flash.help

Offline verze webu Flash help (http://flash.jakpsatweb.cz)

Obsah

Frequently asked questions

Toto jsou dotazy, které často objevují v diskuzi a mailech, jsou zde abych na ně nemusel odpovídat stopadesátkrát. Pokud máte sami nějaký problém nebo nápad, který jste nenašli pomocí vyhledávače ani v diskuzi, stačí, když mi napíšete mail (hozikm@centrum.cz), nebo použijete formulář vlevo.

Q:

Jak udělám 3D objekt?

A:

Trojrozměrné objekty nejdou ve Flashi nakreslit přímo. Jsou v podstatě dvě možnosti jak 3D objekt vytvořit. Buď pomocí ActionScriptu, což je poměrně složitá záležitost a používá se v případě, kdy se jedná o jednodušší objekt, který zpravidla má na něco reagovat (třeba na pohyb myši). No a nebo druhý způsob, kdy externí 3D program vygeneruje pohyb objektu jako serii snímků se statickou grafikou a uloží jako SWF animaci. Takto funguje například Swift3D nebo Vectra3D a spoustu dalších.

Q:

Jak mám vytvořit ve Flashi formulář podle HTML formuláře (se zachovanou funkčností).

A:

Tak tedy, nejprve HTML předloha, třeba tato:

```
<form action="http://www.server.cz/skript.php" method="post">
   <input type="hidden" name="uzivatel" value="165879">
   <input type="text" name="subject" value="dotaz">
   <textarea name="zprava">sem napište, co máte na srdci...</textarea>
   <input type="submit">
</form>
```

A teď jak to udělat ve Flashi:

- skryté pole (hidden) v HTML nejsou vidět a nejdou editovat. Jsou ekvivalentní k Flashovým proměnným. Proto tedy tyto pole nahradíme globálními proměnnými, které můžeme vložit do prvního snímku.

```
uzivatel = 165879;
```

- kolonka (text) je editovatelná a proto ji nahradíme Input textem "Single Line" a do pole "***Var***" napíšeme jméno proměnné. V našem případě "***subject***"

- textová oblast (textarea) je také editovatelná a navíc dovoluje zadat víceřádkový vstup - nahradíme ji tedy Input textem "Multiline" a do pole "***Var***" napíšeme "***zprava***"
- nakonec máme odesílací tlačítko (submit). To ve Flashi nahradíme klasickým ***tlačítkem*** (button), kterému definujeme:

```
on (release) {
   getURL("http://www.server.cz/skript.php", "", "POST");
}
```

Předpokládám, že alespoň trochu ovládáte HTML. Pokud ne, navštivte www.jakpsatweb.cz, kde je vše velmi podrobně vysvětleno.

Q:

Jaký je příkaz pro stažení souboru?

A:

Tento dotaz je zcela irelevantní. Ve Flashi, stejně jako v HTML, nelze definovat, co se má otevřít a co stáhnout na disk. Pokud tedy použijeme příkaz:

```
getURL("http://www.flash-help.wz.cz/stranka.html")
```

...otevře se nám HTML stránka v okně prohlížeče. Pokud však nalinkujeme soubor s příponou, kterou má prohlížeč nastavenu jako nespustitelnou:

```
getURL("http://www.flash-help.wz.cz/soubor.zip")
```

...zobrazí se žádost o stažení souboru.

Q:

Jak mám načíst proměnnou z externího souboru a pak ji do něj zpátky uložit?

A:

Tyto dotazy se poslední dobou hodně množí v mé e-mailové schránce, tak tedy:

- ***Načtení***
  pro načtení proměnné použijeme následující příkaz:

```
loadVariables("soubor.txt", "movieclip", )
```

- soubor.txt .....zdojový soubor
- movieclip .....tzv. ***target*** - je to cíl, kam se natáhnou proměnné. Většinou tato volba slouží ke kontrole, jestli je načítaní dat ukončené. Instanci MovieClipu "movieclip" pak definujeme třeba toto:

```
onClipEvent (data) {
    _root.Play();
}
```

Takto je zaručené, že animace bude pokračovat v přehrávání až po úspěšném natažení všech proměnných (přirozeně musí být animace zastavena pomocí příkazu `Stop()`).

Důležité! - soubor musí být umístěn ve stejné subdoméně, jako animace (bezpečnostní opatření)

- ***Uložení***
  Samotný Flash nemá povolen jakýkoliv zápis dat do souborů (podobně jako JavaScript). Zápis dat proto musíme svěřit nějakému serverovému (pracuje na serveru) skriptu - např. PHP

  V samotné animaci můžeme použít následující příkaz:

```
getURL("skript.php", "", "POST");
```

Soubor ***skript.php*** bude vypadat následovně:

```
<?
   $fp = fopen("soubor.txt", "w");
   fwrite($fp, "nazev_promenne=$nazev_promenne");
   fclose($fp);
?>
```

Nakonec připomenu, že server musí mít nainstalovaný PHP interpreter
(např.: www.webzdarma.cz)

Q:

Jak udělám odkaz na e-mail (aktivní e-mail)?

A:

Stejně jako v HTML:

```
getURL("mailto:hozikm@centrum.cz");
```

Q:

Ve Flashi mi nefunguje čeština u některých fontů, přitom v ostatních aplikacích (Word) je to v pohodě.

A:

Flash má (až to verze MX 2004) problém se zobrazením správné diakritiky u tzv. OpenType fontů (Verdana, Tahoma, v novějších Windowsech i Arial a Times New Roman). Verze MX již podporuje Unicode UTF-8 a slibovala i bezvadnou podporu národních znakových sad - nestalo se. Je tedy nutná úprava souboru ***win.ini*** resp. ***registrů***:

- ***Windows 95/98/98SE/ME***
  - je nutná změna v souboru ***x:\windows\win.ini***
  - stačí přidat do sekce ***[FontSubstitutes]*** následující příkazy:

```
Times New Roman CE,0=Times New Roman,238

…atd. pro každý font, který používáte.
```

- ***Windows XP***
  - tento systém již soubor win.ini nepoužívá, je tedy nutná změna registrů
  - nastavení je možné buď pomocí programu ***regedit***, nebo pomocí editačního souboru (koncovka REG)
  - otevřete si poznámkový blok (notepad) a do něj napište následující příkazy:

```
Windows Registry Editor Version 5.00

[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\
CurrentVersion\FontSubstitutes]
"Times New Roman CE,0"="Times New Roman,238"
"Verdana CE,0"="Verdana,238"

…atd. pro každý font, který používáte.
```

  - nakonec uložte soubor s koncovkou ****.REG*** a spusťte, a na stupidní dotaz systému, jestli jste si jist, odpovězte "Ano"

Tuto úpravu lze rovněž provést nějakým programem. Já znám například ***WGL Assistant***, ale určitě se najdou i další.

Q:

Mám velký problém při posílání proměnných z Flashe do PHP scriptu. Místo znaků s háčky a čárkami se mi objevují nějaké paznaky. Přitom ve Flashi 5 je to bez problému.

A:

Měl jsem podobný problém s češtinou ve Flashi MX. Je to způsobeno tím, že Flash MX začal používat kódování Unicode-UTF 8 a to někdy dělá problémy, zejména při komunikaci s aplikacemi, které UTF-8 neovládají (např. PHP). Stačí vložit někde na začátek animace tento příkaz: `system.useCodepage = true`.

Q:

Jak velkou FrameRate mám používat?

A:

To záleží na povaze animace. Implicitní nastavení je 12fps. Obecně platí, že pro hodně pohyblivou animaci se hodí něco mezi 16 až 25fps - větší framerate už je zbytečná. Naopak pro převážně statickou animaci (menu, textové pole, formuláře), kde se využívá spíše barevných a odstínových transformací, stačí i míň. Mějte však na paměti, že především starší počítače mohou mít s 18 snímky za sekundu problémy. Poněkud odlišnou skupinu tvoří animace, ve kterých se používají akce typu ***Drag*** (uchopení) a ***DuplicateMovieClip*** (duplikování). V těchto animacích je kvůli nutnosti rychlé zpětné vazby potřeba zvednout framerate až na astronomických 50 - 70 fps, nebo použít funkci `updateAfterEvent()`.
Pamatujte na to, že nejvíc počítač zatěžují průhledné objekty (alpha) - nejhůř, když se jich několik překrývá.

Q:

Jak otevřu pomocí příkazu GetURL stránku v novém okně nebo případně v určitém rámu?

A:

Pro tyto účely existuje u příkazu GetURL roletka s názvem "Window"

- `_self` = stejný frame
- `_blank` = nové okno
- `_parent` = frame o úroveň výš
- `_top` = nejvyšší úroveň (celé okno)

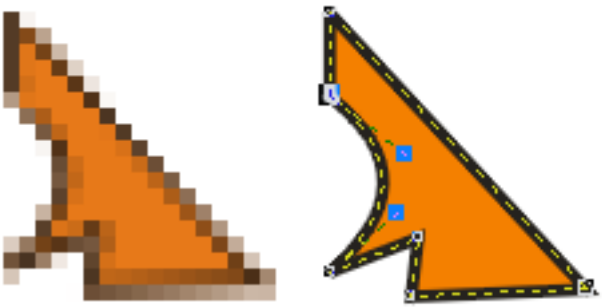
Pokud zaškrtnete "Expression", můžete napsat přímo název rámu, kde se má volaná stránka zobrazit.

```
getURL("http://www.flash-help.wz.cz", "_blank");
```

Seznámení

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)



Možná vás napadne, jaké ***výhody*** přináší vektorová grafika oproti rastrové.

- Je datově mnohem menší
- Objekty nejsou rozměrově nijak limitovány (mohou být zvětšovány bez roztřepení a ztráty kvality)
- Výplně a čáry mohou být průsvitné, nebo vybarvené přechodovou výplní s minimálním zvětšením souboru.

Ale i vektorová grafika má své ***nevýhody*** a proto Flash umožňuje vkládat do animace i rastrové obrázky.

- vektorová grafika se nehodí na příliš složité obrazce (velké množství barev, složitý tvar - fotky)

Jak se vytváří a jak přehrává?

Flash animace se vytváří ve Flash editoru - zde se nakreslí (vloží) obrázky, umístí se do určitých vrstev, nadefinují se jejich pohyby a transformace v časové ose, mohou se přidat zvuky a skripty a nakonec se celá animace exportuje do formátu ***SWF***, který je možno přehrát.

Zde narážíme na první omezení - export do SWF animace je ***nevratný***. Proto pokud budete chtít animaci do budoucna editovat, zachovejte si pracovní dokument (formát ***FLA***)

Pokud jde o přehrávání animací, je to možné buď v ***prohlížeči*** (musí mít nainstalovaný Flash plugin - nové verze MSIE ho mají), nebo ve zvláštním přehrávači. Zde však většina uživatelů narazí na velký problém - kde ho vzít. Proto Flash umožňuje "přibalit" přehrávač k animaci a vytvořit tak EXE soubor (tzv. ***Projektor***) spustitelný na jakémkoliv počítači. Je ale potřeba mít na paměti, že touto operací vzroste velikost animace o cca 500kB, což je mnohdy desetinásobek původní velikosti. Projektor se proto používá zejména u Flashových prezentací a her.

Co lze ve Flashi vytvořit?

Prezentace, kreslené scénky, hry, webové aplikace, animovaná menu, reklamní bannery, celé webové stránky, a tisíce dalších věcí.

Několik dobrých rad pro vytváření webových prezentací

- ***Nedělejte všechno ve Flashi***
  Flash je mocný nástroj, ale nehodí se na všechno. Zejména ne na dlouhé texty, redakční systémy a podobné typy internetových prezentací. Nesnažte se proto používat Flash za každou cenu. Pamatujte na to, že vaše výtvory půjdou obtížněji aktualizovat, obtížněji tisknout a na starších počítačích vaši úchvatnou grafiku nahradí bílá plocha. Pokud tedy publikujete větší text, doporučuji udělat ve Flashi jen část stránky (třeba menu).
- ***Velké soubory jsou tabu!***
  Hlavní předností vektorové grafiky je její datová velikost (tedy vlastně malost). Využívejte toho. Flash dovoluje vkládat rastrové obrázky, ale toho by mělo být využito až ve stavu naprostého zoufalství. Dále zvětšují velikost animace zvuky - někdy i desetkrát! Videosoubory snad není nutné ani komentovat. Snažte se co nejvíc používat kopie symbolů v knihovně - symbol bude uložen jen jednou a jeho instance animaci nezvětšují.
- ***Myslete na zátěž procesoru***
  Co šetří vektorové objekty na velikosti souboru, to užírají procesoru. Ten totiž musí obdržené informace vyhodnotit a správně zobrazit. Nejnáročnější je vykreslování průhledných objektů - nejhůř, když se jich několik překrývá a navzájem se pohybují. Myslete proto na pomalejší počítače - to co vám běží plynule, může způsobit nemalé problémy jinde.
- ***Žádné kýčovitosti***
  Flash je hotovým rájem pro webdesignéry - nejsou zde svazováni nespolehlivým HTML a CSS pozicováním objektů a to pak svádí k nejrůznějším zločinům proti lidskosti. Snažte se tedy svou fantazii držet na uzdě a, i když nemusíte, dodržujte stále blokové schéma webů. Člověka sice potěší text přilétávající po písmenech, ale ne když ho pak musí číst po spirále.
- ***Fonty***
  Flash umožňuje přibalit použitý font do animace a zobrazit ho tak na všech počítačích správně. Přesto však nedoporučuji používat víc než 2 druhy písem a i tyto volit s rozmyslem. Zejména se vyvarujte exotickým písmům. Nikdo nebude číst texty napsané písmem ala "grafiti" nebo "matrix".
- ***Nesnažte se být chytřejší, než uživatel***
  Tuto chybu udělal Microsoft Word a svou činností připomíná spíš výherní automat. Nesnažte se tedy předvídat, co asi bude návštěvník chtít dělat a nechte ho ať si sám vybere. Mluvím především o otvírání oken prohlížeče způsobem "fullscreen", nekontrolovatelné přehrávání zvuků a podobné "znásilňování" uživatelů.

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.

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.

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.

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.

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.

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.

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.

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.

Žádná jiná společnost v současné době nenabízí tak pestrý balík webových služeb jako Macromedia.

Produkty pro webdesign od Macromedia:

Dreamweaver
WYSIWYG editor webových stránek s podporou PHP, ASP, JSP a ColdFusion.

Contribute
Program pro jednoduchou správu webů bez znalosti programování.

Flash
Editor vektorových Shockwave Flash animaci.

Fireworks
Editor rastrových obrázků s kvalitními funkcemi pro web.

Freehand
Editor vektorových obrázků - zvládá i jednoduché Flash animace.

Director
Program pro profesionální tvorbu multimediálních CD-ROM a 3D internetových aplikací.

Authorware
Tvorba interaktivních aplikací pomocí vizuálního programování se zaměřením na e-learning.

Homesite
Jeden z nejpopulárnějších strukturních HTML editorů.

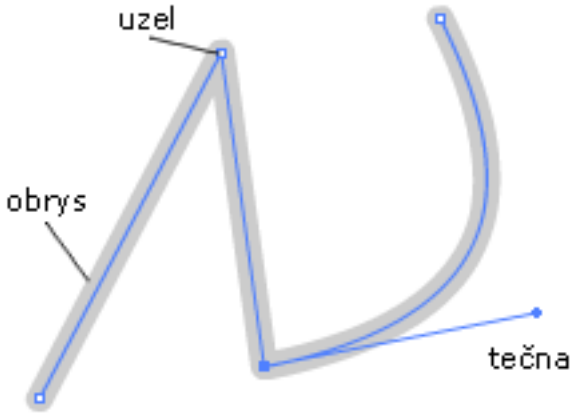
a další...

Základní pravidla

Před samotnou prací s Flashem je třeba si uvědomit několik skutečností, bez kterých se prostě nepohnete. Zkušenějším čtenářům budou nejspíš následující odstavce připomínat mlácení prázdné slámy, ale pro začátečníky je velmi důležité vědět co je to:

Čára

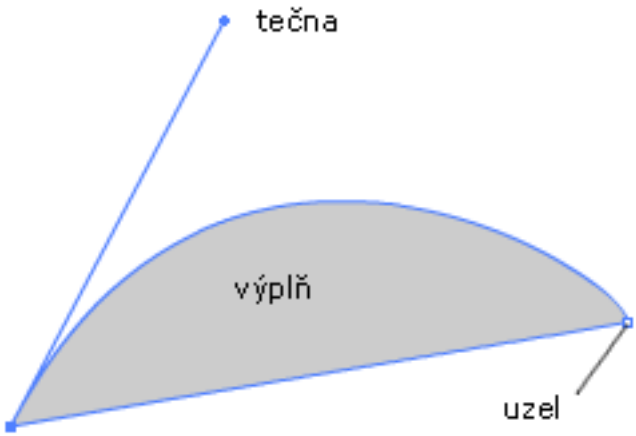
Je to objekt, který je definován jednotlivými kotvícími body (též uzly), které jsou spojeny buď úsečkou, nebo křivkou. Křivka je vždy definována ***tečnou*** v počátečním a koncovém bodě. Čára samotná ***nemá*** výplň, lze ji definovat pouze ***obrys*** (barvu a tloušťku). Bez ohledu jak je obrys čáry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



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.

Pokud chcete vytvořit výplň s obrysem, je to možné jedině použitím výplně v kombinaci s čarou.

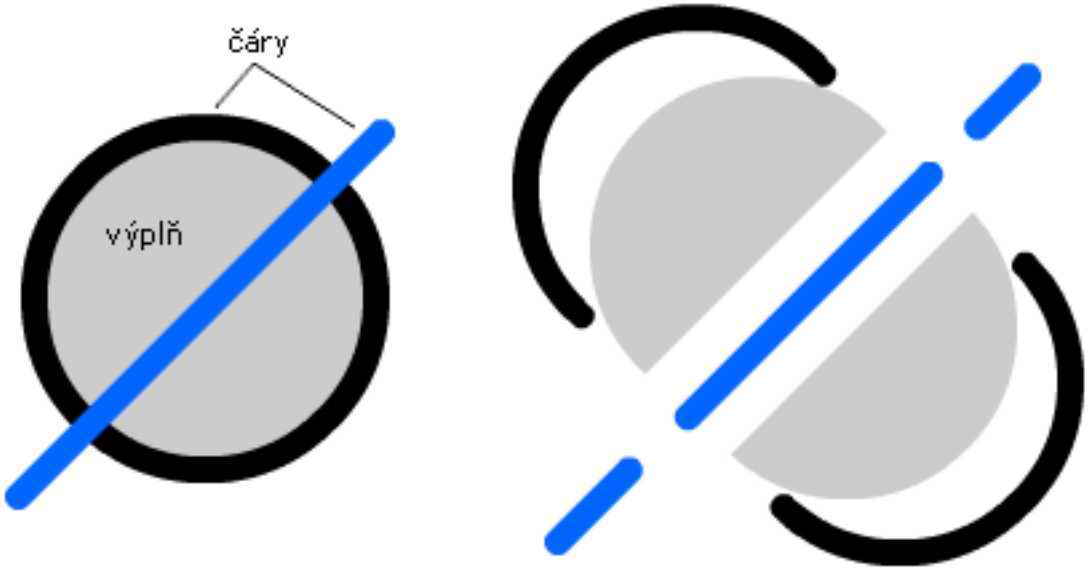


Vrstvy

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***. (Existují i speciální vrstvy, u kterých tato vlastnost neplatí, ale o tom až později.)

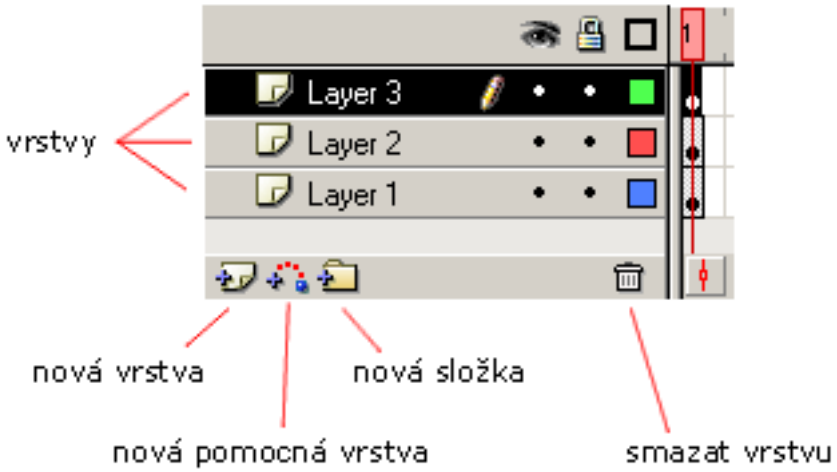
Znamená to tedy, že objekty nakreslené v různých vrstvách na sebe nemají ***žádný vliv***.

Asi vás bude zajímat, co se stane s objekty ve ***stejné vrstvě***. Pokud se ve Flashi nějaké objekty překrývají, automaticky se rozdělí na díly určené jejím průnikem.



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

Vrstva na nejvyšší úrovni je nahoře (Layer 3) a na nejnižší dole (Layer1). Vrstvy lze posouvat uchopením a tažením na požadovanou pozici. Aktivní vrstva (ta, do které právě kreslíme) je zvýrazněna černě (Layer 3) Vrstvy se vytváří a odstraňují následujícími tlačítky:

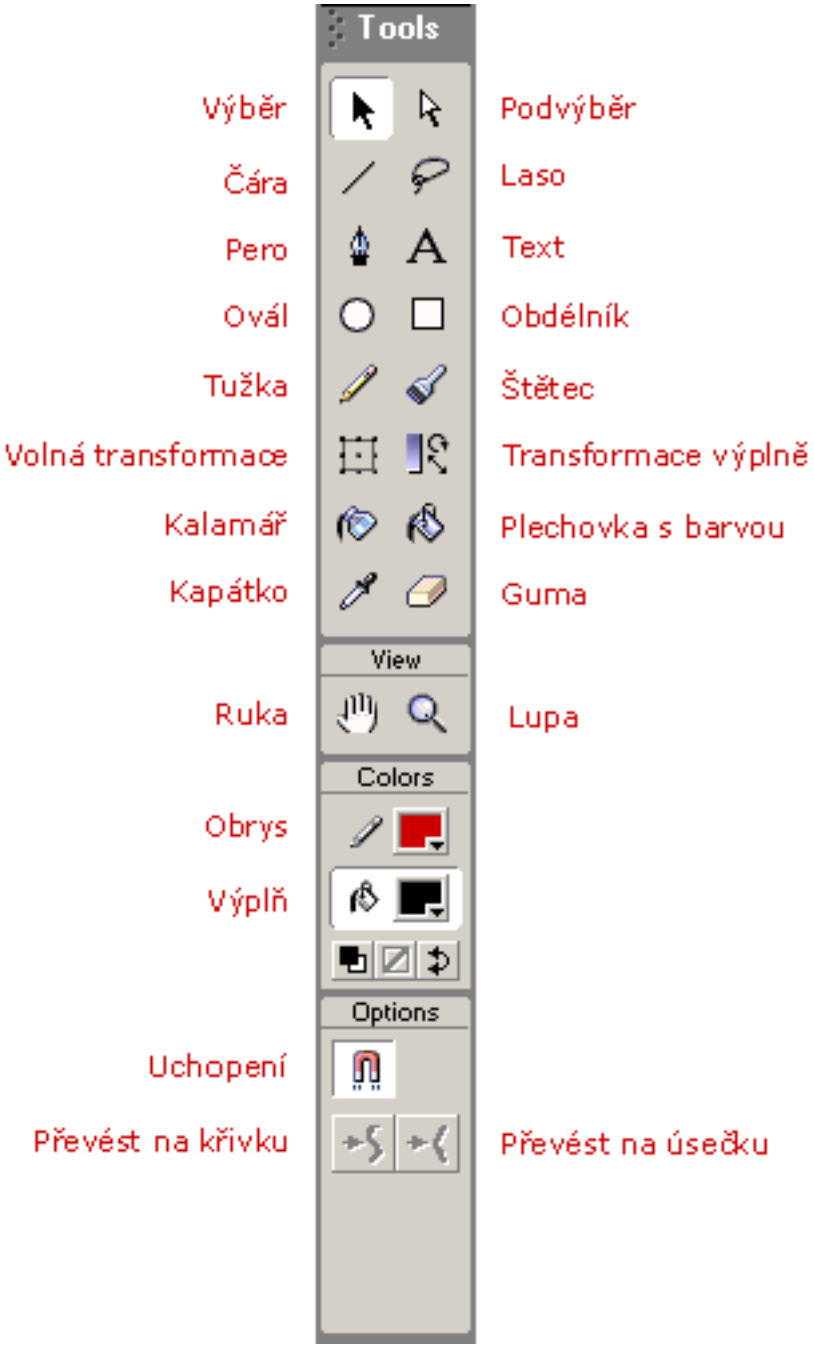


Název vrstvy lze měnit jedním kliknutím na něj. Každou vrstvu lze ***zhasnout*** (symbol oko) nebo ***zamknout*** (symbol zámek - přirozeně). Barevný čtvereček vpravo zobrazuje barvu, která reprezentuje danou vrstvu. Když potom kliknete do prostoru čtverečku, budou objekty v této vrstvě zobrazeny jen ***obrysově*** (v nastavené barvě). Ve verzi 6 (MX) se objevila ***složka***, která umožňuje sbalit určitý počet sousedních vrstev dohromady.

Všechny tyto úpravy neovlivňují konečnou podobu animace - slouží jen ke zjednodušení práce.

Základní nástroje

Abyste mohli vytvořit animaci, musíte mít přirozeně co animovat. V této kapitolce vás naučím, jak ovládat základní nástroje Flashe.



- ***Výběr***
  Slouží k výběru objektu. Jedním kliknutím vyberete jeden objekt (čára, výplň), tažením můžete vybírat více objektů (nebo jen jejich části). V poli "Options" můžete nastavit:
  - Uchopení - objekty se budou uchopovat na již nakreslené obrazce

    
  - Převést na křivku - vyhladí vybranou čáru

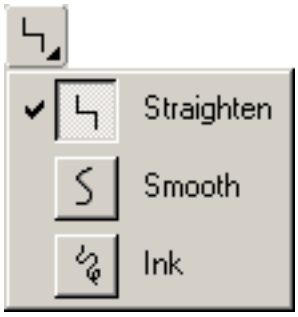
- Převést na úsečku - zruší zakřivení vybrané čáry

- ***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 (vytváříte) zakřivení čáry.
- ***Čára***
  Kreslí rovné čáry. Čáru lze kreslit kliknutím (a držením). Tloušťka obrysu lze zvolit na panelu "Properties"
- ***Laso***
  Má stejné použití, jako výběr, ale je možné vybírat i nečtvercové plochy. Klikněte a pohybem myši ohraničte plochu (objekt), kterou chcete vybrat. Puštěním se plocha sama uzavře
- ***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.
- ***Text***
  Jedním kliknutím umístíte kurzor do stránky a napíšete text, tento text se nebude zalamovat. Pokud místo jednoho kliknutí roztáhnete textovou oblast do určité šířky, slovo při dosažení této šířky automaticky skočí na nový řádek (takovou textovou oblast poznáte tak, že má vpravo nahoře místo kolečka čtvereček). Velikost a druh písma lze zvolit na panelu "Properties"
- ***Ovál a Obdélník***
  Kliknutím a tažením vytvoříte obrazec.

  - V poli "Options" můžete nastavit zakulacení rohů

- ***Tužka***
  Umožňuje kreslit ručně čáry. Na panelu "Options" je možno zvolit, jak má výsledná čára vypadat:

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



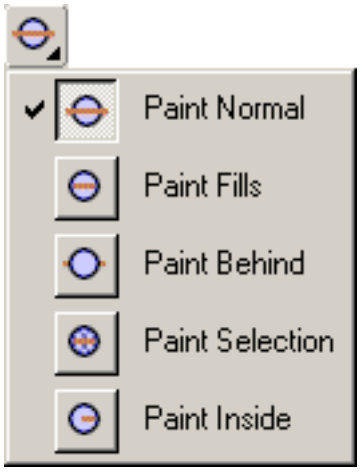
- ***Štětec***
  Narozdíl od tužky nekreslí čáry, ale výplně (nejsou ohraničeny čarami). Při použití tohoto nástroje si můžete na panelu "Options" zvolit:

- Tloušťku stopy (Brush Size)

- Typ stopy (Brush Shape)

- Proměnlivou tloušťku stopy (Use Pressure) - jen pokud máte tablet citlivý na tlak.

- Kam se barva má nanést (Brush Mode)



  - Normal = přes všechny objekty
  - Fills = nekreslí přes čáry
  - Behind = nekreslí přes objekty
  - Selection = kreslí jen přes předem vybraný objekt (pomocí nástroje "výběr" nebo "laso")
  - Inside = kreslí jen přes jeden objekt najednou

- Absolutní, nebo relativní pozici výplně (Lock Fill)

- ***Volná transformace***
  Pomocí tohoto nástroje lze vybraný objekt ručně otáčet, zešikmovat a měnit jeho velikost - viz kapitola barvy a atributy
  V poli "Options" je možné upřesnit, co chcete s objektem udělat:

- Otočit

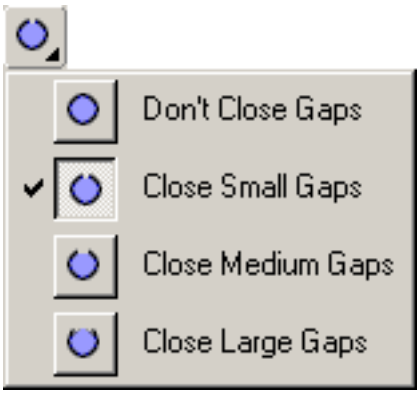
- Zvětšit

- Roztáhnout jednotlivé body

- Zdeformovat

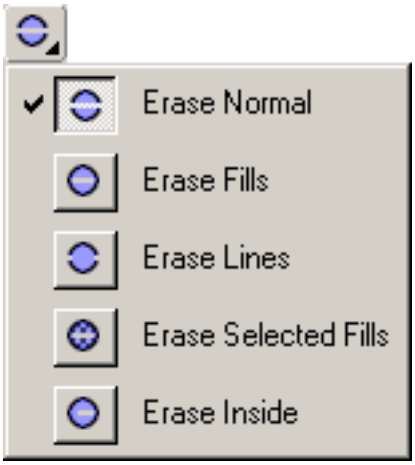
- ***Transformace výplně***
  Umožňuje měnit velikost, natočení a zkosení přechodové výplně. Viz "Barvy a Atributy"
- ***Plechovka s barvou a Kalamář***
  Umožňují měnit barvu výplně respektive barvu čáry. Vybráním tohoto nástroje a klepnutím na objekt se tento přebarví na nastavenou barvu v poli "colors". Pokud klepnete nástrojem "plechovka s barvou na čarami ohraničenou plochu, vyplní se nastavenou výplní.

  - V poli "Options" je možné nastavit jak velké mezery má výplň zacelit



- ***Kapátko***
  Vybráním tohoto nástroje a kliknutím na objekt levým tlačítkem nastavíte barvu objektu do pole colors.
- ***Guma***
  Maže nakreslené objekty. Podobně, jako u nástroje "Štětec" (Brush) je i zde možné 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

Na závěr bych rád upozornil, že některé nástroje jsou jen stručně popsány - většinou ty, které jsou víc než jasné. Předpokládám alespoň minimální znalost nějakého kreslícího programu (když už nic, tak alespoň Paintbrush). Kdo si dnes poprvé sednul k počítači, nechť si trhne lopatkou.

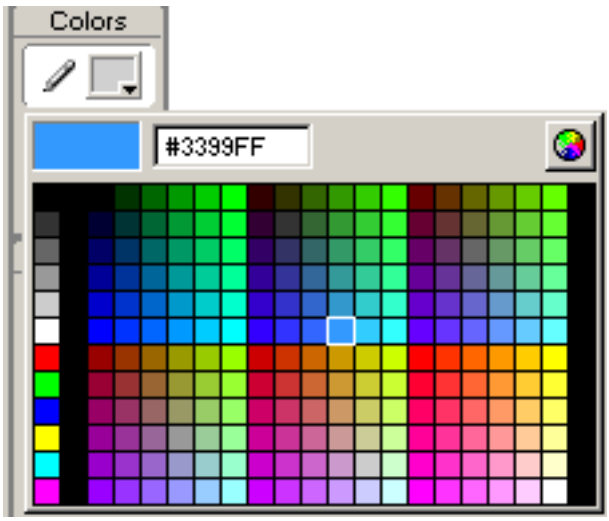
Barvy a atributy

Barvy (Colors)

Z minula už víte, že obrysu čáry a výplni můžeme definovat barvu.

Barvu výplně a čáry můžeme definovat ještě před nakreslením objektu, nebo už nakreslenému (musíme ho ale nejprve vybrat nástrojem "výběr" nebo "laso")

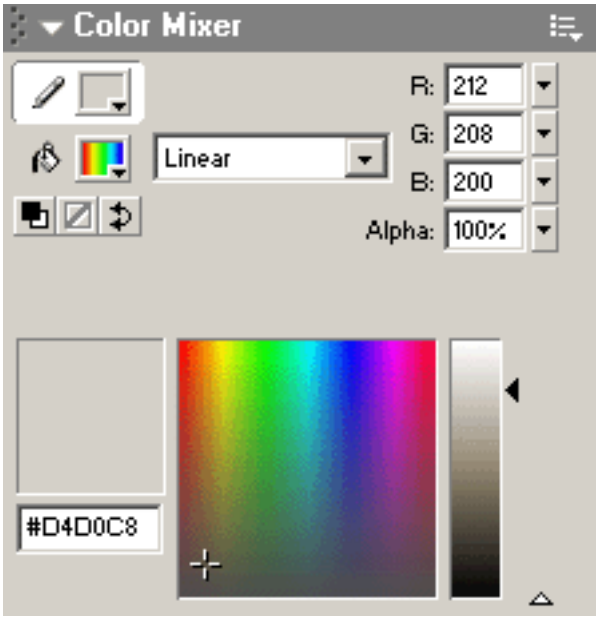
Pokud klepnete na panelu "colors" na rámeček s barvou obrysu či výplně, otevře se vám panel, kde můžete barvu buď vybrat ze seznamu, nebo napsat její hexadecimální (#D4D0C8) hodnotu do pole, anebo zmáčknout tlačítko vpravo nahoře a vybrat barvu z klasické palety windows.



pozn.: po otevření panelu se kurzor změní na kapátko a můžete vybrat barvu z kteréhokoliv místa na obrazovce.

Barevný přechod (Color Gradient)

Narozdíl od barvy obrysu čáry (která může mít jen jednotnou barvu), může mít výplň přechodovou barvu. K vytváření takových výplní slouží panel s názvem "Color Mixer". Zobrazíte ho volbou v menu ***Window - Color Mixer***.



Zde si také můžete vybrat barvu výplně a čáry buď výběrem z 256 barev nebo pomocí HSB (Hue, Saturation, Brightness) palety.

Pokud v rozbalovacím seznamu zvolíte ***Linear*** = lineární přechod nebo ***Radial*** = kruhový přechod, objeví se následující lišta, na které je možno definovat podobu barevného přechodu.

Přechodový bod přidáte jedním kliknutím do místa, kam jej chcete umístit. Odstranění již existujícího bodu provedete uchopením a přetažením "ven". Editaci barvy provedete vybráním bodu (kliknutí) a volbou barvy (vlevo nahoře nebo na HSB paletě dole).

Možná to vysvětluji trochu moc polopaticky, ale raději ještě připomenu, že posunutí bodu se provede uchopením a tažením na novou pozici.

Rastrové výplně

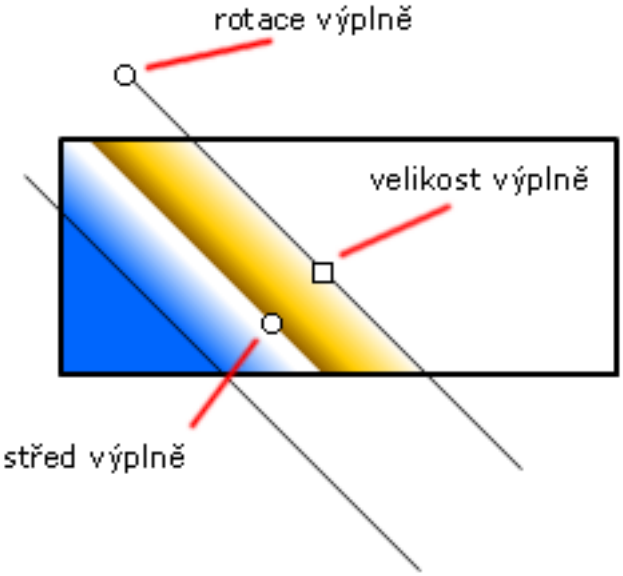
Ve výplni lze místo barvy použít také rastrový obrázek (např.: GIF, JPEG, ...). Tento obrázek pak bude dlaždicovitě vyskládán v prostoru výplně.

Abyste mohli použít jako výplň rastrový obrázek, musíte tento obrázek nejprve vložit do knihovny (jak? - viz kapitola "Knihovna").

Když poté zvolíte na panelu "Color Mixer" jako výplň "bitmap", nabídnou se vám v rámečku níže všechny rastrové obrázky obsažené v knihovně. Jeden z nich zvolte a výsledek bude vypadat třeba takto:

Transformace výplně

Pozici, velikost a natočení přechodové a rastrové výplně lze ovlivnit nástrojem "Transformace výplně" (Fill Transform)
Tímto nástrojem je možno měnit ***pozici středu***, ***natočení*** a ***velikost*** pomocí jednotlivých editačních bodů:



Průhlednost (Alpha)

Jak ***Obrys***, tak i ***Výplň*** může mít definovanou průhlednost. Rovněž jeden (nebo i více) bodů přechodové výplně může mít definovanou průhlednost.

Nastavit ji můžete opět na panelu "Color Mixer".

Abyste vůbec mohli nějakou průhlednost pozorovat, musíte mít více objektů rozmístěných do více vrstev a musí se přirozeně překrývat. (viz základní pravidla)

Pamatujte na to, že průhledné objekty silně zatěžují procesor. Proto se vyvarujte používání velkých průhledných objektů v rychlých pohybových sekvencích.

Transformace

Na panelu "Transform" (zobrazíte jej volbou v menu ***Window*** - ***Transform***) je možné editovat procentuální šířku a výšku, otočení (rotate) a sklon (skew) jakéhokoliv objektu (objektů). Snad jen doplním, že zrušením zaškrtnutí pole "Constain" půjde měnit výška a šířka nezávisle na sobě.

Zarovnání (Align)

Další velmi jednoduchý panel s velmi dobře vizuálně pochopitelnými funkcemi (***Window*** - ***Align***). Zkrátka a dobře - vyberete dva nebo více objektů, zmáčknete jedno z tlačítek a vybrané objekty se srovnají podle toho, co jste zmáčkli.

Snad jen doplním, že "To Stage" znamená zarovnání vzhledem k papíru, nikoliv relativně k vybraným objektům.

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***). Na co je tato sranda vlastně dobrá? Je totiž jedno, kolik instancí symbolu vložíte do animace - symbol bude uložen jen jednou.

Druhy symbolů

Existují 3 základní druhy symbolů:

- ***Grafika (Graphic)***

  Nejjednodušší symbol. 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)

  Tlačítkům se budu blíže věnovat kapitole "řízení animace"
- ***Klip (Movie Clip)***

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

Nový symbol vytvoříte buď v menu ***Insert - New Symbol***. Již nakreslenou grafiku převedete na symbol v menu ***Insert - Convert to Symbol*** (F8)

Instanci ze symbolu na scéně vytvoříte jednoduchým přetažením symbolu z knihovny na scénu.

Instance symbolů můžete vkládat i do jiných symbolů (například do symbolu ***Button*** můžete vložit instanci ***Movie Clipu***). Nelze však (samozřejmě) vkládat instanci symbolu do téhož symbolu (***sám do sebe***).

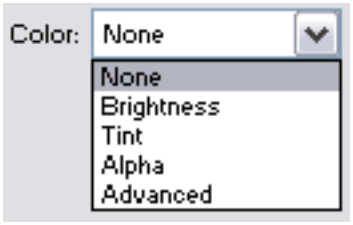
Změnu druhu symbolu provedete pravým kliknutím na název symbolu v knihovně a v menu ***behavior*** určíte druh.

Transformace instance

Asi teď namítnete, že těžko budete do animace vkládat 50 stejných obrázků. Není to tak docela nutné. Říkal jsem, že 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. Jen bych rád upozornil na fakt, že jakákoliv změna instance ***nemá žádný vliv*** na rodičovský symbol a naopak změna symbolu ***ovlivní všechny jeho instance!!***



- ***Výška a šířka, rotace, sklon***
  viz "barvy a atributy"

- ***Jas (brightness)***
  Tato volba umožňuje natavit úroveň jasu celé instance. Myslím, že obrázek opravdu není nutný. Zkrátka: ***+100%*** = bílá, ***-100%*** = černá (***0%*** = beze změny)
- ***Barevný odstín (tint)***
  Tento atribut si lze představit jako barevná fólie před instancí

  Vyberete barvu (buď pomocí palety, nebo napíšete její hodnotu RGB) a nakonec nastavíte krytí (sytost) odstínu.
- ***Průhlednost (alpha)***
  Průhlednost celé instance vůči objektům v nižších vrstvách. (viz vrstvy)

Speciální symboly

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***

viz kapitola "vložení zvuku"

- ***Složka***

Nemá žádnou funkci, slouží jen k přehlednější organizaci objektů v knihovně. Existuje až od verze 6 (MX)

- ***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 - viz také UI Komponenty

- ***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 (například UI komponenty).

- ***Video***

Flash od verze 6 (MX) umožňuje vkládat do animace videa ve formátech MPEG, DV, MOV a AVI

Vložení externích souborů do knihovny lze pomocí příkazu ***Import to Library*** v menu ***File***.

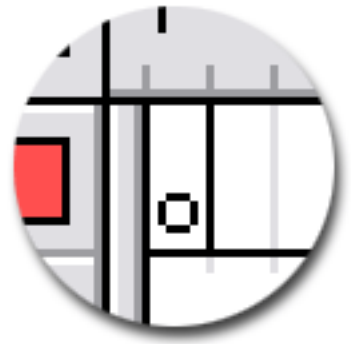
Snímky, pohyb objektů

Doposud jsme se bavili jen o vytváření a úpravě statické grafiky, v této kapitolce vás naučím tuto grafiku rozhýbat. Na začátek ještě jedno upozornění - některé funkce (např. maska) jsou vidět až po exportu animace do formátu SWF. Není to však bezpodmínečně nutné. Na to, jak bude animace vypadat se můžete podívat příkazem "***Test Movie***" v menu "***Control***" (Ctrl + Enter)

Časová osa a snímky

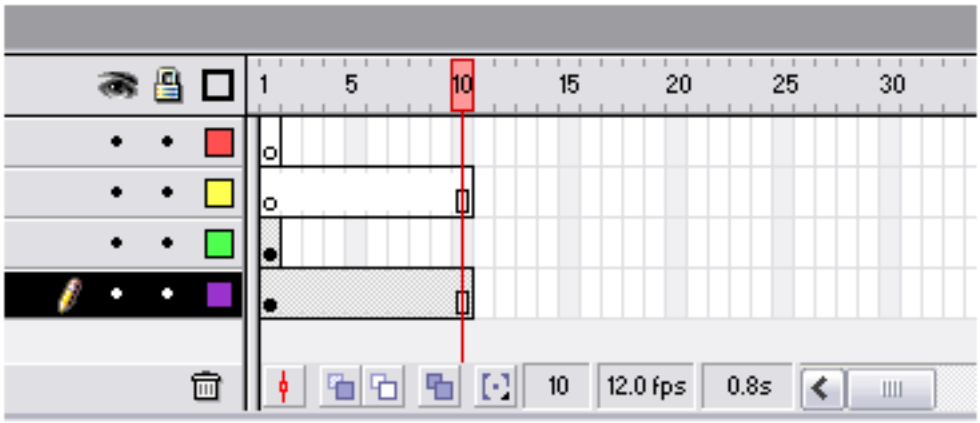
Jak jsem již dříve uvedl, pohyb je jen iluze navozená rychle se přepínající skupinou statických snímků. Snímky jsou tedy ve Flashi rozmístěné do časové osy, která se přehrává určitou rychlostí (Frame Rate).

Základním stavebním kamenem je ***klíčový snímek*** (keyframe). Tento snímek může být buď prázdný (není v něm nic nakresleno) nebo plný (obsahuje nějakou grafiku)



Prázdný klíčový snímek

Klíčový snímek



Myslím, že z tohoto obrázku je více než patrné, jak vypadá prázdný a plný klíčový snímek. Klíčový snímek může být široký jen jako jeden snímek (1. a 3. vrstva shora), nebo může být roztažen do libovolné vzdálenosti na časové ose (2. a 4. vrstva shora na obrázku). Grafika ve 3. vrstvě shora tedy bude vidět jen 1/12 sekundy, zatímco grafiku ve 4. vrstvě uvidíme 10/12 sekundy

Přidávání a mazání snímků:

- F5 - přidat normální snímek (zvětšit klíčový 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

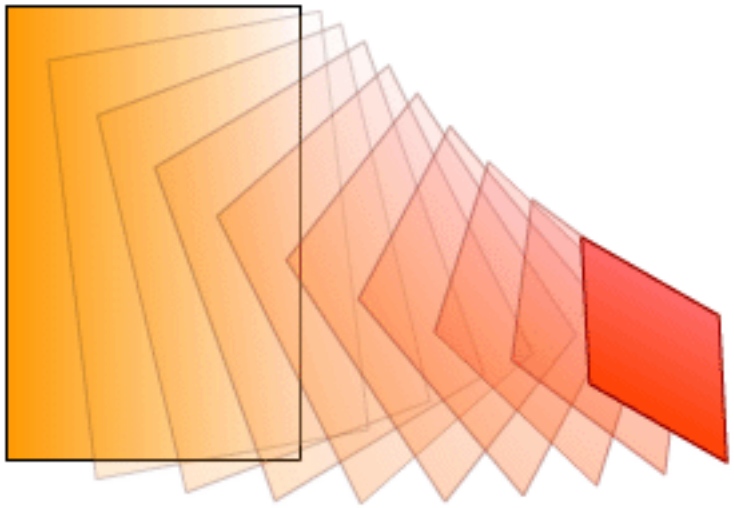
časové ose (i do jiné vrstvy)
- Pozor - výběrem snímku a stiskem DEL smažete jen obsah snímku (platí jen ve verzi MX).

Motion a Shape Tween

Takže nyní, jak vytvořit samotný pohyb. Asi vás teď napadlo, že je možné vytvořit animaci pomocí skupiny klíčových snímků (podobně jako animovaný GIF). Ano, je to možné, ale nedá se říct, že by to bylo nějak praktické.

Ve Flashi existují 2 způsoby, jak ***automaticky*** vykreslit změnu mezi dvěma klíčovými snímky:

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 (více snímků) 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***

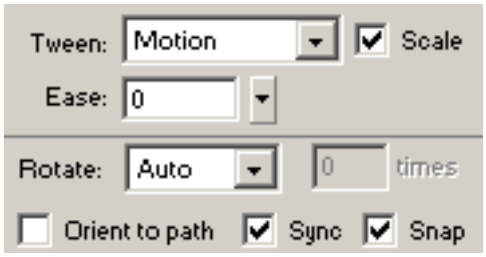
Instance symbolů ve druhém kl. snímku poté můžete upravovat (poloha, velikost, odstín, rotace, průhlednost, zkosení, ... viz zde)

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 ***Motion***

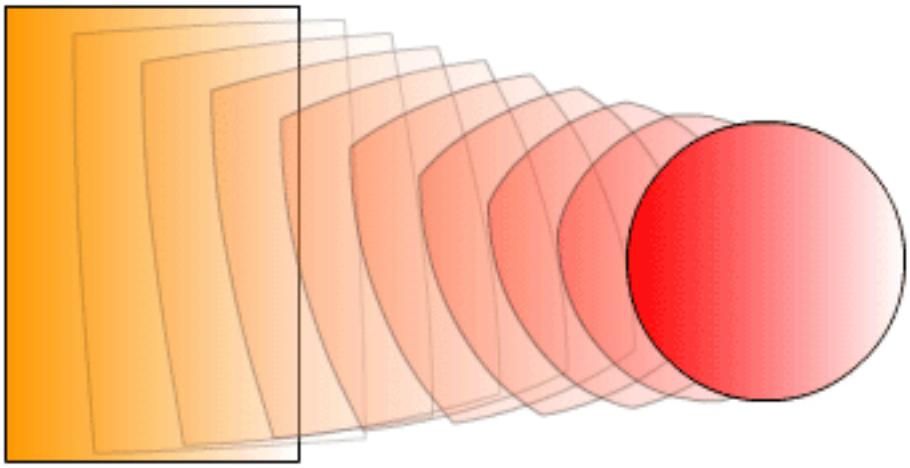
Pozn.: Pohyb lze vytvořit i tak, že do časové osy vložíte jen jeden kl. snímek, který roztáhnete na potřebnou velikost a do něj nakreslíte grafiku (nemusí to být instance). Poté kliknete pravým tlačítkem na první snímek a vyberete ***Create Motion Tween***. Nakreslená grafika se automaticky převede na symbol. Následně kliknete na poslední snímek klíčového snímku a upravte grafiku podle vašich představ (rotace, sklon, velikost ...) a je hotovo.

Na panelu Properties je možné nastavit ještě některé detaily pohybu:



- ***Ease***
  Zde můžete ovlivnit průběh rychlosti pohybu - zpomalování nebo zrychlování
- ***Rotate***
  Pokud necháte na "Auto", bude se objekt otáčet podle koncového snímku. Můžete však přidat rotaci (o 360°) a to po směru, nebo proti směru hodinových ručiček (CW a CCW). Do kolonky vpravo poté zadejte kolikrát se má objekt otočit.
- ***Scale***
  Zaškrtnutím povolíte postupnou změnu velikosti.
- ***Orient to Path***
  Orientovat pohyb na trasu - viz "Pohyb po křivce"
- ***Sync***
  Příkaz Synchronizovat použijte pokud počet snímků v animované sekvenci uvnitř Movie Clipu není sudým násobkem počtu snímků v celkové pohybové sekvenci.
- ***Snap***
  Přichytit k trase pohybu - viz "Pohyb po křivce"

Shape Tween (Tvarové vykreslení)



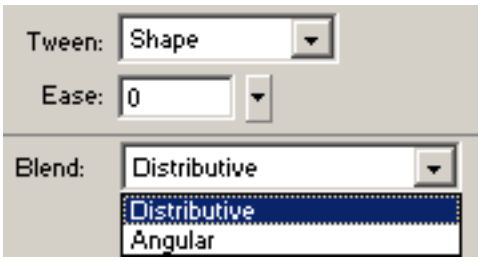
Zatímco u Motion Tweenu jsme mohli pohybovat pouze instancemi stejných symbolů, u Shape Tweenu je to právě naopak. Abychom mohli vytvořit tento pohyb, ***MUSÍ*** počáteční i koncový snímek obsahovat pouze ***čistou grafiku***.

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***

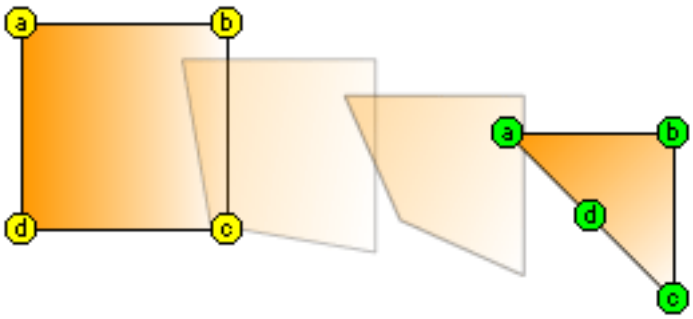
Na panelu Properties je možné nastavit:



- ***Ease***
  - viz dříve
- ***Blend***
  Způsob překreslování v přechodových snímcích:
    - *Distributive* - hladší a zaoblené tvary
    - *Angular* - jasné rohy, rovné čáry

Používání stop tvarů

Jistě jste se setkali s problémem, že při překreslování objektu probíhá přeskupování křivek nevhodným způsobem. Při použití Shape Tweenu je možné určit body, které by měly sobě odpovídat v počátečních a koncových tvarech.



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.

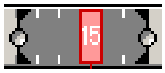
Editace a zobrazení více snímků najednou

Za normálního stavu jde ve Flashi editovat jen jeden snímek. Pokud však chcete například posunout, nebo otočit celou pohybovou sekvenci narazíte na zdánlivě neřešitelný problém. Je sice možné vybrat na časové ose více snímků, ale vy potřebujete vybrat objekty nikoliv snímky.

Pro tyto účely je ve Flashi tzv. ***Onion Skin*** (průsvitný papír)

Ovládá se těmito tlačítky:

- ***Zapnutí průsvitného režimu (Start Onion Skin)***
  Po zmáčknutí se na časové ose objeví rozsah zobrazovaných snímků



  Teď můžete editovat všechny objekty ve vyznačených snímcích
- ***Zobrazení vykreslovaných (tween) snímků*** (Onion Skin).
  Pokud je zapnut Onion Skin, jsou zobrazeny jen klíčové snímky. Zapnutím této volby zobrazíte i vykreslované (tween) snímky.
  Tyto snímky budou zobrazeny průsvitně
- ***Zobrazení vykreslovaných (tween) snímků jen čarami*** (Onion skin Outlines)
  Stejné, jako předchozí, jen s tím rozdílem, že jsou zobrazeny jen obrysy tween snímků.
- ***Změna rozsahu zobr. snímků*** (Modify Onion Markers)
  Zde je možné nastavit stálé zobrazování rozsahu snímků, velikost rozsahu, atd…

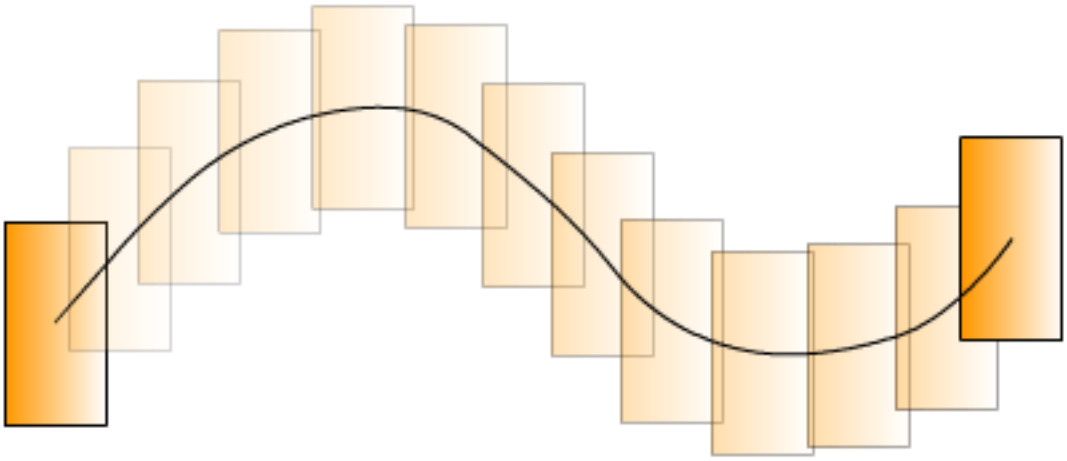
Pohyb po křivce, maska

Toto jsou dva speciální případy Motion Tweenu, kdy je kromě klasické vrstvy použita ještě jedna pomocná.

Pohyb po křivce

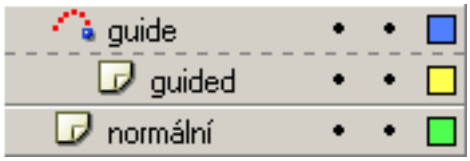
Pokud vytvoříte Motion Tween, bude vždy rozdíl polohy mezi počátečním a koncovým klíčovým snímkem vyplněn pohybem po přímce.

Lze to však udělat i jinak. Pohyb může kopírovat libovolnou trasu. K tomu nám pomůže tzv. vodící vrstva (Guide)

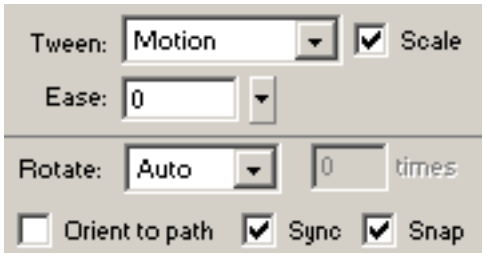


Postup:

- Vytvořte klasickou vrstvu
- Klikněte pravým tl. na vrstvu a zvolte "***Properties***" (lze použít i dvojklik). Poté zvolte Type - ***Guide***
- Poté vytvořte další vrstvu, která ***MUSÍ*** být umístěna pod vodící vrstvou a stejně, jako v předchozím případě otevřete okno "***Properties***", ale zvolte Type - ***Guided***
- Pokud jste všechno udělali správně, mělo by to vypadat takto:



- Do vodící vrstvy (guide) nakreslete čáru
- Ve vrstvě níže (guided) vytvořte klasický Motion Tween, ale v počátečním a koncovém klíčovém snímku uchopte instance za středy a umístěte je na konce křivky (měly by se samy uchopit)
- Případně můžete nastavit na panelu "Properties":



  - Orient to path - objekt se bude natáčet podle tvaru křivky
  - Snap - automatické uchopení ke křivce

- A je hotovo.

Maska

Toto je druhý speciální způsob animace, kdy horní vrstva (mask) obsahuje objekt nebo instanci (musí obsahovat nějakou výplň) a pod ní je druhá vrstva (masked), která obsahuje nějakou grafiku. V konečném důsledku je vidět jen ta část dolní vrstvy, která se ***překrývá*** s objektem v horní vrstvě.

Posup:

- Podobně, jako u pohybu po křivce vytvoříme 2 vrstvy (mask a masked)
- Do horní (mask) vložíme výplň a můžeme vytvořit motion nebo shape tween
- Do dolní (masked) vložíme libovolnou grafiku (může být i pohyblivá)
- Mělo by to vypadat takto:

- A je hotovo

Problémy:

Ve verzi MX (6) může maska obsahovat MovieClip. V našem případě kruh pohybující se úhlopříčně ze shora dolů. Ve verzi 5 (a nižší) to bohužel nefunguje (je nutné vytvořit MotionTween přímo v maskovací vrstvě).

Další problém je spíš bug. Flash má potíže s maskováním systémového písma (tj. písmo, které není vloženo do animace a načítá se ze systému uživatele - poznáte jej tak, že ne něm není potlačeno roztřepení). Takové písmo se v masce prostě nezobrazí. Tím pádem se také nezobrazí komponenty, které takové písmo obsahují. Nemyslím si, že by to byl záměr - spíš jde o neošetřenou chybu přehrávače.

Ve verzi MX2004 (7) ja tato chyba opravena.

Vložení animace do HTML

Takže už máme hotovou animaci a chceme ji upravit tak, aby se dala pohodlně sledovat. Připomenu jen, že je rozdíl mezi pracovní a prohlížecí verzí animace. Ta první má koncovku FLA a druhá SWF. Zatímco soubor FLA slouží k vytváření a úpravám animace, ten druhý je určen jen k prohlížení. Je tedy nutné exportovat animaci ve formátu FLA do formátu SWF. Pozor! - tato akce je ***NEVRATNÁ!!*** Ze SWF animace NIKDY nezískáte původní soubor FLA. Doporučuji tedy pracovní verzi FLA nemazat.

Test animace

V pracovním prostředí Flashe si můžete prohlédnout pohyb animace [Enter], ale pokud animace obsahuje nějaké pokročilejší prvky (Tlačítka, Actionscript), nebude fungovat správně.

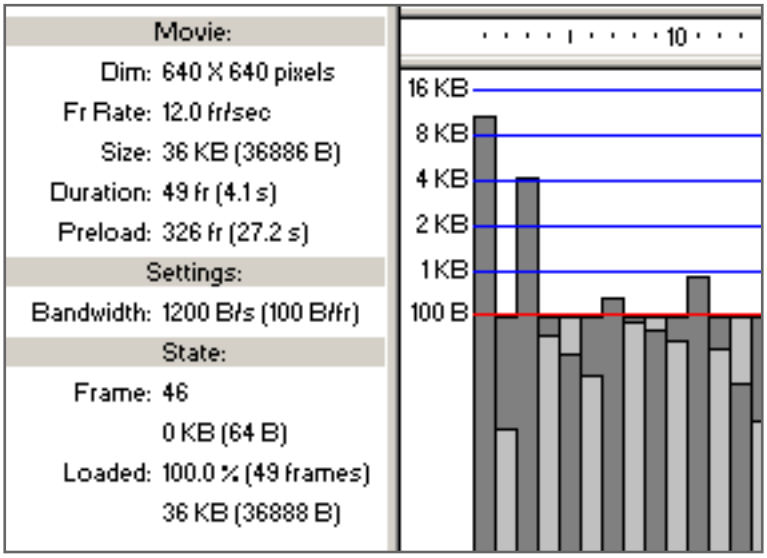
Řešením by bylo exportovat animaci do SWF a v externím prohlížeči ji shlédnout. Lze to však udělat mnohem praktičtěji. Slouží k tomu příkaz "***Test Movie***" v nabídce "***Control***".

Takto si můžete prohlédnout animaci tak, jak bude opravdu vypadat.

Bandwidth profiler

Funkce "***Test Movie***" umožňuje nejen prohlížet animaci, ale i zjistit, kolik dat se musí načíst v každém snímku.

Právě k tomuto účelu slouží Bandwidth profiler. Zobrazíte ho volbou v menu "***View***"



Tento sloupcový graf znázorňuje objem načítaných dat v jednotlivých snímcích. Jakmile sloupec překročí červeně vyznačenou mez, dojde ke zpomalení framerate, což je samozřejmě nežádoucí. První snímek bývá většinou největší - v něm se načítají všechny symboly v knihovně (Library). Aby se zabránilo počátečnímu trhání animace, je nutné použít tzv. "Preloader"

Pokud přehráváte animaci z disku, nebude většinou docházet k žádným problémům. Jiná situace ovšem nastane při prohlížení na internetu pomocí pomalé dial-up linky. Pro zkoušku pomalého načítání v menu "***View***" existuje volba "***Show streaming***". Po zaškrtnutí této volby se bude animace chovat, jako kdyby byla načítaná z webu. Rychlost tohoto virtuálního připojení můžete ovlivnit v menu "Debug" (doporučuji 14.4 kbps)

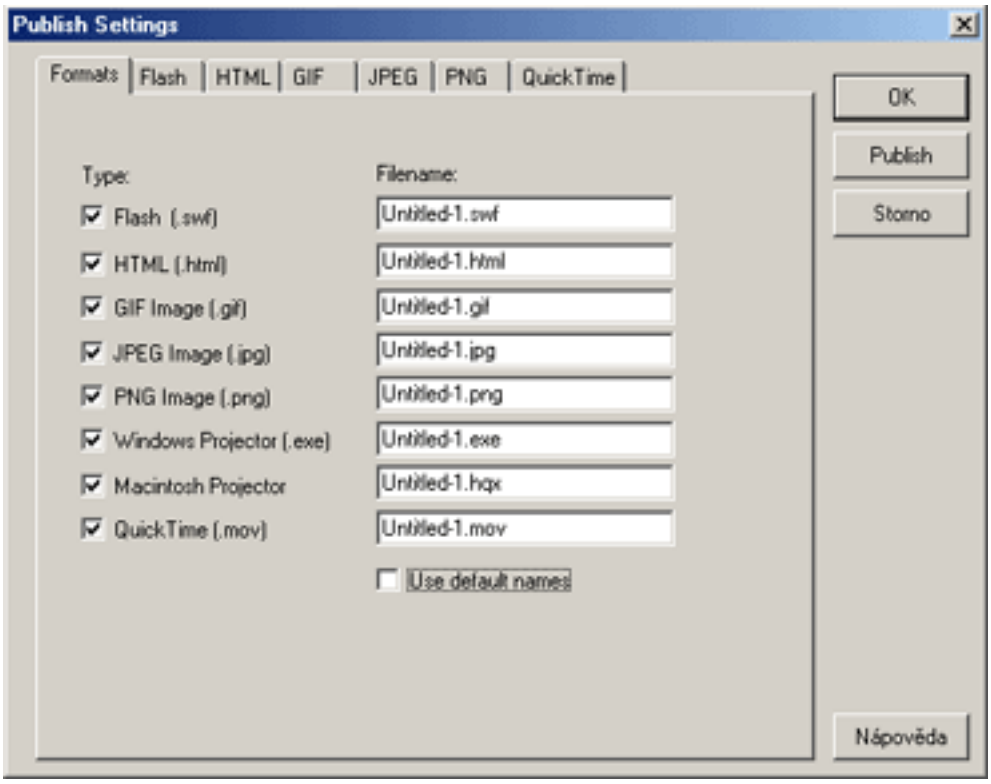
Exportování

Pokud máte animaci hotovu, můžete ji exportovat do formátu SWF (nebo do 14 jiných formátů, které však nebudou umět zobrazit některé funkce) volbou "***Export Movie***" v menu "***File***". Před exportem ještě můžete upřesnit některá nastavení (viz níže)

Publikování

Publikování je prakticky totéž co exportování, jen je v něm možno zároveň exportovat animaci do více druhů souborů

K nastavení publikování slouží volba "***Publish Settings***" v menu "***File***" Zde máte hned 8 možnosti, jak animaci publikovat. Ke každé zaškrtnuté položce se objeví nová kartička sloužící k podrobnému nastavení parametrů. Po nastavení potřebných parametrů stiskněte "OK" pro uložení nastavení nebo "Publish" pro okamžité publikování. Popíši tady dva nejdůležitější exportní formáty:



Flash [SWF]

Klasický flash soubor, který můžete přehrát buď v prohlížeči nebo ve Flash Playeru

- ***Version*** - zde nastavte v jaké verzi chcete animaci exportovat (pokud budete animaci exportovat do nižších verzí, mohou některé funkce dělat problémy.
- ***Load Order***
  pořadí, ve kterém Flash nahraje vrstvy animace pro zobrazení prvního snímku: Bottom Up (Zdola nahoru) nebo Top Down (Shora dolů).
- ***Options***

- *Generate Size Report* = generovat záznam o velikosti animace
- *Protect from import* = zakázat otevření SWF animace jako pracovní FLA soubor (je možné zadat heslo do pole "Password")
- *Omit Trace Actions* = způsobí, že Flash ignoruje akci Trace v aktuální animaci a zabrání oknu Output v otevření a zobrazení komentářů
- *Debugging Permitted* = aktivuje Debugger a umožní ladění animace Flash na dálku (je možné zadat heslo do pole "Password")
- *Compress Movie* = nová funkce, která funguje jen ve verzi 6. Zatrhnutím této volby částečně zmenšíte velikost výsledného souboru.

- ***JPEG Quality***
  kvalita vložených JPEG rastrů
- ***Audio Stream/Event***
  nastaví kvalitu průběžných/událostních zvuků
- ***Override Sound Settings***
  globálně přenastaví vlastnosti zvuků

HTML dokument

Pokud zaškrtnete volbu "***HTML Document***" na kartičce "***Formats***" zatrhne se automaticky i export do SWF. Samotná HTML stránka totiž sama o sobě nemůže obsahovat žádné objekty.

- ***Template***
  Zde zvolíte podle které šablony se má HTML stránka vytvořit
  - *Detect for Flash 3-6* = stránka ještě před startem animace zkontroluje přítomnost aktuálního Flash pluginu v prohlížeči
  - *Flash only* = klasická stránka s vloženou animací

- ***Dimesions***
  Zde můžete určit v jakých jednotkách má mít animace definované rozměry (procenta, pixely nebo neměnit rozměr)
- ***Width, height***
  upravení rozměrů animace (výška, šířka)
- ***Playback***
  Přehrávání
  - *Paused at start* = na začátku zastaveno
  - *Loop* = smyčka
  - *Display Menu* = po kliknutí pravým tl. zobrazovat místní nabídku
  - *Device Font* = používat systémové fonty

- ***Quality***
  Kvalita zobrazení - neovlivňuje velikost animace, ale náročnost na výkon počítače. (snížením kvality se vypne antialiasing a průhlednost objektů)
- ***Window***
  - *Window* = přehraje animaci Přehrávače Flash ve vlastním obdélníkovém okně na webové stránce pro rychlejší animaci
  - *Opaque Windowless* = přesune prvky za animaci Flash (například s dynamickým HTML), aby bylo zabráněno jejich zobrazení přes animaci
  - *Transparent Windowless* = zobrazí pozadí HTML stránky, do které je vložena animace, přes všechny průhledné oblasti animace,ale může animaci zpomalit

- ***HTML Alignment***
  Zarovnání animace
- ***Scale***
  Zde můžete určit, zda se mají objekty přizpůsobovat velikosti okna
- ***Flash Alignment***
  Zarovnání objektů animaci v rámci samotného okna animace.
- ***Show Warning Messages***
  Zobrazovat chybové hlášky při exportu

HTML výstup

Následující HTML kódy Flash vygeneruje při publikování. Abyste si nemysleli, že jde o nějakou černou magii, uvádím stručný přehled.
Flash animace je autonomní objekt, který je přehráván pomocí zásuvného modulu (plug-in). Jsou dvě možnosti jak jej umístit na HTML stránku:

- ***Pomocí tagu EMBED:***

  Embed anglicky znamená "umístit" nebo "pevně vložit". Syntaxe je tato:

```
<EMBED src="animace.swf"
play="false"
loop="false"
quality="high"
scale="noborder"
devicefont="true"
bgcolor="#FFFFFF"
width="550"
height="400"
wmode="opaque"
type="application/x-shockwave-flash"
pluginspage="http://www.macromedia.com/go/getflashplayer">
</EMBED>
```

  EMBED je však v novějších specifikacích (X)HTML nedoporučován a tak se dnes používá novější tag OBJECT

- ***Pomocí tagu Object***

  OBJECT je novější a má víc možností použití, než starší EMBED. Pokud však chcete dodržovat zpětnou kompatibilitu se staršími verzemi prohlížečů, měli byste používat raději EMBED (nebo oboje - EMBED jako alternativní obsah tagu OBJECT)

```
<OBJECT classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000"
codebase="http://download.macromedia.com
/pub/shockwave/cabs/flash/swflash.cab#version=6,0,0,0"
WIDTH="550" HEIGHT="400">

<PARAM NAME="movie" VALUE="animace.swf">
  <PARAM NAME="play" VALUE="false">
  <PARAM NAME="loop" VALUE="false">
```

```
  <PARAM NAME="quality" VALUE="high">
  <PARAM NAME="scale" VALUE="noborder">
  <PARAM NAME="devicefont" VALUE="true">
  <PARAM NAME="bgcolor" VALUE="#FFFFFF">
  <PARAM NAME="wmode" VALUE="opaque">

  Zde je alternativní obsah - často varianta s EMBED
</OBJECT>
```

* tučně vyznačené části jsou nezbytné pro zobrazení animace, ostatní jsou volitelné

Problémy v Mozille

Při vkládání Flash animace pomocí tagu ***OBJECT*** je použito ***ActiveX*** komponenty Flash, která je definována atributem ***ClassID***. Prohlížeče MSIE a Opera podporují ActiveX, takže tento zápis chápou a animaci zobrazí. Mozilla a prohlížeče pro Linux a MacOS ve standardní podobě ActiveX nemají a tak atribut ClassID ignorují a animaci nezobrazí.

Jedno z řešení tohoto problému je použití tagu ***EMBED***, který je ale zastaralý a tedy nevalidní podle nových (X)HTML norem.

Pokud tedy chceme použít OBJECT, musíme místo ActiveX komponenty zavolat přímo ***Flash plugin***:

```
<OBJECT data="animace.swf" type="application/x-shockwave-flash"></OBJECT>
```

Nyní se Flash zobrazí v Mozille, ale ne v IE. IE totiž nezná atribut ***data*** a tak musíme ještě definovat cestu k souboru klasicky tagem `<PARAM name="movie"...`

```
<OBJECT data="animace.swf" type="application/x-shockwave-flash">
<PARAM name="movie" value="animace.swf">
...
</OBJECT>
```

Teď se Flash zobrazí všude, ale chyba v IE způsobuje, že se zobrazená animace neuloží do cache prohlížeče (po reloadu se bude stahovat znova).

Vyřešit se to dá nalinkováním malého SWF, které teprve načte náš velký SWF:

- Vytvořte si prázdnou Flash animaci a do prvního snímku dejte:

  ```
  loadMovie(path, "_root");
  ```

  tento příkaz načte soubor definovaný v proměnné "***path***"
- Tento pomocný SWF nazvěte třeba ***container.swf*** a nahrejte na server spolu se samotnou animací
- Zápis pak bude vypadat následovně:

  ```
  <OBJECT data="animace.swf" type="application/x-shockwave-flash">
  ```

```
<PARAM name="movie" value="container.swf?path=animace.swf">
…
</OBJECT>
```

Za informace o tomto problému a způsobu řešení děkuji týmu Czilla.

Co je nového ve verzi 2004

Jak jste již asi zaregistrovali, v září roku 2003 vyšla nová verze Flashe s názvem Flash MX 2004 profesional. Jak již sám název napovídá, je to spíš upgrade té předchozí.

V následujícím textu budu popisovat ty nejpatrnější novinky, takže to bude taková všehochuť. Zatím nebudu zacházet do úplných podrobností - těm budu věnovat některé nové kapitoly.

Po spuštění

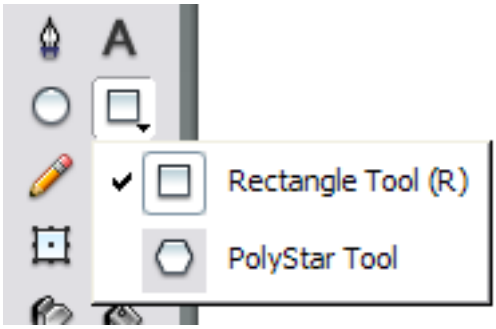
Po spuštění flashe se objeví uvítací ubrazovka s jakýmsi rozcestníkem. V podstatě nás asi nejvíce bude zajímat položka "otevřít" a "nový dokument". Dále jsou zde různé templaty a typy souborů, které je možné vytvořit. Z těch důležitých je to třeba ***externí ActionScript soubor***.

Podle mého názoru je tato sranda vcelku k ničemu. Je to takový hloupý trend pro hloupé uživatele (kterými programáoři Flashe IMHO nejsou) a proto doporučuji vypnout.

Interface

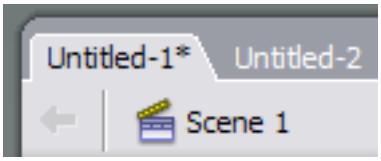
Interface, nebo chcete-li rozhraní, ve verzi 2004 prošlo výraznou omlazovací kůrou.
Na první pohled je jistě patrná větší čistota grafiky a hloubka barev.

Panel nástrojů zůstal prakticky nezměněn, jen pod klasický obdélník přibyl ještě tzv. "PolyStar", což není nic jiného, než mnohoúhelník.



Jen dodám, že po výběru nástroje se na panelu "Properties" objeví tlačítko "Options", kde je možné nastavit parametry mnohoúhelníku

Další pozitivní změna je v uspořádání otevřených dokumentů do přehledných záložek a s tím spojené přemístění panelu, zobrazujícího hierarchii instanci, nad časovou osu.



Snad bych jen vytknul nestřídmost tvůrců při zacházením s místem. Mezi jednotlivými segmenty rozhranní je hodně

nevyužitého místa, které pak na menších monitorech docela schází

Čeština

V české historii došlo k různým úpravám písma. Asi nejznatelnější bylo zavedení diakritického pravopisu namísto spřežkového. Ve škole nás učili, že se tak výrazně zjednodušilo psaní... Vziklo však 28 nestandardních znaků. Dnešní doba jasně ukazuje, že jakýkoliv odklon od standardů (v komunikaci obzvláště) znamená problém. Jaký problém? Tento: *P?íli? ?lu?ouøk? k?è ?pel ?abelsk? ?dy*.

Verze MX (6) slibovala rozlousknutí tohoto problému, a sice nasazením všeléku jménem Unicode. Nakolik se jí to povedlo, to nechť laskavý čtenář zhodnotí sám. Bez úprav registrů a jiné alchymie nebylo možné vpravit háčky a čárky tam, kde byly potřeba. A používání systémových písem, to už bylo čisté Woodoo.

Verze MX 2004 (7) udělala za tím vším tlustou čáru. Nyní už opravdu fungují české fonty bez jakýchkoliv problémů.

Antialiasing písma

Další noční můra, kterou verze 2004 odsouvá do nenávratna

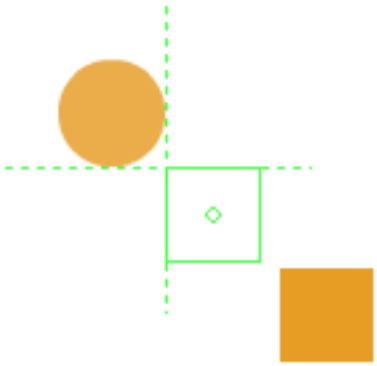
Dříve bylo vytvoření malého čitelného textu téměř nemožné. Pokud jsme totiž použili vkádané fonty (embed), došlo k potlačení roztřepení křivek fontu a tím pádem k jeho naprostému rozmazání. Autoři se snažili tuto situaci řešit systémovými písmy, které ale potom dělaly problémy s češtinou, nebo používali tzv "pixel fonty", které byly vytvořeny už hranatě, ale konečný výsledek často nebyl odpovídající námaze.

Flash MX 2004 nyní umožňuje vložit písma v pixelové podobě a umístit je tak, aby, tak říkajíc, nebylo co vyhlazovat. Výsledkem je perfektně čitelné písmo s malými rozměry.

Trasování polohy a uchopení

K této funkci jsem přišel jak slepý k houslím a přiznám se, že jsem málem spadl pod stůl ;)

Jiste mi dáte za pravdu, že umístit cokoliv ve starších verzích Flashe vedle sebe, nebo na stejnou úroveň bylo běh na dlouhou trať. Teď je to jiné. Pokud přesouváte objekt někde vedle jiného, samy se vám nabídnou možnosti, jak mají vůči sobě oběkty být:



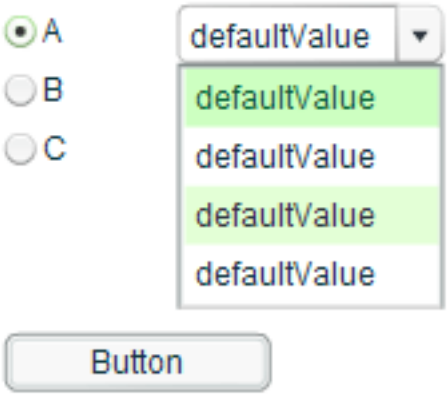
Maska

Jak jsem již psal patřičné kapitole, maska dělala občas problémy při zobrazování systémových fontů a komponentů. V nové verzi byla přepracována a teď už nedělá problémy

Komponenty

Komponenty již v této verzi nejsou jen UI, ale přibyly i typy pro práci s daty (Data Components) nebo s médii (Media Components). Samotné UI komponenty byly zcela přepracovány, jak po stránce vizuální, tak i po stránce skriptové. Z toho tedy jednoznačně vyplývá, že na přehrávačích verze 6 a starších se naše kolonky a tlačítka změní v rozsypaný čaj.

Jinak UI komponenty považuji za docela kvalitní. Z těch nových bych jmenoval například Preloader, vodorovné menu, svislé menu, načítač obrázků, kalendář, textová oblast i se scrollbarem, datová tabulka a další.



Efekty časové osy

Reklamní slogany na verzi 2004 slibovaly vylepšené efekty časové osy. Čekal jsem něco jako nový druh Tweenu. Bohužel nic nového nepřišlo. Byly pouze vytvořeni průvodci pro tvorbu přechodových Tween sekvencí. Proč to dělat jednoduše, když to jde i složitě, že? Jestli něco nemám rád, tak je to to, když ze mě někdo dělá debila. Popravdě si nedokážu představit rozumné použití těchto průvodců, protože nezjednodušují ani nezrychlují práci. Flash je IMHO taková první liga webdesignu a ten, kdo tyto průvodce potřebuje v ní nemá co dělat. Průvodce lze spustit z menu "insert" -> "Timeline effects"

ActionScript

AS byl docela slušně překopán, zejména v oblasti objektů a komponentů. První věc, které si uživatel všimne, je absence "Normal Mode" pro vkládání scriptů. Je tedy nutné vytvářet příkazy přímo v kódu. Dále také přibyly příkazy pro manipulaci se speciálními druhy animaci (Prezentace a Formulář)

Off Topic

Taky už vám ty X-ka lezou krkem. Mně teda jo! Jakoby co nemá v názvu X bylo mimo mísu...: Windows XP, Flash MX, Nvidia GeForce FX, Matrix, X-Files, DivX, Citrix, Linux, Unix, XHTML, XML, Triple X, X-Men, MacOS-X, X-Box, Active-X, Direct-X, Memorex, Xerox,...

Řízení animace

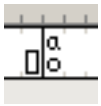
Seznámení s Actionscriptem

Animace lze vytvářet způsobem popsaným v předchozích kapitolách, ale výsledek je neměnný a stálý. Actionscript dovoluje přidat do animací interaktivitu a změnit ji třeba na přehledné menu, webovou stránku, zábavnou hru, nebo v animovaný formulář.

ActionScript je odvozen od JavaScriptu. Jelikož je ale Flash animace autonomní objekt zobrazovaný pomocí Plug-inu, je nutné počítat s tím, že nebudou fungovat objekty jako ***Document*** nebo ***Window***. Další důležitá vlastnost ActionScriptu je, že je ***case-unsensitive*** (na velikosti písmen v názvech příkazů tedy nezáleží).

Skript (seznam příkazů) je možno ve Flashi zacílit buď na:

- ***klíčový snímek v časové ose*** (i na prázdný)



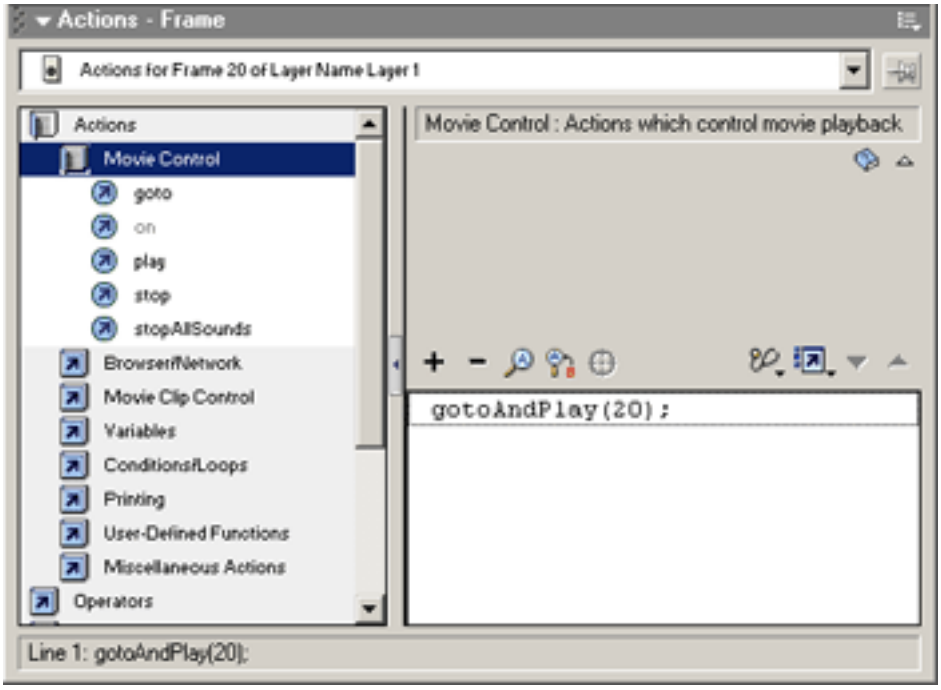
  Poznáte ho podle malého písmene ***a*** ve snímku
  - příkazy jsou vykonány v momentě zobrazení snímku

- ***objekt***:
  - tlačítko
  - movie clip

  - zde jsou příkazy vykonány po určité události (viz tabulka níže)

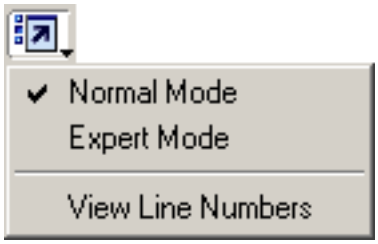
Skript můžete snímku (objektu) zadat tak, že ho vyberete a zmáčknete [F9], nebo kliknete pravým tl. myši a v nabídce vyberete "Actions".

Objeví se okno "***Actions***"



Okno "***Actions***" umí pracovat ve dvou základních režimech: ***Normal*** a ***Expert***

Zatímco v módu Normal se hodnoty příkazů určují pomocí přehledných formulářů, v Expert módu musíte vše psát ručně. Proto bych doporučil méně zkušeným uživatelům ponechat "Normal".



Příkaz vložíte jeho vybráním v seznamu vlevo a přetažením (nebo dvojklikem) do okna vpravo.

Příkazy definované ***snímku*** se provedou ***po zobrazení snímku*** v pořadí, v jakém jsou pod sebou. To je samozřejmě velmi zjednodušeně řečeno (existují různé podmínky a smyčky, které ovlivňují v jakém pořadí a zda vůbec budou příkazy vykonány, ale o tom až později).

Pokud je příkaz definován ***tlačítku*** nebo ***movieclipu*** provedou se příkazy po určité ***události***, kterou můžeme určit:

- ***Tlačítko - události***

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

celý kód by pak mohl vypadat třeba takto:

```
on (press) {
    gotoAndPlay(1);
}
```

Tento zápis znamená, že po ***kliknutí*** nad tlačítkem se provedou akce uvnitř složených závorek - v našem případě - "běž na snímek č. 1".

- ***MovieClip - události***

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

Základní příkazy

Pro začátek byste měli ovládat tyto velmi důležité příkazy:

- ***GoTo - běž na***
  Tento příkaz slouží k přechodu na jiný snímek (nebo scénu) a pokračovat v přehrávání, případně zastavit (GotoAndStop). Snad jen podotknu, že v roletovém menu "Type" lze zvolit specifikaci cíle jako číslo snímku, jeho popis, následující nebo předchozí snímek, nebo hodnotu proměnné.

```
gotoAndPlay("Scene 1", 12);
```

- ***Play & Stop***
  Slouží ke spuštění nebo zastavení animace.
  Bez parametrů

```
play();
stop();
```

- ***GetURL***
  Otevře HTML stránku (nebo soubor). Můžete nastavit ve kterém frame se má otevřít a jestli se mají odesílat hodnoty proměnných (GET, POST).

```
getURL("www.flash-help.wz.cz", "_blank", "POST");
```

Základní operátory

- ***Aritmetické***

přiřazení hodnoty	=	**a = 5** ("a" bude 5)
sčítání (spojení řetězců)	+	**10 + 5 = a** ("a" bude 15)
odčítání	-	**10 - 5 = a** ("a" bude 5)
násobení	*	**10 * 5 = a** ("a" bude 50)
dělení	/	**10 / 5 = a** ("a" bude 2)
zbytek po dělení	%	**12 % 5 = a** ("a" bude 2)
závorky	()	**10 * (5 + 8) = a** ("a" bude 130)
inkrement (zvýšení o 1)	++	a = 5; **a++** ("a" bude 6)
-> ***post***-inkrementace		a = 5; **b = a++** ("b" bude 5; "a" bude 6)
-> ***pre***-inkrementace		a = 5; **b = ++a** ("a" i "b" bude 6)
dekrement (snížení o 1)	--	a = 5; **a--** ("a" bude 4)
-> ***post***-dekrementace		a = 5; **b = a--** ("b" bude 5; "a" bude 4)
-> ***pre***-dekrementace		a = 5; **b = --a** ("a" i "b" bude 4)
přičtení	+=	a = 5; **a+=3** ("a" bude 8)
odečtení	-=	a = 5; **a-=3** ("a" bude 2)
vynásobení	*=	a = 5; **a*=3** ("a" bude 15)
podělení	/=	a = 5; **a/=3** ("a" bude 1,6)

- ***Srovnávací - vrací TRUE/FALSE (pravda/nepravda)***

menší	<	a = 5; b = 6; **a < b** (true)
větší	>	a = 5; b = 6; **a > b** (false)

menší nebo roven	<=	`a = 5; b = 6;` **`a <= b`** `(true)`
větší nebo roven	>=	`a = 5; b = 6;` **`a >= b`** `(false)`
rovnost	==	`a = 3; b = 3;` **`a == b`** `(true)`
striktní rovnost	===	`a = 3; b = "3"` **`a === b`** `(false!)`
nerovnost	!=	`a = 5; b = 6;` **`a != b`** `(true)`
striktní nerovnost	!==	`a = 3; b = "3"` **`a !== b`** `(true!)`

- ***Logické***

a zároveň (AND)	&&	`a = 5; b = 6;` **`a != b && a<b`** `(true)`
nebo (OR)	||	`a = 5; b = 6;` **`a != b || a>b`** `(true)`
negace (NOT)	!	`a = 5;` **`!(a == 5)`** `(false)`

Adresování příkazů

Některé příkazy (např. GoTo, Play, atd...) a všechny proměnné se vztahují jen k vlastní časové ose.

K adresování do jiné časové osy (v MovieClipu) je nutné použít tzv. dot syntaxi (dot = tečka).

pokud tedy budu chtít z hlavní časové osy definovat proměnnou s názvem "pozdrav" v movie clipu pojmenovaném "prvni" bude zápis vypadat takto:

```
prvni.pozdrav = "ahoj";
```

Tento zápis je ***relativní***. Takto definovaná proměnná by fungovala i kdyby se celá situace odehrávala uvnitř jiného movieclipu (namísto na hlavní časové ose).

Podobným způsobem lze definovat proměnnou "pozdrav" z moveclipu "prvni" tak aby platil na hlavní časové ose:

```
_parent.pozdrav = "ahoj";
```

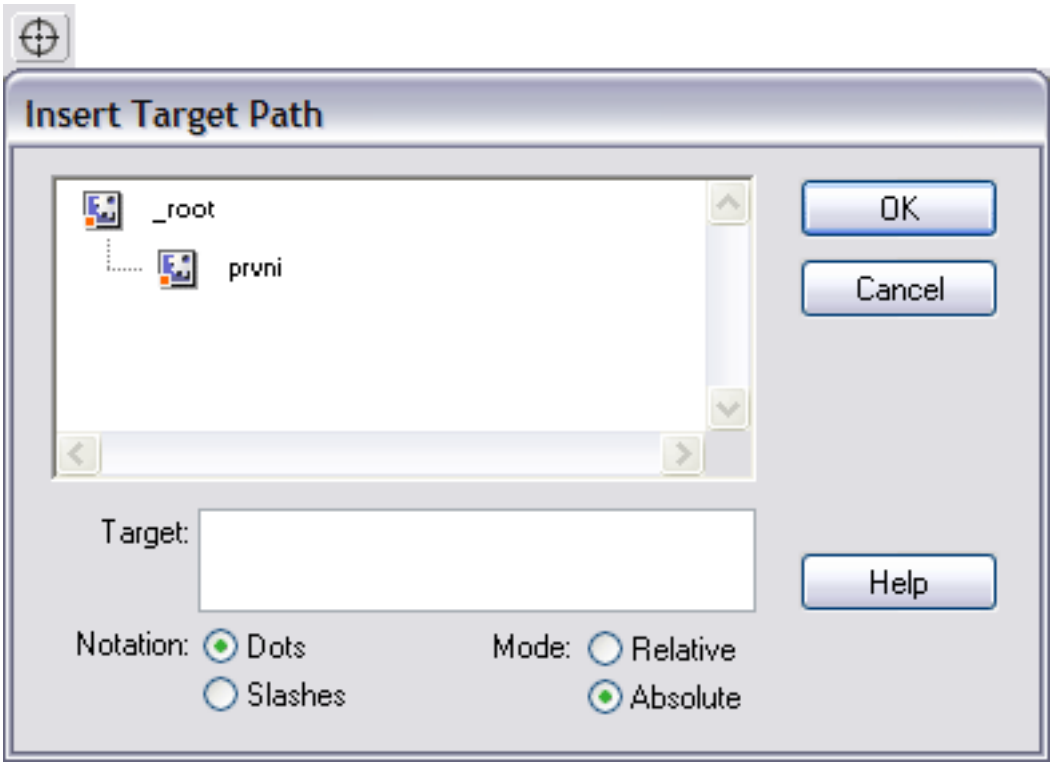
Alias "***_parent***" znamená "rodič" neboli "o úroveň výš".
Pozn.: V relativní definici existuje ještě jeden alias - "this", který znamená "tento". V podstatě není nutné ho použít, protože příkazy s nedefinovanou cestou se implicitně vztahují ke své ose.

Druhá možnost jak definovat cestu je ***absolutní***. Je vždy nutné použít alias "_root" (root=kořen)

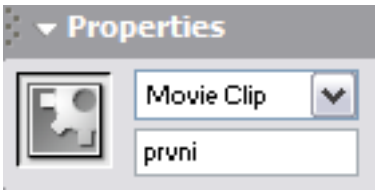
```
_root.prvni.pozdrav = "ahoj";
```

Tento zápis bude odkazovat na stejnou pozici ať už bude definován odkudkoliv.

Pro pohodlné definování cesty slouží tlačítko se symbolem terče:



Důležité!
Abychom mohli jakkoliv scriptově ovlivňovat instanci MovieClipu musíme ji pojmenovat. Jméno instance je ***nezávislé*** na jménu symbolu.



Pokud bychom chtěli definovat velké množství příkazů adresovaných do jednoho MovieClipu, byl by to asi pěkný záhul. Určitým usnadněním je příkaz ***with()***

```
with (_root.prvni.druhy){
    a = 20;
    b = 150;
    c = 150;
}
```

Je totéž jako:

```
_root.prvni.druhy.a = 20;
_root.prvni.druhy.b = 150;
_root.prvni.druhy.c = 150;
```

Proměnné

Co je to proměnná?

Proměnnou si lze představit jako krabici s určitým názvem ve které je uložena určitá informace. Hodnotu proměnné můžeme měnit.

Pozn.: Ve Flashi se před proměnnou nepíše "$" (jako v PHP nebo JavaScriptu). Pokud napíšete nečíselný výraz bez uvozovek, je chápán jako proměnná. Jen dodám, že název proměnné nesmí začínat číslem a nesmí obsahovat nepovolené znaky: / \ : * ? " < >.

Globální a lokální proměnné

- ***Lokální proměnná***
  Existuje jen ve svém bloku kódu (mezi složenými závorkami) - například uvnitř funkce nebo smyčky. Dost často se používá pro kontrolu opakování smyčky. Když jsou příkazy vykonány, přestane proměnná existovat. Její deklarace vypadá takto:

```
var pozdrav = "ahoj"
```

- ***Globální proměnná***
  Existuje ve všech časových osách. vložíte ji příkazem SetVariable. Zápis vypadá takto:

```
pozdrav = "ahoj"
```

Obsah proměnné

Obsah proměnné může mít 3 druhy:

- ***Text***
  Textový obsah proměnné je nutno uzavřít do uvozovek. Může obsahovat i číslice, které jsou však chápány jako text.

```
pozdrav = "ahoj"
pozdrav = "479087"
```

- ***Číslo***
  Číselný obsah proměnné je nutno definovat bez uvozovek. (u pole "value" zaškrtněte "Expression")

```
cislo = 4533
```

- ***Booleovská funkce***
  Booleovská funkce se také definuje bez uvozovek a může nabývat jen dvou hodnot: ***true*** (pravda) a ***false*** (nepravda). Stejný význam má také bitový zápis ***1*** a ***0***.

```
test = true
```

Je velmi důležité správně definovat obsah proměnné, protože to pak má velký vliv na další zpracování. Malá ukázka:

- ```
  a = 15;
  b = 30;
  vysledek = a+b;
  ```

  - proměnná "vysledek" bude mít hodnotu "45"

- ```
  a = "15"
  b = "30"
  vysledek = a+b;
  ```

  - proměnná "vysledek" nyní bude mít hodnotu "1530" !!

Při zpracovávání proměnných se Flash snaží automaticky přizpůsobit typ proměnné dané operaci. Pokud například sčítáte textový řetězec s číslem výsledkem bude vždy textový řetězec.

Někdy je však nutné přesně definovat, jak má Flash chápat obsah proměnné. Pro tyto účely existují funkce ***Number()***, ***String ()*** a ***Boolean()***

```
a = 38;
b = 15;
vysledek = Number(a) + Number(b);
```

- proměnná "vysledek" bude mít hodnotu "53"

```
a = 38;
b = 15;
vysledek = String(a) + String(b);
```

- proměnná "vysledek" bude mít hodnotu "3815"

Použití proměnných

Proměnné je možno použít prakticky kdekoliv místo samotné hodnoty. Příklad:

```
snimek = 38;
GotoAndPlay(snimek);
```

- animace přejde na snímek 38

Pokud chceme definovat obsah jedné proměnné jako obsah druhé, uvedeme její hodnotu jako název druhé proměnné a neuzavíráme jej do uvozovek (zaškrtnout "Expression")

```
a = "ahoj";
b = a;
```

- proměnná "b" bude mít teďtaké hodnotu "ahoj"

Je možné používat všechny matematické operátory pro manipulaci s proměnnými. Opět platí, že názvy proměnných se neuzavírají do uvozovek. Pokud chceme k proměnné například přičíst nějaký textový řetězec, uzavřeme jej do uvozovek (aby nebyl chápán jako proměnná):

```
a = "nazdar"
b = a + "ahoj"
```

- proměnná "b" bude mít teďhodnotu "nazdarahoj"

Formulář

Asi jste si všimli, že při vkládání textu do animace je možno namísto statického vložit i dynamic a input text.

Input a ***Dynamic*** text pracují prakticky na stejném principu. Oba dva musí mít v poli "***Var:***" definovaný název proměnné, který potom ***zobrazují***. Hlavní rozdíl těchto dvou typů textu je, že Dynamic text hodnotu pouze zobrazuje a Input text ji dovoluje i měnit (jako ve webovém formuláři).

Položka "***Maximum Characters***" určuje kolik znaků formulář dovoluje zobrazit.

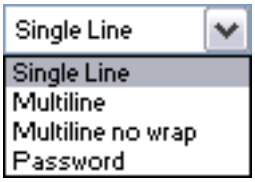
Tlačítko "***Characters***" nastavuje jaké znaky mají být "přibaleny" do animace, nebo zda mají být použité fonty aktuálního počítače.

Tlačítko "***Selectable***" zapíná možnost vybrání textu kurzorem. (i input textu vždy aktivní).

Druhé tlačítko zapíná ***HTML formátování***. Pokud je tato volba zapnuta a obsah proměnné obsahuje HTML tagy ***<b><a><i><font color=... size=... face=...><p><u>*** je text zformátován podle podle nich. Ostatní tagy budou ignorovány.

Poslední tlačítko zapíná ***ohraničení*** a ***výplň*** textového pole.

Dále je možno určit zda má mít formulář jeden (single line) nebo více řádků (multiline), zda se nemají zalamovat (multiline no wrap), nebo zda má být psaný text nahrazen hvězdičkami (password).



Datové pole (Array)

Array (=pole, sada, seskupení, matice) by se dalo přirovnat k proměnným. Narozdíl od proměnných však neobsahuje jednu hodnotu, ale více separovaných hodnot, které jsou očíslovány (index).

Je potřeba si uvědomit, že Array je objekt a potřebuje tedy konstruktor (viz objekty). Malý příklad:

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

Zde jsme tedy definovali objekt Array s názvem "pole". Povšimněte si, že naše pole má 5 hodnot. Důležité je, že položky jsou číslovány (index) od nuly (0, 1, 2...). Pokud bychom tedy chtěli někde použít například 4. hodnotu (index = 3) vypadalo by to takto:

```
gotoAndPlay(pole[3]);
```

Animace by takto přešla na snímek 14

Hodnoty položek v poli jdou však také editovat:

```
pole[4] = "čau";
pole[8] = "nazdar"
```

První příkaz přepsal hodnotu ***poslední*** položky "***ahoj***" na hodnotu "***čau***". Druhý příkaz definoval 8. položku jako řetězec "nazdar".

Asi teď namítnete, že položky s indexy 5, 6 a 7 nejsou definovány. Právě naopak. Pokud vložíte v poli hodnotu do indexu, před kterým jsou ***nedefinované*** indexy, vyplní se tyto ***prázdnými řetězci*** (""). Počet prvků je tedy teď ***9***.

Délku pole (počet prvků) zjistíte vlastností `-pole.length`

Je také možné vytvořit pole s určitým počtem prázdných položek:

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

Metody objektu Array:

contact	spojuje pole dohromady a prvky seřadí za sebe: `pole1.contact(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")`

Vložení zvuku

Do Flash animace je možné vložit zvuk ve formátu ***MP3*** nebo ***WAV***. Pro webové použití doporučuji pouze MP3 (kvůli velikosti).

Abychom mohli kamkoliv umístit zvuk, je nutné jej nejprve importovat do knihovny (viz "Knihovna")

Existují dvě možnosti vkládání zvuků. Pro méně náročné použití se dá zvuk vložit přímo do snímku, pro složitější akce se dá importovat do Actionscript objektu Sound() a pak jej lze použít v jakékoliv situaci.

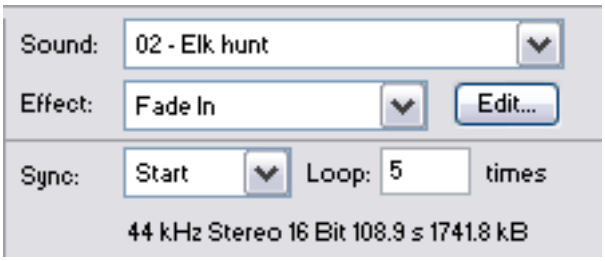
Přímé vložení

Zvuk lze vložit do snímku tak, že vyberete snímek a v okně "***Properties***" zvolíte z roletového menu "***Sound***" zvuk z knihovny.

Roleta "***Effect***" umožňuje nastavit efekty, jako např. zesilování, zeslabování a další spousta volovin. Pokud vám nebude škála efektů postačovat, můžete si nastavit pomocí tlačítka "***Edit***" svůj.

Roleta "***Sync***" umožňuje nastavit co přesně má zvuk udělat (zastavit, spustit, …).

Položka "***Loop***" znamená počet opakování (0 = neustále)



Pozn.: Přehrávání zvuku prakticky není omezeno polohou klíčových snímků na časové ose (zvuk zkrátka hraje dokud mu nevyprší počet opakování nebo dokud není zastaven)

Scriptové vložení

Tento způsob ozvučení využívá předdefinovaného objektu **`Sound()`**
- viz kapitola "Objekty - MOVIE"

Jako první krok je třeba nalinkovat zvuk. V knihovně klikněte pravým tl. na zvuk a zvolte "***Linkage***". Zde zaškrtněte "***Export for ActionScript***" a do pole "***Identifier***" zadejte nějaký název, například "*potlesk*"

Dále do ***prvního*** snímku animace definujte tyto příkazy (najdete je v seznamu příkazů Object - Sound):

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

nyní je možné kdekoliv v animaci použít pro spuštění resp. zastavení zvuku tyto příkazy:

```
zvuk.Start(15,8);*
```

```
zvuk.Stop();
```

* číslo 15 je tzv Offset, což je hodnota v sekundách o kterou se přehrávání posune od začátku. (v našem případě se zvuk začne přehrávat až od 15. sekundy)

číslo 8 je počet opakování (loop).

Objekt Sound() má spoustu dalších nastavení. Podrobnosti viz kapitola "Objekty - MOVIE"

Změna kvality

Pokud chcete před exportem animace změnit kvalitu a parametry zvuku, můžete tak učinit kliknutím na zvuk v knihovně pravým tl. a výběrem položky "Properties".

Tip

Zvuky vkládejte až nakonec. Při průběžném testování animace by se zbytečně prodlužovala doba exportu.

Podmínky a smyčky

Podmínky a smyčky umožňují ovlivňovat sled provádění příkazů na základě nějaké skutečnosti. Příkazy spojené s podmínkami a smyčkami lze vkládat v sekci Actions - Condition/Loops.

Podmínky

Podmínka je útvar, který kontroluje pravdivost nějaké skutečnosti a podle toho buď provede nebo neprovede příkazy napsané uvnitř (případně vykoná jiné). Nejlepší asi bude vysvětlit si vše na následujícím příkladu:

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

Tento zápis znamená, že ***pokud*** (if) proměnná "***vyhra***" bude rovna nule a ***zároveň*** proměnná "***hotovost***" bude menší než 100 provedou se příkazy uvnitř složených závorek (v našem případě "***běž na snímek 10***").

Pokud nebude splněna podmínka uvnitř kulatých závorek, nic se nestane a předchozí tři řádky jakoby neexistovaly.

Jiná možnost:

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

Zde nám přibylo ještě slůvko "***else***" (=jinak). Pokud není splněna podmínka, vykonají se příkazy uvnitř složených závorek za příkazem else (v našem případě "běž na snímek 1").

Ještě jiná možnost:

```
if (vyhra == 0 && hotovost < 100) {
   gotoAndPlay(10);
} else if (vyhra == 50) {
   gotoAndPlay(1);
}
```

Příkaz "***else if***" (=jinak pokud) je podobný příkazu "else", ale příkazy uvnitř se vykonají jen v případě, že podmínka "if" splněna nebude a naopak podmínka
"else if" splněna bude. Pokud tedy nebude splněna ani jedna podmínka, nic se nestane.

Smyčky

Smyčka také kontroluje pravdivost zapsaného výrazu. Pokud je zápis pravdivý, provedou se akce uvnitř smyčky, ale narozdíl od podmínky nepokračuje vykonávání ostatních skriptů, ale znovu se zjišťuje pravdivost výrazu. Pokud je výraz nepravda, pokračuje vykonávání příkazů za smyčkou.

Existuje víc druhů smyček:

- ***While***
  Klasický typ smyčky. Výraz je kontrolován na začátku a dokud je pravdivý jsou příkazy prováděny pořád dokola, pokud ne, běh scriptu pokračuje za smyčkou - viz příklad:

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

  Pokud je proměnná "i" větší než 0, opakuje se stále dokola příkaz, který kopíruje MovieClip a odečte od proměnné "i" jedničku. Tato smyčka se tedy bude opakovat pětkrát. Pokud podmínka přestane být pravdivá, bude pokračovat zpracovávání příkazů za smyčkou.
- ***Do...While***
  Úplně stejný princip jako v předchozím případě, jen z malým rozdílem. Pravdivost zapsaného výrazu se kontroluje ***až na konci*** smyčky. Jistě vám tedy došlo, že příkazy uvnitř smyčky se i v případě nepravdivosti výrazu aspoň jednou provedou. Viz příklad:

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

- ***For***
  Poslední typ smyčky ve Flashi. Tato smyčka je nejsložitější. Uvnitř kulatých závorek jsou definovány 3 výrazy (oddělené středníkem ";"). První výraz se vykoná na začátku prvního cyklu a dál už ne. Druhý výraz smyčka hodnotí a pokud je pravdivý (TRUE), smyčka bude běžet až do chvíle, kdy pravdivý být přestane, nebo bude přerušena. A nakonec třetí výraz. Ten je vykonán na konci každého cyklu:

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

  i = 4 ..... počáteční výraz - provede se před prvním cyklem (v našem případě definování proměnné i = 4)
  i > 0 ..... výraz, jehož pravdivost se kontroluje
  i-- ..... koncový příkaz - je vykonán na konci každého cyklu (v našem případě odečtení 1 od "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 se proměnná "***limit***" bude rovnat nule, bude se opakovat smyčka, uvnitř které je podmínka "***if***" a příkaz zvyšující proměnnou "***i***" o jedna. Pokud proměnná "***i***" dosáhne hodnoty 10, vykoná se příkaz ***break*** a smyčka se přeruší.

Problém

Předchozí příklady ve vás možná evokovaly pocit, že lze vytvořit smyčku, která by stále dokola kontrolovala nějakou skutečnost a podle toho prováděla nebo neprováděla akce:

```
while (skore > 0){
   buton01.enabled = true;
}
buton01.enabled = false;
```

Tento způsob použití smyčky sice není v rozporu se správnou syntaxí, ale v praxi je nepoužitelný. Je totiž nutné si uvědomit jednu důležitou skutečnost - všechny příkazy definované snímkům a objektům jsou provedeny po splnění události (MouseOver, KeyDown, ...) ***v jeden okamžik***! Pokud bychom tedy spustili animaci s předchozí smyčkou, bylo by provedeno v jeden moment nekonečně mnoho operací, což nelze. Abych to shrnul - každá smyčka musí mít ***konečný*** počet cyklů (ať už pomocí proměnné, nebo příkazu "For").

Switch (přepínač)

Switch je zvláštní typ elementu ovlivňujícího tok příkazů, který se objevil ve verzi MX (6). Do jisté míry by se dal přirovnat k podmínce ***IF***. Switch obsahuje několik bloků příkazů, které jsou pojmenovány (1, 2,..). Pokud switchi definujeme jako argument jméno bloku, vykoná příkazy uvnitř. Viz příklad:

```
abc = 2;

switch (abc) {
     case 1 :
          gotoAndPlay(1);
          break;

     case 2 :
          gotoAndPlay(5);
          break;

     case 3 :
          gotoAndPlay(10);
          break;

}
```

Zde je zde je definován argument jako proměnná "***abc***". Vložená hodnota se nyní porovná s názvy bloků a pokud se název shoduje s jedním z bloků, vykonají se příkazy uvnitř. V našem případě `gotoAndPlay(5);`. Následuje příkaz `break;`, který ukončí vykonávání ostatních příkazů - jinak by pokračoval příkaz `gotoAndPlay(10);`

Důležité! Aby se vykonaly příkazy v bloku kódu, musí být
striktně roven (strict equality "===") argument a název bloku kódu. Striktní rovnost je dodržena, když souhlasí jak hodnota, tak i typ proměnné: **`3 !== "3"`**

Názvy bloků nemusí být jenom čísla, může to být i textový řetězec (string) - viz příklad níže

Pokud nebude striktně souhlasit argument s názvem jednoho z bloků, bude pokračovat vykonávání příkazů za přepínačem. Pokud bychom však chtěli, aby se i při nerovnosti argumentu a názvu bloku vykonaly nějaké příkazy, musíme použít blok s názvem **`default:`**.

```
abc = "blablablabla";

switch (abc) {

     case "prvni":
               gotoAndPlay(1);
               break;

     case "druhy":
               gotoAndPlay(5);
               break;

     default:
               Stop();
               break;

}
```

Jak je vidno, argument nesouhlasí s žádným blokem a proto budou vykonány příkazy uvnitř bloku "***default"***- v našem případě `Stop();`

Funkce

Funkce je určitý blok kódu s přiřazeným názvem, který lze kdykoliv provést. Funkci si můžete představit jako "černou skříňku". Když je funkce volána, je jí přiřazen vstup (argumenty). Vykoná několik operací a pak generuje output (vrácenou hodnotu).

K definování funkce slouží příkaz ***function*** (Actions - User defined functions):

Jednoduchá funkce

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

Toto je nejjednodušší definice funkce (bez vstupních a výstupních argumentů). Při zavolání této funkce se jednoduše provedou příkazy uvnitř a šmitec.

Zavolání funkce bez vstupních argumentů po stisknutí tlačítka může vypadat třeba takto:

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

Složitější funkce

Zde už použijeme vstupní argumenty. Viz příklad:

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

Zde jsou hodnoty proměnných nahrazeny jmény argumentů, jejichž seznam MUSÍ být uveden v závorce za jménem funkce (oddělené čárkami).

Když potom voláme funkci, musíme definovat hodnoty argumentů:

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

Vracení hodnot z funkce

Kromě přijímání externích argumentů může funkce vracet hodnoty:

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

Zde máme proměnnou "konecne_skore" a funkci "vypocet" která určí hodnotu proměnné. Funkce "výpočet" by mohla vypadat třeba takto:

```
function vypocet(body, koeficient) {

      return body * koeficient;
}
```

K definici, co má funkce vracet slouží příkaz "***return***". V našem případě se pouze vynásobí argumenty "body" a "koeficient". Proměnná "***konecne_skore***" tak bude mít hodnotu "1300".

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. Například řetězec "ěščřžýáíé" bude vypadat takto:"%EC%9A%E8%F8%9E%FD%E1%ED%E9" nebo v režimu unicode UTF-8 (jen MX verze):"%C4%9B%C5%A1%C4%8D%C5%99%C5%BE%C3%BD%C3%A1%C3%AD%C3%A9"

  Jen připomenu, režim Unicode lze vypnout příkazem: `system.useCodepage = true`

  - opačný efekt má funkce ***unescape()***
- ***eval(nazev 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***"

  pozn.: pokud ovládáte PHP, tak asi víte, že funkce eval() zde dokáže ohodnotit řetězec na jakýkoliv příkaz. To ActionScript nedovoluje.

- ***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`
- ***unescape***
  viz dříve

Atributy instancí

O instancích symbolů z knihovny jsme se bavili v kapitole "Knihovna", ale jen pro jistotu zopakuji, že 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)

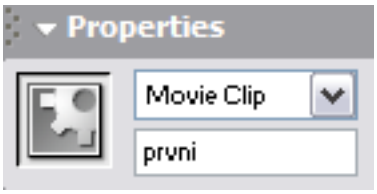
```
setProperty("prvni", _rotation, "30");
```

Tento příkaz otočí instanci "prvni" o 30°
Je možné také použít zjednodušený zápis:

```
prvni._rotation = 30
```

Efekt bude stejný

Připomeňme, že je nutné instanci symbolu pojmenovat.



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. Povšimněte si prosím tečky mezi výrazem ***první*** a ***_rotation*** - ta tam být ***MUSÍ***!

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
- ***_quality*** ("LOW/MEDIUM/HIGH/BEST")
  mám-li být upřímný, nevím v čem se toto liší od "_highquality"
- ***_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 +)

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***.

UI znamená zřejmě User Interface. Komponenty jako takové existují už od verze 5, ale UI komponenty používají nové funkce ActionScriptu, takže ve verzi 5 nefungují.

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

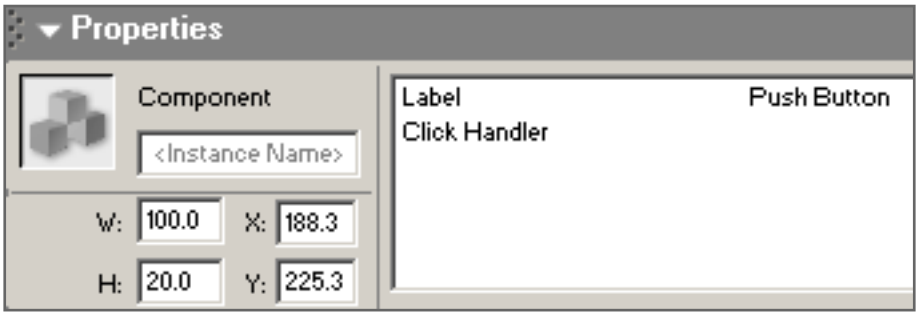
Druhy UI komponentů prakticky vychází z formulářů v jazyce HTML.

- ***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)

1. Push button

První a nejjednodušší z rodiny UI Komponetů. Jedná se o klasické tlačítko

Pokud ho přetáhnete na scénu, zobrazí se na panelu "***Properties***" okno s názvem "***Parameters***":



Právě v tomto okně jsou vyvedeny ony editovatelné proměnné. V tomto případě jen 2.

- ***Label*** (popisek)
  - Jedná se o zobrazovaný text. Kliknutím na položku je možné ji editovat.
- ***Click Handler*** (funkce)
  Do tohoto pole je nutné napsat název ***funkce***, která bude zavolána po kliknutí na tlačítko

```
function akce1() {
          gotoAndPlay(15);
}
```

Pokud tedy do položky "***Click Handel***" napíšete "***akce1***" a v prvním snímku definujete funkci "***akce1***", přejde běh animace po kliknutí na snímek ***15***.

Metody objektu PushButton:

getEnabled	Vrací hodnotu true/false jestli je komponent zablokovaný
getLabel	Vrací hodnotu popisku
setClickHandel	Nastavuje funkci, která bude volána po kliknutí
setEnabled	Umožňuje zablokovat/odblokovat komponent
setLabel	Nastavuje popisek
setSize	Nastavuje rozměry (W, H)

2. Combo Box

Volný překlad by byl nejspíš "roletové menu".

Combo Box obsahuje několik položek, ze kterých je možno jednu zvolit.

- ***Editable*** (true/false)
  Pokud zvolíte "true", půjde, kromě volby jedné možnosti, napsat vlastní hodnota
- ***Labels*** (popisky)
  Pokud klepnete na lupu vpravo, otevře se okno, kde je možné definovat popisky možností
- ***Data***
  Zde přiřadíte možnostem hodnoty, které budou nastavovat
- ***Row Count*** (číslo)
  Počet řádků, které mají být viditelné při rozbalení
- ***Change Handler*** (funkce)
  Funkce, která má být volána při zvolení položky

Editable	false
Labels	[jablka,hrušky,třešně,švestky]
Data	[1,2,3,4]
Row Count	4
Change Handler	akce

```
function akce() {

          gotoAndStop(_root.roletka1.getValue());

}
```

Zde je použita metoda objektu ComboBox ***getValue()*** udávající hodnotu "Data" vybrané položky. Je samozřejmě ***nutné instanci komponentu pojmenovat*** (***roletka1***). Pokud teď vybereme například položku "***hrušky***", přejde animace na snímek ***2***.

Metody objektu ComboBox:

addItem	Přidá položku: `addItem("popisek", data)`
addItemAt	Přidá položku na určené místo (index)
getItemAt	Vrací popisek: `getItemAt(4).label` nebo data: `...(4).data`
getLength	Vrací počet položek
getRowCount	Vrací počet řádků při rozbalení
getScrollPosition	Vrací index položky zobrazené nejvýše v seznamu
getSelectedIndex	Vrací index vybrané položky
getSelectedItem	Vrací popisek/data vybrané položky (viz getItemAt)
getValue	Vrací data vybrané položky
removeAll	Odstraní všechny položky
removeItemAt	Odstranění určité položky (index)
replaceItemAt	Přepsat položku: `replaceItemAt(4, "novynazev", data)`
setChangeHandler	nastavení funkce, která má být volána při zvolení položky
setDataProvider	Nastaví hodnotu dat položek podle pole Array()
setEditable	viz Editable
setItemSymbol	Zobrazení symbolu v rozbalovacím seznamu
setRowCount	Nastaví počet řádků při rozbalení
setSelectedIndex	Nastaví index vybrané položky
setValue	Nastaví hodnotu editovatelné položky (viz Editable)
SortItemsBy	Seřadí položky buď podle popisku nebo podle dat

Ostatní viz dříve…

3. List Box

Pracuje na stejném principu jako Combo Box (v kódu HTML tyto 2 objekty dokonce reprezentuje jeden tag). Jedná se o ***trvale rozbalené*** menu

Jediná odlišnost je v možnosti vybrat více položek najednou (***Select Multiple***)

Metody objektu ListBox:

getSelectedIndices	Vrací indexy vybraných prvků zapsané do pole Array
getSelectedItems	Vrací popisky/data vybraných prvků zapsané do pole Array (syntaxe viz `getItemAt`)
getSelectMultiple	Sděluje jestli jsou povoleny vícenásobné volby
setAutoHideScrollBar	Nastavuje zobrazení posuvníku (true/false)
setSelectedIndices	Nastavuje vybrané prvky podle indexů zapsaných v poli Array
setSelectMultiple	Povoluje nebo zakazuje vícenásobné volby

Ostatní viz dříve…

4. Check Box

Zašktrávací pole. Nabývá jen 2 hodnot: označeno/neoznačeno (true/false)

- ***Label***
  viz dříve
- ***Initial value*** (true/false)
  Počáteční hodnota
- ***Label Placement*** (left/right)
  Pozice popisku
- ***Change Handler***
  viz dříve

Metody objektu CheckBox:

getValue	Získání hodnoty pole (true/false)
setValue	Nastavení hodnoty pole (true/false)
setLabelPlacement	Nastavení pozice popisku ("left"/"right")

Ostatní viz dříve…

5. Radio Button

Podobná funkce jako v předchozím případě s tím rozdílem, že Radio Buttonů musí být 2 a více v jedné skupině a je možné vybrat vždy jen jednu z možností v jedné skupině.

- ***Label***
  viz dříve
- ***Initial State***
  viz dříve
- ***Group Name*** (název skupiny)
  Název skupiny musí být samozřejmě u všech prvků skupiny stejný
- ***Data***
  Data přidělená položce
- ***Label Placement***
  viz dříve
- ***Change Handler***
  viz dříve

Label	Jablka
Initial State	true
Group Name	skupina1
Data	1
Label Placement	right
Change Handler	funkce

Metody objektu RadioButton:

getGroupName	Získání názvu skupiny
setGroupName	Nastavení názvu skupiny
getState	Získání stavu (true/false)
setState	Nastavení stavu (true/false)

Ostatní viz dříve…

6. Scrollbar

Posuvník, který je možno připojit k textovému poli typu "***Multiline***". Pokud scrollbar přesunete "dovnitř" textového pole, automaticky se přichytí a nastaví se target.

- ***Target TextField*** (jméno instance - nikoliv zobrazované proměnné)
  Ovládané textové pole
- ***Horizontal*** (true/false)
  Pokud zvolíte ***true***, bude posuvník vodorovný

Metody objektu ScrollBar:

getScrollPosition	Udává číslo řádku zobrazeného nejvýše
setHorizontal	vodorovný (true) nebo normální (false) posuvník
setLargeScroll	Nastavuje posuv (řádky) v případě kliknutí do dráhy
setScrollPosition	Nastavuje pozici posuvníku (číslo nejvyššího řádku)
setScrollProperties	**setScrollProperties(**celk. počet řádků, č. řádku v nejvyšší pozici jezdce, č. řádku v nejnižší pozici jezdce)
setScrollTarget	Nastaví jméno ovládané instance textového pole
setSize	Nastavuje délku posuvníku v pixelech
setSmallScroll	Nastavuje posuv (řádky) při kliknutí na šipku

Ostatní viz dříve…

7. Scroll Pane

Poslední z rodiny UI komponentů. Tento element se tak trochu podobá ***in line frame*** z HTML. Je to okno, které umí uvnitř zobrazovat ***MovieClipy***. Pokud je MC větší, než okno, zobrazí se patřičný posuvník.

- ***Scroll Content*** (Jméno MovieClipu)
  Zobrazovaný obsah
- ***Horizontal Scroll*** (auto/true/false)
  Vodorovný posuvník
- ***Verical Scroll*** (auto/true/false)
  Svislý posuvník
- ***Drag Content*** (true/false)
  Povolit/zakázat možnost uchopení MC

Metody objektu ScrollPane:

getPaneHeight	Udává šířku pole
getPaneWidth	Udává výšku pole
getScrollContent	Udává název zobrazovaného MovieClipu
getScrollPosition	pozice MC (za příkaz je nutno přidat `.x` nebo `.y`)
loadScrollContent	Načte do pole externí SWF nebo JPEG `(URL, [funkce zavolaná po načtení], [pozice funkce])`
refreshPane	znovunačtení obsahu (např. po změně jeho vlastností)
setDragContent	Povolit/zakázat možnost uchopení MC
setHScroll	Zobrazení vodorovného posuvníku (auto/true/false)
setVScroll	Zobrazení svislého posuvníku (auto/true/false)
setScrollContent	Nastavení zobrazovaného MC
setScrollPosition	Nastavení pozice MC (viz **`getScrollPosition`***)*
setSize(š, v)	Nastavení velikosti pole

Stylování UI komponentů

Stylování komponentů považuji za jejich nejsilnější zbraň. Žel bohu se jedná převážně jen o změnu barvy toho kterého prvku. Existuje několik možností stylování.

1. Globální

Týká se veškerých komponentů v animaci. Používá se objektu `globalStyleFormat`.

Malý příklad:

```
globalStyleFormat.arrow = 0xFF0000;
globalStyleFormat.applyChanges();
```

V prvním řádku je definována ***vlastnost*** objektu globalStyleFormat "***arrow***" jako hexadecimální barevná hodnota (v našem případě červená)

Následuje ***metoda*** applyChanges(), která aplikuje změny.

Důsledek bude takový, že veškeré šipky (ComboBox, ScrollBar, ...) budou obarveny na červeno.

Povšimněte si, že objekt globalStyleFormat nepotřebuje konstruktor (viz Objekty).

2. Skupinové

Jak již víte, objekt globalStyleFormat ovlivňuje ***všechny*** instance komponentů na scéně. My si ale teď vytvoříme ***vlastní objekt***, který bude ovlivňovat jen ***určitou skupinu*** komponentů.

```
skupina1 = new FStyleFormat();

skupina1.addListener(radio1);
skupina1.addListener(radio2);
```

```
skupina1.radioDot = 0xFF0000;

skupina1.applyChanges();
```

Zde již musíme použít konstruktor "new" pro vytvoření svého objektu. Dále musíme nastavit, kterých instancí se bude formátování týkat **addListener(**nazev instance**)**. Přirozeně je nutné instance ***pojmenovat***. Opačný efekt má metoda **removeListener(**název instance**)**.

Ve 4. řádku pak definujeme vlastnost ***radioDot*** jako opět červenou barvu. Nakonec musíme opět ***aplikovat změny***.

Konečný efekt bude obarvení teček RadioButtonů "***radio1***" a "***radio2***

3. Přímé

Třetí možnost je přímé adresování vlastnosti danému komponentu:

```
názevInstance.setStyleProperty("vlastnost", hodnota);
```

Myslím, že je to víc než jasné. Jen nezapomeňte uzavřít název vlastnosti do uvozovek, aby nebyl chápán jako proměnná. Malá ukázka:

```
seznam1.setStyleProperty("arrow", 0xFF0000);
```

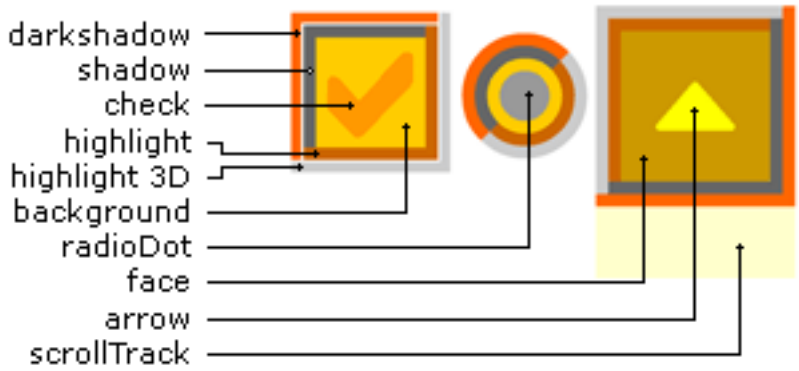
Tento příkaz tedy obarví na červeno šipku v instanci komponentu "***seznam1***

Seznam vlastností:

arrow	Barva šipky
background	Barva pozadí
backgroundDisabled	Barva pozadí u neaktivního komponentu
check	Barva "fajfky" [√] u CheckBox
darkshadow	Barva tmavého stínu
embedFonts [true/false]	Použití přibaleného fontu (font musí být již přibalen do některého textového pole, nebo musí být umístěn v knihovně a musí být definován font komponentu pomocí `textFont`)
face	Barva pozadí tlačítek a vnitřních ploch scrollbarů
fadeRate	Postupné rozsvícení (ms) vybrané položky u Combo a ListBox
PopUpFade	Postupné objevení (ms) seznamu u ComboBox
foregroundDisabled	Obarvení šipek neaktivního komponentu
highlight	Obarvení plochy "highlight"
highlight3D	Obarvení plochy "highlight 3D"
radioDot	Barva tečky RadioButton komponentu
scrollTrack	Barva dráhy posuvníku
selection	barva výběru u ComboBox a ListBox
selectionDisabled	Barva vybrané položky u neaktivního komponentu ListBox

selectionUnfocused	Barva vybrané položky ListBox, který není právě editován
shadow	Barva stínu
textAlign	Zarovnání textu ("left" / "right")
textBold	Tučný text (true/false)
textColor	Barva textu
textDisabled	Barva textu neaktivního komponentu
textFont	Použitý font ("Arial" / "Verdana" /...)
textIndent	Levé odsazení textu (px)
textItalic	Kurzíva (true/false)
textLeftMargin	Levý okraj textu (px)
textRightMargin	Pravý okraj textu (px)
textSelected	Text vybrané položky u ComboBox a ListBox
textSize	Velikost textu (px)
textUnderline	Podtržený text (true/false)

Obrázek některých nejasných ploch:



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...

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***:

```
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".

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.

Například objekt ***Button***

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

Tento zápis edituje vlastnost ***enabled*** na ***nepravda***. Tlačítko s názvem "tlacitko1" tedy nepůjde zmáčknout. Povšimněte si, že není nutné použít konstruktor - tlačítko už existuje na scéně.

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***->**method**

Uvnitř metody je definována proměnná area jako obsah kruhu ($PI*radius^2$). Hodnotu této proměnné potom metoda vrací (return)

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_***.

Popis předdefinovaných objektů

Pro popis všech předdefinovaných objektů s jejich metodami a vlastnostmi prosím navštivte následující kapitoly:

- Objekty "CORE"
  Popis základních objektů Flashe, jako Array, Date, Math, String a další.
- Objekty "MOVIE"
  Popis objektů pro manipulaci s fyzickými prvky scény, jako Button, MovieClip, Color, Sound, TextField a další.

Objekty - CORE

Toto je první část popisu předdefinovaných objektů Flashe. Pokud jste ještě nečetli kapitolu "Objekty - úvod", silně vám to doporučuji!

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

Popis objektů MOVIE naleznete v kapitole objekty "MOVIE".

Array (pole)

Je to objekt obsahující větší množství separovaných hodnot, které jsou očíslovány (index). Jen dodám, že první položka má index 0 (nikoliv 1):

```
pole = new Array("a", "b", "c", "d", "e")
```

potom platí, že:

```
pole [0] = "a";
pole [1] = "b";
pole [2] = "c";
pole [3] = "d";
pole [4] = "e";
```

je možné také definovat prázdné pole s konečným počtem prvků:

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

toto pole bude mít jen 5 prvků

Metody:

contact	spojuje pole dohromady a prvky seřadí za sebe: `pole1.contact(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")`

Vlastnosti:

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

Boolean

Ohodnotí vložený výraz a konvertuje na ***TRUE*** nebo ***FALSE***:

```
test = new Boolean(x)
```

pokud dosadíte za ***x***:

- ***nic*** -> false
- ***0*** -> false
- ***jiné číslo*** -> true
- ***řetězec "1"*** -> true
- ***jiný řetězec*** -> false

Metody:

toString	Vypíše hodnotu true/false jako textový řetězec
valueOf	Vypíše hodnotu true/false jako booleovskou funkci

Date (datum)

Objekt obsahující informace o aktuálním čase a datumu

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

Metody:

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.

Math (matematický)

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

Syntaxe:

```
Math.metoda(výraz);
```

Metody:

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.

Konstanty:

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).

Number (číslo)

Slouží k manipulaci s čísly.

```
cislo = new Number(5)
```

Metody:

toString	převede číslo na řetězec
valueOf	udává původní hodnotu objektu

Konstanty:

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

String (textový řetězec)

Slouží k manipulaci s textovými řetězci.

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

Metody:

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. Jestliže se hodnota objeví více než jednou, je udán index prvního výskytu. Jestliže není hodnota nalezena, je udáno 1. `text.indexOf("abc", 5)`
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.

Objekty - MOVIE

Toto je druhá část popisu předdefinovaných objektů. V první části jsme se bavili převážně o imaginárních objektech (Array, String, ...). Nyní bude řeč 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***.

Podrobněji o této problematice - "Objekty - úvod"

Jen upozorňuji, že drtivá většina objektů "MOVIE" existuje až od verze 6 (MX) a jejich použití vyžaduje pokročilou znalost ActionScriptu.

Button (tlačítko)

Objekt ovlivňující instanci tlačítka na scéně.

Bez konstruktoru.

Metody:

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

Vlastnosti:

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:

```
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 "tlacitko" na snímek č. 5.

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

Capabilites (schopnosti)

Vrací vlastnosti přehrávače případně serveru.
Syntaxe:

```
System.capabilities.vlastnost
```

Vlastnosti:

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

Color (barva)

Umožňuje definovat barvu instance MC.
Potřebuje konstruktor!

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

Metody:

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" - viz Knihovna

```
barva = new Color(kruh);
```
// vytvoření objektu "barva" s cílem na MovieClip "kruh"

```
transformace = new Object();
```
// vytvoření objektu "transformace" pomocí generického objektu "Object()"

```
transformace = { ra: '50', rb: '244', ga: '40', gb: '112', ba: '12', bb: '90', aa: '40', ab: '70'};
```
// přiřazení hodnot

```
barva.setTransform(transformace);
```
// aplikování hodnot v objektu "barva"

Parametry metody:

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).

Key (klávesa)

Objekt upravující výstup z klávesnice.
Bez konstruktoru.

Metody:

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`

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

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.

ukázka AddListener:

```
objekt3 = new Object();
```
// vytvoření generického objektu Object()

```
objekt3.onKeyDown = function () {
	 Play();
}
```
// deklarace ***metody*** onKeyDown (viz Objekty-úvod)

```
objekt3.onKeyUp = function () {
	Stop();
}
```
// deklarace ***metody*** onKeyUp (viz Objekty-úvod)

```
Key.addListener(objekt3);
```
// asociace objektu "objekt3" s objektem Key

Kdybychom teď zmáčkli nějakou klávesu, začalo by přehrávání animace. Po uvolnění klávesy by se animace zastavila.

Mouse (myš)

Objekt upravující výstup z myši

Bez konstruktoru.

Metody:

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

Metody objektů asociovaných pomocí **`addListener()`:**

onMouseDown	Kliknutí
onMouseMove	Pohyb
onMouseUp	Uvolnění

* Způsob použití je obdobný jako u objektu Key

MovieClip

Objekt ovlivňující instanci MovieClipu na scéně.

Bez konstruktoru.

Pozn.:

Je dobré si uvědomit, že hlavní časová osa je v postatě taky MovieClip, který nenese žádné jméno (případně "_root"). Můžete tedy klidně jako název MC napsat "_root" a příkaz tak bude platit pro hlavní časovou osu. Samozřejmě není možné používat příkazy, které na hlavní časové ose postrádají smysl.

Metody:

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

Vlastnosti:

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. Tento způsob navíc umožňuje použít události, které jsou jinak MovieClipu cizí (např. onPress). Takto je možné MC upravit tak aby se choval jako tlačítko.

Použití je stejné jako u objektu button:

```
movieclip.onMouseDown = function () {

      gotoAndPlay(5);

};
```

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

Selection (výběr)

Objekt kontrolující výběr textu, případně MovieClipu a Tlačítka
Bez konstruktoru.

Metody:

addListener	Připojit objekt reagující na událost onSetFocus
getBeginIndex	Číslo prvního znaku výběru
getCaretIndex	Udává pozici kurzoru
getEndIndex	Číslo posledního znaku výběru
getFocus	Udává jméno zobrazované proměnné vybraného objektu. Pokud není vybráno textové pole, vrací pozici objektu.
removeListener	Odpojit objekt přidaný pomocí addListener
setFocus	Nastavit ukazatel do textového pole
setSelection(od, do)	Vybere část textu editovaného textového pole

Události:

onSetFocus	Při změně výběru zavolat metodu připojeného objektu *

**** Ukázka události*** **`onSetFocus`**

```
prvni = new Object();

prvni.onSetFocus = function() {
        submit.enabled = true;
}

Selection.addListener(prvni)
```

Při ***změně výběru*** textového pole se zavolá metoda generického objektu Object ***onSetFocus*** a zavolá se funkce, kde je příkaz na odblokování tlačítka s názvem "***submit***".

Sound (zvuk)

Souží k manipulaci se zvukem

Potřebuje konstruktor!

```
zvuk = new Sound()
```

Metody:

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

Vlastnosti:

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

Události:

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

**** Ukázka nastavení transformace pomocí* `setTransform()`**

Tato metoda umožňuje nastavit hlasitost jednotlivých reproduktorů (L a P) a u stereo zvuků také definovat, která stopa(y) se má v daném reproduktoru přehrávat.

Syntaxe:

```
zvuk.setTransform(transformObjekt)
```

Je tedy nutné definovat nastavení do objektu ***transformObjekt***:

Parametry objektu:

- **`ll`** - Procentuální hlasitost ***levého*** vstupu v ***levém*** reproduktoru
- **`lr`** - Procentuální hlasitost ***pravého*** vstupu v ***levém*** reproduktoru
- **`rr`** - Procentuální hlasitost ***pravého*** vstupu v ***pravém*** reproduktoru
- **`rl`** - Procentuální hlasitost ***levého*** vstupu v ***pravém*** reproduktoru

Definování stereo zvuku jako mono by se nastavilo třeba takto:

```
transformObjekt = new Object();
transformObjekt.ll = 50;
transformObjekt.lr = 50;
transformObjekt.rr = 50;
transformObjekt.rl = 50;

zvuk.setTransform(transformObjekt);
```

Stage (nastavení scény)

Tento objekt umí nastavit vlastnosti scény v přehrávači podobně jako příkaz FSCommand. Přináší však některé novinky - například událost onResize

Bez konstruktoru.

Metody:

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

Vlastnosti:

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

Události:

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

**** Ukázka události* `onResize`**

```
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()***

TextField (textové pole)

Slouží k manipulaci s textovými poli typu dynamic a input text
Bez konstruktoru.

Metody:

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`)

Vlastnosti:

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
wordWrap	Vrací booleovskou hodnotu, zda se text má zalamovat

Události:

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

TextFormat (formát textu)

Objekt definující hodnoty metod ***setTextFormat*** a ***setNewTextFormat*** objektu ***TextField***.
Potřebuje konstruktor!

```
nadpis = new TextFormat();

nadpis.vlastnost = hodnota;
```

```
kolonka3.setTextFormat(nadpis);
```

Metody:

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

Vlastnosti:

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

Fotogalerie

Poslední dobou si hodně žádáte o Fotogalerii, tak si dnes jednu takovou vytvoříme. Obrázky (ve formátu JPEG) samozřejmě nejsou součástí samotné SWF animace, ale načítají se z vnějšku.

Nejprve si vysvětlíme příkaz, který bude pro načítání fotek klíčový:

Pokud umístíme na scénu instanci prázdného MovieClipu s názvem třeba "`photo`", je možné použít ***metodu*** objektu MovieClip `LoadMovie`, která nahrazuje instanci externím SWF nebo JPG souborem:

```
loadMovie("krajinka.jpg", "photo");
```

Tento příkaz by nahradil instanci externím obrázkem (v našem případě "***krajinka.jpg***").

A teď k samotné galerii:

- Nejprve začneme s ***preloaderem***. Použijeme stejný, jako uvádím v příkladech.
  Takže si vytvoříme MovieClip s obdélníkem, který bude zobrazovat načítané procenta, vložíme jeho instanci na scénu a pojmenujeme "***preloader***".
  Snímek roztáhneme na velikost 2 snímků a do druhého vložíme příkazy:

```
loading = Math.round(getBytesLoaded()/getBytesTotal()*100);
        setProperty("preloader", _xscale, loading);
if (loading == 100) {
        play();
} else {
        gotoAndPlay(1);
}
```

  Pokud vám není jasná funkce preloaderu, přečtěte si výše zmiňovanou kapitolu.
- Nyní umístíme do 3. kl. snímku instanci UI komponentu ***ListBox*** a pojmenujeme "***selector***". Tento seznam bude umožňovat volbu fotky. Vyplníme parametr komponentu ***ChangeHandler*** jako "***showphoto***" (je to název funkce zavolané po změně výběru)
- Dále vytvoříme prázdný MovieClip, umístíme instanci na scénu a pojmenujeme "***photo***". Na pozici tohoto MC se právě budou načítat fotky.
- Do třetího snímku definujte tyto akce:

```
photos = new Array("01.jpg", "02.jpg", "03.jpg", "04.jpg", "05.jpg", "06.jpg");

names = new Array("Carrera Coupe 2000", "911 GT1 Coupe 1997",
"911 Turbo Coupe 3.6 2000", "Boxster 2.5 1997", "911 GT3 Coupe 2000",
"Boxster 3.2 S 2000");
```

Tyto 2 pole obsahují jména souborů a jejich popisky.

```
selector.setDataProvider(names);
selector.setAutoHideScrollBar(true);
```

Nejprve je seznam ***Listbox*** s názvem "***selector***" vyplněn názvy fotek uložených v poli "***names***" Dále je nastaveno skrytí posuvníku seznamu v případě, že není potřeba.

```
function showphoto() {
      loadMovie(photos[selector.getSelectedIndex()], "photo");
}
Stop();
```

Tato funkce bude zavolána při výběru položky v ListBoxu. Je zde načten obrázek na pozici MC "***photo***". Místo názvu je hodnota pole "***photos***". V hranatých závorkách je uvedena metoda ***selector.getSelectedIndex()***, která vrací index vybrané položky ListBoxu.

Pokud by byla vybrána položka s indexem ***3***, měl by výraz hodnotu ***photos[3]*** a tedy "***04.jpg***"

- Dále můžeme do vrstvy pod MovieClip umístit další preloader, který bude kontrolovat načítání obsahu MovieClipu. Syntaxe je obdobná, jen s tím rozdílem, že tentokrát bude preloader ukazovat míru načtení neustále. Příkazy tedy definujeme samotnému preloaderu po události ***EnterFrame***:

```
onClipEvent (enterFrame) {
    loading = Math.round(_root.photo.getBytesLoaded()/
    _root.photo.getBytesTotal()*100);
    setProperty(this, _xscale, loading);
}
```

Opět stejný princip: Preloader kontroluje procentuální hodnotu načtení MC "***photo***" a podle toho mění šířku sebe sama (this).

Jak jsem již zmiňoval, preloader je na místě neustále (po načtení ho ale překryje obrázek).

- Nakonec umístíme na scénu tlačítko, které bude spouštět prezentaci (SlideShow). Já jsem použil tlačítko z UI komponentů, ale je možné použít i normální. Jeho samotná funkce je spuštění přehrávání animace.

SlideShow

Nyní máme hotovou galerii. Ale aby to nebylo tak jednoduché, přidáme ještě možnost zobrazit snímky v časových intervalech jako prezentaci.

- Takže tlačítko, které jsem popisoval před chvílí posunulo animaci na snímek 4. Zde se tedy bude odehrávat ona prezentace.
- Na scénu umístíme 2 instance našeho prázdného MC a pojmenujeme "***controller***" a "***slide***". První bude kontrolovat zobrazování fotek ve druhé. Do třetice ještě umístíme někde ***DynamicText***, který bude zobrazovat proměnnou "***name***".
- MC "***controller***" tedy definujeme:

```
onClipEvent (load) {
   i = 0;
   loadMovie(_root.photos[i], "_root.slide");
   _root.name = _root.names[i];
   startcount = false;
}
```

Nejprve je potřeba udělat pár věcí po načtení MC.

Takže jako první je deklarována proměnná ***i*** jako ***nula***. Tato proměnná bude charakterizovat aktuálně zobrazovanou fotku.

Dále je načtena první fotka - její jméno je nataženo z pole "***photos***".
Zmiňované proměnné "***name***" určíme aktuální hodnotu z pole "***names***"

Nakonec je deklarována proměnná ***startcount = false***, která indikuje spuštění odpočtu (viz dále)

```
onClipEvent (enterFrame) {

// Pokud je fotka načtena
if (_root.slide.getBytesLoaded() == _root.slide.getBytesTotal() && startcount ==
false)
{

// nastavit odpočet
countdown = getTimer()+5000;
startcount = true;
}

// Pokud odpočet vypršel
if (startcount == true && countdown<=getTimer()) {

     if (_root.photos.length != i+1) {
            i++;
     } else {
           i = 0;
     }

      // načíst další fotku
    loadMovie(_root.photos[i], "_root.slide");
    _root.name = _root.names[i];
    startcount = false;

    }
}
```

Takže jen stručně:

V každém snímku (enterFrame), dokončeno načítání fotky a pokud již nebylo spouštěno odpočítávání (***startcount == false***), nastavit toto odpočítávání na 5sekund.
(***getTimer()*** udává počet ms od začátku přehrávání animace)

Pokud odpočet vypršel, zvýší se proměnná "***i***" o jednu - pokud již však nebylo dosaženo konce pole "***names***" - v takovém případě by se začalo zase od nuly.

Nakonec se načte příslušná fotka místo instance MC "***slide***" a provede se opět zobrazení jejího názvu.

- Poslední krok je přidání preloaderu se stejnou funkcí jako v předchozím případě, tlačítka pro ***návrat*** na snímek 3 (nebo ***Play()*** - vyjde to nastejno) a samozřejmě příkazu
Stop() do 4. snímku

Pffffff.... no snažil jsem se o co největší jednoduchost - nezadařilo se. Proto bych doporučil stáhnout ukázkový FLA soubor.

Flash GuestBook

Poslední dobou jsem zaplaven dotazy, jak vytvořit knihu návštěv ve Flashi. Dnes si jednu takovou jednoduchou vytvoříme. Jak jsem již zmiňoval v kapitole "FAQ", Flash samotný z bezpečnostních důvodu ***nemůže*** zapisovat do souborů (podobně jako JavaScript).

Zápis dat tedy budeme muset svěřit nějakému serverovému (pracuje na serveru) skriptu. Já používám PHP, ale použít můžete i ASP a další.

Hlavní princip Flashové knihy návštěv je v počátečním ***načtení dat ze souboru*** a následné poslání změn PHP skriptu, který je ***zapíše*** zpět do souboru.

Funkce, kterou použijeme je:

```
loadVariables("zdroj", "cílový MC", [odesílání proměnných] );
```

Funkce load variables načte ze souboru proměnné a odešle je do cílového ***MovieClipu*** (pokud necháme název prázdný, tak na hlavní časovou osu).

Zároveň ale tato funkce umožňuje, kromě ***načítání*** proměnných, proměnné i ***posílat***.

Pokud definujete do třetí položky ***POST*** nebo ***GET***, budou se proměnné i odesílat (nastaveným způsobem)

Pokud v roletovém menu "Location" vyberete místo "Target" položku "***Level***", změní se syntaxe takto:

```
loadVariablesNum("zdroj". "úroveň načtené animace",  [odesílání proměnných] );
```

Tento způsob má smysl jen tehdy, pokud je načteno více animací pomocí příkazu loadMovie().

Takže teď k samotnému návodu:

- nejprve si vytvoříme formulářové prvky (***Input texty***) podle potřeby. Asi nejčastější jsou "***Jméno***", "***Email***", "***Web***" a "***Vzkaz***".

  Těmto polím přiřadíme nějaké proměnné. Já mám "`name`", "`email`", "`web`" a "`message`"
- Klíčový snímek, s těmito formuláři zvětšíme na velikost 2 snímků
- Dále vytvoříme prázdný MovieClip, vložíme na scénu a pojmenujeme "***check***"
- Vytvoříme novou vrstvu a do ***prvního snímku*** definujeme následující příkazy:

```
stop();
system.useCodepage = true;

name = "";
email = "@";
web = "http://";
message = "";

loadVariables("book.txt", "check");
```

Nejprve zastavíme přehrávání animace (vysvětlím později). Dále musíme zakázat kódování Unicode UTF-8. Dělá to

totiž značné problémy s českou diakritikou (nevím proč).
Dále jsou deklarovány dříve zmíněné proměnné.

Nakonec zde máme příkaz načítající proměnné ze souboru ***book.txt*** do MovieClipu ***check***. Tento MovieClip zde není pro legraci. Bude sloužit ke kontrole, zda již bylo stahování dat ze souboru dokončeno či nikoliv. Tomuto MovieClipu definujeme následující příkazy:

```
onClipEvent (data) {
     _root.Play();
}
```

Povšimněte si události. Událost "***Data***" nastane po úspěšném odeslání proměnných do MovieClipu (viz řízení animace). Tím máme zaručeno, že animace bude pokračovat až tehdy, když bude celý obsah souboru načten.

- Vytvoříme novou vrstvu a do druhého kl. snímku vložíme ***Dynamic Text*** typu ***Multiline***, který bude zobrazovat načtenou proměnnou ze souboru (***check.book***) a povolíme mu ***zobrazování HTML tagů***.
- K formulářovým prvkům umístíme tlačítko, kterému definujeme tyto akce:

```
on (release) {

datum = new Date();

if (name != "" && message != "") {

      name = "<p><b><font size=\"11px\" color=\"#D34A33\">"+name+"</font>
      zanechal(a) vzkaz "+datum.getDate()+"."+datum.getMonth()+" v "
      +datum.getHours()+":"+datum.getMinutes()+"</b></p>";

      message = "<p>"+message+"</p><p> </p><p> </p>";

      if (email != "@" && email != "") {
                email = "<p><u><a href=\"mailto:"+email+"\">"
                +email+"</a></u></p>";
      } else {
                email = "";
      }

      if (web != "" && web != "http://") {
                web = "<p><u><a href=\""+web+"\">"+web
                +"</a></u></p><p> </p>";
      } else {
                web = "";
      }
```

Po stisku tlačítka se nejprve vytvoří objekt Date(), který bude sloužit k přidání časového údaje do knihy. Dále následuje podmínka, která kontroluje vyplnění povinných položek (jméno a vzkaz)

Následující příkazy přidávají HTML tagy k hodnotám proměnných ***name*** a ***message***. Jen dodám, že je nutné napsat před každé uvozovky zpětné lomítko \ (aby nebyly chápány jako ActionScript operátor)

Dále jsou upraveny na odkazy proměnné ***email*** a ***web*** - pouze však v případě, že byly vyplněny.

Dále:

```
check.book = name+email+web+message+check.book;
pridat = check.book;
pridat = escape(pridat);
loadVariables("book.php", "", "POST");
name = "";
email = "@";
web = "http://";
message = "";

}
}
```

K proměnné check.***book*** je přidán nový vzkaz a obsah knihy je zkopírován po proměnné ***pridat***.

Hodnota této proměnné je poté zakódována do formátu URL-encoded (viz Předdefinované funkce). Tím se zajistí aby nedošlo k chybné interpretaci některých nestandardních znaků (např. uvozovek).
(Dekódování po načtení provede Flash automaticky)

Pak se odešlou proměnné do skriptu ***book.php*** pomocí HTTP kanálu POST.

Dal by se použít i příkaz getURL, ale to by prohlížeč otevíral cílovou stránku, což v tomto případě není žádoucí. Při použití loadVariables se odešlou jen proměnné.
Nakonec se vymažou formuláře.

- Poslední, co musíme udělat je přidat do druhého snímku příkaz:

```
stop();
```

- PHP stránka ***book.php*** pak bude obsahovat následující PHP skript:

```
<?

if ($pridat != "") {

    $fp = fopen("book.txt", "w");
    fwrite($fp, "book=$pridat");
    fclose($fp);
}

?>
```

Pokud není proměnná ***pridat*** prázdná, otevřít soubor book.txt pro zápis a zapsat do něj řetězec "`book=[hodnota proměnné pridat]`"

Proměnné v souboru musí být zapsány jako při posílání metodou GET, tedy ve formátu URL-encoded:

```
promenna1=texttexttext&promenna2=texttexttext    ...
```

Jen podotknu, že animace může být klidně přímo na stránce ***book.php*** (já ji tak mám).

- ***Pozn.:***

Než budete používat tuto knihu na svých stránkách, doporučuji změnit název proměnné "`pridat`". Pokud by totiž někdo znal název proměnné, mohl by lehce přepsat celý obsah knihy jednoduchým odkazem `book.php?pridat=fuckyou`.
Tuto chybu jsem samozřejmě udělal a jedna chytrá hlava jí využila.
BTW: už to nemusíte zkoušet - opravil jsem to...

Internet Explorer Problém

IE ve své nekonečné moudrosti ponechává v cache otevřené stránky a proto není možné po editaci souboru "book.txt" znovu načítat jeho hodnotu pomocí loadVariables - změny by se ***neprojevily***

Dokonce i po obnovení [F5] si Flash natáhne hodnotu souboru z cache - proto taky je vzkaz přidán jak k proměnné ***přidat*** (ta se bude zapisovat), tak i do proměnné ***check.book*** (která je zobrazována) a načítání tak není nutné.

Pokud zavřete okno prohlížeče, a otevřete jej znovu, budou se již data natahovat ze serveru v pořádku.

SmartClip

Dnes vás seznámím s trochu méně známou funkcí, kterou je vytvoření z obyčejného MovieClipu ***SmartClip (komponent)***.

Obyčejně se při vytváření animací snažíme o co největší interaktivitu - předat otěže animace do rukou uživatele. Dnes to bude trochu naopak. SmartClip totiž na straně diváka ***vypadá jako obyčejný MC***.

Ona chytrost (smart = chytrý) tohoto klipu spočívá v možnosti animátora naprogramovat si určitý prvek (třeba tlačítko), přidělit mu proměnné, naprogramovat způsob chování v určitých situacích a nakonec si vytvořit menu, kde tyto vlastnosti bude možno nastavit.

Pak již stačí vložit onen klip do nějaké animace, nastavit si hodnoty a práce je hotova. V kostce řečeno je to MovieClip s kontrolním panelem (který však má k dispozici jen animátor). V návodu Flashe se dočtete, že je tato funkce ideální při práci v týmu, ale využije ji i jednotlivec.

1. Vytvoření MovieClipu

Jak již bylo řečeno, SmartClip je založen na klasickém MovieClipu. V dnešním příkladu si uděláme kreslicí plátno, na které bude možno malovat štětcem.

- Nejprve vytvoříme MC a pak ještě tzv. ***neviditelné tlačítko*** (tj. tlačítko s grafikou jen ve snímku "HIT")
- tlačítko pak vložíme do MC
- Dále si vytvoříme další MC, do kterého nakreslíme stopu štětce, vložíme do prvního MC a pojmenujeme třeba "***brush***" (brush = štětec)

  Pozn.: Do dalších snímků MC "brush" můžeme nakreslit více druhů tvarů stop.
- ***Tlačítku*** definujeme následující příkazy:

```
on (press, dragOver) {
     pressing = true;
}
on (release, releaseOutside, dragOut) {
     pressing = false;
}
```

  Předchozí zápis znamená, že po ***kliknutí*** nebo ***tažení dovnitř*** je nastavena proměnná ***pressing*** jako ***pravda***.

  Po ***puštění*** tlačítka nebo ***tažení ven*** je nastavena proměnná ***pressing*** na ***nepravda***.
- MovieClipu ***Brush*** pak definujte tyto příkazy:

```
onClipEvent (load) {
      c = new Color(this);
      c.setRGB(_parent.col, 16);
      this.gotoAndStop(_parent.paint);
      setProperty(this, _xscale, _parent.xscl);
      setProperty(this, _yscale, _parent.xscl);
      setProperty(this, _alpha, _parent.alp);
      i = 0;
}
```

Předchozí příkazy se vykonají po načtení MC "brush".

První řádek určuje hodnotu proměnné "***c***" jako ***objekt*** "**`new Color()`**", který nastavuje barvu instance MovieClipu (v našem případě ***this*** = tento, tedy "brush")

Druhý řádek definuje ***metodu*** (method) předchozího objektu - **`setRGB`** - nastavující hodnotu jako barvu v hexadecimálním zápisu (ta šestnáctka na konci) rovnu hodnotě proměnné "***col***", definovanou o úroveň výš (_parent)

Třetí řádek způsobí přeskočení přehrávače uvnitř MC "brush" na snímek definovaný proměnnou "***paint***" (změna tvaru štětce)

Další 3 řádky nastavují atributy MC "brush" (velikost, průhlednost), jejichž hodnoty jsou opět nahrazeny proměnnými.

Uff... a nakonec je proměnná "***i***"nastavena na ***0*** (počítadlo cyklů - viz níže)

- Všechny klíčové snímky v našem SmartClipu (no on je to stále ještě MovieClip...) roztáhneme na šířku 2 snímků
- druhému snímku pak definujeme toto:

```
if (pressing == true) {
      duplicateMovieClip("brush", "brush"+i, i);
      setProperty("brush"+i, _x, _xmouse);
      setProperty("brush"+i, _y, _ymouse);
      i++;
}
gotoAndPlay(1);
```

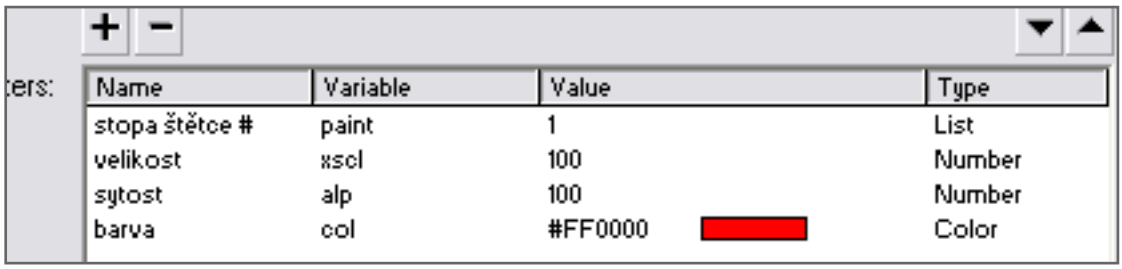
Takže toto znamená, že pokud je naše neviditelné tlačítko zmáčknuto (***pressing == true***), zkopíruje se MC "***brush***" a přidělí se mu jméno ***"brush" + i*** (brush1, brush2, brush3,...) a přenese se do úrovně (***depth***) "***i***" - viz příklad duplikace MovieClipu

A nastaví se této kopii pozice rovna pozici myši.
Nakonec se přičte k počítadlu jednička a přejde se zpět na snímek 1 a celá procedura se opakuje.

2. Přeměna na SmartClip

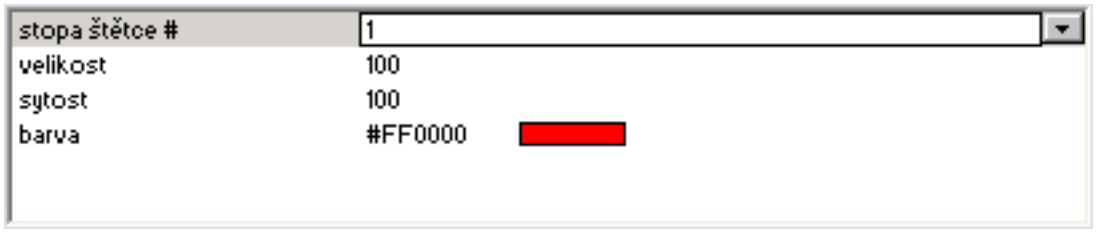
Jak jste si jistě povšimli, nahradili jsme většinu hodnot v předchozích příkazech proměnnými. Tyto proměnné tedy budou zakotvené ve SmartClipu jako ***editovatelné***.

- Pravým klikem na hlavní MC vyvolejte kontextovou nabídku a zvolte "***Component Definition***"
- Mělo by se otevřít stejnojmenné okno. V jeho horní části můžete tlačítkem "+" přidávat editovatelné proměnné:



První sloupec "***Name***" určuje popis (tj. to co uvidí animátor), "***Variable***" je jméno proměnné, "***Value***" je počáteční hodnota a "***Type***" je typ dat (text, číslo, seznam, barva,...).

- pokud to teď necháme svému osudu, mělo by se po kliknutí na SmartClip zobrazit v okně ***Properties*** zhruba toto:



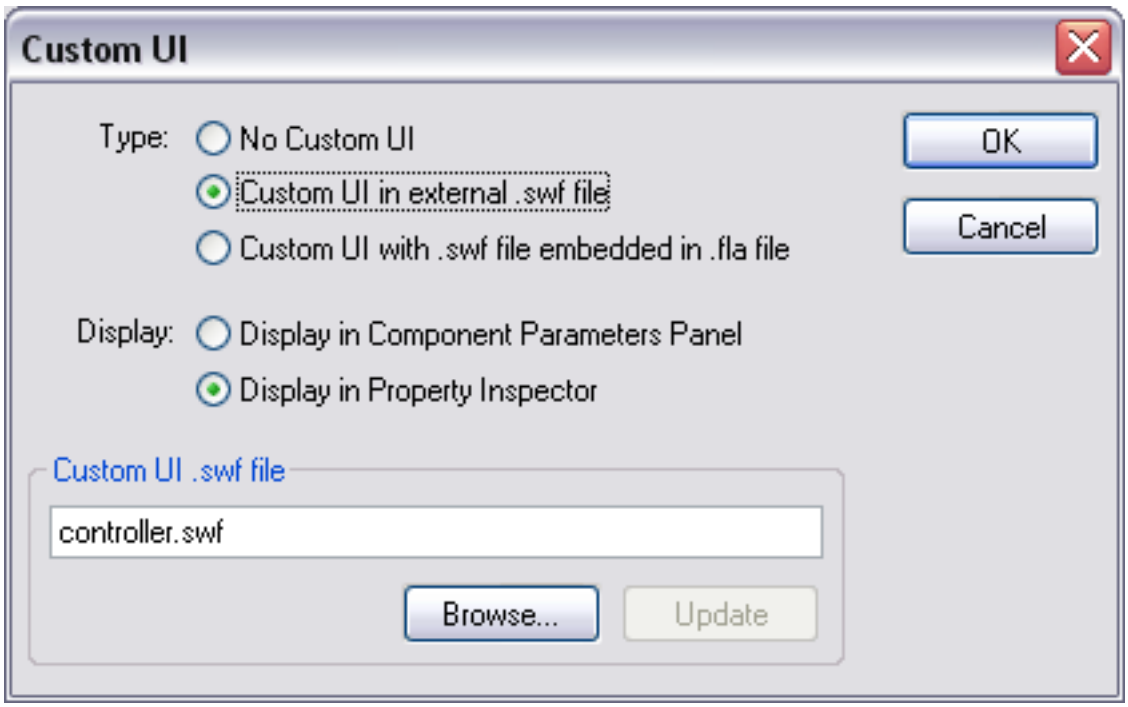
Změnou parametrů a shlédnutím animace příkazem "***Test Movie***" by se měly projevit nastavené hodnoty.

3. Vytvoření speciálního UI rozhraní

Předchozí nastavovací rozhraní je podle mě velice praktické, ale najdou se i tací, kterým připadá příliš sterilní. Maniakům, kteří se vyžívají v kýčovitostech proto poskytuje Flash možnost vytvořit si vlastní rozhraní pomocí samostatné SWF animace, která se zobrazí na místě klasického rozhraní.

- Vytvořte si tedy novou animaci, jejíž rozměry volte podle klasické velikosti okna rozhraní.
- Dále vytvořte MovieClip (klidně i prázdný), vložte jej do animace a pojmenujte "***xch***"

  Povšimněte si, že neříkám "třeba", tento MC se musí jmenovat ***xch***, jinak to nebude fungovat!
- Veškeré proměnné adresujete do tohoto MC (***xch.paint***, ***xch.xscl***, ***xch.alp***, ***xch.col***)
- V okně "***Component Definition***" našeho původního SmartClipu u kolonky "***Custom UI***" klepněte na "***SET***" a v okně "Custom UI" vyplňte položky podle obrázku:



Zde máte na výběr, jestli má být rozhraní nalinkováno z ***vnějšku***, nebo z ***knihovny***. (v tomto příkladu jsem jej pro názornost ponechal vně souboru)

Dále je možné určit, kde se bude rozhraní zobrazovat (v ***Properties*** nebo ve ***vlastním okně***)

Nakonec správně napište cestu k SWF souboru.

- Takto nějak by to mohlo pak vypadat:

Nakonec jen uvedu odkaz na stránku, kde si můžete stáhnout spoustu hotových komponentů:
www.flashcomponents.net

Vlastní kurzor a duplikace MC

V této kapitole jsem se rozhodl spojit 2 příklady dohromady, protože jsou velmi jednoduché a krátké.

1. Vlastní kurzor

Poslední dobou se v mojí poště množí dotazy na vytvoření vlastního kurzoru. Dnes vám vysvětlím jeden způsob, jak ho vytvořit. Tento postup nenahrazuje ani tak kurzor samotný, ale namísto něj zobrazí určitý MovieClip

- Takže nejprve si vytvoříme MovieClip, který může obsahovat jak statickou, tak i pohyblivou grafiku. Je velmi důležité dbát na správnou pozici středu (symbol "+")

- Tento MovieClip vložíme na plochu a definujeme mu následující příkazy:

```
onClipEvent (load) {
            Mouse.hide();
            startDrag(this, true);
}
onClipEvent (mouseMove) {
            updateAfterEvent();
}
```

Předchozí příkazy znamenají, že po načtení animace se nejprve skryje kurzor myši (`Mouse.hide()`), jinak by totiž byl vidět spolu s MovieClipem

A následně se zahájí tažení našeho MC (this=tento), přičemž je je pozice kurzoru uzamčena do středu MC (Lock mouse to center)

Pokud bychom to nechali takto, aktualizovala by se pozice v závislosti na framerate, což by vyžadovalo použití vysoké framerate (18-25). My však použijeme příkaz `updateAfterEvent();`, který při jakémkoliv pohybu myši (***mouseMove***) aktualizuje proměnné ***_xmouse*** a ***_ymouse***, charakterizující pozici myši v animaci. Tím pádem bude pohyb kurzoru zcela plynulý a nezávislý na rychlosti přehrávání.

- To je vše. Jen mějte na paměti, že kurzor by neměl uživatele rušit, a už vůbec ne otravovat.

2. Duplikace MovieClipu

Jak jistě víte z kapitoly "Knihovna", je možné vytvářet velké množství instancí symbolů umístěných v knihovně, aniž by se zvětšila velikost animace a těmto instancím měnit řadu atributů.

Podobnou operaci s MovieClipem umožňuje příkaz `duplicateMovieClip()`. Asi bude nejlepší, když vám jeho funkci předvedu na následujícím příkladu:

- Řekněme, že máme na ploše umístěn MC s názvem "***prvni***"
- Dále do ***1. snímku*** animace umístíme následující příkaz:

```
duplicateMovieClip("prvni", "druhy", 1);
```

Tento příkaz vytvoří kopii MC "prvni" s názvem "druhy". Ta jednička na konci je tzv. ***Hloubka*** (Depth). Toto číslo určuje pozici MC v ose Z. Počáteční MC má depth=0 a tím pádem bude jeho duplikát "druhy" vidět nad ním. (Pokud by tam místo 1 bylo -1, tak by to bylo opačně)

- Efekt předchozího příkazu by byl prakticky nulový (kopie by byla identická a na stejném místě). Proto použijeme příkaz `setProperty()`. Například:

```
setProperty("druhy", _alpha, 50);
setProperty("druhy", _rotation, 15);
setProperty("druhy", _x, 130);
setProperty("druhy", _y, -50);
```

tyto příkazy nastaví duplikátu 50% průhlednost (alpha), otočí ho o 15° (rotation) a definují mu pozici ***_x=130*** a ***_y=-50***

Takže už víte, jak duplikace MC funguje a teď si vytvoříme trošku složitější příklad. Pokud bychom chtěli vytvořit vetší množství kopií s podobnými vlastnostmi, bylo by psaní příkazů docela zdlouhavé. My tedy použijeme smyčku.

- nejprve si tedy, jako v předchozím případě, vytvoříme MC, umístíme jej na plochu a pojmenujeme pro změnu "***bod***"
- ***Tomuto MC*** definujeme následující příkazy:

```
onClipEvent (load) {
          _root.xpos = _x;
          _root.ypos = _y;
}
```

Tento zápis znamená, že po načtení animace se do proměnných ***xpos*** a ***ypos*** zapíše počáteční pozice MC "bod" (***_x*** a ***_y***). Od této pozice se pak budou umísťovat kopie.

- Nyní můžeme do 1. snímku definovat následující příkazy (v ukázce jsem je, kvůli názornosti, definoval tlačítku)

```
var i = 1;
while (i<=450) {
       duplicateMovieClip("_root.bod", "bod"+i, i);
       setProperty("bod"+i, _x, _root.xpos + i);
       setProperty("bod"+i, _y, _root.ypos - Math.tan(i/20)*10);
       i++;
}
```

Zde je použita nejjednodušší smyčka (While). Na samém začátku je definována lokální proměnná i=1. Následuje smyčka, která vykonává příkazy dokud platí, že ***i <= 450*** (menší nebo rovno).

Je tedy duplikován MC ("***bod***") a jeho kopii je přiřazen název ***"bod"+i***.

Následuje nastavení vlastností nově vzniklého MC. V našem případě se definuje ***x*** souřadnice jako ***pozice počátečního MC + i***.

Dále je určena ***y*** souřadnice, tentokrát pomocí goniometrické funkce tangens (***Math.tan***).

Úplně nakonec se k proměnné ***i*** přičte jednička (++)

Pro názornost - 5. cyklus by se mohl nahradit tímto zápisem:

```
duplicateMovieClip("_root.bod", "bod5", 5);
setProperty("bod5", _x, _root.xpos + 5);
setProperty("bod5", _y, _root.ypos - Math.tan(0.25)*10);
```

- ***Pro šťouraly:*** výraz uvnitř funkce tangens je podělen 20 (zvětšení periody) a celá funkce je vynásobena 10 (zvětšení amplitudy).

Ovládání autíčka

Velký počet Flashových her je na motivy automobilových závodů. Základním stavebním kamenem je samozřejmě auto. To se dnes naučíme ovládat.

- Nejprve si vytvoříme MovieClip, který bude představovat ovládané auto a nazveme jej třeba "***car***". Auto samozřejmě bude zobrazeno z ptačí perspektivy.
- MovieClipu "car" poté definujte všechny následující příkazy:
- Použijeme událost "Enter Frame". Ta znamená, že všechny příkazy uvnitř budou vykonávány v každém snímku MovieClipu (i když je třeba jen jeden). Pro ty méně chápavé: příkazy se budou provádět pořád dokola podle rychlosti framerate animace.

```
onClipEvent (enterFrame) {
```

- Dále vytvoříme 2 hlavní podmínky pro stisk levé a pravé šipky. Po stisknutí šipky se automobil otočí o určitý počet stupňů a to změnou atributu ***_rotation***. Jelikož otáčíme MovieClip z něho samého, není třeba psát cestu (z hlavní osy by to muselo být takto: ***car._rotation***). Pokud bychom ovládali tank, stačilo by napsat jen ten jeden příkaz, my však jezdíme s autem (v klidovém stavu se nemůže otáčet) a tak je třeba přidat ještě podmínky, které upravují rotaci podle rychlosti.

```
   if (Key.isDown(Key.LEFT)) {
        if (Math.abs(speed)>=0.2 & Math.abs(speed)<=1) {
                _rotation -= 1;
        } else if (Math.abs(speed)>1 & Math.abs(speed)<=2) {
                _rotation -= 2;
        } else if (Math.abs(speed)>2 & Math.abs(speed)<=3) {
                _rotation -= 3;
        } else if (Math.abs(speed)>3) {
                _rotation -= 6;
        }
   }

   if (Key.isDown(Key.RIGHT)) {
        if (Math.abs(speed)>=0.2 & Math.abs(speed)<=1) {
                _rotation += 1;
        } else if (Math.abs(speed)>1 & Math.abs(speed)<=2) {
                _rotation += 2;
        } else if (Math.abs(speed)>2 & Math.abs(speed)<=3) {
                _rotation += 3;
        } else if (Math.abs(speed)>3) {
                _rotation += 8;
        }
   }
```

- Další podmínka kontroluje stisk šipek nahoru a dolů (zrychlování a brždění). Zde se po stisku patřičné šipky buď zvýší nebo sníží hodnota proměnné "***speed***". Povšimněte si, že když proměnná "***speed***" dosáhne 20, už se dál nezvyšuje, stejně tak, když je rovna -10 (zpátečka), už se dál nesnižuje.

```
if (Key.isDown(Key.UP) & speed<20) {
                speed += 1;
} else if (Key.isDown(Key.DOWN) & speed>-10) {
                speed -= 1;
```

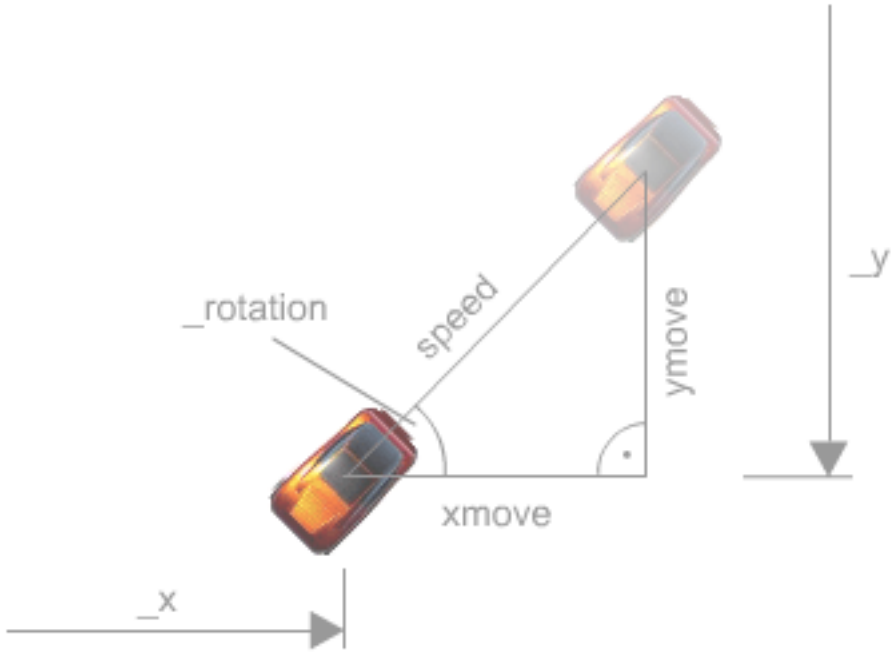
- Pokud bychom to nechali takto, auto by na neutrál drželo pořád svou rychlost. To je samozřejmě nereálné, a proto je nutné, pokud není zmáčknut plyn (šipka nahoru), postupně snižovat rychlost až na nulu.

```
} else {

          if (speed>0.5) {
                        speed -= 0.3;
          } else if (speed<-0.5) {
                        speed += 1;
          } else {
                        speed = 0;
          }
 }
```

- Tak a nakonec jsme se dostali k tomu nejdůležitějšímu, samotný pohyb auta. Co následující 4 řádky dělají? Jednoduše mění každý snímek pozici (_x a _y) MovieClipu "***car***".

```
 xmove = Math.cos((_rotation-90)*(Math.PI/180))*speed;
 ymove = Math.sin((_rotation-90)*(Math.PI/180))*speed;
 _x += xmove;
 _y += ymove;

}
```

- Nechci vám sahat do svědomí, ale věřím, že některé z vás polil studený pot při pohledu na goniometrické funkce ***sinus*** a ***cosinus***. Pro ty méně matematicky nadané jsem proto nakreslil následující obrázek:



Jak známo, sinus je v pravoúhlém trojúhelníku definován jako protilehlá odvěsna ku přeponě a cosinus jako přilehlá odvěsna ku přeponě.

Snad jen dodám, že do goniometrických funkcí se ve Flashi úhly dosazují ***vždy v radiánech***. V jednotkové kružnici platí:

$$\alpha^{rad} = \frac{\pi \cdot \alpha^{\circ}}{180^{\circ}}$$

- Poslední věc, kterou je třeba udělat, je přičíst hodnoty ***xmove*** a ***ymove*** k souřadnicím ***_x*** a ***_y***

Vlastní scrollbar

Každý asi ví, na co je ***scrollbar*** (česky ***posuvník*** případně ***rolovací lišta***). Ve Flashi MX je možné k textovému poli připojit posuvník z UI komponentů, ale tyto prvky jednak zbytečně zvětšují velikost animace a navíc pokud je chcete zobrazit ve FlashPlayeru 5 jednoduše ostrouháte.

My se naučíme vytvořit posuvník, který bude umět posouvat nejen obsah textového pole, ale také například měnit pozici instance MovieClipu.

Posuvník textového pole

Jistě víte, že textové pole typu "***Dynamic***" a "***Input text***" zobrazuje hodnotu definované proměnné. K této funkci se vztahují 2 příkazy, které nyní využijeme:

```
nazevpromenne.scroll
nazevpromenne.maxscroll
```

První jmenovaný prvek vyjadřuje ***aktuální číslo řádku***, které je zobrazené nahoře (tuto hodnotu lze změnit a tím vlastně text posunout) a druhý vyjadřuje opět číslo nejvyššího řádku, ale v případě, že již je vidět ***poslední řádek*** (v podstatě je to počet skrytých řádků). Tato hodnota přirozeně nelze měnit.

Jen dodám, že textové pole musí být typu multiline a text musí přesahovat, jinak by byly předchozí příkazy bezpředmětné.

- Takže nejprve vytvořte ***multiline dynamic text*** na ploše a definujte mu např. proměnnou "***text***" a následně do prvního snímku definujte tyto příkazy:

```
text = "dost dlouhý text.....";
length = 143;
```

  - proměnná "***length***" bude vyjadřovat výšku scrollbaru (můžete ji změnit podle potřeby) - viz dále
- Následně vytvořte 3 tlačítka (šípka nahoru, posuvník, šipka dolů) - v podstatě stačí 2 (šipky jsou stejné - jen otočené)
- Do třetice vytvořte movie clip a do něj šoupněte tlačítko posuvníku. ***Tomuto tlačítku*** (tedy nikoliv movie clipu) definujte tyto příkazy:

```
on (press) {
    startDrag("_root.scroller", false, _root.left, _root.top, _root.right, _root.
bottom);
    _root.dragging = true;
}
on (release, releaseOutside) {
    stopDrag();
    _root.dragging = false;
}
```

  Funkce "***StartDrag***" zahájí tažení movie clipu "***scroller***". Posun je však omezen 4 rozměry (v našem případě proměnné ***left***, ***right***, ***top*** a ***bottom***)

Zároveň je po uchopení definována proměnná "***dragging***" jako pravda (po puštění jako nepravda)

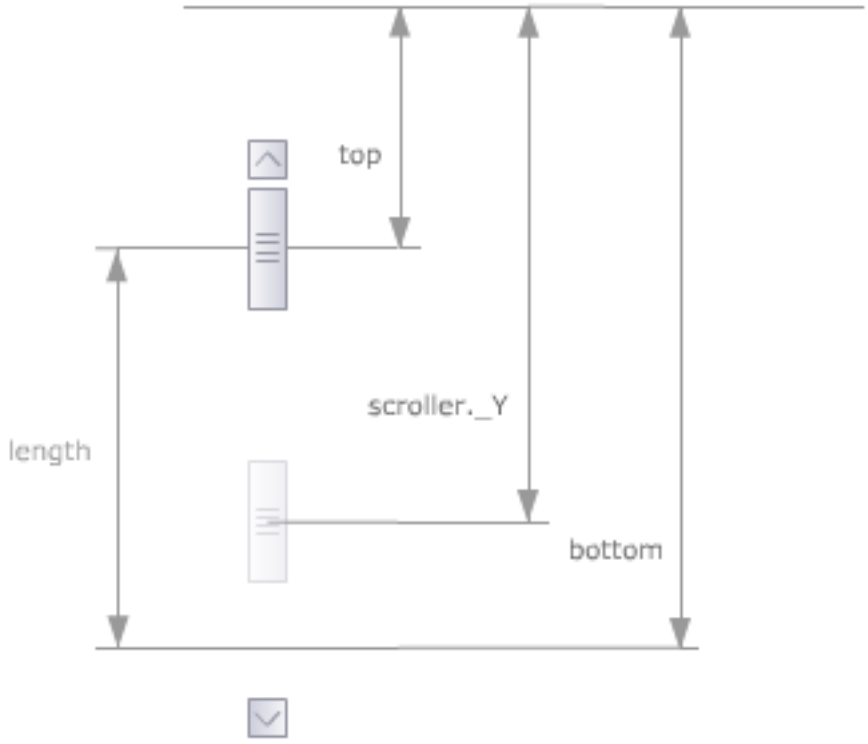
- Nyní umístěte MovieClip s tlačítkem na plochu, pojmenujte jej "***scroller***" a definujte mu tyto příkazy:

```
onClipEvent (load) {
    _root.koeficient = _root.length/(_root.text.maxscroll + 1);
    _root.top = _y;
    _root.left = _x;
    _root.right = _x;
    _root.bottom = _y+_root.length;
}

onClipEvent (enterFrame) {
     if (_root.dragging == true) {
             _root.text.scroll = Math.floor((_y - _root.top)/_root.koeficient);
     }
}
```

Po načtení MC "***scroller***" se vypočítá převodní koeficient (podle délky textu) jako podíl výšky celého scrollbaru a počtu skrytých řádků.

Dále jsou definovány proměnné vymezující posuv ***scrolleru*** tak aby byl možný posuv pouze po ose y na úsečce dlouhé v našem případě ***143 px***

Po uchopení ***scrolleru*** se potom vypočítá proměnná ***text.scroll*** jako zaokrouhlená hodnota podílu ***odjeté vzdálenosti*** od horní hranice posuvníku a ***koeficientu***.



- Další v pořadí jsou šipky:
  Horní definujte tyto příkazy:

```
on (release, keyPress "<Up>") {
    _root.text.scroll--;
```

```
    if (_root.scroller._y-_root.koeficient>_root.top) {
                _root.scroller._y -= _root.koeficient;
    } else {
                _root.scroller._y = _root.top;
    }
}
```

Po kliknutí (nebo šipka nahoru) odečte od proměnné ***text.scroll*** jedničku (tj. posune text směrem dolů)
Dále pokud by nebyla překročena horní hranice, posune ***scroller*** o hodnotu koeficientu nahoru, pokud ano nastaví ***scroller*** na horní hranici.

Dolní šipka funguje přesně opačně:

```
on (release, keyPress "<Down>") {
    _root.text.scroll++;

    if (_root.scroller._y + _root.koeficient < _root.bottom) {
              _root.scroller._y += _root.koeficient;
    } else {
              _root.scroller._y = _root.bottom;
    }
}
```

- Jako poslední krok ještě můžete dodělat design okna a je hotovo.

Movie Clip scrollbar

V minulém díle jsme se bavili o vytváření textového posuvníku. Využívali jsme vlastnosti proměnné scroll a maxscroll. Takto lze však lze scrollovat pouze čistý text a my chceme posouvat třeba i obrázky. Jelikož do textového pole nelze nacpat grafiku, budeme muset použít MovieClip. Podobně jako v předchozím případě, i zde je možné použít již hotový UI Komponent (ScrollPane), ale opět dojde k nepříjemnému zvětšení animace a její nekompatibilitě v přehrávačích 5 a nižších.

- Takže nejprve si musíte vytvořit ***MovieClip***, který bude tvořit ***obsah scrollovaného okna***. Zde se fantazii meze nekladou (může obsahovat prakticky cokoliv - i pohyblivou grafiku)
- Tento MC vložte na plochu a ***pojmenujte*** jej (třeba "window")
- Tomuto MC je nutné definovat následující příkazy:

```
onClipEvent (load) {
        _root.nullpos = _y;
}
```

Tento zápis znamená, že po načtení se nastaví hodnota proměnné ***nullpos*** jako Y souřadnice tohoto MC.

- Posléze si v knihovně vytvořte 3 tlačítka (šipka nahoru resp. dolů a vlastní scroller)
- Vytvořte další MovieClip, do kterého importujete instanci tlačítka scrolleru a celé to vložte na plochu a nazvěte třeba "scroller"
- Dále vložte na plochu instance šipek nahoru a dolů
- Do prvního snímku animace nyní definujte tuto proměnnou:

```
length = 143;
```

Tato proměnná charakterizuje ***výšku scrollbaru*** (rozdíl mezi horní a dolní pozicí v pixelech) Tuto hodnotu ***přizpůsobíte*** velikosti ***vašeho okna***.

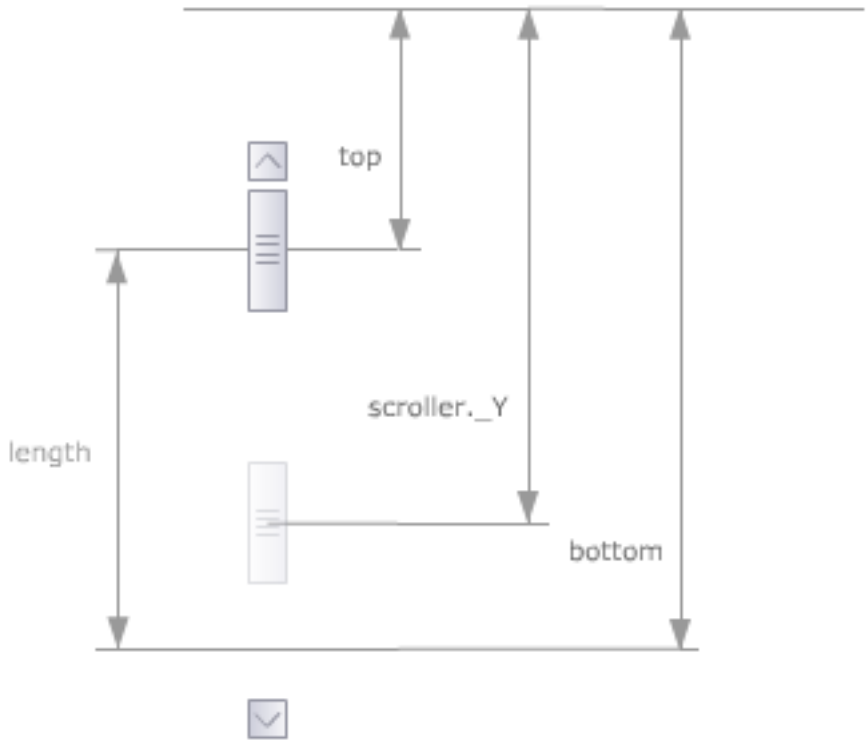
- Nyní vyberte instanci MC "scroller" a definujte jí tyto příkazy:

```
onClipEvent (load) {
      _root.koeficient = _root.length/(_root.window._height - _root.length -
1.5*_height);
      _root.top = _y;
      _root.left = _x;
      _root.right = _x;
      _root.bottom = _y+_root.length;
}

onClipEvent (enterFrame) {
      if (_root.dragging == true) {
            _root.window._y = _root.nullpos - (_y - _root.top)/_root.
koeficient;
      }
}
```

Tento zápis znamená, že po ***načtení*** (load) se definuje hodnota proměnné "***koeficient***" jako podíl ***výšky scrollbaru*** (definovaná v prvním snímku) a výškou MC "***window***" od které je odečtena ještě výška ***scrollbaru*** (lenght) a výška ***scrolleru*** (instance "scroller")

Dále jsou definovány 4 proměnné (top, left, right a bottom), které poté budou vymezovat pohyb ***scrolleru***

Dále po události "***EnterFrame***" a pokud je proměnná ***dragging = true*** vypočítána pozice posouvaného MC "***window***" jako ***počáteční pozice*** posouvaného MC mínus vzdálenost ***scrolleru*** od své maximální horní pozice (***top***) podělená koeficientem.



- Nyní si v knihovně otevřete MC "scroller" a v něm umístěnému tlačítku definujte tyto příkazy:

```
on (press) {
     startDrag("_root.scroller", false, _root.left, _root.top, _root.right,
_root.bottom);
     _root.dragging = true;
}
on (release, releaseOutside) {
     stopDrag();
     _root.dragging = false;
}
```

Zde je napsáno, že po ***kliknutí*** na tlačítko začíná ***tažení*** MC "***window***", které je však limitováno 4 rozměry (v našem případě jsou místo nich proměnné ***left***, ***right***, ***bottom*** a ***top***). Dále se nastaví hodnota proměnné ***dragging*** jako pravda (true).

Po ***puštění tlačítka*** se tažení ukončí a ***dragging*** se nastaví jako false.

- Nakonec zprovozníme šipky nahoru a dolů. Horní šipce definujeme tyto příkazy:

```
on (release, keyPress "<Up>") {
     if (_root.scroller._y-10*_root.koeficient>_root.top) {
```

```
                _root.scroller._y -= 10*_root.koeficient;
                _root.window._y += 10;
        } else {
                _root.scroller._y = _root.top;
        }
}
```

Toto znamená, že po ***kliknutí*** nebo po ***stisknutí šipky nahoru*** se ***scroller*** posune o desetinásobek koeficientu nahoru a ***okno*** se posune o deset pixelů dolů, ale pouze v případě, že by se tímto posunem nedostal ***scroller*** mimo svou dráhu (nad pozici ***top***).
Pokud ano zůstane scroller ve své horní pozici a šmitec.

- Dolní šipka funguje přesně opačně:

```
on (release, keyPress "<Down>") {
        if (_root.scroller._y+10*_root.koeficient<_root.bottom) {
                _root.scroller._y += 10*_root.koeficient;
                _root.window._y -= 10;
        } else {
                _root.scroller._y = _root.bottom;
        }
}
```

- Úplně nakonec ještě můžete dodělat design dráhy scrolleru a přidat masku pro okno.

Preloader

Nejdříve k čemu preloader je. Možná jste se setkali s problémem, že u velké animace spouštěné z internetu dochází prvních 10-30 sekund k nepříjemnému trhání. Je to způsobeno tím, že se animace přehrává rychleji, než se stačí načítat přes pomalou modemovou linku.

Preloader animaci zastaví na začátku a pustí ji dál až je celá načtená.

My si teď společně vytvoříme preloader, který ukazuje načtené procenta.

Nejprve si vytvoříme MovieClip do kterého nakreslíme rámeček, který se potom bude zvětšovat během načítání:

Zde se fantazii meze nekladou. Je velmi důležitá pozice středu (symbol +). Pokud budeme chtít, aby se obdélník zvětšoval směrem doprava, musíme ho nakreslit tak, aby byl střed na jeho levém konci.

Nyní si 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:

```
loading = Math.round(getBytesLoaded()/getBytesTotal()*100);
setProperty(prubeh, _xscale, loading);

if (loading == 100) {
     play();
} else {
     gotoAndPlay(1);
}
```

Na samém začátku je definovaná proměnná "***loading***" jako zaokrouhlená hodnota (***Math.round***) výrazu: "***načtená data/ celková data * 100***".

Na dalším řádku je definována procentuální šířka instance "***prubeh***" jako hodnota proměnné "***loading***".

Nakonec následuje podmínka, že pokud dosáhne proměnná "***loading***" hodnoty 100 má pokračovat přehrávání, pokud ne, má se vrátit na snímek 1.

Do animace ještě můžete vložit ***Dynamic Text***, který bude zobrazovat hodnotu proměnné "***loading***" a případně orámovat načítací pásek.

Nakonec ale ***nezapomeňte*** definovat instanci "***prubeh***" šířku ***0%***, jinak by se na začátku načítání objevila v plné velikosti.

Postupné vypisování textu

Chtěli jste si někdy vytvořit animaci psaní textu ala Matrix? Pokud jde o pár slov, není problém, ale rozdělovat celý odstavec na písmenka a skládat ho postupně dohromady opravdu není to pravé ořechové.

Naštěstí je možné text vypisovat scriptově:

Nejprve si vytvořte jeden klíčový snímek a roztáhněte jej na velikost minimálně 3 snímků. Do něj vložte Dynamic Text Multiline, který bude zobrazovat proměnnou "okno"

Poté vytvořte novou vrstvu a vytvořte 3 klíčové snímky.

Do 1. snímku zadejte tyto příkazy:

```
okno = "";
krok = 0;
```

Do 2. snímku pak tyto:

```
krok++;
play();
```

3. snímek je nejdůležitější:

```
pismeno = substring("Nějaký text._Text na novém řádku******>", krok, 1);

if (pismeno == "_") {
      pismeno = newline;
} else if (pismeno == "*") {
      pismeno = "";
}

if (pismeno == ">") {
      play();
} else {
      okno = okno+pismeno;
      prevFrame();
}
```

A teď co tohle všechno znamená.

V 1. snímku se definuje proměnná "***okno***" jako prázdná a "***krok***" roven nule.

Ve 2. snímku se proměnná "***krok***" zvýší o 1 (++).

Ve 3. snímku se nejdříve definuje písmeno, které se bude tento cyklus připisovat a to pomocí funkce "***substring***".

Funkce ***substring*** dělá to, že z textu v uvozovkách vyřízne od určitého místa určitý počet znaků. Příklad:

```
vyrez = substring("Macromedia Flash MX", 5, 9)
```

proměnná "***vyrez***" teď bude mít hodnotu "***media Fla***". Bylo tedy vyříznuto od 5. znaku devět znaků

Takže teď máme definované písmeno a nyní následuje podmínka, která v případě, že bude ***písmeno bude rovno "_"*** skočí na ***nový řádek*** (newline), a nebo pokud ***bude rovno "*"*** nebude se toto kolo přidávat žádné písmeno (***vytvoření pauzy***)

A nakonec se provede kontrola, jestli je přidávané písmeno rovno ">". Pokud ano, psaní je u konce, pokud ne, je k proměnné "***okno***" je připsáno ***písmeno*** a přehrávání skočí na předchozí snímek, kde ho příkaz ***Play()*** znovu spustí.

Ovládání projektoru

Pokud vkládáme Flash animaci do HTML stránky, můžeme ovlivnit vlastnosti zobrazení pomocí HTML tagů. Pokud však chceme definovat tyto vlastnosti v EXE projektoru, musíme použít příkaz `fscommand()`.

Příkaz fscommand může mít tyto atributy:

- ***fullscreen*** (true/false)
  určuje, zda má být animace přehrávána v módu přes celou obrazovku
- ***allowscale*** (true/false)
  Flash animace má přednastavené, že při změně velikosti okna přehrávače se přizpůsobí jeho obsah, to lze pomocí této funkce vypnout a zaručit tak zobrazení 1:1
- ***showmenu*** (true/false)
  Povoluje nebo zakazuje zobrazení rozšířeného menu po kliknutí pravým tl. a zároveň umožňuje skrýt panel nabídek projektoru.
- ***trapallkeys*** (true/false)
  Umožňuje zablokovat klávesové zkratky Flash Playeru.

Ctrl + Q	ukončí animaci a zavře okno playeru
Ctrl + W	ukončí animaci, okno playeru zůstane otevřené
Ctrl + O	vybrat novy swf k přehrání
Ctrl + F	přepne na fullscreen
Ctrl + H	přepne zobrazení na High Quallity
Ctrl + Enter	spustí přehrávání animace
Ctrl + R	spustí přehrávání znovu od začátku
Ctrl + >	o jeden snímek dopředu
Ctrl + <	o jeden snímek zpět

- ***quit***
  zavřít projektor
- ***exec*** (cesta do aplikace)
  umožňuje posílat příkazy do projektoru nebo do HTML browseru

Příklad komunikace Flash animace a JavaScriptu

Do HTML stránky zkuste vložit tento JavaScript:

```
function theMovie_DoFSCommand(command, args) {
     if (command == "messagebox"){
               alert(args);
     }
}
```

Do Flash animace potom tyto příkazy:

```
fscommand("messagebox","Toto je box zpráv volávaných z Flash.")
```