ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

Obor Informatika

Katedra informačních technologií

POUŽITÍ KASKÁDOVÝCH STYLŮ V INTERNETOVÝCH TECHNOLOGIÍCH

BAKALÁŘSKÁ PRÁCE

Vypracoval: Milan Adam, 2007

Vedoucí bakalářské práce: Ing. Marek Čandík, Ph.D.

Prohlášení

Prohlašuji tímto, že jsem svou bakalářskou práci na téma „Použití kaskádových stylů v internetových technologiích“ jsem vypracoval samostatně a využil jsem pouze pramenů uvedených v seznamu literatury.

V Praze dne 27. 6. 2007 …….

Milan Adam

Poděkování

Na tomto místě bych rád poděkoval vedoucímu bakalářské práce panu Ing. Markovi Čandíkovi, Ph.D. za odborné vedení a cenné rady při vypracování této bakalářské práce.

„Použití kaskádových stylů v internetových technologiích“

„Using of cascading style sheets in Internet technologies“

Souhrn

Tato práce se zabývá problematikou použití kaskádových stylů v internetových technologiích. Kaskádové styly slouží k formátování dokumentu při udržení strukturovanosti jazyka HTM.

V první části mé práce se snažím pomocí odborné literatury a vlastního vyjádření popsat standardizované použití kaskádových stylů. A to především jejich aplikace na dokumenty a používání jednoznačné identifikace elementů tzv. selektorů.

Ve druhé části bakalářské práce jsem na základě teoretických poznatků rozebral tvorbu internetových stránek v praxi a to několika praktickými ukázkami. V poslední části práce jsem vytvořil webové stránky pro výukové účely na Provozně ekonomické fakultě. Při tvorbě stránek jsem se potýkal s několika problémy, které jsem vyřešil pomocí získaných teoretických poznatků.

V celé práci jsem nastínil především použití kaskádových stylů, které je v dnešní době nutností každé moderní internetové prezentace. Dobrá znalost tohoto nástroje se tvůrcům webových stránek zcela jistě vrátí v podobě poptávky po jejich produktech.

Klíčová slova:

Kaskádové styly, HTML, Internet, Element, Selektor, Webdesign.

Summary

This thesis deals with the matter of using of cascading style sheets in Internet technologies. The cascading style sheets are used for formating of document with keeping of the document structure.

In the introduction it attempts, using expert literature and its own expression, to introduce standardized using of cascading style sheets. Mainly its application on documents and using of unique identification of elements – selectors.

In second part of my bachelor's thesis I have explained practical creating of web pages, based on theoretical knowledge, by few practical examples. In the last part of thesis, I have created web pages for educational purpose on Faculty of economics and management. I have had to solve few problems during creating of web pages, which I have solved by acquired theoretical knowledge.

In whole thesis I managed using of cascading style sheets, which is necessary in today's modern web presentation. Web pages creator's can make a profit on using this tool in form of their products demand.

Key words:

Cascading style sheets, HTML, Internet, Element, Selector, Webdesign

Obsah:

1. Úvod

V dnešní době je zřejmé, že internet se rozrostl do rozměrů, jaké jen málokdo mohl tušit. Počet přístupů na tuto mezi-kontinentální síť dosahuje denně závratných čísel a je tedy nasnadě, že se internet stal nedílnou součástí každodenního života většiny obyvatel vyspělých zemí. Bylo jasné, že ve chvíli, kdy bude internet dostupný pro širokou veřejnost, stane se také bezesporu mocnou zbraní marketingového trhu. V počátcích měly webové prezentace na internetu pouze velké mezinárodní firmy. Dnešní situaci můžeme bez ostychu popsat již zevšedněným rčením „Kdo není na internetu, jako by nebyl“. To se velmi podepisuje na tvůrcích webových prezentací, kteří jsou tlačeni do stále pestřejších a designově náročnějších výtvorů.

Toto všechno by dnes zcela jistě nebylo možné bez nástroje, který usnadní formátování dokumentu ve všech možných směrech. Tento nástroj zvaný „kaskádové styly“ je podrobně rozebrán v první části bakalářské práce.

Při vývoji kaskádových stylů je velmi důležitá podpora cílových prohlížečů, na kterých v podstatě záleží, jak se budou styly interpretovat. Dá se tedy říci, že budoucnost kaskádových stylů je částečně v rukou tvůrců internetových prohlížečů.

Webdesignéry čeká při používání kaskádových stylů mnoho nepříjemností souvisejících především v rozdílné interpretaci v klientských prohlížečích. Tyto problémy, na které postupem času narazí každý, kdo vytváří webové prezentace, je možné obejít různými způsoby, ale ve většině případů to má za následek nedodržení předepsaného standardu. Řešení tohoto problému není zatím jasné. Vytvoření jednoho univerzálního prohlížeče je nemožnou volbou, ať už z hlediska konkurence, tvorby či volného výběru produktů.

Zcela určitě jsou však kaskádové styly nedílnou součástí každého moderního internetového dokumentu, který má za úkol zaujmout nejen obsahem, ale také po grafické stránce.

2. Cíl práce a metodika

Má bakalářská práce na téma „Použití kaskádových stylů v internetových technologiích“ bude rozdělena na dvě ucelené části. První část – literární rešerše – se bude zabývat teoretickými poznatky ohledně použití kaskádových stylů. Cílem této první části bude nastínit syntaxi kaskádových stylů včetně aplikace na dokument. Rozeberu zde také kompletně problematiku selektorů a zmíním i otázku dědičnosti. V příloze uvedu také kompletní výpis vlastností nejpoužívanějších kaskádových stylů. Z teoretických poznatků budu vycházet v druhé části mé bakalářské práce, kde uplatním teoretické znalosti použití kaskádových stylů v praxi. Cílem praktické části bude ukázat na několika příkladech použití kaskádových stylů včetně obrazové ukázky. Stěžejní část mé praktické bakalářské práce bude vytvoření webových stránek pro výukové účely v předmětu Webdesign na Provozně ekonomické fakultě. Popis této tvorby bude popsán také v druhé části mé práce.

V první části – literární rešerši - budu vycházet především ze sekundárních dat a to z analýzy dokumentů.

Ve druhé části budu pracovat jak se sekundárními daty, tak i s primárními.

Moje praktická část je směrována k tvorbě webových stránek v oblasti informačních technologií, budu se proto snažit zvolit takový koncept grafického designu, aby korespondoval s oblastí cílového použití. Pro grafické rozvržení použiji kaskádové styly a pokusím se vytvořit grafickou „hlavičku“ dokumentu, která by se v dokumentu vyjímala, ale neodtahovala pozornost od samotného obsahu.

Na rozvržení stránek použiji metodu pozicování pomocí elementu div a s ním spojené potřebné zásahy pomocí kaskádových stylů. Tuto metodu popíši podrobněji v kapitole 4.3. Rozvržení dokumentu pomocí elementu DIV.

Položky horizontálního menu vytvářených stránek budou obsahovat:

- PEF (Provozně ekonomická fakulta)
- LF (Fakulta lesnická a enviromentální)
- TF (Technická fakulta)
- AF (Fakulta agrobiologie, potravinových a přírodních zdrojů)

Položky vertikálního menu budou obsahovat:

- O předmětu
- Přednášky
- Cvičení
- Docházka
- Otázky ke zkoušce
- Výsledky testů
- Vyučující předmětu
- Literatura
- Zajímavé odkazy

Pro jednotlivé fakulty budou stránky laděny do jejich symbolických barev. Tuto záležitost provedu pomocí grafického programu Macromedia Fireworks včetně zakomponování pomocí kaskádových stylů.

V mé práci budou často použity odborné termíny. Pro přehledné a srozumitelné čtení bude k práci přiložen slovník pojmů, kde budou jednotlivé termíny vysvětleny. Vysvětlovaný pojem bude v práci označen kurzívou. Například *element*.

3. Literární rešerše

3.1. Historie HTML

V době, kdy programátoři vymýšleli značkovací jazyk *HTM* (HyperText Markup Language) se zdálo, že tato myšlenka je absolutně dostačující pro účely, kvůli kterým se tento jazyk vyvíjel. Těmito účely bylo umožnit všem uživatelům přenášet a především zobrazovat dokumenty, které by bylo možné zobrazit v čitelné formě bez ohledu na to, jaké zařízení, jaký systém či jaký program by se k tomu použil. Především pro tuto univerzálnost a přenositelnost byl *HTM* jazyk vytvořen a díky ní také zůstal nejpoužívanějším formátem pro přenos dokumentů po internetu až do dnešní doby. [1]

Zdá se, že tento jazyk má před sebou ještě stále slibnou budoucnost a to i přes to, že první dokumenty v tomto formátu byly vytvořeny již před téměř dvaceti lety. Samozřejmostí je, že jazyk je neustále vyvíjen a za dobu své existence dospěl k jistým změnám, ale jeho hlavní idea univerzálnosti je neustále střežena. Hned za myšlenkou univerzálnosti, stála při vytváření jazyka *HTM* vize propojení jednotlivých dokumentů, přístupných co možná nejjednodušším způsobem. Tento problém byl vyřešen implementací hypertextových odkazů, které dali tvář jazyku *HTM*.

Dokumenty tvořené tímto jazykem byly strukturované, nikoliv však formátované. Jediné, co mohl autor dokumentu provést s formátováním, bylo nastavit, která část textu je odkazem či nadpisem, nebo která část textu se má zvýraznit či zkosit. To, jakým způsobem, bylo už jenom na cílovém programu (prohlížeči). Nebylo tedy možné jednoduše pozicovat text ani obrázky na stránce dokumentu. To se ukázalo jako problém ve chvíli, kdy se začali o formát jazyku *HTM* zajímat komerční firmy, které chtěli tento formát použít pro své účely. Byly jimi samozřejmě marketingové aktivity. V tuto chvíli nastupují na

scénu tabulky. Tabulkami je myšlena značka (*tag*) <TABLE>. Tabulky se na čas staly nástrojem pro pozicování a formátování dokumentu.

Jednou z hlavních nevýhod tohoto stylu formátování je však bezesporu doba načítání stránky. Tabulky mají totiž jednu nepříjemnou vlastnost. Tou je zobrazení obsahu až po jeho úplném načtení, což může být problém v případě komplexnějších stránek. Dalším problémem je nestrukturovanost dokumentu. Tento problém je velmi skrytý přesto pro udržení koncepce jazyka *HTM* zásadní. Je-li totiž stránka formátována tabulkami, nejde obsah v kódu za sebou postupně od nejdůležitějšího k méně důležitému, ale podle toho, co se má kdy vykreslit na obrazovce. [1]

Tabulky tedy přinášejí relativní volnost při formátování dokumentu a umožňují dát jí vizuálně přijatelnější podobu. Cenou za to je však velký objem dat, zničení struktury dokumentu a pomalejší vykreslování stránky.

3.2. Kaskádové styly 1

Reakcí na vývoj *HTM* jazyku bylo vytvoření kaskádových stylů - Cascading Style Sheets (*CSS*). Zásadní myšlenkou těchto stylů bylo oddělit informace o obsahu od pokynů pro formátování. *CSS1* vznikly v roce 1996. Mezi první a velmi rozšířený prohlížeč, který podporoval *CSS1*, je možné považovat Internet Explorer verze 3. V tomto prohlížeči se podpora omezovala pouze na písma a barvy. [5]

Daleko lepší podpory se *CSS* dočkaly až v roce 1998, kdy vznikly nové verze prohlížečů Internet Explorer (4) a Netscape Navigator (4). V těchto nových verzích byla již zakomponována podpora všech funkcí *CSS* včetně pozicování. Tyto verze byly samozřejmě za dobu svého používání několikrát aktualizovány s optimalizovanou podporou.

3.3. Kaskádové styly 2

Logickým vývojovým stupněm po *CSS1* bylo vytvoření *CSS2*, které doznalo jistých změn a vylepšeních, jež byly vynuceny praxí. *CSS2* je plně zpětně kompatibilní se starší verzí a zároveň standardizuje nové vlastnosti.

Prohlížet dokumenty si mohou už i různě handicapovaní lidé. Navíc monitor již není jediným zobrazovacím zařízením, na kterém si lidé dokumenty prohlížejí. Z těchto i jiných důvodů byla přidána do *CSS2* podpora více různých médií. Podporována jsou tato zařízení: hlasové syntetizátory, on-line zobrazovače braillova písma, tiskárny braillova písma, malé obrazovky (PDA), tiskárny, slide pro zpětné projektory, obrazovky, znakové terminály a televizní obrazovky.

Například pro hlasové syntetizátory je možné v *CSS2* nastavit pro každý element relativně hodně vlastností. Od velikosti prodlev a hlasitosti, přes sílu a tón hlasu až k směru odkud hlas přichází.

Další vylepšení přišlo v podobě optimalizované práce s *fonty* a jejich samotné stahování. V předchozí verzi bylo možné celkem podrobně popsat kompletní řez písma, ale problém nastal ve chvíli, kdy na klientském počítači nebyl daný druh řezu nainstalován. V takovémto případě se použil jakýkoliv font. Záleželo již jenom na prohlížeči, který font zobrazí. Toto se vyřešilo přidáním selektoru @font-face. Díky němu je možné jak písmo definovat, tak zadat i *URL*, ze které se dá písmo stáhnout v případě, že ho klient nevlastní. [5]

Zásadní bylo také umožnění absolutního pozicování a s tím spjaté nastavení pořadí elementů ve vrstvách (překrývání). Jde v podstatě o to, že každému elementu se nastaví, v kolikáté vrstvě bude zobrazen. Tím je možné dosáhnout například částečného či úplného překrytí některých elementů na stránce a pomocí dynamické části (například *Javascriptu*) řídit jejich skrývání nebo naopak odkrývání.

Jako poslední výraznou novinku bych zmínil vylepšené používání *selektorů*. *Selektorům* jako takovým se budu podrobněji věnovat v kapitole 3.6. Selektory. Jako zmínku bych uvedl například zápis kontextového selektoru A > B, který značí, že zadané vlastnosti se budou aplikovat pouze na *elementy* B obsažené jen v *elementu* A.

3.4. Syntaxe

Tabulku *CSS* tvoří seznam položek. Položkou v tomto případě může být *selektor{definice}* nebo *@-pravidlo. Selektor* popisuje, na který *element* či *elementy* budou určené vlastnosti použity. V případě in-line stylů (viz dále) se *selektor* vynechává. *Selektorem* je v takovémto případě sám *tag*.

Blokové CSS má specifickou syntaxi, která vypadá následovně:

komentář : <!-- text --> , /* text */
@-pravidlo : @direktiva parametry - např. @import url(...)
pravidlo : selektor { definice [; definice] }
selektor : název elementu [#název identifikátoru | .název třídy | složené deklarace]
definice : vlastnost: hodnota

Oproti tomu syntaxe in-line stylu vypadá takto:

<TAG styl>

TAG : všechny tagy kromě BASE, BASEFONT, HEAD, HTML, META, PARAM, SCRIPT, STYLE a TITLE
styl : style="definice [; definice]"
definice : vlastnost: hodnota

3.5. Aplikace CSS

CSS je možné aplikovat na dokument třemi způsoby, a to:

- pomocí definice externího souboru,
- pomocí definice in-line stylu,
- jako samostatný *HTML* tag.

Definice externího souboru je asi nejelegantnějším řešením aplikace *CSS*. Jednou z výhod je bezesporu zamezení duplicity kódu. Další výhodou je snadná úprava stylů v případě velkého počtu dokumentů. Té je dosaženo díky tomu, že do jednoho externího souboru se nadefinují styly pro celý web (v praxi se použije spíše více souborů pro specifikaci jednotlivých sekcí pro úsporu kódu) a v každé stránce je pouze odkaz na tento soubor s *CSS*. Vložení odkazu je možné několika způsoby.

Jeden ze způsobů je pomocí *tagu* <link>:

```
<link href="css.css" rel="stylesheet" type="text/css" media="all">
```

Tímto kódem se docílí namapovaní souboru *css.css* do dokumentu přičemž tento soubor se bude aplikovat na všechny typy médií.

Dalším způsobem je vložit odkaz do *tagu* <style></style>:

```
<style type="text/css">
@import url("css.css");
</style>
```

V případě tzv. in-line definice *CSS* dochází k vepisování jednotlivých stylů přímo do jednotlivých *tagů* pomocí atributu s*tyle*. Přičemž veškeré styly nastavené v daném *tagu* jsou aplikovány pouze na tento jeden *element*. Samozřejmě, že v každé definici stylů platí základní vlastnost CSS, a to dědičnost jednotlivých vlastností. V důsledku to znamená, že veškeré vlastnosti *parent* elementů přebírají jejich *child* elementy. Jiné vlastnosti by se použily pouze v případě, že by byly tyto jiné vlastnosti nadefinovány přímo na *child element*.

Pro příklad uvedu tag s in-line definicí CSS:

```
<div class=“zarovnane“ style=“text-align:left; color: #000000; font-weight:bold“></div>
```

V příkladu by se na tento *element div* použily vlastnosti zarovnání textu vlevo, barva textu černá a tučné písmo.

Jako poslední způsob aplikace stylu na dokument je vložení *tagu* <style></style>. Tato metoda je téměř nepoužitelná v případě, že se má styl použít na více dokumentů. Je velmi obtížné efektivně spravovat tyto styly při větším počtu dokumentů.

Tento způsob vypadá následovně:

```
<style>
div .zarovnane {
        text-align:left;        /* zarovnání textu na levý praporek */
        color: #000000;         /* černá barva textu */
        font-weight:bold;       /* tlusté písmo */
        }
</style>
```

V případě použití více způsobů vkládání *CSS* do dokumentu je pořadí priorit následující. Jako nejdůležitější, a tedy všechny ostatní způsoby přepisující je deklarace in-line stylu. Další v pořadí je styl v definici *tagu* <style></style> a nejmenší prioritu má definice stylu v externím souboru.

3.6. Selektory

Aby bylo možné odlišit elementy stejného druhu, je v *CSS* zahrnut systém identifikace, který je tvořen tzv. *selektory*. Do tohoto systému patří:

- třídy,
- identifikátory,
- složené deklarace,
- pseudo-třídy,
- pseudo-elementy.[5]

3.6.1. Třídy

Narazíme-li na situaci, kdy potřebujeme, nastavit pro některé *elementy* jiné vlastnosti než pro zbytek členů *elementu* stejného typu, přichází na řadu

třídy a identifikátory. Třídy jsou používány v případech, kdy víme, že bude potřeba tento styl nastavit pro více *elementů* daného typu. Znamená to tedy, že třídy nejsou jednoznačnými identifikátory. Toto je jediný rozdíl mezi třídou a identifikátorem. [6]

Nastavení třídy *elementu*:

```
<div class="zarovnane"></div>
```

Nastavení vlastností v *CSS*:

```
div .zarovnane {
        float:left;             /* obtékání elementu textem z prava */
        color: #000000;         /* černá barva textu */
        font-size:12px;         /* velikost písma 12 pixelů */
        font-weight:bold;       /* tlusté písmo */
        }
```

Zásadní v identifikaci třídy je atribut *class v tagu elementu* a symbol „.“ před názvem třídy v definici stylu. Tento symbol říká prohlížeči, že následující slovo bude znamenat název třídy a podle toho se již prohlížeč příslušně postará o zobrazení.

3.6.2. Identifikátory

Jak bylo již dříve zmíněno, identifikátory jsou používány pro jednoznačné označení elementu. Používají se, jak pro užití při definici *CSS*, tak pro účely různých skriptovacích jazyků a to především pro jednoznačný přístup k jednotlivým *elementům*. Nastavení identifikátorů je analogické k nastavení tříd. [6]

Nastavení identifikátoru *elementu*:

```
<div id="zarovnane"></div>
```

Nastavení vlastností v *CSS*:

```
div #zarovnane {
        float:left;             /* obtékání elementu textem z prava */
        color: #000000;         /* černá barva textu */
        font-size:12px;         /* velikost písma 12 pixelů */
        font-weight:bold;       /* tlusté písmo */
        }
```

Zde slouží pro nastavení v *tagu* slovo *id* a symbol „#“ při definici stylu.

3.6.3. Složené deklarace

Jako první vlastnost CSS, kterou je možné zmínit v této sekci je tzv. hromadná deklarace. Hromadná deklarace byla vytvořena především pro úsporu kódu. Je tedy možné nadefinovat vlastnosti *CSS* pro několik různých *elementů* v jednom bloku. Příklad je následující: [6]

```
h1, h2, h3, h4 {
        margin: 10px 10px 20px 10px;    /* odsazení v pořadí (nahoře, vpravo, dole,
                                           vlevo) */
        color: #000000;                 /* černá barva textu */
        font-size:12px;                 /* velikost písma 12 pixelů */
        font-weight:bold;               /* tlusté písmo */
        }
```

V tomto příkladu jsou nastaveny stejné vlastnosti pro nadpisy čtyř úrovní.

Další součást složené deklarace je tzv. kontextová deklarace. Kontextové deklarace umožňují pomocí selektorů odlišit elementy podle toho, kde se v dokumentu nacházejí. Jako první bych uvedl selektor, který se nazývá *selektor* potomka. Díky tomuto *selektoru* je možné definovat styly pro *element*, na který se budou aplikovat vlastnosti pouze v případě, že bude přímým potomkem předem určeného *elementu*. Velmi podobným *selektorem* je velmi starý prvek již z *CSS1* a to prvek, který zajišťuje to samé co *selektor* potomků, s tím rozdílem, že potomek nemusí být přímý, ale pouze vnořen kdekoliv v předem nadefinovaném *elementu*.

Selektor potomka:

```
div > a {
        margin: 3px 5px 3px 5px;        /* odsazení v pořadí (nahoře, vpravo, dole,
                                           vlevo) */
        color: darkblue;                /* tmavě-modrá barva textu */
        font-size:12px;                 /* velikost písma 12 pixelů */
        font-weight:bold;               /* tlusté písmo */
        }
```

Styl bude aplikován v tomto příkladu pouze na *elementy „a“*, které budou přímým potomkem *elementu div.*

Selektor následovníka:

```
div a {
        margin: 3px 5px 3px 5px;        /* odsazení v pořadí (nahoře, vpravo, dole,
                                           vlevo) */
        color: darkblue;                /* tmavě-modrá barva textu */
        font-size:12px;                 /* velikost písma 12 pixelů */
        font-weight:bold;               /* tlusté písmo */
        }
```

V poslední řadě zmíním selektor sourozenců. Díky tomuto *selektoru* je možné nadefinovat styl pro *elementy*, které spolu sousedí. Znamená to tedy, že aby se na *element* vztahovaly vlastnosti, musí být v kódu *HTML* zapsány ihned po sobě.

```
span + p {
        margin: 3px 5px 3px 5px;        /* odsazení v pořadí (nahoře, vpravo, dole,
                                           vlevo) */
        color: darkblue;                /* tmavě-modrá barva textu */
        font-size:12px;                 /* velikost písma 12 pixelů */
        font-weight:bold;               /* tlusté písmo */
        }
```

Kód *HTML*:

```
<span>Text v tagu span</span>
<p>text, na který se bude aplikovat styl CSS</p>
<p>text, na který se nebude styl aplikovat</p>
```

3.6.4. Selektory s atributy

U každého *selektoru* je také možné specifikovat *atributy*, které mají být uvedeny v tagu HTML, aby se daný styl použil.

Selektor s kontrolou existence atributu A[*atribut*]{...} aplikuje styl pouze na element, který obsahuje v HTML zvolený atribut, avšak nezajímá se o hodnotu tohoto *atributu*. Pro tento případ existuje selektor s atributem dané hodnoty A[*atribut*=hodnota]{...}. Takto nastavený styl bude aplikován pouze na *element* A, který bude obsahovat *atribut* rovnající se hodnotě. [6]

```
a[title]{...}                    /* aplikace stylu na odkaz, který má nastavený atribut title */
a[href="http://www.seznam.cz"]{...}    /* aplikace stylu na odkaz, který má nastavenou hodnotu "http://www.seznam.cz" v atributu "href" */
```

Například v ukázce by se vlastnosti aplikovaly na tento *tag:*

```
<a href="http://www.seznam.cz"></a>
```

Existují případy, kdy v *atributu elementu* je nastaveno více hodnot oddělených mezerou. Potřebujeme-li tedy vybrat pouze jednu hodnotu daného *atributu*, je možné použít *selektor* s atributem obsahujícím hodnotu A[atribut~=hodnota]. Podobnou variantou je selektor pro jazykové varianty. Liší se pouze v tom, že atribut musí přímo začínat zvolenou hodnotou. Jazykové selektory vypadají například takto:

```
*[lang|=cs]{...}
```

V příkladu by se vlastnost aplikovala pouze na *element*, který by obsahoval *atribut „lang“*, který by musel začínat na hodnotu *„cs“.*

3.6.5. Pseudo-třídy a pseudo-elementy

Pseudo-třídy a pseudo-elementy umožňují formátovat informace nad rámec zdrojového kódu dokumentu, na rozdíl od ostatních selektorů, které

vycházejí striktně z informací uvedených v kódu dokumentu. Díky pseudo-třídám je možné zpřístupnit informace pro formátování, které nelze zjistit ze struktury dokumentu, dále dynamické změny a výsledky interakce dokumentu s uživatelem. [6]

Pseudo-třída A:first-child je jedinou výjimkou, která vychází ze struktury dokumentu. Pokud použijeme tuto pseudo-třídu, definované vlastnosti budou použity pouze na *element*, který je prvním potomkem *elementu* A. Důležité je ještě poznamenat, že tato pseudo-třída se musí používat pouze v kombinaci se *selektorem* potomka nebo *selektorem* následovníka.

```
div > a:first-child {...}
```

Pseudo-třídy odkazů A:link a A:visited nastaví vlastnosti odkazů podle toho zda-li byl již daný odkaz navštíven či nikoliv. Je samozřejmě možné tuto pseudo-třídu kombinovat se selektory s hodnotou atributu. Uvedu příklad:

```
a[href="http://www.seznam.cz"]:visited {color:darkblue}
```

Tento příklad zajistí, že každý odkaz, který má jako *atribut „href“* s hodnotou "*http://www.seznam.cz*" a již byl uživatelem navštíven, bude zobrazen tmavě-modrou barvou.

Programově se prohlížeč o tom, zda je stránka již navštívena uživatelem nebo ne, dozví z komponenty, která udržuje historii a zaručuje rychlejší načítání stránek (tzv. cachování - ve WindowsXP se tyto informace udržují v adresáři "Temporary Internet Files").

Mezi dynamické pseudo-třídy se řadí A:hover, A:focus a A:active. Varianta A:hover, funguje tak, že v případě, že uživatel ukáže na daný element (například přejede kurzorem myši nad element) aktivuje se vlastnost tohoto stylu. Jakmile uživatel ukáže na jiný element, vrátí se vlastnosti předchozího elementu na původní hodnoty. A:focus funguje obdobně jako A:hover pouze s

tím rozdílem, že element musí být aktivní. To se používá například ve formulářích tak, že prvek přebírá vlastnosti této pseudo-třídy, když je do něj cokoliv zapisováno (funguje obdobně i například u zaškrtávacích políček). Jako poslední prvek z této sekce je tedy A:active. Funguje opět podobně jako dvě předchozí pseudo-třídy. Vlastnost je aktivována v době mezi stiskem tlačítka myši a jeho uvolněním. Všechny tyto prvky je možné kombinovat jak mezi sebou tak i s jinými typy selektorů.

Příklad A:hover:

```
a {...}
a:hover {....}
```

Jako poslední mezi pseudo-třídami je jazyková třída A:lang(). Tento selektor je možné použít při aplikaci na celý dokument ve spolupráci s *tagem* <meta> a jejím atributem lang="...". Znamená to tedy, že je možné aplikovat různé vlastnosti podle jazykových mutací dokumentu.

Příklad:

```
html:lang[cs] {...}
html:lang[en] {...}
```

Nakonec stojí za zmínku pseudo-elementy A:first-line a A:first-letter. A:first-line aplikuje styl pouze na první řádku daného elementu a to podle toho, jak je prohlížečem podle velikosti stránky naformátován. Změní-li se velikost okna prohlížeče, změní se i velikost řádku a tím i počet slov, na která bude styl aplikován. Stejnou vlastnost má i A:first-letter pouze s tím rozdílem, že aplikace stylu se provede pouze na první písmeno elementu.

```
a:first-letter {...}
```

3.7. Jednotky

Některé vlastnosti *CSS* mohou nabývat také číselných hodnot. Tyto jednotky se zadávají buď ve formátu celých čísel, nebo ve formátu reálných čísel, přičemž oddělení desetinné části se provádí tečkou.

Základní jednotky můžeme rozdělit na dvě skupiny a to relativní a absolutní velikosti. Relativní velikosti jsou vztažené k jiné hodnotě. Například k zobrazovacímu zařízení. [6] Mezi tyto velikosti patří:

```
p {
        font-size:2.1em;        /* velikost písma bude 2,1 násobek velikosti písma
                                rodičovského elementu (em = čtverčík)  */
        font-size:0.6ex;        /* velikost písma bude 0,6 násobek velikosti minuskulí písma
                                rodičovského elementu */
        font-size:10px;         /* velikost písma bude 10 obrazovkových bodů */
}
```

Absolutní velikosti se používají v případech, kdy autor zná přesné rozměry výstupního zařízení, na kterém dojde k zobrazení dokumentu. Mezi tyto velikosti patří:

```
p {
        font-size:2cm;          /* velikost písma bude 2 centimetrů (reálných)  */
        font-size:22mm;         /* velikost písma bude 22 milimetrů (reálných)*/
        font-size:1in;          /* velikost písma bude 1 palec (reálný) */
        font-size:30pt;         /* velikost písma bude 30 typografických bodů = 30/72 palce
        */
        font-size:1pc;          /* velikost písma bude 1 pica = 12pt + 1px */

}
```

Další možností jak nastavit velikost hodnoty je v procentickém vyjádření. Procentní hodnota je vždy relativní k nějaké jiné vlastnosti, která je udávána v pevné velikosti. V naprosté většině případů se procentní hodnota vztahuje ke stejné vlastnosti mateřského *elementu*. Může však být vztažena například k formátovacímu kontextu (například šířka omezujícího bloku).

```
p {
        font-size:150%;         /* velikost písma bude 150% velikosti písma nadřazeného
prvku */
}
```

3.8. Barvy

Barva se v *CSS* definuje pomocí klíčového slova, nebo pomocí číselné specifikace RGB (červená, zelená, modrá). Číselné specifikace RGB mohou mít několik forem. [6] Všechny následující příklady budou mít za následek nastavení písma na červenou barvu v elementu *body.*

```
body {
        color:#FF0000;          /* Každá barva má dvě místa kam se zapisují hodnoty
jednotlivých barev. Hodnoty jsou v šestnáctkové soustavě a jsou v rozmezí 00-FF */
        color:#F00;             /* Tímto způsobem se dají zapsat pouze tzv. „bezpečné
barvy“. Každá hodnota může nabývat hodnot 0,3,6,9,C a F */
        color:rgb(255,0,0);     /* Podobné jako první způsob, přičemž hodnoty jsou
v desítkové soustavě a nabývají hodnot 0 - 255 */
        color:rgb(100%,0%,0%);          /* Stejné jako předchozí metoda, ale místo
absolutních hodnot se vkládají hodnoty procentní. Můžou nabývat hodnot 0% - 100% */
}
```

Hodnotou barvy zde rozumím intenzitu vyzařování jednotlivých složek barevného modelu.

3.9. Dědičnost

Některé vlastnosti *CSS* se ve stromové struktuře dokumentu "dědí". Znamená to, že ve chvíli, kdy není u *elementu* definována vlastnost, která se dědí, je automaticky převzata tato vlastnost z *elementu* nadřazeného. A to nejenom od přímého rodiče, ale prohlížeč pokračuje stromovou strukturou stále výše a výše, dokud nenalezne nastavenou danou vlastnost a poté ji nastaví u všech podřazených *elementů*. Jakmile tuto vlastnost ve stromové struktuře nenalezne, nastaví hodnotu, která je přednastavena v prohlížeči. Dědičnost se dá přepsat pouze tím, že u *elementu*, u kterého chceme jinou hodnotu vlastnosti, ji nastavíme přímo na tento *element*. Mezi takto děděné vlastnosti se započítává většina nastavení týkajících se fontů. Například velikost písma, barva písma nebo řez písma. [6]

3.10. Kaskádovitost

Styly jako takové může dokument získat od tří zdrojů. Patří mezi ně autor dokumentu, uživatel a klientský prohlížeč. Jak jsem již dříve napsal, definuje autor styly buď přímo v dokumentu, nebo v externích souborech. Uživateli je umožněno si v některých prohlížečích nastavit styly svoje a upravit si tak vzhled dokumentů podle svého. Základem pro všechny styly je výchozí tabulka stylů, kterou má každý prohlížeč přednastavenu a hodnoty z ní se použijí tehdy, není-li v předchozích dvou zdrojích daná vlastnost definována.

Přidělování stylů funguje tak, že se jednotlivé styly ohodnotí podle váhy. Jakmile se tedy setkají dvě stejná pravidla u jednoho *elementu*, použije se to s větší váhou. Styly definované autorem a uživatelem mají vždy přednost před styly klientského prohlížeče. Autorův styl má přednost před uživatelovým vždy, jedinou výjimku tvoří direktiva "!important", která nastaví prioritu uživatelova stylu nad autorovu. [5]

3.11. Přehled vlastností CSS

Počet vlastností *CSS* je opravdu veliký a jejich kompletní výpis by vydal na celou knihu. Proto jejich část uvádím v příloze č. 1. Kompletní výpis vlastností je možné najít na internetových stránkách World Wide Web Consorcium (W3C), které jsou umístěny na http://www.w3c.org.

3.12. Různá interpretace CSS

Všeobecně známou záležitostí je poměrně různá interpretace *CSS* různými prohlížeči. S těmito eventuelními odlišnostmi musí autor dokumentu počítat. Tento problém je možné obejít několika způsoby (tzv. "*hacky*"). Těmto *„hackům“* se budu věnovat níže. Nyní bych rád zmínil několik zásadních odlišných interpretací u dvou nejrozšířenějších prohlížečů a to Internet

Exploreru 6 a Mozilly 1.7. Ostatní prohlížeče jako je Opera nebo Safari nejsou zcela určitě zanedbatelné, ale v mé práci s nimi nebudu pro jednoduchost počítat.

3.12.1. BOX-model

Každý *element* je tvořen tzv. box-modelem, který udává celkovou výšku a šířku elementu. Mezi vlastnosti box-modelu se započítávají

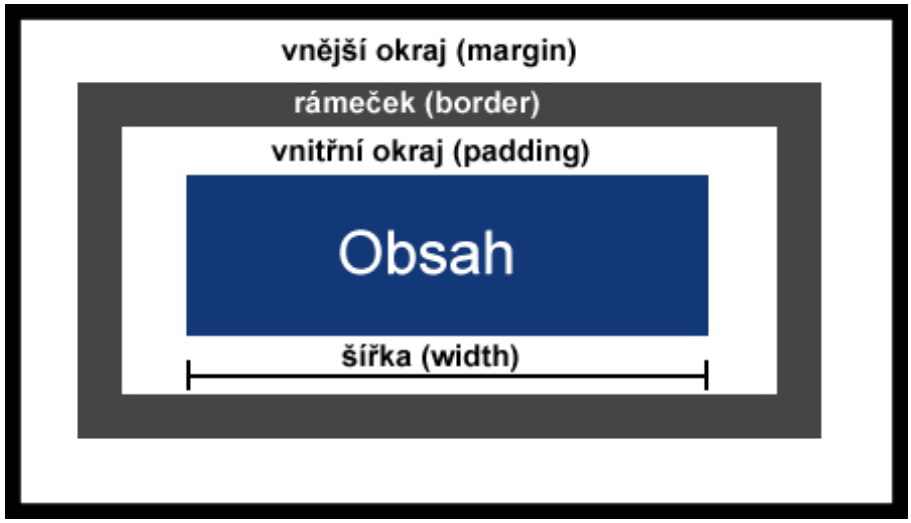
- margin (vnější odsazení),
- border (rámeček),
- padding (vnitřní odsazení),
- samotný obsah elementu.

Všechny tyto vlastnosti je možné nastavit buď jedním číslem pro všechny čtyři strany (horní, dolní, pravá a levá), nebo zvlášť pro jednotlivé strany. Existuje i možnost tzv. shorthandu, který se zadává v pořadí horní část, pravá část, dolní část a levá část.

Příklad shorthandu:

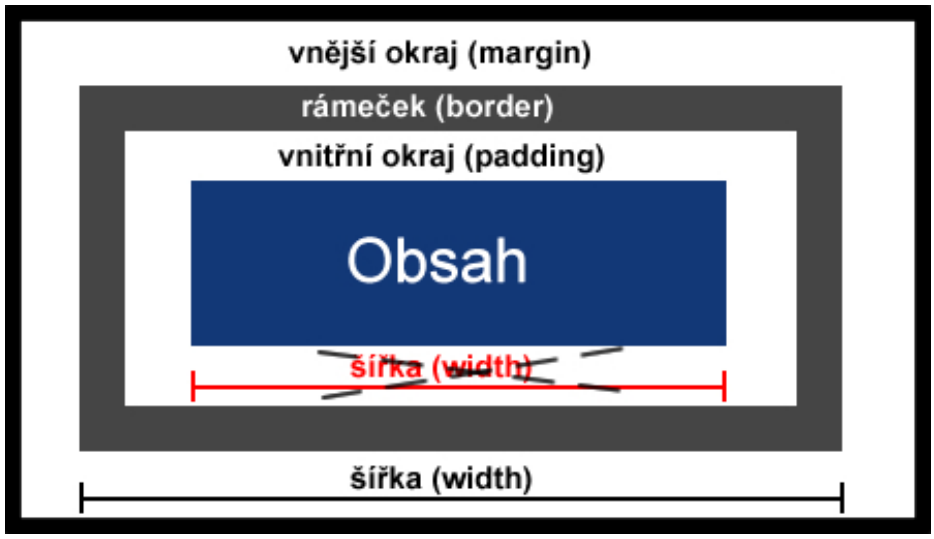
```
div {
        margin:10px 5px 8px 12px;               /* Nastavení odsazení v posloupnostech
nahoře 10 pixelů, vpravo 5 pixelů, dole 8 pixelů a vlevo 12 pixelů  */
}
```

Nyní se dostáváme k rozdílné interpretaci. Podle modelu W3C vypadá box-model následovně.



Obrázek č.1: BOX-model podle standardu

Avšak Internet Explorer 5 i 6 se chovají k box-modelu jinak. Rozdíl je v tom, že nesprávně započítává do šířky obsahu i padding a v některých případech i border. Vše je srozumitelně znázorněné na dalším obrázku.



Obrázek č.2: BOX-model v IE

To je samozřejmě problém při pozicování různých elementů a autor musí s těmito nesrovnalostmi počítat.

Je několik způsobů jak na box-model "vyzrát". Mezi nejjednodušší patří nastavení v hlavičce dokumentu doctype na striktní model. V tomto případě by se Internet Explorer (IE) měl chovat striktně podle W3C. Tento způsob však není příliš spolehlivý.

Příklad hlavičky striktního dokumentu (*XHTML*):

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
p://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

Další možností jsou již výše zmiňované "*hacky*". Dalo by se říci, že tyto "hacky" jsou vlastně využité chyby prohlížečů.

Nyní uvedu příklady, které zajistí, že daná vlastnost se v IE interpretuje a v Mozille ne.

```
div{
        width=100px;
        width="100px";
        width:"100px";
        _width:100px;
}
```

Všechny tyto zápisy by Mozilla ignorovala a IE by je interpretoval, přestože jsou podle standardu chybné. Pomocí těchto hacků je možné i relativně bezpečně směrovat styly pouze pro cílové typy prohlížečů.

Asi nejkvalitnějším a validním způsobem, jak vyřešit problém s nastavením vnitřního okraje (padding), je bezesporu použití dvou stejných *elementů* vnořených do sebe a následné nastavení vnitřnímu *elementu atribut* margin. Pak se bude jako šířka obsahu interpretovat jako šířka vnitřního *elementu* a vnější *element* bude v podstatě použit jako rámeček.

3.12.2. Natahování elementu DIV

V dnešní době je moderní pozicovat prvky v dokumentu především pomocí blokových *elementů* tzv. DIVů. Zde se opět IE a Mozilla rozcházejí v jedné zásadní vlastnosti. Jakmile je u divu nastavena pevná šířka a výška a jeho obsah přesáhne výšku , je v IE automaticky DIV natažen na potřebnou velikost. V Mozille tomu tak není. Znamená to, že v Mozille by v takovémto případě obsah divu přesahoval vně.

S tímto souvisí i problém, který je chybou v Mozille a to, že jsou-li v jednom mateřském *divu* dva a více *divů*, které mají nastavenou vlastnost float (obtékání), je automaticky výška mateřského *divu* nastavena na 0.

Oba předchozí problémy je možné obejít pomocným divem, který bude na konci obsahu mateřského *divu* a bude mít nastaveny vlastnosti "height:1px" a "clear:both".

Zmíněná vlastnost "clear:both" zruší obtékání a donutí tak Mozillu, aby prodloužila mateřský *div* až na konec pomocného elementu.

4. Programová realizace

V této sekci se budu věnovat praktickým ukázkám použití *CSS* a následnému použití při moderním rozvržení stránky (layoutu). Pro jednoduchost budu uvádět kód *HTML* bez hlavičky i bez nastavení *DOCTYPE* dokumentu. K jednotlivým příkladům bude zobrazena grafická ukázka dokumentu před a po aplikaci daného stylu. Rozsáhlost všech dostupných a využitelných funkcí *CSS* je natolik veliká, že v těchto ukázkách použiji pouze ty nejzákladnější a nejpoužívanější funkce. Pro názornost použití *CSS* to však bude bezesporu stačit.

4.1. Formátování textu

Formátování textu je jednou ze základních funkcí *CSS*. Funguje v téměř nezměněné podobě již od první verze *CSS*. Pomocí této direktivy je možné nastavit většinu známých vlastností písma od řezu, velikosti přes tučnost a styl až k nastavení rodiny písma. Tato vlastnost se v *CSS* uvádí klíčovým slovem "*font*-" a navazujícím přívlastkem, který určuje přesnou vlastnost, kterou chceme aplikovat. Pro úsporu kódu je možné použít kolektivní direktivu "*font*", ve které je ale nutné dodržet striktně pořadí zadávaných vlastností a to řez, tučnost, velikost a jméno rodiny písma.

Ukázka formátování textu:

HTML:
```
<p>Formátovaný text</p>
```

CSS:
```
p {
	font: italic bold 15px Verdana;		/* Kurzíva, tučné, velikost 15 obrazových bodů a rodina písma "Verdana" */
}
```

Dokument bez použitého formátování:

Obrázek č.3: Ukázka neformátovaného textu

Dokument s použitým formátováním:

Obrázek č.4: Ukázka formátovaného textu

Použití tohoto stylu je zcela zřejmé a intuitivní. Jak jsem již výše popsal, při nepoužití direktivy *CSS* je písmo nastaveno na přednastavenou hodnotu závisející na klientském prohlížeči. Je tedy v zájmu zachování vytvořeného designu, aby tvůrce dokumentu nastavil striktně všechny hodnoty týkající se textu.

4.2. Pozicování

Pro pozicování se v CSS používá příkaz position, který může nabývat hodnot relative (přednastavená hodnota ve většině prohlížečích), absolute, static a v Mozille navíc ještě hodnoty fixed (znamená, že poloha je fixovaná na jedno místo a to i v případě scrollování dokumentu). My se budeme zabývat

prvními dvěma hodnotami. Hodnota relative znamená, že na pozici *elementu* s touto vlastností jsou ostatní elementy závislé vzhledem k poloze. Opačné je to u hodnoty absolute. V tomto případě je *element* s touto nastavenou vlastností vyjmut z toku dokumentu. Pak jsou tedy všechny ostatní *elementy* s nastavenou hodnotou „position: relative“ nezávislé na tomto dokumentu z hlediska své pozice.

V souvislosti s pozicováním se používají vlastnosti CSS které jsou schopny daný element posunout vzhledem k jejich mateřskému *elementu* (position:absolute) popřípadě pouze k okolnímu textu (position:relative). Tyto vlastnosti jsou top, left, bottom a right.

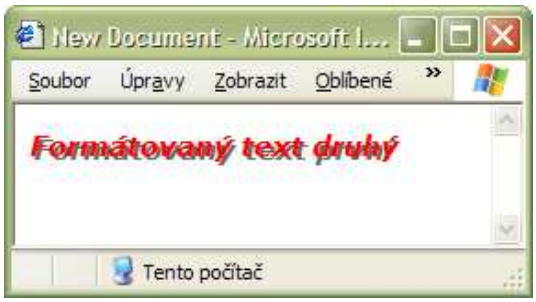
Nyní nastíním jak je možné pomocí pozicování překrýt text a vytvořit tak efekt stínu.

HTML:
```
<p id="prvni">Formátovaný text první</p>
<p id="druhy">Formátovaný text druhý</p>
```
CSS:
```
p {
        font: italic bold 15px Verdana;
        }
#prvni {
        position:relative;      /* Nastavení relativního pozicování */
        color:#666666;          /* Nastavení barvy stínu na šedou */
        }
#druhy {
        position:absolute;      /* Nastavení absolutního pozicování */
        color:#FF0000;          /* Nastavení barvy textu na červenou */
        top:13px;               /* Posunutí elementu o 13 pixelů ze shora vzhledem
k elementu body*/
        left:8px;               /* Posunutí elementu o 8 pixelů z leva vzhledem k elementu
body */
        }
```

Dokument bez použitého formátování:

Obrázek č.5: Ukázka neformátovaného textu bez použití absolutního pozicování.

Dokument s použitým formátováním:



Obrázek č.6: Ukázka formátovaného textu s použitím absolutního pozicování.

Je zde vidět, že překrývání je zajištěno automaticky a aplikuje se způsobem, kdy element vložený v kódu později, je v dokumentu na svrchnější vrstvě. Pro změnu pořadí vrstev existuje direktiva z-index, která nabývá číselných hodnot. Tyto číselné hodnoty v podstatě znamenají to, na kterém pořadí se vrstva daného *elementu* zobrazí. Pokud bychom například chtěli v předchozí ukázce přehodit vrstvy dvou textových elementů p, stačilo by u prvního *elementu* nastavit z-index vyšší než u elementu druhého.

4.3. Rozvržení dokumentu pomocí elementu DIV

Jak jsem již na začátku práce zmínil, rozvržení dokumentu (layout) je možné provést několika způsoby.

Patří mezi ně:

- tabulkový layout
- layout tvořený *divy*
- layout tvořený pomocí absolutního pozicování

Nyní vysvětlím na jednoduchém příkladu, jak se tento problém řeší pomocí blokových elementů *div*. V podstatě se za sebe „poskládají“ jednotlivé *elementy div*, přičemž každý element je jakýmsi kontejnerem pro jednotlivé části stránky. Dále se elementy pomocí obtékání (v relativním pozicování) nebo pomocí vrstev (v absolutním pozicování) nastaví tak, aby výsledné rozvržení vypadalo tak, jak si autor přeje.

Jako příklad uvedu dokument, ve kterém je požadována grafická webová stránka, která má obsahovat hlavičku s grafickým logem a upoutávkou. Dále musí mít dokument vertikální ovládací část (menu) situovanou v levé části obrazovky pod hlavičkou. Posledním požadavkem je, aby se hlavní část stránky, kde se bude zobrazovat hlavní obsah jednotlivých sekcí, nacházel na pravé straně od menu a zároveň pod hlavičkou.

Pro splnění požadavků stačí použít tři *elementy div* a to pro každou část jeden. V kódu budou *elementy* za sebou, a jelikož *element div* je blokový (na konci *elementu* je automaticky zalomení řádku), byly by zobrazeny všechny pod sebou. Proto se použije vlastnost obtékaní „float“, která zaručí, že *elementy* budou vedle sebe.

HTML:

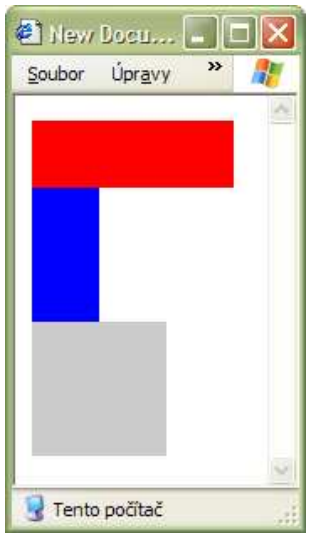
```
<div id="hlavicka"></div>
<div id="menu"></div>
<div id="obsah"></div>
```

CSS:

```
#hlavicka {
        width:120px;                    /* Nastavení šířky kontejneru */
        height:40px;                    /* Nastavení výšky kontejneru */
        background-color:#FF0000;       /* Nastavení barvy pozadí na červenou */
}
#menu {
```

```
        float:left;                 /* Nastavení obtékání z leva */
        width:40px;                 /* Nastavení šířky kontejneru */
        height:80px;                /* Nastavení výšky kontejneru */
        background-color:#0000FF;   /* Nastavení barvy pozadí na modrou */
}
#obsah {
        float:left;                 /* Nastavení obtékání z leva */
        width:80px;                 /* Nastavení šířky kontejneru */
        height:80px;                /* Nastavení výšky kontejneru */
        background-color:#CCCCCC;   /* Nastavení barvy pozadí na šedou */
}
```

Dokument bez použití vlastnosti obtékání:



Obrázek č.7: Ukázka *elementů div* při nepoužití atributu obtékání (float).

Dokument po aplikaci obtékání:

Obrázek č.8: Ukázka *elementů div* s použitím atributu obtékání (float)

Do výsledných kontejnerů je následně možné vkládat jednotlivé části stránek. K deformaci stránek nedojde především díky nastavení velikosti kontejnerů. Znamená to, že nedojde k roztažení *divu* v případě, že jeho obsah je větší než kontejneru. Výjimku tvoří vertikální roztažení v Internet Exploreru 6 i 7. Zde dojde k nežádoucímu natažení kontejneru a možnému rozhození stránky.

5. Přínosy bakalářské práce

Praktická část této bakalářské práce je věnována vytvoření webových stránek pro výuku předmětu WebDesign na Provozně ekonomické fakultě. V dosavadním studiu jsem zatím neabsolvoval předmět WebDesign.

Pro vytvoření „hlavičky“ stránek jsem použil fotografii, na které jsou vyfoceny počítačové monitory. Tuto fotografii jsem pořídil ve fotobance www.sxc.hu, přičemž tato fotka nebyla s žádnými restrikcemi, tedy je volně k veřejnému použití.

Rozvrhl jsem si stránku na pět částí. První z nich je „hlavička“, ve které je zobrazen název předmětu společně s grafikou obsahující i již zmíněnou fotku monitorů. Tu jsem upravoval pomocí demo verze programu Macromedia Fireworks. Další částí je horizontální menu, obsahující odkazy na stránky WebDesignu jednotlivých fakult. Následuje vertikální menu obsahující navigaci pro každou fakultu zvlášť. Hlavní část stránky, kterou je kontejner pro obsah jednotlivých dokumentů, je situován pod horizontálním menu a zároveň napravo od vertikálního menu. Vše je zakončeno tzv. „patičkou“, kde bude zobrazeno, kdo stránky vytvořil, popřípadě datum či další informativní text.

V dokumentu jsem použil pseudo-třídu „hover“ pro dynamickou změnu položky v horizontálním i vertikálním menu. V případě přejetí myší nad jednou z položek vertikálního menu se změní pozadí a pomocí vlastnosti padding se posune text doleva, aby tvořil dynamický efekt vyjetí položky. U horizontálního menu jsem zvolil pouze podtržení položky při přejetí myší.

V dolní části obrazovky je pomocný odkaz „^^^ NAHORU ^^^“, který při aktivaci přemístí obrazovku na horní část dokumentu. Tento prvek je integrován pouze pro zpříjemnění prohlížení stránek z důvodu odpadnutí nutnosti scrollování při větším obsahu stránek.

Pro pozicování jednotlivých kontejnerů, které obsahují vlastní části stránek, jsem použil blokový element *div* spolu s *atributem CSS* obtékání (float),

který zaručí, že je možné položit více *divů* vedle sebe do řádky. Vytvořil jsem si tedy především kontejner pro celou stránku o šířce 700 pixelů a stoprocentní výšce. Dále jsem pod sebe vložil kontejnery pro „hlavičku“ a horizontální menu. Poté jsem musel použít již zmíněnou vlastnost float, která zajistila, aby ležely kontejnery pro vertikální menu a samotný obsah vedle sebe. A nakonec jsem vložil kontejner pro „patičku“.

Nyní popíši několik problémů, se kterými jsem se potýkal při programovém vytváření těchto webových stránek.

Všechny zásadní otázky, které jsem musel při vytváření řešit, vycházeli z již popsaného problému rozdílné interpretace *CSS* různými prohlížeči. První problém týkající se box modelu nastal u horizontálního menu, kde při nastavení kontejneru a vlastnosti padding byla stránka různě zobrazena v IE a v Mozille. Tuto nesrovnalost jsem vyřešil použitím pomocného divu, který jsem zasadil do kontejneru a nastavil mu vlastnost margin ve velikostech původně očekávaných od direktivy padding.

Zde je část kódu stránek, kde jsem řešil tento problém:

HTML:
```
<div id="menu_container">
        <div id="menu_container_inner">
        ...
        </div>
</div>
```

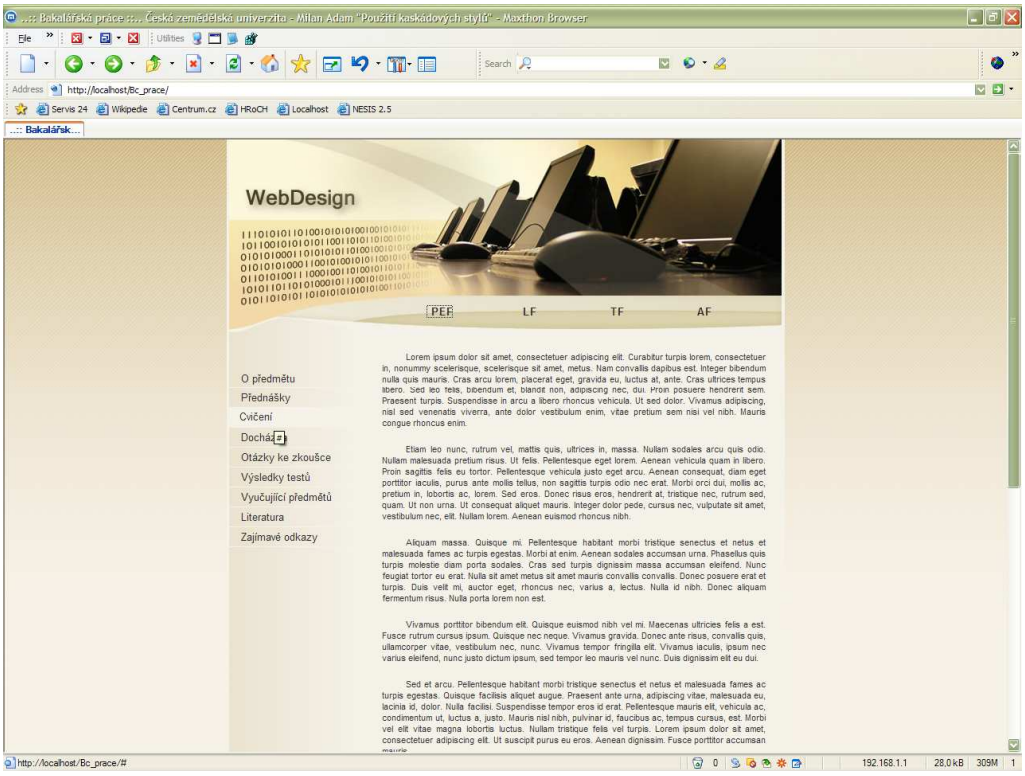
CSS:
```
#menu_container {
        width:700px;
        height:50px;
        padding:0;
        margin:0;
        text-align:left;
}
#menu_container_inner {
        width:694px;
        height:50px;
        padding:0;
        margin:0 3px 0 3px;
}
```

Dalším problémem, který jsem musel řešit, byla chyba v Mozille týkající se přenastavení výšky mateřského *divu* při použití vlastnosti float (popsáno v kapitole 3.12.2 Natahování elementu DIV). Znamená to, že kontejner pro celou stránku, pomocí kterého je nastaven optický rámeček a především výchozí barva pozadí stránky, byl v Mozille nastaven na výšku 1 pixel. Stalo se tak proto, že kontejner pro vertikální menu a samotný obsah stránky jsou položeny vedle sebe právě pomocí vlastnosti float. Znamenalo by to, že celé pozadí stránky i s rámečkem by nebylo vidět. Problém jsem vyřešil použitím pomocného *divu* s následujícím nastavení:

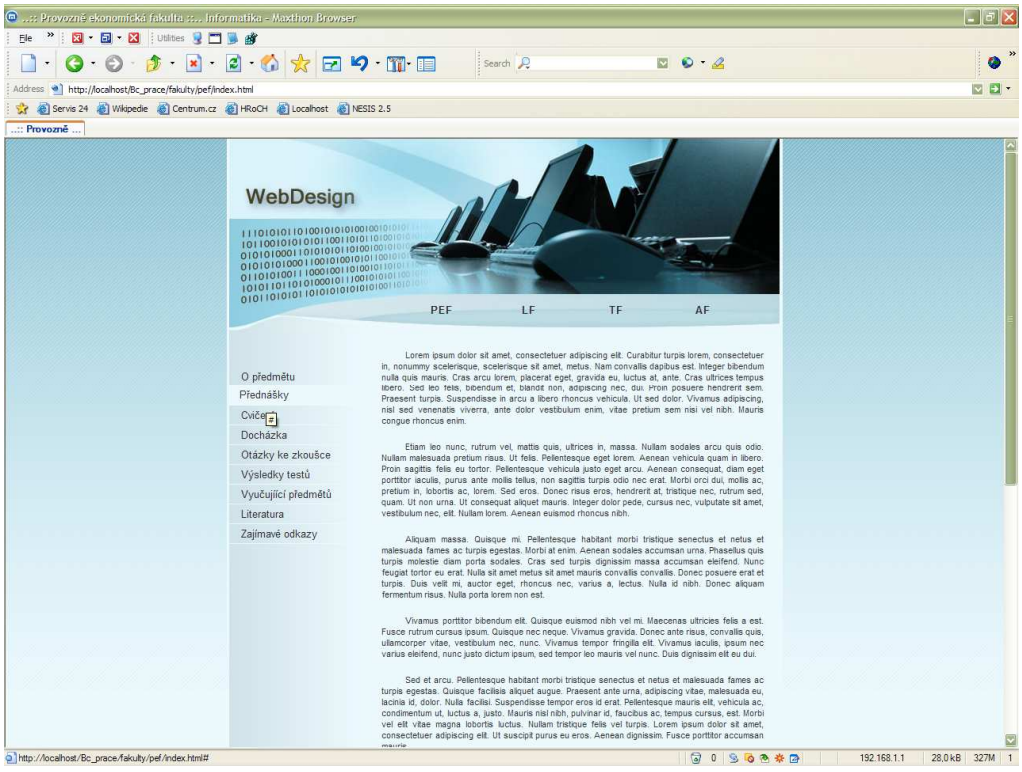
CSS:
```
.clear {
        clear: both;
        height: 1px;
        overflow: hidden;
}
```

Zásadní je v tomto řešení vlastnost „clear: both“. Ta zaručí, že dojde ke zrušení obtékání u tohoto elementu a donutí tak Mozillu, aby natáhla kontejner přes pomocný *div*. Dojde zde sice k natažení celkové stránky o jeden pixel, ale v zájmu dodržení bezbariérovosti stránek je to přijatelné.

Výsledné stránky ve žluté barvě a následné zabarvení stránek do modré vypadají následovně:



Obrázek č.9: Ukázka vytvořené webové stránky ve žluté barvě



Obrázek č.10: Ukázka vytvořené webové stránky v modré barvě

6. Závěr

V dnešní době patří internet a technologie s ním spjaté mezi alfu a omegu informačních technologií. Již v počátcích pociťovali tvůrci internetových dokumentů potřebu nástroje, který by dokázal udržet strukturovanost dokumentů a zároveň by dovedl formátovat dokument podle potřeb autora.

Pro tyto potřeby vznikl nástroj zvaný kaskádové styly – Cascading Style Sheets. V době kdy byla vyvinuta a „vypuštěna“ první verze tohoto nástroje, se tento projekt nedočkal příliš velké podpory u cílových prohlížečů. Toto se změnilo s příchodem druhé verze, která je používána s některými obměnami dodnes.

Cílem první části mé bakalářské práce bylo popsat teoretické poznatky o kaskádových stylech. Detailněji jsem se zabýval syntaxí kaskádových stylů včetně aplikace na dokument. Znalost syntaxe je klíčová pro správné užití kaskádových stylů. Popisuje standardy korektního zápisu *CSS*. Dále jsem se zaměřil na identifikaci *elementů* pomocí tzv. selektorů. Selektory slouží k jednoznačnému odlišení *elementů* v dokumentu.

V teoretické části jsem se také dotkl tématu rozdílné interpretace *CSS* v různých prohlížečích. Jde o jednu ze slabin *CSS*, kterou lze sice obejít pomocí tzv. „*hacků*“, ale v drtivé většině případů to má za následek nedodržení standardů práce s *CSS*. Přestože přisuzuji tuto záležitost jako slabinu *CSS*, je to ve skutečnosti chyba cílových prohlížečů, které jsou příčinou této různé interpretace.

V praktické části jsem vycházel z těchto teoretických poznatků. Jako ukázku použití *CSS* jsem vytvořil několik ukázkových příkladů. Mezi tyto příklady jsem začlenil také dnes velmi využívanou metodu jak rozvrhnout design dokumentu. Jde o pozicování pomocí *elementu div*.

Cílem praktické části bylo vytvořit internetové stránky pro výukové účely předmětu WebDesign na Provozně ekonomické fakultě. Při tvorbě jsem se

potýkal s několika problémy. Jedním z nich byla právě rozdílná interpretace v prohlížečích. Při tvorbě stránek jsem se zaměřil na optimalizaci pouze pro dva nejrozšířenější prohlížeče a to Internet Explorer 6.0 a Mozilla 1.7.5.

Zdrojový kód vytvořených stránek, respektive hlavní strany (*HTML* i *CSS*), je v příloze č. 2.

Pro jednodušší orientaci v množství odborných termínů jsem vytvořil přílohu č. 3 – slovník pojmů.

Kaskádové styly jsou nedílnou součástí internetových technologií. Tvůrci internetových stránek jsou marketingovým trhem tlačeni ke stále pestřejším výtvorům. Pro stále atraktivnější webové stránky mohou využít právě kaskádové styly.

7. Seznam literatury

[1] BROŽA, P. Programování www stránek pro úplné začátečníky. 1. vyd. Praha: ComputerPress, 2000.
[2] GRUSOVÁ, L. CSS pro úplné začátečníky. 1. vyd. Praha: Computer press, 2003.
[3] KUČERA, M. HTML – tipy a triky od profesionálů. 1. vyd. Praha: UNIS Publishing, 2001.
[4] KUČERA, M. CSS Úvod do kaskádových stylů. 1. vyd. Praha: Computer press, 2002.
[5] PROKOP, M. CSS kaskádové styly pro webdesignéry. 2. vyd. Praha: Computer press, 2005.
[6] STANÍČEK, P. CSS Kaskádové styly – kompletní průvodce. 1. vyd. Praha: Computer press, 2003.

Seznam obrázků:

8. Přílohy

Příloha č. 1 – Výpis nejpoužívanějších vlastností CSS [6]

Písmo

Vlastnost - atribut	Hodnota - implicitní hodnoty tučně	Význam	Příklad
font-family	všechny rodiny písma	Druh písma, font	font-family: Arial CE, sans-serif
font-style	normal italic oblique	normální kurzíva skloněné	font-style: italic
font-variant	normal small-caps	normální kapitálky	Font-variant: Small-caps
font-size	xx-small x-small small medium large x-large xx-large výška procento	nejmenší menší malé střední velké větší největší výška zvětšení	font-size: large
font-weight	normal bold bolder lighter 100, 200, 300, 400, 500, 600, 700, 800, 900	normální tučné tučnější světlejší duktus vyjádřený číslem	font-weight: bolder
font	všechny možné předchozí hodnoty nebo systémové písmo		font: italic bold 20px Arial

Text/odstavec

Vlastnost - atribut	Hodnota - implicitní hodnoty tučně	Význam	Příklad
text-decoration	none underline overline	bez dekorace podtržení "nadtržení"	text-decoration: underline

	line-through blink	přeškrtnutí blikání	
text-transform	none capitalize uppercase lowercase	bez změny Začátky Slov Velké VELKÁ PÍSMENA malá písmena	text-Transform: capitalize
word-spacing	normal délka	mezislovní mezera zvětšená o délku	word-spacing: 10px
letter-spacing	normal délka	prostrkání znaků zvětšené o délku	letter-spacing: 5px
line-height	normal výška násobek procento	výška řádku absolutní výška násobek zvětšení	line-height: 8px
text-indent	délka procento	odsazení prvního řádku	text-indent: 50px;
text-align	left right center justify	zarovnání vlevo vpravo na střed do bloku	text-align: left
vertical-align	baseline sub super top text-top middle bottom text-bottom procento	na řádek dolní index horní index co nejvýše vršek k vršku střed na střed co nejníže spodek ke spodku procento výšky	vertical-align: super
display	block inline list-item none	blokový řádkový seznam nezobrazí se	display: block display: none
white-space	normal pre nowrap	normální text předformátovaný nezalamovat	white-space: nowrap

Barvy a pozadí

Vlastnost - atribut	Hodnota - implicitní hodnoty tučně	Význam	Příklad
color	**barva**	barva písma	color:blue
background-color	**barva** transparent	barva pozadí průhledné pozadí	background-color: yellow
background-image	**none** url(cesta)	obrázek na pozadí	background-image: url('pozadi.jpg')
background-repeat	repeat **no-repeat** repeat-x repeat-y	pozadí se opakuje neopakuje opakuje v ose X nebo v ose Y	background-repeat: no-repeat
background-attachment	**scroll** fixed	pozadí se posouvá pozadí se neposouvá při scrollování	background-attachment:fixed
background-position	**top**, center, bottom **left**, center, right, délka, procento	Poloha obrázku na pozadí (nejčastěji pokud se neopakuje)	background-position: right 50%
background	všechny výše uvedé hodnoty		background: url('pozadi.jpg') no-repeat scroll silver center bottom

Velikost

Vlastnost - atribut	Hodnota - implicitní hodnoty tučně	Význam	Příklad
width	**auto** šířka procento	automatická šířka nastavená šířka procento z nadřazeného	width:120px

		elementu	
height	**auto** výška procento	automatická výška nastavená výška procento z nadřazeného elementu	height:500px
float	left right **none**	umístění plovoucího (obtékaného) objektu či zda je neplavec	float: left
clear	left right both **none**	čekání na skončení plovoucích objektů zleva, zprava, obou, nebo žádných	clear:both

Okraje

Vlastnost - atribut	Hodnota - implicitní hodnoty tučně	Význam	Příklad
margin	délka procento **auto**	šířka **vnějšího** okraje procento automatický okraj	margin:5px
margin-top margin-left margin-bottom margin-right	jako u margin	**vnější okraj** horní levý spodní pravý	margin-left:3px
padding	délka procento	šířka **vnitřního** okraje procento	padding:10px
padding-top padding-left padding-bottom padding-right	jako u padding	horní vnitřní okraj levý spodní pravý	padding-right:12px

Rámečky

Vlastnost - atribut	Hodnota - implicitní hodnoty tučně	Význam	Příklad
border-width	thin **medium** thick *délka*	šířka rámečku slabá, normální tlustá nastavená	border-width:5px
border-top-width border-left-width border-bottom-width border-right-width	jako u border-width	horní šířka rámečku levá spodní pravá	border-top-width:8px
border-color	barva	barva rámečku	border-color:red
border-style	**none**, dotted, dashed, solid, double, groove, ridge, inset, outset	**Druh rámečku** žádný, tečkovaný, čárkovaný, plný, dvojitý, příkop, val, ďolík, návrší	border-style:solid
border-top border-left border-bottom border-right	barva, tloušťka a styl	celkové vlastnosti strany rámečku	border-top:1px black solid
border	barva, tloušťka a styl	všechny vlastnosti rámečku	border:3px black dashed

Příloha č. 2 - Zdrojový kód vytvořených stránek

HTML:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1250" />
<title>..:: Bakalářská práce ::.. Česká zemědělská univerzita - Milan Adam "Použití
kaskádových stylů"</title>
<link rel="stylesheet" type="text/css" media="all" href="css/css.css">
<meta name="author" content="Milan Adam">
</head>

<body>
	<center>
		<div id="site">
			<div id="header"></div>
			<div id="menu_container">
				<div id="menu_container_inner">
					<div id="menu_left"></div>
					<div id="menu_right">
						<div class="menu_item_first"><a
href="./fakulty/pef/index.html" title="Provozně ekonomická fakulta"
class="menu_a">PEF</a></div>
						<div class="menu_item"><a
href="./fakulty/lf/index.html" title="Fakulta lesnická a enviromentální"
class="menu_a">LF</a></div>
						<div class="menu_item"><a href="#"
title="Technická fakulta" class="menu_a">TF</a></div>
						<div class="menu_item"><a href="#"
title="Fakulta agrobiologie, potravinových a přírodních zdrojů "
class="menu_a">AF</a></div>
					</div>
				</div>
			</div>
			<div id="window">
				<div id="cont_left">
					<br />
					<br />
					<br />
					<a href="#" title="#" class="left_cont">O
předmětu</a>
					<a href="#" title="#"
class="left_cont">Přednášky</a>
					<a href="#" title="#"
class="left_cont">Cvičení</a>
```

```
                                        <a href="#" title="#"
class="left_cont">Docházka</a>
                                        <a href="#" title="#" class="left_cont">Otázky ke
zkoušce</a>
                                        <a href="#" title="#" class="left_cont">Výsledky
testů</a>
                                        <a href="#" title="#" class="left_cont">Vyučujíící
předmětů</a>
                                        <a href="#" title="#"
class="left_cont">Literatura</a>
                                        <a href="#" title="#" class="left_cont">Zajímavé
odkazy</a>
                                </div>
                                <div id="cont_right">
                                        <div id="text_cont">
                                                <p>Lorem ipsum dolor sit amet,
consectetuer adipiscing elit. Curabitur turpis lorem, consectetuer in, nonummy
scelerisque, scelerisque sit amet, metus. Nam convallis dapibus est. Integer bibendum
nulla quis mauris. Cras arcu lorem, placerat eget, gravida eu, luctus at, ante. Cras
ultrices tempus libero. Sed leo felis, bibendum et, blandit non, adipiscing nec, dui. Proin
posuere hendrerit sem. Praesent turpis. Suspendisse in arcu a libero rhoncus vehicula.
Ut sed dolor. Vivamus adipiscing, nisl sed venenatis viverra, ante dolor vestibulum
enim, vitae pretium sem nisi vel nibh. Mauris congue rhoncus enim. </p>
<p>
Etiam leo nunc, rutrum vel, mattis quis, ultrices in, massa. Nullam sodales arcu quis
odio. Nullam malesuada pretium risus. Ut felis. Pellentesque eget lorem. Aenean
vehicula quam in libero. Proin sagittis felis eu tortor. Pellentesque vehicula justo eget
arcu. Aenean consequat, diam eget porttitor iaculis, purus ante mollis tellus, non sagittis
turpis odio nec erat. Morbi orci dui, mollis ac, pretium in, lobortis ac, lorem. Sed eros.
Donec risus eros, hendrerit at, tristique nec, rutrum sed, quam. Ut non urna. Ut
consequat aliquet mauris. Integer dolor pede, cursus nec, vulputate sit amet, vestibulum
nec, elit. Nullam lorem. Aenean euismod rhoncus nibh. </p>
<p>
Aliquam massa. Quisque mi. Pellentesque habitant morbi tristique senectus et netus et
malesuada fames ac turpis egestas. Morbi at enim. Aenean sodales accumsan urna.
Phasellus quis turpis molestie diam porta sodales. Cras sed turpis dignissim massa
accumsan eleifend. Nunc feugiat tortor eu erat. Nulla sit amet metus sit amet mauris
convallis convallis. Donec posuere erat et turpis. Duis velit mi, auctor eget, rhoncus nec,
varius a, lectus. Nulla id nibh. Donec aliquam fermentum risus. Nulla porta lorem non
est. </p>
<p>
Vivamus porttitor bibendum elit. Quisque euismod nibh vel mi. Maecenas ultricies felis
a est. Fusce rutrum cursus ipsum. Quisque nec neque. Vivamus gravida. Donec ante
risus, convallis quis, ullamcorper vitae, vestibulum nec, nunc. Vivamus tempor fringilla
elit. Vivamus iaculis, ipsum nec varius eleifend, nunc justo dictum ipsum, sed tempor
leo mauris vel nunc. Duis dignissim elit eu dui. </p>
<p>
```

Sed et arcu. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Quisque facilisis aliquet augue. Praesent ante urna, adipiscing vitae, malesuada eu, lacinia id, dolor. Nulla facilisi. Suspendisse tempor eros id erat. Pellentesque mauris elit, vehicula ac, condimentum ut, luctus a, justo. Mauris nisl nibh, pulvinar id, faucibus ac, tempus cursus, est. Morbi vel elit vitae magna lobortis luctus. Nullam tristique felis vel turpis. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut suscipit purus eu eros. Aenean dignissim. Fusce porttitor accumsan mauris.

Aliquam nec diam. Nam non lectus auctor justo consectetuer elementum. Sed sapien lacus, vulputate ut, viverra sed, accumsan id, turpis. Nullam dignissim. Fusce quis pede. Nam id odio. Quisque auctor feugiat nisl. Praesent pulvinar scelerisque urna. Phasellus sagittis auctor mauris. Morbi mauris sapien, cursus vitae, sollicitudin sit amet, fermentum eget, dolor. Ut eu ante venenatis ante malesuada mattis. Donec nec leo. Vivamus luctus metus eu pede. Maecenas cursus eleifend mauris. Ut erat velit, egestas id, auctor quis, faucibus at, turpis. Curabitur tempus ligula. Proin ipsum dolor, sollicitudin non, hendrerit ut, condimentum ac, nunc. Nunc ut justo in elit consequat gravida.

In scelerisque velit nec odio viverra sagittis. In semper ligula at purus. In metus. Phasellus placerat vulputate nunc. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Fusce pellentesque, orci a pharetra porta, magna lacus bibendum libero, nec laoreet massa metus et diam. Pellentesque in libero. Donec justo. Aenean quis pede. Morbi in magna. Integer augue mauris, adipiscing vel, gravida eu, vulputate eu, felis. Aliquam erat volutpat. Praesent eget risus ut nunc tempus molestie. Aenean sapien justo, consequat et, bibendum quis, laoreet sit amet, eros. Vestibulum et turpis. Nulla dolor eros, lobortis eu, dignissim quis, fermentum vitae, erat.

Pellentesque et nisi in arcu adipiscing hendrerit. Aenean sem tellus, facilisis sed, laoreet a, venenatis eu, diam. Etiam eleifend pede eget augue gravida rutrum. Nunc bibendum tincidunt nisi. Nullam sapien risus, blandit in, tristique ac, faucibus sed, risus. Mauris consequat, orci eget adipiscing porttitor, nisl elit tincidunt odio, aliquet mattis felis tortor pharetra mauris. Suspendisse aliquam rhoncus sem. Maecenas accumsan nulla non orci. Fusce quis nulla. Nunc sit amet lacus vel magna vehicula pulvinar. Proin eros nibh, mattis vitae, sollicitudin a, pellentesque a, lacus. In hac habitasse platea dictumst. Curabitur egestas scelerisque dui. Integer lobortis tincidunt purus. Integer quam tortor, accumsan vitae, malesuada eu, interdum vitae, orci. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Pellentesque urna metus, mattis et, elementum non, aliquet sed, orci. Cras quis neque sed purus commodo convallis.

Suspendisse eget est. Ut ultrices vestibulum nulla. Mauris tortor dui, nonummy in, lobortis non, pretium in, lectus. Etiam placerat vehicula urna. Fusce augue sapien, posuere eu, euismod sit amet, eleifend vitae, tellus. Cras lectus orci, sagittis in, facilisis nec, hendrerit nec, sapien. Etiam suscipit. Cras non justo pulvinar nisi consequat congue. Donec orci sem, congue accumsan, sagittis ut, pharetra a, sapien. Proin eu ipsum.

```
                                        <div id="up_cont">
                                            <a href="#header" title="Nahoru"
id="up">^^^ NAHORU ^^^</a>
                                        </div>
                                    </div>
                                </div>
                                <div class="clear"></div>
                            </div>
                            <div id="footer"><p id="footer_text">Created by - Milan Adam
&copy;</p></div>
                            <div class="clear"></div>
                        </div>
                </center>
</body>
</html>
```

CSS:

```
/* Nastaveni celeho tela stranky (fixed - pozadi se nehybe pri rolovani)*/
body {
        background:#F7F2E8 url(../obr/background_body.jpg) repeat-x fixed;
        margin:0;
        padding:0;
        font-family:Arial, sans-serif;
        font-size:11px;
        text-align:center;
}
/* Vycentrovana cast stranky - slouzi jako kontejner pro vlastni obsah stranky */
#site {
        background:#F5F2E9;
        width:700px;
        _height: 100%;
        min-height: 100%;
        position:relative;
        margin:0;
        padding:0;
        text-align:center;
}

/* Fotomontaz pocitacu v hlavicce stranky */
#header {
        background:#F5F2E9 url(../obr/background_header.jpg) no-repeat;
        margin:0 3px 0 3px;
        padding:0;
        width:694px;
        height:196px;
}
```

```
/* Div slouzici jako kontejner pro pomocny div, ve kterem je situovano menu a
pomocna cast na levo od menu */
#menu_container {
	width:700px;
	height:50px;
	padding:0;
	margin:0;
	text-align:left;
}

/* Pomocny div */
#menu_container_inner {
	width:694px;
	height:50px;
	padding:0;
	margin:0 3px 0 3px;
}

/* Pomocny div na levo od menu (kvuli vystupu pozadi hlavicky do teto roviny) */
#menu_left {
	background:#F5F2E9 url(../obr/background_menu_left.jpg) no-repeat;
	margin:0;
	padding:0;
	width:150px;
	height:50px;
	float:left;
}

/* Nastaveni divu pro menu */
#menu_right {
	background:#F5F2E9 url(../obr/background_menu_right.jpg) no-repeat;
	margin:0;
	padding:0;
	width:544px;
	height:50px;
	float:left;
}

/* MNEU - horizontalni */

.menu_item_first {
	width:110px;
	height:38px;
	margin:12px 0 0 60px;
	_margin:12px 0 0 30px;
	padding:0;
	float:left;
```

```
	text-align:center;
}

.menu_item {
	width:110px;
	height:38px;
	margin:12px 0 0 0;
	padding:0;
	float:left;
	text-align:center;
}

a.menu_a {
	letter-spacing:0.08em;
	text-decoration:none;
	width:128px;
	height:38px;
	font-size:13px;
	color:#2D2D2D;
	font-weight:bold;
	padding:0 0 0 5px;
}
a.menu_a:hover {
	font-weight:bolder;
	text-decoration:underline;
}

/* KONEC Mh */

/* Kontejner pro hlavni cast s obsahem */

#window {
	width:694px;
	clear:both;
	padding:0;
	margin:0 3px 3px 3px;
	background:#F5F2E9 url(../obr/background_cont_left.jpg) repeat-y;
}

/* Kontejner pro vertikalni menu */

#cont_left {
	width:177px;
	height:477px;
	margin:0;
	padding:0;
	background:#F5F2E9 url(../obr/background_cont_left.jpg) repeat-y;
```

```
	float:left;
	text-align:left;
}

/* Pravy kontejner / OBSAH */

#cont_right {
	width:512px;
	margin:0 3px 0 0;
	padding:0;
	float:left;
}

/* Konetejenr pro text obsahu */

#text_cont {
	margin:20px 15px 20px 15px;
	padding:0;
	text-align:justify;
	font-size:11px;
}

/* MENU - vertikalni */

#cont_left a {
	display:block;
	width:144px;
	padding:4px 18px 4px 15px;
	background:none;
	border-bottom:1px solid #F5F2E9;
	text-decoration:none;
	color:#2D2D2D;
	font-size:13px;
}
#cont_left a:hover {
	background:#F5F2E9;
	padding:3px 20px 5px 13px;
}

/* KONEC Mv */

/* Pomocny div pro natahnuti materskeho divu (pro NN) */

.clear {
	clear: both;
	height: 1px;
	overflow: hidden;
```

```
}

/* Kontejner pro odkaz na vrchol stranky */

#up_cont {
	text-align:center;
	margin:15px 0 0 0;
	padding:0;
}

/* Odkaz na vrchol stranky */

a#up {
	text-decoration:underline;
	color:#333333;
	font-weight:bold;
}

a#up:hover {
	text-decoration:overline;
	color:#000000;
}

/* Nastaveni paticky stranky */

#footer {
	position:absolute;
	clear:both;
	width:694px;
	height:25px;
	margin:3px 3px 0 3px;
	_margin:0 3px 0 3px;
	background:#EAE3D0;
}

/* Text v paticce */

#footer_text {
	margin:5px 0 0 0;
	padding:0;
	color:#939393;
	font-size:11px;
}

/* Text obsahu */

p {
```

```
        text-indent:30px;
        font-size:11px;
        color:#333333;
}
```

Příloha č. 3 - Slovník pojmů

Tag – do češtiny přeloženo jako značka. V terminologii jazyka HTM znamená označení pro jeden element dokumentu.
Element – prvek, na který je možné rozložit dokument.
Div – je bloková značka jazyka HTM. Bloková značka znamená, že za tímto elementem je automaticky vložen znak pro zalomení řádku. Z toho vyplývá, že bez dalších pomocných zásahů může být na jedné řádce pouze jeden blokový element.
HTML – HyperText Markup Language – zkratka pro jazyk definující text (obsah dokumentu) pomocí tzv. značek. Hypertext je anglický výraz pro tzv. hypertextové odkazy.
XHTML – vychází z jazyka HTM a zároveň z XML. Použití je shodné s HTML.
CSS – Cascading Style Sheets – anglický výraz pro kaskádové styly, které mají za úkol definovat formatovani dokumentu.
Child – anglický výraz pro potomka, tedy v terminologii HTML značící element obsažený v jiném elementu. Znamená to tedy například, že pro element <HTML> jsou všechny ostatní elementy child elementy.
Parent – anglický výraz pro rodiče, tedy v terminologii HTML značící element obsahující jiné elementy. Například pro element <BODY> je parent elementem pouze element <HTML>.
Font – anglický výraz pro písmo.
Hack – anglický termín pro slovo rozseknout. V počítačové terminologii znamená tento termín nestandardní postup k překonání problému, který by mohl být klasickým způsobem neřešitelný, popřípadě řešitelný pouze z části..
URL – Uniform Resource Locator – neboli jednotný lokátor zdrojů. Slouží ke specifikaci umístění zdrojů.
Javascript – skriptovací jazyk vycházející z objektově orientovaného programovacího jazyku Java. Je uzpůsoben pro použití ve webových prohlížečích. Díky tomuto jazyku je možné zakomponovat do HTML dynamické prvky, jakými může být například kontrola vstupních údajů ve formulářích.

Atribut – neboli vlastnost, značící v terminologii CSS funkci jednotlivého formátování.

Doctype – jedná se o nastavení typu (struktury) dokumentu XML popřípadě SGML (HTML), která je pomocí klíčového slova !ELEMENT uvedena v souboru *.dtd.