Západočeská univerzita v Plzni

Fakulta pedagogická

Bakalářská práce

ZOBRAZENÍ ČÍSEL V POČÍTAČI

Karel Veverka

Plzeň 2012

Prohlašuji, že jsem diplomovou práci vypracoval samostatně s použitím uvedené literatury a zdrojů informací.

Plzeň, 28 června 2012

……

vlastnoruční podpis

OBSAH

1 ÚVOD

Tématem bakalářské práce je vytvoření distančního kurzu Zobrazení čísel v počítači. Součástí práce je CD s offline verzí kurzu, zdrojovými soubory programu ProAuthor, ve kterém byl kurz vytvořen, a také ostatní soubory, které byly využity ke tvorbě kurzu nebo textové části práce.

Při vytváření kurzu a při psaní textové části bakalářské práce jsem čerpal z literatury uvedené v kapitole 9.

Samotný text práce je rozdělen do několika kapitol. Nejdříve se stručně podíváme na číselné soustavy (2), jaké jsou a k čemu se vůbec používají a částečně si projdeme vyhotovení v kurzu, včetně ukázek výstupů v levých rámcích distančního kurzu. Následovat bude kapitola (3) popisující programovací jazyk používaný v HTML stránkách, ve kterém jsou napsány obslužné skripty v kurzu a s ním související kaskádové styly CSS. V kapitole číslo (4) budou popsány programy, pomocí kterých byl kurz vytvářen, jejich přínos při tvorbě a bude nastíněna i práce s nimi. V následující kapitole (5) budou zmíněna úskalí, která bylo nutno při tvorbě kurzu překonat. A konečně v poslední kapitole (6) se zamyslíme nad možnostmi využití kurzu a možnostmi jeho rozšíření o další výukové materiály v nových studijních článcích.

2 ČÍSELNÉ SOUSTAVY – NÁPLŇ DISTANČNÍHO KURZU

Výukový kurz je zaměřen na studenty, kteří nemají žádné znalosti o číselných soustavách, ale věřím, že na své si přijdou i ti, kteří jsou problematiky znalí. Cílem kurzu je seznámit studenty se základními používanými číselnými soustavami, naučit je mezi těmito soustavami vzájemně převádět, osvětlit problematiku zobrazení celých a desetinných čísel v binární soustavě, a v poslední řadě naučit provádět aritmetické operace v binární soustavě.

Kurz sestává z pěti hlavních kapitol či, chcete-li, tematických celků. Jsou to:

1. Prohlížeč Internet Explorer
2. Číselné soustavy a převody mezi nimi
3. Zobrazení celých čísel
4. Zobrazení čísel v pohyblivé řádové čárce
5. Aritmetické operace v binární soustavě

V následujících částech textu si nastíníme obsah jednotlivých kapitol a na vybrané části se podíváme podrobněji.

2.1 KAPITOLA „PROHLÍŽEČ INTERNET EXPLORER“

Kapitola popisuje, jak změnit nastavení, které může znepříjemňovat využívání offline verze kurzu.

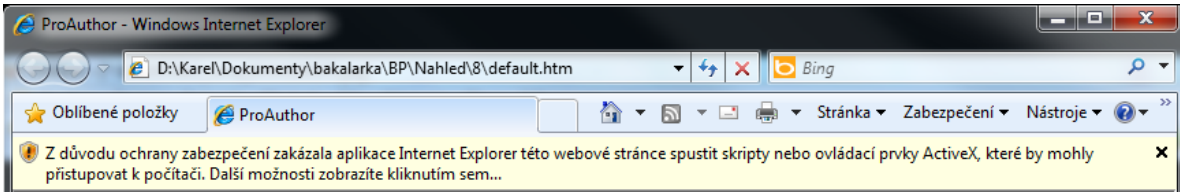
2.1.1 NASTAVENÍ ZABEZPEČENÍ INTERNET EXPLORERU

Úvodní kapitola kurzu je věnována nastavení webového prohlížeče. Ačkoliv se to nezdá, je poměrně důležitá, jelikož bez správného nastavení by kurz nefungoval tak, jak je zamýšleno.

Pokud spouštíme distanční kurz z lokálního úložiště (pevný disk, CD), může se stát, že IE[1] zobrazí varování o spouštění prvku aktivních prvků, jak ukazuje obrázek. Varovný panel se zobrazí při spouštění stránek obsahujících skripty (javascript, ActiveX prvků),

[1] Aplikace IE 8 a nižší zobrazují varovný panel nahoře, IE 9 má přepracovaný design a varovný panel zobrazuje ve spodní části okna

a varuje uživatele před spuštěním potenciálně nebezpečného programového kódu, který by mohl přistupovat k souborům v počítači a ohrozit tak jeho tak úroveň zabezpečení.



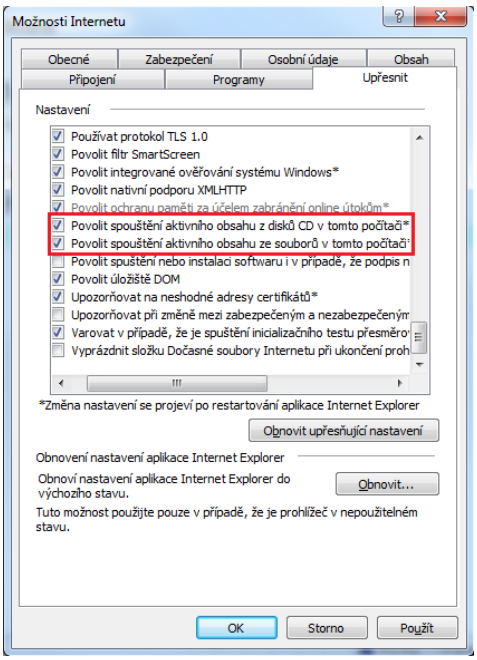
Obrázek 1 - Zabránění spuštění aktivního obsahu

Toto nastavení může znepříjemnit práci s kurzem, a proto si na následujících řádcích popíšeme, jak jej dočasně vypnout.

DŮLEŽITÉ!! Berte na vědomí, že následující změny by měly být pouze dočasné a před jejich provedením byste měli pečlivě zvážit, zda souborům, kvůli kterým zabezpečení vypínáte, můžete důvěřovat.

Nastavení v IE 8 změníte následujícím způsobem:

1. Klikněte na nabídku **Nástroje** a zvolte **Možnosti Internetu**
2. Přejděte na kartu **Upřesnit** a najděte oblast **Zabezpečení**, ve které podle potřeby zaškrtněte:
   - Povolit spouštění aktivního obsahu z disků CD v tomto počítači
   - Povolit spouštění aktivního obsahu ze souborů v tomto počítači



Obrázek 2- Nastavení spouštění aktivního obsahu aplikace IE

Nastavení uložte a restartujte prohlížeč IE. Po skončení práce s kurzem nezapomeňte tyto možnosti vrátit na jejich výchozí hodnoty.

2.2 „KAPITOLA „ČÍSELNÉ SOUSTAVY A PŘEVODY MEZI NIMI“

V této kapitole se student dozví základní informace o číselných soustavách, jsou zde uvedeny příklady jejich použití a naučí se mezi soustavami převádět, což je věc, bez které se v dalších kapitolách neobejde.

Číselná soustava je způsob reprezentace čísel. Rozdělujeme je na dva druhy, podle způsobu určení čísla z dané reprezentace:

1. Nepoziční číselné soustavy – např. zápis římských číslic
2. Poziční soustavy – jmenujme soustavy dvojkovou, osmičkovou, desítkovou a šestnáctkovou, ale známe třeba soustavu šedesátkovou (používá se pro měření času), ale pracovat můžeme se soustavou trojkovou, šestkovou aj.

Každé číslo v poziční soustavě má část celočíselnou a část desetinnou, které jsou od sebe odděleny desetinnou čárkou (v anglosaských zemích se používá desetinná tečka).

Libovolné celé číslo lze vyjádřit $F_Z \geq 0$ v soustavě o základu Z můžeme vyjádřit mnohočlenem:

$$F_Z = a_m Z^m + a_{m-1} Z^{m-1} + .. + a_0 Z^0 = \sum_{i=0}^{m} a_i Z^i$$

My se budeme podrobněji zabývat soustavami se základem Z = 2, Z = 8, Z = 10 a Z = 16.

V rámci studia se budeme setkávat převážně se soustavami desítkovou, dvojkovou, osmičkovou a šestnáctkovou. Podívejme se na ně podrobněji.

1. DESÍTKOVÁ SOUSTAVA (DEKADICKÁ)

Tato v běžném životě nejpoužívanější číselná soustava je jeho každodenní součástí. Na rukou máme obvykle deset prstů, na prstech se v dětství učíme počítat, a tak je tato soustava pro naše životy zcela přirozená. Pro zajímavost uvedu, že pravděpodobný původ této soustavy má své kořeny právě v počtu prstů na rukách u zdravého člověka.

2. DVOJKOVÁ SOUSTAVA (BINÁRNÍ)

Tato soustava používá pouze číslice 0 a 1, což má mnoho výhod. Můžeme si určit, že hodnota 1 bude představovat vyšší hodnotu napětí a 0 nižší - toto označení pojmenujme jako pozitivní logiku. Se dvěma stavy definovanými pomocí těchto dvou hodnot se můžeme setkat i při programování, kdy od sebe rozlišují hodnoty 1 = *true* (pravda, popř. platí) a 0 = *false* (nepravda, popř. neplatí). Dále tato soustava nalézá uplatnění v elektronických spínacích obvodech

3. OSMIČKOVÁ SOUSTAVA (OKTALOVÁ)

Využívá osmi číslic od 0 po 7. Dříve se tato soustava používala u starších typů sálových počítačů s logickými obvody, procesory a mikroprocesory, ale dnes je její využití spíše omezené. Nicméně uveďme alespoň jeden příklad, kdy se s ní můžeme setkat. Unixu znalí uživatelé zajisté narazili na příkaz *chmod* sloužící pro změnu přístupových práv souboru.

4. ŠESTNÁCTKOVÁ SOUSTAVA (HEXADECIMÁLNÍ)

Tato soustava využívá 16 číslic (a znaků) - jsou to číslice 0-9 a znaky A, B, C, D, E, F reprezentující číselné hodnoty 10, 11, 12, 13, 14, 15. Šestnáctková soustava se používá k popisu dat na adresové a datové sběrnici. Hexadecimální vyjádření dat se v počítačové technice značí symbolem h (nebo H) za šestnáctkovým číslem, takže adresa na šestnáctibitové sběrnici může mít např. hodnotu 3F80H. Dále se s šestnáctkovou soustavou můžeme setkat u zápisu RGB hodnoty barvy, což se používá například v HTML kódu, kde zápis začíná křížkem a následuje 6 hexadecimálních číslic hodnoty barvy (#FFFF00 značí žlutou barvu aj.).

2.2.1 PŘEVODY Z DESÍTKOVÉ SOUSTAVY DO JINÝCH

Pro převod čísla z desítkové soustavy můžeme využít dvě metody:

1. Metoda postupným odečítáním
2. Metoda postupným dělením

1. METODA POSTUPNÝM ODEČÍTÁNÍM

Vstupní číslo $F_{(10)}$ (zapsané v desítkové soustavě) se rozkládá odečítáním zmenšujících se mocnin základu Z (základ soustavy, do níž převádíme).

Postupujeme tak, že si nejdříve zjistíme nejbližší mocninu základu Z, která bude rovna převáděnému číslu nebo bude nižší. Poté budeme schopni vytvořit tabulku mocnin, se kterou budeme pracovat.

Od vstupního čísla sestupně (podle hodnoty) odečítáme jednotlivé mocniny z tabulky mocnin, dokud nedojdeme k mocnině Z^0.

Samotný výpočet si nejprve ukážeme na dvojkové soustavě. Od čísla na vstupu zkusíme odečíst mocninu Z^x - pokud by výsledek rozdílu byl větší nebo rovný nule, rozdíl provedeme a na pravou stranu si zapíšeme 1. Pokud by výsledek rozdílu vyšel menší než 0, pak jej neprovádíme a na pravou stranu si zapíšeme 0. Pokračujeme další mocninou z naší tabulky mocnin Z^{x-1} stále stejným způsobem, dokud nedojdeme k mocnině Z^0.

Toto platí pro dvojkovou soustavu, kde je situace nejjednodušší. Rozdíl buď provést lze (zapisujeme 1) nebo nelze (zapisujeme 0) a pokračujeme dalším krokem.

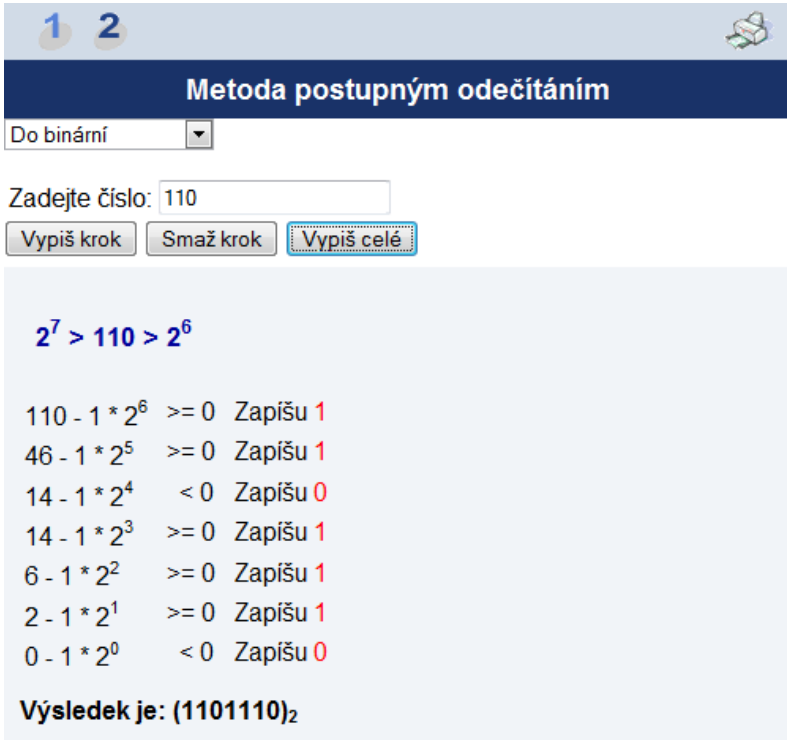
U soustav o základu $Z > 2$ nám může nastat situace, při které musíme rozdíl se stejnou mocninou provést opakovaně, protože se do čísla na vstupu "vejde" několikrát.

V takovém případě si v situaci, kdy rozdíl vyjde větší nebo rovný nule na pravou stranu vždy připočteme jedničku a jako výsledek zapíšeme počet rozdílů, kolikrát jsme je provedli.

Výpočet je téměř totožný, ale pamatujme na to, že v jednom kroku můžeme aktuální mocninu Z^x odečítat vícekrát.

Počítání rozdílů opakujeme, dokud nedojdeme v tabulce mocnin k Z^0.

Nakonec vypíšeme jednotlivé cifry, které jsme si zapsali na pravou stranu, shora dolů.



Obrázek 3 - Metoda postupným odečítáním

2. METODA POSTUPNÝM DĚLENÍM

Vstupní číslo $F_{(10)}$ zapsané v desítkové soustavě dělíme číslem odpovídajícím základu soustavy Z, do níž budeme převádět. Jako mezivýsledky se zapisují zbytky po dělení.

Obrázek 4 - Metoda postupným dělením

Dostaneme-li se do stavu, z nějž už nebude možné dále dělit (na vstupu je číslo menší nebo rovné Z-1), provedeme poslední dělení a jednotlivé zbytky vypíšeme zespoda nahoru.

2.2.2 PŘEVODY MEZI OBECNÝMI SOUSTAVAMI

Pokud potřebujeme převést číslo z osmičkové nebo šestnáctkové soustavy do soustavy dvojkové (popřípadě naopak), můžeme postupovat dvěma způsoby:

1. jednou z možností výpočtu je využití mezi-převodu do desítkové soustavy, který jste se naučili v minulých kapitolách

2. můžeme využít skutečnosti, že čísla *8* a *16* jsou mocninami čísla *2 (8 = 2^3, 16 = 2^4).* Z této skutečnosti vyplývá, že jedna číslice v osmičkové soustavě je tvořena právě třemi číslicemi v soustavě dvojkové (analogicky k tomu pak jedna číslice v šestnáctkové soustavě je tvořena právě čtyřmi číslicemi v soustavě dvojkové). Tato metoda je výrazně rychlejší a jednodušší.

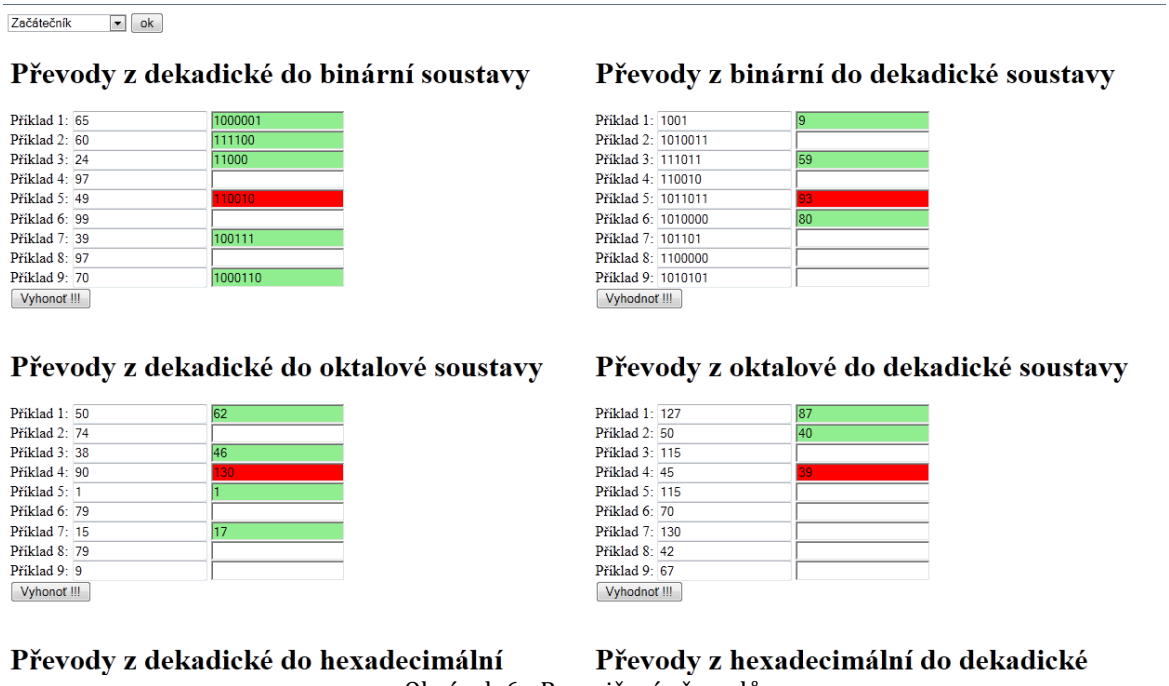
Obrázek 5 - Převod mezi obecnými soustavami

2.2.3 CVIČENÍ – PŘEVODY MEZI SOUSTAVAMI

Na závěr kapitoly je pro studenty připraveno cvičení. V něm by si měli ověřit nabyté znalosti na praktických příkladech převodů. Domnívám se, že je důležité naučit se převádět co nejlépe, a k tomu je zapotřebí spočítat mnoho příkladů na převody. Studenti KVD se budou s převody v různých předmětech setkávat po celou dobu studia. Proto jsem v této kapitole přistoupil k vytvoření generátoru příkladů, jenž má několik stupňů obtížnosti zadávaných příkladů (liší se maximální hranicí čísla pro převod).

Student si zvolí stupeň obtížnosti, vygeneruje si příklady a do vstupního pole vedle zadaného příkladu vepisuje výsledky. Aby si mohl ověřit, zda příklady vypočítal správně, je pod každým celkem tlačítko pro vyhodnocení výsledků v daném celku. Celkem generátor obsahuje 6 celků, převádíme v něm čísla ze soustav dvojkové, osmičkové a šestnáctkové do soustavy desítkové a naopak.

Pokud student zjistí, že mu převody dělají stále problém, bude vhodné opětovně prostudovat předchozí studijní články. Jako pomůcky je možné použít tužku a papír, kam si bude student zapisovat postup výpočtu a do políček zapíše jen výsledky. Zdatnější studenti pak mohou počítat z hlavy a zapisovat výsledky rovnou. Druhý postup je náročnější, ale student si při něm lépe zapamatuje mocniny základů používaných číselných soustav.



Obrázek 6 - Procvičení převodů

2.3 KAPITOLA „ZOBRAZENÍ CELÝCH ČÍSEL“

V předchozí kapitole jsme si řekli, jak se převádí čísla mezi soustavami. Tím jsme si i částečně zodpověděli otázku, jak se zobrazují celá čísla v počítači. Ale v počítačích se pracuje i se zápornými čísly. V této kapitole si ukážeme, jak se dají záporná čísla zobrazovat. V zásadě používáme tři metody:

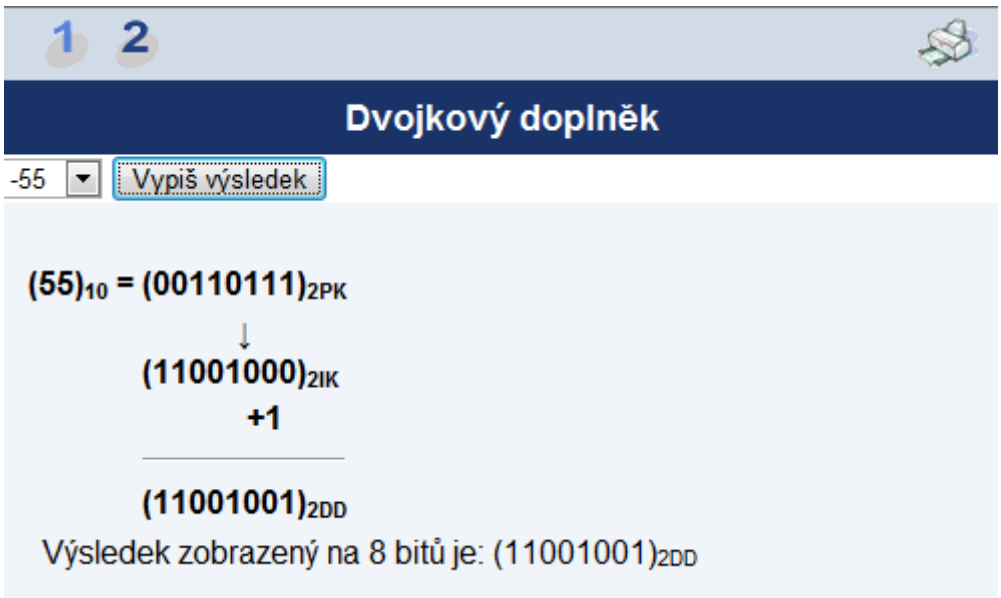
1. Přímý kód
2. Inverzní kód (jednotkový doplněk)
3. Dvojkový doplněk

2.3.1 DVOJKOVÝ DOPLNĚK

Dvojkový doplněk se používá nejčastěji pro vyjádření záporných čísel. Kladná čísla v doplňkovém kódu se nijak neliší od kladných čísel zapsaných v přímém kódu.

Záporná se čísla se vypočítají dvěma způsoby

1. Jako doplněk čísla F do 2n, kde n = počet bitů
2. Druhé vyjádření spočívá v tom, že záporné číslo zapíšeme jako kladné v přímém kódu, provedeme inverzi všech bitů a k výsledku přičteme 1



Obrázek 7 - Dvojkový doplněk

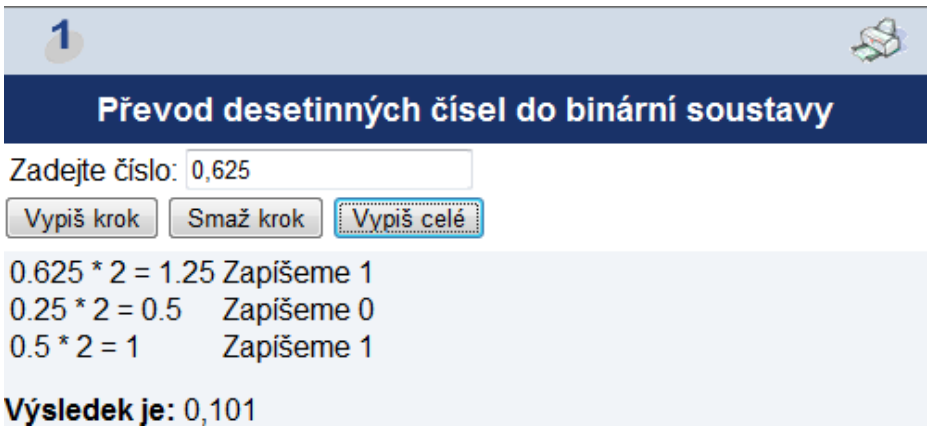
2.4 KAPITOLA „ZOBRAZENÍ ČÍSEL S ŘÁDOVOU ČÁRKOU“

Jak jsme si řekli dříve, každé číslo v poziční soustavě má i desetinnou část. V této kapitole seznámím studenty s tím, jak se tato část převádí do dvojkové soustavy ze soustavy desítkové, a také obráceně. V příkladech budeme brát v úvahu pouze část vpravo za desetinnou čárkou, celočíselnou část jsme si probrali při převodech mezi soustavami. Pokud bychom museli převést číslo, které má část celočíselnou i desetinnou, pak věřte, že tyto dvě části se počítají zvlášť a výsledek se pouze složí dohromady a oddělí desetinnou čárkou.

2.4.1 PŘEVOD Z DESÍTKOVÉ DO DVOJKOVÉ SOUSTAVY

Pro převod je vhodné postupovat postupným násobením desetinného čísla F základem dvojkové soustavy - tzn. číslem 2.

V každém kroku zapisujeme celočíselnou část (tu před desetinnou čárkou) a v případě, že výsledek násobku je větší než 1, pak od něj 1 odečteme.



Obrázek 8 - Převod desetinných čísel

2.5 KAPITOLA „ARITMETICKÉ OPERACE V BINÁRNÍ SOUSTAVĚ“

V této kapitole se naučíme základní početní operace v binární soustavě, ukážeme si sčítání a odčítání.

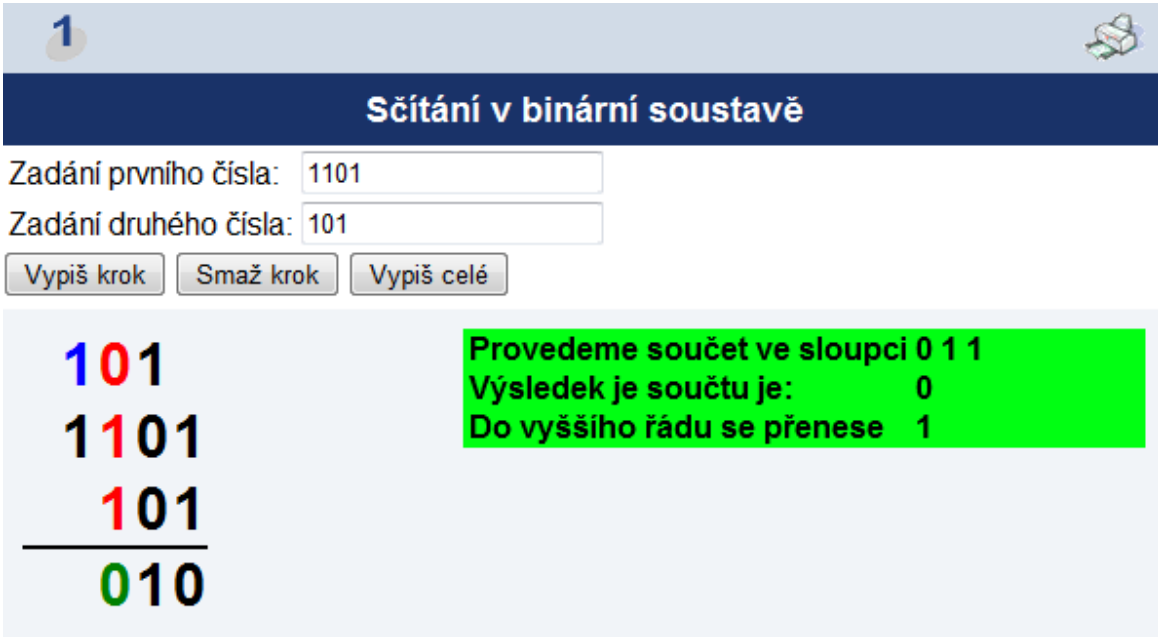
2.5.1 SČÍTÁNÍ

Sčítání ve dvojkové soustavě je velmi důležitou matematickou operací, slouží jako základ pro další číselné operace (např. pro odečítání).

Ve dvojkové soustavě se při sčítání můžeme setkat s následujícími situacemi:

- 0 + 0 = 0
- 0 + 1 = 1; 1 + 0 = 1
- 1 + 1 = 10

V posledním případě nám došlo k tzv. přenosu do vyššího řádu. Při přenosu do vyššího řádu do pole výsledku zapíšeme druhou číslici a první číslici připočteme v následujícím kroku součtu.



Obrázek 9 - Sčítání

Uvedl jsem několik ukázek z kapitol kurzu. V každé kapitole jsem se při vytváření stránek snažil o co nejvýstižnější zobrazení výpisu, pokud to bylo možné, tak aby odpovídal postupu, který bychom prováděli při počítání na papír. Myslím si, že takováto možnost vizualizace velmi ulehčí pochopit postup výpočtu, jenž nemusí být při zobrazení celého vypočteného příkladu na první pohled patrný. V každé části však mohu studentům doporučit jediné – čím více příkladů spočítají, tím lépe, jelikož jedině propočtením mnoha příkladů si celý systém výpočtů osvojí.

3 JAVASCRIPT A KASKÁDOVÉ STYLY

Levé rámce kurzu jsou většinou tvořeny ručně psanými HTML stránkami obsahujícími javascriptový kód a často i kaskádové styly. Na tyto dvě technologie se v následujících řádcích podíváme trochu podrobněji. Na některé problémy spojené s kaskádovými styly a na ukázky kódů se podíváme v následujících kapitolách.

3.1 JAVASCRIPT

Javascript je skriptovací jazyk určený ke tvorbě interaktivních webových stránek.

- Programy v něm napsané jsou nezávislé na platformě, na které běží
- Je interpretovaný jazyk, to znamená, že je překládán řádek po řádku až při běhu programu
- Na základě předchozího bodu je jasné, že programy jsou při běhu pomalejší a prohlížeč funguje jako interpreter
- Javascript je tzv. Case Sensitive jazyk, což znamená, že si při psaní musíme kontrolovat velikost písmen příkazů a názvů proměnných (proměnná vstup není stejná jako proměnná Vstup)

3.1.1 VÝVOJ JAVASCRIPTU

Programovací jazyk Javascript byl vytvořen v roce 1995 společností Nestcape pod původním názvem Livescript a poprvé byl implementován v prohlížeči Nestcape Navigator 2 ve verzi 1.0. Díky nastupující oblibě programovacího jazyku Java byl však ještě před vydáním prohlížeče přejmenován na Javascript.

Původním záměrem bylo, aby pomocí skriptů bylo možno vyhodnotit uživatelské vstupy (například validace formulářových polí) a data nemusela putovat od klienta na server a zase zpět jen pro informaci, zda je konkrétní pole správně vyplněné. S tehdejším pomalým vytáčeným internetovým připojením to byla opravdu velmi vítaná funkce.

Javascript byl dále vyvíjen a Netscape Navigator 3 přinesl podporu verze 1.1 tohoto programovacího jazyka. Rychlý nástup Javascriptu udělal z Nestapu leadera na trhu webových prohlížečů, což donutilo konkurenční Microsoft, aby přinesl podporu

do vlastního prohlížeče IE. To se také stalo a krátce po Nestcape Navigatoru 3 vyšel i prohlížeč IE 3, přinášející vlastní implementaci Javascriptu pod názvem JScript (název byl záměrně jiný, aby se předešlo patentovým sporům se společností Netscape).

Javascript neměl žádné standardy popisující syntaxi nebo vlastnosti a nová implementace od Microsoftu jen potvrdila tento problém. Proto bylo rozhodnuto, že se Javascript musí standardizovat.

V roce 1997 byl odeslán návrh jazyka ve verzi 1.1 Evropské asociaci výrobců počítačů[2] ke standardizaci syntaxe a sémantiky multiplatformního skriptovacího jazyka, na které se podíleli programátoři z firem Sun, Borland, Microsoft, Nestcape a dalších, aby po několika měsících práce vytvořili standard ECMA-262 definující nový skriptovací jazyk nazvaný ECMAScript. V následujícím roce přijala Mezinárodní standardizační organizace jako standard (ISO/IEC-16262). Od té doby prohlížeče s různým stupněm úspěchu používají ECMAScript jako základ pro implementaci Javascriptu.

Od té doby se Javascript stal důležitou součástí každého významného webového prohlížeče a již neslouží k pouhému vyhodnocení uživatelského vstupu, ale interaguje téměř s každou součástí okna prohlížeče a jeho obsahu. Stal se natolik významnou součástí moderního webu, že jej musí podporovat i mobilní webové prohlížeče.

3.1.2 UMÍSTĚNÍ SKRIPTU DO HTML STRÁNKY

Javascriptový kód nebude nikdy spuštěn, pokud nebude buď přímo součástí stránky, nebo na něj nebude uvnitř stránky alespoň odkazováno. K tomu, aby prohlížeč poznal, kde začíná a končí kód, který může interpretovat, slouží párový HTML tag *<script></script>*. U skriptu umístěného uvnitř stránky se nachází samotný kód přímo uvnitř tagů *<script></script>* a u skriptů umístěných v externím souboru je na něj odkazováno parametrem src uvnitř tagu *<script>.*

[2] ECMA - European Computer Manufacturers Association

1. Skript umístěný uvnitř stránky

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
 <head>
  <title>Titulek stránky</title>
 </head>
 <body>Text stránky.
 <script language="JavaScript" type="text/javascript" >
  // javascriptový kód
 </script>
 </body>
</html>
```

2. Skript umístěný v externím souboru, cesta k souboru je definována atributem *src* – může být umístěný v hlavičce i těle dokumentu

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
 <head>
  <title>Titulek stránky</title>
 </head>
 <body>
  Text stránky. <br>
 <script language="JavaScript" type="text/javascript" src="skript.js">
 </script>
</body>
</html>
```

Předchozí odstavce Vám nastínily, že Javascript je dominantním programovacím jazykem ve webových prohlížečích, což byl také důvod, proč jsem jej zvolil pro psaní obslužných skriptů v kurzu. Dále jsem uvedl, jak poznat, kde je zdrojový kód umístěn (ať již je součástí stránky, nebo je v externím souboru), abyste se na něj mohli podívat sami.

Další popis skriptovacího jazyka by byl již nad rámec této práce, můžete však čerpat z literatury uvedené v kapitole 9, ze které jsem při tvorbě kurzu čerpal i já.

3.2 KASKÁDOVÉ STYLY (CSS)

Kaskádové styly jsou další významnou součástí moderního webu. Jejich hlavní výhodou je, že od sebe oddělují obsah stránky (data) a vzhled (formátování). V této kapitole nastíním historii, způsob vkládání do HTML stránky a některé důležité vlastnosti pro jejich použití, o kterých se ještě zmíním v kapitole 5.

3.2.1 HISTORIE A VZNIK KASKÁDOVÝCH STYLŮ

V počátcích internetu měly HTML dokumenty poměrně jednoduchou strukturu. Jak se internet rozšiřoval, vznikalo stále více webů a jejich autoři se chtěli od těch druhých

lišit. Tím se struktura stávala složitější a do jazyka HTML přibývaly nové prvky (tagy) umožňující například úpravu textu (<FONT>,<BIG> aj.).

Postupem času se zdrojový kód stránek stále více komplikoval, protože obsahovaly mnoho prvků určujících vzhled stránky, ať již ve formě samotných tagů nebo jejich atributů.

Vznikajícího nepořádku si všimli lidé z W3C (World Wide Web Consortium) a v roce 1995 začali zveřejňovat průběh prací na projektu označovaném CSS. V roce 1996 byla vydána specifikace CSS 1, v roce 1998 CSS 2 a v současné době se pracuje na verzi CSS 3.

3.2.2 UMÍSTĚNÍ STYLŮ DO HTML STRÁNKY

Stejně jako u Javascriptu máme několik možností, jak vložit námi definované styly do HTML stránky.

1. Styl bude umístěný uvnitř stránky, bude uzavřený uvnitř párového tagu *<style></style>*

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
 <head>
  <title>Titulek stránky</title>
 <style type="text/css">
  h2    {color: blue; font-style: italic}/* nastavení barvy a kurzívy nadpisu */
 </style>
 </head>
 <body>
  <h2>Nadpis druhé úrovně. </h2> <br>
 </body>
</html>
```

2. Styl bude umístěný v externím souboru – může být odkazováno na několik stylů z různých umístění

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
 <head>
  <title>Titulek stránky</title>
 <link rel="stylesheet" type="text/css" href="prvni_styl.css">
 <link rel="stylesheet" type="text/css" href="druhy_styl.css">
 </head>
 <body>
  <h2>Nadpis druhé úrovně. </h2> <br>
 </body>
</html>
```

Obsah souboru první_styl.css

```
h2      {color: blue; font-style: italic}
```

3. Styl bude umístěný přímo uvnitř elementu – tento způsob zápisu není příliš výhodný a moc se nepoužívá

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
 <head>
  <title>Titulek stránky</title>
</head>
 <body>
  <h2 style="color: blue">Nadpis druhé úrovně. </h2> <br>
 </body>
</html>
```

3.2.3 NĚKTERÉ VLASTNOSTI KASKÁDOVÝCH STYLŮ

Kaskádové styly umožňují nastavit vzhled a vlastnosti prvků, které by za použití HTML tagů a jejich atributů ani nastavit nešly. Další zajímavou vlastností je, že pomocí stylů můžeme nastavit vzhled mnoha prvků najednou. Máme-li na stránce nadpisy první úrovně (<h1>), nemusíme v HTML kódu nastavovat barvu pro každý jednotlivý nadpis zvlášť, ale stačí nám na to jeden řádek v CSS kódu:

```
h2 {color: blue} /* nastaví všem nadpisům druhé úrovně modrou barvu */
```

Ale další zajímavou vlastností je, že lze seskupovat i prvky, kterým chceme přiřazovat stejný vzhled:

```
h1, h2 {color: blue} /* nastaví nadpisům první a druhé úrovně modrou barvu */
```

To, že máme vlastnosti určené pro nadpisy takhle pěkně pohromadě, můžeme využít, pokud budeme chtít změnit barvu těchto nadpisů. Při použití stylů nám stačí změnit jednu vlastnost, bez použití stylů bychom museli měnit barvu každého jednotlivého prvku zvlášť (a to můžeme mít projekt s desítkami webových stránek a stovkami nadpisů).

1. **KASKÁDA**

Další velmi důležitou věcí u stylů je kaskáda, což je opatření, jímž se cíleně zabrání konfliktům pravidel, pokud jsou definována na různých místech nebo různými způsoby. Obecně platí, že přednost mají pravidla, která jsou přesnější a jsou definována konkrétněji.

V následujícím příkladu vidíme, že odstavec vyhovuje všem třem pravidlům definovaným pomocí stylu, ale nejkonkrétnějším ukazatelem je v tomto případě identifikátor *id*, proto bude barva textu odstavce zelená.

```
Definujme různý styl pro odstavec, třídu a identifikátor:
<p class="trida1" id="pozn1">Text odstavce</p>

p {color: blue}
p.trida1 {color: yellow}
pozn1 {color: green}
```

Konkrétněji lze definovat styl už jen přímo u samotného tagu atributem *<style>*, v následujícím příkladu by byla i přes všechna ostatní výše uvedená pravidla barva textu oranžová:

```
<p class="trida1" id="pozn1" style="color:orange">Text odstavce</p>
```

Jsou-li definice jedné vlastnosti stejně konkrétní, pak záleží na pořadí, ve kterém jsou pravidla uvedena. Vždy se použije to pravidlo, které je uvedeno jako poslední.

```
p {color: black}
p {color: blue}
p {color: green; color: red}
```

Odstavec podle výše uvedeného příkladu se bude řídit poslední definicí barvy, a tudíž bude červený.

2. PRAVIDLO !IMPORTANT

Umožňuje upřednostnit deklaraci stylu s tímto pravidlem nad deklaracemi bez tohoto pravidla a nezáleží pak na konkrétním určení prvku, ani na pořadí definice dalších stylů – toto pravidlo dává stylu nejvyšší prioritu, což znamená, že text v následujícím příkladu bude zelený.

```
p {color: black}
p {color: blue !important}
p {color: green; color: red
```

3. DĚDIČNOST

Další vlastností, kterou kaskádové styly mají, je dědičnost. Některé vlastnosti mohou být zděděny z rodičů na potomky. U každé vlastnosti je určeno, zda je dědičná nebo není. Pokud je a my ji nedefinujeme zvlášť, bude tato zděděna z nadřazeného prvku. Pokud vlastnost dědičná není, použije se výchozí hodnota, kterou obvykle určuje nastavení internetového prohlížeče.

Uvažme následující příklad:

```
body {color: green}
```

Barva textu je dědičná, a jelikož je definována pro celé tělo dokumentu, bude všechen text na stránce zobrazen zelenou barvou, pokud nebude výslovně definováno jinak. Hodnotu *color* zdědí všichni potomci tagu <body>, což jsou v podstatě všechny prvky vepsané mezi tagy <body></body>.

4. VELIKOST SOUBORŮ

Poslední důležitou vlastností, kterou kaskádové styly přináší, je celkové zmenšení množství přenášených dat, čímž se zvýší i rychlost načítání stránek. Pomocí kaskádových stylů bývají zápisy často kratší, a hlavně jeden zápis je definován i pro několik desítek prvků najednou, což by se bez použití stylů muselo definovat pro každý prvek zvlášť. Stejný soubor lze využít na více stránkách najednou, což ve výsledku ušetří mnoho práce. Tento jediný soubor navíc stačí stáhnout pouze jednou, protože internetové prohlížeče si ho uloží do dočasné paměti, kde bude přístupný pro všechny stránky, které se na něj odkazují.

4 PROGRAMY POUŽÍVANÉ PRO TVORBU KURZU

Pro tvorbu kurzu a věcí s ním spojených nebylo zapotřebí příliš velkého množství programů. V zásadě byly za potřebí tyto:

1. Poznámkový blok či jiný textový editor
2. Autorský program ProAuthor
3. Firebug – rozšíření pro internetový prohlížeč Mozilla Firefox

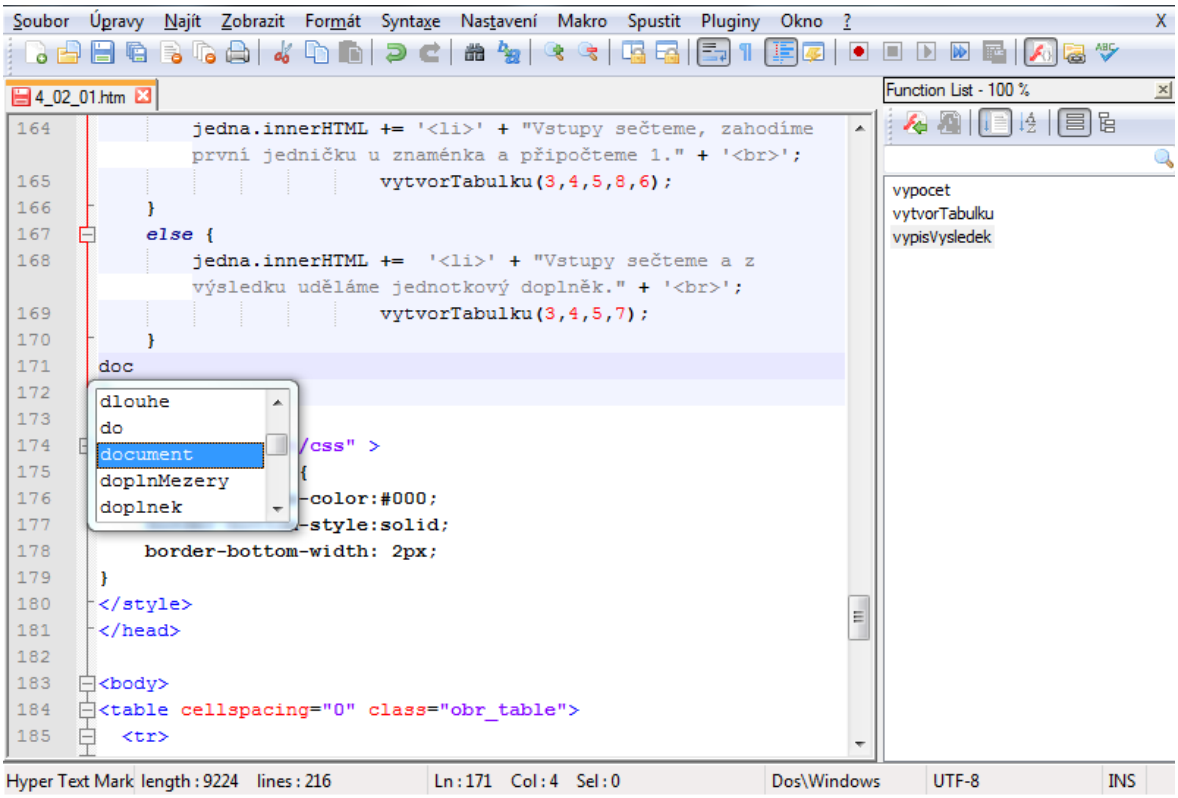
O každém programu se zmíním, popíšu, jak se s ním pracovalo a v čem mi byl při vytváření kurzu užitečný.

4.1 POZNÁMKOVÝ BLOK

Pro vytváření webových stránek, souborů kaskádových stylů a javascriptových souborů nám stačí základní editor textu. Já jsem však použil textový editor Notepad++, jenž se hodí k editaci zdrojových kódů lépe. Mezi hlavní výhody patří zvýraznění syntaxe zdrojového kódu, přičemž je možné volit z mnoha programovacích jazyků a sám program zvolí podle přípony souboru vhodnou syntaxi. Další funkcí, kterou jsem často využíval, bylo automatické dokončování slov, které mi zvláště u psaní delších názvů (jež nejsou v javascriptu nikterak výjimečné) ušetřilo zlomek času a zabránilo chybám kvůli nechtěným překlepům.

Obě zmíněné funkce jsou součástí základní instalace programu. Notepad++ navíc podporuje rozšíření pomocí pluginů, které se mohou hodit. Já jsem si zvykl používat jeden nazvaný Function List, jenž přidá k pravému okraji obrazovky sloupec s vytvořenými funkcemi v javascriptu a pomocí dvojkliku umožňuje rychle přejít na začátek funkce v libovolně dlouhém zdrojovém kódu.

Pro možnosti tvorby zdrojových kódu je možné využít i jiné alternativy. Nabízí se editor Poznámkový blok od společnosti Microsoft, který je součástí instalace Windows. Z českých programů je k dispozici program PSPad, který je ke stažení a použití zdarma a má velmi podobné funkce jako mnou používaný editor Notepad++. K práci s weby existují i profesionální editory, od společnosti Adobe je to program Dreamweaver, který je ale dle mého názoru pro využití při tvorbě bakalářské práce příliš složitý a není důvod jej používat, když i jednodušší editory svůj účel splní.



Obrázek 10 - Notepad++

4.2 PROAUTHOR

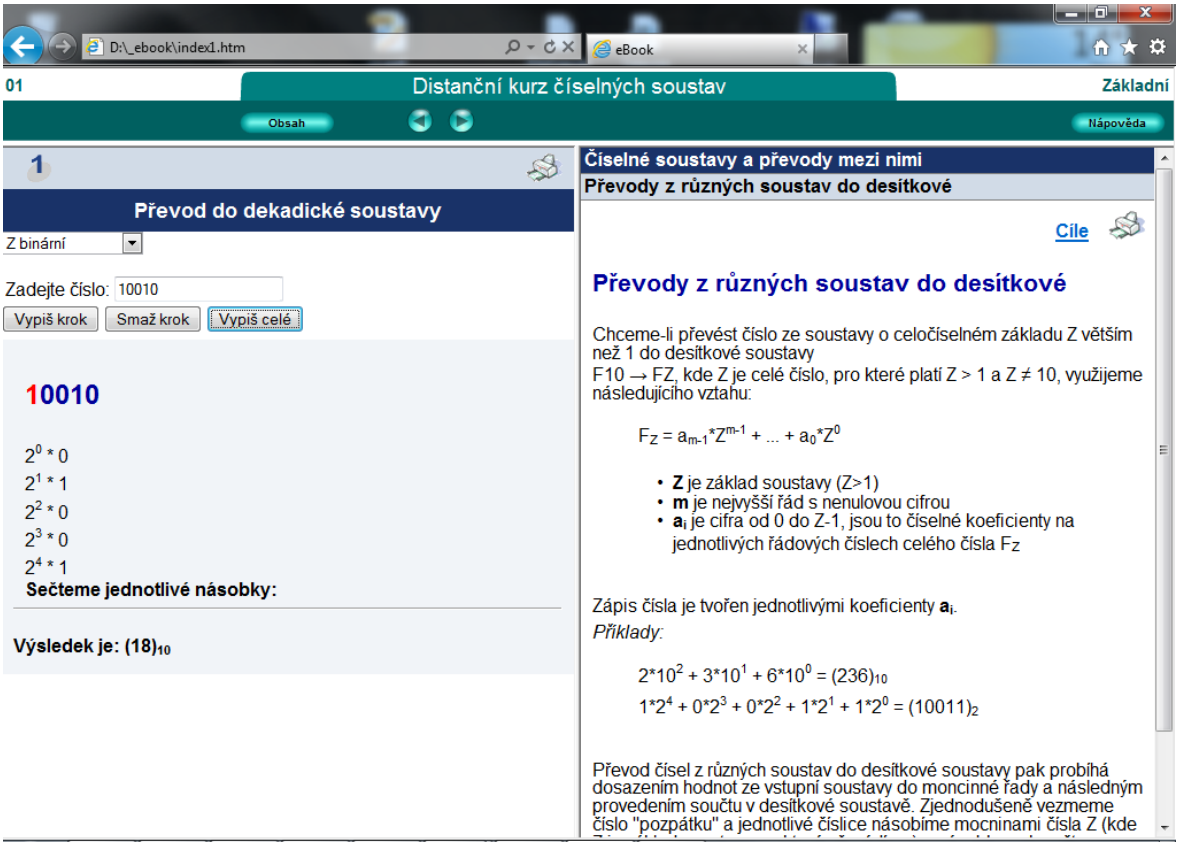
S programem ProAuthor jsem se setkal již při studiu, kdy jsem se s ním při vytváření krátkého distančního kurzu setkal poprvé. V programu se příjemně pracovalo, a proto mi přišlo logické, že bych jej měl použít i pro tvorbu bakalářské práce. Pro mě jako studenta na KVD vyplývala další výhoda z toho, že jsem měl program k dispozici na většině počítačů na katedře a pro studenty je možné opatřit i studentskou licenci k programu a používat jej doma k tvorbě kurzu z pohodlí domova.

ProAuthor je velmi užitečný nástroj pro tvorbu e-learningových kurzů. Program má intuitivní ovládání a od uživatele nejsou požadovány žádné speciální znalosti programování nebo html jazyka. Tím pádem se může soustředit pouze na vytváření samotného obsahu a celkové struktury kurzu. Obsah sestává z textové části, jejíž tvorba je podobná jako v kterémkoli jiném textovém editoru, ale je obohacen o možnost vložení multimediálních souborů, např. animací, videí, obrázků, ale i textových nebo html souborů. Kurz je členěn do několika celků – mezi větší celky patří studijní kapitoly, které

mohou obsahovat různé vzdělávací objekty – jmenujme např. studijní články, úkoly, cvičení, testy aj.

Po výsledném zpracování kurzu jej lze exportovat do několika výstupních formátů, ať již to jsou formáty AICC a SCORM, což jsou formáty vhodné pro import do LMS[3], nebo jako knihu Rtf dokumentů, kam se exportují pouze obrázky a text a nakonec je možnost exportu jako E-book, do kterého je exportován i tento kurz. Exportovaný kurz je rozdělen na 3 celky:

1. Horní navigační panel s možností přechodu na Obsah obsahuje navigační šipky
2. Pravý rámec obsahuje vždy text studijního článku
3. Levý rámec pak obsahuje obrázky, animace, a jiný multimediální obsah



Obrázek 11 - E-book

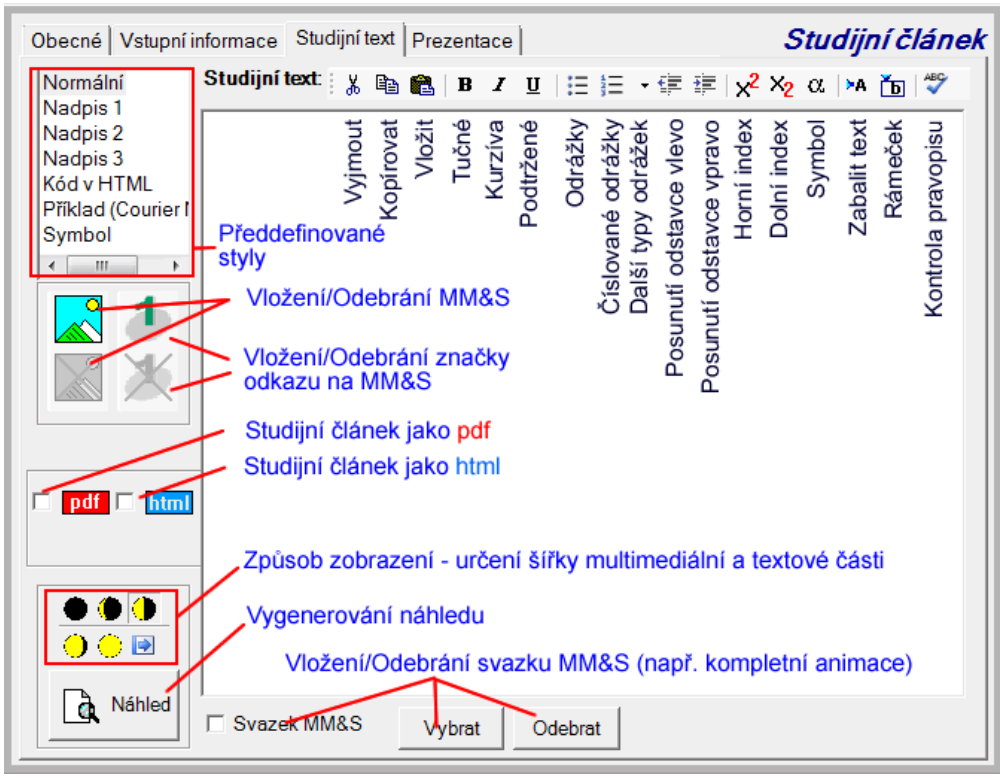
Pro zobrazení exportovaného e-booku je doporučeno používá prohlížeč MS IE, pro který je výstupní formát e-book koncipován. Z dnešního hlediska, kdy IE ztrácí tržní podíl,

[3] LMS (Learning Management Systém) je aplikace řešící administrativu a organizaci výuky v rámci eLearningu

se může toto řešení zdát značně omezující, ale pokud o této nutnosti víme dopředu, můžeme se na ní připravit. Jediný problém by mohl nastat uživatelům různých distribucí OS Linux, kde je ale k dispozici prohlížeč Mozilla Firefox, jenž jsem používal pro testování ve verzi 11.0 a zběžně jsem v něm zkoušel otevřít i exportovaný e-book. Nesetkal jsem se s žádným problémem zobrazení stránek nebo chybného výpočtu, proto si troufnu tvrdit, že použití v tomto prohlížeči je také možné.

4.2.1 TVORBA KURZU V PROGRAMU PROAUTHOR

Tvorba kurzu prakticky sestává ze dvou částí a v podstatě kopíruje rozložení exportovaného e-booku, jak jsme si jej popsali v předchozí části textu. Autor vytváří zvlášť obsah pro pravý rámec a pro levý rámec. ProAuthor umožňuje nastavit velikost levého



Obrázek 12 - Formátování textu v programu ProAuthor

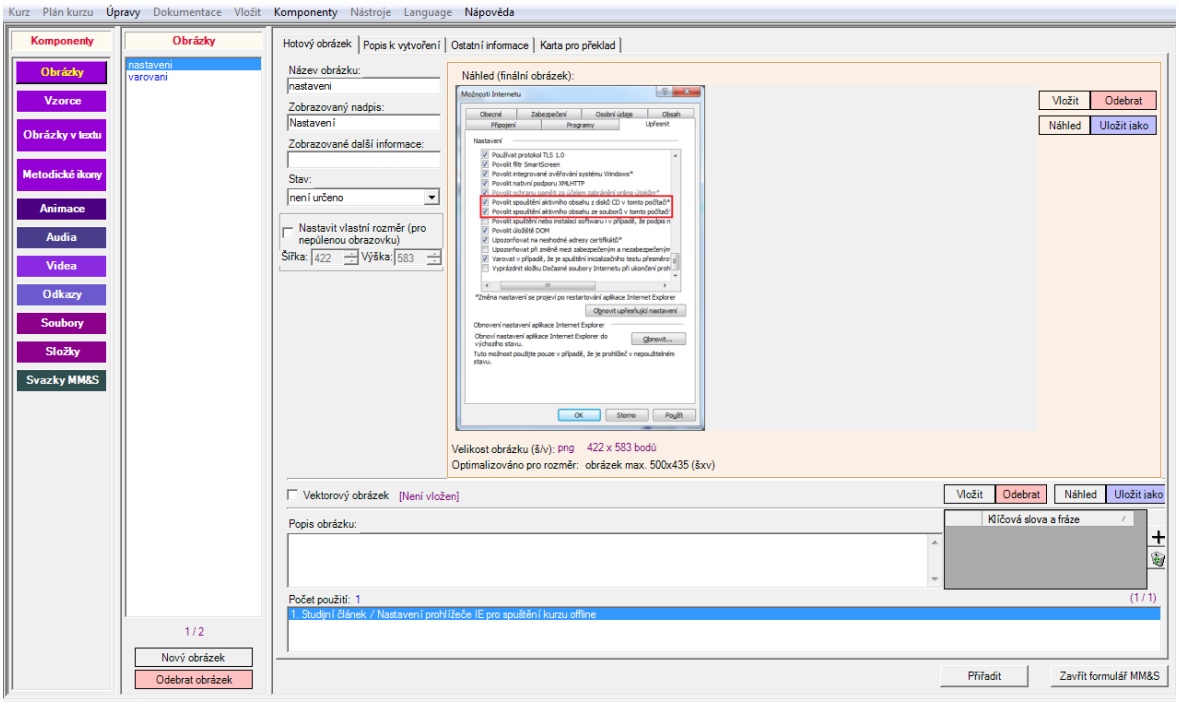
rámce, ale dle mého názoru ne úplně ideálním způsobem. Prakticky dochází k tomu, že se napevno nastaví šířka rámce v pixelech, ale podle obrázků bych spíše nabyl dojmu, že bude použito procentuální rozdělení obrazovky. Jelikož se pro zobrazení používá *Huml* tag <frameset></frameset>, který umožňuje rozdělení okna procentuálním poměrem, zvolil bych spíše tento způsob. Při použití pevného rozměru levé části sice máme jistotu,

že námi zvolená multimediální komponenta bude mít pro zobrazení dostatek místa, ale pravá část s textem může být zkrácena, jelikož při změně velikosti okna dochází ke smršťování/roztáhnutí textu na menší/větší počet řádků, což může esteticky na dnešních velkých širokoúhlých monitorech působit poněkud nevzhledně. Na malých monitorech může být naopak text příliš zhuštěný.

Pravý rámec obsahuje text studijního článku, který je možné zadat několika způsoby. Můžeme mít text předpřipravený ve formátu *pdf* nebo *html* a importovat ho do ProAuthoru, nebo jej můžeme vytvořit přímo v samotném programu. K tomu nám budou nápomocny funkce pro formátování textu, které sice nedosahují takových možností, jaké známé z komerčních i nekomerčních kancelářských balíků, ale pro tvorbu e-kurzu by měly dostačovat.

Při tvorbě pravého rámce bych rád vyzdvihl jednu funkci programu, jejíž důležitost by mohla být podceněna, ale její integrace do programu je velmi výhodná. Je to funkce *Kontrola pravopisu*. Při vkládání textu a četném opakovaném čtení se nám může stát, že lehce přehlédneme nějaký překlep, jelikož texty už známe téměř zpaměti. Proto je kontrola pravopisu velkou výhodou, jelikož nemusíme texty kopírovat do externího textového editoru a kontrolovat chyby v něm. Navíc se mi osobně líbilo zvýraznění chyby více než třeba v programu Microsoft Word, jelikož je výraznější a nedá se přehlédnout.

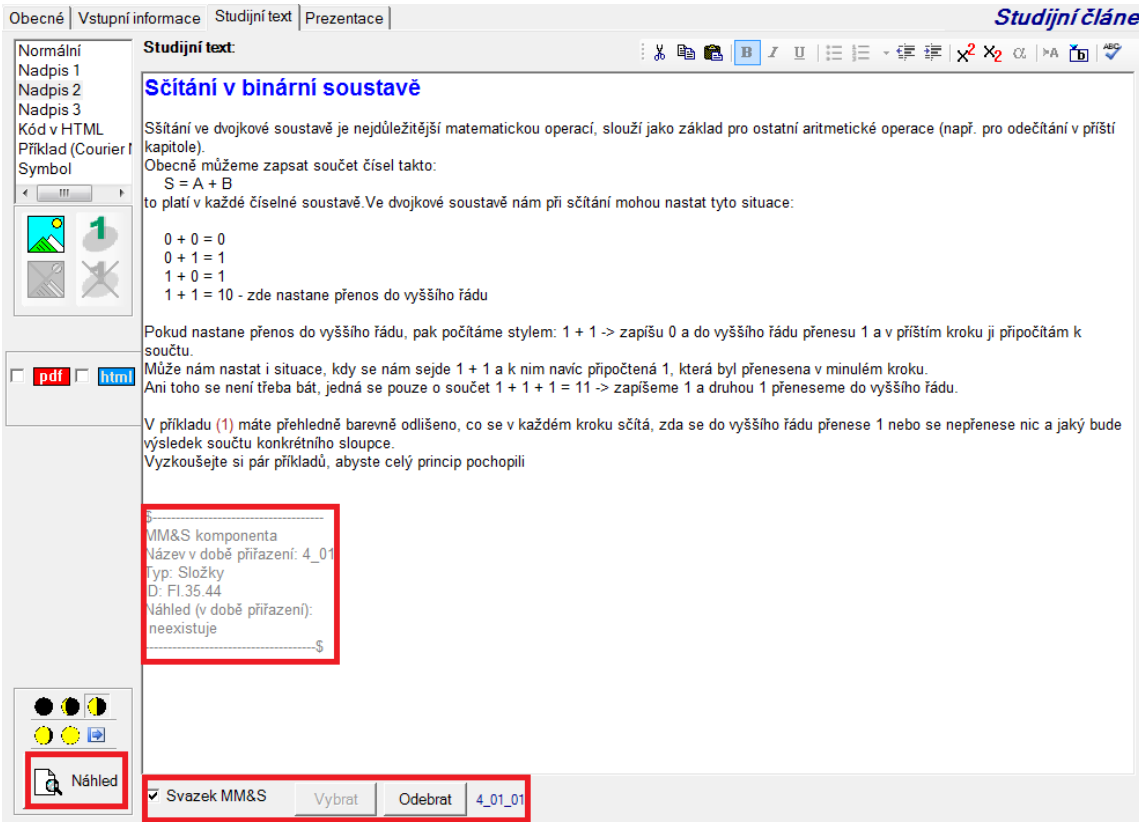
Levý rámec je tvořen komponentami, na něž se vkládají odkazy do studijního článku. Pro vložení komponenty je třeba kliknout na tlačítko „Vložit MMS“. Seznam komponent, které se dají do studijního článku vložit je poměrně rozsáhlý. Po výběru komponenty je nutné kliknout na tlačítko „Přiřadit“ a svázat ji tak s konkrétním studijním článkem. Tím svázáním se ProAuthor řídí při exportování do e-booku, kdy pro každou přiřazenou komponentu vytvoří zvláštní stránku a do horního panelu levého rámce vloží čísla odkazující na jednotlivé komponenty, Ty je možné svázat se studijním článkem pomocí tlačítka „Vložit odkaz na MMS“ a tím poskytnout studentovi zpětnou vazbu ve studijním článku na komponentu, o které je v textu řeč.



Obrázek 13 - Komponenty MMS

Na tomto místě musím ještě zmínit jeden prvek, se kterým jsem v ProAuthoru pracoval. Pokud vkládáme do levého rámce námi vytvořené html soubory, je zapotřebí někde v kurzu ProAuthoru vytvořit složku, kam si je uložíme. Já jsem je vkládal do adresáře *Data\Komponenty\folder*, v něm jsem si vytvořil adresářovou strukturu tak, aby odpovídala jednotlivým kapitolám kurzu.

Do studijního článku vkládáme MMS komponentu, konkrétně komponentu Složky. V ní klikneme na tlačítko *Vložit soubory* a vybereme soubory, které chceme k danému studijnímu článku přiřadit. Tím zajistíme, že ProAuthor při exportu překopíruje soubory z daného adresáře k souborům, které ke studijnímu článku generuje. Stisknete-li však tlačítko *Náhled* u studijního článku, zjistíte, že se v levém rámci MMS komponenta nezobrazuje, ačkoliv je ve studijním článku vložená. Aby se zobrazila, je potřeba vytvořit svazek MMS. Po kliknutí na tlačítko *Vybrat* dole v okně vyskočí nové okno s možností vložení svazku.



Obrázek 14 - Svazek MMS

Při vkládání jsem narazil na jedno omezení, jež bylo, že vkládaný soubor nesměl být umístěn ve složce s kurzem. Je to sice drobnost, ale je třeba na to pamatovat a mít zdrojové soubory připraveny i mimo tuto složku. Jako svazek MMS se dá vložit buď animace ve formátu *sof*, ale přes roletkovou nabídku lze vybrat i soubor *html*. Proto se nezalekněte, pokud ve výchozím výběru neuvidíte v cílovém adresáři žádné soubory.

Když jsme se naučili vkládat do levého rámce *html* soubory, pak musím zmínit nutnost opakování celého postupu při změně zdrojového souboru, kdy se musí znovu provést vložení souborů ze složky a opětovné vytvoření svazku MMS. Tím jsme vyřešili zobrazení levého rámce při náhledu a exportu.

Poslední drobnou věcí, se kterou jsem při tvorbě kurzu zápasil, a na kterou bych rád upozornil studenty, kteří se s nimi mohou také setkat, byly drobné zádrhele při vkládání textu. V kurzu často používám horní a dolní index. Na obrázku výše je znázorněno, že ProAuthor s indexy pracovat umí, ale problémy mu dělal text vložený klávesovou zkratkou *Ctrl+V*, který rozhodil formátování textu, a vznikly artefakty, při kterých po vložení dolního indexu byl zbytek textu také formátován jako dolní index.

Pokud budete při tvorbě kurzu pracovat s indexy, dejte si na toto pozor a text raději formátujte přímo v ProAuthoru ručně.

Řekli jsme si základní principy práce s programem ProAuthor, některá úskalí, se kterými jsem se při práci setkal, a která se netýkají přímo práce v samotném prostředí programu ProAuthor, zmíním v následující kapitole. Na závěr bych rád zmínil, že práce s programem je velmi intuitivní, a pokud máte za úkol vytvořit jednoduchý distanční kurz, tak bude tato operace snadná a rychlá.

4.3 FIREBUG

Firebug je oblíbený a mocný doplněk pro internetový prohlížeč Mozilla Firefox, který se mi při tvorbě webových stránek do levého rámce kurzu stal neocenitelným pomocníkem. Zpočátku jsem pro zobrazování stránek používal prohlížeč IE[4], ale jeho chybová konzole byla příliš stručná a nijak neusnadnila hledání chyby v kódu. Ve výstupu Firebugu se zobrazují chyby javascriptu, po kliknutí na chybu přejde přímo na část zdrojového kódu, kde chyba nastala. Velkou výhodou je možnost editace zdrojového kódu přímo v internetovém prohlížeči, nemusí se přepínat do externího editoru a drobné chyby (překlepy v názvu proměnných nebo tagů) se dají upravit přímo za běhu.

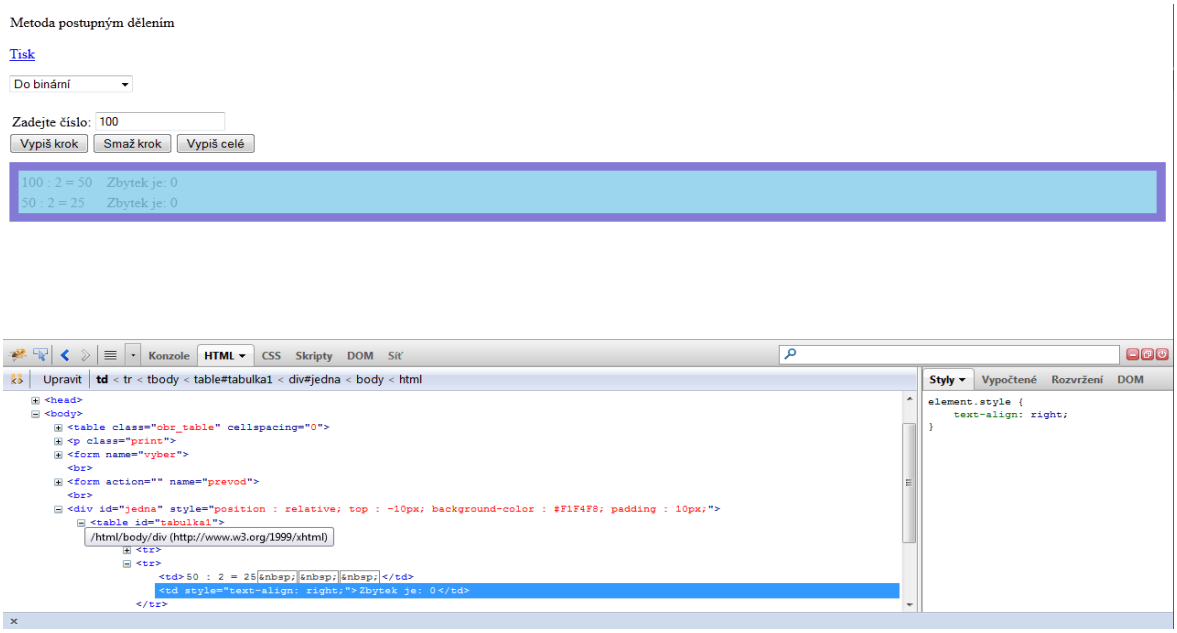
Při testování skriptů jsem často potřeboval vědět obsah proměnných při běhu cyklu, k čemuž je možné využít umístění breakpointů do kódu a následné krokování cyklu, přičemž si člověk může v každém kroku prohlédnout hodnoty, které ho zajímají.

Další důležitou věcí pro mě byla práce s kaskádovými styly. Firebug je umožňuje přímo editovat, ale hlavně zobrazuje aktuálně použité styly na jakémkoliv prvku na stránce, a to včetně informací o zděděných pravidlech.

Poslední příjemnou věcí, na kterou se dá rychle zvyknout je tlačítko *Inspect*, díky němuž můžete kliknout na jakýkoliv text nebo jiný element na stránce, tam bude viditelně orámován, aby byl zvýrazněn proti zbytku stránky a ve spodní části s okna (popřípadě v druhém okně, záleží na tom, jakým způsobem Firebug používáte) s kódem bude vyznačena přímo ta část, která odpovídá vámi vybrané oblasti. Vy si pak můžete příslušnou část kódu prozkoumat a u elementu prohlédnout nabalené styly.

[4] Tehdy se jednalo ještě o verzi 6, dnes ve verzi 9 disponuje IE podobnými nástroji též

Zmínil jsem pouze funkce, které pro mě měly při tvorbě kurzu nějaký přínos, Firebug toho nabízí mnohem více. A rovněž je třeba uvést, že podobné nástroje jsou dnes součástí mnoha prohlížečů – Opera nabízí pro ladění stránek vlastní součást zvanou Dragonfly a IE ve verzi 9má pod klávesovou zkratkou F12 schované Nástroje pro vývojáře a prohlížeč Google Chrome nabízí také Nástroje pro vývojáře.



Obrázek 15 - Pracovní okno s doplňkem Firebug

5 ÚSKALÍ VYTVÁŘENÍ KURZU

Při vytváření kurzu jsem se setkal s řadou problémů, které jsem musel překonat. Některé souvisí s používanými programy, jiné pak osvětlí postup při vývoji částí kódu, jenž za dobu tvorby prošel mnoha změnami.

5.1 VYHODNOCENÍ VSTUPU

Příklady použité v kurzu reagují na vstupní údaje zadané uživatelem. Uživatel zadá vstup, kliknutím na tlačítko spustí skriptový kód a dojde k vypočtení výsledku. Důležité ovšem bylo zajistit, aby byl vstup v požadovaném tvaru podle určitých pravidel. Potřebujeme-li například převést číslo v binárním tvaru do jiné číselné soustavy, očekáváme, že na vstupu bude číslo složené z 1 a 0.

Má první myšlenka vedla k následujícímu postupu – uživatel zadá vstupní údaj, stiskne tlačítko a skript nějakou svou funkcí ověří, zda je vstupní hodnota složena jen z 1 a 0. Pokud tomu tak není, bude na to uživatel upozorněn. Níže přikládám kousek kódu, který se o toto měl postarat:

```
function checkPole(vstup,delka_pole,pozice_ve_formulari) {
        var pole_znaku =
        ['0','1','2','3','4','5','6','7','8','9','A','B','C','D','E','F','a','b',
        'c','d','e','f'];
        var delka = vstup.length;
        for(var i = 0; i <= delka-1; i++) {
        obsahuje = false;
              for (var j = 0; j <= delka_pole; j++) {
                    if (vstup.charAt(i) == pole_znaku[j]) { obsahuje=true; }
              }
              if (obsahuje === false) {
                    alert('Zadejte nové číslo');
                    return false;
              }
        }
}
```

Jak je z kódu patrné, funkce prochází vstup po jednotlivých znacích a porovnává každý znak se znaky v poli. Jsou-li všechny vstupní znaky v pořádku, pokračuje se ve výpočtu, ovšem nastane-li situace, kdy se na vstupu nachází nepovolený znak, vyskočí uživateli v prohlížeči výstražné okno, které ho vyzve k zadání nového vstupu. Jak velká část se bude procházet při porovnávání, to je určené dynamicky vstupním parametrem.

Funkce byla sice funkční, ale nebyla uživatelsky příliš přívětivá. Jelikož jsem chybné vstupy mnohokrát testoval, tak mi výstražná okna začínala vadit, a tak jsem přemýšlel o jiné podobě ošetření zadání vstupních údajů. Nakonec jsem dospěl k názoru, že nejlepší by bylo, aby uživatel nemohl chybné vstupní údaje zadat vůbec. Zmáčkl by špatnou klávesu a její údaj by se do vstupního pole vůbec nezaznamenal. Tento postup však vyžadoval pozměněný kód, který by se o to vyhodnocení postaral.

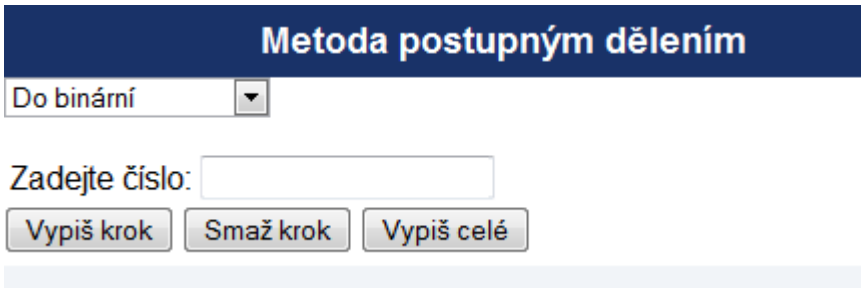
```
function checkPole(e) {
        var poleCisel =
        [48,49,50,51,52,53,54,55,56,57,65,66,67,68,69,70,97,98,99,100,101,102];
        var poleZnaku = [8,9,46,37,39];
        var obsahuje = false;
        var delka = 1;

        if (idE('soustava')[5]) { delka = idE('soustava').value; }
        e = e || window.event;

        var  kod = (e.charCode === undefined) ? e.keyCode : e.charCode;
        if (e.ctrlKey) return false;

        for (var i = 0; i <= delka; i++) {
             if (kod == poleCisel[i]) {
                   obsahuje = true;
             }
        }

        for (var j = 0; j <= 4; j++) {
             if (e.keyCode == poleZnaku[j]) {
                   obsahuje = true;
             }
        }

       if (!obsahuje) { return false; }
}
```

Funkce zachytává stisknutí klávesy ve vstupních polích a vyhodnotí, která klávesa je stisknuta – na rozdíl od dřívějšího kódu se neporovnává zobrazený znak, ale porovnává se jeho číselná hodnota vepsaného znaku podle ASCII tabulky (charcode) nebo přímo číselná hodnota stisknuté klávesy (keycode) – toto rozlišení bylo nutné kvůli tomu, že na událost *onkeypress* reagují internetové prohlížeče různou návratovou hodnotou a stávalo se, že mi prohlížeč zakázal vypsat znak, i když měl být zápis povolený. Nicméně přínosem tohoto postupu bylo, že jsem mohl použít některé klávesy, které zpříjemňují ovládání pomocí klávesnice (klávesy Backspace, Del, Tab a kurzorové šipky).

[5] idE je mnou vytvořená funkce - ekvivalent pro document.getElementById(), aby se zkrátil zápis

5.2 SJEDNOCENÍ OVLÁDÁNÍ DO EXTERNÍHO SOUBORU

Zpočátku k zobrazení výpisu sloužilo jediné tlačítko, po jehož stisku došlo k výpočtu a k zobrazení výsledku. Po vytvoření několika stránek zobrazených v levém rámci e-learningového kurzu jsem zjistil, že se v nich buď opakuje, nebo je velmi podobná část kódu, s proto jsem tento koncept později přehodnotil tak, aby se samotný výpočet dal zobrazovat po jednotlivých krocích a co nejvíce se přiblížil postupu, jako bychom jej dělali za pomoci papíru a tužky. Myslím si, že toto řešení umožní uživateli lépe pochopit principy výpočtu, než kdyby viděl celé řešení příkladu vypracované a měl by si jednotlivé kroky výpočtu odvodit.



Obrázek 16 - Krokování výpočtu

Přijetí tohoto konceptu však znamenalo poměrně zásadní přepracování stávajících stránek a také změnu tvorby těch následujících. Stránka v levém rámci se starala jen o výpočet, jenž následuje bezprostředně po ztrátě focusu (zaměření) vstupního pole a veškeré ovládání stránky, včetně přiřazení obsluhy událostí jednotlivým prvkům na stránce je uloženo v externím souboru. Pro mě to mělo nespornou výhodu v tom, že jsem se již dále mohl soustředit pouze na naprogramování samotných výpočtů a jediné, co mi dělalo trochu problém, bylo někdy složitější uložení jednotlivých kroků výpisu.

V části, která sloužila k ovládání stránky, jsem také narazil na některá úskalí. Na stránkách kvůli použití externího souboru pro ovládání používám u některých prvků stále stejný atribut *id*. V externím souboru se po načtení stránky přiřadí jednotlivým prvkům obsluhy událostí (právě podle *id*), ale ne na každé stránce jsou všechny prvky, pro které se události přiřazují. Přiřazení události neexistujícímu prvku by způsobilo chybu skriptu. I pro tuto situaci existuje v javascriptu řešení. Podívejte se na následující kód.

```
function initF() {
        var vstup1 = idE('vstup1');
        vstup1.onkeypress = checkPole;
        vstup1.onfocus = resetuj;
        vstup1.onblur = vypocet;

        if (idE('vstup2')) {
             var vstup2 = idE('vstup2');
             vstup2.onkeypress = checkPole;
             vstup2.onfocus = resetuj;
             vstup2.onblur = vypocet;
        }

        idE('krokVpred').onclick = krokVpred;
        idE('krokVzad').onclick = krokVzad;
        idE('vypisCele').onclick = vypisCele;
        document.body.onload = resetuj;
        if (idE('soustava')) {
             idE('soustava').onchange = resetuj;
        }
}
```

Z ukázky je patrné, že pomocí ověření atributu *id* lze jednoduše ověřit, zda je prvek na stránce přítomen. Pokud ano, přiřadí se mu události, pokud na stránce není, nedojde k chybě v javascriptovém souboru.

5.3 ÚPRAVA SOUBORŮ V LEVÉM RÁMCI E-KURZU

V ProAuthoru se dají vytvářet levé rámce kurzu několika způsoby. Možnosti vkládání komponent jsme si ukázali výše, já se v této kapitolce zaměřím na některé drobnosti, na které se dá narazit, a se kterými je třeba tím pádem počítat.

Při vložení komponent v podobě obrázků/animací se tyto přiřadí ke studijnímu článku a program si upraví strukturu webové stránky podle počtu připojených komponent. To je ta nejjednodušší možnost, protože se nemusíme o nic starat a víme, že design stránek bude stejný v celém kurzu. Jde především o design záhlaví v levém rámci.

Obrázek 17 - Záhlaví levého rámce

Na obrázku vidíte, jak správný výsledek vypadá. Pokud však do levé části vytváříte vlastní obsah do html souborů, budete potřebovat několik věcí upravit tak, aby byl zachován sjednocený design kurzu. Vámi vytvořený html soubor neobsahuje část záhlaví,

kterou do stránky přidává ProAuthor. Proto je třeba ji vytvořit ručně. Je třeba přidat a upravit následující část kódu:

```
<table cellspacing="0" class="obr_table">
<tr>
<td class="nadpis_table">
        <table cellpadding="0" cellspacing="0" class="cisla">
              <tr>
                    <td class="cisla">
                          <a href="1_01_01.htm"><img border="0"
        src="../but/but1.gif" width="23" height="26" />
                          </a>
                    </td>
                    <td class="cisla">
                          <a href="1_01_02.htm"><img border="0"
        src="../but/but2s.gif" width="23" height="26" />
                          </a>
                    </td>
              </tr>
        </table>
</td>
</tr>
<tr>
      <td class="ObrTitle">Metoda postupným dělením</td>
</tr>
</table>
<p class="print">
      <a href="javascript:window.print();">
            <img src="../img/tisk.gif" border="0" width="32" height="27"
       hspace="0" alt="Tisk" />
      </a>
</p>
```

Následující část kódu je převzata ze souboru, jehož výsledná podoba je na obrázku výše. Při vytváření je třeba si dát pozor na několik věcí:

- V tabulce s atributem *class=cisla* (označena červeně a tučně) musí být tolik buněk <td></td>, kolik komponent, což jsou námi vytvořené jednotlivé html soubory, vkládáme – v tomto případě stránky dvě, jindy více nebo méně
- Modře vyznačená část kódu je velmi důležitá – při kliknutí na obrázek čísla dojde k načtení stránky do levého rámce. Proto je nutné zkontrolovat, aby odkazy pod jednotlivými čísly směřovaly na správné stránky. To se musí upravit v každé stránce ručně

5.4 ÚPRAVA KASKÁDOVÝCH STYLŮ PRO RUČNĚ VYTVÁŘENÉ STRÁNKY

V minulé kapitolce jsme si ukázali, jak je třeba pozměnit kód, aby nám zachoval jednotný design kurzu, ale celé by to nestačilo, pokud by se patřičně neupravily kaskádové styly v souboru. Pro správné zobrazení záhlaví (zachování barevného sjednocení) je třeba

připojit do našich stránek soubor s kaskádovým stylem, který si ProAuthor automaticky generuje. To je snadná záležitost spočívající v přidání jednoho řádku kódu do hlavičky (mezi párové tagy <head></head> každého html souboru, takže hlavička našich souborů by měla mít variaci na následující řádky kódu:

```
<head>
<title>Grafický průběh výpočtu z desítkové soustavy dělením</title>
<link rel="stylesheet" type="text/css" href="../style/style.css" />
</head>
```

Titulek stránky bude samozřejmě pokaždé jiný, ale především je třeba se zaměřit na správné určení souboru se stylem (ta část kódu vyznačená tučně) a odkazovat na něj relativně.

Druhou věcí, na kterou si musíme dávat pozor v souvislosti se styly, je dědičnost. Po exportování kurzu do podoby e-booku se mi rozhodil design tabulek vytvořených na mých stránkách. Písmena vypadala trochu jinak, ale hlavně se řádky překrývaly. Díky Firebugu jsem zjistil, že ProAuthorem generovaný styl přidává výšku řádku tabulky (<tr>) napevno k tagu <body>. Dejte si pozor na to, že se Vám tato možnost může stát a pro jistotu si styly definujte buď přímo v *html* souborech nebo do vlastního externího *css* souboru.

6 MOŽNOSTI VYUŽITÍ KURZU

Kurz, jenž se vám dostal k rukám, byl koncipován jako výukový materiál pro studenty KVD na fakulty pedagogické Západočeské univerzity. Zabývá se částí předmětu UIN[6], konkrétně číselnými soustavami, se kterými se mnoho studentů v tomto předmětu seznámí poprvé. Z toho vyplývá, že cílí na studenty začátečníky, ale věřím, že několik poznatků využijí i ti, kteří se s číselnými soustavami setkali při svém dřívějším studiu. Ti mohou informace v něm uvedené brát jako formu opakování.

Kurz obsahuje pouze část z osnovy předmětu. Studenti s jeho pomocí získají základní informace o číselných soustavách a naučí se s nimi pracovat. Osvojí si dovednosti převodu mezi soustavami, naučí se možnosti zobrazení celých a desetinných čísel a budou schopni počítat základní aritmetické operace ve dvojkové soustavě. Po prostudování kurzu a důkladném procvičení si počítání by měl student úspěšně absolvovat zápočtový test, jenž se dané problematiky týká.

Kurz jsem se snažil dělat interaktivně v těch částech, ve kterých to bylo možné. Postupoval jsem tak z jediného důvodu – při počítání postupujeme po jednotlivých krocích, málokdy máme k dispozici příklad vypočítaný, a pokud ano, pak je pro nás složitější přijít na uvedený postup. V kurzu je tedy v pravém rámci obsažen teoretický postup výpočtu a v levém je k dispozici výstup počítání, který se zpravidla dá krokovat tak, aby student viděl, jak probíhá výpočet krok po kroku v souladu s tím, jako by jej psal na papír on sám.

Kurz obsahuje pouze malou část z celkového množství látky probírané v předmětu a zabývá se těmi nejpodstatnějšími základy, na kterých se bude stavět v následujících hodinách výuky. A proto je ideálním adeptem na rozšíření, umím si představit, že by časem existoval distanční kurz předmětu UIN, ve kterém by byla zpracována kompletní látka předmětu.

Rozšíření také nebude příliš komplikované, jelikož je kurz vytvářen v programu ProAuthor, k němuž budou k dispozici zdrojové kódy, a při pokračování se na ně dá navázat přidáním dalších kapitol s novými studijními články. Zda by byl použit

[6] UIN – Úvod do informatiky

princip výpisu, jenž byl zvolen v této práci, by záleželo na zvážení autora. Skript, jenž se o výpis stará je univerzální, nebyl by proto problém jej vložit do dalších studijních článků a na stránce s výpočtem se zabývat pouze kódem výpočtu, stejně jako je tomu v případě této práce.

7 ZÁVĚR

Cílem této bakalářské práce bylo vytvořit distanční výukový kurz pro Zobrazení čísel v počítači. Kurz měl obsahovat studijní články, ve kterých by se studenti naučili pracovat s čísly tak, jak se s nimi pracuje v počítači, naučili se převodům mezi číselnými soustavami a základním výpočtům.

Myslím, že nejpovedenější částí kurzu je levý rámec v e-booku, kam jsem se snažil naprogramovat výpočty příkladů tak, aby odpovídaly postupu, jenž by studenti použili při ručním psaní postupu při počítání na papír.

Kurz by se měl dát použít především pro studenty studující informatiku, jelikož se týká části probírané látky.

Text bakalářské práce popisuje náplň kurzu, a dále pak obsahuje popis programů, ve kterých je kurz vypracován a zmiňuje problémy, se kterými jsem se při tvorbě kurzu musel vypořádat. Poslední kapitola se pak zabývá možností využití kurzu a zmiňuje teoretickou možnost budoucího rozšíření kurzu a další kapitoly z předmětu UIN.

8 SEZNAM OBRÁZKŮ

9 SEZNAM LITERATURY

1. ANTOŠOVÁ, Marcela a Vratislav DAVÍDEK. Číslicová technika. České Budějovice: Kopp, 2003, 305 s. ISBN 80-7232-206-0.
2. HRONEK, Jiří a Jiří MAZURA. Struktura počítačů. Učební text. Katedra informatiky, UP Olomouc, 2007.
3. ZAKAS, Nicholas C. JavaScript pro webové vývojáře: programujeme profesionálně. Vyd. 1. Překlad Lukáš Krejčí. Brno: Computer Press, 2009, 832 s. ISBN 978-80-251-2509-0.
4. MEYER, Eric A. Eric Meyer o CSS: kompletní průvodce. Vyd. 1. Překlad Jan Pokorný. Brno: Zoner Press, 2007, 560 s. ISBN 978-80-86815-64-0.

10 RESUMÉ

The purpose of the thesis is to create an educational e-learning course. It's supposed to aim at beginner students. In this course students should learn basic information about representing numbers in computing, and acquire skills to convert numbers from one numbering system into another.

The first chapter shows how the thesis is structured into chapters and each chapter contains a short description of its content.

The second chapter describes the content of the e-learning course itself, giving examples of what students should learn and is filled in with attendant pictures.

The third chapter describes javascript scripting language and cascade style sheets, which were massively used in the process of creation of the course.

The fourth chapter is about programs and tools used to make this course, giving a short description of each on of them.

The fifth chapter focuses on problems that had to be faced when creating the course and gives a small amount of hints how to avoid the mistakes that were made and had to be corrected.

The sixth chapter contains final thoughts. Whom is the e-learning material intended and speaks about the possibilities of future developement.

11 PŘÍLOHY

Přílohou je CD s textem bakalářské práce, dále se na něm nachází kurz dělaný v programu ProAuthor a offline verze kurzu exportovaná ve formě e-booku.

Obsah složek na CD:

- Offline verze kurzu se nachází ve složce **E-book**
- Kurz ve formě zdrojových souborů programu ProAuthor se nachází ve složce **ProAuthor**
- Text bakalářské práce ve formátech *.docx* a *.pdf* se nachází ve složce **Bakalářská práce**
- Doprovodné obrázky použité v textu bakalářské práce nebo v kurzu se nacházejí ve složce **Obrázky**
- Soubory levých rámců kurzu ve formátu *.html*, včetně obslužných souborů javascriptu ve formátu *.js* jsou umístěny ve složce **Zdrojové kódy**