Zpravodaj Československého sdružení uživatelů TeXu

Robert Mařík

Vkládání JavaScriptů pdfTeXem prakticky

Zpravodaj Československého sdružení uživatelů TeXu, Vol. 17 (2007), No. 2, 72–83

Persistent URL: http://dml.cz/dmlcz/150034

Reference

[1] Taco Hoekwater and Hans Hagen. *LuaTEX reference manual*, 2007. http://context.aanhet.net/luatex/snapshot/manual/luatexref-t.pdf.
[2] Adobe systems Incorporated. *PDF reference manual, v. 1.7*, 6 edition, 2006. http://www.adobe.com/devnet/acrobat/.
[3] Hàn Thế Thành, Sebastian Rahtz, Hans Hagen, and Hartmut Henkel. *The pdfTEX user manual*, 2007. http://sarovar.org/docman/view.php/106/66/pdftex-s.pdf.

Summary: Using pdfTEX V: current typesetting position

One of the extension by which pdfTEX improved TEX is a possibility to obtain a current typesetting possition. It can be used for placing objects later. This article brings a description of related primitives and examples of their usage.

Vít Zýka vit.zyka@seznam.cz

Vkládání JavaScriptů pdfTEXem prakticky

Robert Mařík

Úvod

V poslední době se na CTAN objevuje stále více balíčků, využívajících vkládání JavaScriptů do PDF dokumentů (animate, animfig, cooltooltips, dps, eCards, exerquiz, fancytooltips, jeopardy, jj_game, pdfanim)

JavaScripty umožňují získat dokumenty, které mají větší možnosti než klasické statické dokumenty, jejich funkčnost se však projeví pouze v prohlížeči Adobe Reader nebo Adobe Acrobat (což je poněkud omezující pro uživatele alternativních programů na čtení PDF).

Cílem tohoto článku je ukázat několik jednoduchých příkladů na vkládání JavaScriptů, které by po příslušném rozšíření umožnily dosáhnout podobných možností, jaké mají výše uvedené balíčky. Budeme se přitom vždy snažit udělat příklady názorné a co nejjednodušší. Článek je doprovázen souborem `testjs.tex`, který obsahuje všechny popisované JavaScripty a některé doplňující komentáře a souborem `testjs.pdf`, který vznikl překladem pdfLATEXem. Tyto soubory jsou

dostupné na `user.mendelu.cz/marik/latex/testjs.zip`. Je více než vhodné mít soubor `testjs.pdf` při čtení kapitoly Praktické ukázky otevřený a popsané funkce při čtení hned zkoušet.

Vkládání JavaScriptů je možno realizovat v programu pdfTEX pomocí primitivu `\pdfstartlink` ... `\pdfendlink`. My však budeme až na dvě výjimky využívat poněkud komfortnější a dobře dokumentovaný balíček `eForms`. V této souvislosti bych rád upozornil, že k dnešnímu dni (duben 2007) byl tento balíček v repozitářích MiKTEXu a na CTAN neaktuální a díky tomu například balíček `fancytooltips` nepracoval korektně. Pro vlastní experimentování proto doporučuji stažení poslední verze balíčku `eForms`, který je na `www.acrotex.net` jako součást AcroTeX eDucation Bundle.

Jistou nevýhodou při vkládání JavaScriptu pomocí pdfTEXu a prohlížení následného souboru pomocí volně šiřitelného programu Acrobat Reader je, že uživateli se nezobrazují případné chyby, ke kterým dojde v JavaScriptech při práci s PDF dokumentem. Pouze vidíme, že se očekávaná akce neprovedla nebo nedokončila, ale nemáme informaci o tom, na kterém místě došlo k chybě. Při vytváření komplikovanějších aplikací je proto neocenitelnou pomůckou Adobe Acrobat Professional. Tento program umožňuje editovat a odladovat JavaScripty, hlásí případné chyby a umožňuje volat potřebné funkce a kontrolovat hodnoty proměnných prostřednictvím JavaScript konzole.

Začínáme od hlavičky

V textu článku si okomentujeme jednotlivé části souboru `testjs.tex`, který používá pdfLATEXem vložené JavaScripty. Ukážeme si techniky, které umožní přiřazovat akce tlačítkům, pracovat s matematickými výrazy, zjišťovat polohu myši na stránce a pracovat s jednoduchými animacemi. Přitom upozorníme, jak se podobná technika používá v některém z balíčků vyjmenovaných v úvodu.

Nejprve založíme hlavičku souboru, načteme nějaký styl formátující dokument pro čtení na obrazovce a balíčky, které budeme potřebovat.

```
\documentclass{article}
\usepackage[pdftex,screen,designii]{web}
\usepackage[pdftex]{eforms}

\usepackage[czech]{babel}
\usepackage[IL2]{fontenc}
```

Makra ze stylu `exerquiz.sty` nám poslouží pro pohodlnou práci s matematickými výrazy. Knihovna `dljslib.sty` umožní vynechávat hvězdičku v násobení a umožní `2*x` zapsat jako `2x`. Poté založíme prostředí, kam vepíšeme proměnné a funkce, které budeme používat v JavaScriptech dokumentu:

```
\usepackage[czech,pdftex,noxcolor]{exerquiz}
\usepackage[ImplMulti]{dljslib}
\begin{insDLJS}[BulletinCsTug]{bull}{bulletin}
var shiftAN;
var cislo=1;
```

Práce s matematikou

Funkce `pocitej` zjistí, zda je v políčku jménem `vstupx` číslo. Pokud ne, vypíše varovné hlášení a skončí. Pokud ano, vezme řetězec z políčka jménem `funkce` a zpracuje ho funkcemi `ParseInput` a `stripWhiteSpace` definovanými v balíčku `exerquiz`. Tyto funkce zajistí například převod textu tvaru `\sin^2 (|2x|)` na matematický výraz použitelný v JavaScriptech dokumentu, nastaví proměnnou `ok2Continue` definovanou opět balíčkem `exerquiz` na false pokud je tento převod neúspěšný (například nepárová závorka) a pokud je převod úspěšný, uloží se výsledný řetězec do textové proměnné `funkce`. Pokud byl převod neúspěšný, funkce skončí. Jinak dosadíme hodnotu políčka `vstupx` do této funkce a odchytáváme přitom konstrukcí `try catch (e)` případné chyby (způsobené například použitím jiné proměnné než x). Výsledek zobrazíme v políčku `fx`. Výpočty jsou uvnitř konstrukce `with(Math)`, při zadávání funkce stačí tedy psát `sin` místo `Math.sin`:

```
function pocitej()
{
  if(isNaN(this.getField("vstupx").value))
  {app.alert("Zadej cislo."); return null;}
  var funkce=this.getField("funkce").value;
  ok2Continue=true;
  funkce=stripWhiteSpace(funkce);
  funkce=ParseInput(funkce);
  pole=this.getField("fx");
  if (!ok2Continue)
  {
    pole.value="Opravte funkci";
    return null;
  }
  with(Math){
    var x=this.getField("vstupx").value;
    try { var _f=eval(funkce); pole.value=_f; } catch (e)
    { pole.value="Chyba!"; };
  }
};
```

Zjišťování polohy myši na stránce

Nadefinujeme funkci `CtiSouradnice`, která vypočte vzdálenost kurzoru myši od počátku soustavy souřadnic (v levém dolním rohu) a sestaví řetězec obsahující informaci o pohlaví uživatele, poloze myši, vzdálenosti od počátku a zda bylo při stisknutí tlačítka volajícího tuhle funkci stisknuto tlačítko `Shift` (tato informace je předávána v proměnné `shiftAN`). Výsledek se zobrazí jako několikařádkový text v políčku s názvem `souradnice` a pokud je vzdálenost kurzoru od počátku dostatečně velká, bude text červený, jinak bude černý:

```
35  function CtiSouradnice()
    { with (Math){
        var vzdalenost=round(sqrt(pow(mouseX,2)+pow(mouseY,2)));
      }
      hlaseni="Jste "+this.getField("pohlavi").value+"."+"\n"+
40    "Mate mys na souradnicich "+mouseX+" a "+mouseY+"."+"\n"+
      "Vzdalenost od stredu je "+vzdalenost;
      if (shiftAN)
         hlasenib="stisknuty."; else hlasenib="pusteny.";
      hlaseni=hlaseni+"\n"+"Shift byl pri kliknuti "+hlasenib;
45    this.getField("souradnice").value=hlaseni;
      if (vzdalenost>200)
      this.getField("souradnice").textColor=color.red;
      else this.getField("souradnice").textColor=color.black;
    };
```

Ovládání animace

Funkce `animuj` je použita k ovládání animace o pěti snímcích. Aktuální snímek je v proměnné `cislo`, která je přístupná a společná v rámci celého dokumentu. Funkce posune proměnnou `cislo` na další snímek nebo zpět na začátek animace a umístí ikonu z tlačítka `ikona.cislo` do tlačítka `animace`. Poté ukončíme prostředí, kde definujeme funkce a proměnné.

```
    function animuj()
    {
      cislo=cislo+1;
      if (cislo==6) cislo=1;
55    f=this.getField('animace');
      g=this.getField("ikona."+cislo);
      f.buttonSetIcon(g.buttonGetIcon());
    };

60  \end{insDLJS}
```

Praktické ukázky

Tvoříme jednoduchá tlačítka

Balíček `eCards` umožňuje vytvářet jistou elektronickou analogii oboustranně popsaných paměťových kartiček, které někteří studenti cizích jazyků používají při výuce cizích slovíček. V dokumentu jsou umístěny strany s otázkou, strany s odpovědí a případně strany s nápovědou. Uživatel má k dispozici tlačítko, po jehož stisknutí přejde dokument na náhodně vybranou otázku. Z této otázky vedou hypertextové odkazy na nápovědu k řešení a k řešení. Poté je možno volit další náhodnou otázku. Hypertextové odkazy dokáže vložit každý zkušenější uživatel LaTeXu, podíváme se proto na problém, jak vložit do textu tlačítko a asociovat mu akci JavaScriptu.

Definujme příkaz, který vytvoří žluté (příkaz `\BG{1 1 0}`) tlačítko s asociovanou akcí, která se provede při stisku tohoto tlačítka (příkaz `\A{\JS{...}}`). Toto tlačítko umístíme ručně na konci každé strany. Kliknutí na tlačítko na první straně způsobí přechod na druhou stranu[1], kliknutí na druhé straně způsobí přechod na stranu první a na dalších stranách je náhodně vybrána první nebo druhá strana a dokument přejde na tuto náhodně vybranou stranu:

```
\def\stranka{\null\par\vfill
  \pushButton[\BG{1 1 0}\A{\JS{
      str=this.pageNum;
      if (str==0)
      this.pageNum=1;
      else
      {
        if (str==1) this.pageNum=0;
        else
        this.pageNum=Math.floor(Math.random()*2);
      }
    }}]{stranka}{6cm}{10bp}}
```

Tlačítka jsou hojně využívána v balíčku `jeopardy`, který simuluje hru známou u nás jako Riskuj. Na obrazovce je vytvořeno hrací pole z tlačítek a pod každým tlačítkem je textové pole neviditelné pro TeX(v hboxu šířky nula). Tlačítkům na hrací desce jsou přiřazeny akce, které při stisknutí toto tlačítko skryjí a způsobí odskok na stránku s příslušnou otázkou. Na stránce s otázkou je každá z nabízených odpovědí uvozena tlačítkem, po jehož stisknutí se přičte či odečte bodová hodnota otázky od celkového počtu bodů (podle toho zda je odpověď správná nebo špatná) a zobrazí se opět stránka s hrací deskou. Tím, že jsme skryli tlačítko vedoucí na otázku, se odkrylo textové pole, které bylo dosud schované

[1] Balíček `eCards` ve skutečnosti nepracuje s čísly stran, ale s hypertextovými kotvami.

pod tlačítkem. Do tohoto textového pole uložíme řetězec informující o správnosti zvolené odpovědi.

Vyhodnocujeme matematické výrazy

Jednou z předností balíčku `exerquiz` je možnost vkládat do textu otázky na něž je odpovědí matematický výraz a automaticky vyhodnocovat správnost odpovědi. Autor dokumentu zadává správnou odpověď a interval, kde nejsou problémy s vyhodnocováním tohoto výrazu (funkce jedné či dvou proměnných, vektorová funkce, rovnice, atd.). Odpověď vložená uživatelem se numericky porovná v několika náhodných bodech z tohoto intervalu se správnou odpovědí a odsud se posoudí správnost či nesprávnost odpovědi. Vhodnou volbou porovnávací funkce je možno testovat nejen jestli jsou odpovědi stejné (nejčastější případ), ale i jestli se liší o aditivní konstantu (je-li úkolem vypočítat integrál) nebo multiplikativní konstantu (partikulární řešení homogenní lineární diferenciální rovnice) a podobně. Ukážeme si, jak probíhá transformace funkce, kterou uživatel zadá do textového pole, na numerickou hodnotu.

Vytvoříme textové políčko s implicitní hodnotou $2\sin^2(|x|)+1.3$, které automaticky nastavuje velikost fontu podle délky řetězce (volba `\textSize{0}`), při opuštění (např. tabelátorem nebo kliknutím na jiné pole – viz volba `\AAOnBlur`) se ověří, že uživatel zadal matematicky platný výraz (funkce `stripWhiteSpace`, `ParseInput` a proměnná `ok2Continue`, které jsou součástí balíčku `exerquiz`) a zobrazí se v okně (viz `app.alert`), jak byl výraz rozpoznán a převeden na výraz použitelný v JavaScriptech:

```
$f(x)={} $ \textField[\V{2sin^2(|x|)+1.3}\textSize{0}
\AA{\AAOnBlur{\JS{
      fc=stripWhiteSpace(event.value);
      ok2Continue=true;
      vystup=ParseInput(fc);
      if (ok2Continue)
        app.alert("Zadal jste funkci y="+fc+",
            kterou chapu jako "+vystup,2);
    }}}]{funkce}{6cm}{16bp}
```

Nyní vytvoříme textové pole pro vepsání hodnoty x, tlačítko pro spuštění výpočtu a textové pole pro vyplnění výsledku. Do tohoto textového pole je možno zapisovat text pouze pomocí JavaScriptů (volba `\FfReadOnly`):

```
$x=$\textField{vstupx}{3cm}{12bp}
\pushButton[\CA{dosadit do f(x)}\A{\JS{pocitej();}}]
{pocitej}{}{12bp}
$y=$\textField[\Ff{\FfReadOnly}]{fx}{3cm}{12bp}
```

Vhodnou volbou porovnávací funkce je možno například testovat, zda uživatel

správně našel řešení lineární diferenciální rovnice prvního řádu. V tomto případě se mi osvědčilo testovat že

- zadá-li uživatel odpověď ve tvaru $f(x, C)$, zadaná funkce obsahuje konstantu C a je v této konstantě lineární (testuji pomoci dělícího poměru při konkrétní volbě proměnné x)
- $f(x, C) - f(x, 0) \neq 0$ a
- výrazy $\frac{f(x,C)-f(x,0)}{A}$ a $\frac{f(x,C)-B}{A}$ jsou pro konkrétní hodnoty proměnné C konstantní vzhledem k x,

kde A je partikulární nenulové řešení asociované homogenní rovnice a B je jedno z partikulárních řešení nehomogenní rovnice.

Na konec stránky dáme makro `\stranka` popsané v předchozím textu:
`\stranka\newpage`

Používáme radiobutony

Balíček `dps` (das Puzzle Spiel) definuje příkazy pro tvorbu her založených na párování otázek a odpovědí. Při každém spárování otázky a odpovědi se objeví část skrytého nápisu, při nesprávném pokusu je započítán trestný bod. Toto probíhá zcela automaticky.

Výběr otázky a odpovědi je realizován pomocí radiobutonů. Jednoduché použití radiobutonů ukazuje následující příklad: první dvě políčka nemají asociovánu žádnou akci a slouží pouze k uchování příslušné hodnoty pro pozdější použití. K poslednímu políčku je připojena akce, která v tomto případě vyvolá chybové hlášení a vrátí všechny radiobutony do výchozího stavu:

```
\textbf{Zatrhněte vaše pohlaví: }
Muž \radioButton{pohlavi}{10bp}{10bp}{kluk}\quad
Žena \radioButton{pohlavi}{10bp}{10bp}{holka}\quad
Jiné \radioButton[\A{\JS{
    app.alert("Jste muz nebo zena. Hadejte znovu a lepe!");
    this.resetForm(["pohlavi"]);
  }}]{pohlavi}{10bp}{10bp}{anijedno}
\hfill
```

Na konci řádku je umístěno tlačítko, nastavující zaškrtnutí pomocí JavaScriptu. Akce asociovaná s tlačítkem je v sekci `\AAMouseEnter` což znamená, že na tlačítko není nutno klikat, stačí na něj najet kurzorem myši:

```
\pushButton[\TU{Po najeti na tlacitko se oznacil muz.}
\CA{Jsem muz}
\AA{\AAMouseEnter{\JS{
      this.getField("pohlavi").value="kluk";
   }}}]{mujbutton}{1.5cm}{12pt}
```

Informaci o tom, které políčko je zatrženo, zjišťujeme ve funkci `CtiSouradnice` definované v úvodu a použité v následující podkapitole.

Přepínání radiobutonů pomocí JavaScriptů je použito například v balíčku `jeopardy` při tvorbě hry pro dva hráče – po odpovědi jednoho hráče a upravení jeho celkového skóre je automaticky přepnuto na druhého hráče a kladné či záporné body z další otázky budou automaticky započítány druhému hráči (pokud se ten otázky nevzdá a ručně nepřepneme zpět na hráče prvního)

Hledáme myš

V balíčku `fancytooltips` jsou přes celou stranu dokumentu vytvořena průhledná tlačítka, na která lze umisťovat obrázky (z PDF souborů). V textu na stránce jsou klíčová slova překryta průhlednými tlačítky. Při stisku tohoto tlačítka je na stránku umístěn obrázek[2] – většinou doprovodný text k danému klíčovému slovu. Defaultní poloha tohoto doprovodného textu je vpravo nahoře, při volbě `movetips` se však obrázek objeví u kurzoru myši (tj. v bezprostřední blízkosti označeného klíčového slova). K tomu je potřeba znát polohu myši na stránce.

V úvodu jsme definovali funkci `CtiSouradnice`, která pracuje s polohou myši na stránce. Pro hrátky s touto funkcí si nachystejme několikařádkové (volba `\FfMultiline`) textové pole do kterého můžeme zapisovat pouze my a ne čtenář dokumentu (volba `\FfReadOnly`).

```
\textField[\Ff{\FfReadOnly}\Ff{\FfMultiline}]
{souradnice}{6cm}{2cm}
```

Dále definujme tlačítko, které po kliknutí volá periodicky jednou za vteřinu funkci `CtiSouradnice`. Při kliknutí na toto tlačítko je informace o tom, zda jsme měli stisknutý Shift, uložena do proměnné `shiftAN` a vypisována do políčka spolu s informací, který radiobuton z předchozí podkapitoly je zatržen (viz definice funkce `CtiSouradnice` v hlavičce dokumentu). Funkce `CtiSouradnice` také demonstruje, jak lze měnit například barvu textu.

```
150   \pushButton[
      \TU{Po kliknuti se zacne periodicky sledovat poloha mysi}
      \CA{Start}\A{\JS{
          shiftAN=event.shift;
          souradnice = app.setInterval('CtiSouradnice()', 1000);
155     }}]
      {zacni}{2cm}{12pt}
```

Je vhodné použít také tlačítko, které periodické volání funkce `CtiSouradnice` ukončí:

[2] Umísťováním obrázků na tlačítka se budeme zabývat v následující podkapitole

```
\pushButton[\TU{Po kliknuti se prestane sledovat poloha mysi}
\CA{Stop}\A{\JS{
    app.clearInterval(souradnice);
  }}]{skonci}{2cm}{12pt}
```

Trik s `event.shift` umožňuje měnit chování tlačítka. Například umožňuje jedním tlačítkem spouštět animaci dopředu i dozadu. V balíčku `exerquiz` je možno jedním tlačítkem zobrazit správnou odpověď na otázku (klik), nebo vyvolat odskok na vzorový postup řešení (shift+klik).

Na konec stránky dáme opět makro `\stranka`:

```
\stranka\newpage
```

Animujeme obrázky

Při tvorbě animace musíme do souboru nejprve umístit tlačítka na kterých jsou nastaveny snímky animace jako ikony. Většinou jsou tato tlačítka nastavena jako neviditelná pro TEX i pro Acrobat Reader. My pro ikony použijeme tlačítka neviditelná pro Acrobat, ale viditelná pro TEX (tj. mající nenulové rozměry při umisťování do horizontálního seznamu při sazbě). Abychom si problém více osahali, uveďme nejprve tlačítka, která těmto skrytým ikonám odeberou nebo nastaví příznak `hidden` a tím se zobrazí nebo skryjí v textu. Protože tlačítka která později vytvoříme budou mít názvy ve tvaru `ikona.`číslo, můžeme se odvolávat pouze na jedno konkrétní tlačítko pomocí např. `this.getField("ikona.1")`, nebo na všechna tlačítka najednou pomocí `this.getField("ikona")`:

```
\def\tlacitko#1#2#3{
\pushButton[\CA{#1}\A{\JS{
    this.getField("#2").hidden=#3;
  }}]{zobrazit}{2cm}{12pt}}%
\tlacitko{Zobrazit 1}{ikona.1}{false}
\tlacitko{Zobrazit}  {ikona}  {false}
\tlacitko{Schovat 4} {ikona.4}{true}
\tlacitko{Schovat}   {ikona}  {true}
```

Skrývání tlačítek je použito například ve stylu `jeopardy` ke skrytí tlačítka odkazujícího na otázku, na kterou již bylo odpovězeno. Podobně je možné neprůhledným tlačítkem nebo textovým polem překrýt část textu na stránce a po splnění určitých podmínek toto tlačítko odstranit. Toto je použito například ve stylu `jeopardy` při tvorbě hry, spočívající v postupném odkrývání skrytého obrázku.

Nyní použijeme stránky 1 až 5 ze souboru `cisla.pdf` a vytvoříme neviditelná (volba `/F6`) tlačítka se snímky animace. Pokud bychom chtěli, aby tlačítka byla neviditelná i pro TEX, vložíme je do boxu s nulovými rozměry.

```
\def\VlozIkonu#1{
\mbox{\immediate\pdfximage page #1{cisla.pdf}%
\pdfstartlink user{%
        /Subtype /Widget /F 6 /T (ikona.#1) /FT /Btn /Ff 65536
        /H /N  /BS << /W 1 /S /S >>
        /MK << /TP 1
        /I \the\pdflastximage\space 0 R
        /IF << /SW /A >>
        >>
      }%
      \phantom{\pdfrefximage \pdflastximage}%
      \pdfendlink\hss}}

\VlozIkonu1 \VlozIkonu2 \VlozIkonu3 \VlozIkonu4 \VlozIkonu5
```

Význam jednotlivých parametrů uvedených v definici tlačítka lze najít v [6]. Nyní vložíme tlačítko, ve kterém budeme zobrazovat vlastní animaci. Bude větší než ikony vložené v předchozím textu, abychom demonstrovali i umisťování a zvětšování snímku animace na tlačítku. Abychom viděli celé rozměry, vložíme tlačítko do rámečku:

```
\fbox{\pdfstartlink user{%
    /Subtype /Widget /F 4 /T (animace) /FT /Btn /Ff 65536
    /H /N /BS
    << /W 1 /S /S >> /MK << /TP 1
    /IF <</A[0.5 0.5]/SW /B>> >> }%
  \vbox to 3cm {\vss\hbox to 3cm{\hss}}\pdfendlink}
```

Následující tlačítko obsahuje akci, která při umístění kurzoru myši nad tlačítko (příkaz `\AA{\AAMouseEnter{...}}`) spustí animaci opakovaným voláním funkce `animuj` a při opuštění tlačítka animaci zastaví. Funkce `app.setInterval` a `app.clearInterval`, které k tomu použijeme, jsme již jednou použili při zjišťování polohy myši na stránce:

```
\pushButton[\CA{Rozbehni animaci} \RC{Animace bezi}
\BC{1 0 0}
\AA{ \AAMouseEnter{\JS{
      bezicianimace=app.setInterval('animuj()', 400);
    }} \AAMouseExit{\JS{
      app.clearInterval(bezicianimace);
    }}}]{posun}{}{12pt}
```

Tlačítko, které provede jenom jeden krok v animaci, vytvoříme snadno:

```
\pushButton[\CA{Krok} \A{\JS{
```

```
      animuj();
    }}] {posun}{}{12pt}
```

Následující tlačítka přímo nastaví příslušný snímek v animaci:

```
\def\nastavit#1{\pushButton[\CA{Nastav #1}
  \A{\JS{
      f=this.getField("ikona."+#1);
      g=this.getField('animace');
      g.buttonSetIcon(f.buttonGetIcon());
      cislo=#1;
    }}]{nastavit}{}{12pt}}

\nastavit1 \nastavit2 \nastavit3 \nastavit4 \nastavit5
```

A na závěr uvedeme sérii tlačítek, která mění polohu a velikost animace na tlačítku. Všechny akce jsou v sekci `\AAMouseEnter` a stačí tedy umístit kurzor myši nad příslušné tlačítko:

```
\def\zarovnani#1#2#3{
\pushButton[\BC{1 0 0}\CA{#1}
\AA{\AAMouseEnter{\JS{
      f=this.getField("animace");
      f.buttonAlignX=#2;
      f.buttonAlignY=#3;
    }}}]{posun}{}{12pt}}
\zarovnani{na stred}     {50}  {50}
\zarovnani{do rohu}      {100} {100}
\zarovnani{do rohu dolu} {100} {0}
\hfill
\pushButton[\BC{1 0 0}\CA{roztahni}
\AA{\AAMouseEnter{\JS{
      f=this.getField("animace");
      f.buttonScaleWhen=scaleWhen.always;
    }}}]{posun}{}{12pt}
\pushButton[\BC{1 0 0}\CA{stahni}
\AA{\AAMouseEnter{\JS{
      f=this.getField("animace");
      f.buttonScaleWhen=scaleWhen.never;
    }}}]{posun}{}{12pt}
```

Náměty k dalšímu čtení

Vkládání tlačítek, textových polí, radiobutonů a základní manipulace s nimi jsou i s příklady popsány v dokumentaci balíčku `eForms` [5].

Specifikaci formátu PDF je věnována kniha [6]. Zde jsou vysvětleny významy tagů z řádků 179–184 a 193–196. Jiná možnost jak vytvořit objekt s požadovanými vlastnostmi je naklikat si požadované vlastnosti v klikacím programu (např. Adobe Acrobat Professional), uložit PDF soubor, otevřít jej obyčejným textovým editorem, podle názvu tlačítka nalézt příslušnou část PDF dokumentu a poznačit si tagy, které vytvořil klikací program.

Možnostem JavaScriptů a popisu rozšíření tohoto jazyka v PDF je věnována kniha [1], začátečníkům poslouží ještě příručka [2].

Problematika vkládání akcentovaných znaků je částečně řešena v manuálu k balíčku `dps.sty` [4].

Technika tvorby animací u balíčků `animfig` a `pdfanim` vychází z práce pana Holečka a pana Sojky [3]. Prakticky stejná technika byla použita pro umísťování obrázků balíčkem `fancytooltips`.

Summary: Inserting JavaScripts with pdfLaTeX in practice

The article describes few posibilities how to use JavaScript language available in Adobe Reader browser to enhance possibilities and effects in PDF files created by pdfLaTeX. Among others, we describe shortly technical background of some related LaTeX packages available on CTAN.

Reference

[1] Acrobat JavaScript Scripting Reference, `http://partners.adobe.com/public/developer/en/acrobat/sdk/pdf/javascript/AcroJS.pdf`

[2] Acrobat JavaScript Scripting Guide, `http://partners.adobe.com/public/developer/en/acrobat/sdk/pdf/javascript/AcroJSGuide.pdf`

[3] Holeček, J., Sojka, P.: A New Method for Directly Embedding Animation into PDF, Lecture Notes in Computer Science, Vol. 3130/2004, 179–191

[4] Story, D. P.: Das Puzzle Spiel, `http://www.math.uakron.edu/~dpstory/dps.html`

[5] Story, D. P.: eForms Package Manual, `http://www.acrotex.net/data/aeb/manuals/eformman_p.pdf`

[6] PDF Reference, `http://www.adobe.com/devnet/pdf/pdf_reference.html`

Robert Mařík
marik@mendelu.cz