

# ActionScript 2.0

## Podstawy

---

# ActionScript

- Podobieństwo do JavaScript
  - Fragmenty kodu, które można wykorzystywać wielokrotnie
  - Istnieją edytory AS
  - Łatwy do nauki przez nie-programistów
  - Flash bazuje na składni ActionScript 2.0
  - Case sensitivity
-

# Zmienne

- Kontenery na dane odpowiedniego typu
    - `var liczba:Number`
    - Kontener zapamiętuje pewną wartość
    - Może ona być zmieniana dynamicznie
    - Każdy kontener ma symbol
    - Można zobaczyć zawartość kontenera wykorzystując debugger i funkcję **trace**
      - `trace (nazwa_zmiennej);`
-

# Klasy i obiekty

- Klasy to wzorce zachowania obiektów
  - Obiekt to pewna instancja klasy
    - Pies jest przedstawicielem Zwierząt. Pies jest obiektem, Zwierzęta są klasą
  - Wszystkie obiekty są instancjami pewnej klasy
  - Obiekty to również typy danych takich jak muzyka, grafika, tekst, wartości liczbowe
-

# Właściwości (properties)

- Każda klasa ma predefiniowany zbiór właściwości.
  - Każdy obiekt klasy ma swój własny zbiór wartości tych właściwości
    - Obiekt Movie Clip ma właściwości: `_height`, `_width`, `_rotation`
  - Można zdefiniować i zmieniać właściwości każdego obiektu klasy.
-

# Metody

- Czynność, którą obiekt może wykonać
    - Klasa Sound ma metodę setVolume
  - Kiedy obiekt wywołuje jakąś metodę mówimy że metoda jest wywoływana lub że inny obiekt woła metodę
  - Możemy definiować własne metody
-

# Konstrukcja tworzenia obiektów

- Aby korzystać z obiektu, należy go utworzyć (utworzyć instancję)

```
myColor = new Color();
```

```
myDate = new Date ();
```

```
mySound = new Sound();
```

---

# Wołanie metody obiektu

- Następnym krokiem po utworzeniu obiektu jest przypisanie nowych wartości do właściwości.
- Można tego dokonać bezpośrednio lub przez wykonanie metody innego obiektu

```
myDisplay = new Display();
```

```
myDate = new Date ();
```

```
// call the method of the constructed object
```

```
myDisplay.time = myDate.getdate();
```

```
myDisplay.name = "Ekran 1";
```

---

# Dot Syntax

- Do określenia wartości i wywołania metod używa się tzw. dot syntax;
    - `movieclip1._rotation = 45`
    - OBIEKT WŁAŚCIWOŚĆ WARTOŚĆ
  - Można używać zapisu z wieloma kropkami zachowując hierarchię
-

# Dot Syntax

- Methods są wywoływane w ten sam sposób
    - `mouse. gotoAndPlay ("scene1", 20)`
  - Nawiasy po nazwie `gotoAndPlay` oznaczają że jest to metoda a nie właściwość!
  - W nawiasach przekazywane są argumenty lub parametry
  - Do metody możemy nie przekazać argumentów ale nawiasy i tak muszą wystąpić
-

# Dalsze uwagi

- Średnik kończy każde zdanie

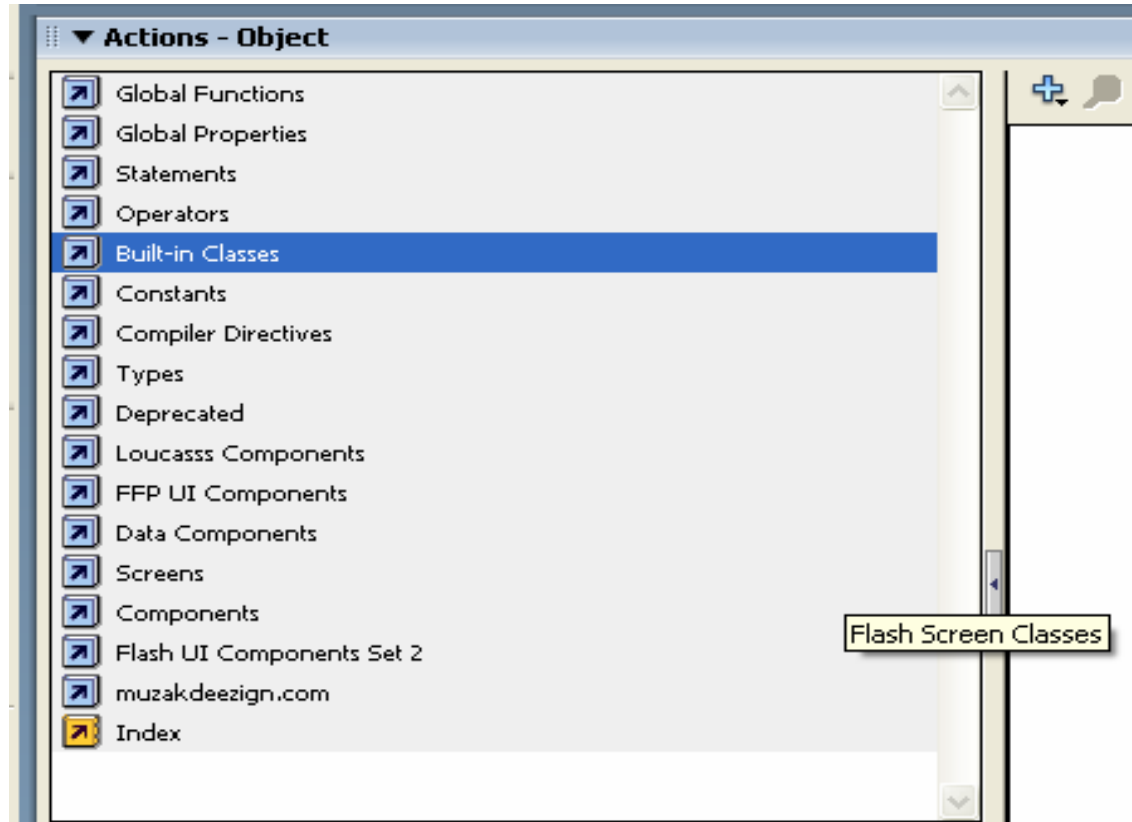
- `stopAllSounds();`  
`play();`

- Nawiasy `{ }` grupują zdania w pewien blok

- `on (release) {`  
`stopAllSounds();`  
`play();`  
`}`

---

# Edytory ActionScript - SEPY



# ActionScript 2.0

- Podobny do każdego innego języka programowania
  - Kontrola typów
    - Opcjonalna
    - Dobra aby unikać błędów w czasie wykonania
    - W edytorach są podpowiadane ich nazwy
      - `var myVar:Number`
        - Wykorzystanie `var` nie jest wymagane ale mówi Flash'owi że deklaruje się zmienną określonego typu
-

# Funkcje

---

# Tworzenie funkcji

- Potrzebujemy nazwy funkcji i bloku zdań do jej wykonania

```
function functionName () {  
    Statement one;  
    Statement two;  
    Statement three;  
}
```

- Tworzenie funkcji nie powoduje jej wykonania !!
-

# Przekazywanie informacji do funkcji

## ■ Składnia

- ❑ `function moveBall():Void{`
  - `ball._x +=10;`
  - `ball._y +=10;`
- ❑ `}`

# Funkcja z parametrami

- Parametry są zmiennymi które mogą być użyte w funkcji
  - Dostarczamy listę identyfikatorów pomiędzy nawiasami w deklaracji funkcji

```
function moveClip (theClip, xDist, yDist) {  
    theClip._x += xDist;  
    theClip._y +=yDist;  
}
```
  - Należy unikać definiowania stałych wartości –  
mniejsza elastyczność
-

# Wywołanie funkcji z parametrami

- Kolejne parametry (stałe lub zmienne) rozdzielamy przecinkami
- Tak czy inaczej każdy paramter ma wartość (stałą lub zmienną)

```
on (release){  
    moveClip (clip1, 100, 200)  
}
```

# Zakończenie funkcji

- Używamy instrukcji ***return***
  - ActionScript wykonuje ją domyślnie, nawet jeśli nie użyjemy jej w sposób jawny
  - Funkcja może zwracać wartość.
  - Należy poinformować o tym w deklaracji funkcji
    - *return wyrażenie*
-

# Czas życia funkcji (widoczność)

- Funkcja zdefiniowana w ramce jest widoczna tzn. można ją wywołać tylko z obiektów tej ramki. Przejście do innej ramki powoduje że kod funkcji ginie.
  - Można zdefiniować kod w Master Frame. Wówczas jest do niej dostęp z każdego miejsca
-

# Typy w AS 2.0

- Typy (= klasy) dla obiektów niewidocznych
    - Array
    - Boolean
    - Number
    - Object
    - String
    - + klasy użytkownika zdefiniowane przez `class { ... }`
  - Typy (= klasy) dla obiektów wizualnych
    - MovieClip
    - Button
    - TextField
    - Component
  - Dla obiektów wizualnych, zalecane jest zawarcie w nazwie nazwy typu!
-

# Pełna lista typów

- Accordion
  - Alert
  - Array\*
  - Binding
  - Boolean\*
  - Button\*\*
  - Camera\*\*
  - CheckBox
  - Color\*
  - ComboBox
  - ComponentMixing
  - CustomActions
  - DataField
  - DataGrid
  - DataHolder
  - DataSet
  - DataType
  - Date\*
  - DateChooser
  - Delta
  - DeltaItem
  - DeltaPacket
  - Endpoint
  - Error
  - Function\*\*
  - Label
  - LoadVars\*\*
  - LocalConnection\*\*
  - Log
  - MediaController
  - MediaDisplay
  - MediaPlayback
  - Menu
  - MenuBar
  - Microphone\*\*
  - MovieClip\*
  - MovieClipLoader
  - NetConnection\*\*
  - NetStream\*\*
  - Number\*
  - Object\*
  - PendingCall
  - PopUpManager
  - PrintJob
  - ProgressBar
  - RadioButton
  - RadioButtonGroup
  - RDBMSResolver
  - ScrollPane
  - SharedObject\*\*
  - Slide
  - SOAPCall
  - Sound\*
  - String\*
  - TextArea
  - TextField\*\*
  - TextFormat\*\*
  - TextInput
  - TextSnapshot
  - Tree
  - TypedValue
  - Video\*\*
  - Void
  - WebService
  - Connector
  - Window
  - XML\*
  - XMLConnector
  - XMLNode\*
  - XMLSocket\*
  - XUpdateReceiver
-

# Programowanie skryptowe we Flash

- Skrypty lokalne, wariant statyczny
    - Kod ActionScript jest dołączony do pewnej ramki i pewnej linii czasowej
    - Kod jest wykonany jeśli ta ramka jest wyświetlana
  - Skrypty lokalne, wariant statyczny
    - Kod Action script code jest doczepiony jako obsługa zdarzenia (event handler)
    - Kod jest wykonywany gdy wystąpi to zdarzenie
    - Kod rozpoczyna się od klauzuli “on” lub “onClipEvent”
  - Skrypty globalne
    - Jako pliki z rozszerzeniem .as
    - Kod nie jest wykonywany ale mamy do niego dostęp z dowolnego miejsca
-

# Ulokowanie metod globalnych

- Tworzymy plik .as
  - Tworzymy kod klasy
  - Tworzymy master frame i doczepiamy kod
  - Utrzymujemy zmienne obiekty w głównej linii czasowej (`_root`).
  - Dodajemy kod globalny (“global”) do pierwszej wykonywanej ramki
-

# Przykład: klasa `Array`

---

Method	Description
<code>Array.concat()</code>	Concatenates the parameters and returns them as a new array.
<code>Array.join()</code>	Joins all elements of an array into a string.
<code>Array.pop()</code>	Removes the last element of an array and returns its value.
<code>Array.push()</code>	Adds one or more elements to the end of an array and returns the array's new length.
<code>Array.reverse()</code>	Reverses the direction of an array.
<code>Array.shift()</code>	Removes the first element from an array and returns its value.
<code>Array.slice()</code>	Extracts a section of an array and returns it as a new array.
<code>Array.sort()</code>	Sorts an array in place.
<code>Array.sortOn()</code>	Sorts an array according to a field in the array.
<code>Array.splice()</code>	Adds and removes elements from an array.
<code>Array.toString()</code>	Returns a string value representing the elements in the <code>Array</code> object.
<code>Array.unshift()</code>	Adds one or more elements to the beginning of an array and returns the array's new length.

---

# Klasa Array

```
var passwords_array:Array = new Array("mom:glam", "ana:ring",  
    "jay:mag", "anne:home", "regina:silly");  
function order(a, b):Number  
{  
    var name1:String = a.split(":")[0];  
    var name2:String = b.split(":")[0];  
    if (name1<name2) { return -1; } else  
    if (name1>name2) { return 1; }  
    else { return 0; }  
}  
//nieposortowanemom:glam,ana:ring,jay:mag,anne:home,regina:silly  
passwords_array.sort(order);  
//posortowane ana:ring,anne:home,jay:mag,mom:glam,regina:silly
```

---