# DUM č. 17 v sadě

# 28. Inf-4 Jednoduchá hra Had ve Flashi (ActionScript)

Autor: Robert Havlásek

Datum: 08.06.2013

Ročník: 5AV

Anotace DUMu: Flash - teorie: Spojení proměnné uvnitř skriptu s dynamickým textem na ploše. Komponenty pro interakci s uživatelem.

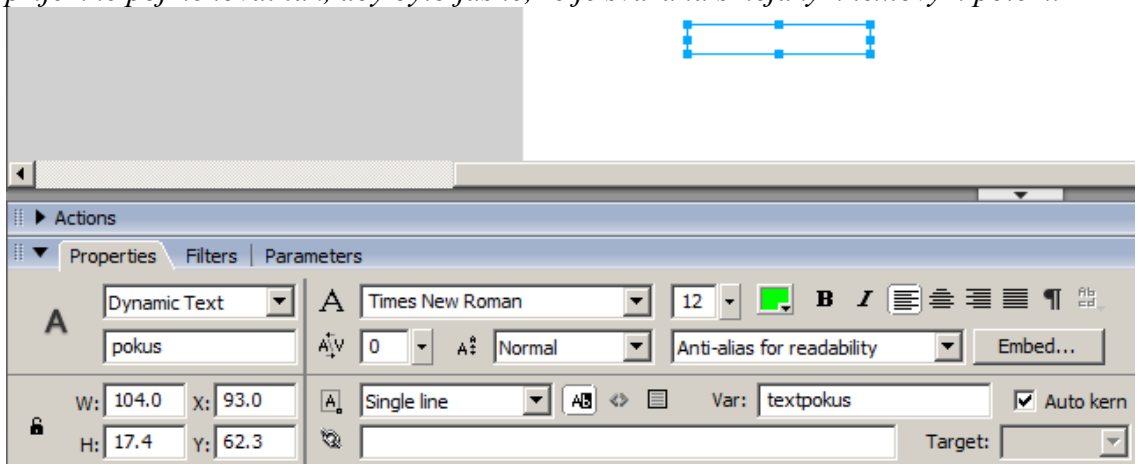


evropský sociální fond v ČR
EVROPSKÁ UNIE
MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY
OP Vzdělávání pro konkurenceschopnost
INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ

Dosud jsme v programu zobrazovali pouze ladící informace, a to pomocí funkce trace. V tomto DUMu si ukážeme, jak komunikovat s uživatelem, jak mu zobrazovat při běhu programu informace v textových polích a naopak jak od uživatele informace získat.

## *Spojení textového pole s proměnnou*

V ploše vyrobíme text pomocí nástroje A, můžeme do něj napsat nějaké znaky, ale taky nemusíme. Důležité je této textové komponentě v panelu Properties přepnout, že zobrazuje dynamický text (výběrem „Dynamic Text“ vlevo) a nastavit jí (v poli „Var:“) proměnnou, se kterou se tento text sváže – kdykoliv se nastaví/změní hodnota uvedené proměnné, text se překreslí. V obrázku níže je text svázán s proměnnou textpokus. Navíc jsme nastavili Instance name celého objektu na pokus.
*Pedagogická poznámka: I když je Instance name pro prosté zobrazování textu v podstatě zbytečná, doporučuji nastavovat obě jména – do „Var“ dáváme text, pomocí „Instance name“ měníme vlastnosti (barvu, font, umístění, zarovnání, ...). Proměnnou „Var:“ je příjemné pojmenovat tak, aby bylo jasné, že je svázaná s nějakým textovým polem.*




Do kódu plochy napíšeme jediný řádek:

```
textpokus="Toto je náš první zobrazený text na GML...";
```

Program spustíme, kód plochy se provede a my hned vidíme první zádrhel – textové pole je příliš krátké a celý text se do něj nevleze. Pomůže nastavit textovému poli vlastnost `autoSize` na jednu z hodnot `"left"`, `"right"` a `"center"`. Text je jednak zarovnán (k levému okraji, doprava nebo doprostřed) a druhak je v případě potřeby pole zvětšeno, takže se do něj text vleze celý. Funguje to i „zpětně“ (pokud nejprve text přiřadíme a pak `autoSize` nastavíme).
Vyzkoušíme:

```
textpokus="Toto je náš první zobrazený text na GML...";
_root.pokus.autoSize="left";
```

Další významnou vlastností textového pole je jeho schopnost zobrazovat HTML text – nastavíme-li mu vlastnost `html`.
Vyzkoušíme:

```
textpokus="Toto je náš <b>první</b> zobrazený text "+newline+"na <a
tref='http://www.gml.cz/'>GML</a>...";
_root.pokus.autoSize="left";
_root.pokus.html=true;
```

Funkce `newline` do řetězce vloží znak pro nový řádek.

*Pedagogická poznámka: V různých verzích Flashe je různá defaultní hodnota pro vlastnost autoSize i pro vlastnost html. Třeba v námi používané Flash 8 je autoSize=" a html=true.*

Z dalších významných vlastností jmenujme `background` (zda je viditelné pozadí za polem), `backgroundColor` (barva, která je vidět), `border` (zda je vidět okraj pole), `borderColor` (barva okraje). Vlastnosti `hScroll` a `scroll` určují, zda textové pole umožňuje horizontální a vertikální skrolování (=posouvání posuvníkem) v případě, že text není vidět celý.
I proměnná, kterou jsme při návrhu zadávali staticky za „Var:“, lze dynamicky nastavit/změnit pomocí vlastnosti `variable`.

Celé textové pole lze vyrobit dynamicky, za běhu programu, příkazem plochy

```
_root.createTextField("jmeno", hloubka, pozice_x, pozice_y, sirka, vyska);
```

## *Vstupní a výstupní formulářové prvky*

Paleta obsahující formulářové prvky se jmenuje Components a nebývá v standardním rozložení pracovní plochy zobrazena. V menu zvolíme Window – Components (klávesová zkratka CTRL+F7).
Na obrázku vidíme (shora):

Label – popisek, obsahuje pouze text

Button (Tlačítko) – s textem a možností kliknutí uživatelem

ComboBox – umožňuje jednořádkový výběr z několika položek.

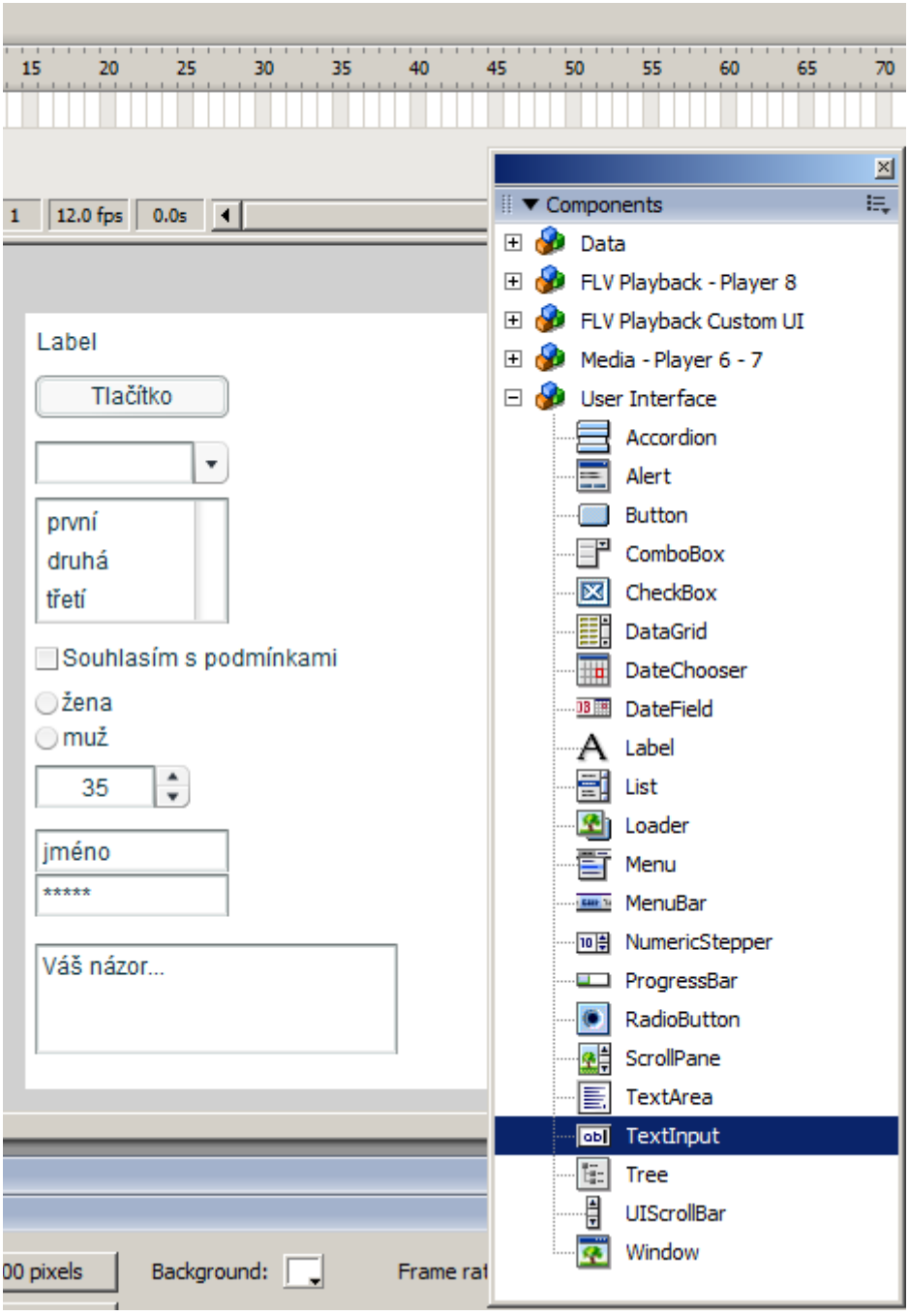
List – rovněž výběr několika položek, ale víceřádkový.

CheckBox – „zatržítko“, používá se pro jednu volbu ano/ne.

RadioButton – umožňuje výběr jedné možnosti ze skupiny nabízených možností.

TextInput – umožňuje zadat uživateli jednořádkový text. Vyrobil jsem jej zde dvakrát, podruhé jsme mu ve vlastnostech nastavili, že má skrývat znaky (např. v případě zadávání hesla).

TextArea – umožňuje zadat uživateli víceřádkový text.



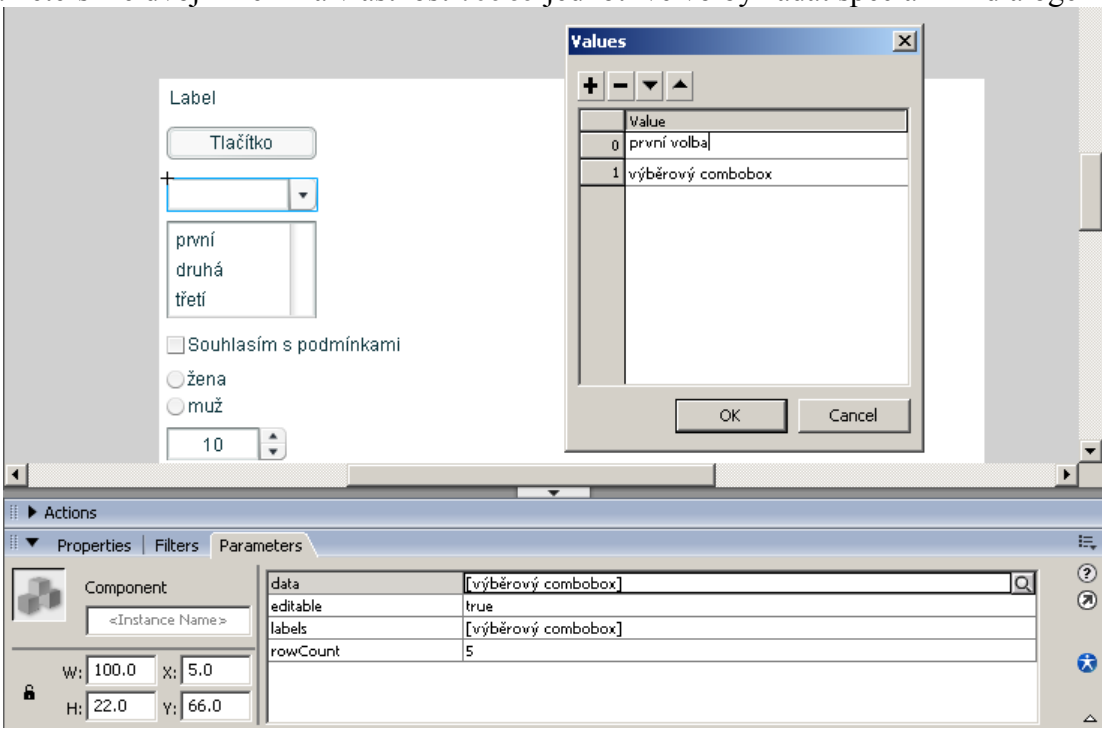
## *Vlastnosti a metody fomrulářových prvků*

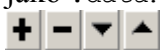
Všechny formulářové prvky mají vlastnost `.visible` (udává, zda je prvek vidět) a `.enabled` (zda je aktivní, lze tedy s ním aktivně pracovat; není „zašedlý“).

Label obsahuje vlastnost `.text`, která udává, co daný label zobrazuje. Významnou vlastností je též `.autosize`, která (podobně jako u textového pole na první straně tohoto DUMu) říká, zda se velikost labelu bude zvětšovat, dáme-li do něj moc dlouhý text. Totožná je též vlastnost `.html`, která říká, umí-li label zobrazovat hypertext s HTML značkami.

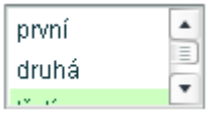
Button obsahuje vlastnost `.label` – co je na tlačítko napsáno. Vlastnost `.labelPlacement`, jež udává, jak bude text v rámci tlačítka zarovnaný ("left", "right", "top", "bottom").
*Pozn: Vlastnosti .toggle a .selected studentům obvykle zatajím; pokud se zeptají, doporučím nepoužívat – připadá mi nešťastné, když se tlačítko chová jako CheckBox.*

ComboBox obsahuje ve vlastnosti `.data` položky, z nichž lze jednu vybrat. Obvyklé je jejich zadání už při tvorbě programu – vybereme-li nějaký combobox, v panelu Properties v sekci Parameters lze dvojklikem na vlastnost `.data` jednotlivé volby zadat speciálním dialogem:




Podobně funguje vlastnost `.labels`, která by měla mít stejný počet položek – zatímco `.data` obsahuje (strohé) informace pro program, `.labels` obsahuje informace pro uživatele, obvykle košatější, několik slov, s diakritikou… Typicky `combobox.data` je `[1,2,3]`, `combobox.labels` je `[první, druhá, třetí]`. Pokud `.labels` neuvedeme, mají stejnou hodnotu jako `.data`.
Tlačítka  fungují pro přidání položky do comboboxu, její odebrání a přesun označené položky níž nebo výš.
Chceme-li položky ComboBoxu zadávat/měnit/mazat při běhu programu, lze na to použít metody `addItem()` (přidá novou položku na konec), `addItemAt()` (přidá novou položku na zadané místo, položky pod ní posune o jedno níž), `removeItemAt()` (odebere položky ze zadaného místa), `replaceItemAt()` (zamění položku na zadaném místě za jinou) či `getItemAt()` (vrátí hodnotu položky na zadaném místě). *Pozn.: Obvykle se studenti spokojí s výčtem těchto metod, neukazujeme si, jak fungují.*

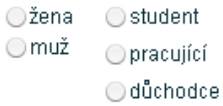
Pro programátory je nejzajímavější událost `Combobox.change`, která se volá v okamžiku, kdy si uživatel vybere nějakou z položek. Můžeme tak na jeho výběr hned reagovat.

List je (co do vlastností) velmi podobný komponentě Combobox, vizuálně se ale liší – zobrazuje víc řádků. Počet položek, které zvládne zobrazit, je dán fyzickou velikostí komponenty, pokud ji pomocí nástroje Free Transform Tool zvětšíme či zmenšíme, položky budou vidět třeba jen částečně, například: 

Vlastnosti a metody má List stejné jako ComboBox, navíc disponuje vlastností `multipleSelection` (defaultně ovšem nastavenou na false) –nastavení na true uživateli umožňuje vybírat z Listu víc položek najednou (pomocí Ctrl+klikání nebo Shift+klikání).

CheckBox má kromě vlastnosti `.selected` (v níž udává, zda je právě zatržený nebo ne) ještě vlastnost `.labelPlacement`, která určuje místo, kde je text umístěný (vlevo od čtverečku, vpravo od čtverečku, nahoře nad čtverečkem nebo dole pod zaškrtávacím čtverečkem).

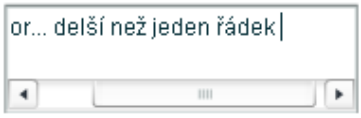
U RadioButtonu je patrně nejdůležitější vlastnost `.groupName`, pomocí níž lze ovlivnit, které radiobuttony budou tvořit jeden celek a které zase jiný. Například:



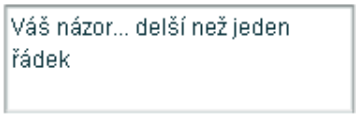

Abychom si mohli vlevo vybrat jednu ze dvou možností a vpravo jednu ze tří možností, musíme mít v levých radiobuttonech jinou `.groupName` než v těch pravých.

TextInput má vlastnost `.text` (co je v něm napsáno), vlastnost `.editable` (zda uživatel smí do textinputu právě psát) a vlastnost `.password` (zda náhodou nepíšeme heslo – v tom případě uživatel nevidí vlastní znaky, ale vidí hvězdičky).

TextArea má rovněž vlastnost `.text` (která navíc může obsahovat i znaky konce řádku) i vlastnost `.editable`. Má i vlastnost `.html` (jestli se celý text interpretuje jako hypertext – viz 1. strana dole). Jako poslední zmíníme vlastnost `.wordWrap` (defaultně je true), která povoluje či zakazuje zalamování řádků, pokud je text delší než šířka textarea komponenty. Nastavíme-li `.wordWrap` na false, přibude u delších textů horizontální posuvník:



,
při nastavení `.wordWrap` na true delší text rovnou zalomí:




Vzhled popisovaných formulářových prvků a jejich funkce jsou standardizované pro všechna zařízení a programy, není vhodné je používat k jinému účelu (např. radiobutton jako „zaškrtávátko“ typu ano/ne, případně Button jako „zaškrtávátko“ pomocí Button.toggle).

*Pedagogická poznámka: Výčet formulářových prvků je zde velmi dlouhý a ještě ne úplný (vynechali jsme Menu, MenuBar, Datagrid, Loader, ...), obvykle jej provádím rychlou frontální výukou. S případným příkladem na konci. V příštím DUMu některé prvky použijeme.*

## *Komplikace se zachytáváním událostí on(keyPress)*

V naší hře Had jsme do této chvíle neměli žádnou interaktivní komponentu. Z toho důvodu chytaly událost keyPress všechny ostatní symboly. Umístíme-li do animace některou z výše uváděných komponent (tlačítko, checkbox, …), Flash přestane keyPress posílat všem a začne jej posílat pouze aktivní komponentě (tj. té, co má focus).

Řešení této situace je několik. Jednak můžeme symbol `_root.hlava` „zaregistrovat“ jakožto příjemce klávesových událostí keyPress – funkcí `addEventListener`, která má dva parametry: string popisující událost, v našem případě "keyPress", a jméno komponenty, která ji má nově zachytávat. Případně místo `addEventListener("keyPress",_root.hlava)` můžeme použít funkci objektu key: `Key.addListener(_root.hlava);` .

Jednodušší je ale přestat používat obsolentní `on (keyPress "<Left>") {smer=2;}` a místo těchto čtyř příkazů začít používat testování stisknuté klávesy (pomocí objektu `Key` a jeho funkce `isDown`) přímo v `onClipEvent(enterFrame).`
Kód hlavy by pak začínal:

```
onClipEvent(enterFrame)
 { if (Key.isDown(Key.LEFT)) smer=2;
   if (Key.isDown(Key.RIGHT)) smer=0;
   if (Key.isDown(Key.UP)) smer=1;
   if (Key.isDown(Key.DOWN)) smer=3;
   if (smer==0) {_root.hlava._x = _root.hlava._x + 10;}
   …
```

Výhodou tohoto přístupu je, že můžeme (třeba v jiné hře) v rámci jednoho enterFrame testovat víc stisknutých kláves zároveň, případně na to reagovat (např. pohybem po diagonále při stisknutých dvou kurzorových šipkách).