Vytvoření studijních materiálů pro předmět technologie WWW

Stanislav Pěrka

Bakalářská práce
2006

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Prohlašuji, že jsem na celé diplomové práci pracoval/a samostatně a použitou literaturu jsem citoval/a.

Ve Zlíně, 16. 06. 2006 .....

jméno diplomanta

Abstrakt

Hlavním cílem práce bylo vytvořit studijní materiál pro předmět Technologie WWW. Tyto materiály jsou zaměřeny na vytváření interaktivních internetových stránek, založených na technologiích Macromedia Flash včetně programování v jazyce ActionScript a standardech SOAP a WSDL pro implementaci webových služeb. Práce je rozdělena na dvě části. V první je popsán Macromedia Flash a Webové služby, ve druhé části jsou vytvořeny praktické příklady.

Main object of my work was create a study materials for the subject of Technology WWW. These materials are focused on producing interactive website based on Macromedia Flash technology including programming in the ActionScript language and standards SOAP and WSDL which is used to implement web services. The work is divided into two parts. The first part talks about Macromedia Flash and the web services and in the second part are practical examples.

OBSAH

ÚVOD

Cílem mé bakalářské práce bylo vytvoření studijních materiálů pro předmět technologie WWW. Tento předmět se zabývá vytvářením webových aplikací. Zaměřil sem se na vývoj interaktivních aplikací založených na technologiích Macromedia Flash a standardech SOAP, WSDL pro implementaci webových služeb.

Teoretická část je rozdělena na dvě části.V první části jsem popsal prostředí Macromedia Flash a nejdůležitější části jazyka ActionSkript pro programování aplikací a interaktivity webových stránek. V druhé části jsou popsány základní pojmy webových služeb a obecný princip.

V praktické části jsou ukázkové příklady, které si mohou studenti vypracovat. V příkladech je popsán přesný postup vytváření aplikací pro lepší pochopení Macromedia Flash a vytváření webových služeb.

I. TEORETICKÁ ČÁST

1 UVOD DO FLASH

1.1 Co je to Flash?

Zjednodušeně řečeno je to animace - tedy skupina snímků, které se mezi sebou vyměňují (podobně jako animovaný ***GIF***). Tato definice však není úplně přesná. Zatímco u GIFu se střídají ***rastrové*** obrazy (tvořené sítí bodů - ***pixelů***), u Flashe se jedná o ***vektorové*** obrazy (grafika je definovaná pomocí ***čar*** a ***výplní***). A v neposlední řadě ještě připomeňme, že animace může být ovlivňována ***ActionScriptem*** (programovací jazyk podobný JavaScritpu).[1] Ve Flash je tedy možné vytvořit prezentace, kreslené scénky, hry, webové aplikace, animovaná menu, reklamní bannery, celé webové stránky, a tisíce dalších věcí.

1.2 Trocha historie

Počátky Flashe spadají někdy do roku 1994. Tehdy vlastně ještě nešlo o Flash, jak jej známe dnes. Jmenoval se SmartSketch a byl založen na Javě. Od tohoto směru se však ustoupilo. Java jako programovací jazyk totiž nevyhovoval nárokům na rychlost a spolehlivost. Když se někdy v roce 1995 objevily prohlížeče podporující zásuvné moduly typu PLUG-IN, byl SmartSketch přejmenován na FutureSplash Animator a byla zcela změněna jeho podoba. Macromedia v této době pracovala na svém projektu s názvem Shockwave.[1]

V roce 1996 Macromedia kupuje FutureSplash Animator a vzniká tak Macromedia Flash 1.0. Tato verze sice ještě neobsahuje skriptování ActrionScript, ale nastiňuje nadějný směr vývoje webových animací.[1]

Následuje verze 2, která dovoluje základní skriptovou manipulaci s přehráváním animace. Objevují se prvky jako tlačítko a grafika, vzniká proměnná.

Verze 3 přináší ozvučení animací a s tím spojené příkazy, dále pak příkazy typu fscommand a možnost skriptově načítat SWF animace.[1]

Verze 4. je doslova revoluční. Celý ActionScript je přepracován, vzniká velké množství příkazů. Vznikají funkce, příkazy pro movieclip, podmínky a smyčky, nové operátory, načítání proměnných ze souboru a spoustu dalších.[1]

Verze 5 je opět přelomová. Vznikají objekty se svými metodami a vlastnostmi. Je možno vytvářet vlastní funkce. Vznikají komponenty. Většina příkazů je přeorientována

na objekty. Vzniká přehlednější dot syntaxe a podporována je komunikace se serverem pomocí XML Socket.[1]

Verze 6 (MX) přináší podstatné rozšíření objektů a metod. Vznikají UI komponenty, vzniká spolupráce s videem. Flash player 6 podporuje obousměrný streamovaný přenos zvuku a videa pomocí kamer a mikrofonů. Je vyvinut nový komunikační protokol RTMP a Server-side ActionScript pro komunikaci se serverovými službami a serverový balík Flash Communication Server MX.[1]

A konečně 7. verze pojmenovaná MX 2004. Toto označení ve vás možná evokuje pocit, že se jedná jen o nějaký upgrade, ale není tomu tak. Tato verze má dvě podoby: MX 2004 a MX 2004 Pro. Obě verze nabízí vylepšené rozhraní, nové efekty (zejména pro práci s textem - podpora CSS!), vylepšenou časovou osu, rozšíření kompatibility importovaných objektů (PDF, EPS, …), konečně také panel "historie", šablony pro vytvoření složitých Motion a Shape Tweenů, vylepšené trasování bitmap a ActionScript 2.0, který se vyznačuje větší přímočarostí a robustností.[1]

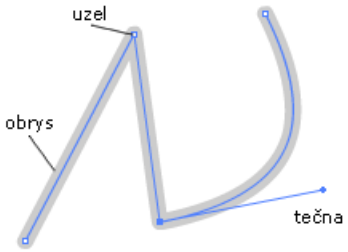
Verze Professional navíc nabízí podporu externího ActionScriptu, lepší zpracování video souborů, podporu zvuků ve formátu MIDI pro mobilní zařízení, vylepšené formuláře a jednoduché vytváření slidů a prezentací ve speciálním módu.[1]

1.3 Základní pravidla

Před samotnou prací s Flashem je třeba si vysvětlit několik skutečností bez kterých se neobejdete.

1.3.1Čára

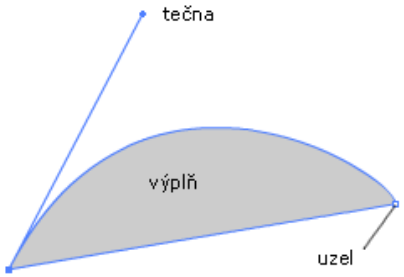
Je to objekt, který je definován jednotlivými kotvícími body (též uzly), které jsou spojeny buď úsečkou, nebo křivkou. Křivka je vždy definována ***tečnou*** v počátečním a koncovém bodě. Čára samotná ***nemá*** výplň, lze ji definovat pouze ***obrys*** (barvu a tloušťku). Bez ohledu jak je obrys čáry tlustý, vždy se jedná o čáru, nikoliv o polygon.[2]



Obr.1.Popis čáry vykreslené pomocí vektorů[2]

1.3.2 Výplň

Je to plocha s definovanou barvou. Ve Flashi ***může*** (narozdíl od např. CorelDRAW) výplň existovat ***nezávisle*** na čáře. Zní to divně, ale výplň zde není ohraničena čarou, ale má svoje vlastní uzly spojené přímkami a křivkami, které se chovají stejně, jako čára. Přestože je výplň ohraničena svou "speciální" čarou, ***nelze*** ji definovat obrys.[2]

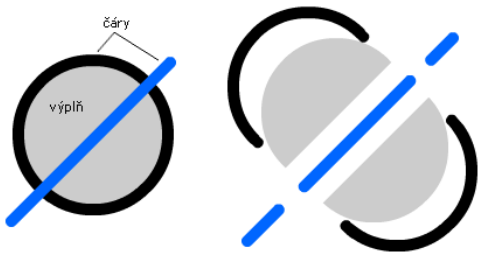


Obr.2.Popis výplně vykreslené pomocí vektorů[2]

1.3.3 Vrstvy

Flash animace umí pracovat v několika vrstvách. Vrstvy si lze představit, jako průhledné fólie s nakreslenými objekty, které jsou položeny na sobě a dohromady tvoří celý obrázek. Důležitou vlastností vrstev je, že jsou na sobě ***nezávislé***. Znamená to tedy, že objekty nakreslené v různých vrstvách na sebe nemají ***žádný vliv***.[2]

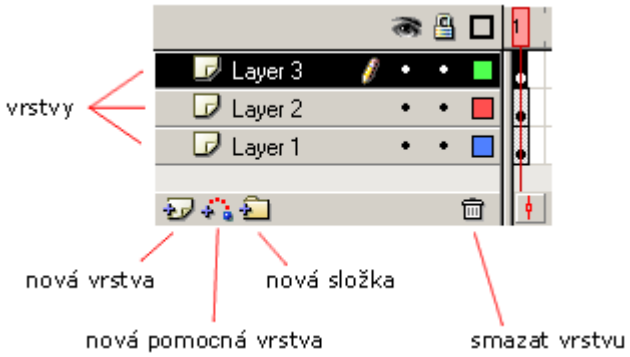
Pokud se objekty překrývají, rozdělí se na díly určené jejím průnikem.



Obr.3.Objekty v různých a stejných vrstvách[2]

pozn.: dělící rovinu ***neovlivňuje*** tloušťka obrysu čáry.

Vrstva na nejvyšší úrovni je nahoře (Layer 3) a na nejnižší dole (Layer1). Vrstvy se vytváří a odstraňují následujícími tlačítky:

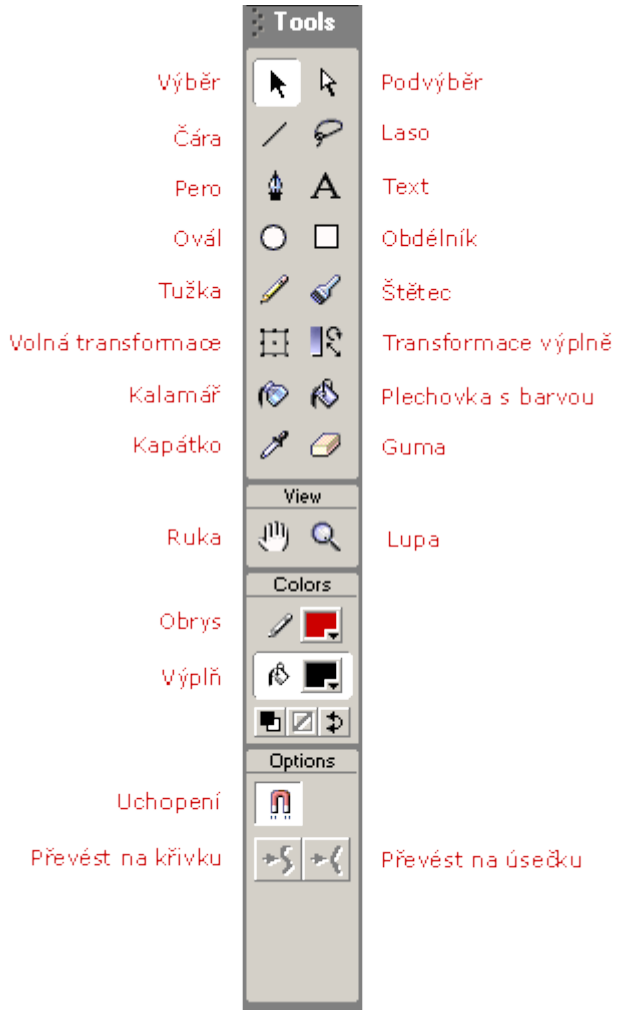


Obr.4.Popis tlačítek pro práci s vrstvami[2]

Název vrstvy lze měnit jedním kliknutím na něj. Každou vrstvu lze ***zhasnout*** (symbol oko) nebo ***zamknout*** (symbol zámek). Barevný čtvereček vpravo zobrazuje barvu, která reprezentuje danou vrstvu. Když potom kliknete do prostoru čtverečku, budou objekty v této vrstvě zobrazeny jen ***obrysově*** (v nastavené barvě).[2]

1.4 Základní nástroje

V této kapitole se dozvíte jak ovládat základní nástroje Flash.



Obr.5.Popis panelu nástrojů[3]

Výběr

Slouží k výběru objektu. V poli "Options" můžete nastavit:

- Uchopení - objekty se budou uchopovat na již nakreslené objekty
- Převést na křivku - vyhladí vybranou čáru
- Převést na úsečku - zruší zakřivení vybrané čáry

Podvýběr

Slouží k editaci jednotlivých uzlů čáry (nebo obrysu výplně). Zobrazení uzlů provedete jedním kliknutím na objekt. Vybráním uzlu a následném zmáčknutím "ALT" a tažením měníte (vytvářite) zakřivení čáry.[3]

Čára

Kreslí rovné čáry. Tloušťka nastavíme na panelu "Properties"

Laso

Stejné využití jako výběr, ale můžeme označovat plochy libovolného tvaru.

Pero

Kombinuje nástroje "Podvýběr" a "Čára". Jedním kliknutím vytvoříte bod a dalším klikem (o kousek dál) vytvoříte druhý bod, který se spojí s prvním. Umožňuje také přidávat a ubírat uzly.[3]

Text

Vložení textu.V panelu Properities je možné nastavit vlastnosti.

Ovál a Obdélník

Kliknutím a tažením vytvoříte obrazec.

Tužka

Umožňuje kreslit čáry rukou. V panelu "Options" můžete nastavit podobu čáry:

- Straighten - nakreslená čára se změní v rovné úsečky
- Smooth - čára bude mít podobu hladké křivky
- Ink - čára zůstane taková, jakou jste ji nakreslili

Štětec

Kreslí výplně. V panelu "Options" můžete nastavit:

- Tloušťku stopy (Brush Size)
- Typ stopy (Brush Shape)
- Proměnlivou tloušťku stopy (Use Pressure)

Volná transformace

Pomocí tohoto nástroje lze vybraný objekt ručně otáčet, zešikmovat a měnit jeho velikost

V poli "Options" je možné upřesnit, co chcete s objektem udělat:

- Otočit
- Zvětšit
- Roztáhnout jednotlivé body
- Zdeformovat

Transformace výplně

Umožňuje měnit velikost, natočení a zkosení přechodové výplně.

Plechovka s barvou a Kalamář

Umožňují měnit barvu výplně nebo barvu čáry.

Kapátko

Kliknutím na libovolný objekt levým tlačítkem nastavíte barvu objektu do pole colors.

Guma

Maže nakreslené objekty. Můžeme v panelu "options" nastavit:

- Co se má smazat
    - Normal = no comment
    - Fills = maže jen výplně
    - Lines = jen čáry
    - Selected fills = jen předem vybrané výplně
    - Inside = maže jen jeden objekt najednou
- Velikost a tvar stopy gumy
- Smazání celého objektu jedním kliknutím

Ruka a Lupa

Ruka umožňuje posouvat pohled na papír (kliknutím a tažením) a lupa ovládá přiblížení.

Obrys a Výplň

Umožňuje nastavit barvu obrysu a výplně před použitím nástroje

Panel Options

Na tomto panelu se zobrazují doplňující funkce každého nástroje

1.5 Knihovna

Každý nakreslený objekt (skupina objektů) lze ve Flashi převést na tzv. ***symbol*** a umístit jej do knihovny (library) - knihovnu zobrazíte stiskem F11. Poté lze vložit z knihovny do animace kopii tohoto symbolu (tzv. ***instanci***). Je jedno, kolik instancí symbolu vložíte do animace - symbol bude uložen jen jednou.[4]

1.5.1 Druhy symbolů

Existují 3 druhy symbolů:

Grafika (Graphic)

Může obsahovat pouze nehybnou grafiku.

Tlačítko (Button)

Obsahuje 4 snímky (Up, Over, Down, Hit).

- *Up* - snímek, který je vidět normálně

- *Over* - snímek, který se zobrazí při přejetí myši nad tlačítkem
- *Down* - snímek, který se zobrazí po kliknutí na tlačítko
- *Hit* - tento snímek není nikdy viditelný - určuje oblast, která má být citlivá na kliknutí (nemusí se shodovat s předchozími objekty)

Klip (Movie Clip)

Symbol, který může obsahovat samostatnou pohyblivou animaci. (má vlastní časovou osu)

1.5.2Transformace instance

Instance je vlastně kopie symbolu. Tato kopie ale nemusí být úplně identická - lze upravit tyto parametry:

- Výška a šířka (width, height)
- Rotace (rotate)
- Sklon (skew)
- Jas (brightness)
- Barevný odstín (tint)
- Průhlednost (alpha)

První 3 body znáte už z panelu "transformace", poslední 3 jsou nové. Abyste je mohli editovat, musíte jednou kliknout na instanci a na panelu "properties" se objeví roletové menu.

Kromě jmenovaných 3 druhů symbolů může být také do knihovny vložen:

- ***Rastrový obrázek***

  platí pro něj podobná pravidla, jako pro grafiku, ale jeho instanci lze upravit jen výšku, šířku, rotaci a sklon.
- ***Zvuk***

  Můžeme použít základní druhy zvukových formátů jako-MID,WAV,MP3
- ***Složka***

  Slouží k přehlednější organizaci objektů v knihovně.
- ***Komponent***

  Je to zvláštní typ MovieClipu, který plní nějakou funkci (např. výběrové pole). Po jeho vložení do animace je možno v okně "Properties" nastavit některé parametry komponentu. Tento symbol bývá také někdy označován jako SmartClip.[4]
- ***Font***

  Font můžete do knihovny umístit klepnutím na tlačítko  a zvolením "New

Font". Následně vyberete jeden font, který máte nainstalován ve Windows a do kolonky výše napíšete jeho nové jméno, které jej bude v animaci reprezentovat. Tento postup nedoporučuji používat pro klasické texty a formuláře. Při psaní textu se font sám přibalí v potřebném rozsahu a u formulářových prvků (Input a Dynamic Text) je možné přesně definovat, jaké znaky se mají přibalit. Umístění fontu do knihovny se dá využít jedině pro potřeby ActionScriptu.[4]

- ***Video***
  Flash umožňuje vkládat do animace videa ve formátech MPEG, DV, MOV a AVI.

1.6 Snímky, pohyb objektů

V této části si ukážeme jak rozhýbat objekty na časové ose.

1.6.1 Časová osa a snímky

Snímky jsou ve Flashi rozmístěné do časové osy, která se přehrává určitou rychlostí (Frame Rate).Základním stavebním kamenem je ***klíčový snímek*** (keyframe). Tento snímek může být buď prázdný (není v něm nic nakresleno) nebo plný (obsahuje nějakou grafiku).[5]

Přidávání a mazání snímků:

- F5 - přidat normální snímek
- F6 - přidat klíčový snímek (obsahuje stejnou grafiku jako předchozí kl. snímek)
- F7 - přidat prázdný kl. snímek
- Smazat snímky můžete jejich vybráním, kliknutím pravým tlačítkem a zvolením ***Remove Frames***
- Vybráním jednoho nebo více snímků a jejich přetažení na novou pozici je můžete přesunout na libovolné místo v časové ose (i do jiné vrstvy)
- Pozor - výběrem snímku a stiskem DEL smažete jen obsah snímku (platí jen ve verzi MX).

1.6.2 Motion Tween (Pohybové vykreslení)

Vytvoříme ho tak, že umístíme za sebe 2 klíčové snímky. První klíčový snímek uděláme delší. Čím je první kl. snímek delší, tím pomalejší a plynulejší bude pohyb.

Pozor, je nutné, aby první i druhý klíčový snímek obsahoval ***instance stejných symbolů!!***

Pokud tedy dám do prvního kl. snímku instance symbolů ***A*** a ***B***, musím dát do druhého zase ***A*** a ***B***. Instance symbolů ve druhém kl. snímku poté můžete upravovat (poloha, velikost, odstín, rotace, průhlednost, zkosení. Poté klikněte pravým tlačítkem na první snímek a zvolte ***Create Motion Tween***. Jiná možnost, jak to provést je vybrat první snímek a na panelu ***Properties*** se objeví roletové menu ***Tween***. Zde změňte položku ***None*** na ***Morión.*** [5]

Na panelu Properties je možné nastavit ještě některé detaily pohybu:

Ease – Postupné zrychlování nebo zpomalování pohybu

Rotate - Pokud necháte na "Auto", bude se objekt otáčet podle koncového snímku. Můžete nastavit rotaci o 360 po směru nebo protisměru hodinových ručiček. Do kolonky vpravo se zadává kolikrát se má objekt otočit.

Scale - Zaškrtnutím povolíte postupnou změnu velikosti.

Orient to Path - Orientovat pohyb na trasu

Sync - Příkaz Synchronizovat použijte pokud počet snímků v animované sekvenci uvnitř MovieClipu není sudým násobkem počtu snímků v celkové pohybové sekvenci.[5]

Snap – Přichycení k trase pohybu.

1.6.3 Shape Tween (Tvarové vykreslení)

Zatímco u Motion Tweenu jsme mohli pohybovat pouze instancemi stejných symbolů, u Shape Tweenu je to právě naopak. Abychom mohli vytvořit tento pohyb, ***MUSÍ*** počáteční i koncový snímek obsahovat pouze ***čistou grafiku***.[5]

Vytvoření tohoto pohybu je podobné jako v předchozím případě. Nejprve vytvoříme 2 klíčové snímky (nesmí obsahovat instance) a první roztáhneme na požadovanou velikost. Poté vybereme první snímek a na panelu ***Properties*** zvolíme v roletovém menu ***Tween*** položku ***Shape.***[5]

V panelu Properties je možné nastavit:

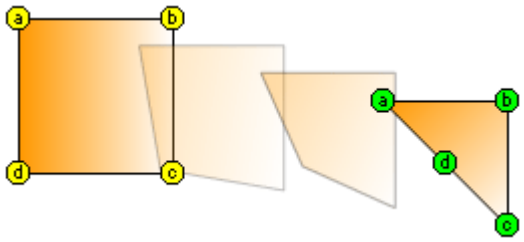
- ***Ease*** – Plynulé zrychlování nebo zpomalování pohybu.
- ***Blend*** - Způsob překreslování v přechodových snímcích:

  Distributive - hladší a zaoblené tvary

  Angular - jasné rohy, rovné čáry

1.6.4 Používání stop tvarů

Při překreslování objektu se může stát, že přeskupování křivek probíhá nevhodným způsobem. Při použití Shape Tweenu je možné určit body, které by měly sobě odpovídat v počátečních a koncových tvarech.



Obr.6.Postupná deformace objektu s nastavenými body(Hint)[5]

Klikněte na první snímek sekvence a v menu ***Modify*** vyberte ***Shape*** - ***Add Shape Hint*** Objeví se vám červené kolečko s písmenem (a-z), posuňte ho na danou křivku (musí zežloutnout). Poté vyberte poslední snímek pohybu a posuňte všechny body do svých pozic (musí zezelenat). Bod odstraníte jeho uchopením a přetažením mimo pracovní plochu.[5]

2 ACTION SCRIPT

ActionScript jako takový vám umožňuje vložit do projektu v programu Flash pokyny k různým akcím i pokyny spojené s logikou.Podobně jako všechny jazyky, i Action skript obsahuje mnoho prvků, jako sou slova, interpunkce a struktura.[6]

Actionscript dovoluje přidat do animací interaktivitu a změnit ji třeba na přehledné menu, webovou stránku, zábavnou hru, nebo v animovaný formulář.

2.1 Řízení animace

Skript je možno ve Flashi vložit buď na:

- ***klíčový snímek v časové ose*** - příkazy se vykonají v aktuálně přehraném snímku.
- ***objekt***:
    - tlačítko
    - movie clip

V tomto případě jsou příkazy vykonány po určité události

Skript můžete do snímku nebo objektu vložit tak že ho označíte a zmačknete F9 nebo kliknete na panel ActionScript.

Pokud je příkaz vložen tlačítku nebo movieClipu provedou se příkazy po určité události.

Tab.1.Události tlačítka

Press	zmáčknutí levého tl. myši
Release	puštění levého tl. myši
Release Outside	puštění tl. mimo instanci
Key Press	zmáčknutí klávesy
Roll Over	přejetí kurzorem dovnitř
Roll Out	přejetí ven
Drag Over	tažení dovnitř
Drag Out	tažení ven

Příklad:

```
on (release) {
    gotoAndPlay(5);
}
```

Tento zápis znamená že po kliknutí na tlačítko se provedou příkazy uvnitř složených závorek.

Všechny události, které se používají pro tlačítko je možné použít i pro MovieClip.

Tab.2.Události MovieClipu

Load	po načtení nebo vygenerování MC
Unload	po odstranění MC z časové osy
Enter Frame	akce jsou vykonány v každém snímku MC
Mouse Move	pohnutí kurzorem myši (kdekoliv v animaci)
Mouse Down	stisk levého tl. myši (kdekoliv v animaci)
Mouse Up	uvolnění levého tl. myši (kdekoliv v animaci)
Key Down	stisk klávesy
Key Up	uvolnění klávesy
Data	obdržení údajů z LoadVariables a LoadMovie

2.2 Adresování příkazů

Flash je rozdělen na hlavní časovou osu _root a na dílčí movie clipy. Pokud z hlavní časové osy odkazujete na nějaký movie clip, který je na hlavní scéně, stačí použít název instance.

Problém ale vzniká, pokud v tomto movie clipu je ještě nějaký jiný movie clip. Tento druhý movie clip totiž sám o sobě není na hlavní scéně. Je v movie clipu a přesně to také musíme sdělit programu.[7] Takže například pro absolutní adresu movie clip(auto) na hlavní scéně byste použili tento skript:

_root.auto

Příkaz _root určuje hlavní časovou osu. Pak už stačí přidat nějakou funkci, třeba:

_root.auto.gotoAndStop(2);

Instance my_movieclip se tak přesune na druhý snímek a zastaví se. Pozor! Přesune se časová osa movie clipu, nikoliv hlavní časová osa. Ta zůstane nezměněna. Pokud potom máte movie clip(pneu) v movie clipu(auto), vypadala by absolutní cesta takto[7]:

_root.auto.pneu.gotoAndStop(2);

Nyní by se instance movie clipu "pneu" v instanci movie clipu "auto" přesunula na druhý snímek a zastavila by se. Absolutní cesta je neměnná a je úplně jedno odkud na cílenou instanci ukazujeme. Výše uvedený skript by se správně provedl, kdyby byl zapsán na hlavní časovou osu, i kdyby byl vložen do nějaké jiné instance nějakého symbolu. Problém by to dělalo, pokud by cesta byla zapsána relativně.[7]

Pokud byste na hlavní časovou osu přidali skript:

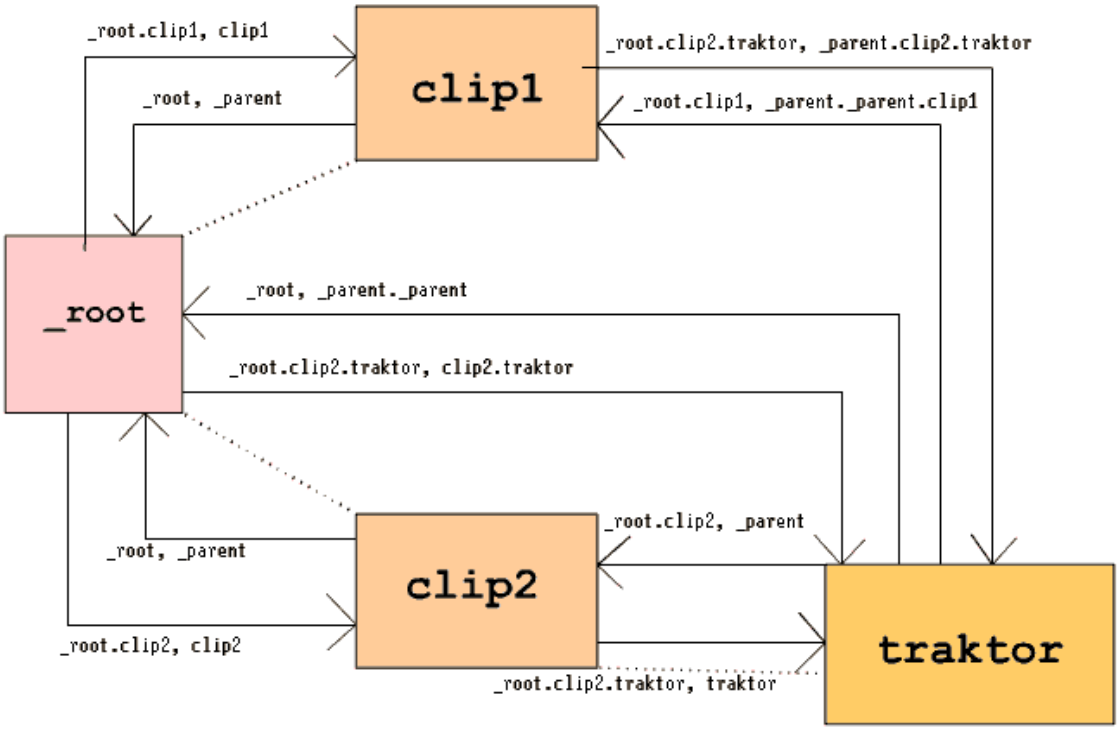
bomba.gotoAndPlay(3);

Instance bomba by se přesunula na třetí snímek a pokračovala v přehrávání jen pokud by bylo umístěna na hlavní scéně. Pokud by instance bomba byla skryta v nějakém jiném movie clipu (např. ve výše jmenované instanci "auto"), Flash by hledaný objekt nenašel. Pokud totiž cesta nezačíná _root, Flash začíná s hledáním na úrovni, kde je skript zapsán[7].

Pokud se zahrabáváte s cestami stále níž a níž, musíte používat přesnou cestu, ale pokud ukazujete z MovieClipu v nižší úrovni na vyšší, můžete použít příkaz **_parent**, který

určuje nadřazený symbol. Pokud potřebujete, aby byl skript namířen na symbol, na kterém je skript umístěn, použijte příkaz **this**.[7]

Přehledně to všechno ukazuje následující obrázek:



Obr.7.Popis správného adresování

2.3 Proměnné

Deklarace proměnných je vcelku jednoduchá. Samotnou deklaraci můžete umístit kamkoliv, ale doporučuje se zadeklarovat proměnnou tam, kde ji budete používat. Ale jinak je to v podstatě jedno - může být na movie clipu i na časové ose.

Samotná deklarace vypadá takto:

```
var moje_promenna=25;
jina_promenna="Hello World";
_global.promenna=x;
```

Všechny tři deklarace jsou možné. První dvě jsou v podstatě totožné, slovo var je tam v podstatě zbytečné, je to přežitek ze staršího Flashe. Pokud tam var nebude, vlastně se vůbec nic neděje. Poslední deklaraci používejte jen tehdy, pokud víte, že budete s proměnnou často pracovat v různých úrovních. Proměnnou můžete pojmenovat

jakkoliv, ale nesmí začínat číslem, nebo nějakým nevhodným znakem jako > < : " ? * apod. Také nesmí být identická s nějakým klíčovým slovem Flashe.[8]

2.4 Datové pole (Array)

Pole narozdíl od proměnných však neobsahují jednu hodnotu, ale více separovaných hodnot, které jsou očíslovány (index). Je potřeba si uvědomit, že Array je objekt a potřebuje tedy konstruktor.[9]

Příklad:

pole = new Array("*a***", "***b***", "***c***", *14*, "***ahoj***");**

Je také možné vytvořit pole s určitým počtem prázdných položek:

pole = new Array(5);

Tab.3.Metody objektu Array

concat	spojuje pole dohromady a prvky seřadí za sebe: `pole1.concat(pole2, pole3, …)`
join	vypíše obsah pole jako řetězec a vloží mezi položky separátor uvedený v závorce: `pole.join("+")`
pop	odstraní poslední prvek z pole a vypíše jeho hodnotu: `promenna = pole.pop()`
push	Přidá prvky na konec a vypíše novou délku pole: `promenna = pole.push("hrušky", "jablka")`
reverse	Převrátí pořadí prvků: `pole.reverse()`
shift	Odstraní první prvek a vypíše jeho hodnotu: `promenna = pole.shift()`
slice	Vyřízne část pole a udá jej jako nové pole: `pole.slice(2,5)` …bude vyříznut 2. - 5. prvek
sort	Seřadí prvky pole buď podle velikosti, nebo pomocí funkce
sortOn	Seřazení podle názvu prvku
splice	odstraňuje/přidává prvky: `pole.splice(2,5,"nový1", "nový2",...)` bude odstraněno 5 prvků od prvku 2 a od tohoto místa budou vloženy prvky "nový1" a "nový2"
toString	vypíše hodnoty prvků oddělené čárkami
unshift	Vloží prvky na začátek a vypíše novou délku pole: `pole.unshift("první", "druhý", "třetí")`

2.5 Podmínky a smyčky

Podmínky a smyčky umožňují ovlivňovat sled provádění příkazů na základě nějaké skutečnosti.

Podmínky

Podmínka je útvar, který kontroluje pravdivost nějaké skutečnosti a podle toho buď provede nebo neprovede příkazy napsané uvnitř závorek. Stejné jako v C++.

Příklad:

```
if (vyhra == 0 && hotovost < 100) {
  gotoAndPlay(10);
} else {
  gotoAndPlay(1);
}
```

Smyčky

Smyčka také kontroluje pravdivost zapsaného výrazu. Pokud je splněn příkazy se vykonávají stále dokola.

Existuje víc druhů smyček:

- ***While***

  Klasický typ smyčky. Výraz je kontrolován na začátku a dokud je pravdivý jsou příkazy prováděny pořád dokola.

  příklad:

  ```
  i = 5;
  while (i > 0){
     myMC.duplicateMovieClip("newMC" + i, i);
     i--;
  }
  ```

- ***Do...While***

  Úplně stejný princip jako v předchozím případě, jen z malým rozdílem.
  Pravdivost zapsaného výrazu se kontroluje až na konci smyčky.

  příklad:

  ```
  i = 5;
  do {
     myMC.duplicateMovieClip("newMC" + i, i);
     i--;
  } while (i > 0);
  ```

- ***For***

  Poslední a nejpoužívanější typ smyčky ve Flashi. Uvnitř kulatých závorek jsou definovány 3 výrazy (oddělené středníkem ";"). Stejný princip jako v C++.

```
for (i = 4; i > 0; i--) {
  myMC.duplicateMovieClip("newMC" + i, i);
}
```

Někdy potřebujeme provádění smyčky přerušit, nebo naopak vrátit zpět na začátek (zkontrolovat pravdivost výrazu). Pro tyto účely existují příkazy ***Break*** (přerušení) a ***Continue*** (zpět na začátek). Viz příklad:

```
i = 0;
while (limit = 0) {
                    if (i >= 10) {
                                  break;
                                  }
                      i++;
                      }
```

Tento zápis znamená, že pokud proměnná "***i***" dosáhne hodnoty 10, vykoná se příkaz ***break*** a smyčka se přeruší.

Switch (přepínač)

Switch je zvláštní typ elementu ovlivňujícího tok příkazů, který se objevil ve verzi MX (6).

Má stejnou syntaxi jako v C++.

příklad:

```
abc = 2;
switch (abc) {
    case 1 :
        gotoAndPlay(1);
        break;
    case 2 :
        gotoAndPlay(5);
        break;

    case 3 :
        gotoAndPlay(10);
        break;
}
```

2.6 Funkce

Funkce je určitý blok kódu s přiřazeným názvem, který lze kdykoliv provést. Když je funkce volána, je jí přiřazen vstup (argumenty). Vykoná několik operací a pak generuje output (vrácenou hodnotu). K definování funkce slouží příkaz ***function.*** Syntaxe je stejná jako v C++.

- **Jednoduchá funkce bez vstupu a výstupu:**

```
function prihlasovaci_formular() {
   prvni_kolonka = "Jan Novák";
   druha_kolonka = "720623/9876";
   treti_kolonka = "podmenapivo";
}
```

Volání funkce:

```
on (release) {
     prihlasovaci_formular();
}
```

- **Funkce se vstupními argumenty:**

```
function prihlasovaci_formular(jmeno, rodne_cislo, heslo) {
   prvni_kolonka = jmeno;
   druha_kolonka = rodne_cislo;
   treti_kolonka = heslo;
}
```

Volání funkce:

```
on (release) {
   prihlasovaci_formular("Jan Novák", "720623/9876",
"podmenapivo");
}
```

- **Funkce se vstupními argumenty a výstupem:**

```
function vypocet(body, koeficient) {
     return body * koeficient;
}
```

Volání funkce:

```
konecne_skore = vypocet (100, 13)
```

Předdefinované funkce

Jsou to funkce definované samotným flashem v každé animaci.

Boolean(výraz) - Slouží k transformaci obsahu proměnné na booleovskou hodnotu (true/false)

escape(řetězec) - Konvertuje textový řetězec (string) na URL-encoded.

eval(název proměnné nebo vlastnosti) - Řetězec, který má být chápán jako název proměnné nebo vlastnosti.

```
pozdrav1 = "ahoj";
pozdrav2 = "čau";
pozdrav3 = "nazdar";
pocitadlo = 2;
zobrazit = eval("pozdrav" + pocitadlo);
```

- proměnná "***zobrazit*** bude mít hodnotu "***čau***"

getProperty(instance, vlastnost) - Funkce získávající vlastnosti instance MovieClipu na scéně

getTimer() - Udává počet milisekund, které uplynuly od doby, kdy začalo movie hrát

getVersion() - Tato funkce vrací textový řetězec udávající verzi operačního systému a Flash přehrávače. Například: `WIN 5,0,17,0`

isFinite(číslo) - Vrací `true`, pokud je číslo konečné a `false`, pokud je nekonečné

isNaN(výraz) - Udává `true` pokud výraz není číslo, jinak `false`

Number(výraz) - Konvertuje výraz na číslo. Pokud je konvertován textový řetězec obsahující písmena je vrácena hodnota `NaN`

parseFloat(řetězec) - Funkce analyzující čísla v textovém řetězci po znacích tak dlouho, dokud nenarazí na nenumerický znak:

`parseFloat("-2")` udá `-2`

`parseFloat("2.5")` udá `2.5`

`parseFloat("3.5e6")` udá `3.5e6` nebo `3500000`

`parseFloat("abcdefgh")` udá `NaN`

`parseFloat("abcdefgh6976")` udá `NaN`

`parseFloat("123abcd584efgh")` udá `123`

parseInt(výraz, radix) - Podobně jako v předchozím případě analyzuje číslo z textového řetězce. Umožňuje konvertovat i jiná, než dekadická čísla (až 36ková čísla):
`parseInt ("123")` udá `123`

```
parseInt ("123abcdefg") udá 123
parseInt ("123", 10) udá 123
parseInt ("3E8", 16) udá 1000
parseInt ("0x3E8") udá 1000
parseInt ("abcdefg123abc123") udá NaN
```

String **-** Konvertuje výraz na textový řetězec. Pokud obsahuje čísla, jsou také chápány jako text

targetPath **-** Vrací dot zápis pozice instance MovieClipu. Například: `_root.prvni.druhy`

2.7 Atributy instancí

Každé instanci je možno definovat:

- Výška a šířka (width, height)
- Rotace (rotate)
- Sklon (skew)
- Jas (brightness)
- Barevný odstín (tint)
- Průhlednost (alpha)

Tyto (a ještě jiné) vlastnosti lze měnit během přehrávání pomocí ActionScriptu ale ***pouze*** u instancí ***MovieClipu***.

K tomu slouží příkaz ***setProperty*** (Actions - MovieClip control)

setProperty("prvni", _rotation, "30"***)***;

Tento příkaz otočí instanci "prvni" o 30°

Je možné také použít zjednodušený zápis:

prvni.***_rotation*** = 30

Efekt bude stejný

Pokud bychom chtěli vlastnost instance místo nastavení získat, je to možné například takto:

nejaka_promenna = ***getProperty(***"prvni", _rotation***)***;

nebo opět zkráceně:

nejaka_promenna = ***prvni._rotation***

V našem případě by byla proměnná rovna 30.

Myslím, že popisovat všechny atributy nebude nutné. Snad jen u některých by mohli vzniknout nějaké nejasnosti:

_focusrect (true/false) - určuje, zda se má objevovat žlutý rámeček kolem tlačítek při navigaci pomocí klávesy TAB

_highquality (0/1/2) - nastavuje kvalitu antialiasingu (0-nejhorší, 2-nejlepší)

_name ("nový název") - změní název instance

_soundbuftime (sekundy) - hodnota udává počet sekund, o které se tekoucí zvuk prebuffer zadrží přenášená data ve vyrovnávací paměti. Nastavená hodnota je 5 sekund.

_visible (true/false) - určuje, zda má být objekt vidět nebo ne

_xscale, ***_yscale*** (procento) - procentuální výška a šířka instance

X a Y Position (pixel) - pozice vzhledem k levému hornímu rohu papíru (nahoru -, dolů +, doleva -, doprava +)

2.8 UI komponenty

Co je to komponent?

Komponent, někdy také zvaný SmartClip je MovieClip, který je naprogramovaný pro určitý účel, z něhož jsou na povrch "vyvedeny" některé proměnné, které jsou důležité pro chod komponentu. Tyto proměnné pak může animátor editovat. Komponent si lze představit jako zamčenou černou skříňku s ovládacím panelem (který však vidí ***jen animátor***). Komponent ze strany uživatele vypadá jako ***obyčejný MovieClip***. Komponenty jako takové existují už od verze 5, ale UI komponenty používají nové funkce ActionScriptu, takže ve verzi 5 nefungují.[9]

Instanci UI komponentu umístíte na scénu jeho přetažením z okna "***Components***" (Window -> Components). Jen doplním, že všechny příkazy ovlivňující UI komponenty naleznete v sekci "***Flash UI Components***"

Druhy komponentů

Druhy UI komponentů prakticky vychází z formulářů v jazyce HTML.

- ***PushButton*** (tlačítko)
- ***ComboBox*** (roletové menu)
- ***ListBox*** (roletové menu - trvale otevřené)
- ***CheckBox*** (zaškrtávací pole)
- ***RadioButton*** (výběrové pole)
- ***Scrollbar*** (posuvník)
- ***SrollPane*** (něco jako frame - umožňuje zobrazovat MovieClip)

2.9 Objekty - úvod

Objekt je element sloužící k získání nějakých informací nebo k nastavení nějakých vlastností. Například objekt ***Date()*** obsahuje informace o aktuálním čase a datumu. Drtivá většina přednastavených objektů ve Flashi má své ***metody*** (methods). Jsou to vlastní funkce definované objektu. Například metoda ***getMonth()*** vytáhne z objektu ***Date()*** číslo aktuálního měsíce. Metoda ***getMilisecond()*** zase získá počet milisekund, atd.[9]

Většina objektů má svůj ***konstruktor***. To je funkce, která vytvoří objekt. Můžeme si to představit jako funkci, která vytvoří instanci přednastaveného objektu v naší animaci. Abychom tedy mohli pracovat s naším objektem ***Date()***, je nutné jej nejprve vytvořit pomocí konstruktoru ***new***:[9]

```
datum = new Date();
hodiny = datum.getHours();
minuty = datum.getMinutes();
sekundy = datum.getSeconds();
milisekundy = datum.getMiliseconds();
```

Zde je tedy vytvořen objekt typu ***Date()*** s názvem "***datum***"**.** Dále jsou definovány proměnné "hodiny", "minuty",...atd jako ***metody*** objektu "***datum***".

Kromě metod může mít objekt i vlastnosti. Například objekt ***Sound()***

```
zvuk = new Sound();
zvuk.attachSound("potlesk");

zvuk.start();
delkazvuku = zvuk.duration;
```

Zde je vytvořen objekt ***Sound()*** s názvem "***zvuk***" a je mu přidělen zvuk z knihovny s názvem "potlesk". Následuje metoda ***start(),*** která spustí zvuk. Nakonec je vlastnost ***duration*** (délka) uložena do proměnné "delkazvuku". [9]

Říkal jsem, že většina objektů má konstruktor. Některé ho ale nemají. Bývají to většinou "hmotné" objekty (***Button***, ***MovieClip***, ***TextField***,...). Tyto objekty jsou již vlastně vytvořeny tím, že je umístíme na scénu a pojmenujeme. Naopak imaginární objekty (***Date***, ***Sound***, ***Color***, ***Array***) existují jen v kódové podobě a je tedy nutné je vytvořit pomocí konstruktoru.[9]

Například objekt ***Button***

```
tlacitko1.enabled = false;
```

Tento zápis edituje vlastnost ***enabled*** na ***nepravda***. Tlačítko s názvem "tlacitko1" tedy nepůjde zmáčknout.

2.9.1 Vlastní objekt

Kromě předdefinovaných objektů je možné vytvořit i vlastní objekt.

Nejprve je nutné vytvořit ***Konstruktor*** objektu. Není to nic víc, než obyčejná funkce, jejíž název je shodný s názvem objektu, který bude vytvářet, takže například:

```
function Kruh() {
        this.addProperty("polomer",getRadius, setRadius);

        function getRadius() {
                return radius;
        }
        function setRadius(hodnota) {
                radius = hodnota;
        }
        this.getArea = function() {
                area = Math.PI * radius * radius;
                return area;
        }
}
```

Zde je definována konstrukční funkce ***Kruh()***. Uvnitř je nejprve definována vlastnost "polomer". K této vlastnosti musí být připojeny 2 funkce. Jedna (***getRadius***) získává hodnotu vlastnosti a druhá (***setRadius***) vlastnost nastavuje. Funkce ***getRadius()*** pouze vrátí hodnotu proměnné "***radius***". Funkce ***setRadius()*** nastaví hodnotu proměnné "***radius***" na hodnotu argumentu "***hodnota***". Nakonec ještě definujeme ***metodu***, která bude vracet obsah kruhu. Metodu můžete definovat pomocí příkazu ***Actions->User-Defined Function->Metod***. Uvnitř metody je definována proměnná area jako obsah kruhu ($PI*radius^2$). Hodnotu této proměnné potom metoda vrací (return).[9]

Máme tedy vytvořený konstruktor a nyní můžeme vytvořit samotný objekt:

prvni = ***new Kruh()***;
prvni.***polomer*** = 35;
vysledek = prvni.***getArea()***;

V prvním řádku je vytvořen objekt ***Kruh()*** s názvem "***prvni***"

Ve druhém řádku je definována vlastnost "***polomer***" jako ***35***. Ve třetím řádku je deklarována proměnná "***výsledek***" jako výstup metody ***getArea()***.

Proměnná "výsledek" bude rovna ***3848,45***

pozn.: Metodu je také možné definovat vně objektu. Potom je nutné použít tuto syntaxi:

```
Kruh.prototype.getArea = function() {
        area = Math.PI * radius * radius;
        return area;
}
```

"***prototype***" je ***konstrukční vlastnost*** každé vytvořené funkce. Všechny metody a vlastnosti adresované ***prototype*** přejdou po vytvoření objektu do vlastnosti "***_proto_***" Každá volaná vlastnost a metoda objektu je potom hledána v objektu ***_proto_***.[9]

2.10 Objekty - CORE

Objekty CORE (=jádro) jsou na nejvyšším místě v objektové hierarchii. Jedná se o imaginární objekty - nejsou hmotné (Array, Boolean, Date,...), většinou tedy potřebují konstruktor.

2.10.1 Array (pole)

Je to objekt obsahující větší množství separovaných hodnot, které jsou očíslovány (index).

Vytvoření pole:

```
pole = new Array(5)
```

Tab.4.Metody objektu Array

concat	spojuje pole dohromady a prvky seřadí za sebe: `pole1.concat(pole2, pole3, ...)`
join	vypíše obsah pole a vloží mezi položky separátor uvedený v závorce: `pole.join("+")`
pop	odstraní poslední prvek z pole a vypíše jeho hodnotu: `promenna = pole.pop()`
push	Přidá prvky na konec a vypíše novou délku pole:

	`promenna = pole.`***`push`***`("hrušky", "jablka")`
reverse	Převrátí pořadí prvků: `pole.`***`reverse()`***
shift	Odstraní první prvek a vypíše jeho hodnotu: `promenna = pole.`***`shift()`***
slice	Vyřízne část pole a udá jej jako nové pole: `pole.`***`slice`***`(2,5)` ... bude vyříznut 2. - 5. prvek
sort	Seřadí prvky pole buď podle velikosti, nebo pomocí funkce
sortOn	Seřazení podle názvu prvku
splice	odstraňuje/přidává prvky: `pole.`***`splice`***`(2,5,"nový1", "nový2",...)` bude odstraněno 5 prvků od prvku 2 a od tohoto místa budou vloženy prvky "nový1" a "nový2"
toString	vypíše hodnoty prvků oddělené čarkami
unshift	Vloží prvky na začátek a vypíše novou délku pole: `pole.`***`unshift`***`("první", "druhý", "třetí")`

Tab.5.Vlastnosti objektu Array

length	Udává délku pole (nikoliv počet prvků - počítá i vynechané indexy)

2.10.2 Date (datum)

Objekt obsahující informace o aktuálním čase a datumu

`datum = new` ***`Date()`***

Tab.6.Metody objektu Date

getDate	Udává den v měsíci v souladu s místním časem.
getDay	Udává den v měsíci v souladu s místním časem.
getFullYear	Udává čtyřciferný rok v souladu s místním časem.
getHours	Udává hodinu v souladu s místním časem.
getMilliseconds	Udává milisekundy v souladu s místním časem.
getMinutes	Udává minuty v souladu s místním časem.
getMonth	Udává měsíc v souladu s místním časem.
getSeconds	Udává sekundy v souladu s místním časem.
getTime	Udává počet milisekund od půlnoci 1.ledna 1970 univerzálního času.
getTimezoneOffset	Udává rozdíl v minutách mezi lokálním časem počítače a univerzálním časem.
getUTCDate	Udává den (datum) v měsíci v souladu s univerzálním časem.
getUTCDay	Udává den v týdnu v souladu s univerzálním časem.
getUTCFullYear	Udává čtyřciferný rok v souladu s univerzálním časem.
getUTCHours	Udává hodinu v souladu s univerzálním časem.
getUTCMilliseconds	Udává milisekundy v souladu s univerzálním časem.
getUTCMinutes	Udává minuty v souladu s univerzálním časem.
getUTCMonth	Udává měsíc v souladu s univerzálním časem.
getUTCSeconds	Udává sekundy v souladu s univerzálním časem.
getYear	Udává rok v souladu s místním časem.
setDate	Udává den v měsíci v souladu s místním časem.
setFullYear	Nastavuje celý rok v souladu s místním časem.
setHours	Nastavuje hodiny v souladu s místním časem.

setMilliseconds	Nastavuje milisekundy v souladu s místním časem.
setMinutes	Nastavuje minuty v souladu s místním časem.
setMonth	Nastavuje měsíc pro objekt Date v souladu s místním časem.
setSeconds	Nastavuje sekundy pro objekt Date v souladu s místním časem.
setTime	Nastavuje datum pro specifikovaný objekt Date v milisekundách.
setUTCDate	Nastavuje datum specifikovaného objektu Date v souladu s univerzálním časem.
setUTCFullYear	Nastavuje rok specifikovaného objektu Date v souladu s univerzálním časem.
setUTCHours	Nastavuje hodinu specifikovaného objektu Date v souladu s univerzálním časem.
setUTCMilliseconds	Nastavuje milisekundy specifikovaného objektu Date v souladu s univerzálním časem..
setUTCMinutes	Nastavuje minutu specifikovaného objektu Date v souladu s univerzálním časem.
setUTCMonth	Nastavuje měsíc reprezentovaný specifikovaným objektem Date v souladu s univerzálním časem.
setUTCSeconds	Nastavuje sekundy specifikovaného objektu Date v souladu s univerzálním časem.
setYear	Nastavuje rok pro specifikovaný objekt Date v souladu s místním časem.
toString	Udává řetězcovou hodnotu reprezentující datum a čas uložený ve specifikovaném objektu Date.
date UTC	Udává počet milisekund mezi půlnocí 1. ledna 1970 univerzálního času a určitým časem.

2.10.3 Math (matematický)

Slouží k vykonávání složitějších matematických operací (např. goniometrické funkce)

Bez konstruktoru.

Syntaxe:

```
Math.metoda(výraz);
```

Tab.7.Metody objektu math

abs	Vypočítá absolutní hodnotu.
acos	Vypočítá arc cosinus.
asin	Vypočítá arc sinus.
atan	Vypočítá arc tangens.
atan2	Vypočítá úhel z osy x do bodu.
ceil	Zaokrouhlí číslo nahoru na nejbližší celé číslo.
cos	Vypočítá cosinus.
exp	Vypočítá exponenciální hodnotu.
floor	Zaokrouhlí číslo dolů na nejbližší celé číslo.
log	Vypočítá přirozený logaritmus.

max	Udává větší ze dvou celých čísel.
min	Udává menší ze dvou celých čísel.
pow	Vypočítá x zvýšené na mocninu y.
random	Udává pseudo-náhodné číslo mezi 0.0 a 1.0.
round	Zaokrouhluje na nejbližší celé číslo.
sin	Vypočítá sinus.
sqrt	Vypočítá čtvercový kořen (odmocninu).
tan	Vypočítá tangens.

Tab.8.Konstanty objektu math

E	Eulerova konstanta a základ přirozeného logaritmu (přibližně 2,718).
LN2	Přirozený logaritmus dvou (přibližně 0,693).
LOG2E	Základ 2 logaritmu e (přibližně 1,442).
LN1	Přirozený logaritmus 10 (přibližně 2,302).
LOG10E	Základ 10 logaritmu e (přibližně 0,434).
PI	Poměr obvodu kruhu k jeho průměru (přibližně 3,14159).
SQRT1_2	Reciproční kořenu čtverce (odmocnina) 1/2 (přibližně 0,707).
SQRT2	Kořen čtverce (odmocnina) 2 (přibližně 1,414).

2.10.4 Number (číslo)

Slouží k manipulaci s čísly.

cislo = new ***Number(5)***

Tab.9.Metody objektu number

toString	převede číslo na řetězec
valueOf	udává původní hodnotu objektu

Tab.10.Konstanty objektu number

MAX_VALUE	maximální použitelné číslo - cca $1.79*10^{308}$
MIN_VALUE	minimální použitelné číslo - cca $5*10^{-324}$
NaN	hodnota Not-a-Number (nečíselná)
NEGATIVE_INFINITY	záporné nekonečno
POSITIVE_INFINITY	kladné nekonečno (5 / 0 = POSITIVE_INFINITY)

2.10.5 String (textový řetězec)

Slouží k manipulaci s textovými řetězci.

text = new ***String("ahoj")***

Tab.11.Metody objektu string

charAt	Udává znak na dané pozici (index)

	`text.charAt(index).`
charCodeAt	Udává hodnotu znaku na daném indexu jako 16-bitové celé číslo mezi 0 a 65535. `text.charCodeAt(index).`
concat	Kombinuje text dvou řetězců a udává nový řetězec. `text.concat(hodnota1,...hodnotaN)`
fromCharCode	převede ASCII zápis na znak. `zavinac = String.fromCharCode(64)` (zavinac bude "@")
indexOf	Hledá řetězec a udává index hodnoty specifikované v argumentech.
lastIndexOf	Udává poslední výskyt podřetězce uvnitř řetězce, který se objeví před počáteční pozicí specifikovanou v argumentu nebo udá 1, jestliže není nalezen.
slice	Vytahuje část řetězce a udává nový řetězec.

split	Rozděluje objekt String na pole řetězců (Array) podle separátoru. (podobně jako PHP funkce "Explode")
	`text = "abcxdef";`
	`pole = text.split("x");`
	`// pole bude (abc, def)`

substr	Vyřízne určitý počet znaků od indexu znaku [start] `text.substr(start, [délka])`
substring	Udává znaky mezi dvěma indexy specifikovanými v argumentech do řetězce. `text.substring(od, do)`
toLowerCase	Konvertuje řetězec na malá písmena a udává výsledek.
toUpperCase	Konvertuje řetězec na velká písmena a udává výsledek.
charAt	Udává znak na dané pozici (index) `text.charAt(index).`

2.11 Objekty - MOVIE

Nyní si něco řekneme o objektech, které ovlivňují fyzické prvky na scéně (Button, TextField, MovieClip). Typické pro tyto objekty je, že většinou nepotřebují konstruktor - jsou deklarovány vytvořením fyzického prvku na scéně a jeho ***pojmenováním***.[9]

2.11.1 Button (tlačítko)

Objekt ovlivňující instanci tlačítka na scéně.

Bez konstruktoru.

Tab.12.Metody objektu Button

getDepth	udá pozici v ose Z (hloubka neboli pořadí zobrazení)

Tab.13.Vlastnosti objektu button

enabled	Vrací nebo nastavuje TRUE/FALSE podle toho, jestli je (má být) tl. zablokováno nebo ne

tabEnabled	Povolit navigaci pomocí TAB
tabIndex	Pořadí výběru objektu při navigaci tlačítkem TAB
trackAsMenu	Umožňuje jinému tlačítku převzít událost "release"
useHandCursor	použít kurzor "ruka"

Události:

Tyto příkazy dovolují definovat funkci, která se má provést po určité události, ve snímku. Je tak možné instancím tlačítek a MovieClipů definovat akce už dopředu:[9]

```
tlacitko.onRelease = function () {
        gotoAndPlay(5);
};
```

Tento zápis je ekvivalentní k definici příkazů přímo v instanci tlačítka "tlacitko":

```
on (release) {
            gotoAndPlay(5);
}
```

V obou případech přejde animace po kliknutí na tlačítko na snímek č. 5.

Tab.14.Události tlačítka

onDragOut	Tažení ven
onDragOver	Tažení dovnitř
onPress	Kliknutí
onRelease	Puštění tlačítka myši
onReleaseOutside	Puštění tlačítka myši mimo instanci
onRollOut	Přejetí ven
onRollOver	Přejetí dovnitř

2.11.2 Capabilites (schopnosti)

Vrací vlastnosti přehrávače případně serveru.

Syntaxe:

System.capabilities.

Tab.15.Vlastnosti objektu capabitites

hasAccessibility	Přítomnost pomocných komunikačních zařízení
hasAudio	Přehrávání audia
hasAudioEncoder	Přítomnost dekodéru audia
hasMP3	Přítomnost dekodéru MP3
hasVideoEncoder	Přítomnost dekodéru videa
pixelAspectRatio	Poměr stran pixelů (standardně 1)
screenColor	Barevnost monitoru
screenDPI	Počet bodů/palec monitoru (DPI = dots per inch)
screenResolutionX	Rozlišení monitoru v ose X
screenResolutionY	Rozlišení monitoru v ose Y

2.11.3 Color (barva)

Umožňuje definovat barvu instance MC.

Potřebuje konstruktor!

barva = new Color(***nazevInstanceMC***);

Tab.16.Metody objektu color

getRGB	Vrací hexadecimální hodnotu barvy (0xRRGGBB)
getTransform	Vrací hodnotu barevné transformace nastavenou setTransform
setRGB	Nastaví hexadecimální hodnotu barvy
setTransform	Nastaví barevnou transformaci - viz níže

Příklad na `setTransform()`***:***
Tato metoda nastavuje hodnoty, které známe z panelu "properties"

```
1  barva = new Color(kruh);
2  transformace = new Object();
3  transformace = {ra:'50',rb:'244',ga:'40',gb:'112',ba:'12',bb:'90',aa:'40',ab:'70'};
4  barva.setTransform(transformace);
```

Popis:
1 Vytvoření objektu "barva" s cílem na MovieClip "kruh"
2 Vytvoření objektu "transformace" pomocí generického objektu "Object()"
3 Přiřazení hodnot
4 Aplikování hodnot v objektu "barva"

Tab.17.Parametry metody setTransform

ra	procento červené (-100 až 100).
rb	vyrovnání červené (-255 až 255).
ga	procento zelené (-100 až 100).
gb	vyrovnání zelené (-255 až 255).
ba	procento modré (-100 až 100).
bb	vyrovnání modré (-255 až 255).
aa	procento afla (-100 až 100).
ab	vyrovnání alfa (-255 až 255).

2.11.4 Key (klávesa)

Objekt upravující výstup z klávesnice.

Bez konstruktoru.

Tab.18.Metody objektu key

addListener	Přidat objekt, který bude možno ovládat
getAscii	Vrací ASCII kód poslední zmáčknuté klávesy
getCode	Vrací kód poslední zmáčknuté klávesy
isDown	Vrací TRUE/FALSE podle toho, jestli je klávesa zmáčknuta
isToggled	Vrací TRUE/FALSE pokud je klávesa aktivována (platí pro

	NumLock a CapsLock)
removeListener	Odebrat objekt přidaný příkazem addListener

Konstanty:

Syntaxe:

Key.***konstanta***

Tab.19.Konstanty objektu key

BACKSPACE	
CAPSLOCK	
CONTROL	
DELETEKEY	
DOWN	šipka dolů
END	
ENTER	
ESCAPE	
HOME	
INSERT	
LEFT	šipka doleva
PGDN	
PGUP	
RIGHT	šipka doprava
SHIFT	
SPACE	mezerník
TAB	
UP	šipka nahoru

Tab.20.*Metody objektů asociovaných pomocí* **`addListener()`**

onKeyDown	Stisk tlačítka
onKeyUp	Uvolnění tlačítka

Ukázka:

Použití v podmínce:

```
onClipEvent (enterFrame){
    if(Key.isDown(Key.RIGHT)){
            _x += 10;
    }
}
```

Pokud zde stiskneme šipku vpravo (Key.isDown bude TRUE), posune se každý snímek MovieClip (this) o 10px vpravo.

2.11.5 Mouse (myš)

Objekt upravující výstup z myši

Bez konstruktoru.

Tab.21.Metody objektu mouse

addListener*	Přidat objekt, který bude možno ovládat
hide	Schovat kurzor
removeListener	Odebrat objekt přidaný příkazem addListener
show	Zobrazit kurzor

Tab.22.*Metody objektů asociovaných pomocí* **`addListener()`**

onMouseDown	Kliknutí
onMouseMove	Pohyb
onMouseUp	Uvolnění

* Způsob použití je obdobný jako u objektu Key

2.11.6 MovieClip

Objekt ovlivňující instanci MovieClipu na scéně.

Bez konstruktoru.

Tab.23.Metody objektu MovieCip

attachMovie	Připojuje MC z knihovny `(IDjméno, jméno, hloubka)`
createEmptyMovieClip	Vytvořit prázdný MC `(jméno, hloubka)`
createTextField	Vytvořit uvnitř MC textové pole. `(jméno, hloubka, x, y, šířka, výška)` Jeho vlastnosti umožňuje nastavit objekt TextFormat (viz níže)
duplicateMovieClip	Duplikovat MC `(novéJméno, hloubka, objekt*)` * Pokud definujete objekt, budou jeho vlastnosti zkopírovány do nového MC
getBounds	Vrací souřadnice rámečku, ohraničujícího MC v definovaném souřadnicovém prostoru např `(._root)`
getBytesLoaded	Vrací hodnotu načtených bytů animace do MC
getBytesTotal	Vrací hodnotu celkové velikosti animace do MC
getDepth	Vrací hloubku instance MC
getURL	Přejde na URL adresu a umožňuje odesílat proměnné `(URL, okno, proměnné)`
globalToLocal	Konvertuje globální souřadnice na lokální souř. MC
gotoAndPlay	Přejde na snímek MC a spustí přehrávání
gotoAndStop	Přejde na snímek MC
hitTest	Metoda kontrolující kolize 2 MC nebo MC a bodu na

	scéně. Vrací TRUE/FALSE `(cílovýMC)` nebo `(x, y, true/false*)` * true = uvažovat skutečný tvar instance (nikoliv celý rámeček)
loadMovie	Načte SWF animaci do přehrávače `(URL, proměnné)`
loadVariables	Načte a zároveň odešle proměnné z (do) souboru `(URL, proměnné)`
localToGlobal	Konvertuje lokální souřadnice MC na souř. scény
nextFrame	Posune MC o jeden snímek dopředu
play	Spustí přehrávání MC
prevFrame	Vrátí MC o jeden snímek zpět
removeMovieClip	Odstraní instanci MC na scéně
setMask	Použije cílový MC jako masku `(maskovacíMC)`
startDrag	Započne tažení MC myší `(zamknoutNaStřed, vlevo, vpravo, nahoře, dole*)` * ohraničující prostor, kde je možný pohyb
stop	Zastaví přehrávání MC
stopDrag	Ukončí tažení MC
swapDepths	Výměna hloubky s cílovým MC `(cílovýMC)`
unloadMovie	Odstraní načtenou SWF animaci z přehrávače

Tab.24.Vlastnosti objektu MovieClip

enabled	viz objekt Button
focusEnabled	Povolit výběr objektu modou setFocus
hitArea	Definuje plochu jiného MC jako citlivou na kliknutí (událost onPress - viz níže)
tabChildren	Povolit nebo zakázat TAB navigaci mezi vnořenými instancemi uvnitř MC (TRUE/FALSE) - implicitně povoleno
tabEnabled	Povolit navigaci pomocí TAB - (TRUE/FALSE) - implicitně povoleno
tabIndex	viz dříve
tracAsMenu	viz Button
useHandCursor	viz Button

Události:

Podobně jako u objektu Button, i zde je možné definovat události vně instance už dopředu. Takto je možné MC upravit tak aby se choval jako tlačítko.

Použití je stejné jako u objektu button:

```
movieclip.onMouseDown = function () {
      gotoAndPlay(5);
};
```

Tab.25.Události objektu MovieClip

onData	obdržení údajů z LoadVariables nebo LoadMovie

onDragOut	tažení ven
onDragOver	tažení dovnitř
onEnterFrame	akce jsou vykonány v každém snímku MC
onKeyDown	stisk klávesy
onKeyUp	uvolnění klávesy
onKillFocus	odznačení výběru instance (např. navigací TAB)
onLoad	načtení nebo vygenerování MC
onMouseDown	stisk levého tl. myši (kdekoliv v animaci)
onMouseMove	pohyb myši (kdekoliv v animaci)
onMouseUp	uvolnění levého tl. myši (kdekoliv v animaci)
onPress	stisk levého tl. myši ***nad instancí***
onRelease	uvolnění levého tl. myši ***nad instancí***
onReleaseOutside	uvolnění levého tl. myši ***mimo instanci***
onRollOut	přejetí kurzorem dovnitř
onRollOver	přejetí kurzorem ven
onSetFocus	označení instance (např. navigací TAB)
onUnload	odstranění MC z časové osy

2.11.7 Sound (zvuk)

Souží k manipulaci se zvukem

Potřebuje konstruktor!

zvuk = new Sound()

Tab.26.Metody objektu sound

attachSound	Připojit zvuk z knihovny `("IDjméno")`
getBytesLoaded	Počet načtených bytů zvuku
getBytesTotal	Celková velikost zvuku
getPan	Vrací vyvážení zvuku (L/P reproduktor: -100/+100)
getTransform	Vrací informace o transformaci zvuku
getVolume	Vrací hlasitost
loadSound	Načte externí MP3 do objektu
setPan	Nastaví vyvážení
setTransform	Nastaví transformaci *
setVolume	Nastaví hlasitost
start	Spustí zvuk
stop	Zastaví zvuk

Tab.27.Vlastnosti objektu sound

duration	Délka zvuku v milisekundách - `jen pro čtení`
position	Pozice přehrávání (ms) - `jen pro čtení`

Tab.28.Udalosti objektu sound

onLoad	Po načtení zvuku zavolá definovanou funkci
onSoundComplete	Po skončení přehrávání zavolá definovanou funkci

2.11.8 Stage (nastavení scény)

Tento objekt umí nastavit vlastnosti scény v přehrávači podobně jako příkaz FSCommand.

Bez konstruktoru.

Tab.29.Metody objektu stage

addListener	Připojení vlastního objektu, který bude reagovat na událost onResize
removeListener	Odpojení předchozího objektu

Tab.30.Vlastnosti objektu stage

align	Zarovnání animace v přehrávači: `("vert_horiz")` `"T"` - nahoře, `"B"` - dole, `nic` - uprostřed `"L"` - vlevo, `"R"` - vpravo, `nic` - uprostřed např.: `"BL"` - vlevo dole, `"R"` - vpravo uprostřed
height	Výška okna v pixelech
scaleMode	Volba zobrazení (podobně jako u FScommand) `"exactFit", "showAll", "noBorder", "noScale"` (implicitní je `"showAll"`)
showMenu	Zobrazit menu `(true/false)`
width	Šířka okna v pixelech

Tab.31.Události objektu stage

onResize	Po změně velikosti animace (např. vlivem změny velikosti okna prohlížeče) zavolat funkci

Příklad:

```
pokus = new Object();
pokus.onResize = function () {
        _root.Play()
}
Stage.addListener(pokus);
```

Nejprve je vytvořen generický objekt ***Object***, kterému je definována metoda ***onResize***. Nakonec je objekt připojen objektu ***Stage***. Pokud tedy dojde ke změně proporcí okna přehrávače je zavolána funkce a vykonán příkaz ***Play().***[9]

2.11.9 TextField (textové pole)

Slouží k manipulaci s textovými poli typu dynamic a input text

Bez konstruktoru.

Tab.32.Metody objektu textField

addListener	Připojit vlastní objekt, který bude reagovat na události (`onChanged`, atd...)
getDepth	Vrací hloubku pole
getFontList	Vrací datové pole Array s názvy všech nainstalovaných fontů v operačním systému + názvy připojených fontů
getNewTextFormat	Vrací formátování nastavené pomocí `setNewTextFormat`
getTextFormat	Vrací formátování nastavené pomocí `setTextFormat`
removeListener	Odpojit objekt připojený metodou `addListener`
removeTextField	Odstraní textového pole MovieClipu vytvořené metodou `createTextField` (objekt MovieClip - viz výše)
replaceSel	Odstraní vybranou část textu a nahradí ji novým textem
setNewTextFormat	Nastaví formátování pro nově vkládaný text
setTextFormat	Nastaví formátování (viz objekt `TextFormat`)

Tab.33.Vlastnosti objektu textField

autoSize	Nastavuje zarovnání textu a povoluje automatickou změnu velikosti (podle velikosti textu)
background	Určuje, zda se má zobrazovat pozadí (TRUE/FALSE)
backgroundColor	Nastavuje barvu pozadí (implicitně bílá - 0xFFFFFF)
border	Určuje, zda se mají zobrazovat okraje (TRUE/FALSE)
borderColor	Nastavuje barvu okrajů (implicitně černá - 0x000000)
bottomScroll	Vrací číslo dolního viditelného řádku (u Multiline TextField)
embedFonts	Vrací booleovskou hodnotu, zda má pole přibalené fonty
hscroll	Nastavuje horizontální scrolling
html	Vrací booleovskou hodnotu, zda pole zobrazuje HTML tagy
htmlText	Zobrazí v poli text a zformátuje podle HTML tagů
length	Vrací počet znaků
maxChars	Nastavuje maximální počet zobrazovaných znaků
maxhscroll	Vrací maximální hodnotu ***`hscroll`***
maxscroll	Vrací maximální hodnotu ***`scroll`***
multiline	Vrací booleovskou hodnotu, zda je pole typu Multiline
password	Vrací booleovskou hodnotu, zda je pole typu Password
restrict	Nastavuje povolené znaky, které může uživatel napsat do pole: `"A-Z 0-9"` povoleny jsou znaky A-Z a číslice `"A-Z^Q"` povoleny jsou znaky A-Z ***kromě*** znaku Q
scroll	Nastavuje vertikální scrolling
selectable	Vrací booleovskou hodnotu, zda půjde text v poli vybrat
tabEnabled	Povolit navigaci pomocí TAB
tabIndex	Pořadí výběru objektu při navigaci tlačítkem TAB
text	Zobrazí v poli text

textColor	Nastavuje barvu textu v poli
textHeight	Vrací výšku bloku textu [px]
textWidth	Vrací šířku bloku textu [px]
type	Nastavuje typ pole - `("dynamic")` nebo `("input")`
variable	Nastavuje proměnnou, která se bude zobrazovat v poli

Tab.34.Události objektu textField

onChanged	Po editaci pole zavolá funkci
onKillFocus	Ukončení editace
onScroller	Po scrollování zavolá funkci
onSetFocus	Začátek editace

2.11.10 TextFormat (formát textu)

Objekt definující hodnoty metod ***setTextFormat*** a ***setNewTextFormat*** objektu ***TextField***.

Potřebuje konstruktor!

```
nadpis = new TextFormat();
nadpis.vlastnost = hodnota;
kolonka3.setTextFormat(nadpis);
```

Tab.35.Metody objketu textFormat

getTextExtend("text")	Vrací šířku a výšku řetězce v závislosti na nastaveném formátování v pixelech
	`getTextExtend("text").width`
	`getTextExtend("text").height`

Tab.36.Vlastnosti objektu textFormat

align	zarovnání (left, right, center)
blockIndent	Odsazení textu
bold	Tučné (TRUE/FALSE)
bullet	Zobrazit každý odstavec jako položku seznamu
color	Barva textu
font	Použitý font
indent	Odsazení prvního řádku odstavce
italic	Kurzíva
leading	Velikost řádkování
leftMargin	Levý okraj
rightMargin	Pravý okraj
target	Cílové okno odkazu
size	Velikost textu
underline	Podtržený text
url	Vytvořit odkaz

2.12 Offline komunikace mezi HTML(XHTML) a Flashem

Ukážeme si jak do flash poslat nějaké proměnné s použitím HTML

Teď si popíšeme validní a použitelný zápis flashového objektu do XHTML.

```
<object type="application/x-shockwave-flash"
data="menu.swf?pro1=1&pro2=0" width="200" height="416">
<param name="movie" value="menu.swf?pro1=1&pro2=0" />
<param name="menu" value="false" />
<param name="quality" value="best" />
<param name="bgcolor" value="#FFFFFF" />
<param name="loop" value="auto" />
<img src="banner.gif" width="200" height="416" alt="banner" />
</object>
```

V tomto kódu jsou za adresou swf souboru napsány proměnné(pro1=1,pro2=0), které se pošlou do flash.

Adresa: „**menu.swf?jazyk=cz&stranka=kontakt**“

S těmito proměnnými může flash bez problému pracovat a upravit si tak prostředí ofline bez použití serverových skriptu.Např. změna jazyka.

Tab.37.Popis značek

type	mime typ dokumentu, souboru, který pod objektem načítáváme
data	adresa k flash souboru
width a height	šířka a výška objektu

Samotná značka(tag) <param> označuje jednotlivé parametry, které chceme odevzdat flashi nebo jen způsob jak jej zobrazit.

Tab.38.Popis parametrů html kódu

movie	adresa k flash souborumenu: hodnoty[false/true], povoluje nebo zakazuje zobrazit menu po kliknutí pravým tlačítkem myši
quality	hodnoty[best/high/medium/low/auto], nastavuje zobrazení flashe v kvalitě. Používá se hlavně při flashech, které mají složité animace, složité zobrazení některých animací
bgcolor	hodnota[web color-#000000-černá], nastaví barvu pozadí flashe. Vidíme ji, například když ještě flash není načtený.
Loop	hodnota[false/true/auto], jestli se animace celého flashe má opakovat

2.13 Komunikace Flash s PHP

Flash samotný z bezpečnostních důvodu **nemůže** zapisovat do souborů (podobně jako JavaScript). **Zápis** dat tedy budeme muset svěřit nějakému serverovému skriptu. Použíjeme PHP, ale použít můžete i ASP a další. Proměnné můžeme PHP poslat pomocí funkce **sendAndLoad**.

Příklad:

```
1  var data:LoadVars = new LoadVars();
2  data.promenna=8;
3  data.sendAndLoad("http://www.napohodku.com/staniik/nazvy.php", data,
4  "POST");
5  data.onLoad = function(){
6                    info=data.vystup;
                 }
```

Popis:

1 Vytvoření objektu LoadVars názvem „data“
2 Proměnná kterou budeme chtít poslat
3 Poslání obsahu dat objektu „data“ php skriptu na určené adrese
4 Čekání na odpověď od php skriptu
Až se data načtou, provede se skript mezi složenýma závorkama.

S poslanými proměnnými můžete v php skriptu pracovat tak, jako by tam byly přímo deklarované. Odpověď pro flash se píše pomocí příkazu„echo“.

Například takto:

echo "vystup=".$obsah;

Flash po načtení vloží proměnnou „vystup“ do objektu „data“.

2.14 Preloader

Preloader animaci zastaví na začátku a pustí ji dál až je celá načtená. Funkci si vysvětlíme na příkladu který ukazuje načtená procenta. Jako první vytvoříme dynamický text do kterého budeme vkládat načtená procenta a pojmenujeme ho “procenta“. Na začátek naší animace vyčleníme 2 snímky a do nich vložíme instanci MovieClipu a nazveme ji například "prubeh"

Následně vytvoříme novou vrstvu a do druhého snímku vložíme tyto akce:

```
1  loading = Math.round(getBytesLoaded()/getBytesTotal()*100);
2  _root.procenta.text=loading;
3  if (loading == 100) {
4      play();
5  } else {
6      gotoAndPlay(1);
7  }
```

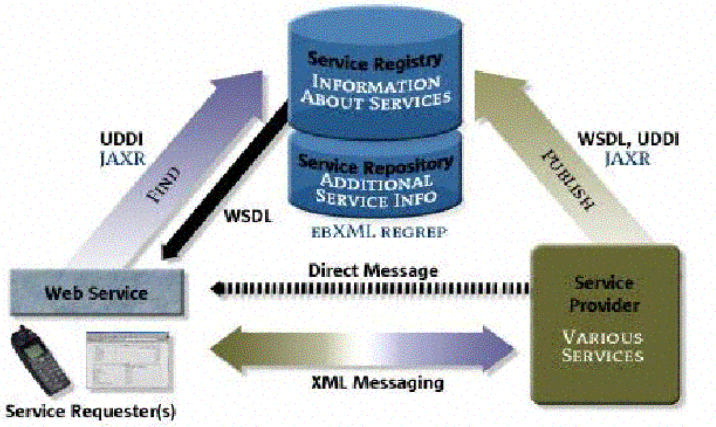
Popis:

1 Definování proměnné **loading** jako zaokrouhlení výrazu **načtená data/celková*100**
2 Cesta k textovému poli
3 Podmínka.Pokud dosáhnou načená data 100, pak přehrávání pokračuje.
Pokud ne, tak se vrátí na první snímek

3 WEBOVÉ SLUŽBY

3.1 Co jsou webové služby

Jedná se o systém navržený pro podporu přenosu informací mezi počítači různých platforem. Jde tedy o obdobu technologií určených pro vzdálené volání funkcí v distribuovaných systémech jako CORBA či RMI. Na rozdíl od nich, jsou však webové služby navržené pro komunikaci mezi různými platformami (operačními systémy, programovacími jazyky, architekturami počítačů...).[11]



Obr.8. Schéma použití webové služby

3.2 Webové služby - slovníček základních pojmů

HTTP – HyperText Transfer Protocol

HTTP je internetový protokol sloužící k výměně dat mezi internetovým serverem a klientem. Tento systém komunikace se používá například při prohlížení internetových stránek. Klient sestaví požadavek, který odešle na server. Součástí tohoto požadavku je název požadované stránky a parametry. Jakmile server požadavek obdrží, kontaktuje stránku, předá jí parametry a čeká na odpověď. Po získání odpovědi ze stránky ji obratem odesílá zpět klientovi. Obsah odpovědi může být jakéhokoliv typu. Nejčastěji je to kód webové stránky, který prohlížeč následně zpracuje a zobrazí. Odpovědí ale mohou být i XML data, která jsou klíčová právě pro webové služby.[11]

XML – eXtensible Markup Language

XML je značkovací jazyk, právě tak jako HTML. Na rozdíl od HTML nejsou názvy značek přesně definované. Názvem může být jakékoliv klíčové slovo, které většinou

slouží k popisu dat, která jsou v těle značky uzavřena. Příkladem může být název proměnné a její hodnota:

```
<jmeno>Josef</jmeno>
```

Díky této jednoduché syntaxi se jazyk XML stal standardním formátem pro výměnu dat mezi jinak vzájemně nekompatibilními systémy.[11]

SOAP – Simple Object Access Protocol

SOAP je předpis, kterým se řídí vlastní komunikace mezi klientem a serverem. Komunikace probíhá přes HTTP protokol a je uskutečňována výměnou XML dat ve speciálním tvaru. O celou tuto záležitost se stará SOAP zprostředkovatel, například Microsoft SOAP Toolkit. Toolkit tvoří dvě základní části, SOAP Server a SOAP Client. Každá tato část se na své straně sítě stará o zpracovávání SOAP dat.[11]

Webová služba – Web Service

Webová služba je seskupení všech částí, které na serverové straně dohromady poskytují klientům přístup k požadovaným datům. Mezi tyto části patří SOAP Server, COM komponenta (popřípadě více komponent) a dva XML soubory s příponami WSDL a WSML. Tyto soubory popisují stavbu COM komponenty a díky nim se ve výsledku webová služba navenek tváří jako objekt, jehož metody může klient volat.[11]

WSDL – Web Services Description Language

Soubor s příponou WSDL je jedním z XML souborů popisujících webovou službu. Obsahuje jméno služby, seznam a popis jejích metod. Do popisu metod patří jména a datové typy všech parametrů spolu s informací o datovém typu návratové hodnoty. Soubor nakonec obsahuje URL adresu příjemce (listenera), který s pomocí SOAP Serveru celou komunikaci na straně serveru zprostředkovává.[11]

WSML – Web Services Meta Language

WSML soubor je druhý XML soubor, který obsahuje informace o COM komponentě. Tento soubor využívá SOAP Server při vlastním volání COM komponenty.[11]

3.3 Jak fungují webové služby

Webové služby reprezentují nástroj pro inovaci a zefektivnění obchodních procesů v rámci mezipodnikové organizace, nejsou novou technologií. Technologicky jsou

webové služby modulární obchodní aplikací se specifikovanými rozhraními, přístupnými a provozovanými v internetovém prostředí.[12]

Jednoduše řečeno - jedná o technologii, která umožňuje integrovat libovolné aplikace provozované na různých platformách a ovládat je prostřednictvím webového rozhraní, tj. z běžného internetového prohlížeče.[12]

Potenciál webových služeb je obrovský. Jejich využití může znamenat velký přínos zejména pro firmy. Dosud totiž bylo velice obtížné integrovat jednotlivé aplikace, které jsou často naprogramovány v mnoha různých programovacích jazycích a navíc jsou provozovány na nejrůznějších platformách. Jejich vzájemné propojení proto nefunguje vždy zcela spolehlivě. Teprve webové služby, resp. otevřený standard XML (eXtensible Markup Language), který webové služby využívají, komunikaci mezi aplikací a klientem značně usnadňuje.[12]

Webové služby prostě naplno využívají toho, že programovací jazyk (je-li možné jej takto nazvat) HTML byl od samého počátku vývoje nezávislý na používané platformě. Proto byl vybrán jako nejvhodnější základ pro univerzální řešení, které by umožnilo naplnit filozofii webových služeb, tj. propojit běžně nepropojitelné aplikace. Tím je již zmíněné XML.[12]

3.3.1 Informace dostupné kdekoliv

Snahou stratégů webových služeb je dát uživatelům možnost přistupovat k potřebným informacím, a to kdykoliv a kdekoliv - a navíc z libovolného zařízení: nejen z běžného osobního počítače či notebooku, ale také z kapesního počítače a nebo mobilního telefonu. A protože mezi jednotlivými zařízeními jsou nemalé rozdíly, musí webové služby fungovat jako dokonale univerzální rozhraní mezi zdrojem dat a uživatelem.[12]

Vzhledem k tomu, že v dnešní hektické době jsou právě informace jedním z faktorů, které často a tvrdě ovlivňují osudy firem, mohou webové služby vychylovat pomyslnou misku vah na stranu jejich uživatelů - potenciálních vítězů.[12]

3.3.2 Poskytovatel služeb

Service Provider, tj. subjekt, který službu poskytuje. Jde o softwarovou či hardwarovou platformu, která zajišťuje provoz vlastních webových služeb.[12]

3.3.3 Registr služby

Service Registry, tj. místo, kde jsou uloženy informace o webových službách a jejich poskytovatelích. Místo, na kterém uživatelé mohou vyhledávat poskytovatele a které současně poskytuje informace potřebné pro navázání komunikace mezi uživatelem a poskytovatelem služeb.[12]

3.3.4 Klient

Service Requestor, je z obchodního hlediska ten, kdo vyhledává požadovanou funkci. Z hlediska architektury se jedná o aplikaci, která funkci volá, tedy zpravidla aplikace běžící ve webovém prohlížeči ovládaná uživatelem, popřípadě aplikace, která nemá vlastní rozhraní, které nahrazuje právě prostřednictvím webových služeb.Pro vzájemnou spolupráci jednotlivých částí je důležité, aby byly definovány operace, které zajistí jejich vzájemnou interakci. Jsou to publikování, vyhledávání a propojení.[12]

3.3.5 Publikování

Interakce mezi poskytovatelem služeb a jejich registrem. Bez této operace není fakticky možné webovou službu zpřístupnit klientům a tedy používat ji v praxi.[12]

3.3.6 Vyhledávání

Komunikace klienta s registrem služeb dává uživateli potřebné informace o poskytovatelích webových služeb, resp. samotných webových službách a jejich možném využití.[12]

3.3.7 Propojení

Poslední - a také nejdůležitější - operací je interakce mezi poskytovatelem a uživatelem. Díky ní je možné webovou službu využívat.[12]

3.5 Využití webových služeb a protokolu SOAP při komunikaci

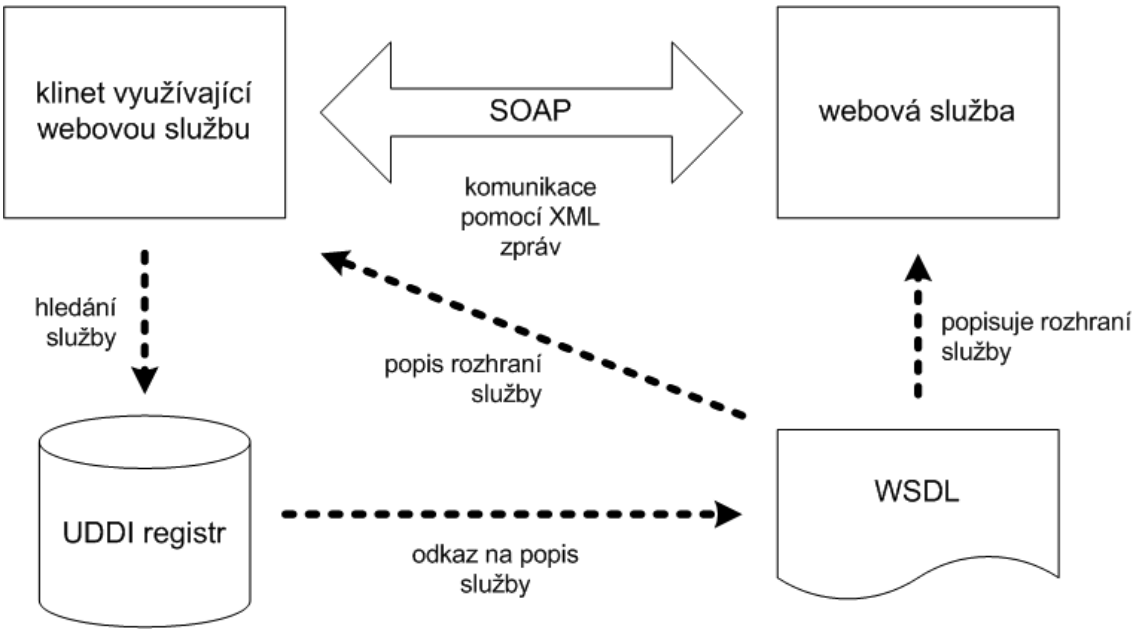
Jak jsme již naznačili, webové služby umožňují jednoduchou komunikaci mezi aplikacemi ve velmi heterogenním prostředí, protože komunikace je založena na platformě nezávislých standardech – především na jazyce XML a protokolu HTTP. Aplikace si mezi sebou posílají XML zprávy, které přenášejí dotazy a odpovědi jednotlivých aplikací. Celá infrastruktura webových služeb je založena na třech základních technologiích:[13]

- SOAP (Simple Object Access Protocol) – protokol používaný pro komunikaci;

- WSDL (Web Services Description Language) – standardní formát pro popis rozhraní webové služby;
- UDDI (Universal Description, Discovery and Integration) – standardní mechanismus umožňující registraci a vyhledávání webových služeb.

Vzájemné vztahy mezi těmito třemi technologiemi jsou zachycené na obrázku 9. Ke každé webové službě by měl být k dispozici její formální popis v jazyce WSDL. Z tohoto popisu již jde automaticky vygenerovat soapový požadavek. Ve větších systémech nebo přímo v otevřeném prostředí Internetu se popis služby může zaregistrovat do UDDI registru. Ten slouží jako jakýsi telefonní seznam („zlaté stránky“), který umožňuje vyhledávání služeb s určitými parametry.[13]

Klient, který chce využít webovou službu, získá buď přes UDDI, nebo přímo její popis. Z něj je jasné, jakou strukturu má mít soapová zpráva a kam se má webové službě poslat, aby ji rozpoznala.[13]



Obr.9. Vztah tří základních technologií (SOAP, WSDL a UDDI)

3.5.1 SOAP

SOAP je protokol pro posílání zpráv XML a je základem webových služeb. Ostatní standardy jako WSDL a UDDI vznikly až později po uvedení SOAPu a jen dále rozšiřují jeho možnosti a snadnost použití. SOAP umožňuje zaslání XML zprávy mezi dvěma aplikacemi a pracuje tedy na principu peer-to-peer. Zpráva je jednosměrný

přenos informace od odesílatele k příjemci, ale díky kombinování několika zpráv můžeme pomocí SOAPu snadno implementovat běžné komunikační scénáře.

Nejčastěji se SOAP používá jako náhrada vzdáleného volání procedur (RPC), tedy v modelu požadavek/odpověď. Jedna aplikace pošle v XML zprávě požadavek druhé aplikaci, tak požadavek obslouží a výsledek zašle jako druhou zprávu zpět původnímu iniciátorovi komunikace. V tomto případě bývá webová služba vyvolána webovým serverem, který čeká na požadavky klientů a v okamžiku, kdy přes HTTP přijde soapová zpráva, spustí webovou službu a předá jí požadavek. Výsledek služby je pak předán zpět klientovi jako odpověď.

První verze (1.0) protokolu SOAP vznikla na konci roku 1999 jako výsledek společné práce firem DevelopMentor, Microsoft a UserLand, které chtěly vytvořit protokol pro vzdálené volání procedur (RPC) založený na XML. Protokol navazoval na o rok mladší, jednodušší a méně flexibilní protokol XML-RPC. V průběhu roku 2000 se k podpoře přihlásila i firma IBM a nová verze SOAPu 1.1 byla zaslána W3C konsorciu. Verze SOAPu 1.1 je dnes nejpoužívanější a v diplomové práci se budeme zabývat právě jí. Na půdě W3C konsorcia nyní probíhá práce na uvolnění prvního skutečného standardu SOAP 1.2 v rámci pracovní skupiny pro XML protokol.[13]

3.5.2 Struktura zprávy

Zpráva v SOAPu je jednoduchý XML dokument, který má kořenový element `Envelope`. V této obálce jsou pak uzavřeny dva elementy `Header` (hlavička) a `Body` (tělo). Hlavička je přitom nepovinná a používá se pro přenos pomocných informací pro zpracování zprávy – například identifikaci uživatele, autentizační informace (jméno, heslo) apod.[13]

O to nejdůležitější se stará tělo zprávy, v němž se přenášejí informace identifikující volanou službu a předávané parametry, resp. návratové hodnoty služby. SOAP používá jmenné prostory pro identifikování jednotlivých částí XML zprávy. Obálka, hlavičky a tělo zprávy patří do jmenného prostoru [13]

Ukázka jednoduché zprávy SOAP

```
<SOAP-ENV:Envelope
 xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
 SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
   <SOAP-ENV:Body>
    <m:GetLastTradePrice xmlns:m="urn:x-example:services:StockQuote">
      <symbol>MOT</symbol>
```

```
    </m:GetLastTradePrice>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Tento příklad ukazuje velmi jednoduchý SOAP požadavek na zjištění posledního známého kurzu akcie s kódem MOT. Ukázková zpráva pro jednoduchost neobsahuje hlavičku, ale pouze tělo. V něm je požadavek na vyvolání vzdálené funkce GetLastTradePrice s parametrem pojmenovaným symbol s hodnotou MOT (kód akcií firmy Motorola).

Jak by mohla vypadat XML zpráva s výsledkem přibližuje příklad níže.[13]

Ukázka zprávy s odpovědí

```
<SOAP-ENV:Envelope
 xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
 SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
 <SOAP-ENV:Body>
  <m:GetLastTradePriceResponse
   xmlns:m="urn:x-example:services:StockQuote">
   <Price>14.5</Price>
  </m:GetLastTradePriceResponse>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

3.6 Kódování dat

V ukázkách SOAP zpráv je použit ještě jeden jmenný prostor. Ten identifikuje způsob kódování přenášených dat do XML. Se SOAPem můžeme použít libovolný způsob serializace, standard SOAPu jeden rovnou definuje. Standardní serializace je schopná do XML převést graf obsahující otypované objekty. V praxi se nejčastěji používají běžné skalární datové typy (jako čísla, řetězce apod.), které definují XML schémata. Navíc soapové kódování definuje způsob serializace složených datových typů – seznamů skalárních hodnot (polí) a struktur. Lze serializovat i reference na objekty.[13]

Obecně platí, že se hodnoty ukládají vždy jako obsah elementů, jedna hodnota do jednoho elementu. Datové typy mohou být definovány buď v externím XML schématu nebo přímo v XML zprávě. Druhý způsob je jednodušší a tedy i používanější. Např. mé osobní údaje by mohly být zakódovány do XML pro potřeby SOAPu následujícím způsobem:[13]

```
<jméno xsi:type="xsd:string">Jan Novak</jméno>
<email xsi:type="xsd:string">Jan@Novak.cz</email>
<věk xsi:type="xsd:int">24</věk>
```

Předpokládáme přitom, že prefixy `xsi` a `xsd` jsou svázány se jmenným prostorem `http://www.w3.org/2001/XMLSchema-instance`, resp. `http://www.w3.org/2001/XMLSchema`. Odpovídající samostatné XML schéma by pak vypadalo následovně:

```
<xsd:element name="jméno" type="xsd:string"/>
<xsl:element name="email" type="xsd:string"/>
<xsl:element name="věk" type="xsd:int"/>
```

3.7 UDDI

UDDI nabízí mechanismy pro registrování, kategorizování a vyhledávání webových služeb. UDDI funguje jako velký adresář, který obsahuje informace o subjektech (firmách) a jimi poskytovaných službách. Samotný registr pracuje rovněž jako webová služba a komunikace s ní tedy opět probíhá pomocí SOAPu.[13]

UDDI registr obsahuje následující čtyři druhy entit:

podnikatelské entity (firmy) – business entity

U každé firmy v registru jsou zaznamenány základní údaje jako název, stručný popis a kontaktní údaje. Každé firmě mohou být přiřazeny klasifikační identifikátory, které určují oblasti jejího podnikání a geografickou polohu.[13]

služby – business service

Ke každé firmě jsou v registru uloženy seznamy služeb, které firma poskytuje. Každá služba je opět popsána a obsahuje seznam šablon vazeb, které ukazují na technické údaje nutné pro využití služby.[13]

šablony vazeb – binding template

Šablony popisují, jak a kde je možné se službou komunikovat. Typicky je tato informace popsána odkazem na WSDL soubor s definicí rozhraní služby. Každá šablona kromě toho odkazuje na typ služby, který implementuje.[13]

typy služeb – service typ

Typ služby definuje abstraktní službu. Funguje tedy jako obdoba rozhraní, jak je známe např. z Javy. Několik firem může nabízet stejný druh služby se stejným rozhraním a tedy i typem služby. Typ služby je popsán tzv. technickým modelem (tModel).[13]

Typická práce s UDDI probíhá tak, že vývojář prohledá registr a najde si služby, které potřebuje. Získá pro ně popis WSDL a může je začít rovnou používat. Dodejme ještě, že UDDI nemusí obsahovat jen popisy webových služeb ve WSDL, lze do něj ukládat

popisy služeb v libovolném formátu. Z důvodu interoperability se však společně s UDDI používá právě SOAP a WSDL.[13]

II. PRAKTICKÁ ČÁST

Příklad č.1

Základní animace

Vytvořte objekt, který se po načtení začne zvědšovat a zastaví se v určité poloze.

poz: Animace se nesmí znovu opakovat.

Postup:

V prvním kroku vytvoříme objekt (např. kruh) po té vytvoříme v 15-tém snímku klíčový snímek (F5) a v tomto snímku objekt zvědším pomocí nástroje **Free Transform** v panelu **Tools**. Aby nám mezi dvěma klíčovími snímky proběhl plynulí přechod musíme oblast mezi snímkem 1 a 15 označit a v panelu **Properities** nastavit **Tween** na **Shape**.

Jako poslední krok vytvoříme novou vrstvu. A aby se nám animace neopakovala stále dokola tak do 15-tého snímku nové vrstvy vložíme skript:

```
Stop();
```

Příklad č.2

Reakce na událost

Vytvořte kruh, který bude reagovat na dvě události. Pokud na něm bude kursor myši, tak se zvětší do určené velikosti. Pokud s myší odjedeme mimo, tak se postupně zmenší do původní polohy.

Postup:

Vytvoříme MovieClip(ctrl+F8) a vložíme do něj animaci zvětšení kruhu(použijeme postup z příkladu č.1).Navíc, ale do prvního snímku animace vložíme skript „stop();“, aby se nám kruh nezačal zvětšovat hned po spuštění.

Na konec vložíme do MovieClipu tento skript:

```
1  on(rollOver){
2                  play();
3                  ven=0;
4                  }
5  on(rollOut){
6              ven=1;
7              i=this._currentframe;
8              }
```

```
 9  onClipEvent(enterFrame){
10                                  if(ven==1){
11                                          gotoAndStop(i);
12                                          i--;
13                                          }
14                                   }
```

Popis:

1-4	Zpuštění animace v MovieClipu po najetí myší
7	Do proměnné i se vloží aktuální pozice přehrávání MovieClipu
10-13	Po splnění podmínky se začne animace přehrávat zpět

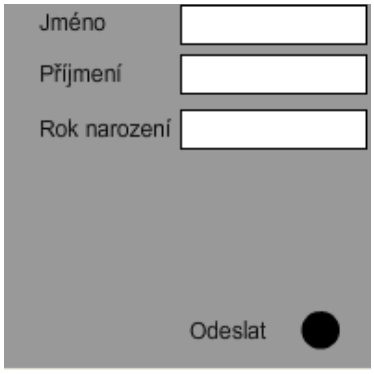
Příklad č.3

Formulář

Vytvořte jednoduchý formulář, který bude mít 3 textová pole a jedno tlačítko pro odeslání informací. Do polí se bude vkládat jméno, příjmení a rok narození. Po stisknutí tlačítka se data zpracují a přes formulář se vypíše text „jmenuješ se ‚xxx' ‚xxx' a narodil ses ‚xxx' ".

Postup:

Na plochu vložte 3 textová pole a nastavte je v kolonce **Properities** jako **Input text**. Jména instancí textových polí pojmenujte jako "**jmeno, prijmeni, rok**". Tlačítko můžeme použít z postupu v příkladu č.2. Dále vytvořte MovieClip(instanci pojmenujte jako "**vypis"**) a vytvoříme v něm dvě vrstvy. Do dolní vrstvy dáme obdélník jako pozadí pro výpis zprávy a do horní vrstvy dáme textové pole(instanci pojmenujte jako "**informace"**), který nastavíme jako **Dinamic text** a typ dáme jako **multiline**. Tento MovieClip bude z počátku mimo obrazovku, ale po odeslání dat pomocí tlačítka přijede z leva a bude obsahovat informace odeslané z formuláře,



Obr.10.Formulář

Nakonec do tlačítka vložíme tento skript:

```
onClipEvent(load){
                        ukaz=0;
                        _root.vypis._x=-100;
                        }
on(rollOver){
                    play();
                    jed=0;
                    }
on(rollOut){
                jed=1;
                i=this._currentframe;
                }
on(release){
if(_root.jmeno.text!="" && _root.prijmeni.text!="" && _root.rok.text!="")
                        {
                        ukaz=1;
            _root.vypis.informace.text="Jmenuješ se "+_root.jmeno.text+"
"+_root.prijmeni.text+" a narodil ses v roce "+_root.rok.text;
                        }
                }
onClipEvent(enterFrame){
                                if(ukaz==1)
                                        {
                                        if(_root.vypis._x<95){
                                                                _root.vypis._x+=15;
                                                                }
                                        }
                                if(jed==1){
                                        gotoAndStop(i);
                                        i--;
                                        }

                                }
```

Popis:

1-4	Po načtení flashe se provedou příkazy uvnitř složených závorek
14-20	Vypsání dat do MovieClipu „vypis“
23-28	Jestliže proměnná ukaz=1(po stisknutí tlačítka), MovieClip „vypis“ přijede z leva doprostřed obrazovky

Příklad č.4

Propojení flash a PHP

Vytvořte formulář podobný jako v příkladu č.3 a přes něj odešlete data PHP skriptu(pomocí funkce **sendAndLoad**), který je zapíše do souboru. Po odeslání dat se vypíšou všechna zaznamenaná data na obrazovku.

Postup:

Vytvoříme stejný formulář jako v příkladu č.3 jenom s tím rozdílem že do MovieClipu "**vypis**" vložíme tlačítko(instanci pojmenujeme "**krizek**") a tlačítko pro odeslání dat pojmenujeme jako "**tlacitko**". V tomto příkladu nebude psát skript do tlačítka jako v předešlém příkladu, ale ukážeme si jak celý skript zapsat do prvního snímku, což může být mnohem výhodnější při dohledáváni chyb, protože pokud bychom dělali větší stránku s více tlačítky a různými objekty, mohlo by se nám rozdělovaní programu mezi jednotlivé objekty vymstít.

Do prvního snímku vložíme tento skript:

```
1  _root.vypis._x=-100;
2  _root.tlacitko.onRollOver = function(){
3                                                  _root.tlacitko.play();
4                                                  jed=0;
5                                                  }
6  _root.tlacitko.onRollOut = function(){
7                                                 jed=1;
8                                                 i=_root.tlacitko._currentframe;
9                                                 }
10 _root.tlacitko.onRelease = function()
11 {
12 if(_root.jmeno.text!="" && _root.prijmeni.text!="" && _root.rok.text!="")
13                         {
14                         ukaz=1;
15                         }
16         var info:LoadVars = new LoadVars();
17         info.jmeno = _root.jmeno.text;
18         info.prijmeni = _root.prijmeni.text;
19         info.rok = _root.rok.text;
20         info.sendAndLoad("http://localhost/priklad2/book.php", info, "POST");
21         info.onLoad = function() {_root.vypis.informace = info.vystup;}
22 }
23 _root.vypis.krizek.onRelease = function(){ukaz=0;}
24 onEnterFrame = function()
25 {
26 if(jed==1){
27                 _root.tlacitko.gotoAndStop(i);
28                 i--;
29                 }
30 if(ukaz==1){
31                   if(_root.vypis._x<95){
```

```
32                                  _root.vypis._x+=15;
33                                  }
34              }
35              else if(ukaz==0){
36                          if(_root.vypis._x>-100){
37                                                  _root.vypis._x-=15;
38                                                  _root.jmeno.text="";
39                                                  _root.prijmeni.text="";
40                                                  _root.rok.text="";
41                                                  }
42                          }
43                      }
```

Popis:

16 Vytvoření kontejneru pro odeslání proměnných
17-19 Deklarace proměnných k odeslání
20 Poslání obsahu kontejneru „info“ do PHP scriptu
21 Funkce která stále běží dokud se nenačtou data odeslaná PHP skriptem zpět do flash.

PHP skript:

```
1  <?
2  if ($jmeno != "") {
3  $data="<p>".$jmeno." ".$prijmeni." ".$rok."</p>";
4  $fp = fopen("book.txt", "a");
5  fwrite($fp, "$data", 102400);
6  fclose($fp);
7  }
8  $obsah = file_get_contents("book.txt");
9  echo "vystup=".$obsah;
10 ?>
```

Příklad č.5

Klient v PHP-název státu

Jednoduchý program pro zjištění celého nazvu státu z jeho skratky.

Webovou službu voláme z adresy

'**http://www.webservicex.net/country.asmx?WSDL’.**

Část zdrojového kódu wsdl souboru:

```
<wsdl:definitions targetNamespace="http://www.webserviceX.NET">
        <wsdl:types>
        <s:schema elementFormDefault="qualified"
targetNamespace="http://www.webserviceX.NET">
        <s:element name="GetCountryByCountryCode">
```

```
        <s:complexType
        <s:sequence>
<s:element minOccurs="0" maxOccurs="1" name="CountryCode" type="s:string"/>
</s:sequence>
</s:complexType>
</s:element>
.....
```

Z wsdl kódu můžeme vyčíst že metoda pro zjištění názvu státu má název **GetCountryByCountryCode** a proměnná do které budeme dávat zjišťovanou zkratku státu je **CountryCode.** Další metody této webové služby můžeme zjistit vypsáním adresy do internetového prohlížeče.

Níni k samotnému PHP skriptu:

Jako první musíte připojit k vašemu PHP skriptu Knihovnu **nuSOAP**. PHP 4.x nemá podporu pro SOAP volání, tak je třeba tuto knihovnu použít. Knihovna nuSOAP je univerzální knihovnou jak pro WS klienty, tak pro WS servery.

Knihovnu stáhnete na adrese „**http://sourceforge.net/projects/nusoap/“.**

Skript:

```
1  <?
2  include ("nusoap.php");
3  $client = new soapclient('http://www.webservicex.net/country.asmx?WSDL',
4  true);
5  $vystup = $client->call('GetCountryByCountryCode',
6                    array(array('CountryCode'=>'cz')));
7  print_r($vystup);
   ?>
```

Popis:

2 připojení knihovny nuSOAP
3 připojení k webové službě
4 volaná metoda
5 poslání dat do proměnné
6 výpis dat poslané od WS serveru

Výstup bude vypadat takto:

```
    [GetCountryByCountryCodeResult] => <NewDataSet>
  <Table>
    <countrycode>cz</countrycode>
    <name>Czech Republic</name>    // žádaná odpověď
  </Table>
</NewDataSet>
```

Jak je vidět byl poslaný v XML formátu.Odpověď je mezi tagy “<name></name>“.

Příklad č.6

Počasí

Vytvořte program pro zjišťování počasí(teplota a podnebí) ve všech český meteorologických stanicích pomocí služby Global weather na adrese:

http://www.webservicex.net/globalweather.asmx?WSDL

Postup:

Jako první si vytvořte v HTML seznam SELECT do kterého vložíte názvy meteorologických stanic a tlačítko pro odeslání dat. Prostředí by mohlo vypadat například jako na obr.10.

Obr.11.Prostředí pro příklad č.6

Volání v PHP skriptu bude obdobné jako v předchozím příkladu.V tomto příkladu je také ukázané jak z odpovědi v XML formátu parsovat potřebná data a pracovat s nimi jako s proměnnými.

Skript:

```
1   <?
2   if($_POST['f']){
3   include ("nusoap.php");
4   $mesta=$_POST['mesta'];
5   $client = new soapclient('http://www.webservicex.net/globalweather.asmx?
6  WSDL',true);
7   $vystup = $client->call('GetWeather',array(array('CityName'=>$mesta,
8  'CountryName'=>'Czech Republic')));
9   $data=(join(" ",$vystup));
10  if(strpos($data,"<SkyConditions>")!=false)
11  {
12          $pozice_mraky=strpos($data,"<SkyConditions>")+16;
13          $delka_mraky=strpos($data,"</SkyConditions>")-$pozice_mraky;
14          $mraky="Je ".substr($data,$pozice_mraky,$delka_mraky-1);
```

```
15
16  if($mraky=="Je partly cloudy"){$adresa="./castecne_zamraceny.jpg";}
17  else if($mraky==" Je overcast"){$adresa="./zatazeno.jpg";}
18  else if($mraky==" Je mostly clear"){$adresa="./vedsinou_jasno.jpg";}
19  else if($mraky==" Je mostly cloudy"){$adresa="./vedsinou_zatazeno.jpg";}
20  else{$adresa="./vedsinou_zatazeno.jpg";}
21  }
22  else{$mraky="Podnebí neni zadáno";}
23
24  $pozice_teploty=strpos($data,"<Temperature>")+13;
25  $teplota=substr($data,$pozice_teploty,20);
26
27  $vypis = "<p><h2>".$mesta."</h2><p><strong>Teplota je:
28 </strong>".$teplota."<p> ".$mraky."<br><IMG src='".$adresa."'width='90'
29 hspace='60'>";
30  }
31  ?>
32
33  <html>
34  <head>
35          <title>Pocasi</title>
36  </head>
37  <body>
38  <FORM method="post" action="<? $_SERVER['PHP_SELF'];?>" name="f">
39                  <select name="mesta">
40                  <option value="Holesov">Holesov</option>
41                  <option value="Karlovy Vary">Karlovy Vary</option>
42                  <option value="Ostrava / Mosnov">Ostrava</option>
43                  <option value="Praha / Ruzyne">Praha</option>
44                  <option value="Brno / Turany">Brno</option>
45                  </select>
46                  <INPUT type="submit" value="poslat" name="f">
47  </FORM>
48  <? echo $vypis; ?>
49  </body>
50  </html>
51
```

Popis:

5-8	Poslání dat webové službě
10-14	Parsování aktuální informace o podnebí z odpovědi webové služby, která je v XML formátu
16-22	Podmínky pro vložení určitého obrázku, který znázorňuje aktuální podnebí poslané webovou službou.
24-25	Parsování dat teploty
27-29	Vypis informací
33-51	HTML prostředí s definovaným seznamem SELECT

ZÁVĚR

Macromedia Flash je výborný nástroj pro vytváření webových stránek a aplikací, který se stále vyvijí a v budoucnu se mu dostane ještě vetší popularity než dnes. Při vytváření této práce jsem zjistil, že pro masivnější rozšíření flashe chybí to, že není možné takto vytvořené stránky optimalizovat pro vyhledávače. Tato nepříjemná vlastnost, však bude v budoucnu odstraněna v dalších verzích flashe, což slibují vývojáři.

Webové služby jsou poměrně mladou technologií, která ještě na svůj zlatý věk čeká. Ale již dnes je možno využít mnoha užitečných služeb, které jsou praktické pro zlepšení kvality webových stránek, ale také k použití na nejrůznější aplikace pro počítače různých platforem.

SEZNAM POUŽITÝCH ZDROJŮ

[1] Flash seznámení. Dostupné z: <http://flash.jakpsatweb.cz/>

[2] Flash-Základní pravidla. Dostupné z: <http:<//flash.jakpsatweb.cz/>

[3] Flash-Nástroje. Dostupné z: <http:<//flash.jakpsatweb.cz/>

[4] Flash-knihovna. Dostupné z: <http:<//flash.jakpsatweb.cz/>

[5] Flash-Snímky a pohyb. Dostupné z: <http:<//flash.jakpsatweb.cz/>

[6] Franclin,D.,Makar,J. Macromedia Flash MX ActionScript. Praha:SoftPress 2003. ISBN 80-86497-43-7

[7] Cílové cesty. Dostupné z: http://oflashi.net/

[8] Proměnné. Dostupné z: http://oflashi.net/

[9] Flash - Objekty. Dostupné z: <http://flash.jakpsatweb.cz/>

[10] Offline komunikace mezi HTML a Flash. Dostupné z: <http://www.flash.cz/>

[11] Webové služby. Dostupné z: <http://www.daquas.cz/fox/>

[12] Jak fungují webové služby.Dostupné z: <http://interval.cz/>

[13] Webové služby. Dostupné z: <http://www.kosek.cz>

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

PHP	Skriptovací jazyk
HTML	Jazyk pro tvorbu WWW stránek

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH

P 1 : Přenosné médium CD s vytvořenými materiály