

# PROGRAMOWANIE W ŚRODOWISKU FLASH

## wykład 3

Paweł Wozzkowski  
SWSIM 2010

[www.wozzkowski.com](http://www.wozzkowski.com)



# DICTIONARY

Obiekt, którego nazwy zmiennych lub klucze mogą być zmiennymi

```
var arr:Array = new Array();  
arr[0] = "wartosc";
```

```
var obj:Object = new Object();  
obj["variable"] = "wartosc";
```

```
var dict:Dictionary = new Dictionary();  
dict[arr] = "wartosc1";  
dict[obj] = "wartosc2";
```



# DICTIONARY

## PRZYKŁAD UŻYCIA

```
var pA:Point = new Point();  
var pB:Point = new Point();  
var pC:Point = new Point();
```

```
var nazwy:Dictionary = new Dictionary();  
nazwy[pA] = "A";  
nazwy[pB] = "B";  
nazwy[pC] = "C";
```

```
pA.x;      // współrzędna x  
pA.y;      // współrzędna y  
nazwy[pA]; // "A"
```



# DISPLAY OBJECTS

Display list - hierarchiczna lista wyświetlanych obiektów

Aby obiekt był widoczny na scenie, należy dodać go do listy wyświetlania.

```
kontener.addChild(obiekt);
```

```
kontener.removeChild(obiekt);
```

Najważniejsze obiekty:

- MovieClip
- Sprite
- Shape
- Bitmap
- Loader



# DISPLAY OBJECTS WŁAŚCIWOŚCI

Każdy obiekt wyświetlany na ekranie ma podstawowe, wspólne właściwości:

- `x, y`
- `width, height`
- `scaleX, scaleY`
- `rotation`
- `alpha`
- `visible`
- `mouseX, mouseY`
- `parent`



# MOVIECLIP WŁAŚCIWOŚCI

Właściwości obiektu MovieClip wg ActionScript 3.0 Language and Components Reference:

- `currentFrame`
- `currentLabel`
- `currentLabels`
- `currentScene`
- `enabled`
- `framesLoaded`
- `scenes`
- `totalFrames`
- `trackAsMenu`

A gdzie x,y, itd.???



# KLASA

Klasa - definiuje obiekt, określa jego atrybuty (właściwości) oraz zachowania (metody).

Struktura klasy:

package

- class
  - variable
  - constant
  - function
    - variable
    - constant



# KLASA - SKŁADNIA

```
package com.woszkowski.swsim
{
    public class Wyklad3
    {
        public var liczbaStudentow:uint = 30;

        public function Wyklad3():void
        {
            trace(liczbaStudentow);
        }
    }
}
```



# DZIEDZICZENIE

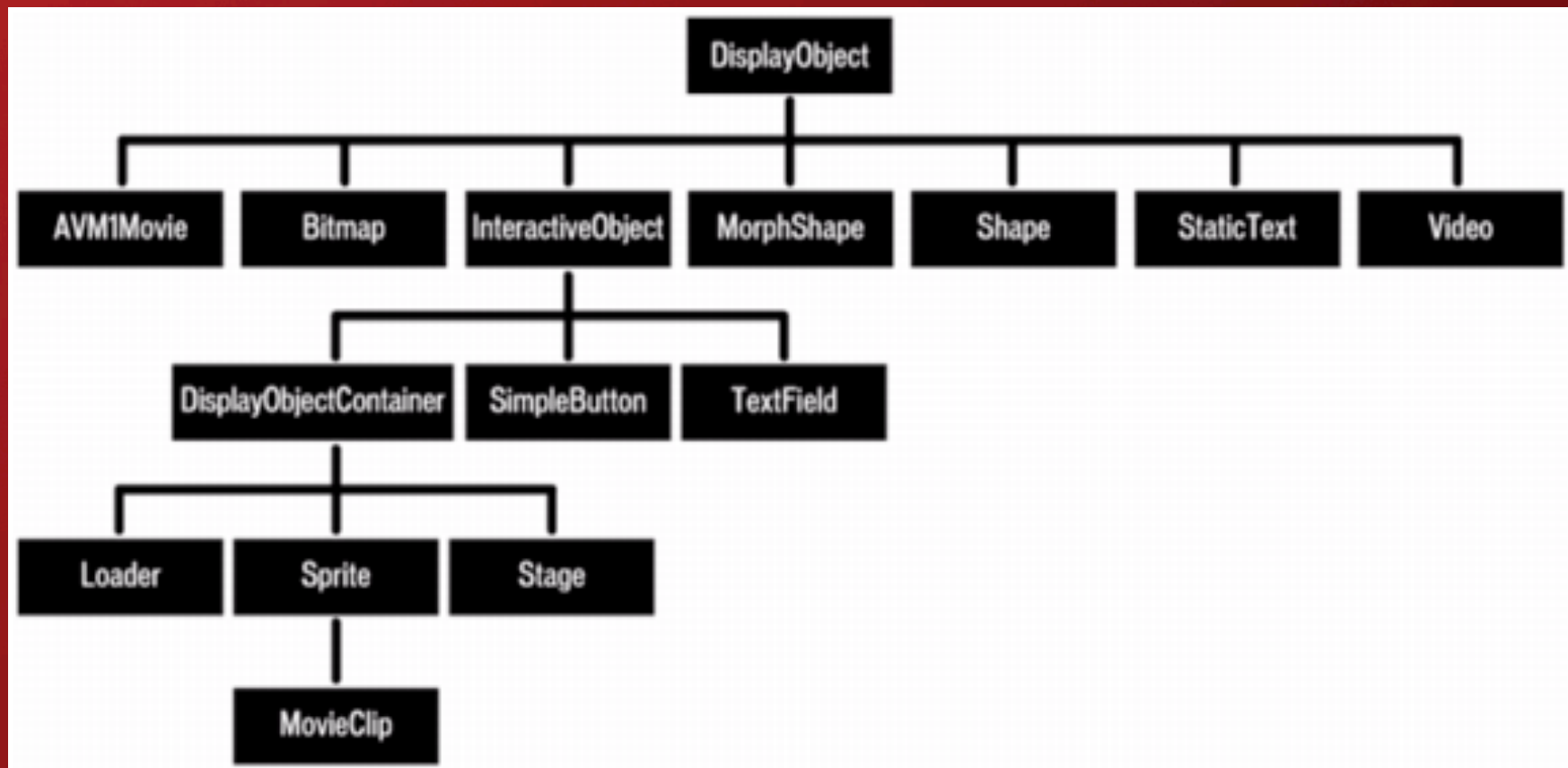
```
public class Pojazd
{
    public var kolor:String;
    public function ruszaj()
    {
        trace("ruszam");
    }
}
```

```
public class Samochod extends Pojazd
{
    public var typPaliwa:String;
    public function zatankuj()
    {
        trace("tankuję");
    }
}
```

```
var auto:Samochod = new Samochod();
auto.typPaliwa = "diesel";
auto.kolor = "czerwony";
auto.zatankuj();           // tankuję
auto.ruszaj();            // ruszam
```



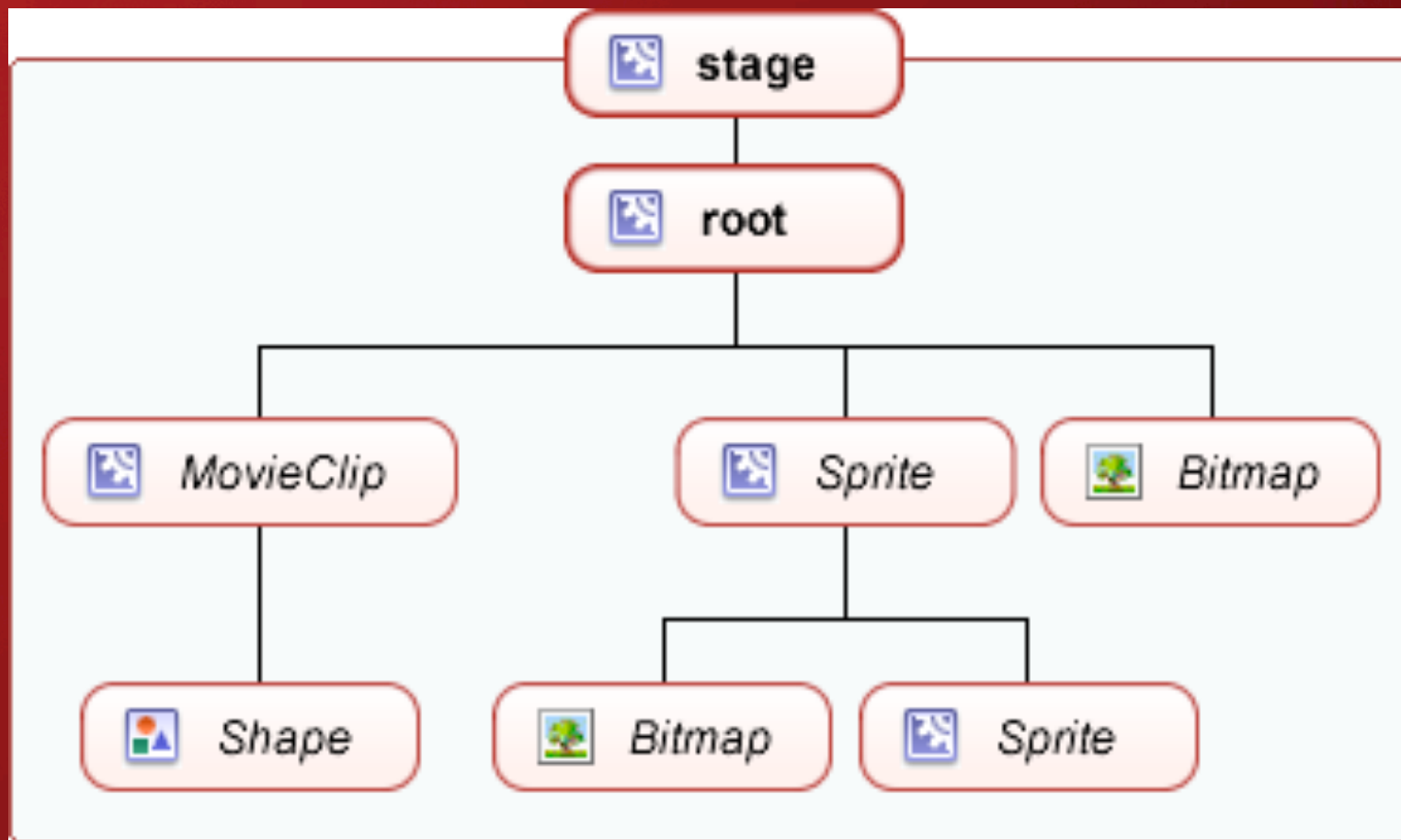
# MOVIECLIP WŁAŚCIWOŚCI CD.



MovieClip > Sprite > DisplayObjectContainer > InteractiveObject > DisplayObject > EventDispatcher > Object



# PRZYKŁADOWA LISTA WYŚWIETLANIA



# DISPLAY INDEX

```
addChild(kolo);  
addChild(kwadrat);  
addChild(gwiazda);
```

```
removeChild(kwadrat);
```

```
addChildAt(kwadrat, 0);
```



# LISTA OBIEKTÓW

## NOWY OBIEKT

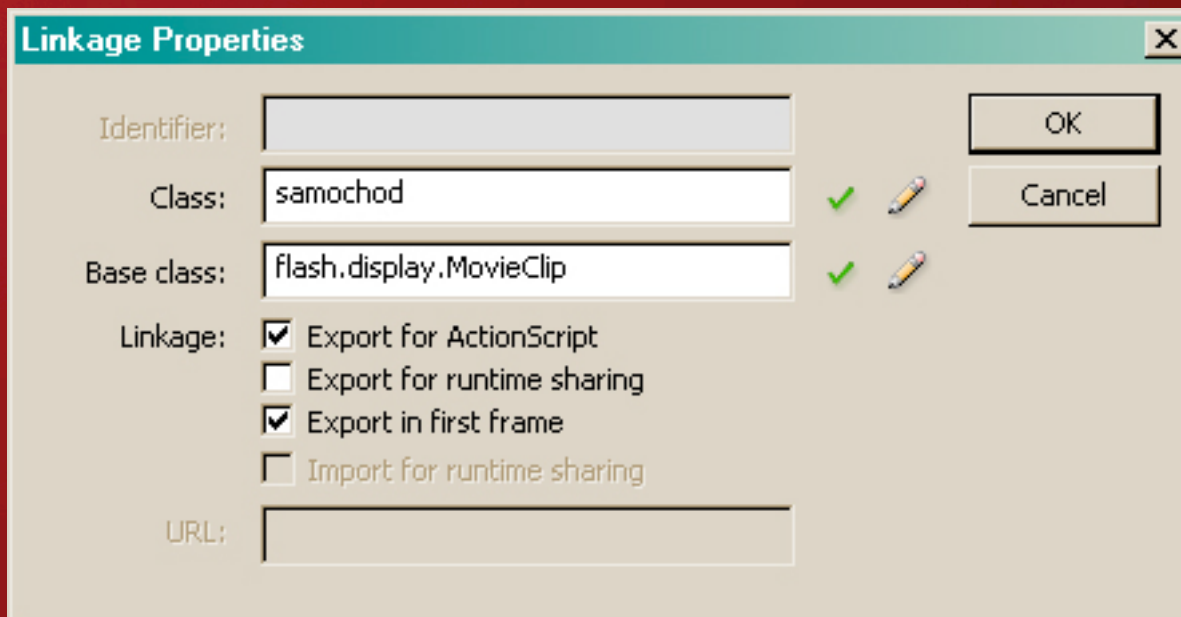
```
var nowySprite:Sprite = new Sprite();  
addChild(nowySprite);
```

```
var nowyMC:MovieClip = new MovieClip();  
addChild(nowyMC);
```



# LISTA OBIEKTÓW BIBLIOTEKA

Dodawanie obiektów z biblioteki:  
obiekt w bibliotece -> Linkage



```
var s:MovieClip = new samochod();  
addChild(s);
```



# BITMAP

```
Bitmap(bitmapData:BitmapData = null, pixelSnapping:String = "auto",  
smoothing:Boolean = false)
```

Obrazek w bibliotece

```
var obraz:BitmapData = new ObrazBiblioteka(0, 0);  
var bitmapa:Bitmap = new Bitmap(obraz);  
addChild(bitmapa);
```



# DISPLAY OBJECTS

## INNE METODY

**numChildren** - liczba obiektów w obiekcie

**getChildAt** - referencja do obiektu wg indexu

**getChildByName** - referencja do obiektu wg nazwy

**getChildIndex** - odczyt indexu obiektu

**setChildIndex** - ustawianie indexu obiektu

**swapChildren** - zamiana indexu obiektów wg referencji

**swapChildrenAt** - zamiana indexu obiektów wg indexu



# GRAPHICS

Wektorowe kształty możemy rysować z poziomu ActionScript w obiektach typu Graphics.

Instancje klasy Graphics znajdują się jako właściwość "graphics" w następujących obiektach:

- Shape
- Sprite
- MovieClip



# GRAPHICS

## DOSTĘPNE METODY

lineStyle

lineGradientStyle

beginFill

beginGradientFill

beginBitmapFill

endFill

clear

moveTo

lineTo

curveTo

drawCircle

drawEllipse

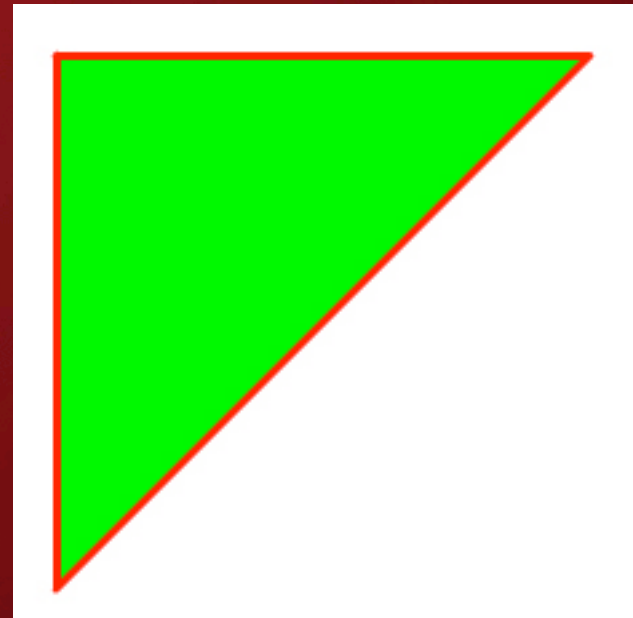
drawRect

drawRoundRect



# GRAPHICS - PRZYKŁADY

```
var s:Sprite = new Sprite();  
addChild(s);  
  
s.graphics.lineStyle(3, 0xFF0000);  
s.graphics.beginFill(0x00FF00);  
s.graphics.moveTo(100, 100);  
s.graphics.lineTo(300, 100);  
s.graphics.lineTo(100, 300);  
s.graphics.lineTo(100, 100);  
s.graphics.endFill();
```



# ZDARZENIA (EVENTS)

Zdarzenia wywoływane są, gdy "coś" się stanie.

Wirtualnie cały ActionScript działa na zasadzie zdarzeń.

Niemal wszystko co się dzieje, generuje zdarzenia:

- kiedy klikniemy w przycisk
- kiedy zostanie załadowany plik
- kiedy zmienimy rozmiar flasha
- kiedy wyjedziemy myszką poza obszar flasha
- nawet kod umieszczony na timeline zostaje uruchamiany w wyniku zdarzenia oznajmiającego wyświetlenie zawartości tej klatki
- ...



# ZDARZENIA (EVENTS)

Zdarzenia mogą generować tylko klasy dziedziczące z `EventDispatcher`.

"Słuchanie" zdarzeń:

```
obiekt.addEventListener(nazwa_zdarzenia, metoda);
```

Usuwanie reakcji na zdarzenie:

```
obiekt.removeEventListener(nazwa_zdarzenia, metoda);
```

Generowanie zdarzeń:

```
dispatchEvent(new Event(nazwa_zdarzenia));
```



# ZDARZENIA (EVENTS)

Główną klasą odpowiedzialną za obsługę zdarzeń jest klasa **Event**. Różne rodzaje zdarzeń mają inne klasy, wszystkie dziedziczą jednak z klasy **Event**.

Dlaczego nazwy zdarzeń definiowane są stałe w odpowiadającym ich klasach?

```
przycisk.addEventListener("klik", metoda); // ok  
przycisk.addEventListener(MouseEvent.CLICK,  
handler); // Error!
```



# PRIORYTETY

Do każdego zdarzenia może być przypisanych wiele metod:

```
obiekt.addEventListener(MouseEvent.CLICK, metoda1);  
obiekt.addEventListener(MouseEvent.CLICK, metoda2);  
obiekt.addEventListener(MouseEvent.CLICK, metoda3);
```

Nasłuchującym metodom można nadawać priorytety:

```
obiekt.addEventListener(MouseEvent.CLICK, metoda1, false, 0);  
obiekt.addEventListener(MouseEvent.CLICK, metoda2, false, 1);  
obiekt.addEventListener(MouseEvent.CLICK, metoda3, false, 2);
```



# ZDARZENIA - PRZYKŁADY

```
var urlLoader:URLLoader = new URLLoader();
urlLoader.addEventListener(Event.COMPLETE, completeListener);
urlLoader.load(new URLRequest("plik.txt"));
function completeListener(e:Event):void {
    trace("Plik wczytany");
}
```

```
przycisk.addEventListener(MouseEvent.CLICK, clickListener);
private function clickListener(e:MouseEvent):void {
    trace("KLIK");
}
```



# ZDARZENIA - PRZYKŁADY

```
public class Game extends EventDispatcher {  
    public static const GAME_OVER:String = "gameOver";  
    private function endGame():void {  
        dispatchEvent(new Event(Game.GAME_OVER));  
    }  
}
```

```
var gra:Game = new Game();  
gra.addEventListener(Game.GAME_OVER, koniecGry);  
private function koniecGry(e:Event):void {  
    ...  
}
```



# ZDARZENIA PRZEKAZYWANY OBIEKT

```
function completeListener(e:Event):void {  
  
}
```

`e.type` - typ zdarzenia

`e.target` - obiekt, który wywołał zdarzenie

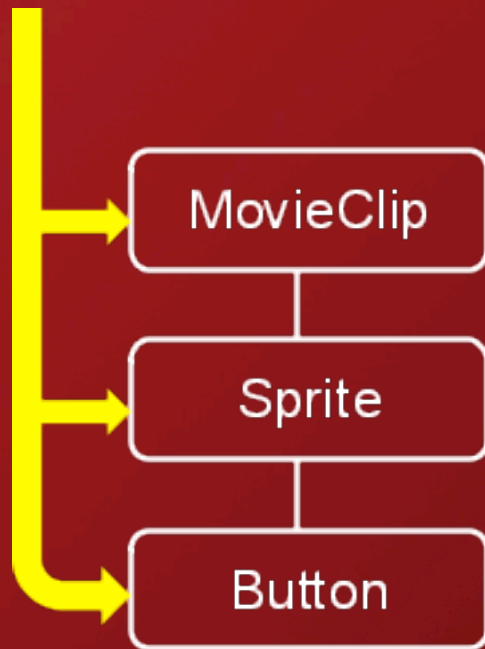
`e.currentTarget` - obiekt w którym wywołano `addEventListener`

```
var urlLoader:URLLoader = new URLLoader();  
urlLoader.addEventListener(Event.COMPLETE, completeListener);  
urlLoader.load(new URLRequest("plik.xml"));  
function completeListener(e:Event):void {  
    var xml:XML = new XML(e.target.data);  
}
```



# PROPAGACJA ZDARZEŃ

Zdarzenia MouseEvent przenikają wгłęb listy wyświetlania i są wywoływane na kolejnych obiektach.



Propagację można zatrzymać, ustawiając `obiekt.mouseChildren = false`



# ZDARZENIA GENEROWANE PRZEZ STAGE

Funkcja obiektu `stage` nie ogranicza się wyłącznie do bycia kontenerem wyświetlanych obiektów. `stage` generuje i przyjmuje wiele zdarzeń. Przede wszystkim dotyczy to zdarzeń związanych z myszką i klawiaturą.

```
obiekt.addEventListener(MouseEvent.CLICK,
zacznijPrzesuwanie);
```

```
stage.addEventListener(MouseEvent.CLICK,
zakonczPrzesuwanie);
```



# OSADZANIE PLIKÓW SWF NA STRONIE WWW

- osadzanie tradycyjne

```
<object classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000"  
codebase="http://download.macromedia.com/pub/shockwave/cabs/  
flash/swflash.cab#version=6,0,29,0" width="600" height="400">  
  <param name="movie" value="plik.swf">  
  <param name="quality" value="high">  
  <embed src="plik.swf" quality="high" pluginspage="http://  
www.macromedia.com/go/getflashplayer" type="application/x-  
shockwave-flash" width="100%" height="100%"></embed>  
</object>
```



# OSADZANIE PLIKÓW SWF NA STRONIE WWW

- osadzanie zgodne z xhtml

```
<object type="application/x-shockwave-flash" data="plik.swf"  
width="600" height="400">  
<param name="movie" value="plik.swf" />  
<param name="menu" value="false" />  
</object>
```



# OSADZANIE PLIKÓW SWF NA STRONIE WWW

- osadzanie za pomocą SWFObject (zalecane!!!)

```
<script type="text/javascript" src="swfobject.js"></script>
```

```
<script type="text/javascript">  
var flashvars = {  
};  
var params = {  
  menu: "false"  
};  
var attributes = {  
};  
swfobject.embedSWF("plik.swf", "flashA", "600", "400",  
"9.0.0", "expressInstall.swf",  
  flashvars, params, attributes);  
</script>
```



# ATRYBUTY OBIEKTÓW OBJECT I EMBED

- width, height
- classid
- codebase (param), pluginspage (embed)
- movie (param), src (embed)
- id, name
- swliveconnect (dla fscommand)
- play, loop
- menu
- quality
- scale
- align, valign
- wmode
- bgcolor
- base
- flashvars



# WMODE

- window (zalecane, jeżeli tylko jest taka możliwość)
  - zawsze na wierzchu
- opaque
  - pozwala na wyświetlanie innych obiektów nad flashem
  - problemy z polskimi znakami w polu input i inne niespodzianki
- transparent
  - opaque + przezroczystość
  - problemy z polskimi znakami w polu input i inne niespodzianki



# PRZEKAZYWANIE DANYCH DO FLASHA

- GET

`plik.swf?param1=val1&param2=val2&...`

- FLASHVARS

```
var flashvars = {  
    param1:"val1",  
    param2:"val2"  
};
```

`<param name="flashvars" value="param1=val1&param2=val2">`

- NAMED ANCHORS

`http://www.mojastrona.pl/#osoba,Jan\_Kowalski,13`



# PRZEKAZYWANIE DANYCH DO FLASHA

- GET i FLASHVARS

```
var fvars:Object = LoaderInfo(this.root.loaderInfo).parameters;  
trace(fvars.param1); // val1  
trace(fvars.param2); // val2
```

- NAMED ANCHORS

```
var myHref:String = String(ExternalInterface.call("flashGetHref"));  
var anchors:Array = myHref.split("#");  
try {  
    var params:Array = anchors[1].split(",");  
    var param2:String = params[1];  
    trace(param2); // Jan_Kowalski  
} catch(e:Error) { }
```



# POBIERANIE DANYCH

Podstawową metodą pobierania danych jest wczytywanie plików tekstowych za pomocą klasy **URLLoader**.

Podstawowe formaty wczytywanych danych:

- **variables**
- **text**
  - **xml**
  - **json**
- **binary (ByteArray)**



# POBIERANIE DANYCH VARIABLES

plik.txt:

```
zmienna1=val1&zmienna2=val2
```

```
var loader:URLLoader = new URLLoader();  
loader.dataFormat = URLLoaderDataFormat.VARIABLES;  
loader.addEventListener(Event.COMPLETE, completeHandler);  
loader.load(new URLRequest("plik.txt"));
```

```
private function completeHandler(e:Event):void {  
    var loader:URLLoader = URLLoader(e.target);  
    trace(loader.data["zmienna1"]); // val1  
    trace(loader.data["zmienna2"]); // val2  
}
```



# POBIERANIE DANYCH XML

dane.xml

```
<?xml version="1.0" encoding="utf-8"?>
  <dane>
    <item id="1">
      <foto>obrazek1.jpg</foto>
      <tytul>Lorem ipsum dolor 1</tytul>
      <link>http://www.onet.pl</link>
    </item>
    <item id="2">
      <foto>obrazek2.jpg</foto>
      <tytul>Lorem ipsum dolor 2</tytul>
      <link>http://www.wp.pl</link>
    </item>
  </dane>
```



# POBIERANIE DANYCH XML

```
var loader:URLLoader = new URLLoader();  
loader.addEventListener(Event.COMPLETE, completeHandler);  
loader.load(new URLRequest("dane.xml"));
```

```
private function completeHandler(e:Event):void {  
    var loader:URLLoader = URLLoader(e.target);  
    var mojeDane:XML = new XML(loader.data);
```

```
    trace(mojeDane.item[0].link); // http://www.onet.pl  
    trace(mojeDane.item[1].@id); // 2
```

```
}
```



# POBIERANIE DANYCH JSON

JSON (JavaScript Object Notation) is a lightweight data-interchange format.

<http://json.org/>

json.txt

```
{"zmienna1":"val1","zmienna2":"val2"}
```

```
import com.adobe.serialization.json.JSON;  
var a:Object = new Object();  
a.zmienna1 = "val1";  
a.zmienna2 = "val2";  
trace(JSON.encode(a));
```



# POBIERANIE DANYCH JSON

```
import com.adobe.serialization.json.JSON;
var loader:URLLoader = new URLLoader();
loader.addEventListener(Event.COMPLETE, completeHandler);
loader.load(new URLRequest("json.txt"));
```

```
private function completeHandler(e:Event):void {
    var loader:URLLoader = URLLoader(e.target);
    var a:Object = JSON.decode(loader.data);
```

```
    trace(a.zmienna1); // val1
    trace(a.zmienna2); // val2
```

```
}
```



# DŹWIĘK

Najczęstsze czynności związane z dźwiękiem:

- doładowywanie zewnętrznych plików mp3
- odtwarzanie, zatrzymywanie i pauzowanie
- odtwarzanie strumienia dźwięku w czasie jego ładowania
- kontrolowanie głośności i balansu
- pobieranie tagów ID3 (mp3)
- przechwytywanie dźwięku z mikrofonu



# DŹWIĘK - KLASY

flash.media.Sound

flash.media.SoundChannel

flash.media.SoundLoaderContext

flash.media.SoundMixer

flash.media.SoundTransform

flash.media.ID3Info

flash.media.Microphone



# DŹWIĘK - WCZYTYWANIE

```
var s:Sound = new Sound(); s.addEventListener  
(Event.COMPLETE, onSoundLoaded);
```

```
s.load(new URLRequest("muzyka.mp3"));
```

```
function onSoundLoaded(event:Event):void {  
    var snd:Sound = event.target as Sound;  
    snd.play();  
}
```



# DŹWIĘK - WCZYTYWANIE

Odtwarzanie z buforowaniem podczas wczytywania:

```
var s:Sound = new Sound();  
var req:URLRequest = new URLRequest("muzyka.mp3");  
var context:SoundLoaderContext = new  
    SoundLoaderContext(8000, true);  
s.load(req, context);  
s.play();
```



# DŹWIĘK

## KONTROLA ODTWARZANIA

```
var snd:Sound = new Sound(new URLRequest("muzyka.mp3"));  
var channel:SoundChannel = snd.play();
```

```
// pauza / stop
```

```
var pausePosition:int = channel.position;  
channel.stop();
```

```
// play
```

```
channel = snd.play(pausePosition);
```



# DŹWIĘK GŁOŚNOŚĆ I BALANS

Na starcie:

```
var snd:Sound = new Sound(new URLRequest("muzyka.mp3"));  
var trans:SoundTransform = new SoundTransform(0.6, -1);  
var channel:SoundChannel = snd.play(0, 1, trans);
```

Podczas odtwarzania:

```
var snd:Sound = new Sound(new URLRequest("muzyka.mp3"));  
var channel:SoundChannel = snd.play(0, 1, trans);  
// gdzieś dalej  
var trans:SoundTransform = new SoundTransform(0.6, -1);  
channel.soundTransform = trans;
```



# VIDEO

Najczęstsze czynności związane z video:

- wyświetlanie i kontrolowanie filmu video
- odtwarzanie filmów z zewnętrznych plików
- pobieranie meta danych oraz tzw. cue points
- przechwytywanie obrazu z kamery



# VIDEO - ODTWARZANIE

```
var nc:NetConnection = new NetConnection();
nc.connect(null);
var ns:NetStream = new NetStream(nc);
ns.addEventListener(AsyncErrorEvent.ASYNC_ERROR, aHand);
ns.play("video.flv");
function aHand(event:AsyncErrorEvent):void
{
    // ignoruj błąd
}
var vid:Video = new Video();
vid.attachNetStream(ns);
addChild(vid);
```



# VIDEO - KONTROLOWANIE

```
// play  
ns.play("video.flv");
```

```
//pauza  
ns.pause();
```

```
// stop  
ns.pause();  
ns.seek(0);
```

```
// play / pauza  
ns.togglePause();
```



DZIĘKUJĘ

