helion.pl
KSIĘGARNIA INTERNETOWA · WYDAWNICTWO

IDŹ DO
PRZYKŁADOWY ROZDZIAŁ
SPIS TREŚCI

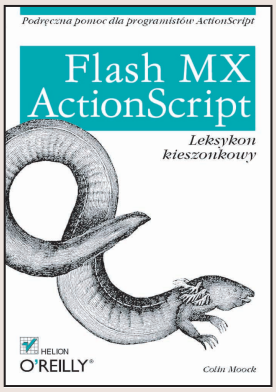
KATALOG KSIĄŻEK
KATALOG ONLINE
ZAMÓW DRUKOWANY KATALOG

TWÓJ KOSZYK
DODAJ DO KOSZYKA

CENNIK I INFORMACJE
ZAMÓW INFORMACJE O NOWOŚCIACH
ZAMÓW CENNIK

CZYTELNIA
FRAGMENTY KSIĄŻEK ONLINE

Flash MX. Action Script. Leksykon kieszonkowy



Autor: Colin Moock
Tłumaczenie: Paweł Janociński
ISBN: 83-7361-190-8
Tytuł oryginału: ActionScript for Flash MX Pocket Reference
Format: B5, stron: 172

ActionScript to język skryptowy pakietu Macromedia Flash służący do tworzenia praktycznie każdego rodzaju treści – od graficznego interfejsu użytkownika i gier do sekwencerów dźwięku i animowanych wygaszaczy ekranu. Składniowo jest niemal identyczny z językiem JavaScript, lecz jest dostosowany do obsługi elementów stworzonych w programie Flash. ActionScript pozwala na tworzenie kodu strukturalnego, obiektowego oraz stanowiącego dowolne połączenie tych dwóch rodzajów.

Ta niewielka książeczka będzie Twoją podręczną pomocą, do której będziesz mógł się odwołać, gdy zapomnisz argumentów potrzebnej Ci właśnie funkcji lub gdy będziesz potrzebował przypomnieć sobie składnie rzadziej używanej konstrukcji ActionScriptu. Znajdziesz w niej krótkie, zwięzłe i zrozumiałe opisy wszystkich elementów składających się na ten język programowania. Jest to niezbędna pozycja dla wszystkich programistów Flasha.

Wydawnictwo Helion
ul. Chopina 6
44-100 Gliwice
tel. (32)230-98-63
e-mail: helion@helion.pl

Spis treści

Wstęp ***5***

Tworzenie kodu ActionScript ***6***

Wyświetlanie informacji kontrolnych 7
Zalecane sposoby organizacji kodu 8
Poszukiwanie kodu 9

Używanie klipów filmowych ***10***

Poziom głębokości klipu filmowego 13
Odwoływanie się do klipów filmowych 14

Składnia języka ActionScript ***15***

Komentarze 15
Białe znaki 16
Ograniczniki wyrażeń (średniki) 16
Rozpoznawanie wielkości znaków 16
Identyfikatory 17
Słowa kluczowe 17

Zmienne ***18***

Zmienne listwy czasowej 18
Zmienne globalne 20
Zmienne lokalne 21
Podpowiedzi do kodu 21

Typy danych ***21***

Zasady konwersji typów danych 23
Jawna konwersja typów danych 23
Określanie typu danych i klasy 27
Typ danych number 28
Typ danych string 28
Typ danych boolean 31
Typy danych null i undefined 32
Typ danych object 32
Typ danych function 32
Typ danych movieclip 33

Tablice *33*
Operatory *35*
Instrukcje warunkowe i pętle *37*
Instrukcje if-else if-else 38
Instrukcja switch 39
Instrukcja while 40
Instrukcja do-while 41
Instrukcja for 41
Instrukcja for-in 42
Instrukcje break i continue 43
Pętle, odświeżanie ekranu i maksymalna liczba iteracji 44
Tworzenie i używanie funkcji *45*
Obsługa zdarzeń *48*
Właściwości obsługujące zdarzenia 48
Odbiorcy zdarzeń 50
Procedury obsługi onClipEvent() i on() 51
ActionScript zorientowany obiektowo *55*
Praca z grafiką *58*
Praca z tekstem *59*
Komponenty i graficzny interfejs użytkownika *62*
Współpraca z zewnętrznymi mediami i danymi *64*
Wczytywanie ilustracji i plików .swf 65
Wczytywanie plików dźwiękowych 65
Wczytywanie stron internetowych 67
Wczytywanie zmiennych 67
Wczytywanie kodu XML 69
Trwałe połączenia gniazd 70
Ograniczenia związane z zabezpieczeniami 70
Współpraca z przeglądarkami internetowymi *72*
Komunikacja z JavaScriptem 74
Lokalizacja źródeł pomocy, przykładów i bibliotek kodu ... *74*
Opis języka ActionScript *76*
Skorowidz *165*

Zmienne lokalne

Zmienne lokalne są tworzone za pomocą instrukcji *var* wewnątrz ciała funkcji. Są one dostępne tylko w tej funkcji i są usuwane gdy funkcja kończy działanie. W przedstawionej poniżej funkcji zastosowano dwie zmienne lokalne, `time` i `hour`, służące do obliczania bieżącej godziny:

```
// zwraca bieżącą godzinę
// w formacie dwunastogodzinnym
function getHours12() {
  var time = new Date();
  var hour = time.getHours();
  if(hour > 12) {
    hour -=12;
  } else if (hour == 0) {
    hour = 12;
  }
  return hour;
}
```

Podpowiedzi do kodu

W tabeli 3. przedstawiono przyrostki nazw zmiennych, których stosowanie powoduje wyświetlanie w panelu *Actions*, po wprowadzeniu nazwy zmiennej, rozwijanej listy zawierającej możliwe kontynuacje wyrażenia (nazywane *podpowiedziami do kodu* — ang. *code hinting*). Przykładowo, aby włączyć wyświetlanie podpowiedzi dla zmiennej oznaczającej pole tekstowe, można jej nadać nazwę `output_txt`.

Typy danych

Język ActionScript posiada wbudowane następujące typy proste: *number*, *string*, *boolean*, *undefined* i *null* oraz następujące typy

Tabela 3. Przyrostki powodujące wyświetlanie podpowiedzi

Przyrostek	Reprezentowany typ danych lub klasa
_mc	MovieClip
_array	Array
_str	String
_btn	Button
_txt	TextField
_fmt	TextFormat
_date	Date
_sound	Sound
_xml	XML
_xmlsocket	XMLSocket
_color	Color
_video	Video
_ch	FCheckBox
_pb	FPushButton
_rb	FRadioButton
_lb	FListBox
_sb	FScrollBar
_cb	FComboBox
_sp	FScrollPane

złożone: *object*, *function* i *movieclip*. ActionScript jest językiem traktującym typy danych zupełnie swobodnie, co oznacza że:

- Zmienne mogą przechowywać dowolny typ danych.
- Dla funkcji nie deklaruje się typów zwracanych ani typów parametrów.
- Pojemniki z danymi (zmienne, elementy tablic i właściwości obiektów) mogą zmieniać swój typ w dowolnym momencie.
- Klasy nie są formalnie odrębnymi typami (wszystkie klasy są traktowane jako dane typu *object*).

Jeśli wartość pewnego typu jest używana w kontekście wymagającym zastosowania innego typu danych, interpreter ActionScript dokona automatycznej konwersji tej wartości na odpowiedni typ danych. Przykładowo, instrukcja *if* wymaga, aby wynik wyrażenia warunkowego był typu *boolean*. Jeśli jednak będzie to wartość innego typu, interpreter automatycznie przekonwertuje ją na ten typ danych przed sprawdzeniem warunku:

```
var options = new Array();
// tablica options przekonwertowana na typ boolean
// przyjmuje wartość true, więc warunek jest spełniony.
if (options) {
  displayOptions();
}
```

Typ wartości może być zmieniony również bezpośrednio, co wyjaśniono dokładniej w podrozdziale „Jawna konwersja typów danych". Zasady dotyczące zarówno jawnego, jak i automatycznego konwertowania typów danych przedstawiono w poniższych tabelach.

Zasady konwersji typów danych

W tabeli 4. przedstawiono wynik konwersji każdego z typów danych na typ *number*.

W tabeli 5. przedstawiono wynik konwersji każdego z typów danych na typ *string*.

W tabeli 6. przedstawiono wynik konwersji każdego z typów danych na typ *boolean*.

Jawna konwersja typów danych

Poniżej opisano sposoby konwersji dowolnych wartości na typy *string*, *number* i *boolean*.

Tabela 4. Konwersja na typ number

Oryginalne dane	Wynik konwersji na typ number
`undefined`	`0`
`null`	`0`
`boolean`	`1`, jeśli oryginalną wartością była `true` (prawda); `0`, jeśli oryginalną wartością była `false` (fałsz)
Łańcuch zawierający liczbę	Odpowiadająca jej wartość liczbowa, jeśli łańcuch składał się wyłącznie z liczb dziesiętnych, białych znaków, wykładnika, kropki dziesiętnej, znaku + lub - (np. `-1.485e2`)
Inny łańcuch	Łańcuchy puste i takie, które nie definiują liczby (z rozpoczynającymi się od `x`, `0x` czy `FF` włącznie) są konwertowane na wartość `NaN` (ang. *not a number*) oznaczającą, że nie jest to wartość liczbowa
`"Infinity"`	`Infinity` (nieskończoność)
`"-Infinity"`	`-Infinity` (minus nieskończoność)
`"NaN"`	`NaN`
Tablica	`NaN`
`object`	Wartość zwracana przez metodę *`valueOf()`* obiektu
`movieclip`	`NaN`

Tabela 5. Konwersja na typ string

Oryginalne dane	Wynik konwersji na typ string
`undefined`	`""` (pusty łańcuch)
`null`	`"null"`
`boolean`	`"true"`, jeśli oryginalną wartością była `true` (prawda); `"false"`, jeśli oryginalną wartością była `false` (fałsz)
`NaN`	`"NaN"`
`0`	`"0"`
`Infinity` (nieskończoność)	`"Infinity"`
`-Infinity` (minus nieskończoność)	`"-Infinity"`

Tabela 5. Konwersja na typ string — ciąg dalszy

Oryginalne dane	Wynik konwersji na typ string
Inna wartość liczbowa	Łańcuch odpowiadający tej wartości, dla `944.345` będzie to `"944.345"`
Tablica	Lista wartości tablicy oddzielonych przecinkami
`object`	Wartość zwracana przez metodę *`toString()`* obiektu; domyślnie metoda ta zwraca `"[object Object]"`, ale może być również dostosowana (przykładowo dla obiektu *`Date`* metoda ta zwraca czytelną datę, taką jak `"Sun May 18 11:38:10 CET 2003"`)
`movieClip`	Ścieżka bezwzględna do instancji klipu filmowego, rozpoczynająca się od poziomu dokumentu w odtwarzaczu Flash Player, np. `"_level0.ball"`

Tabela 6. Konwersja na typ boolean

Oryginalne dane	Wynik konwersji na typ boolean
`undefined`	`false`
`null`	`false`
`NaN`	`false`
`0`	`false`
`Infinity`	`true`
`-Infinity`	`true`
Inna wartość liczbowa	`true`
Niepusty łańcuch znakowy	`true`, jeśli łańcuch może zostać przekonwertowany na prawidłową liczbę różną od zera, w przeciwnym razie `false`; specyfikacja ECMA-262 określa, że niepusty łańcuch zawsze jest konwertowany na wartość `true`, (Flash odstępuje od niej dla zachowania wstecznej zgodności z Flashem 4)
Pusty łańcuch znakowy (`""`)	`false`
Tablica	`true`
`object`	`true`
`movieclip`	`true`

Konwersja na typ string

Wartość może zostać przekonwertowana na typ *string* za pomocą metody *toString()*, funkcji *String()* lub przez konkatenację tej wartości z pustym łańcuchem tekstowym:

```
// utworzenie zmiennej zawierającej wartość typu number
var i = 10;
// konwersja zmiennej i na łańcuch tekstowy za pomocą
var a = i.toString();  // toString()
var b = String(i);     // String()
var c = i + ""         // konkatenacji
```

Aby przekonwertować wartość liczbową na łańcuch reprezentujący ją w zapisie szesnastkowym, należy zastosować metodę *toString()* z argumentem 16, oznaczającym podstawę systemu liczbowego:

```
var x = 255;
trace(x.toString(16));   // wyświetli: ff
```

Konwersja na typ number

Wartość może zostać przekonwertowana na typ *number* za pomocą funkcji *Number()*, *parseInt()*, *parseFloat()* lub przez odjęcie zera:

```
// utworzenie zmiennej zawierającej wartość typu boolean.
var flag = true;
// konwersja zmiennej flag na liczbę za pomocą...
var a = Number(flag);   // Number()
var b = flag - 0;       // odjęcia zera
```

Funkcja *parseInt()* wybiera z łańcucha znaków pierwszą napotkaną liczbę całkowitą, o ile pierwszym znakiem łańcucha (nie licząc znaków białych) jest prawidłowa wartość liczbowa. W przeciwnym razie zwracana jest wartość `NaN`:

```
parseInt("1a")          // zwraca 1
parseInt("1.3a")        // zwraca 1
parseInt("wynik: 2000") // zwraca NaN
```

Aby przekonwertować wartość szesnastkową na liczbę dziesiętną, łańcuch zawierający tę wartość powinien rozpoczynać się od `0x`:

```
// konwersja łańcucha traktowanego jako liczba szesnastkowa
// (niejawne użycie podstawy 16) daje 255
parseInt("0xFF")
```

Można również podać jawnie podstawę `16` jako drugi argument funkcji *parseInt()*:

```
// konwersja łańcucha traktowanego jako liczba szesnastkowa
// (jawne użycie podstawy 16) daje 255
parseInt("FF", 16)
```

Funkcja *parseFloat()* zwraca pierwszą liczbę zmiennoprzecinkową napotkaną w łańcuchu tekstowym, o ile pierwszym znakiem łańcucha (nie licząc znaków białych) jest prawidłowa wartość liczbowa. W przeciwnym razie funkcja zwraca `NaN`:

```
// zwraca 2.5
parseFloat("2.5 to wynik łączny")
// zwraca NaN
parseFloat("Razem: 2.5")
```

Konwersja na typ boolean

Wartość może być przekonwertowana na typ *boolean* za pomocą funkcji *Boolean()*:

```
var obj = new Object();
// zwraca wartość true (zgodnie z tabelą 6)
Boolean(obj)
```

Określanie typu danych i klasy

Aby sprawdzić typ danych dowolnej wartości, należy zastosować operator *typeof*:

```
trace(typeof 11);   // wyświetla: "number"
var name = "Jarek";
trace(typeof name)  // wyświetla: "string"
```

Aby sprawdzić czy obiekt należy do określonej klasy, należy użyć operatora *instanceof*. Poniższy kod przedstawia utworzenie nowej instancji typu *Array* oraz sprawdzenie, czy należy ona do klasy *Array*:

```
var list = ["jeden", "dwa", "trzy"];
// wyświetla: true
trace(list instanceof Array);
```

Typ danych number

ActionScript posiada jeden typ danych, *number*, reprezentujący zarówno całkowite, jak i zmiennoprzecinkowe wartości liczbowe (w formacie zmiennoprzecinkowym o podwójnej precyzji oferuje on dokładność do 15 cyfr znaczących). Wartość liczbowa składa się z sekwencji cyfr, po której następuje opcjonalna kropka dziesiętna oraz opcjonalny wykładnik:

```
3
3.14
15e5  // oznacza 1500000
```

Początkowy znak (`+` lub `-`) jest opcjonalny. Jeśli wartość rozpoczyna się od znaków `0x`, jest traktowana jako szesnastkowa, zaś jeśli rozpoczyna się od `0` — jako ósemkowa:

```
0xCC  // szesnastkowa wartość CC odpowiada liczbie dziesiętnej
      ↳204
```

W tabeli 7. przedstawiono specjalne wartości typu *number*.

Typ danych string

Typ danych *string* reprezentuje dane tekstowe (litery, znaki interpunkcyjne i inne znaki). Wyrażenie znakowe składa się z dowolnej

Tabela 7. Specjalne wartości typu number

Wartość	Opis
`NaN`	Wartość zwracana w przypadku wystąpienia błędu podczas operacji liczbowej, takiej jak dzielenie zera przez zero, czy błędu konwersji
`Number.MIN_VALUE`	Najmniejsza reprezentowana wartość liczbowa (wartości mniejsze są traktowane jako `0`)
`Number.MAX_VALUE`	Największa reprezentowana wartość liczbowa (wartości większe powodują przekroczenie zakresu i są traktowane jako nieskończoność, `Infinity`)
`Infinity`	Dowolna liczba większa niż `Number.MAX_VALUE` (np. wynik przekroczenia zakresu w górę)
`-Infinity`	Dowolna liczba większa niż `-Number.MAX_VALUE` (np. wynik przekroczenia zakresu w dół)

kombinacji znaków ujętej w pojedynczy lub podwójny cudzysłów. Przykładowo:

```
"Witam na mojej stronie internetowej."  // podwójny cudzysłów
'Witam na mojej stronie internetowej.'  // pojedynczy cudzysłów
```

Aby w łańcuchu ujętym w podwójny cudzysłów umieścić znak takiego cudzysłowu, należy poprzedzić go znakiem `\`, co przedstawiono w poniższym przykładzie:

```
"Powiedziała \"cześć\" i uśmiechnęła się."
```

Od wersji szóstej Flash Playera możemy włączać do łańcucha tekstowego znaki Unicode, podając ich czteroznakowy szesnastkowy kod liczbowy poprzedzony sekwencją `\u`. Przykładowo:

```
\u00A9  // znak "copyright"
\u2014  // znak pauzy
```

W wersji piątej Flash Playera wszystkie sekwencje Unicode (rozpoczynające się od `\u`) muszą określać znaki ze zbiorów Latin 1 lub Shift-JIS — pozostałe nie będą wyświetlone prawidłowo.

Dla łatwego wprowadzania najczęściej spotykanych znaków specjalnych, język ActionScript wyposażono w odpowiedni zestaw skrótów, wymienionych w tabeli 8.

Tabela 8. Sekwencje specjalne

Sekwencja specjalna	Znaczenie
`\b`	Znak *backspace* (o kodzie ASCII 8)
`\f`	Znak *form feed* (o kodzie ASCII 12)
`\n`	Znak nowej linii (o kodzie ASCII 10)
`\r`	Znak *powrotu karetki* — CR (o kodzie ASCII 13), w systemie Windows razem ze znakiem nowej linii oznacza koniec wiersza
`\t`	Znak tabulacji (o kodzie ASCII 9)
`\'`	Pojedynczy cudzysłów
`\"`	Podwójny cudzysłów
`\\`	Znak lewego ukośnika (\); sekwencja jest konieczna do wstawienia tego znaku, gdyż w innym wypadku zostanie on zinterpretowany jako początek sekwencji specjalnej
`\x`*dd*	Znak strony kodowej Latin 1 o podanym numerze (*dd* oznacza dwie cyfry szesnastkowe)
`\u`*dddd*	Znak Unicode o podanym numerze (*dddd* oznacza cztery cyfry szesnastkowe)

Znaki Unicode mogą być tworzone także za pomocą metody *String.fromCharCode()*, do której jako argument należy podać listę oddzielonych przecinkami kodów dziesiętnych lub szesnastkowych żądanych znaków. Przykładowo, poniższe wyrażenie zwraca znak *copyright*:

```
String.fromCharCode(169)
```

Pomimo że Flash Player 6 w pełni obsługuje Unicode, pakiet Flash MX tego nie potrafi. Narzędzie to pozwala na stosowanie wyłącznie znaków ze zbiorów Latin 1, Shift-JIS i MacRoman. Aby użyć znaków innych języków, należy wykonać jedną z poniższych czynności:

- Podczas działania programu użyć metod *XML.load()* lub *LoadVars.load()* do wczytania zewnętrznych plików XML lub plików ze zmiennymi zapisanych w formacie Unicode
- Podczas tworzenia kodu zapisać tekst w zewnętrznym pliku tekstowym z rozszerzeniem *.as* w formacie Unicode, a następnie zaimportować go za pomocą dyrektywy `#include`
- Podczas tworzenia kodu stworzyć każdy znak osobno poprzez zastosowanie albo szesnastkowych sekwencji specjalnych albo metody *String.fromCharCode()*

Przykładowo, poniższy kod ilustruje utworzenie globalnej zmiennej o nazwie `euro`, która zawiera znak *euro*. Zmienna ta jest następnie użyta w polu tekstowym `price_txt` do wyświetlenia na ekranie.

```
_globals.euro = "\u20AC";
this.createTextField("price_txt", 1, 100, 100, 200, 20);
price_txt.text = "99 " + euro;
```

Gdy prosta wartość znakowa jest stosowana w kontekście wymagającym użycia obiektu *String*, interpreter ActionScript automatycznie *opakuje* ją obiektem *String*. Pozwala to na wywoływanie metod *charAt()* czy *indexOf()* dla łańcucha oraz na sprawdzenie liczby znaków za pomocą właściwości `length`.

```
var name = "Karol";
trace(name.indexOf("b"));  // wyświetli: -1
trace(name.length);        // wyświetli: 5
```

Pełną listę metod i właściwości, których można używać z łańcuchami znaków można znaleźć w opisie klasy *String*.

Typ danych boolean

Typ danych *boolean* służy do reprezentacji logicznych wartości określających prawdziwość lub fałszywość stwierdzenia. Zatem istnieją tylko dwie wartości tego typu: `true` (prawda) i `false` (fałsz).

Wartości tego typu są najczęściej stosowane w pętlach i instrukcjach warunkowych.

Typy danych null i undefined

ActionScript zawiera dwa typy reprezentujące brak danych: *null* i *undefined*, których jedynymi wartościami są odpowiednio `null` i `undefined`.

Wartość `undefined` jest zwracana przez interpreter automatycznie, jeśli zmienna, właściwość czy element tablicy, do którego nastąpiło odwołanie nie ma jeszcze przypisanej wartości.

W przeciwieństwie do tego, wartość `null` jest używana przez programistów do zaznaczenia, że zmienna, właściwość czy element tablicy celowo nie ma przypisanej wartości.

Warto zwrócić uwagę na fakt, że `null` i `undefined` są przez operator porównania (`==`) traktowane jako równe. Aby jednoznacznie je rozróżnić, należy zastosować operator ścisłej równości, który jest oznaczany trzema znakami równości (`===`). Operator ten przed porównaniem wartości sprawdza czy oba wyrażenia są tego samego typu.

Typ danych object

W języku ActionScript typ *object* służy do przechowywania zbiorów logicznie powiązanych ze sobą danych (nazywanych *właściwościami*) oraz funkcji (nazywanych *metodami*). Więcej informacji na ten temat można znaleźć w rozdziale „ActionScript zorientowany obiektowo" niniejszej książki.

Typ danych function

Typ danych *function* reprezentuje funkcje języka ActionScript. Są to fragmenty kodu, których można używać wielokrotnie

(w niektórych językach są nazywane *podprocedurami*). Więcej informacji na ten temat można znaleźć w rozdziale „Tworzenie i używanie funkcji".

Typ danych movieclip

Typ danych *movieclip* reprezentuje klipy filmowe, podstawowe pojemniki Flasha. Klipy filmowe są stosowane tak samo jak obiekty, ale należą do osobnego typu danych ze względu na sposób, w jaki interpreter przydziela i zwalnia ich zasoby (patrz *http://moock.org/asdg/technotes/movieclipDatatype/*). Więcej informacji na ten temat można znaleźć w rozdziale „Używanie klipów filmowych" niniejszej książki.