Podstawy Flash ActionScript. Dodawanie obsługi zdarzeń do animacji. Tworzenie przycisków.

ActionScript to język skryptowy pozwalający na tworzenie interaktywnych animacji. Jego debiut miał miejsce wraz z narzędziem Macromedia Flash 3. Ograniczał się on jedynie do obsługi prostych zdarzeń wejścia od myszy oraz pozwalał na ładowanie plików zewnętrznych (np. nawigacja po animacji). Od początku firma Macromedia dążyła do tego, by język ten był podobny składniowo do JavaScript lecz jednocześnie szybszy i bardziej funkcjonalny. W narzędziu Macromedia Flash MX 2004 premierę miała druga wersja AS. Jeszcze bardziej zbliżono język do JavaScript, dodano obsługę klas i obiektów co z kolei zaowocowało składnią „kropkowaną" - np. nazwa obiektu.wlasciwosc . Niestety język nadal był podczas tworzenia obiektu swf przetwarzany do formy AS 1.0 i dopiero wykonywany. W związku z tym obie wersje języka pracowały na tej samej wirtualnej maszynie (AcrionScript Virtual Machine 1 – AVM1).

Wraz z narzędziem Adobe Flash CS3 Professional zadebiutowała 3, prawdopodobnie ostatnia wersja AcrionScript. Wersja ta diametralnie różni się od wersji 1.0 oraz 2.0. Zmiany pomiędzy tymi wersjami są tak duże, że firma Adobe musiała zmienić „silnik" obsługujący ten język – od teraz jest to AcrionScript Virtual Machine 2 (AVM2 bądź AVM+). Animacje korzystające z poprzednich wersji języka nie są kompatybilne z nową wersją – nie można więc ich w żaden sposób łączyć ze sobą (komunikować czy np. dołączać jako symbole do nowych projektów).

Wiele osób korzystających wcześniej z narzędzia Flash bardzo niechętnie podeszło do nowej wersji języka. Oznacza ono dla nich naukę od nowa. Dlatego tworzą oni najczęściej animacje dla AcrionScript 2.0 i zapewne na nim poprzestaną (nawet najnowsza wersja Adobe Flash CC posiada możliwość obsługi wcześniejszej wersji tego języka). Oczywiście nie jest to rozsądne podejście – nowy język pozwala pisać o wiele efektywniejsze animacje, łata wiele niedociągnięć poprzedników, łata luki krytyczne poprzedniej wirtualnej maszyny, które pozwalały na zdalne przejęcie komputera itd. itp. Poza tym ta wersja języka jest nadal rozwijana, a tym samym wspierana przez firmę Adobe. AS 3.0 to w pełni obiektowy język. Wraz z nim do standardu AS dodano:
- obsługę wyjątków – pomaga w odpluskwianiu projektu
- dynamiczne typy zmiennych – możliwość zmiany typu zmiennej w trakcie wykonywania animacji
- zamknięte klasy – statyczne definicje klas zwiększające efektywność kodu
- izolowane metody – metody są na stałe związane z danymi instancjami swoich klas
- E4X – pozwala na proste, dynamiczne tworzenie treści XML
- wyrażenia regularne – dodano obsługę tychże wyrażeń
- przestrzenie nazw – pozwala na wielokrotną definicję zmiennych o tych samych nazwach, lecz w różnych przestrzeniach nazw
- dodanie typu integer (int) oraz unsigned int (uint) – przyspieszone, efektywniejsze obliczenia dla liczb całkowitych
- nowy model listy wyświetlania
- nowy model zdarzeń – model od teraz bazuje na nasłuchiwaniu zdarzeń dla danych obiektów + obsługuje

Tak jak PHP czy JavaScript, tak i ActionScript posiada swoje cechy.
1. Zmienne.

Zmienne w AS deklaruje się podobnie jak w JS. Różnica polega na tym, że każda zmienna musi posiadać swój typ. Chociaż trzecia wersja wprowadza typ dynamiczny (jaki ma miejsce e JS oraz PHP) to on także musi zostać zadeklarowany. Ponadto każda deklaracja zmiennej musi zostać poprzedzona słowem kluczowym var. Przykłady deklaracji zmiennych:

```
var dowolna:*; //zmienna dowolnego (dynamicznego) typu; domyśla wartość niesprecyzowana
var bool:Boolean; //zmienna typu prawda/fałsz; domyślnie fałsz (0)
var numeryczna:Number; //typ liczbowy (całkowite, zmiennoprzecinkowe); domyślnie NaN
```

```
var calkowita:int; //typ liczb całkowitych; domyślnie 0
var naturalna:uint; //typ liczb całkowitych dodatnich; domyślnie 0
var ciag:String; //zmienna ciągu znakowego; domyślnie null
var object:Object; //zmienna typu obiektowego; domyślnie null
var pusta:void; //typ pusty; nie ma najmniejszego sensu czegoś takiego deklarować (przydaje się
                //przy funkcjach)
```

Przy wartościach liczbowych każda wartość inna niż liczba powoduje nadanie zmiennej wartości NaN (Not a Number). W praktyce oznacza to, iż zmienna ta posiada jakąś wartość lecz nie można jej uznać dwóch zmiennych tej wartości za takie same. Jak więc sprawdzać, czy zmienna nie osiągnęła takiej wartości? Można tego dokonać poprzez odpowiednią, wbudowaną funkcję:

```
isNaN(zmienna_liczbowa)
```

wartość true oznacza, że wartość zmiennej nie posiada liczbowego odpowiednika.

Istnieje wyjątek od reguły przy deklaracji nowych zmiennych, przy którym nie trzeba (wręcz nie wolno) używać słowa var. Mianowicie nie trzeba go używać gdy deklarowana jest zmienna wewnątrz obiektu typu dynamicznego:

```
var mojObiekt:Object = new Object();
mojObiekt.nowaZmienna = 56;
```

Trzeba pamiętać, że nie można zadeklarować dwa razy tej samej zmiennej:

```
var x:int = 5;
var x:String =”Napis”; //błąd – zmienna x została zadeklarowana już wcześniej!
```

Zasięg zmiennych liczbowych w ActionScript 3.0:

Typ	Akceptowalne wartości
Number	4.9406564584124654e-324 do 1.79769313486231e+308
int	-2147483648 do 2147483647 (tylko liczby całkowite)
uint	0 to 4294967295 (tylko liczby całkowite)

Usuwanie zmiennych i ich wartości w czasie życia skryptu:

```
//deklaracja nowego dynamicznego obiektu
var mojObiekt:Object = new Object();
mojObiekt.nowaZmienna = 56;

//usunięcie zmiennej dynamicznej dynamicznego obiektu
delete mojObiekt.nowaZmienna;

//obiektu nie da się usunąć
mojObiekt = null;

//typów podstawowych także nie da się usuwać
//bool
bool = false;
//int, uint
```

```
liczba = 0;
//typ numeryczny
liczba = NaN;
//typ znakowy
str = null;
```

Podobnie jak JavaScript oraz PHP, ActionScript także posiada możliwosc tworzenia zmiennych tablicowych. Najprostszą formą zmiennej tablicowej jest stworzenie jej w taki sposób:

```
var talica:Array = new Array(1,2,3,4);
```

Inne metody zostaną przedstawione później, w ramach opisywania klas i obiektów.

2. Funkcje

Funkcje w AS należy rozpatrywać tak samo jak to miało miejsce w JS oraz PHP. Oznacza to, iż w funkcjach umieszcza się fragment kodu, który można później wielokrotnie wykorzystywać w kolejnych częściach programu. ActionScript, podobnie jak JavaScript (oraz jak aktualnie także C++ czy Java), pozwala deklarować dwa rodzaje funkcji:
- nazwane, posiadające swoją nazwę, parametry oraz kod; są to klasyczne funkcje pozwalające na na wywołanie ich w dowolnym momencie życia naszej aplikacji. Zaletą tego rozwiązania jest możliwość nadawania ich parametrom różnych wartości w różnych częściach kodu. Przykład:

```
//funkcja zwróci potege wprowadzonej liczby
function potega(liczba:Number):Number {
	return liczba * liczba;
}
```

- nienazwane – funkcje nie posiadające swoich nazw. Tak jak ich nazwane odpowiedniki mogą posiadać parametry jednak ich wywołanie jest jednorazowe – tylko w chwili ich pisania; najbardziej charakterystycznym momentem ich wywołania jest po prostu przypisanie ich wartości zwrotnej do danej zmiennej. Innym rozwiązaniem jest podanie ich definicji w parametrze innej funkcji, która korzysta z wywołania callback (czyli funkcja z podanego parametru wykonuje się w chwili, gdy funkcja ją wywołująca kończy swoje działanie – jest to tzw. wykonywanie synchroniczne). Przykład:

```
//funkcja zwróci potęgę wprowadzonej przez parametr liczby i przypisze ten wynik do zmiennej

var liczba:Number = 144;
var wynik:Function = function(liczba):Number {
	return liczba * liczba;
}
```

Funkcje bez słowa kluczowego return nie zwracają nic. W takim wypadku możemy je deklarować w ten sposób:

```
function funkcja():void {
	//kod, który ma się wykonać
}
//funkcja nic nie zwróci (brak słowa return, typ zmiennej zdefiniowany na pusty)!
```

Jeżeli próbowalibyśmy wywołać funkcję, która w deklaracji nie posiada argumentów z

jakimkolwiek argumentem, np.

funkcja(3);

to kod spowoduje błąd w wykonywaniu animacji bądź/i zostanie przerwane wykonywanie wszystkich skryptów (tak jak w JavaScript).

Flash, tak jak i PHP, pozwala na podawanie predefiniowanych wartości parametrów funkcji. Jeżeli przy wywołaniu funkcji podamy własną wartość, wtedy to ona zostanie wykorzystana w ciele funkcji; w przeciwnym wypadku funkcja przyjmie wartość domyślną. Przykład:

```
function mojaFunkcja(napis:String = ”Tekst zastępczy”):void {
	trace(napis);
}

mojaFunkcja();
mojaFunkcja(”Witaj w ActionScript”);
```

INFORMACJA: Użyta w przykładzie funkcja trace służy jako tzw. funkcja odpluskwiająca. Pozawala ona śledzić wartości podanych do niej parametrów – zobaczyć jaką wartość przyjmują bądź czym są (w przypadku projektów). Powyższy przypadek pozwala po prostu wyświetlić napis, który podaliśmy pod zmienną.

Oczywiście funkcje mogą przyjmować więcej niż jeden parametr; zasada dodawania parametrów z domyślą wartością jest taka sama jak w innych językach programistycznych Przykłady:

```
//przykład poprawnie napisanej funkcji pierwszy argument jest obligatoryjny, drugi już nie
function dzialaj(liczba:int, tekst:String = ”Nieznany”):void {
	//tutaj nasz kod
}

//ta funkcja spowoduje błąd; nie można podac pierwsze parametru z domyślą wartością, a drugiego
//jako obligatoryjny!
function dzialaj(tekst:String = ”Nieznany”, liczba:int):void {
	//tutaj nasz kod
}

//poprawna wersja funkcji powyżej:
function dzialaj(tekst:String = ”Nieznany”, liczba:int = 5):void {
	//tutaj nasz kod
}
```

W AS 3.0 istnieje jeszcze jedna możliwość deklarowania parametrów zmiennych. Mianowicie podajemy jeden parametr, który staje się automatycznie tablicą wszystkich podanych wartości. Najlepiej ilustruje to poniższy przykład:

```
function wieleZmiennych(...zmienne):void {
	trace(”Dodałeś ” + zmienne.length + ”zmienne Ich wartości(wymienione po przecinku): ” + zmienne);
}

wieleZmiennych();
```

```
wieleZmiennych(1,2);
wieleZmiennych(”Hej hop!”):
wieleZmiennych(”Alfabet”,15,14.3,true);
```

3. Obiekty.

Obiekty to, w najprostszym ujęciu tematu, zmienne mogące posiadać szereg nowych, własnych zmiennych. Zmienne danego obiektu mogą z kolei posiadać swoje własne typy; mogą być zagnieżdżone bądź składować inne obiekty. Obiekt możemy utworzyć poprzez generyczną (podstawową, ogólną) klasę Object:

```
var mojObiekt:Object = new Object();
```

Oczywiście możemy także dołączyć do niego swoje własne zmienne:

```
mojObiekt.nowaZmienna = ”Moja nowa zmienna”;
mojIbiekt.numerek = 14.5;
```

Istnieje również możliwość dodawania funkcji do obiektu (tak jak ma to miejsce w klasach, z tym iż obiekt dostaje je niejako „w locie”):

```
mojObiekt.wswietlCos = function():void {
	trace(”Wyświetlam coś!”);
}
```

Bardzo ciekawym i pomocnym aspektem jest możliwość przypisywania już utworzonych funkcji jednego obiektu do nowo tworzonego obiektu:

```
var obj1:Object = new Object();
obj1.wartosc = 45;
obj1.pokazWartosc = function():void {
	trace(this.wartosc);
}

var obj2:Object = new Object();
obj2.wartosc = 100;
obj2.pokazWartosc = obj1.pokazWartosc;

obj1.pokazWartosc(); //wyswietli 45
obj2.pokazWartosc(); //wyswietli 100
```

Dlaczego tak się dzieje? Wpływa na to użycie pseudozmiennej this; Odwołuje się ona bowiem do aktualnie wybranego obiektu, a nie obiektu, w którym np. dana funkcja (metoda) została utworzona.

Powyższy przykład ma jednak pułapkę. Wyobraźmy sobie, że tworzymy 2 obiekty, które posiadają pole name. Teraz oba z nich mają mieć funkcje pokazNazwe, która to miałaby wyświetlać ich nazwy. W tym momencie chcielibyśmy sobie ułatwić zadanie i na samym początku naszego skryptu utworzyć odpowiednią funkcję:

```
function pokazNazwe():void {
	trace(this.name);
}
```

```
var obj1:Object = new Object();
obj1.name = ”nazwa1”;
obj1.pokazNazwe = pokazNazwe;

var obj2:Object = new Object();
obj2.name = ”nazwa2”;
obj2.pokazNazwe = pokazNazwe;

obj1.pokazNazwe();
obj2.pokazNazwe();
```

W powyższym przykładzie spotka nas niestety zawód – każdorazowe wywołanie metody pokazNazwe wyświetli napis „root1”. Dzieje się tak dlatego, że funkcja dodana jest w głównej części kodu AS. Od wersji 3.0 wszystko w skrypcie jest obiektem – także animacje, sceny czy ujęcia. Tym samym otrzymujemy automatycznie nadaną nazwę aktualnie wybranej sceny (root1).

INFORMACJA: Powyższa sytuacja pokazała, że nawet w przypadku gdy sami niczego nie napiszemy w skrypcie (pusty) to samo narzędzie Adobe Flash Professional zrobi to za nas. Dlatego bardzo ważnym jest by np. nie deklarować zmiennych, które są już dostępne jako zmienne sceny/animacji (name, x, y itd.) oraz funkcji (np. addChild).

Prócz klasy Object AS posiada kilka innych wbudowanych typów. Jednym z nich jest Array, czyli tablica pozwalająca na składowanie wielu zmiennych pod kolejnymi jej indeksami. Inne to (nie wszystkie lecz najczęściej używane):
- Date (aktualny czas; porównywanie dat i wiele innych)
- Dictionary (możliwość składowania zmiennych słownikowo; klasa charakterystyczna dla ActionScript 3.0)
- Error (klasa generyczna; istnieje więcej klas zbierających/przrtwarzających błędy skryptów)
- MovieClip (jest bazą dla wszelkich animacji oraz symboli)
- RegExp (wyrażenia regularne; nowość w ActionScript 3.0)
- StyleSheet (klasa składująca oraz pozwalająca na zarządzanie stylami animacji)
- URLVariables (pozwala na nadawanie/przechwytywanie zmiennych adresu strony; odpowiednik $_GET z PHP; nowość w ActionScript 3.0)
- XML/XMLList (służy do tworzenia/edycji plików bądź zasobów XML)

4. Instrukcje warunkowe.

Instrukcje warunkowe wyglądają tak samo jak w innych poznanych językach. Przypomnijmy więc ich składnie:

a) instrukcja warunkowa jeżeli
```
if (warunek) {
	//kod
}
```
b) instrukcja warunkowa jeżeli...w przeciwnym razie
```
if (warunek) {
	//kod
}
else {
	//kod domyślny przy niespełnieniu warunku
}
```

c) instrukcja warunkowa jeżeli...jeżeli nie to... w przeciwnym razie

```
if (warunek) {
	//kod
}
else if(warunek) {
	//kod drugiego warunku
}
else {
	//kod domyślny gdy żaden warunek nie zostanie spełniony
}
```

d) instrukcja przełącz-wybierz:

```
switch(zmienna) {
	case 0: //kod wykonany gdy zmienna == 0
		break;
	case 1: //analogicznei jak wyzej
		break;
	default: //jezeli warunki nie zostaną spełnione to wykona się ten kod
}
```

Budowa warunków oraz ich użytkowanie nie różni się od poznanego w PHP.

5. Pętle programowe

Pętle w ActionScript 3.0 w większości niczym nie różnią się od pętli dostępnych w PHP czy JavaScript. Dla przypomnienia:

a) pętla powtarzania for

```
var i:int; //wazne by zadeklarować zmienną przed użyciem pętli!
for (i = 0; i < 20; i++) {
	trace(i);
}
```

b) pętla powtarzania dopóki (while)

```
var i:int;
while(i < 10) {
	trace(i);
	i++:
} //na wyjsciu 9
```

c) pętla powtarzania rób...dopóki (do...while)

```
var i:int = -1;
do {
	i++;
	trace(i);
} while (i < 5 && i > -1);
//w pętli i jest zwiększone o 1 więc zawsze będzie większe od -1, z kolei wyświetli wartości od 0 do 5 (bo na ostatnim wyświetleniu i będzie 4 + 1)
```

ActionScript 3.0 posiada także dwie, charakterystyczne dla siebie pętle: for...in oraz for each... in

a) for...in

```
var obj:Object = {x: 100, str:”Jestem w obiekcie!”};
for (var i:String in obj) {
	trace(”Zmienna obiektu ” + i + ”zawiera wartość ” + obj[i]);
}
```

Proszę zauważyć, że w tym wypadku jako iteratora użyliśmy zmiennej typu String. Dzięki temu możemy wyciągnąć nazwy zmiennych (właściwości) tworzonego przez nas obiektu i odwoływać się do nich (zmienne te są dostępne zarówno jako np. obj.str jak i obj[”str”]);
b) for each...in

```
var obj:Object = {x: 100, str:”Jestem w obiekcie!”};
for each (var zmienna in obj) {
        trace(zmienna);
}
```

Różnica pomiędzy for...in a for each...in polega na tym, że w pierwszym przypadku możemy bez najmniejszego problemu uzyskać dostęp do nazwy indeksu/zmiennej, której wartość pobieramy. W zaprezentowanym przed chwilą kodzie nie mamy możliwości pobrać wartości/nazwy indeksu,a tylko jego samą wartość.

6. Wybrane obiekty ActionScript 3.0 i ich najczęściej używane składowe

a) Array – pozwala na tworzenie, przechowywanie oraz odwoływanie się do zmiennych tablicowych.
Właściwości (pola):
-length:uint – podaje liczbę elementów w tablicy (0 – brak elementów, każda nieujemna wartość to rzeczywista ilość zadeklarowanych wartości w tablicy)
Użycie:

```
var tablica:Array = new Array(1,2,3,4,5);
trace (tablica.lenght);
```

Metody (funkcje):
- Array(...values) – tworzy tablice z podanymi wartościami
- Array(numElements:int = 0) – tworzy tablice z określoną ilością wartości (wartości puste, do uzupełnienia)
- indexOf(searchElement:*, fromIndex:int = 0):int - wyszukuje w tablicy element wskazany pod parametrem searchElement (opcjonalnie zaczyna od wskazanego indeksu); zwraca numer indeksu, pod którym dana wartość została zapisana bądź -1 w przypadku gdy danej wartości nie ma w tablicy
- join(sep:*):String – łączy ze sobą wszystkie elementy tablicy separując je elementem podanym pod zmienną sep, po czym zwraca efekt połączenia jako ciąg znakowy
- reverse():Array – odwraca położenie elementów tablicy i zwraca to odwrócenie jako nową tablicę
- push(...args):uint – dodaje wartości na koniec tablicy i zwraca jej nową długość
- pop():* - usuwa ostatni element z tablicy i zwraca jego wartość
- map(callback:Function, thisObject:* = null):Array – funkcja tworzy dla każdego elementu aktualnej funkcji tablicę z elementami przekazanymi przez funkcję callback; funkcja może też nadać elementy tylko dla wskazanego elementu przez thisObject.

Więcej o tablicach: http://help.adobe.com/pl_PL/FlashPlatform/reference/actionscript/3/Array.html

b) MovieClip – klasa odpowiada za obsługę klipów filmowych poprzez skrypty ActionScript
Właściwości (pola):
- currentFrame:int – numer ramki aktualnie pokazywanej w klipie
- isPlaying:Boolean – wskazuje, czy dany klip jest aktualnie odtwarzany
- totalFrames:int – całkowita liczba klatek klipu
- currentScene:Scene – zwraca obiekt sceny aktualnie wyświetlanej animacji
- scenes:Array – zwraca tablicę scen, które zostały umieszczone w ramach wskazanego klipu wideo

Metody (funkcje):
- MovieClip() - towrzy nowy, pusty klip filmowy

- stop():void – zatrzymuje odtwarzanie
- nextFrame():void – przesuwa głowicę do następnej klatki i zatrzymuje się na niej
- gotoAndPlay(frame:Object,scene:String = null): void – przesuwa gowicę pod wskazaną klatkę i rozpoczyna jej odtwarzanie
- gotoAndStop(frame:Object,scene:String = null):void – analogicznie do poprzedniej metody, lecz zatrzymuje odtwarzanie
- nextScene():void – przesuwa głowicę do następnej sceny
- prevScene():void – cofa głowicę do poprzedniej sceny

Więcej informacji:
http://help.adobe.com/pl_PL/FlashPlatform/reference/actionscript/3/flash/display/MovieClip.html

c) URLVariables – pozwala na pobranie i wykorzystanie danych przekazanych przy pomocy metody GET

Metody (funkcje):
- decode(source:String):void – dekoduje z podanego ciągu wartości GET i zapisuje je jako zmienne w swoim obiekcie. Przykładowo ciąg:
http://www.example.com?imie=Paweł&wiek=25
przetworzy na:
nazwaObiektu.imie = Paweł;
nazwaObiektu.wiek = 25;

Więcej informacji:
http://help.adobe.com/pl_PL/FlashPlatform/reference/actionscript/3/flash/net/URLVariables.html

d) MouseEvent – obiekt odpowiada za obsługę zdarzeń myszy. Docelowym odbiorcą obsługi zdarzenia jest zawsze najgłębiej zagnieżdżony węzeł widoczny na liście wyświetlania.

Właściwości (pola):
- altKey:Boolean – określa, czy zdarzeniu myszy towarzyszyło wciśnięcie klawisza ALT
- shiftKey:Boolean- jak poprzednio, lecz dla klawisza SHIFT
- controlKey:Boolean/ctrlKey:Boolean – jak poprzednio, lecz dotyczy klawisza CTRL/Command (Mac)
- localX:Number/localY:Number – podaje pozycję kursora myszy X/Y „przetłumaczoną” ze sceny globalnej na pozycję w elemencie, dla którego wystąpiło zdarzenie
- stageX:Number/stageY:Number – współrzędna X/Y kursora myszy względem globalnej sceny (nietłumaczony)
- target:Object – cel zdarzenia (na którym obiekcie dokonano kliknięcia)
- type:String – określa typ zdarzenia (można porównywać ze stałymi)

Przykładowe stałe:
- CLICK:String = „click” – definicja wartości właściwości type zdarzenia myszą (kliknięcie lewym przyciskiem myszy)
- DOUBLE_CLICK:String = „doubleClick” - podwójne kliknięcie myszy
- MOUSE_WHEEL:String = „mouseWheel” - zdarzenie generowane przez suwak myszy
- RIGHT_CLICK:String = „rightClick” - zdarzenie na prawy przycisk myszy

Więcej informacji:
http://help.adobe.com/pl_PL/FlashPlatform/reference/actionscript/3/flash/events/MouseEvent.html

e) KeyboardEvent – obiekt odpowiada za obsługę zdarzeń klawiatury. Zawiera podobne właściwości jak zdarzenie myszy.

Wybrane właściwości:
- keyCode:uint – numer wciśniętego klawisza
- keyLocation:uint – numer klawisza w ujęciu położenia na klawiaturze

Stałe:
- KEY_DOWN:String = „keyDown” - zdarzenie następuje gdy klawisz zostanie naciśnięty
- KEY_UP:String = „keyUp” - zdarzenie następuje gdy klawisz zostanie odciśnięty

f) TextEvent – zdarzenie odpowiada za zdarzenia z klawiatury gdy użytkownik wprowadza tekst bądź pojedynczy znak tekstowy.
Właściwości:
- currentTarget:Object – wskazuje obiekt, przy którym zdarzenie zostało wywołane
- target:Object – miejsce docelowe zdarzenia
- text:String – zawiera znak lub sekwencję znaków wprowadzonych przez użytkownika

Stałe:
LINK:String = „link” - określa zdarzenie uruchamiane np. przy wybraniu odnośnika URL
TEXT_INPUT:String = „textInput” - określa zdarzenie wprowadzania tekstu
Więcej informacji:
http://help.adobe.com/pl_PL/FlashPlatform/reference/actionscript/3/flash/events/TextEvent.html

g) DisplayObjectContainer – klasa bazowa wszystkich klas mogących służyć jako kontenery obiektów ekranowych z listy wyświetlania.

Wybrane właściwości:
- width:Number – określa szerokość obiektu w pikselach
- height:Number – określa wysokość obiektu w pikselach
- name:String – nazwa obiektu
- stage:Stage – stół montażowy, do którego należy obiekt
- x:Number, y:Number, z:Number – wskazuje ułożenie początkowe obiektu względem kolejnych osi X,Y, Z
- visible:Boolean – decyduje o widoczności danego elementu
- tabIndex:int – ustawia kolejność wybierania (przełączania) obiektów w animacji za pomocą klawisza TAB
Wybrane metody:
- addChild(child:DisplayObject):DisplayObject – dodaje nowy kontener-element do elementu nadrzędnego
- contains(child:DisplayObject):Boolean – sprawdza, czy dany obiekt jest dzieckiem aktualnie sprawdzanego (testowanego) obiektu
- getChildIndex(child:DisplayObject):int – zwraca indeks dziecka należącego do obiektu; brak obiektu powoduje wynik ujemny (-1)
- removeChildAt(index:int):DisplayObject – usuwa obiekt wskazany pod danym indeksem i zwraca go jako wynik
- stopAllMovieClips():void – zatrzymuje odtwarzanie wszystkich animacji obiektów
Więcej informacji:
http://help.adobe.com/pl_PL/FlashPlatform/reference/actionscript/3/flash/display/DisplayObjectContainer.html

7. Operatory.

Operatory w ActionScript są takie same jak w JavaScript. Ich spis oraz zastosowanie znajduje się pod adresem: http://help.adobe.com/en_US/FlashPlatform/reference/actionscript/3/operators.html

8. Wszelkie pozostałe informacje na temat ActioScript 3.0 można znaleźć pod adresem http://help.adobe.com/en_US/ActionScript/3.0_ProgrammingAS3/WS5b3ccc516d4fbf351e63e3d118a9b90204-7fce.html