramki:

```
ButtonObject1._alpha = 20;  
ButtonObject1.enabled = false;
```

Button._height : *Number*

Button._width : *Number*

Określa wymiary przycisku

Button._name : *String*

Nazwa obiektu przycisku

Button._rotation : *Number*

Obrót przycisku względem lewego górnego rogu

Button.tabEnabled : *Boolean*

True jeżeli przycisk jest w łańcuch przełączania pomiędzy elementami klawiszem Tab

Button.tabIndex : *Number*

Numer pozycji przełączania klawiszem Tab

Button._target : *String*

Ścieżka absolutna do przycisku

```
Edit1 = ButtonObject1._target;
```

Button.trackAsMenu : *Boolean*

Jeżeli true dany przycisk przejmuje wszystkie zdarzenia zwolnienia klawisza myszy nawet spoza obszaru sprita

Button.useHandCursor : *Boolean*

Jeżeli false przycisk będzie miał kursor strzałki zamiast kursora linku o ile jest dla niego zdefiniowana akcja myszy

```
ButtonObject1.useHandCursor = false;
```

Button._x : *Number*

Button._y : *Number*

Przesunięcie buttona względem aktualnej pozycji, domyślnie 0,0

Button._xmouse : *Number*

Button._ymouse : *Number*

Pozycja kursora myszy na przycisku

Button._visible : *Boolean*

Określa czy przycisk jest widoczny

Button.onDragOut : *Function*

Wywoływane gdy użytkownik wciśnie przycisk myszy wewnątrz przycisku i przesunie mysz poza obszar

Button.onDragOver : *Function*

Wywoływane gdy użytkownik wciśnie przycisk myszy na zewnątrz przycisku i przesunie mysz na jego obszar

Button.onKeyDown : *Function*

Wywoływane po naciśnięciu klawisza Użyj funkcji `Key.getCode()` i `Key.getAscii()` aby uzyskać kod klawisza

Button.onKeyUp : *Function*

Wywoływane po zwolnieniu klawisza

Button.onKillFocus : *Function*

Wywoływane gdy przycisk straci możliwość przyjmowania zdarzeń klawiatury

Button.onPress : *Function*

Wywoływane po naciśnięciu lewego przycisku myszy na przycisku

Button.onRelease : *Function*

Wywoływane po zwolnieniu przycisku myszy

Button.onReleaseOutside : *Function*

Wywoływane gdy użytkownik wciśnie przycisk myszy wewnątrz przycisku, przesunie mysz poza obszar i zwolni przycisk myszy

Button.onRollOut : *Function*

Wywoływane gdy wskaźnik myszy opuści obszar przycisku

Button.onRollOver : *Function*

Wywoływane gdy wskaźnik myszy wejdzie w obszar przycisku

Button.onSetFocus : *Function*

Wywoływane gdy przycisk przejmie zdarzenia klawiatury

Math

Klasa ta udostępnia funkcje i wartości matematyczne.

Math.abs(x:Number) : *Number*

Wartość absolutna liczby

```
Edit1 = Math.abs(-1.45);
```

Math.acos(x:Number) : *Number*

Oblicza akosinus.

Math.asin(x:Number) : *Number*

Oblicza asinus.

Math.atan(x:Number) : *Number*

Oblicza atangens.

Math.atan2(y:Number, x:Number) : *Number*

Oblicza kąt od punktu x,y do osi x w radianach (od -PI do PI)

Math.ceil(x:Number) : *Number*

Zaokrągla liczbę w górę do najbliższej wartości całkowitej

Math.cos(x:Number) : *Number*

Oblicza kosinus

Math.exp(x:Number) : *Number*

Funkcja exp

Math.floor(x:Number) : *Number*

Zaokrągla liczbę w dół do najbliższej wartości całkowitej

Math.log(x:Number) : *Number*

Oblicza logarytm naturalny

Math.max(x1:Number, x2:Number) : *Number*

Zwraca większą wartość z 2 liczb

Math.min(x1:Number, x2:Number) : *Number*

Zwraca mniejszą wartość z 2 liczb

Math.pow(x:Number, y:Number) : *Number*

Zwraca z podniesione do potęgi y

Math.random() : *Number*

Zwraca losową liczbę z zakresu 0.0 do 1.0.

Math.round(x:Number) : *Number*

Zaokrągla liczbę do wartości całkowitej

Math.sin(x:Number) : *Number*

Oblicza sinus

Math.sqrt(x:Number) : Number

Oblicza pierwiastek kwadratowy

Math.tan(x:Number) : Number

Oblicza tangens

Zmienne matematyczne

Wbudowane zmienne które mogą być wykorzystane w obliczeniach matematycznych

Math.E : Number

Podstawa logarytmu naturalnego (ok 2.718).

Math.LN2 : Number

Logarytm naturalny z 2 (ok 0.693).

Math.LOG2E : Number

ok 1.442.

Math.LN10 : Number

Logarytm naturalny 10 (ok 2.302).

Math.LOG10E : Number

ok 0.434

Math.PI : Number

Liczba PI (ok 3.14159).

Math.SQRT1_2 : Number

Pierwiastek kwadratowy z 1/2 (ok 0.707).

Math.SQRT2 : Number

Pierwiastek kwadratowy z 2 (ok 1.414).

Przykład:

```
Edit1 = "Pole koła o promieniu 5 wynosi " + Math.PI * Math.pow(5,2);
```

Date

Klasa ta reprezentuje obiekt z datą i godziną. Może być to czas aktualny lub dowolnie ustalony przez użytkownika.

Czas UTC to tak zwany uniwersalny czas skoordynowany, jest on niezależny od pór roku ani od strefy czasowej.

Czas lokalny to czas urzędowy uwzględniający zmiany czasu letniego i zimowego a także strefę czasową w której znajduje się komputer.

Na przykład dla czasu CET który obowiązuje w większości krajów Europy kontynentalnej w zimę jest 1 godzina różnicy pomiędzy czasem lokalnym a UTC a w lato 2 godziny.

new Date()

tworzy obiekt klasy Date z aktualnym czasem lokalnym

przykład wyświetlenia aktualnego roku

```
d = new Date();
Edit1 = d.getFullYear();
```

new Date(year:Number, month:Number [, date:Number [, hour:Number [, minute:Number [, second:Number [, millisecond:Number]]]]])

Tworzy obiekt klasy Date z podanym czasem lokalnym

year: rok

month: numer miesiąca od 0 do 11

date: dzień od 1 do 31

hour: godzina od 0 do 23

minute: minuta od 0 do 59

second: sekunda od 0 do 59

millisecond: 1/1000 sekundy od 0 do 999

Przykład utworzenia daty 12 lutego 1990 roku

```
mydate = new Date(1990, 1, 12);
```

Przykład obliczenia ilości dni pomiędzy 2 datami 1 stycznia 1980 a 14 marca 2009 roku

```
date1 = new Date(1980, 0, 1);
date2 = new Date(2009, 2, 14);
days = ( date2.getTime() - date1.getTime() ) / (1000 * 60 * 60 * 24);
Edit1 = days;
```

new Date(timeValue:Number)

Tworzy obiekt klasy Date z czasem podanym w formie milisekund które upłynęły od 1 stycznia 1970 roku czasu UTC

Przykład utworzenia czasu 3 sekund po 1 stycznia 1970 roku czasu UTC

```
d = new Date(3000);
Edit1 = d;
```

Date.getDate() : Number

Zwraca dzień miesiąca

Date.getDay() : Number

Zwraca dzień tygodnia

Date.getFullYear() : Number

Zwraca rok 4 cyfrowy

Date.getHours() : Number

Zwraca godzinę

Date.getMilliseconds() : Number

Zwraca milisekundy

Date.getMinutes() : Number

Zwraca minuty

Date.getMonth() : Number

Zwraca miesiąc

Date.getSeconds() : Number

Zwraca sekundy

Date.getTime() : Number

Zwraca milisekundy do północy 1 stycznia 1970 czasu UTC

Date.getTimezoneOffset() : Number

Zwraca różnicę czasu w sekundach do czasu UTC

Date.getYear() : Number

Zwraca rok

Date.getUTCDate() : Number

Date.getUTCDay() : Number

Date.getUTCFullYear() : Number

Date.getUTCHours() : Number

Date.getUTCMilliseconds() : Number

Date.getUTCMinutes() : Number

Date.getUTCMonth() : Number

Date.getUTCSeconds() : Number

Date.getUTCYear() : Number

Funkcje takie same ale zwracają czas przeliczony na UTC

Date.setDate() : Number

Date.setFullYear() : Number

Date.setHours() : Number

Date.setMilliseconds() : Number

Date.setMinutes() : Number

Date.setMonth() : Number

Date.setSeconds() : Number

Date.setTime() : Number

Date.setYear() : Number

Funkcje które modyfikują czas w obiekcie Date

Date.toString() : *String*

Zwraca czas w formacie ciągu znaków

Date.UTC() : *Number*

Ilość milisekund pomiędzy 1 stycznia, 1970, czasu UTC, a czasem zapisanym w obiekcie

Klasy zmiennych

Arguments

Obiekt reprezentujący wewnątrz funkcji listę jej parametrów

arguments.callee : *Function*

Wskaźnik do funkcji która wywołuje dana funkcję

arguments.caller : *Function*

Wskaźnik do funkcji wywoływanej

arguments.length : *Number*

Ilość parametrów

Przykład:

```
function getArgCount(param_arg1, param_arg2, param_arg3)
{
    return (arguments.length);
}

Edit1 = getArgCount("par1", "par2", "par3");
```

Boolean

Klasa ta reprezentuje zmienną typu boolean czyli prawda lub fałsz (true lub false)

Boolean.toString() : *String*

Zwraca tekstową reprezentację zmiennej ("true" lub "false")

Boolean.valueOf() : *Boolean*

Zwraca wartość obiektu ("true" lub "false")

System szesnastkowy

System ten najczęściej używa się do określania wartości koloru. Kolor zapisuje się w postaci liczby heksadecymalnej o 6 znakach.

W systemie heksadecymalnym pojedyncza cyfra zamiast 10 wartości przyjmuje wartości od 0 do 15, cyfry powyżej 9 oznaczają się literami alfabetu a,b,c,d,e,f lub A,B,C,D,E,F wielkość liter nie ma znaczenia.

Aby odróżnić liczby dziesiętne od heksadecymalnych należy poprzedzić je znakami 0x, Inaczej liczba szesnastkowa w której akurat nie ma cyfry o wartościach powyżej 9 mogłaby się pomylić z liczbą dziesiętną.

Przykłady liczb szesnastkowych oraz odpowiadające im wartości dziesiętne

```
0x2 = 2
0x9 = 9
0xF = 15
```

0x10 = 16
0x18 = 32
0xFF = 255

Kolor w grafice komputerowej określa się za pomocą 3 liczb odpowiadających intensywności barw czerwonej, zielonej i niebieskiej. Wszystkie kolory można uzyskać przez mieszanie barw podstawowych w odpowiednich proporcjach. Intensywność każdej barwy może przyjmować wartości od 0 oznaczającej brak koloru do 255 oznaczającej maksymalną jasność barwy składowej. Maksymalne świecenie wszystkich barw tworzy kolor biały a brak jakiegokolwiek świecenia tworzy kolor czarny.

Aby zakodować pojedynczy kolor składowy nazywany także kanałem, wystarczy 2 cyfry w kodzie szesnastkowym. Maksymalna wartość to 255 czyli 0xFF. Kod koloru tworzony jest przez podanie liczby szesnastkowej o 6 cyfrach:

0xRRGGBB

gdzie zamiast cyfr RR umieszcza się intensywność koloru czerwonego, GG zielonego a BB niebieskiego.

przykłady kolorów zapisanych w formie szesnastkowej:

0x000000 czarny
0xFFFFFFFF biały
0xFF0000 czerwony
0x00FF00 zielony
0x0000FF niebieski
0x808080 szary 50%

Color

Klasa ta służy do modyfikacji macierzy koloru Sprita. Za pomocą macierzy Sprite może zmieniać kolor lub przezroczystość.

Wzmocnienie koloru to procentowe zwiększenie danego kanału wszystkich elementów graficznych w Spricie, przesunięcie koloru jest dodawane do aktualnego kanału wszystkich elementów. Np ustalając przesunięcie koloru czerwonego na 255 a innych kolorów na -255 spowodujemy że wszystkie obiekty w spricie staną się czerwone niezależnie od tego jaki wcześniej miały kolor.

new Color(target:Sprite)

Tworzy nowy obiekt typu kolor powiązany z danym Spritem

Color.getTransform() : Object

Pobiera aktualną macierz koloru sprita. Jest to obiekt zawierający następujące atrybuty:

ra procent wzmocnienia koloru czerwonego (-100 do 100).
rb przesunięcie koloru czerwonego (-255 do 255).
ga procent wzmocnienia koloru zielonego (-100 do 100).
gb przesunięcie koloru zielonego (-255 do 255).
ba procent wzmocnienia koloru niebieskiego (-100 do 100).
bb przesunięcie koloru niebieskiego (-255 do 255).
aa procent wzmocnienia krycia (-100 do 100).
ab przesunięcie wartości krycia (-255 do 255).

Domyślna macierz zawiera wzmocnienie 100 i przesunięcie 0 dla każdego kanału.

Color.setTransform(matrix:Object)

Ustala nową macierz koloru sprita

Color.getRGB() : *Number*

Zwraca wartość liczbową odpowiadającą kodowi koloru składająca się z wartości przesunięć kanałów rb,gb i bb

Color.setRGB(0xRRGGBB:Number)

Ustawia przesunięcie kolouru w aktualnej macierzy na daną wartość liczbową (zapisuje je w polach rb,gb,bb)

Przykład, narysuj Sprite i w nim szare kółko. Wyjdź ze sprita i wpisz kod ramki:

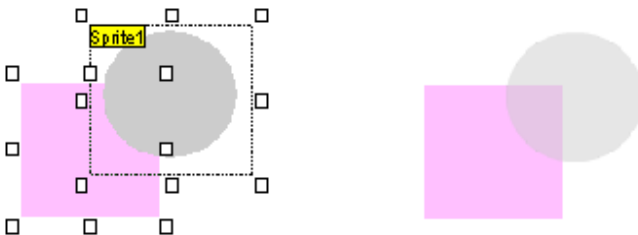
```
c = new Color(Sprite1);  
c.setRGB(0xFF0000);
```

Sprite zamieni się na kolor czerwony:



Przykład zmiany krycia do 50%

```
c = new Color(Sprite1);  
m = c.getTransform();  
m.aa = 50;  
c.setTransform(m);
```



Number

Klasa ta reprezentuje obiekt liczbowy

Number.toString()

Zwraca postać ciągu znaków

Number.valueOf()

Zwraca wartość liczbową obiektu

Number.MAX_VALUE

Największa możliwa wartość liczbowa, ok 1.79E+308.

Number.MIN_VALUE

Najmniejsza możliwa wartość liczbowa, ok 5e-324.

Number.NaN

Wartość wyrażenia do porównania czy obiekt jest liczbą, z ang. Not a Number (NaN).

Number.NEGATIVE_INFINITY

Wartość dodatniej nieskończoności

Number.POSITIVE_INFINITY

Wartość ujemnej nieskończoności

Sound

Klasa ta umożliwia manipulację dźwiękiem

new Sound([target:Sprite])

Tworzy nowy obiekt typu Sound

Jeżeli podamy parametr Sprite, obiekt ten kontroluje tylko dźwięki w tym Spricie

Sound.attachSound("idName":String)

Przypisuje do obiektu dźwięk o danym identyfikatorze. Domyślnie jest to nazwa pliku z którego został zaimportowany do projektu. Nazwę można zmienić w oknie "Film" > "Dźwięki"

Sound.getBytesLoaded()

Jeżeli dźwięk jest ładowany z pliku zwraca ilość pobranych bajtów

Sound.getBytesTotal()

Zwraca całkowitą wielkość pliku dźwiękowego

Sound.getPan()

Zwraca wartość balansu od -100 (lewy kanał) do 100 (prawy kanał)

Sound.getTransform()

Zwraca obiekt o następujących atrybutach

ll: natężenie lewej ścieżki w lewym głośniku
lr: natężenie lewej ścieżki w prawym głośniku
rl: natężenie prawej ścieżki w lewym głośniku
rr: natężenie prawej ścieżki w prawym głośniku

wartości od 0 do 100

Sound.getVolume()

Zwraca natężenie dźwięku od 0 do 100

Sound.loadSound(url:String)

Pobiera dźwięk MP3 z adresu internetowego

Sound.setPan(balance:Number)

Ustala balans od -100 do 100

Sound.setTransform(mixer:Object)

Ustala miksowanie kanałów
mixer to obiekt o atrybutach: ll,lr,rl,rr
patrz także getTransform()

Sound.setVolume(volume:Number)

Ustala natężenie dźwięku od 0 do 100

Sound.start()

Rozpoczyna odtwarzanie dźwięku od początku

start(secondOffset:Number)

Rozpoczyna odtwarzanie dźwięku od danej sekundy

start(secondOffset:Number, loop:Number)

Rozpoczyna odtwarzanie dźwięku od danej sekundy i daną ilością powtórzeń

Sound.stop()

Zatrzymuje dźwięk

Sound.duration

Długość dźwięku w milisekundach

Sound.id3

Wskaźnik do obiektu ID3 w pliku MP3 o ile jest obecny
Zawiera między innymi następujące atrybuty

| | |
|--------------------|---------------|
| Sound.id3.comment | Komentarze |
| Sound.id3.album | Tytuł albumu |
| Sound.id3.genre | Gatunek |
| Sound.id3.songname | Nazwa utworu |
| Sound.id3.artist | Wykonawca |
| Sound.id3.track | Numer ścieżki |
| Sound.id3.year | Rok wydania |

Dostępne są także atrybuty po nazwach zdefiniowanych dla specyfikacji ID3, między innymi:

| | |
|------|-------------------|
| COMM | Komentarz |
| TALB | Tytuł albumu |
| TBPM | Tempo (na minutę) |
| TCOM | Kompozytor |
| TCOP | Prawa autorskie |
| TDAT | Data |
| TEXT | Autor tekstu |

Sound.position

Pozycja aktualnego odtwarzania dźwięku w milisekundach

Sound.onID3

Funkcja wywoływana gdy dane ID3 są dostępne do odczytu

Sound.onLoad

Funkcja wywoływana gdy dźwięk się załaduje z pliku

Sound.onSoundComplete

Funkcja wywoływana gdy dźwięk skończy odtwarzanie

String

Klasa ta reprezentuje ciąg znaków alfanumerycznych. Znaki w ciągu indeksujemy od 0 (pierwszy znak ciągu do liczby o 1 mniejszej od długości ciągu)

Litery w ciągu "Pamela" będą miały następujące indeksy:

```
0 P
1 a
2 m
3 e
4 l
5 a
```

String.length : *Number*

Liczba określająca aktualną ilość znaków w ciągu

String.charAt(x:Number) : *String*

Zwraca znak na pozycji x (od 0)

String.charCodeAt(x:Number) : *Number*

Zwraca kod ASCII znaku w postaci liczby z pozycji x (od 0)

String.concat(val1:String, ... valN:String) : *String*

Tworzy i zwraca połączenie ciągu oraz podanych parametrów,

```
stringA = "Hello";
stringB = "World";
Edit1 = stringA.concat(" ", stringB);
```

uzyskamy tekst "Hello World"

String.fromCharCode(c1:Number,c2,...cN) : *String*

Zwraca ciąg składający się ze znaków w kodach ASCII

```
Edit1 = "dog"+String.fromCharCode(64)+"house.net";
```

uzyskamy tekst dog@house.net

String.indexOf(substring:String) : *Number*

Zwraca indeks pierwszego wystąpienia podciągu znaków od 0 na początku, lub -1 gdy podciąg nie został znaleziony

String.indexOf(substring:String, startIndex:Number) : *Number*

Zwraca indeks wystąpienia podciągu znaków począwszy od indeksu startIndex

String.lastIndexOf(substring:String) : *Number*

Zwraca indeks ostatniego wystąpienia podciągu znaków, lub -1 gdy podciąg nie został znaleziony

String.lastIndexOf(substring:String, startIndex:Number) : *Number*

Zwraca indeks ostatniego wystąpienia podciągu znaków zaczynając szukanie od indeksu startIndex

String.slice(start:Number) : *String*

Zwraca podciąg znaków od znaku start do końca

String.slice(start:Number, end:Number) : String

Zwraca podciąg znaków składających się od znaku start do znaku end

String.split("delimiter":String) : Array

Rozbija ciąg na podciągi dzielone za pomocą ciągu delimiter i zwraca tablicę ciągów

```
s = "Maria:Pamela:Sandra";  
a = s.split(":");  
Edit1 = a[1];
```

String.substr(start:Number) : String

Zwraca podciąg znaków od pozycji start do końca, jeżeli start jest liczbą ujemną zwraca podciąg liczony od końca

String.substr(start:Number, n:Number) : String

Zwraca podciąg n znaków od pozycji start

String.substring(start:Number, end:Number) : String

Zwraca podciąg znaków składających się od znaku start do znaku end nie włączając znaku end

String.toLowerCase() : String

Zwraca ciąg znaków zamienionych na małe litery bez zmiany oryginalnego obiektu

String.toUpperCase() : String

Zwraca ciąg znaków zamienionych na duże litery bez zmiany oryginalnego obiektu

Stage

Klasa Stage odpowiada obiektowi filmu Flash umieszczonemu w oknie przeglądarki

Stage.align : *String*

Wyrównanie obiektu Flash w oknie przeglądarki

"T" top center

"B" bottom center

"L" center left

"R" center right

"TL" top left

"TR" top right

"BL" bottom left

"BR" bottom right

Stage.height : *Number*

Stage.width : *Number*

Wysokość i szerokość filmu w pikselach

Stage.scaleMode : *String*

Tryb skalowania filmu w przeglądarce, dostępne wartości to "exactFit", "showAll", "noBorder" i "noScale"

Stage.showMenu : *Boolean*

True jeżeli całe menu kontekstowe jest dostępne, false jeżeli dostępne jest menu w postaci ograniczonej

Stage.addListener(myListener:Object)

Dodaje listener który sprawdza czy film został przeskalowany w przeglądarce

Stage.removeListener(myListener:Object) : *Boolean*

Usuwa listener dodany komendą addListener

Stage.onResize : *Function*

Wskaźnik do funkcji która otrzymuje powiadomienie o przeskalowaniu filmu w przeglądarce. Dodatkowo parametr scaleMode musi być ustawiony na "noScale".

System

System

System.setClipboard(string:String) : Boolean

Kopiuje ciąg tekstowy do schowka systemowego

System.showSettings()

Wyświetla panel ustawień dla przeglądarki Flash

System.showSettings(n:Number)

Wyświetla panel ustawień o zakładce numer n :

0 Privacy

1 Local Storage

2 Microphone

3 Camera

System.exactSettings : Boolean

True jeżeli ustawienia dostępu dotyczą tylko dokładnie danej domeny, false jeżeli dotyczą domenu i poddomen w danej domenie.

System.useCodepage : Boolean

Jeżeli false Flash traktuje zewnętrzne pliki tekstowe jako Unicode, true jeżeli pliki są zapisane w stronie kodowej. Dotyczy to plików wczytywanych przez klasę LoadVars.

System.onStatus : Function(genericError:Object)

Wywoływane w przypadku wystąpienia błędu pluginu Flash

System.security

Obiekt ten zawiera informacje o pozwoleniach dostępu dla zewnętrznych plików SWF ładowanych do głównego filmu.

System.security.allowDomain("domain1":String, "domain2", ... "domainN")

Zezwala plikom SWF załadowanym dynamicznie z podanych domen na dostęp do głównego filmu SWF. Np. jeżeli ładujemy zewnętrzny plik ze ścieżki `http://www.selteco.com/path/external_animation_file.swf` i chcemy aby ten plik miał dostęp do elementu `_root` w filmie głównym na pierwszej ramce należy wpisać:

```
System.security.allowDomain("www.selteco.com");
```

System.security.allowInsecureDomain("domain":String)

Zezwala plikom z domeny `domain` na dostęp do głównego pliku SWF jeżeli został on pobrany przez protokół HTTPS a plik ładowany jest ze ścieżki `http` a nie `https`.

System.security.loadPolicyFile(url: String)

Pobiera plik XML zezwoleń z danego adresu internetowego

Przykład pliku:

```
<cross-domain-policy>
<allow-access-from domain="*" to-ports="507" />
<allow-access-from domain="*.foo.com" to-ports="507,516" />
```

```
<allow-access-from domain="*.bar.com" to-ports="516-523" />
<allow-access-from domain="www.foo.com" to-ports="507,516-523" />
<allow-access-from domain="www.bar.com" to-ports="*" />
</cross-domain-policy>
```

System.capabilities

Obiekt ten zawiera informacje o możliwościach systemu na którym plik Flash jest wykonywany

System.capabilities.avHardwareDisable : *Boolean*

Czy kamera i mikrofon są dostępne dla Flasha

System.capabilities.hasAccessibility : *Boolean*

Czy system ma usprawnienia dla niepełnosprawnych

System.capabilities.hasAudio : *Boolean*

Czy system może odtwarzać dźwięk

System.capabilities.hasAudioEncoder : *Boolean*

Czy system może zapisywać dźwięk

System.capabilities.hasEmbeddedVideo : *Boolean*

Czy system może odtwarzać pliki wideo

System.capabilities.hasMP3 : *Boolean*

czy system może odtwarzać pliki MP3

System.capabilities.hasPrinting : *Boolean*

Czy system może drukować

System.capabilities.hasScreenBroadcast : *Boolean*

System.capabilities.hasScreenPlayback : *Boolean*

Czy sytem obsługuje komunikację serwera Flash Communication Server.

System.capabilities.hasStreamingAudio : *Boolean*

Czy system może odtwarzać dźwięk strumieniowy

System.capabilities.hasStreamingVideo : *Boolean*

Czy system może odtwarzać wideo strumieniowe

System.capabilities.hasVideoEncoder : *Boolean*

Czy system może zapisywać wideo do pliku np z kamery

System.capabilities.isDebugger : *Boolean*

Czy wersja pluginu jest wersją do debugowania

System.capabilities.language : *String*

Język systemu w postaci dwuliterowego kodu, np "en" angielski

System.capabilities.localFileReadDisable : *Boolean*

Czy zablokowany jest dostęp do systemu plików na dysku

System.capabilities.manufacturer : *String*

Wytwórca pluginu Flash

System.capabilities.os : *String*

System operacyjny

```
Edit1 = "System Version: " + System.capabilities.os;
```

System.capabilities.pixelAspectRatio : *Number*

Ilość pikseli fizycznych na logiczne monitora, z reguły 1

System.capabilities.playerType : *String*

Typ pluginu, dostępne wartości: "StandAlone", "External", "PlugIn", lub "ActiveX".

System.capabilities.screenColor : *String*

Kolor wyświetlacza, dostępne wartości: "color", "gray", "bw".

System.capabilities.screenDPI : *Number*

Rozdzielczość ekranu w pikselach na cal, z reguły 72

System.capabilities.screenResolutionX : *Number*

Pozioma rozdzielczość ekranu

System.capabilities.screenResolutionY : *Number*

Pionowa rozdzielczość ekranu

```
X = System.capabilities.screenResolutionX;
```

```
Y = System.capabilities.screenResolutionY;
```

```
Edit1 = "Resolution: " + X + " x " + Y;
```

System.capabilities.serverString : *String*

Ciąg zmiennych zakodowanych w postaci wywołania URL

System.capabilities.version : *String*

Wersja pluginu

```
Edit1 = "Flash Version: " + System.capabilities.version;
```

TextField

Klasa ta odpowiada polom tekstowym.

Dla pól tekstowych należy odróżnić nazwę zmiennej od nazwy obiektu pola. Do pól tekstowych jako obiektów odwołujemy się po ich nazwie a nie po nazwie zmiennej. Aby sprawdzić nazwę pola zaznacz je i wciśnij F2. Nazwa pola to z reguły EditField1, EditField2 itd.

Dodatkowo opcja Obiekt ActionScript musi być zaznaczona (po wciśnięciu F2)

TextField.autoSize : *Boolean*

Jeżeli true pole automatycznie rozciągnie się aby objąć cały tekst

TextField.background : *Boolean*

Pole ma jednolite tło, w przeciwnym razie jest przezroczyste

TextField.backgroundColor : *Number*

Kolor tła

```
EditField1.backgroundColor = 0xb0b0b0;
```

TextField.border : *Boolean*

Pole ma obramowanie

TextField.borderColor : *Number*

Kolor obramowania

```
EditField1.borderColor = 0xff0000;
```

TextField.bottomScroll : *Number*

Indeks ostatniej widocznej linii tekstu

TextField.condenseWhite : *Boolean*

Jeżeli true w polu o formacie HTML wszystkie znaki nowej linii i dodatkowe spacje zachowują się jak w standardzie HTML tzn są ignorowane

TextField.embedFonts : *Boolean*

Jeżeli true używana jest czcionka z pliku Flash jeżeli false czcionka systemowa

TextField._height : *Number*

Całkowita wysokość pola w pikselach

TextField.hscroll : *Number*

Pozycja tekstu przewiniętego w poziomie w pikselach

TextField.html : *Boolean*

Jeżeli true pole interpretuje znaczniki HTML

TextField.htmlText : *String*

Kod HTML pola, może zawierać następujące znaczniki:

 nowa linia

, <i>, <u> pogrubienie, kursywa, podkreślenie, należy zamykać: , </i>, </u>

 wypunktowanie

 nazwa czcionki, należy zamykać:
 kolor tekstu, należy zamykać:
 rozmiar czcionki, należy zamykać:

```
EditField1.html = true;  
Edit1 = "<b>bold text</b>";
```

TextField.length : *Number*
ilość znaków tekstu

TextField.maxChars : *Number*
maksymalna dopuszczalna ilość znaków w polu, null = brak ograniczenia

TextField.maxhscroll : *Number*
Maksymalna możliwa wartość przewinięcia poziomego

TextField.maxscroll : *Number*
Maksymalna możliwa wartość przewinięcia pionowego

TextField.menu : *ContextMenu*
wskaźnik do menu kontekstowego pola

TextField.mouseWheelEnabled
Jeżeli true pole obsługuje zdarzenia scrollera myszy

TextField.multiline : *Boolean*
jeżeli true pole może zawierać wiele linii

TextField._name : *String*
Nazwa obiektu pola

TextField._parent : *MovieClip*
Wskaźnik do sprita zawierającego to pole

TextField.password : *Boolean*
Pole typu hasło, znaki są maskowane

TextField.restrict : *String*
Zestaw znaków które użytkownik może wprowadzić do pola.

Przykłady:

Dopuszcza tylko cyfry
EditField1.restrict = "0123456789";

To samo co wyżej
EditField1.restrict = "0-9";

Dopuszcza tylko cyfry i duże litery
EditField1.restrict = "A-Z 0-9";

Znak ^ zabrania wprowadzania danego znaku

Zabrania wprowadzania znaku *
EditField1.restrict = "^*";

Zabrania wprowadzania cyfr

```
EditField1.restrict = "^0-9";
```

Jeżeli chcesz w wyrażeniu użyć znaku ^ lub - lub \ musisz poprzedzić go \

TextField._rotation : *Number*

Obrót pola tekstowego o dany kąt

TextField.scroll : *Number*

Pionowe przewijanie pola, indeks pierwszej widocznej linii

TextField.selectable : *Boolean*

Jeżeli true pozwala na zaznaczanie tekstu w polu

TextField.tabEnabled : *Boolean*

Jeżeli true pole uczestniczy w łańcuchu przełączania klawiszem tab

TextField.tabIndex : *Number*

Indeks elementu w łańcuchu przełączania klawiszem tab

TextField._target : *String*

Ścieżka absolutna do obiektu

TextField.text : *String*

Tekst w polu

TextField.textColor : *Number*

Kolor tekstu

TextField.textHeight : *Number*

TextField.textWidth : *Number*

Rozmiar tekstu wewnątrz pola

TextField.type : *String*

"input" tekst może być wprowadzany

"dynamic" tekst nie może być wprowadzany

TextField._url : *String*

Adres internetowy pliku który stworzył to pole

TextField.variable : *String*

Nazwa zmiennej powiązanej z polem, z reguły Edit1 dla pola EditField1

TextField._visible : *Boolean*

True jeżeli pole jest widoczne

TextField._width : *Number*

Całkowita szerokość pola w pikselach

TextField.wordWrap : *Boolean*

jeżeli true linie są przełamywane jeżeli przekraczają szerokość pola

TextField._x : *Number*

TextField._y : *Number*

Pozycja x i y pola

TextField._xmouse : *Number*

TextField._ymouse : *Number*

Pozycja kursora myszy

TextField._xscale : *Number*

TextField._yscale : *Number*

Skala pola w poziomie i pionie w procentach

TextField.addListener()

Dodaje listener do obsługi zdarzeń zmiany tekstu w polu

TextField.getFontList() : *Array*

Zwraca listę czcionek dostępnych w systemie jako tablicę

Metodę tę należy wywołać dla globalnej klasy TextField a nie dla jednego z pól

```
a = TextField.getFontList();
```

```
Edit1 = a.join();
```

TextField.getDepth()

Zwraca głębokość obiektu (czyli wzajemne nakładanie się)

TextField.removeListener() : *Boolean*

Usuwa listener

TextField.removeTextField()

Usuwa pole utworzone komendą MovieClip.createTextField()

TextField.replaceSel(text:String)

Zamienia tekst zaznaczony w polu nowym tekstem

TextField.replaceText(beginIndex:Number, endIndex:Number, text:String)

Zamienia tekst w polu od beginIndex do endIndex nowym tekstem

Fukncja dostępna w pluginie Flash Player 8 lub wyższej

Obsługa zdarzeń pola tekstowego

TextField.onChanged : *Function*

Funkcja wywoływana gdy pole zostało zmienione

Przykład, narysuj pola tekstowe Edit1 i Edit2, wklej kod ramki:

```
EditField1.onChanged = function ()
{
    Edit2 = Edit1;
}
```

cokolwiek napiszesz w polu Edit1 skopiuje się do pola Edit2

TextField.onKillFocus : *Function*

Funkcja wywoływana gdy pole straci przyjmowanie znaków z klawiatury

TextField.onScroller : *Function*

Funkcja wywoływana gdy pole zostało przewinięte

TextField.onSetFocus : *Function*

Funkcja wywoływana gdy pole rozpocznie przyjmowanie znaków z klawiatury

Przykład, narysuj pola tekstowe Edit1 i Edit2, wklej poniższy kod ramki:

```
EditField1.onChangeed = function()
{
  Edit2 = "tekst zmieniony";
}

EditField1.onKillFocus = function()
{
  Edit2 = "koniec wpisywania";
}

EditField1.onSetFocus = function()
{
  Edit2 = "zaczynij wpisywanie";
}
```

Przykład utworzenia pola które wyświetla podpowiedź, znikającą po kliknięciu na to pole:

```
EditField1.html = true;
Edit1 = "<i><font color='#808080'>Search</font></i>";

EditField1.onSetFocus = function()
{
  EditField1.html = false;
  Edit1 = "";
  EditField1.onSetFocus = 0;
}
```

Formatowanie tekstu

TextField.getNewTextFormat()

Tworzy i zwraca nowy obiekt formatujący tekst który zostanie zastosowany do nowo wprowadzanego tekstu

TextField.getTextFormat()

Zwraca domyślny obiekt formatujący tekst

TextField.getTextFormat(index: Number)

Zwraca obiekt formatujący tekst od znaku index

TextField.getTextFormat(start: Number, end: Number)

Zwraca obiekt formatujący tekst od znaku start do znaku end

TextField.setNewTextFormat(tf: TextFormat)

Ustala domyślne formatowanie tekstu

TextField.setNewTextFormat(index: Number, tf: TextFormat)

Ustala formatowanie tekstu od znaku index

TextField.setNewTextFormat(start:Number, end:Number, tf:TextFormat)

Ustala formatowanie tekstu od znaku start do znaku end

Klasa TextFormat

TextFormat.align : *String*

Wyrównanie tekstu do lewej, prawej lub wycentrowanie
Wartości "left", "right" lub "center"

TextFormat.blockIndent : *Number*

Wcięcie paragrafu w punktach, dotyczy wszystkich linii tekstu

TextFormat.bold : *Boolean*

Tekst jest pogrubiony

TextFormat.bullet : *Boolean*

Tekst ma wypunktowanie

TextFormat.color : *Number*

Kolor tekstu

TextFormat.font : *String*

Nazwa czcionki

TextFormat.indent : *Number*

Wcięcie pierwszej linii tekstu

TextFormat.italic : *Boolean*

Tekst jest kursywą

TextFormat.leading : *Number*

Odległość pionowa pomiędzy liniami tekstu

TextFormat.leftMargin : *Number*

Lewy margines tekstu

TextFormat.rightMargin : *Number*

Prawy margines tekstu

TextFormat.size : *Number*

Wielkość czcionki w punktach

TextFormat.tabStops : *Array[Number]*

Tablica pozycji tabulatorów w pikselach

TextFormat.underline : *Boolean*

Tekst podkreślony

TextFormat.url : *String*

TextFormat.target : *String*

Link internetowy i target linku, np `_self`, `_blank` itd

Przykład:

Narysuj nowe pole Edit1, kliknij je 2 razy i zaznacz opcję HTML, kliknij OK.

Wklej poniższy kod ramki:

```
Edit1 = "www.selteco.com - Kliknij";
```

```
tf = new TextFormat();  
tf.font = "Tahoma";  
tf.color = 0x0000ff;  
tf.bold = true;  
tf.url = "http://www.selteco.com";
```

```
EditField1.setTextFormat(0,15,tf);
```

CSS

TextField.StyleSheet

Klasa ta pozwala na formatowanie pola tekstowego za pomocą kodu CSS, styli kaskadowych.

Przykład stylu:

```
.heading { font-family: Arial, Helvetica, sans-serif; font-size: 24px; font-weight: bold; }  
.mainBody { font-family: Arial, Helvetica, sans-serif; font-size: 12px; font-weight: normal; }
```

TextField.StyleSheet.clear()

Usuwa formatowanie za pomocą stylów

TextField.StyleSheet.getStyle(styleName:String) : Object

Zwraca obiekt stylu o nazwie styleName z atrybutami np. fontWeight = bold
FontSize = 24px, fontFamily = Arial, Helvetica, sans-serif itd

Przykład

```
css = new TextField.StyleSheet();  
  
css.parseCSS(".header { font-size:24pt; color:#0000FF; font-family:times;}");  
  
headerObject = css.getStyle(".header");  
  
Edit1 = headerObject.fontSize;
```

TextField.StyleSheet.getStyleNames() : Array

Zwraca tablicę nazw styli, na przykład „.heading”, „.mainBody”

TextField.StyleSheet.load(url:String)

Pobiera style z adresu internetowego

TextField.StyleSheet.parseCSS(cssText:String) : Boolean

Tworzy styl z zawartości ciągu tekstowego, zwraca false w przypadku błędu

TextField.StyleSheet.setStyle(name:String, style:Object)

Dodaje styl o nazwie name i zawartości style do kolekcji

```
my_styleSheet:TextField.StyleSheet = new TextField.StyleSheet();  
styleObj = new Object();  
styleObj.color = "#000000";  
styleObj.fontWeight = "bold";  
my_styleSheet.setStyle("emphasized", styleObj);
```

TextField.StyleSheet.transform(style:Object) : TextFormat

Zamienia obiekt typu styleSheet na obiekt TextFormat

TextField.StyleSheet.onLoad : Function(success:Boolean)

Funkcja wywoływana po załadowaniu styli z pliku, success jest true jeżeli operacja zakończyła się pomyślnie.

Przykład tworzenia stylu i dodania go do pola edycji.

Narysuj pole Edit1 i wklej kod ramki

```
css = new TextField.StyleSheet();  
css.parseCSS(".header { font-size:24pt; color:#0000FF; font-family:times;}");  
EditField1.styleSheet = css;  
  
EditField1.html = true;  
EditField1.multiline = true;  
  
Edit1 = "<p class=\"header\">The Dog</p><p>The dog is brown</p>";
```

XML

Klasa ta pozwala na wczytywanie i manipulacje plikami XML.

Plik XML składa się z tzw tagów:

Przykład pliku XML

```
<globe name="World">
  <continent code="na">North America</continent>
  <continent code="sa">South America</continent>
  <continent code="eu">Europe</continent>
  <continent code="af">Africa</continent>
  <continent code="as">Asia</continent>
  <continent code="au">Australia</continent>
</globe>
```

Plik ten zawiera węzeł główny globe i 6 podwęzłów continent, każdy z nich ma atrybut code

XML.attributes : *Array*
Obiekt z atrybutami aktualnego węzła

XML.childNodes : *Array*
Tablica podwęzłów

XML.firstChild : *XMLNode*
Wskaźnik do pierwszego podwęzła

XML.ignoreWhite : *Boolean*
Jeżeli true puste węzły są ignorowane

XML.lastChild : *XMLNode*
Wskaźnik do ostatniego podwęzła

XML.loaded : *Boolean*
Określa czy plik jest już załadowany w całości

XML.nextSibling : *XMLNode*
Wskaźnik do następnego węzła na tym samym poziomie

XML.nodeName : *String*
Nazwa węzła, zawarta pomiędzy nawiasami < >

XML.nodeType : *Number*
Typ węzła, 1 węzeł < >, 3 węzeł tekstowy pomiędzy < > a </ >

XML.nodeValue : *String*
Wartość węzła w przypadku węzła tekstowego (nodeType == 3)

XML.parentNode : *XMLNode*
Wskaźnik do rodzica

XML.previousSibling : *XMLNode*
Wskaźnik do poprzedniego węzła na tym samym poziomie

XML.status : *Number*

Stan przetwarzania pliku XML

0 Brak błędu

-2 Sekcja CDATA nie ma zamknięcia

-3 Deklaracja XML nie ma zamknięcia

-4 Deklaracja DOCTYPE nie ma zamknięcia

-5 Komentarz nie ma zamknięcia

-6 Błędny element

-7 Brak pamięci

-8 Atrybut nie ma zamknięcia

-9 Brak odpowiadającego taga zamykającego

-10 Brak odpowiadającego taga otwierającego

XML.xmlDecl : *String*

Deklaracja pliku XML o ile istnieje

XML.setRequestHeader(headerName:String, headerValue:String)

W przypadku gdy plik jest pobierany z internetu tu można dodać dodatkowe nagłówki i parametry wywołania

XML.appendChild(childNode:XMLNode)

Dodaje podwęzeł na końcu listy

XML.cloneNode(deep:Boolean) : *XMLNode*

Kopiuje i zwraca węzeł wraz z podwęzłami do głębokości deep

XML.createElement(name:String) : *XMLNode*

Tworzy i zwraca nowy element drzewa o nazwie name

XML.createTextNode(text:String) : *XMLNode*

Tworzy węzeł tekstowy

XML.getBytesLoaded() : *Number*

Zwraca ilość bajtów pobranych przy czytaniu pliku

XML.getBytesTotal() : *Number*

Zwraca rozmiar pliku XML

XML.hasChildNodes() : *Boolean*

Zwraca true jeżeli aktualny węzeł ma podwęzły

XML.insertBefore(childNode:XMLNode, beforeNode:XMLNode)

Wstawia węzeł przez inny węzeł

XML.load(url:String)

Ładuje plik XML z internetu

XML.parseXML(source:String)

Przetwarza dane XML z ciągu tekstowego

XML.removeNode()

Usuwa węzeł

XML.send(send(url:String, [target:String]))

Wysyła dane XML do pliku na stronie www

XML.sendAndLoad(url:String, targetXMLObject:XML)

Wysyła plik XML na adres www i pobiera odpowiedź serwera w formacie XML do innego obiektu XML

XML.toString() : String

Zwraca dane XML w formacie tekstowym

XML.docTypeDecl: String

Deklaracja !DOCTYPE pliku XML

XML.onData : function ()

Funkcja wywoływana gdy skończy się pobieranie pliku xml

XML.onLoad : function (success:Boolean)

Funkcja wywoływana w czasie pobierania pliku xml

Przykład 1

narysuj pole Edit1 i wklej ten kod ramki

```
str = "<root><node/></root>";
```

```
xml = new XML(str);
```

```
rootNode = xml.firstChild;
```

```
Edit1 = rootNode.nodeName;
```

Przykład przechodzenia przez drzewo XML

narysuj pole Edit1 i wklej poniższy kod ramki:

```
str = "<globe name=\"World\">Continents<continent code=\"na\">North  
America</continent><continent code=\"sa\">South America</continent><continent  
code=\"eu\">Europe</continent><continent  
code=\"af\">Africa</continent><continent  
code=\"as\">Asia</continent><continent  
code=\"au\">Australia</continent></globe>";  
  
xml = new XML(str);  
  
globeNode = xml.firstChild;  
  
Edit1 = "Status: " + xml.status + " ";  
  
Edit1 = Edit1 + globeNode.nodeName + ", " + globeNode.attributes.name + ":"  
";  
  
continentNode = globeNode.firstChild;  
  
while(continentNode!=null)  
{  
    if(continentNode.nodeType==1)  
    {  
        Edit1 = Edit1 + continentNode.nodeName;  
        Edit1 = Edit1 + " [" + continentNode.attributes.code + "] ";  
  
        continentText = continentNode.firstChild;  
        Edit1 = Edit1 + continentText.nodeValue + ", ";  
    }  
    continentNode = continentNode.nextSibling;  
}
```

Kod ten utworzy następujący wynik:

Status: 0 globe, World: continent [na] North America, continent [sa] South America, continent [eu] Europe, continent [af] Africa, continent [as] Asia, continent [au] Australia,

LoadVars

Klasa ta pozwala na import parametrów do pliku SWF z pliku tekstowego. Parametry w pliku tekstowym są w postaci
param1=value1¶m2=value2 .. itd

Po załadowaniu zmiennych możemy się do nich odwoływać poprzez nazwę atrybutu obiektu LoadVars np
LoadVars.param1

new LoadVars()
Tworzy nowy obiekt

LoadVars.setRequestHeader(headerName:String, headerValue:String)
Dodaje dodatkowe nagłówki do wywołania pliku przez internet

LoadVars.decode(params:String)
Wczytuje i przetwarza na zmienne ciąg znaków w postaci param1=value1¶m2=value2

LoadVars.getBytesLoaded() : *Number*
Ilość dotychczas załadowanych bajtów funkcjami LoadVars.load() lub LoadVars.sendAndLoad()

LoadVars.getBytesTotal() : *Number*
Całkowita wielkość pliku ze zmiennymi

LoadVars.load(url:String) : *Boolean*
Wczytuje dane zmiennych z danego adresu, zmienne muszą być w postaci tekstowej:
param1=value1¶m2=value2 .. itd

LoadVars.send(url:String) : *Boolean*
Wysyła zmienne do danego adresu internetowego w postaci ciągu
url?param1=value1¶m2=value2

LoadVars.sendAndLoad(url:String) : *Boolean*
Wysyła zapytanie pod dany adres www i wczytuje odpowiedź serwera

LoadVars.toString() : *String*
Zwraca parametry w postaci
param1=value1¶m2=value2

LoadVars.contentType : *String*
Typ MIME danych

LoadVars.loaded : *Boolean*
Zwraca true jeżeli dane zostały pobrane w całości

LoadVars.onData : *function*
Funkcja wywoływana gdy zakończy się pobieranie danych

LoadVars.onLoad : *function*
Funkcja wywoływana w czasie pobierania danych

Przykład:

```
lv = new LoadVars();  
lv.decode("name=Pamela&age=25");  
Edit1 = lv.name + " is " + lv.age + " old.";
```

Funkcje nie obsługiwane w programie Alligator Flash Designer

trace()

Funkcja trace nie działa w standardowym pluginie Flash w przeglądarce. Zamiast trace(zmienna) należy użyć:

```
Edit1 = zmienna;
```

break, continue

Powodują wyjście z pętli lub przejście do początku pętli lecz są ignorowane

case

Należy użyć zestawu funkcji if

class

Aktualnie nie można definiować własnych klas

for in

Aktualnie nie można enumerować atrybutów obiektu

? :

Instrukcja warunkowa nie jest obsługiwana

i = j = 2;

wielokrotnie przypisanie nie jest obsługiwane, należy użyć:

```
j = 2;  
i = j;
```

{ }

Inicjalizator atrybutów obiektów

Zamiast

```
object = { attr1 : "value1", attr2 : "value2" }
```

Należy użyć

```
object = new Object();  
object.attr1 = "value1";  
object.attr2 = "value2";
```