GRADA

Jiří Kosek

PRŮVODCE

HTML

TVORBA DOKONALÝCH WWW STRÁNEK

Podrobný průvodce

- srozumitelný návod na vytváření stránek v češtině
- kompletní popis HTML 3.2 a kaskádových stylů
- automatické generování dokumentů a vytváření formulářů
- nejnovější návrh jazyka HTML 4.0, úvod do dynamického HTML
- vkládání multimediálních objektů do stránek


GRADA

Jiří Kosek

HTML

TVORBA DOKONALÝCH WWW STRÁNEK

Podrobný průvodce

GRADA Publishing 1998

Jiří Kosek

HTML
tvorba dokonalých WWW stránek
podrobný průvodce

Sazba knihy byla připravena v typografickém systému TeX z písma Computer Modern ve variantě CS-font.

ISBN 80-7169-608-0

Stručný obsah

Podrobný obsah

Předmluva

Vážení čtenáři,

kniha, kterou právě držíte v ruce, by vám měla být neúnavným a nevyčerpatelným společníkem v říši tvorby webovských stránek. Kromě kompletního popisu jazyka HTML v ní naleznete i všechny další potřebné informace a mnoho praktických rad a postřehů.

Kniha je uspořádána tak, že již po přečtení prvních několika desítek stran budete schopni vytvářet první kvalitní stránky. Po nezbytném úvodu, který vás provede historií jazyka HTML, a po seznámení s adresami používanými ve Webu totiž následuje kapitola popisující základy jazyka HTML.

Další kapitoly jdou již více do hloubky a seznámí vás i s těmi nejskrytějšími zákoutími tvorby stránek: s tvorbou a vkládáním obrázků a s formátováním. Následuje velice důležitá kapitola, která popisuje způsoby zpřístupnění vašich stránek světu. A to jak z hlediska technologického, tak i „marketingového“ — dozvíte se, co udělat, aby se o vašich stránkách dozvěděli ostatní uživatelé Internetu.

Následuje kapitola popisující vytváření tabulek. Kromě klasického využití je popsáno, jak využít tabulky při vývoji atraktivních stránek s náročným grafickým designem.

Další kapitola o dynamicky generovaných dokumentech je malým výletem do světa programování. Vysvětlíme si principy automatického generovaní dokumentů, CGI-skriptů a protokolu HTTP. Na konci této náročnější kapitoly naleznete oddechovou část popisující zařazení počítadla přístupů na stránku. S touto kapitolou úzce souvisí i další kapitola o vytváření formulářů.

Desátá kapitola je důležitá zejména pro čtenáře vašich stránek. Obsahuje mnoho rad, jak vytvářet stránky, které se příjemně a snadno čtou.

Tématu naší mateřštiny na Webu je věnována celá kapitola. Web původně podporoval pouze anglický jazyk a tak i dnes výroba stránek v češtině přináší jistá úskalí.

Následující kapitoly popisují technologie, které jsou opravdu nové — většina z nich je dnes ve věku batolete. Dozvíte se, jak řídit vzhled stránek pomocí kaskádových stylů, vkládat do stránek aplety v Javě či rozdělit okno prohlížeče na několik částí pomocí rámů. Podíváme se i na to, co přináší nejnovější návrh jazyka HTML 4.0. Vědce a inženýry potěší část věnovaná vkládání matematických a chemických vzorců. Kapitola 14. je malým úvodem do dynamického HTML, které vám umožní vytvářet vysoce interaktivní stránky.

Kniha se zabývá jazykem HTML a ne ovládáním HTML-editorů. Předposlední kapitola nás však seznámí s několika programy, které vám po zvládnutí HTML mohou usnadnit a urychlit práci.

P

Konečně poslední kapitola zasazuje HTML do širšího kontextu a popisuje jazyk SGML, z kterého HTML vychází.

Pro čtení knížky nepotřebujete žádné speciální znalosti, stačí, když umíte uživatelsky pracovat s Webem pomocí nějakého prohlížeče. Pokud ne, můžete potřebné znalosti a dovednosti získat například v knize „Internet — první kroky českého uživatele“ [12].

Na adrese `http://manes.vse.cz/~xkosj06/html/` je domovská stránka knížky. Kromě jiného tam naleznete všechny ukázky stránek uvedené v knize a informace o novinkách, které se týkají publikování na Webu. Na e-mailové adrese `xkosj06@vse.cz` přivítám vaše připomínky a dotazy.

Příjemné a nerušené čtení Vám přeje

Jirka Kosek

Praha – Podolí, 30. září 1997

Typografické konvence

Aby byl text knihy pro čtenáře srozumitelnější, používám různé typografické konvence.

- Pro zápis elementů, atributů, výpisů stránek a programů používám `neproporcionální písmo`.
- Názvy programů jsou od ostatního textu odlišeny použitím *kurzívy*. Kromě toho kurzívu používám pro zvýraznění nových pojmů v textu.
- V případě potřeby používám uzavření obecného pojmu do francouzských uvozovek. Tento pojem se pak v praxi vždy nahradí nějakou konkrétní hodnotou. (Např. *«soubor»* se nahradí konkrétním jménem souboru.)

Kniha obsahuje mnoho ukázek zápisu stránek v jazyce HTML. Za většinou těchto ukázek ihned následuje výsledek jejich zobrazení v prohlížeči. Větší obrázky a tabulky jsou umísťovány nezávisle a v textu je na ně uváděn odkaz.

Některé úseky textu jsou označeny piktogramy. Jejich význam je následující:

Takto označený text obsahuje důležitou informaci, jejíž neznalost vám může zkomplikovat život.

Text obsahuje informaci, jejíž znalost vám může život usnadnit. Většinou zde naleznete různé tipy a triky, jak vylepšit vaše stránky a zefektivnit jejich tvorbu.

Informace uvedené v takto označeném textu jsou zajímavé, ale jejich neznalost negativně neovlivní vaše základní životní funkce.

1. Úvod

Na tomto místě by se slušelo v několika odstavcích opěvovat dnešní význam Internetu pro společnost a zdůraznit, že o masové rozšíření Internetu se zasloužila služba World-Wide Web. Tímto námětem, jehož podstata je dnes známá snad každému, již byly popsány tuny papíru. Uděláme proto něco pro naše lesy a tento obligátní úvod vynecháme. Je přece nad slunce jasné, že pokud si koupíme knížku o tom, jak vytvářet WWW-stránky, máme jistou představu o tom, co je to Internet i Web. Tato představa je o to cennější, že byla získána praktickými zkušenosti a ne četbou článků tu vyzdvihujících Internet do nebe a tu varujících před smrtelným nebezpečím, které hrozí (návody na výrobu drog, výbušnin atd.).

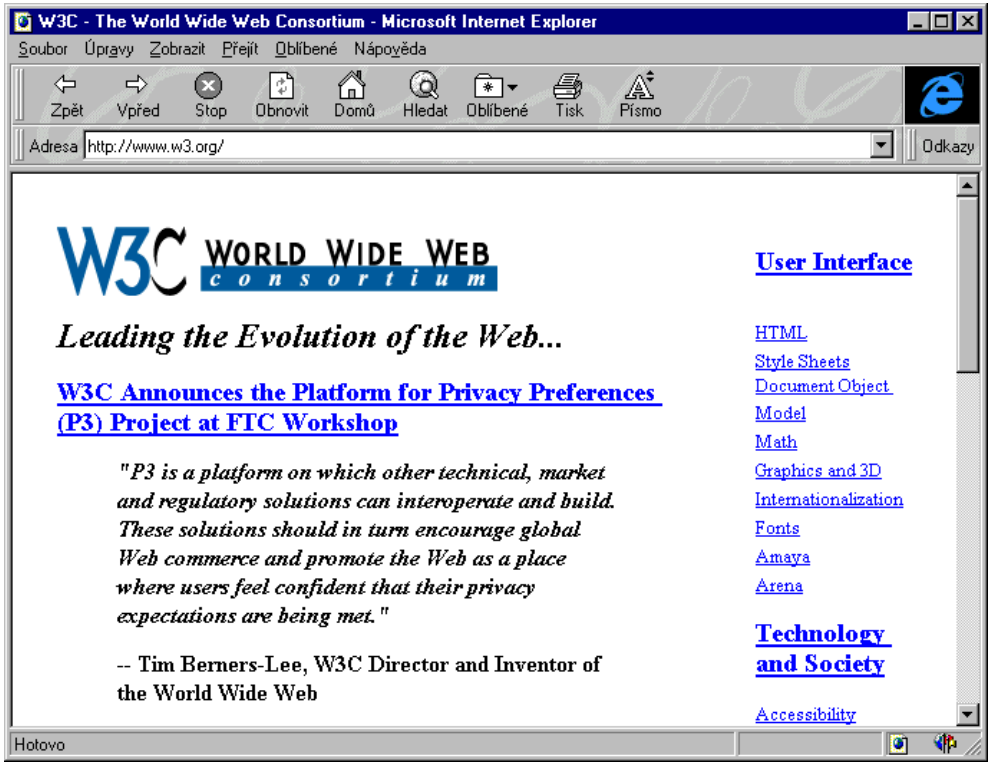
Každému je tedy jasné, jak vypadá WWW-stránka. Malá ukázka je na obrázku 1-1 na následující straně. Na tomto místě připomene, že název *WWW--stránka* může být trochu zavádějící. Ač hovoříme o stránce, může mít po vytištění na tiskárně jedna WWW-stránka několik listů papíru. Výraz *stránka* je však natolik vžitý, že se jej budeme držet i my.

Pro takovou stránku je typické, že obsahuje *odkazy* na další stránky, případně i na jiné zdroje Internetu. Odkazy vytvářejí velice složitou strukturu, protkávají stránky na celém světě tenkými nitkami odkazů. Z toho také vznikl název *World--Wide Web* — *web* je anglicky pavučina a *world-wide* můžeme přeložit jako celosvětový.

Forma organizace informací, kdy z jednoho místa vedou odkazy na další související místo, je v poslední době hodně populární a používá se pro ni název *hypertext*. Podobným způsobem je organizována i nápověda v mnoha programech a operačních systémech (naše oblíbené *Windows* nevyjímaje).

Když už jsme u těch pojmů — Webu samozřejmě přísluší i dnešní nejmódnější přívlastek — *multimediální*. Nikoho dnes již nepřekvapí, že WWW spojuje text, grafiku, animace, hudbu a bůhvíco ještě dohromady. Porovnáme-li to se stavem třeba jen před deseti lety, uvědomíme si, jaký obrovský skok za tu dobu technika udělala — a to jak technologický, tak cenový. To, co před pár lety mohla na počítačích dělat pouze NASA či bohatá filmová studia, může dnes na svém pécečku dělat téměř každý doma.

Zatím jsme se však nezmínili o tom, co je na Webu nejdůležitější. Co myslíte vy? (Pokud myslíte na `http://www.playboy.com`, jste vedle.) Velkou výhodou World-Wide Webu (a celého Internetu vůbec) je nezávislost na platformě. Co si pod tím představit? Zkrátka to, že tutéž stránku si můžeme prohlížet na počítači, kde běží *Windows*, stejně tak jako na (Power)Macintoshi s operačním systémem firmy Apple či na nějaké pracovní stanici, kde pro změnu běží *UNIX* s grafickou nadstavbou *X-Window*. O tom, jak je tato vlastnost důležitá, se

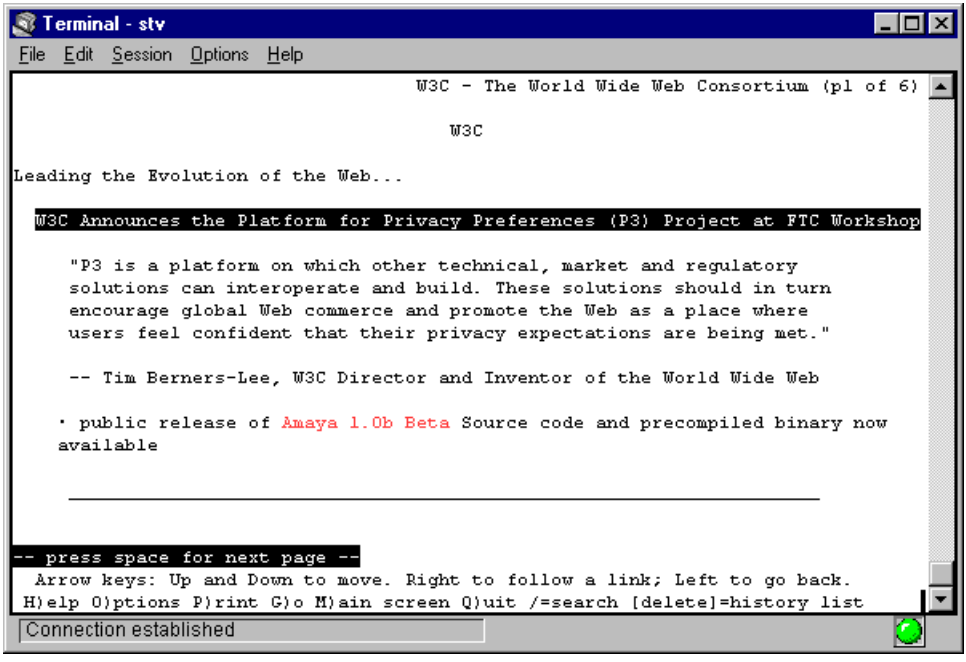


Obr. 1-1: WWW-stránka

přesvědčil každý, kdo se pokoušel přenášet např. nějaký textový soubor z Macintoshe na PC. Na obrázku 1-2 na následující straně si můžete prohlédnout, jak vypadá stránka v textovém prohlížeči Lynx pracujícím pod Unixem — jedná se přitom o zobrazení stejné stránky jako na obrázku 1-1.

Jak je této nezávislosti dosaženo? Celý Internet, Web nevyjímaje, je postaven na standardech (normách), které popisují způsoby komunikace v Internetu, formáty používaných datových struktur apod. Tyto standardy se přitom neváží na nějaké konkrétní vlastnosti daného operačního systému, takže je lze většinou bez větších potíží implementovat na všech používaných platformách. V oblasti Internetu jsou standardy popsány v tzv. *RFC-dokumentech*. Ty jsou v Čechách dostupné např. na adrese `ftp://ftp.vse.cz/pub/docs/rfc`. Odkazy na konkrétní RFC si uvedeme vždy, když budeme probírat odpovídající látku.

A jaké jsou hlavní pilíře Webu? Jedním z hlavních je jazyk *HTML (HyperText Markup Language)* — tento jazyk slouží k popisu toho, co WWW-stránka obsahuje. Pokud toužíte vidět, jak vypadá zápis některé stránky v HTML, vyberte

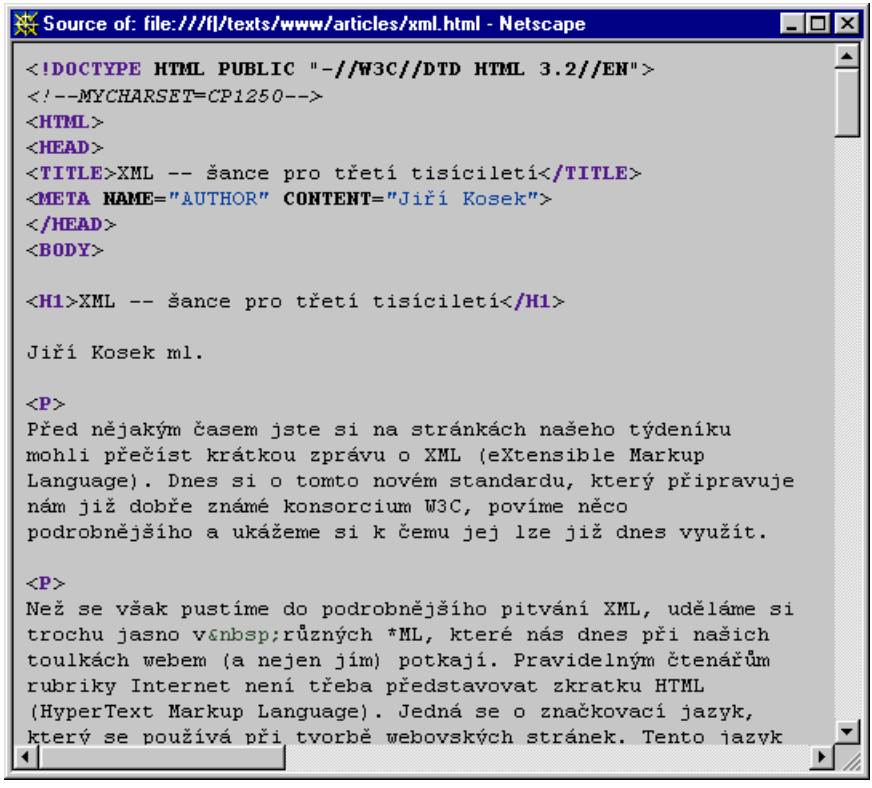


Obr. 1-2: WWW-stránka a její zobrazení v prohlížeči Lynx

v *Netscape* v menu příkaz View ▷ Document Source nebo v *Exploreru* Zobrazit ▷ ▷ Zdroj. V nově otevřeném okně se objeví zápis stránky v HTML (viz obr. 1-3 na následující straně). Jak jste možná vytušili, budeme se v naší knize zabývat ponejvíce jím. Konec této kapitoly je věnován popisu jistě zajímavé historie tohoto jazyka.

Řekli jsme si, že WWW-stránky, vytvářejí hypertextový informační systém. Struktura hypertextu je vytvářena odkazy mezi stránkami. Odkazy se zapisují rovněž ve standardizované podobě; pomocí tzv. *URL (Uniform Resource Locator)*. O URL — adresách používaných ve Webu — pojednává celá druhá kapitola.

Třetím základním kamenem je protokol *HTTP (HyperText Transfer Protocol)*. Ten zajišťuje přenos WWW-stránek ze serveru do počítače uživatele. Se základními rysy HTTP se seznámíme v osmé kapitole.



Obr. 1-3: Zápis stránky v HTML

1.1 Co potřebujeme pro tvorbu vlastních stránek

Když se ještě jednou pozorně zadíváte na obrázek 1-3, zjistíte, že v HTML se samotný text WWW-stránky doplní značkami, které určují jeho význam. WWW-stránka je tedy obyčejný textový soubor — takový, jaký lze vytvořit třeba v *Poznámkovém bloku* nebo v editoru v *Norton Commanderu*. Protože se stránka vytváří v editoru a zapisuje se v jazyce HTML, budeme jí někdy také říkat *HTML dokument*.

Pro naše první pokusy nám tedy vystačí *Poznámkový blok*, který je standardní součástí *Windows*. V anglických verzích jej nalezneme pod názvem *Notepad*. Pokud se budeme tvorbou WWW-stránek živit, asi si časem seženeme nějaký výkonnější editor, který nám umožní definovat si různá makra a vůbec bude nabízet mnohem větší možnosti pro editaci textových souborů. Takových programů se dnes dá sehnat mnoho. Od těch komerčně šířených, mezi něž patří např. *MultiEdit*, až po velice kvalitní sharewarové nebo freewarové produkty (např. *Programmer's File Editor*, *JED*). Výkonnější editor nám však nepřinese

možnost tvorby lepších WWW-stránek, pouze nám může práci trochu usnadnit a zpříjemnit.

Druhým programem, který budeme potřebovat, ale který již jistě všichni máme, je *prohlížeč*. Tím budeme kontrolovat, zda námi vytvářená stránka vypadá opravdu tak, jak jsme zamýšleli. Dnes na poli prohlížečů vedou dva programy — *Netscape Navigator* od firmy Netscape Communications a *Internet Explorer* od Microsoftu. Určitě už jeden z nich máme na počítači nainstalován.

Abyste mohli využít maximum vlastností HTML, které jsou v této knize prezentovány, měli byste si opatřit *Navigator* verze 3.0 nebo novější či *Explorer* rovněž verze 3.0. Starší verze programů nemusí podporovat všechny vlastnosti HTML, které si zde popíšeme. Jazyk HTML se totiž stále velice rychle vyvíjí.

Poslední, co budete k tvorbě stránek pro Web potřebovat, je znalost jazyka HTML. V této oblasti by vám měla pomoci právě tato kniha.

Pro své první pokusy s tvorbou stránek vůbec nepotřebujete být připojeni k Internetu. Stránky budete vytvářet na svém pevném disku, odkud je bude prohlížeč načítat. Nemusíte se tak bát, že vám rodiče nebo manželka zatrhnou tvorbu stránek kvůli velkému účtu za telefon.

Dnes se rovněž stále častěji objevují programy, které tvrdí, že s nimi můžeme vytvářet WWW-stránky bez znalosti HTML. Na první pohled vypadají podobně jako nějaký textový procesor (např. *MS Word*) a podobně se i ovládají, ale pracují se soubory ve formátu HTML. Vzhledem k rychlému vývoji HTML nám však tyto programy málokdy umožní využívat všech jeho aktuálních možností. Navíc mnohdy negenerují HTML, které odpovídá standardu.

Je proto vždy lepší znát HTML — člověk pak má reálnou představu o tom, co lze na Webu vytvořit a co ne. Navíc může výsledek vyprodukovaný některým z těchto *HTML-editorů* zkontrolovat a upravit k obrazu svému. Myslím si, že pro tyto editory nastane opravdu zlaté období teprve až se vývoj HTML zastaví. I přesto si v předposlední kapitole ukážeme, čím nám mohou tyto programy pomoci. V této kapitole se rovněž dozvíme, jaké jsou možnosti konvertování do HTML z jiných formátů běžně používaných na osobních počítačích (např. dokumentů z *Wordu*).

1.2 Historie a vývoj HTML

První definici jazyka HTML vytvořil v roce 1991 Tim Berners-Lee jako součást projektu WWW, který měl umožnit vědcům zabývajícím se fyzikou vysokých energií komunikaci a sdílení výsledků výzkumu po celém světě. Ne náhodou proto celý projekt vznikal v CERNu (Centre Européenne de Rechere Nucléaire, Evropské centrum jaderného výzkumu), který leží na švýcarsko-francouzských hranicích nedaleko Ženevy. Tato verze HTML je známá pod označením HTML 0.9. Umožňovala text rozčlenit do několika logických úrovní, použít několik druhů zvýraznění textu a zařadit do textu odkazy a obrázky.

Berners-Lee při návrhu HTML nepředpokládal, že by autoři WWW-stránek museli tento jazyk znát. První verze WWW-softwaru byla napsána pro operační systém *NextStep* a obsahovala jak prohlížeč, tak i integrovaný editor WWW-stránek. Když však Marc Anderssen se svými kolegy z NCSA (National Center for Supercomputing Applications) psal známý prohlížeč *Mosaic*, považoval za příliš obtížné implementovat do programu rovnou i editor HTML. Díky tomuto rozhodnutí a tomu, že ne každý provozuje na svém počítači *NextStep*, je dnes nutné, aby autoři profesionálních stránek znali HTML.

Požadavky uživatelů na WWW vzrůstaly, a tak producenti různých prohlížečů obohacovali HTML o některé nové prvky. Aby byla zachována kompatibilita mezi jednotlivými modifikacemi HTML, vytvořil Berners-Lee pod hlavičkou IETF (Internet Engineering Task Force) návrh standardu HTML 2.0, který zahrnoval všechny v té době běžně používané prvky HTML. Verze HTML 2.0 má zároveň dvě úrovně. První z nich (Level 1) pouze málo rozšiřuje předchozí verzi HTML. Level 2 navíc definuje práci s formuláři. Specifikaci HTML 2.0 nalezneme v RFC dokumentu číslo 1866.

Další rozšíření jazyka známá jako HTML+ zahrnují zejména rozšíření HTML o vytváření tabulek a matematických vzorců. Rovněž se zde objevují prvky, které umožňují precizněji kontrolovat výsledný vzhled textu — lepší obtékání obrázků textem a styly dokumentů. Dave Raggett z laboratoří Hawlett-Packard HTML+ formalizoval a vytvořil jeho deklaraci DTD (Document Type Declaration) v jazyce SGML — na jaře roku 1995 tak vznikl návrh standardu HTML 3.0.

✍ *Teď si možná řeknete co je to za knížku, když už na straně 20 se to hemží samými neznámými pojmy jako DTD a SGML. SGML je metajazyk, který slouží k definici jiných jazyků — tedy i HTML. Této definici se pak říká DTD. Podrobný výklad vztahu mezi SGML, DTD a HTML naleznete v poslední kapitole.*

Některé prvky HTML 3.0, jako např. tabulky, podporovaly novější verze prohlížečů *Mosaic* a *Netscape*. Kompletní podporu pro všechny rysy HTML 3.0 nabízel pouze experimentální prohlížeč *Arena*. Ten je bohužel k dispozici pouze pro operační systémy typu *Unix*.

Na počátku roku 1996 již bylo jasné, že HTML 3.0 bylo tak mohutným skokem vpřed, že se nenašel nikdo, kdo by dokázal implementovat prohlížeč s jeho podporou. Vývoj standardů Webu v té době již koordinovalo konsorcium W3C, jehož členy jsou mimo jiné přední softwarové firmy. Členové W3C se tedy shodli na vlastnostech, o které rozšíří HTML 2.0, a vytvořili tak HTML 3.2. HTML 3.2 však zdaleka neobsahuje vše z HTML 3.0. Z verze 3.0 zůstaly v podstatě jen okleštěné tabulky. Ostatní nové prvky HTML 3.2 jsou jen jakousi směskou, kterou v té době podporovaly nejnovější prohlížeče. Opakoval se tedy v podstatě stejný postup jako při vzniku verze 2.0 — jazyk se sjednotil na průniku možností těch nejrozšířenějších prohlížečů. Kromě tabulek přibyly ve verzi 3.2 zejména možnosti lepší kontroly formátování včetně mnohem volnějšího výběru použitých druhů písma — logický ústupek požadavkům na graficky perfektně vypadající stránky. Další podstatné rozšíření se týkalo podpory Java-apletů. Tato verze HTML nese kódové jméno Wilbur a od ledna 1997 je doporučením konsorcia W3C — znamená to, že by ji měli všichni používat, aby byla ve Webu zajištěna kompatibilita.

Většina dnešních prohlížečů však již nabízí další rozšíření nad rámec HTML 3.2. Jejich používaní je však dvousečné — na jednu stranu nejsou nijak standardizována a některé prohlížeče jim nemusí rozumět, na stranu druhou jsou mnohdy velmi užitečná a jejich používaní a rozšíření může urychlit jejich zařazení do standardu. Mezi takováto rozšíření patří především rámy (frames) a různé skriptové jazyky (např. JavaScript).

Posledním hitem, který s HTML úzce souvisí, jsou kaskádové styly dokumentů (CSS), které jsou doporučením W3C od prosince 1996. Zatím je však podporuje pouze *Internet Explorer*, *Netscape Navigator* a experimentální prohlížeč a HTML-editor *Amaya*, který je opět určen pro platformu *Unix*.[1]

Na jaře roku 1997 zvěřejnilo W3C další plány na rozšíření HTML pod kódovým názvem Cougar. Cougar v sobě zahrnuje HTML 3.2 společně s běžně používanými konstrukcemi jako jsou rámy, skripty a obecné vkládání objektů. Dalšími novinkami byla podpora vícejazyčných dokumentů. Na začátku července 1997 uveřejnilo W3C návrh HTML 4.0, který vznikl drobnými úpravami Cougaru a vytvořením jediného komplexního dokumentu popisujícího návrh standardu.

V naší knize nejprve podrobně probereme HTML 3.2. Pak se zaměříme na jeho další rozšíření (zejména z HTML 4.0), protože je možné, že v době, kdy čtete knihu, budou již součástí standardního HTML.

[1] Na přenosu *Amayi* do prostředí *Windows* se usilovně pracuje. K dispozici by měla být koncem roku 1997.


HTTP
TELNET
GOPHER
FILE
FTP

2. Adresy ve WWW — URL

Než se pustíme do hlubin jazyka HTML, seznámíme se s adresami používanými ve světě Webu a Internetu vůbec. Víme, že na světě je mnoho počítačů, které nabízejí různé služby — WWW, přenos pošty, FTP, diskusní skupiny a jistě mnoho dalších. Bylo by tedy záhodno, aby jeden druh adresy umožnil obsáhnout celou paletu služeb na všech možných serverech. Zároveň by jedna adresa měla ukazovat právě na jednu službu či zdroj informací na Internetu. Poslední požadavek, který je rovněž velice důležitý — adresa by měla být snadno čitelná jak pro člověka, tak pro počítač. Všechny tyto požadavky v sobě spojuje *URL (Uniform Resource Locator)*.

2.1 Tvar URL

Samotný pojem URL nám může připadat neznámý, ale s adresou zapsanou v tomto tvaru jsme se již jistě setkali. Většina z nás někdy četla internetové noviny Neviditelný pes. Možná si ještě vzpomenete, že adresa Neviditelného psa je `http://pes.eunet.cz`. A právě tato adresa není ničím jiným než URL.

Pokusíme-li se toto URL dešifrovat, podle prvních pár písmen zjistíme, že ukazuje na nějaký zdroj, který je přístupný pomocí protokolu HTTP — půjde tedy o webovskou stránku. Za dvěma lomítky následuje adresa příslušného WWW-serveru.

Zatím vše vypadá velice jednoduše a možná se divíte, proč je tématu URL věnována celá kapitola. Odpověď je vcelku jednoduchá — výše mého autorského honoráře závisí na počtu kapitol;-) Ne, tak to vážně není. Pravdou je, že obecný tvar URL je poněkud složitější:

«schéma»`://`*«uživatel»*`:`*«heslo»*`@`*«počítač»*`:`*«port»*`/`*«cesta»*`;`*«parametry»*
`?`*«dotaz»*`#`*«fragment»*

Samozřejmě, že vše by mělo být na jednom řádku za sebou, ale to by se nám vzhledem k formátu knihy na stránku nevešlo.

První částí URL je *«schéma»*. To určuje typ zdroje, na který adresa ukazuje. Nejčastěji se dnes setkáme asi se schématem `http`, které identifikuje webovské stránky. Přehled dalších schémat uvádí tabulka 2-1 na následující straně. Tento přehled samozřejmě není konečný — s přibýváním nových služeb budou přibývat i nová schémata.

Schéma	Popis
`file`	soubor na lokálním disku
`ftp`	soubor přístupný přes službu FTP (File Transfer Protocol)
`gopher`	Gopher protcol
`http`	webovské stránky (HyperText Transfer Protocol)
`mailto`	adresa elektronické pošty
`news`	diskusní skupiny
`nntp`	diskusní skupina na určitém serveru
`prospero`	adresářová služba Prospero
`telnet`	terminálový přístup ke vzdálenému počítači
`wais`	Wide Area Information Server

Tab. 2-1: Přehled nejpoužívanějších schémat

Většina z dalších částí URL je nepovinná, my si je však pro úplnost popíšeme všechny:

«uživatel» Udává jméno, pod kterým se přihlašujeme k nějaké službě (např. k telnetu nebo k neanonymnímu FTP).

«heslo» Udává heslo, kterým se *«uživatel»* přihlašuje ke službě. Heslo se od uživatelského jména odděluje dvojtečkou.

Tímto způsobem by však měla být zadávána pouze veřejná hesla. Pokud takto uvedeme soukromé heslo, může nám ho někdo přečíst přes rameno. Heslo rovněž není šifrováno během přenosu po Internetu a pro šikovného hackera není problém heslo zjistit.

«počítač» Udává adresu počítače. Ta může být zadána buď jako doménová (`manes.vse.cz`) nebo IP (146.102.56.1). Tato část se vyskytuje téměř ve všech URL. Pokud je pro přístup k počítači uvedeno i uživatelské jméno, odděluje se od adresy počítače zavináčem (`@`).

«port» Udává číslo portu, na kterém je služba dostupná. Většina schémat má implicitní číslo portu a není ho potřeba uvádět. V případě, že

služba pracuje na nestandardním portu, odděluje se číslo portu od adresy počítače dvojtečkou.

«cesta» Obvykle určuje cestu a jméno souboru, který je požadován. Od předchozí části URL se odděluje lomítkem (`/`), které není součástí cesty.

«parametry» Za středníkem je možno uvést parametry. Tuto část URL využívají pouze některá schémata.

«dotaz» Za otazníkem může být zadání dotazu. Tato část se opět používá jen někdy. Nejčastější použití je asi při předávání dotazů různým vyhledávacím službám v Internetu.

«fragment» Za mříží[1] (`#`) můžeme uvést jméno tzv. fragmentu. Fragment se používá v případech, kdy chceme odkázat na určitou část zdroje určeného pomocí předchozí části URL (např. na určitou sekci webovské stránky).

✍ *Fragment ve skutečnosti není součástí definice URL v [5], ale jedná se o rozšíření zavedené až v HTML. Prakticky je však fragment součástí URL a proto jsme jej zahrnuli do popisu URL. V definici URL rovněž nenalezneme definici parametrů a dotazu odděleně od cesty — vše se zde skrývá pod cestou.*

Jak jsme si ukázali, URL obsahuje různé rezervované znaky '`/`', '`:`', '`@`', '`;`', '`?`' a '`#`'. Mezi další takové znaky patří ještě '`&`', '`=`', '`%`', '`+`', '`{`', '`}`', '`|`', '`\`', '`^`', '`~`', '`[`' a '`]`'. Tyto znaky bychom měli používat pouze k vyhrazeným účelům. Pokud má být součástí URL některý z těchto znaků, musíme ho přepsat do tvaru `%`*«xx»*, kde *«xx»* je hexadecimální kód znaku. Tímto způsobem by se měly přepisovat i znaky s kódem vyšším než 127 — tedy i české znaky s diakritickými znaménky.

Ostatní znaky (písmena anglické abecedy, číslice a znaky '`$`', '`-`', '`_`', '`.`', '`!`', '`*`', '`'`', '`(`', '`)`', '`,`', '`"`') mohou být používány bez omezení. Pokud by URL mělo obsahovat mezery, nahradí se znakem plus '`+`'.

Pokud bychom tedy do URL chtěli dát text: „`Einstein pravil: E=mc^2`“, museli bychom text překódovat následovně: `Einstein+pravil:+E%3Dmc%5E2`.

[1] Někdy se pro původní anglický výraz hash-mark používá i další překlad — kriminál.

2.2 Schémata

V této části si ukážeme, jak vypadají URL pro nejčastěji používaná schémata. Probereme je v pořadí, které odpovídá jejich používání v dnešní době.

Pokud vás to po schématu `http` nebo `mailto` přestane bavit, klidně přeskočte až na další sekci. Služby odpovídající ostatním schématům se totiž nepoužívají již tak masově jako Web a elektronická pošta.

Schéma HTTP

Toto schéma patří mezi nejpoužívanější a jeho obecný tvar je:

`http://`*«počítač»*`:`*«port»*`/`*«cesta»*`?`*«dotaz»*`#`*«fragment»*

Počítač je adresa webovského serveru, který danou stránku obsahuje. Port uvádíme pouze v případech, kdy se liší od standardní hodnoty 80. Cesta obsahuje adresář a jméno souboru se stránkou. Soubory obsahující stránky mají nejčastěji příponu `.html`. Od té doby, co se do Internetu vložil Bill a jeho [kampəni], rozrůstá se počet dokumentů, které mají jen třípísmennou koncovku `.htm`.

Část za otazníkem je nepovinná a v dřevních dobách služby World-Wide Web se používala pro zadání hledaného textu. Dnes je použití širší a tímto způsobem se velmi často předávají parametry programům, které generují WWW-stránky dynamicky.

Fragment se používá v případech, kdy se chceme odkázat na určitou část stránky — třeba jen na jeden obrázek. URL pak může vypadat třeba takto: `http://www.kdesi.cz/pub/doc/grafika.html#obr1`.

Při zápisu URL musíme dát pozor na velikost písmen. Servery pracují většinou pod operačními systémy *UNIX* a *Windows NT*, které rozlišují velikost písmen ve jménech souborů a adresářů. Toto omezení se však netýká např. jména počítače, které je na velikosti písmen nezávislé.

Představte si, že máme následující URL `http://www.vse.cz`*. URL obsahuje pouze schéma a adresu serveru. Jak ale server pozná, kterou stránku po něm požadujeme? Většina serverů je nakonfigurována tak, že v případě absence jména souboru se stránkou doplní nějaké standardní jméno. Nejčastěji jím bývá* `index.html`*,* `default.html` *či* `welcome.html`*. Zcela stejný mechanismus pracuje i v případech, kdy je součástí URL cesta, ale neobsahuje na svém konci jméno souboru.*

Schéma mailto

Na rozdíl od ostatních schémat neslouží `mailto` k určení konkrétního informačního zdroje na síti. Jeho účel spočívá v možnosti zapsat adresu elektronické pošty, na kterou lze poslat dopis. Tvar je nejjednodušší ze všech schémat:

`mailto:`*«e-mailová adresa»*

«e-mailová adresa» je obyčejná adresa elektronické pošty.[2] Moje e-mailová adresa zapsaná jako URL by tedy byla `mailto:xkosj06@vse.cz`.

Některé e-mailové adresy obsahují znak procento, který se používá v případech, kdy potřebujeme dopis poslat do nějaké jiné sítě přes gateway (tj. mimo Internet). V tomto případě musíme znak procento do URL zakódovat jako `%25`.

Schéma FTP

Služba FTP slouží k přenosu souborů mezi počítači. Většinou se však používá ke stahování souborů ze vzdálených serverů. URL se schématem `ftp` slouží k určení souborů na vzdálených serverech. Služba FTP standardně sídlí na portu 21 a pokud požadujeme jiný port, musíme ho uvést za dvojtečkou po adrese serveru.

Soubory přístupné pomocí FTP můžeme rozdělit do dvou velkých skupin: (1) soubory přístupné komukoliv — tzv. anonymní FTP a (2) soubory přístupné pouze konkrétnímu uživateli po zadání hesla.

Pokud chceme vytvořit URL, které ukazuje na anonymní FTP-zdroj, nemusíme uvádět uživatelské jméno. Klientský program se spojí s FTP serverem a přihlásí se jako uživatel `anonymous` a jako heslo pošle naši e-mailovou adresu.

Formát URL pro schéma FTP je následující:

`ftp://`*«uživatel»*`@`*«počítač»*`:`*«port»*`/`*«cesta»*`;type=`*«typ»*

Při použití anonymního FTP přitom vynecháme část *«uživatel»*`@` a uvádění portu také většinou není potřeba. *«typ»* určuje způsob, jakým se má soubor stáhnout (`d` – adresář, `a` – přenos textového souboru a `i` – přenos binárního souboru). Tento typ se však většinou neuvádí a je plně na bedrech klientského programu, aby zvolil správnou metodu.

«cesta» se skládá ze jmen podadresářů a ze jména souboru, na který URL ukazuje (*«dir1»*`/`*«dir2»*`/.../`*«dirN»*`/`*«jméno»*). Pokud cesta neobsahuje jméno souboru, zobrazí klient obvykle seznam všech souborů v daném adresáři.

[2] Přesněji — adresa odpovídající formátu popsanému v RFC 822.

✍ *Někdy může být důležité uvědomit si, že adresář «dir1» se chápe jako relativní. URL* `ftp://xkosj06@manes.vse.cz/public_html/index.html` *ukazuje na adresář* `public_html`*, který je v mém domovském adresáři, a v něm na soubor* `index.html`*. Kdybych však chtěl vytvořit URL ukazující na soubor v nějakém adresáři odvíjejícím se od kořenového adresáře (např.* `/etc/passwd`*), musel bych před udání adresáře v URL uvést lomítko. To se však musí zakódovat jako* `%2F` *(hexadecimální hodnota kódu lomítka je 2F). Kdyby si tedy správce systému chtěl stáhnout soubor se jmény uživatelů, mohl by použít URL* `ftp://root@manes.vse.cz/%2Fetc/passwd`*. (Druhou věcí je, že dobře zabezpečený server by takovýto požadavek měl ihned zkraje odmítnout.)*

2

Na závěr ukázka jednoho zcela typického URL, na kterém se nalézá právě popis standardu URL: `ftp://ftp.vse.cz/pub/docs/rfc/rfc1738.txt`.

Schéma news

Toto schéma se používá pro odkazy na diskusní skupinu nebo na určitý článek v diskusní skupině. URL může nabývat jeden ze tří základních tvarů:

`news:*`
`news:`*«diskusní skupina»*
`news:`*«id»*`@`*«počítač»*

První zápis se používá jako odkaz na všechny dostupné diskusní skupiny. Druhý způsob odkazuje na jednu určitou diskusní skupinu — např. `comp.text`. Poslední způsob ukazuje na konkrétní příspěvek. V tomto případě je potřeba uvést jednoznačný identifikátor příspěvku (*«id»*) a doménovou adresu počítače, odkud byl příspěvek do skupiny zaslán (např. `news:4123@ucbvax.berkeley.edu`).

Schéma NNTP

Protokol NNTP (Net News Transfer Protocol) slouží rovněž k přenosu příspěvků z diskusních skupin. Na rozdíl od předchozího schématu se však odkazujeme na příspěvek na určitém konkrétním NNTP-serveru. Tvar URL je následující:

`nntp://`*«počítač»*`:`*«port»*`/`*«diskusní skupina»*`/`*«číslo příspěvku»*

V případě vynechání čísla portu se předpokládá standardní číslo 119. Typické URL pak vypadá např. takto: `nntp://vse470.vse.cz/vse.pocitacova-sit/4478333`.

☺ Většina NNTP-serverů je nakonfigurována tak, že spolupracuje pouze s lokálními klienty (ve smyslu počítačů připojených do lokální sítě jedné instituce). Příspěvek diskusní skupiny určený pomocí schématu `nntp` pak není přístupný z celého světa. Je proto lepší využívat schéma `news`.

Schéma telnet

Služba telnet slouží pro přihlášení ke vzdálenému počítači. URL pro schéma `telnet` má tvar:

`telnet://`*«uživatel»*`:`*«heslo»*`@`*«počítač»*`:`*«port»*`/`

Většinou se však udává pouze adresa počítače. Standardní číslo portu je 23. Udání uživatelského jména a hesla je většinou zbytečné, protože po připojení si většina vzdálených počítačů vyžádá přihlášení již v režimu práce na vzdáleném terminálu. Jejich uvedení má pouze informativní význam pro uživatele.

Schéma gopher

Schéma `gopher` se používá pro identifikaci zdrojů dostupných pomocí služby Gopher. URL má tvar:

`gopher://`*«počítač»*`:`*«port»*`/`*«cesta»*

«cesta» přitom může mít jeden z následujících tří tvarů:

«typ»«selektor»
«typ»«selektor»`%09`*«dotaz»*
«typ»«selektor»`%09`*«dotaz»*`%09`*«gopher+ text»*

V URL se vyskytuje „magický“ znak `%09`, který reprezentuje tabulátor. Standardním portem služby gopher je 70. *«typ»* je jednociferné číslo, které udává typ informace (viz tabulka 2-2).

Číslo typu	Druh informace
0	textový soubor
1	adresář
2	CSO-server pro vyhledávání telefonních čísel
3	chyba
4	BinHex soubor pro počítač Macintosh
5	binární archiv pro MS-DOS
6	soubor zakódovaný metodou uuencode
7	vyhledávací server
8	relace telnetu
9	binární soubor

Tab. 2-2: Typy informací v gopheru

«selektor» určuje informaci, o kterou máme na serveru zájem. Jeho syntaxe vychází z pravidel zavedených službou gopher a je zajímavé, že na prvním místě je opět uveden typ informace. V URL se pak typ informace zdvojí. Např. seznam gopher-serverů v Čechách naleznete na URL: `gopher://gopher.cesnet.cz/11/.gopherinfo/cz-gophers`. V případě prázdného selektoru je odpovědí serveru hlavní menu.

Ve spojení s typem 7 můžeme za selektorem uvést *«dotaz»*, jehož výsledek se nám vrátí jako odpověď. Služba gopher byla vylepšena do služby gopher+. Jejích možností je možno využít použitím poslední části URL *«gopher+ text»*. Pro gopher+ však musí být v URL vždy přítomen selektor a dotaz (i kdyby měl být prázdný).

Schéma file

Toto schéma se z koncepce URL trošku vymyká. Primárně totiž slouží ke specifikaci souborů na lokálním disku počítače a neurčuje tedy jedinečně nějaký zdroj informací v celém Internetu. V URL se tedy vynechává část specifikující adresu počítače. URL proto obsahuje poněkud netypicky tři lomítka za sebou:

`file:///`*«cesta»*

Formát *«cesty»* závisí na použitém operačním systému. Ve Windows, která používají nešťastný způsob, kde má každý disk své písmenko, může URL vedoucí k souboru `index.html` v adresáři `www` na disku `f:` vypadat takto: `file:///f:/www/index.html`.

V případech, kdy uvedeme v URL adresu počítače, se ke stažení souboru obvykle použije protokol FTP.

2.3 Relativní URL

To, že každé URL jednoznačně určuje informační zdroj v rámci celého Internetu, je velice důležitá a přínosná vlastnost. Má však i své stinné stránky. Dejme tomu, že máme elektronickou verzi nějaké knihy. Tato kniha je rozdělena po kapitolách do několika souborů. Co kapitola, to jeden soubor. Jeden soubor s obsahem přitom obsahuje odkazy na ostatní soubory (kapitoly). V tomto souboru bychom samozřejmě mohli uvádět celá URL všech kapitol (včetně schématu, adresy počítače, adresáře a jména souboru). Tato metoda však vede k překlepům při opisování poměrně dlouhých URL a rovněž může dojít k akutnímu zauzlování prstů na klávesnici. Druhou nevýhodu je, že při přesunu celé knihy na jiný server musíme měnit všechny odkazy.

URL, která obsahují schéma, adresu počítače a případné určení cesty k dokumentu, se nazývají *absolutní URL*. Z výše zmíněných příčin by však bylo užitečné, kdyby šlo napsat URL, jež by říkalo: „Načti soubor `prvni_kapitola.html`,

který je na stejném místě jako aktuální dokument“. Pro tyto účely bylo vytvořeno *relativní URL*.

Relativní URL obsahuje jen část informace o umístění zdroje. Aby mohlo být z relativního URL získáno absolutní, které ukazuje na informaci přesně, musí být část informace převzata z kontextu. Kontextem bývá nejčastěji URL dokumentu, ve kterém se relativní URL vyskytuje. URL, které vytváří potřebný kontext, se nazývá *základní*. Jestliže známe základní a relativní URL, můžeme je složit do výsledného absolutního URL, které již přesně určuje požadovaný informační zdroj.

Vše si ilustrujeme na malém příkladě. Dejme tomu, že na adrese `http://www.server.cz/kniha/obsah.html` je uložen obsah knihy. Ve stejném adresáři jsou soubory `prvni_kapitola.html`, `druha_kapitola.html` atd. Při použití relativních URL stačí, když v souboru `obsah.html` budou odkazy na jednotlivé soubory (odkazy nyní budou relativní a budou to pouze prostá jména souborů `prvni_kapitola.html` a `druha_kapitola.html`). Za základní URL se bude považovat URL `http://www.server.cz/kniha/obsah.html`. Po složení základního URL s relativními dostaneme správná absolutní URL jednotlivých kapitol:

```
http://www.server.cz/kniha/prvni_kapitola.html
http://www.server.cz/kniha/druha_kapitola.html
```

Relativní URL mohou kromě jména souboru či relativně určených podadresářů obsahovat i znaky ‘`.`’ a ‘`..`’. Tečka má přitom význam aktuálního adresáře a dvě tečky slouží k přechodu na nadřazenou úroveň v adresářové struktuře.

Mezi našimi vlastními stránkami bychom měli používat výhradně relativní URL. Nebude nám pak činit problém přenést celou strukturu dokumentů na jiný server.

Relativní URL samozřejmě nelze použít pro všechna schémata. Pro schémata `file`, `ftp`, `http` a `nntp` je lze použít vždy. U schémat `mailto`, `news` a `telnet` nemá použití relativních URL smysl. U ostatních schémat (`gopher`, `prospero` a `wais`) pouze někdy.

Určení základního URL

Z předchozího vyplývá, že pro správné získání absolutního URL složením základního a relativního je nezbytně nutné správně určit základní URL. Při zjišťování základního URL se postupuje následujícími kroky, dokud není zjištěno.

1. Základní URL je uvedeno přímo v dokumentu, pro který jej zjišťujeme. V HTML pro tyto účely slouží element `<BASE>`.

2. Za základní URL se vezme základní URL nadřazeného objektu. Tento způsob není příliš obvyklý a využije se např. u dokumentů, které jsou části dopisu zakódovaného podle specifikace MIME.[3]
3. Za základní URL se vezme URL dokumentu.
4. V případě, že neznáme ani URL dokumentu, považujeme základní URL za prázdné.

V praxi se nejčastěji uplatní třetí varianta, stejně jako v našem příkladě.

Skládání základního a relativního URL

Před chvílí jsme ukázali, jak se v některých situacích skládá relativní a základní URL. Přesný algoritmus skládání základního a relativního URL dohromady do absolutního URL je popsán v [8]. My se pokusíme popsat celý proces jednodušeji. Pro tuto chvíli budeme předpokládat, že URL má tvar:

«schéma»`://`*«lokace»*`/`*«cesta»*`;`*«parametry»*`?`*«dotaz»*`#`*«fragment»*

Oproti naší původní definici jsme pouze několik částí URL shrnuli pod jednu část *«lokace»*.

Základním předpokladem je zjištění základního URL tak, jak jsme to popsali v předchozí sekci. Poté aplikujeme postupně následující pravidla:

1. Jestliže je základní URL prázdné, absolutní URL je stejné jako relativní a jsme hotovi.
2. Jestliže je prázdné relativní URL, absolutní URL je stejné jako základní a jsme hotovi.
3. Jestliže relativní URL začíná určením schématu, bude absolutní URL stejné jako relativní a jsme hotovi.
4. Jestliže je část *«lokace»* relativního URL neprázdná, absolutní URL získáme složením schématu základního URL s relativním URL a jsme hotovi.
5. Jestliže je v relativním URL před částí *«cesta»* lomítko (`/`), absolutní URL získáme jako schéma a lokaci základního URL složené s relativním URL a jsme hotovi.
6. Jestliže je cesta v relativním URL prázdná (a nepředchází jí lomítko), pak se absolutní URL skládá ze schématu, lokace, cesty základního URL a dále:

[3] MIME (Multipurpose Internet Mail Extensions) je standard, který umožňuje pomocí elektronické pošty přenášet v jednom dopise pohromadě několik dalších dokumentů či multimediálních souborů. MIME je popsáno v RFC 2045 a RFC 2046 (`ftp://ftp.vse.cz/pub/docs/rfc/rfc2045.txt` a `ftp://ftp.vse.cz/pub/docs/rfc/rfc2046.txt`).

a) v případě, že relativní URL obsahuje parametry, pak z těchto parametrů; v opačném případě se připojí parametry ze základního URL a

b) v případě, že relativní URL obsahuje dotaz, pak se připojí tento dotaz; v opačném případě se připojí dotaz ze základního URL a

c) v případě, že relativní URL obsahuje fragment, pak se připojí tento fragment.

A jsme hotovi.

7. Ze základního URL je odstraněno vše za posledním lomítkem. K tomu se připojí cesta, parametry, dotaz a fragment z relativního URL. Předtím, než se tento výsledek uloží do absolutního URL, se provedou následující úpravy:
   a) Všechny výskyty '`./`', kde tečka je celý segment[4] cesty, se odstraní.
   b) Jestliže cesta končí tečkou jako celým segmentem, tato tečka se odstraní.
   c) Odstraní se všechny výskyty '*«segment»*`/../`', kde *«segment»* je celý segment cesty různý od '`..`'. Při odstraňovaní se postupuje zleva, dokud je co odstraňovat.
   d) Končí-li cesta na '*«segment»*`/..`', kde *«segment»* je celý segment cesty různý od '`..`', odstraní se '*«segment»*`/..`'.

Tento poněkud suchý a zdlouhavý popis osvěžíme praktickou ukázkou skládání základního URL s různými relativními URL. Za základní URL položíme `http://www.manicka.cz/pub/users/index.html`. V tabulce 2-3 si můžeme prohlédnout, jak dopadnou výsledky skládání s různými relativními URL.

Relativní URL	Výsledné (absolutní) URL
`uzivatele.html`	`http://www.manicka.cz/pub/users/uzivatele.html`
`spejbl/drevaky.html`	`http://www.manicka.cz/pub/users/spejbl/drevaky.html`
`../globus.html`	`http://www.manicka.cz/pub/globus.html`
`../info/logo.gif`	`http://www.manicka.cz/pub/info/logo.gif`
`#sekce3`	`http://www.manicka.cz/pub/users/index.html#sekce3`
`ftp://ftp.vse.cz/pub`	`ftp://ftp.vse.cz/pub`
`../..`	`http://www.manicka.cz/`
`../`	`http://www.manicka.cz/pub/`
`..`	`http://www.manicka.cz/pub/`

Tab. 2-3: Ukázky skládání URL

[4] Segment cesty je nejmenší část cesty ohraničená ze předu i zezadu lomítkem nebo okrajem cesty.

Víme, že se pomocí dvou teček (`..`) můžeme dosat o úroveň výš v adresářové struktuře. Tuto metodu lze používat pouze pro změnu adresáře v cestě. Pokoušet se tímto způsobem např. o změnu adresy serveru by bylo zcela nesprávné. Např. složení našeho základního URL s `../../../www.seznam.cz` nepovede k absolutnímu URL `http://www.seznam.cz`, ale k absolutnímu nesmyslu.

3. Základy HTML

V této kapitole se naučíme vytvářet jednoduché webovské stránky. Jednoduché v tomto případě neznamená, že půjde o nějaké slaboduché či nedomrlé stránky. Jednoduché bude pouze vlastní vytváření stránek. Po zvládnutí této kapitoly budete schopni vytvořit stránku, která velice přehledným způsobem prezentuje různé informace.

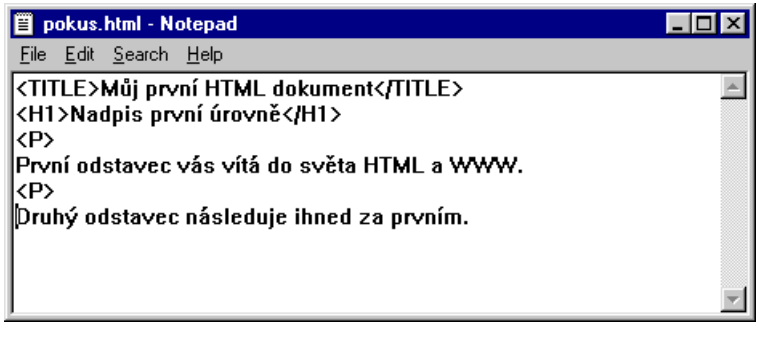
3.1 Naše první stránka

Z předchozího již víme, že webovské stránky neboli HTML dokumenty jsou obyčejné textové soubory. Textový soubor pak obsahuje koktejl namíchaný jednak ze samotného textu a druhak ze speciálních značek, které všemu dávají tu správnou chuť a barvu — určují význam jednotlivých částí textu.

Pro vytvoření naší první stránky proto budeme potřebovat jednoduchý textový editor. V prostředí *Windows* nám pro tyto účely bohatě poslouží program *Poznámkový blok* (*Notepad*). Spustíme jej a napíšeme v něm následující text:

```
<TITLE>Můj první HTML dokument</TITLE>
<H1>Nadpis první úrovně</H1>
<P>
První odstavec vás vítá do světa HTML a WWW.
<P>
Druhý odstavec následuje ihned za prvním.
```

☺ Pokud jste ve *Windows* a používáte českou klávesnici, máte možná problémy s napsáním znaků menšítko (<) a většítko (>). Na české klávesnici je lze velice pohodlně získat stiskem Pravý-Alt + < respektive Pravý-Alt + >.

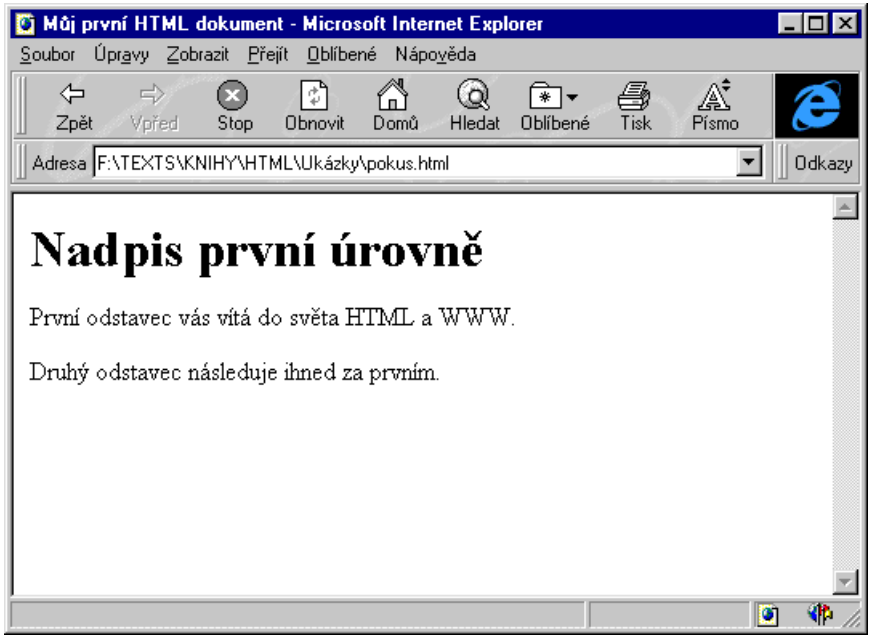


Obr. 3-1: Vytváření stránky v Notepadu

Abychom si napsaný text mohli prohlédnout v prohlížeči, musíme jej nejprve uložit do souboru. V *Poznámkovém bloku* tedy vybereme z menu příkaz Soubor ▷ ▷ Uložit jako... (File ▷ Save As...). Jako jméno souboru můžeme uvést v podstatě cokoliv, přípona souboru by však měla být `.html`.[1] My naši pokusnou stránku uložíme do souboru `pokus.html`.

Nyní je správný čas pro spuštění prohlížeče. Spustíme tedy *Netscape Navigator* nebo *Internet Explorer*. V menu *Netscape* vybereme příkaz File ▷ Open resp. Soubor ▷ Otevřít v *Exploreru*, abychom mohli otevřít a zobrazit námi vytvořený HTML dokument. Vybereme soubor `pokus.html` a výběr potvrdíme.

Náš první smělý pokus by se měl v prohlížeči zobrazit podobně jako na obrázku 3-2.



Obr. 3-2: Zobrazení naší první stránky v Internet Exploreru

V případě, že zobrazení je nesprávné, musíme zkontrolovat zápis naší první stránky v *Poznámkovém bloku*. Opravíme případné překlepy, soubor znovu uložíme a v prohlížeči stiskneme tlačítko Obnovit nebo Reload. Pokud jsme vše opravili správně, stránka by měla být v pořádku.

Nyní se podívejme na význam jednotlivých značek, které jsme použili při zápisu naší první stránky. Vžitý termín pro tyto značky je *tag*. Tagy určují význam textu, který je mezi ně uzavřen:

[1] Pokud pracujete v MS-DOSu nebo ve *Windows 3.x*, zkraťte příponu na `.htm`.

- tag <TITLE> a odpovídající ukončovací tag </TITLE> vymezují text názvu celého dokumentu;
- mezi tagy <H1> a </H1> je nadpis stránky;
- tag <P> od sebe odděluje jednotlivé odstavce textu.

Jak vidíme, každý tag se skládá ze znaku '<' (symbol menší než), svého jména a znaku '>' (symbol větší než). Obvykle se tagy vyskytují v párech — příslušný ukončovací tag má před svým jménem ještě znak '/' (lomítko). V našem příkladě <H1> sdělilo prohlížeči, že následující text je nadpis, a </H1> vyznačilo konec tohoto nadpisu.

Celému textu mezi počátečním a ukončovacím tagem se říká *element*. Někdy budeme o elementu hovořit i ve smyslu tagu, který se obecně používá k nějakému účelu (např. „element H1 slouží k vyznačení nadpisu první úrovně").

Z ukázky je rovněž patrné, že některé tagy se nemusí vyskytovat v párech — např. <P>. Jedná se o tagy, kde si místo správného výskytu ukončovacího tagu umí prohlížeč domyslet sám. Dojde mu tedy, že když v textu odstavce narazí na tag pro začátek odstavce, musí nejprve ten předchozí odstavec ukončit.

☺ HTML v názvech tagů nerozlišuje mezi malými a velkými písmeny. Význam následujících čtyř tagů je tedy zcela stejný: <title>, <TITLE>, <TiTle>, <tItLE>. V budoucnosti oceníte, když budete neustále používat pouze jeden z těchto způsobů zápisu. Mně osobně se nejvíce osvědčil druhý způsob, protože tagy zapsané velkými písmeny lze velmi snadno rozlišit od okolního textu.

Správná kostra

Jistě se shodneme na tom, že naše první ukázka byla opravdu velice jednoduchá. Jen pro zajímavost — aby stránka vyhovovala specifikaci HTML, stačí, když obsahuje název stránky uzavřený mezi tagy <TITLE> a </TITLE>. Tento název stránky se nejčastěji zobrazuje jako titulek okna prohlížeče. Neměl by proto být příliš dlouhý — doporučuje se délka maximálně 64 znaků.

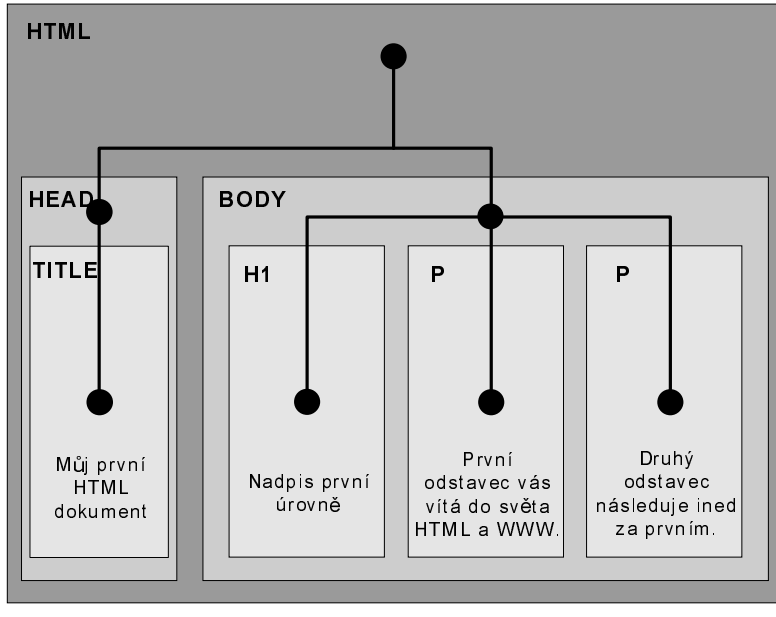
Ač se naše ukázka zobrazila v prohlížeči správně, nebyla zcela správná z hlediska formálního. Profesionál by ji zapsal takto:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">
<HTML>
<HEAD>
<TITLE>Můj první HTML dokument</TITLE>
</HEAD>
<BODY>
```

```
<H1>Nadpis první úrovně</H1>
<P>
První odstavec vás vítá do světa HTML a WWW.
<P>
Druhý odstavec následuje ihned za prvním.
</BODY>
</HTML>
```


Výsledek v prohlížeči se vůbec nezmění, odlišný je pouze zápis v HTML. První řádka `<!DOCTYPE...>` určuje použitou verzi HTML (v našem případě verzi 3.2). Tato klauzule nám zaručí, že stránka půjde zpracovat různými nástroji vyvinutými pro práci s dokumenty na bázi SGML.

Kromě toho přibyly na stránce tři nové elementy. Ty vymezují hlavičku dokumentu (`<HEAD>...</HEAD>`), tělo dokumentu (`<BODY>...</BODY>`) a konečně celý dokument (`<HTML>...</HTML>`). Dokument si tedy můžeme představit jako stromově uspořádanou hierarchii jednotlivých elementů. Na nejvyšší úrovni je element `HTML`. Ten se skládá z hlavičky `HEAD` a těla dokumentu `BODY`. Tělo dokumentu se pak skládá z nadpisu `H1` a dvou odstavců `P`. Dokument je tak vlastně jakýmsi kontejnerem, který obsahuje různé elementy. Každý element je však opět kontejnerem — může tedy obsahovat buď další elementy nebo již vlastní text. Vždy se však po určité době dostaneme k samotnému textu — elementy, ve kterých je text postupně obsažen, určují jeho význam.



Obr. 3-3: Hierarchická struktura HTML dokumentu

Naše poznatky zobecníme a získáme tak jakousi kostru či šablonu, které by měly vyhovovat všechny námi vytvářené stránky:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">
<HTML>
<HEAD>
    <TITLE>«název dokumentu»</TITLE>
</HEAD>
<BODY>
    «tělo dokumentu»
</BODY>
</HTML>
```


☺ Šikovné je si tuto šablonu uložit do souboru a při tvorbě nové stránky ji pouze zkopírovat. Nemusíme pak vše pokaždé zdlouhavě opisovat. (To oceníme zejména u první řádky.)

3.2 Základní členění dokumentu

Již od základní školy nám vážení pedagogové vštěpovali do hlavy, že bychom při psaní slohové práce měli celý text vhodně rozdělit do několika odstavců. Odstavec je totiž základní jednotkou, která se využívá při logickém členění textu.

K oddělení jednotlivých odstavců se v HTML používá tag `<P>`. Prohlížeč před zobrazením jednoho odstavce provádí následující úpravy:

1. všechny konce řádků převede na mezery;
2. pokud bezprostředně za sebou následuje více mezer, nahradí je mezerou jedinou;
3. nakonec takto získaný text odstavce zalomí tak, aby se vešel do okna prohlížeče.

Praktický důsledek tohoto postupu je, že mezi slova můžeme psát mezer kolik chceme, vždy se však zobrazí pouze jedna. Rovněž můžeme ukončovat řádky na libovolném místě — konce řádek v prohlížeči stejně budou většinou na jiných místech.

☺ Pro větší přehlednost je vhodné odstavce kromě tagu <P> oddělovat i prázdnou řádkou.

Abychom si vše demonstrovali, následující „hustá“ forma zápisu

```
<P>První odstavec.<P>Druhý je delší.<P>A třetí ihned
následuje.</P>
```

dává v prohlížeči stejné výsledky jako tato přehlednější a místy trochu rozvláčná forma:


```
<P>
První           odstavec.

<P>
Druhý je delší.

<P>
A třetí ihned následuje.
</P>
```

Vidíme tedy, že se mezera za počátečním tagem na konci řádky ignoruje. A obdobně se ignoruje i mezera na konci řádky, která předchází ukončovacímu tagu na další řádce.

To, že se text zarovnává podle aktuální velikosti okna, je velice užitečná vlastnost — HTML dokument lze zobrazit na obrazovkách s různým rozlišením. V některých případech však potřebujeme, aby se řádka zalomila na určitém přesně daném místě. Požadovaného efektu dosáhneme použitím tagu <BR> v místě požadovaného zlomu. Malá ukázka:

```
S naší firmou se můžete spojit na adrese<BR>
Židlička, a.s.<BR>
Kulatá 7<BR>
379 12 Dolní Horní
```

dopadne v prohlížeči takhle:

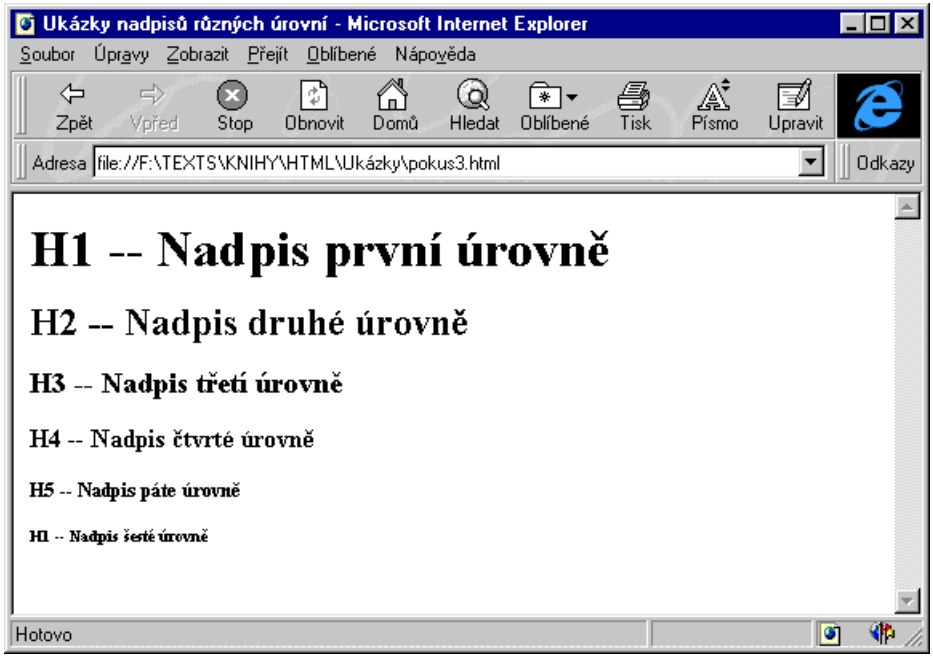
S naší firmou se můžete spojit na adrese
Židlička, a.s.
Kulatá 7
379 12 Dolní Horní

V naší první stránce jsme kromě odstavců použili ještě jeden element, který sloužil k členění dokumentu. Jednalo se o element H1 použitý k vytvoření nadpisu. HTML nám dává k dispozici nadpisy šesti úrovní. Nejdůležitější je přitom nadpis první úrovně (H1) a nejméně důležitý je nadpis šesté úrovně (H6). Nadpisy

vyšší úrovně (s nižším číslem) jsou obvykle zobrazovány větším a výraznějším písmem. Nadpis můžeme obecně zapsat jako

<H*«n»*>*«text nadpisu»*</H*«n»*>

kde *«n»* je číslo úrovně od jedné do šesti. Na obrázku 3-4 si můžeme prohlédnout, jak se jednotlivé úrovně nadpisů zobrazí v prohlížeči. Pro procvičení si ukážeme i zápis v HTML:



Obr. 3-4: Ukázka všech šesti úrovní nadpisů

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">
<HTML>
<HEAD>
    <TITLE>Ukázky nadpisů různých úrovní</TITLE>
</HEAD>
<BODY>
<H1>H1 -- Nadpis první úrovně</H1>
<H2>H2 -- Nadpis druhé úrovně</H2>
<H3>H3 -- Nadpis třetí úrovně</H3>
<H4>H4 -- Nadpis čtvrté úrovně</H4>
<H5>H5 -- Nadpis páté úrovně</H5>
```

```
<H6>H1 -- Nadpis šesté úrovně</H6>
</BODY>
</HTML>
```

První nadpis, který v dokumentu použijeme, by měl být `H1`. Při následném jemnějším členění dokumentu pomocí dalších úrovní nadpisů bychom neměli úrovně nadpisů přeskakovat.

Používání nadpisů páté a šesté úrovně je poněkud problematické — zobrazují se většinou menším písmem než je běžné písmo textu odstavce. Dokument pak trošku ztrácí na přehlednosti. V těchto případech je zcela na místě zvážit možnost rozdělení jednoho dokumentu do několika samostatných. Tím snížíme počet použitých úrovní nadpisů o jeden. Měli bychom si zapamatovat, že příliš hluboká úroveň zanoření nadpisů v jednom dokumentu znesnadňuje orientaci v textu.

Je ustáleným zvykem, že první nadpis v dokumentu je totožný s titulkem (názvem) stránky. Výjimku tvoří dokumenty, které jsou částí nějaké větší ucelené skupiny dokumentů. Nadpis dokumentu by měl odpovídat názvu kapitoly nebo části a titulek by měl dokument identifikovat v širším kontextu. Pro titulek se osvědčilo použít název celé skupiny dokumentů doplněný o název aktuálního dokumentu.

Pokud chceme od sebe dvě části stránky oddělit opticky výrazněji, můžeme použít tag `<HR>`. Ten do stránky vloží horizontální čáru přes celou šířku okna prohlížeče.

3.3 Měníme typy písma

Dnes je zcela samozřejmou vlastností každého textového editoru, že umožňuje v jednom dokumentu najednou používat různé druhy písma — tučné, kurzívu, různě velké atd. Obdobné možnosti nám nabízí i HTML v podobě elementů, které slouží ke změně použitého druhu písma. Tyto elementy můžeme rozdělit na dvě skupiny. Na elementy sloužící k logickému vyznačování a na ty sloužící k fyzickému vyznačování.

Při prvním způsobu označujeme části textu tagy, které prohlížeči říkají: „toto je důležité a chci to mít zvýrazněno“, „tohle je název nějakého citovaného díla“, „toto je proměnná“ apod. Text tedy označujeme podle významu. Prohlížeč při zobrazování použije písmo, které je pro daný druh informace obvyklé v tom kterém hostitelském operačním systému. Uživatel se tedy cítí jako doma, protože podle druhu písma snadno rozezná druh informace.

Při druhém způsobu přímo určujeme použitý druh písma: „tento text bude tučně“, „tenhle kurzívou“. V tomto případě má nad způsobem zobrazení dokonalou kontrolu autor stránky.

Vidíme, že spíše logické styly odpovídají filosofii jazyka HTML — slouží k určení významu a ne pouze vizuálních vlastností daného textu.

Pokud chceme změnit styl písma pro nějaký text, stačí tento text uzavřít mezi příslušný počáteční a ukončovací tag. V naší ukázce bude změněn druh zobrazení slova `slovo`:

```
Do normálního textu umístíme <TAG>slovo</TAG>, které chceme
zvýraznit.
```


Na místě fiktivního elementu `<TAG>` může být jakýkoliv z logických (viz tab. 3-1) nebo fyzických (viz tab. 3-2 na následující straně) stylů písma.

<TAG>	Popis elementu
<CITE>	Nejčastěji je používán pro označování názvů knih, článků či jiných citací. Obvykle bývá zobrazován jako kurzíva.
<CODE>	Indikuje ukázku kódu nějakého programu nebo HTML stránky. Používá se pro velmi krátké ukázky kódu; pro víceřádkové výpisy je určen element <PRE>. Element <CODE> bývá obvykle zobrazován neproporcionálním písmem.
<EM>	Bývá používán pro zvýraznění určité fráze. Nejčastěji je zobrazován jako kurzíva.
<KBD>	Používá se pro vyznačení textu, který je zadáván uživatelem. To lze využít např. v uživatelských příručkách. Bývá zobrazován neproporcionálním písmem.
<SAMP>	Používá se pro zápis výstupů z různých programů a skriptů. Nejčastěji je zobrazován jako neproporcionální písmo.
<STRONG>	Bývá používáno jako silné zvýraznění a nejčastěji je zobrazováno jako tučné písmo.
<VAR>	Používá se pro označení proměnných. Obvykle je zobrazován kurzívou.
<DFN>	Vyznačuje termín, který je právě definován.

Tab. 3-1: Přehled logických stylů písma

<TAG>	Popis elementu
<B>	Vyznačuje v textu tučné písmo.
<I>	Vyznačuje kurzívu.
<TT>	Vyznačuje neproporcionální písmo.
<U>	Vyznačuje podtržený text.
<STRIKE>	Vyznačuje přeškrtnuté písmo.
<BIG>	Použije se větší písmo.
<SMALL>	Použije se menší písmo.
<SUB>	Vyznačuje dolní index.
<SUP>	Vyznačuje horní index.

Tab. 3-2: Přehled fyzických stylů písma

Na následujícím kódu si ukážeme, jak se dají používat logické styly.

```
Jestliže si <EM>nejste</EM> jisti správností vašeho dokumentu,
použijte příkaz <CODE>html-check <VAR>file</VAR> | more</CODE>
pro zkontrolování správnosti souboru <VAR>file</VAR>.
```

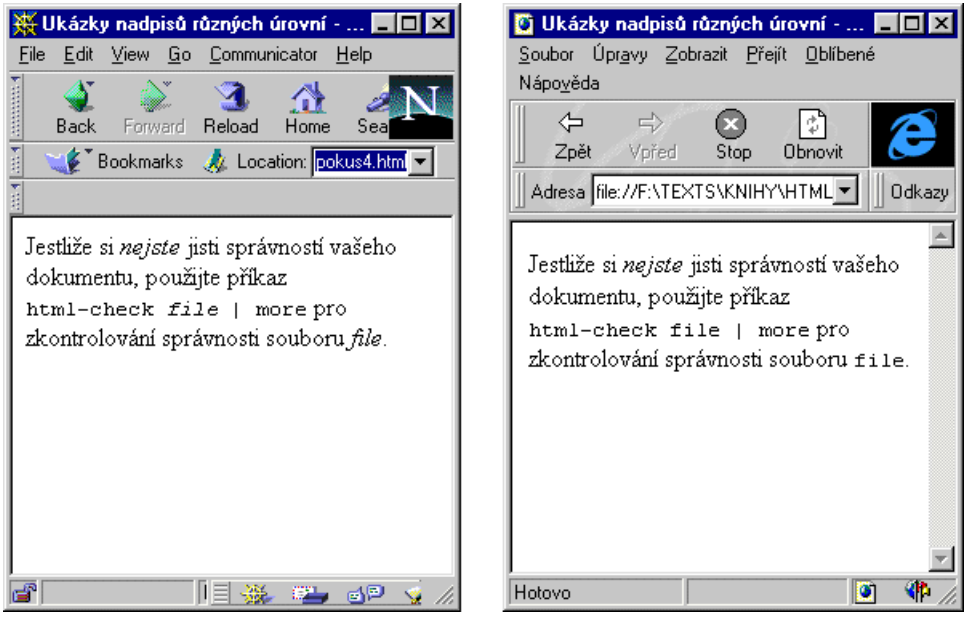
Na obrázku 3-5 na následující straně vidíme, že logické styly písma se mohou zobrazit v různých prohlížečích různě. *Netscape Navigator* používá k zobrazení proměnných (element `VAR`) kurzívu, kdežto *Internet Explorer* neproporcionální písmo. U fyzických stylů písma se prohlížeč snaží dodržet požadovaný druh písma. Pouze v případech, kdy dané písmo nemá k dispozici, může použít jiný způsob zvýraznění — např. změnu barvy.

Abychom neošidili fyzické styly písma, malá ukázka jejich použití:

```
<P><B>Albert Einstein</B> je mimo jiné autorem rovnice
<I>E=mc<SUP>2</SUP></I>. Jeho <U>Speciální teorie
relativity</U> nadělala vrásky mnoha studentům fyziky.

<P><BIG>Stanu se menším</BIG> a ještě menším <SMALL>až budu
nejmenší na celém světě</SMALL>. A to pak zapiju koncentrovaným
roztokem C<SUB>2</SUB>H<SUB>5</SUB>OH.
```

Albert Einstein je mimo jiné autorem rovnice $E=mc^2$. Jeho Speciální teorie relativity nadělala vrásky mnoha studentům fyziky.

Stanu se menším a ještě menším až budu nejmenší na celém světě. A to pak zapiju koncentrovaným roztokem C_2H_5OH.



Obr. 3-5: Odlišnost zobrazení logických stylů písma

Pokud do sebe vnoříme několik úrovní typů písma, není jednoznačně určen výsledný typ písma. Pokud použijeme `<I>abc<B>def</B>ghi</I>`, bude text `abc` a `ghi` určitě zobrazen kurzívou. Standard HTML však neříká nic o tom, zda má být `def` zobrazeno tučně nebo tučnou kurzívou.

Při podrobnějším studiu stránek, které potkáme při svém brouzdání po Internetu, se poměrně často setkáme se dvěma chybami. Tyto chyby spočívají v tom, že jednotlivé elementy se kříží a nedodržují tak hierarchickou strukturu vzájemného vnořování tak, jak jsme si ukázali v první části této kapitoly. Při tvorbě vlastních stránek bychom měli mít na paměti následující dvě pravidla:

- Nikdy nesmíme jednotlivé elementy překřížit. Nesmíme tedy používat konstrukce typu: `<CITE>Bylo <B>nás</CITE> pět</B>`.
- Pokud chceme typ písma změnit pro více než jeden odstavec, měli bychom text každého odstavce uzavřít do vlastních tagů pro začátek a konec daného typu písma. Zabráníme tak tomu, aby se nám křížil element `<P>` s elementem pro typ písma. Pokud toto pravidlo porušíme, na první pohled se nic nestane, protože dnešní prohlížeče jsou k formátu vstupních souborů velmi benevolentní.

3.4 Odkazy

Pravým kořením každé stránky jsou odkazy. Odkaz je zvýrazněná část stránky, za kterou se skrývá URL. Tím, že aktivujeme odkaz, dáme prohlížeči příkaz k zobrazení stránky s tímto URL. Před vytvořením každého odkazu si musíme rozmyslet dvě věci:

- URL zdroje, na který odkaz bude ukazovat;
- text, který bude odkaz označovat.

Pro vložení odkazu do webovské stránky pak poslouží následující konstrukce:

`<A HREF="`*«URL»*`">`*«text odkazu»*`</A>`

Nejčastější použití odkazů je přímo v textu, kde vytvoříme odkaz na další informace týkající se určitého pojmu:

```
Mnoho zajímavých informací o službě WWW lze nalézt na serveru
<A HREF="http://www.w3.org/">konsorcia W3C</A>. Naleznete zde
i informace o nejnovější verzi jazyka HTML.
```

V prohlížeči bývá text odkazu obvykle podtržen a zobrazen odlišnou barvou:

Mnoho zajímavých informací o službě WWW lze nalézt na serveru konsorcia W3C.
Naleznete zde i informace o nejnovější verzi jazyka HTML.

Vraťme se však k samotnému zápisu odkazu v HTML. Součástí tagu `<A>` je zde i text `HREF="`*«URL»*`"`, což je pro nás novinka. Jedná se o tzv. *atribut*. Atributy slouží k určení vlastností určitého tagu. V našem případě atribut `HREF` slouží k určení URL, kam vede odkaz. Atributy je možné uvádět pouze u počátečních tagů (tedy u `<TAG>` a ne u `</TAG>`) a obecně jich může být i více. Obecný tvar zápisu tagu s atributy je

`<`*«název tagu»* *«atribut1»*`=`*«hodnota»* *«atribut2»*`=`*«hodnota»* `... >`

Hodnota atributu by měla být uzavřena v uvozovkách. To není nutné, pokud se hodnota skládá jen z malých a velkých písmen anglické abecedy, číslic, pomlčky a tečky.

✍ *Hodnotu atributu můžeme kromě uvozovek uzavřít i do apostrofů. Lepší je však používat uvozovky, protože opticky lépe oddělují hodnotu atributu od ostatního textu.*

U některých elementů existují i atributy, u kterých není potřeba uvádět žádnou hodnotu. Tyto atributy slouží pouze jako přepínače ano/ne — při jejich uvedení je aktivována určitá vlastnost elementu. Tím, že se neuvádí hodnota, nepoužívá se ani rovnítko. Obecně bychom to mohli zapsat jako:

`<`*«název tagu»* *«atribut1»* `... >`

Samozřejmě, že existují elementy, ve kterých lze použít oba dva druhy atributů. Pokud lze u nějakého elementu použít více atributů, nezáleží na pořadí, v jakém je uvedeme.

Již jsme si řekli, že hodnoty atributů se uzavírají do uvozovek. Může se však stát, že jako součást hodnoty atributu potřebujeme použít i uvozovku (např. jako součást URL u atributu `HREF`). V tomto případě je nutné místo uvozovky použít tzv. *znakovou entitu*, která zabrání zmatkům v hodnotě atributu. Znakových entit existuje mnoho — pro uvozovky existuje entita `&quot;`. Podobná entita existuje i pro znaky '`<`' a '`>`', které slouží k oddělení názvů tagů od textu. Menšítko lze v HTML zapsat jako `<` a většítko jako `>`. Vidíme, že všechny znakové entity začínají znakem '`&`' — pokud chceme tento znak zapsat do textu stránky, musíme použít entitu `&`. Ukážeme si, že použité znakové entity se zobrazí jako požadované znaky:

```
<P>Vyřešte nerovnici 3<VAR>x</VAR> - 7 &lt; 2

<P>Mezi přední poskytovatele ekonomických informací patří
i firma Dun &amp; Bradstreet.
```

Vyřešte nerovnici $3x - 7 < 2$

Mezi přední poskytovatele ekonomických informací patří i firma Dun & Bradstreet.

Vraťme se však k odkazům. U popisu URL jsme si řekli, že poslední částí URL může být fragment, který určuje konkrétní část celého dokumentu. Abychom mohli používat fragmenty, musíme nejprve v cílových dokumentech definovat návěstí. Návěstí je jméno části stránky. Toto jméno se pak používá jako fragment při tvorbě odkazů. Návěstí definujeme opět pomocí elementu `A`, tentokráte však s použitím atributu `NAME`:

`<A NAME="`*«jméno»*`">`*«označený text»*`</A>`

Vše demonstrujeme i na konkrétním příkladě. Dejme tomu, že máme stránku `bilance.html`, která obsahuje bilance firmy v jednotlivých letech. Víme, že z jiných dokumentů budeme vytvářet odkazy přímo na bilance jednotlivých let. Označíme proto jednotlivé bilance návěstími. Soubor `bilance.html`:

```
...
<H3><A NAME="rok1995">Bilance za rok 1995</A></H3>
```

```
V roce 1995 dosáhla naše firma...
```

K odkázání na bilanci za rok 1995 pak poslouží URL `bilance.html#rok1995`. Fragmenty lze používat i k odkazům v rámci jednoho dokumentu. Na začátek souboru `bilance.html` bychom mohli klidně umístit odkazy na bilance jednotlivých let:

```
<P><A HREF="#rok1994">Bilance za rok 1994</A>
<P><A HREF="#rok1995">Bilance za rok 1995</A>
<P><A HREF="#rok1996">Bilance za rok 1996</A>
```


3.5 Seznamy

HTML bylo vytvořeno pro psaní dobře strukturovaných dokumentů. Typickým strukturovaným prvkem, objevujícím se ve většině dokumentů, jsou různé seznamy. V HTML nalezneme podporu hned pro tři druhy seznamů — nečíslované, číslované a definiční. Při vytváření nečíslovaného seznamu postupujeme následovně:

1. celý seznam zahájíme tagem `<UL>`;
2. napíšeme potřebný počet položek seznamu, každou položku zahájíme tagem `<LI>`;
3. seznam ukončíme tagem `</UL>`.

Malá ukázka jednoduchého seznamu:

```
Nabízené druhy ovoce:
<UL>
    <LI>Jablka
    <LI>Hrušky
    <LI>Švestky
    <LI>Banány
</UL>
```

Nabízené druhy ovoce:

- Jablka
- Hrušky
- Švestky
- Banány

Pod jednou položkou seznamu může být skryto i více odstavců. Stačí je oddělit tagem `<P>`.

U číslovaných seznamů je před každou položku seznamu automaticky umisťováno pořadové číslo. Vytváření těchto seznamů je zcela identické s vytvářením nečíslovaných seznamů. Jednotlivé položky se opět uvozují tagem `<LI>`. Pro seznam však místo elementu `UL` použijeme `OL`:[2]

```
Pořadí ovoce podle prodejnosti:
<OL>
    <LI>Jablka
    <LI>Hrušky
    <LI>Švestky
    <LI>Banány
</OL>
```

Pořadí ovoce podle prodejnosti:

1. Jablka
2. Hrušky
3. Švestky
4. Banány

Definiční seznamy se od předchozích dvou liší. S výhodou je lze použít např. pro vytvoření slovníčku pojmů, který obsahuje termíny a jejich vysvětlení. K vytvoření seznamu slouží element `DL` (Definition List — definiční seznam). Postup vytvoření definičního seznamu je následovný:

1. seznam zahájíme tagem `<DL>`;
2. před definovaný termín zapíšeme tag `<DT>`;
3. mezi termín a jeho definici umístíme tag `<DD>`;
4. opakujeme kroky 2 a 3 pro každý termín v seznamu;
5. seznam ukončíme tagem `</DL>`.

Vytvořit takto slovníček protokolů používaných v Internetu není žádný problém:

```
<DL>
    <DT>HTTP
    <DD>Transportní protokol využívaný k přenosu souborů
        obsahujících popis WWW-stránek v jazyce HTML.
    <DT>FTP
    <DD>Transportní protokol používaný k přenosu souborů.
    <DT>NNTP
    <DD>Transportní protokol používaný k přenosu news.
</DL>
```

[2] Zkratka UL pochází z anglického Unordered List (neuspořádaný seznam) a OL z Ordered List (uspořádaný seznam).

HTTP
Transportní protokol využívaný k přenosu souborů obsahujících popis WWW-stránek v jazyce HTML.
FTP
Transportní protokol používaný k přenosu souborů.
NNTP
Transportní protokol používaný k přenosu news.

☺ Vidíme, že definované termíny nejsou příliš výrazné. Mnohem lépe vypadá definiční seznam, ve kterém termíny zvýrazníme např. pomocí elementu `STRONG`.

```
<DL>
    <DT><STRONG>HTTP</STRONG>
    <DD>Transportní protokol využívaný k přenosu souborů
        obsahujících popis WWW-stránek v jazyce HTML.
    <DT><STRONG>FTP</STRONG>
    <DD>Transportní protokol používaný k přenosu souborů.
    <DT><STRONG>NNTP</STRONG>
    <DD>Transportní protokol používaný k přenosu news.
</DL>
```

HTTP
Transportní protokol využívaný k přenosu souborů obsahujících popis WWW-stránek v jazyce HTML.
FTP
Transportní protokol používaný k přenosu souborů.
NNTP
Transportní protokol používaný k přenosu news.

Seznamy mohou být do sebe libovolně vnořovány. To znamená, že součástí položky jednoho seznamu může být celý další seznam.

```
Pro tvorbu stránek v jazyce HTML můžeme použít:
<OL>
    <LI>Obyčejné textové editory
        <UL>
            <LI>Notepad
            <LI>Programmer's File Editor
            <LI>Emacs
        </UL>
    <LI>Textové editory podporující vytváření struktury HTML
        dokumentu
```

```
        <UL>
            <LI>HotMetal Pro
            <LI>asWedit
        </UL>
    <LI>WYSIWYG editory HTML stránek
        <UL>
            <LI>FrontPage
            <LI>Netscape Editor
            <LI>WebMagic Pro
        </UL>
</OL>
```


Pro tvorbu stránek v jazyce HTML můžeme použít:

1. Obyčejné textové editory
   - Notepad
   - Programmer's File Editor
   - Emacs
2. Textové editory podporující vytváření struktury HTML dokumentu
   - HotMetal Pro
   - asWedit
3. WYSIWYG editory HTML stránek
   - FrontPage
   - Netscape Editor
   - WebMagic Pro

3.6 Předformátovaný text

Normální text dokumentu je zarovnáván podle velikosti okna prohlížeče a je obvykle zobrazen proporcionálním písmem. Tento způsob zobrazování je však pro výpisy programů nebo textové tabulky nevhodný. V těchto případech potřebujeme, aby se text zobrazil přesně tak, jak je zapsán — aby se zachovaly konce řádků, všechny mezery a aby text se zobrazil neproporcionálním písmem. V HTML pro tyto účely nalezneme element `PRE`. Krátký program v jazyce C, který slouží k výpočtu faktoriálu, můžeme zapsat takto:

```
<PRE>
#include &lt;stdio.h&gt;

long f (long n) {
    return( n==0 ? 1 : n * f(n-1) );
    }
```

```
main() {
    printf("%ld\n", f(10));
    }
</PRE>

#include <stdio.h>

long f (long n) {
    return( n==0 ? 1 : n * f(n-1) );
    }

main() {
    printf("%ld\n", f(10));
    }
```


Vidíme, že zápis textu programu mezi tagy `<PRE>` a `</PRE>` je téměř shodný s výsledkem v prohlížeči. Jediný rozdíl spočívá v nahrazení znaků menšítka a většítka příslušnými znakovými entitami.

Uvnitř elementu `PRE` můžeme používat elementy pro změnu typu písma (např. `I`, `B` a `U`) a rovněž odkazy (`<A ...>`). Použití různých druhů písma umožňuje v zápisu zdrojových textů programů použít zvýraznění syntaxe a zlepšit tak jejich srozumitelnost. Následující kód v HTML:

```
<PRE>
<B>function NSD</B> (A, B: Longint): Longint;
<I>{Funkce vrací největšího společného dělitele čísel A a B.}
{K výpočtu je použit modifikovaný Euklidův algoritmus.}</I>
<B>begin</B>
  A:= Abs(A);
  B:= Abs(B);
  <B>while</B> (A&lt;&gt;0) <B>and</B> (B&lt;&gt;0) <B>do</B>
  <B>begin</B>
    <B>if</B> A&gt;B <B>then</B> A:= A <B>mod</B> B
           <B>else</B> B:= B <B>mod</B> A;
  <B>end;</B>
  <B>if</B> A=0 <B>then</B> NSD:= B
         <B>else</B> NSD:= A;
<B>end;</B>
</PRE>
```

zobrazí v prohlížeči úhledně zformátovaný prográmek:

```
function NSD (A, B: Longint): Longint;
{Funkce vrací největšího společného dělitele čísel A a B.}
{K výpočtu je použit modifikovaný Euklidův algoritmus.}
begin
  A:= Abs(A);
  B:= Abs(B);
  while (A<>0) and (B<>0) do
  begin
    if A>B then A:= A mod B
           else B:= B mod A;
  end;
  if A=0 then NSD:= B
         else NSD:= A;
end;
```


Samozřejmě, že ruční doplňování tagů do textu programu není zrovna moc pohodlné a rychlé. Tento proces lze naštěstí zautomatizovat a více si o něm povíme v části věnované konvertorům.

3.7 Začleňování citátů

Dalším elementem, o kterém jsme se dosud nezmínili, je `BLOCKQUOTE`. Používá se zejména v případech, kdy do stránky zařazujeme delší citaci jiného díla. Text umístěný mezi tagy `<BLOCKQUOTE>` a `</BLOCKQUOTE>` bývá od ostatního textu oddělen malou vertikální mezerou a navíc je obvykle zleva (či z obou stran) odsazen.

```
Dílo B. Henryho je mnohdy velmi kontroverzní, jako například
<CITE>Zenonova aporie Achilles a želva</CITE>:
<BLOCKQUOTE>
Zenon ukazoval, že Achilles nedostihne želvu, i když běží
rychleji než ona.
<P>
Celá tisíciletí bylo toto zjištění chápáno jako paradox neboli
aporie. Nikdo si však neuvědomil skutečný paradox odtud
vyplývající: Achilles nedostihne želvu, která běží daleko
pomaleji než on, takže želva běžící před ním doběhne jeho.
<P>
PS. Zlí jazykové však tvrdí, že Achilles nedostihne želvu
proto, že nemá v pořádku achilovky.
</BLOCKQUOTE>
```

Dílo B. Henryho je mnohdy velmi kontroverzní, jako například *Zenonova aporie Achilles a želva*:

> Zenon ukazoval, že Achilles nedostihne želvu, i když běží rychleji než ona.
>
> Celá tisíciletí bylo toto zjištění chápáno jako paradox neboli aporie. Nikdo si však neuvědomil skutečný paradox odtud vyplývající: Achilles nedostihne želvu, která běží daleko pomaleji než on, takže želva běžící před ním doběhne jeho.
>
> PS. Zlí jazykové však tvrdí, že Achilles nedostihne želvu proto, že nemá v pořádku achilovky.

Kromě použití elementu `BLOCKQUOTE` si všimněte vhodného použití `CITE`. Díky tomu, že obsah elementu `BLOCKQUOTE` bývá v prohlížeči odsazen, využívá se často v případech, kdy potřebujeme nějaký text odsadit.

3.8 Podpisy stránek

Bývá dobrým zvykem uvést na každé stránce spojení na autora stránky. Nejčastěji se uvádí pouze e-mailová adresa, někdy však i adresa úplná. Tato adresa by měla být uzavřena mezi tagy `<ADDRESS>` a `</ADDRESS>`. Ve většině prohlížečů se použití elementu `ADDRESS` nedotkne zobrazení (pouze některé prohlížeče zobrazují adresu kurzívou), ale může být využito různými programy, které automaticky indexují webovské stránky. My si ukážeme dva nejčastější způsoby použití:

```
<ADDRESS>Jiří Kosek -- xkosj06@vse.cz</ADDRESS>

<ADDRESS>
Výzkumný ústav nesmyslů<BR>
Cimrmanova 15<BR>
Liptákov<BR>
123 45<BR>
e-mail: jara@vun.cz
</ADDRESS>
```

Konce řádků v adrese nesmíme zapomenout explicitně vyznačit pomocí tagu `<BR>`. Uvedené adresy můžeme ještě vylepšit tím, že z e-mailových adres vytvoříme odkazy, které budou sloužit k vyvolání poštovního programu. Pro odeslání dopisu autorovi stránky pak stačí pouze pár kliknutí myší.

```
<ADDRESS>Jiří Kosek --
<A HREF="mailto:xkosj06@vse.cz">xkosj06@vse.cz</A></ADDRESS>
```

```
<ADDRESS>
Výzkumný ústav nesmyslů<BR>
Cimrmanova 15<BR>
Liptákov<BR>
123 45<BR>
e-mail: <A HREF="mailto:jara@vun.cz">jara@vun.cz</A>
</ADDRESS>
```

Jiří Kosek -- xkosj06@vse.cz
Výzkumný ústav nesmyslů
Cimrmanova 15
Liptákov
123 45
e-mail: jara@vun.cz

Nyní již toho víme dost, abychom se mohli pustit do samostatné tvorby vlastních stránek. V další kapitole se naučíme naše stránky zatraktivnit zařazením obrázků.


<IMG SR

4. Obrázky

V této kapitole se dozvíme, jak naše stránky ozdobit pomocí obrázků. To nám přinese hned několik výhod. Naše stránky se stanou zajímavější pro jejich čtenáře. Prezentace některých informací v grafické podobě je také mnohem efektivnější než v textové. Kromě základních informací o zařazování obrázků se naučíme upravit obrázky tak, aby byly naše stránky rychle přístupné i pro uživatele připojené pomocí pomalých telefonních linek. Nezatajíme ani „fígle“ jako transparentní a animované obrázky, které používají profesionální designéři webovských stránek.

4.1 Vložení obrázku do stránky

Prvním předpokladem pro vložení obrázku do stránky je jeho uložení ve vhodném formátu. Prohlížeče dnes podporují dva grafické formáty GIF a JPEG a pomalu začínají podporovat i poměrně nový formát PNG (čti ping), který byl vyvinut speciálně pro potřeby Webu. Obrázek většinou získáme buď jeho naskenováním z papírové předlohy nebo nakreslením v nějakém grafickém editoru. Musíme jej uložit v jednom z formátů GIF nebo JPEG. Měli bychom tedy získat soubor s příponou `.gif` nebo `.jpg`.

Dejme tomu, že máme obrázek mouchy v souboru `moucha.gif`. Tento obrázek zkopírujeme do adresáře se stránkou, do které chceme obrázek vložit. Ke vložení obrázku slouží nepárový element `IMG`. Soubor s obrázkem specifikujeme pomocí atributu `SRC`, který slouží k zapsání URL obrázku. Samozřejmě, že lze používat i relativní URL, takže naše první stránka s obrázkem bude opravdu jednoduchá:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">
<HTML>
<HEAD>
    <TITLE>Stránka s obrázkem</TITLE>
</HEAD>
<BODY>
<H1>Stránka s obrázkem</H1>

<P>Jen se na ní podívejte, jak mi baští večeři.

<P><IMG SRC="moucha.gif">

<P>Co byste řekli vy, kdyby vám moucha snědla večeři?
</BODY>
</HTML>
```

Obr. 4-1: První stránka s vloženým obrázkem

Všimněte si, že jsme obrázek vložili jako samostatný odstavec. Kdybychom to neudělali, byl by obrázek přilepen vpravo za textem prvního odstavce a celá stránka by nevypadala tak pěkně.

I když dnes mnoho uživatelů používá k toulkám po Webu grafické prohlížeče jako *Netscape* nebo *Explorer*, nemalá část uživatelů používá i prohlížeč *Lynx*, který pracuje v textovém režimu. Nemůže tedy zobrazovat obrázky, ale pouze vlastní text stránky. Aby nebyli uživatelé znakových prohlížečů takto diskriminováni, existuje u tagu `<IMG>` atribut `ALT`, jenž slouží k přidání krátkého popisu k obrázku. Ten se zobrazí místo obrázku v prohlížečích, které nepracují v grafickém režimu. Náš tag pro zařazení obrázku opravíme na:

```
<IMG SRC="moucha.gif" ALT="Moucha, která jí večeři">
```

☺ Atribut `ALT` by se měl používat u všech obrázků. Kromě již zmíněného účelu ho ocenění i ostatní uživatelé. Například při pomalé rychlosti linky, kdy je přenos stránky pomalý, jsou před definitivním přenesením obrázků zobrazeny alespoň vysvětlující popisky. Někteří uživatelé dokonce stahují jen ty obrázky, o kterých si myslí, že je potřebují. Náš stručný popis jim

tak ušetří mnoho času a zprostředkovaně i peněz (nejen čas, ale i telekomunikační poplatky jsou peníze).

Navíc obsah tohoto atributu může využít hlasový syntetizátor při převodu HTML stránky do řeči. To je zcela v souladu s aktivitami konsorcia W3C při vývoji technologií, které mají umožnit využívání Webu lidem s postižením zraku.

Obrázky jsou na atributy opravdu bohaté. Dalším z nich je atribut `ALIGN`, který určuje způsob zarovnání obrázku a okolního textu. Může nabývat pěti hodnot: `TOP`, `MIDDLE`, `BOTTOM`, `LEFT` a `RIGHT`.

Při použití prvních tří způsobů je obrázek součástí řádky textu. Při použití `TOP` se s řádkou zarovnává horní okraj obrázku. Při použití `MIDDLE` je obrázek vzhledem k řádce vertikálně vycentrován. Konečně při použití `BOTTOM` je s řádkou zarovnán spodní okraj obrázku.

Použití hodnot `LEFT` a `RIGHT` má zcela odlišný význam. V tomto případě je obrázek umístěn u levého resp. pravého okraje stránky a je obtékán textem. Význam jednotlivých hodnot nám přiblíží obrázek 4-2.

Obr. 4-2: Význam jednotlivých hodnot atributu ALIGN

Při používání obtékání obrázků můžeme narazit na jeden problém. Například většinou nevypadá moc hezky, když nadpis obtéká obrázek. To se však může snadno stát v případech, kdy předcházející text není dost dlouhý na obtečení celého obrázku. V těchto případech je možné použít tag `<BR>` s atributem `CLEAR=LEFT`, `CLEAR=RIGHT` nebo `CLEAR=ALL`. Text uvedený za `<BR>` bude pokračovat až pod obrázky. V případě, že u atributu `CLEAR` uvedeme hodnotu `LEFT`, bude text pokračovat pod obrázky vlevo, v případě hodnoty `RIGHT` pod obrázky vpravo. Pokud použijeme `CLEAR=ALL`, bude text pokračovat až pod všemi obtékanými obrázky a nezáleží na tom, zda byly umístěny vlevo či v pravo.

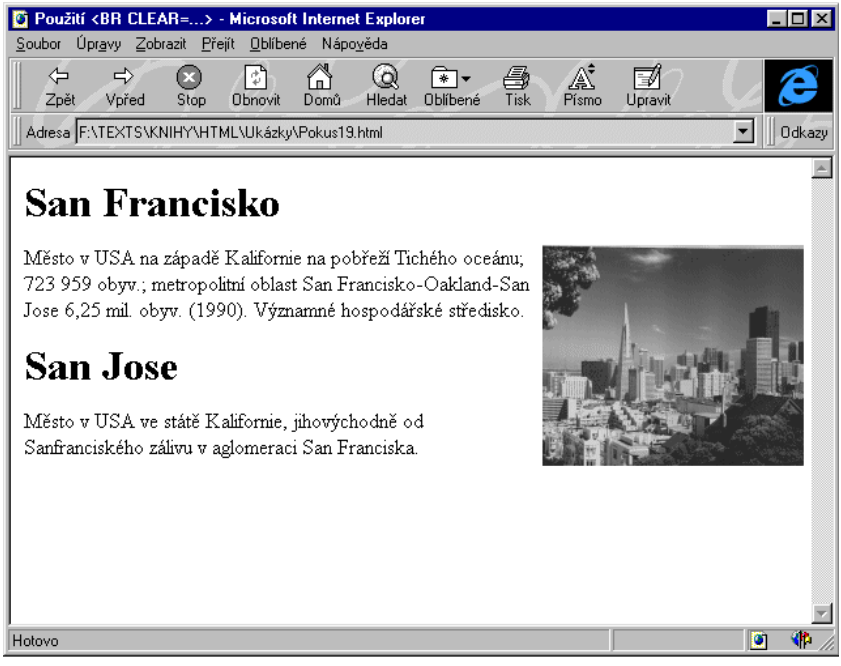
Praktická ukázka je na obrázku 4-3 na následující straně. V horním okně je ukázána stránka, kde nebylo použito `<BR CLEAR=...>`. V dolním pak stránka, která se liší pouze zařazením tagu `<BR>` na vhodné místo:

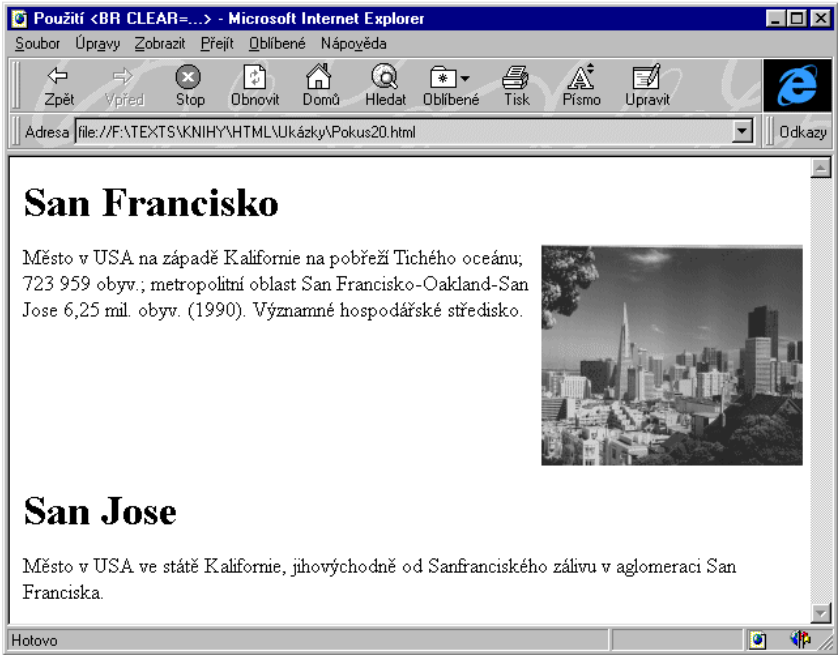

```
<H1>San Francisko</H1>
<IMG SRC="sf.gif" ALIGN=RIGHT>
Město v USA na západě Kalifornie na pobřeží Tichého oceánu; 723
959 obyv.; metropolitní oblast San Francisko-Oakland-San Jose
6,25 mil. obyv. (1990). Významné hospodářské středisko.

<BR CLEAR=RIGHT>

<H1>San Jose</H1>
Město v USA ve státě Kalifornie, jihovýchodně od Sanfranciského
zálivu v aglomeraci San Franciska.
```

Dalšími atributy, které mají při vkládání obrázků své využití, jsou atributy `WIDTH` a `HEIGHT`. Jejich hodnotou je požadovaná šířka a výška obrázku v pixelech. Pokud je při vkládání obrázku uvedeme, obrázek se zvětší či zmenší na požadovanou velikost.

Používání těchto atributů ke změně velikosti obrázku není příliš vhodné. Pokud obrázek zvětšíme, dostaneme většinou místo pěkného obrázku ošklivou zubatici. Pokud obrázek zmenšíme, dosáhneme často horších výsledků, než když obrázek zmenšíme v nějakém specializovaném programu jako je *Adobe PhotoShop*.

Nikdy nepoužíváme atributy `WIDTH` a `HEIGHT` ke zmenšení obrázku. Výsledkem je totiž malý a ne příliš kvalitní obrázek; přitom po síti se musel obrázek stahovat v původním rozlišení a tedy poměrně dlouho.





Obr. 4-3: Použití atributu CLEAR u elementu BR

I přesto však mají atributy WIDTH a HEIGHT uplatnění. Pokud jimi specifikujeme skutečné rozměry obrázku, může pro něj prohlížeč vyhradit místo již při načtení samotné stránky. Nemusíme tedy čekat na přenesení obrázků. Stránka je pak hned napoprvé zobrazena se správným layoutem a nemusí se několikrát zbytečně reformátovat a překreslovat.

Obrázek může být i odkazem, tj. kliknutí na něj vyvolá načtení jiné stránky. K dosažení tohoto efektu stačí jako text odkazu uvést obrázek:

```
<A HREF=«URL»><IMG SRC="moucha.gif" ALT="Obrázek mouchy"></A>
```

Obrázek, který slouží jako odkaz, má kolem sebe většinou zobrazen modrý rámeček. Šířka tohoto rámečku v pixelech je určena atributem BORDER elementu IMG. Pokud nám rámeček vadí, můžeme jeho zobrazení potlačit použitím BORDER=0:


```
<A HREF=«URL»><IMG SRC=moucha.gif ALT="Obrázek mouchy"
                   BORDER=0></A>
```

Všimněte si, že popis obrázku jako hodnota atributu je uzavřen do uvozovek (obsahuje totiž mezery a znaky s diakritickými znaménky). Oproti tomu název souboru s obrázkem a velikost rámečku do uvozovek uzavřeny být nemusí, protože se skládají pouze z písmen, číslic a tečky.

V této sekci se zmíníme ještě o dvou atributech. Jsou jimi HSPACE a VSPACE. Jejich hodnotou je počet pixelů, které určují velikost prázdného místa okolo obrázku. HSPACE udává přitom velikost tohoto prostoru vlevo a vpravo od obrázku a VSPACE nad a pod obrázkem.

☺ Pokud do stránky vložíme obrázek tak, aby byl zprava obtékán textem (ALIGN=LEFT) je dobré zároveň u obrázku nastavit hodnotu HSPACE na 5–10. Jinak je text na obrázek příliš připláclý (viz obr. 4-2 na straně 59).

Naše poznatky o vkládání obrázků můžeme shrnout do obecného tvaru elementu IMG:

```
<IMG SRC=«URL» ALT=«popis» ALIGN=«způsob zarovnání»
     WIDTH=«šířka» HEIGHT=«výška» BORDER=«šířka rámečku»
     HSPACE=«horizontální mezera» VSPACE=«vertikální mezera»>
```

4.2 Zrychlování přenosu obrázků

Pokud zařazujeme do naší stránky obrázky, musíme si uvědomit, že přenesení i malého obrázku ze serveru do prohlížeče zabere nejméně tolik času jako přenesení samotné stránky v HTML. Cílem každého autora je vytvářet stránky, které se budou zobrazovat co nejrychleji. To znamená, že i velikost obrázků by měla být co nejmenší. Ponecháme teď stranou výběr vhodného formátu GIF, JPEG nebo PNG a seznámíme se s některými obecnými metodami urychlujícími natažení obrázků.

Podívejme se nejprve na to, co určuje výslednou velikost obrázku. Ta je ovlivněna především dvěma faktory:

- *rozlišením obrázku*, které se většinou udává jako součin výšky a šířky obrázku v pixelech (např. 320×200);
- *barevnou hloubkou*, která udává počet bitů potřebných pro uchování informace o barvě jednoho pixelu — např. 8bitová barevná hloubka odpovídá 256barevným obrázkům ($2^8 = 256$).

Ponecháme-li stranou kompresi, která může velikost obrázku podstatně snížit, můžeme velikost obrázku spočítat jako *rozlišení* $\times$ *barevná hloubka*. Jelikož velikost obrázku je přímo úměrná rozlišení i barevné hloubce, vedou i cesty pro snížení velikosti obrázku přes snížení rozlišení a snížení barevné hloubky.

Při určování rozlišení obrázku si musíme uvědomit, že obrázek zařazený do stránky se zobrazuje tak, že jeden pixel obrázku odpovídá jednomu pixelu obrazovky. Obrázek tedy před zobrazením není nijak zmenšován.[1] Běžně používané zobrazovací režimy se pohybují od rozlišení 640×480 do $1\,024 \times 768$. Při návrhu stránky bychom měli vycházet z nejslabšího rozlišení, na kterém bude stránka prohlížena. Nemá tedy cenu na stránku umisťovat obrázek širší než asi 600 bodů. Tento obrázek navíc zabere v podstatě celou šířku okna prohlížeče.

Toto omezení si musíme uvědomit zejména pokud zařazujeme obrázky vzniklé naskenováním. Pokud při skenování nastavíme příliš velké rozlišení, dostaneme obrázek široký i vysoký několik tisíc pixelů.

Pro každý obrázek je proto dobré rozmyslet si, jak nejmenší ještě může být, aby neztratil svůj význam. Při snižování rozlišení totiž mizí z obrázku detaily a snižuje se vypovídací hodnota obrázku. Příliš malý obrázek také nemusí se stránkou tvořit estetický celek. Když konečně určíme minimální rozlišení, které ještě vyhoví našim požadavkům, stačí rozlišení obrázku snížit pomocí nějakého grafického editoru. V sharewarovém programu *Paint Shop Pro* k těmto účelům

[1] Pokud zmenšení nezařídíme pomocí atributů `WIDTH` a `HEIGHT`, což by — jak už víme — bylo fatální chybou.

slouží příkaz Image ▷ Resample. Pokud jsme obrázek kreslili v nějakém vektorovém kreslítku (*Corel Draw!*), je lepší obrázek znovu vyexportovat do GIFu v požadovaném rozlišení.

Snížením počtu barev lze dosáhnout také velkých úspor velikosti. Obrázek v TrueColoru (24bitová barevná hloubka, což odpovídá možnosti použít přes 16 milionů různých odstínů barev) je třikrát větší než 256barevný obrázek se stejným rozlišením. Pro jednoduché ikony vystačíme s pár barvami. Fotografie sice vypadají v TrueColoru nejlépe, ale snížení počtu barev někdy fotografii ani tolik neublíží.

☺ Osvědčenou metodou pro zjištění minimální použitelné barevné hloubky obrázku je její postupné snižování. Po určitém počtu kroku dostaneme obrázek, který je „ugly“ (česky odporný). Vrátíme se tedy k předchozí verzi a to je ta se správnou barevnou hloubkou. V *Paint Shop Pro* můžeme barevnou hloubku snižovat pomocí Colors ▷ Decrease Color Depth.

Existuje ještě jedna možnost, jak se vyhnout kompromisním řešením. Na stránku můžeme umístit malé obrázky (jakésy náhledy), které budou odkazy na původní pestrobarevný a velký obrázek. Uživatel si podle náhledu vybere obrázek, který jej zajímá. Stažení jeho kvalitnější verze už ho nebude iritovat tak, jak by jej mohlo iritovat stahování stránky se všemi obrázky v původní velikosti.

Původní obrázky můžeme ponechat pojmenované tak, jak byly. U náhledů stačí před jméno přidat nějaký ustálený prefix, abychom se v tom všem vyznali. Jako prefix se osvědčilo používat například podtržítko. Vložení našeho obrázku mouchy by pak vypadalo asi takto:

```
<A HREF="moucha.gif"><IMG SRC="_moucha.gif"></A>
```

Soubor `_moucha.gif` přitom bude mít menší rozlišení než `moucha.gif` a bude tedy také menší a jeho přenos po síti bude rychlejší.

Poslední způsob zefektivnění přenosu velkých obrázků po síti, který si ukážeme, je založen na tzv. *prokládání obrázků*. Prokládané obrázky nejsou v grafickém souboru uloženy řádek po řádku od shora. Nejprve je uložen každý osmý řádek počínaje řádkem 0, pak opět každý osmý počínaje čtvrtým řádkem, pak každý čtvrtý počínaje druhým řádkem a nakonec všechny liché řádky. Tím, že jsou data uložena takto, může být hrubý náhled obrázku zobrazen mnohem dříve než dorazí všechna data. Jistě jste sami zažili, že na některých stránkách se obrázky postupně vyhlazují a zaostřují až dosáhnou plné kvality zobrazení. Toho bylo dosaženo právě použitím prokládání obrázků.

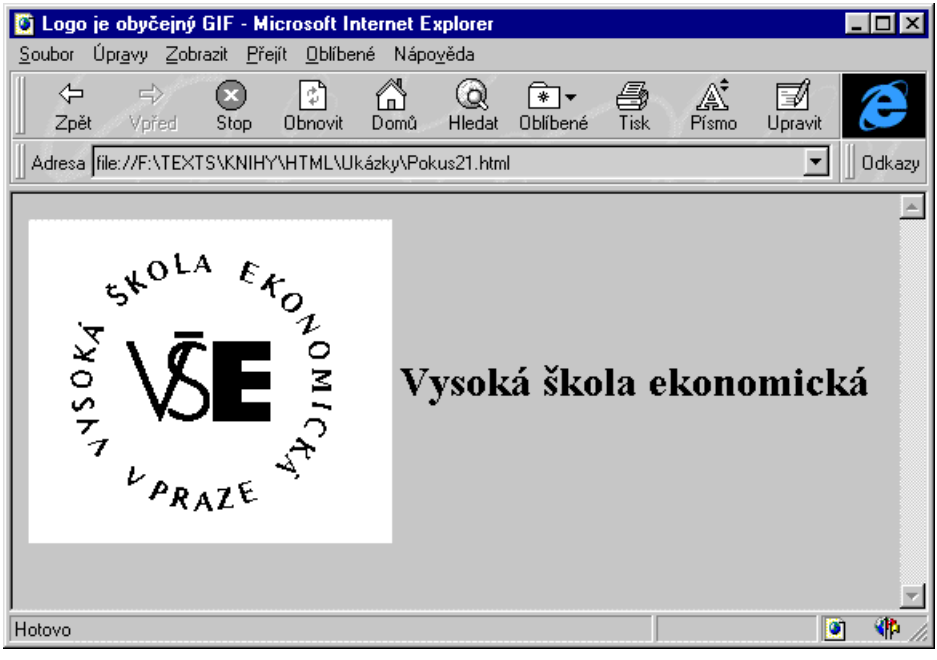
Výše popsaný způsob prokládání je podporován ve formátu GIF. Prokládání podporují i formáty PNG a JPEG. Jeho nastavení se vyplatí pouze u větších obrázků. U malých ikon, které mají pár kilobytů, je skoro zbytečné. To, že chceme obrázek uložit jako prokládaný, lze nastavit ve většině grafických editorů

při výběru ukládaného typu souboru. Téměř vždy nalezneme u formátu GIF volbu Interlaced (viz např. obrázek 4-6 na straně 67).

☺ Pokud na stránkách opakovaně používáme stejné obrázky (např. firemní loga apod.), můžeme zobrazení stránek rovněž urychlit. Obrázek, který se opakuje, uložíme pouze do jednoho adresáře (např. do `/images` nebo `/icons`). Ze stránek se pak na obrázek budeme vždy odkazovat pomocí příslušného relativního URL. Po prvním načtení zůstane obrázek ve vyrovnávací paměti prohlížeče a při jeho zařazení na dalších stránkách se již nebude stahovat ze sítě.

4.3 Transparentní obrázky

Velkou nevýhodou většiny grafických formátů je to, že tvar obrázku je omezen na obdélník. Budeme-li chtít na stránku vložit např. nějaké logo, které je kulaté, dočkáme se nepříjmeného efektu. Barva pozadí stránky může být v každém prohlížeči jiná, neshoduje se s pozadím obrázku a okolo obrázku se pak objeví nehezký obdélník (viz obr. 4-4).



Obr. 4-4: Když je barva pozadí stránky jiná než barva pozadí obrázku

Naštěstí lze tento problém vyřešit použitím speciální vlastnosti formátu GIF. Ten ve své verzi GIF89a (pochází z roku 1989) umožňuje definovat jednu barvu jako průhlednou (transparentní). Na místech, která mají tuto barvu, pak prosvítá pozadí stránky.

Jak v obrázku definovat transparentní barvu si ukážeme na příkladě obrázku loga VŠE. Logo budeme mít uloženo v souboru `VSElogo.gif`. K úpravám použijeme výborný sharewarový program *Paint Shop Pro*.[2] Obdobným způsobem jde transparentní barvu nastavit i v jiných grafických programech a editorech.

První krok spočívá ve výběru barvy, která bude transparentní. V našem případě je výběr jednoduchý. Logo je černé na bílém pozadí. My potřebujeme právě toto bílé pozadí vytvořit jako průhledné.

GIF podporuje maximálně 256barevné obrázky. V těchto obrázcích je barva jednotlivých bodů uchovávána nepřímo. Pro každý bod je zde uloženo 8bitové číslo, tzv. *index*. Tento index ukazuje do *palety*, což je tabulka, kde je pro každý index uložena barva v modelu RGB. Pro vytvoření transparentního obrázku potřebujeme znát právě index barvy, kterou chceme učinit průhlednou.

V *Paint Shop Pro* (dále jen PSP) zjistíme index barvy velice snadno. Nejprve otevřeme obrázek s logem pomocí File ▷ Open.... Na nástrojové liště stiskem ikony  vybereme nástroj Dropper (kapátko). Kapátkem najedeme v obrázku na tu barvu, kterou chceme mít transparentní. Na pravé straně okna PSP odečteme index této barvy (viz obr. 4-5 na následující straně). V našem případě je index 255.

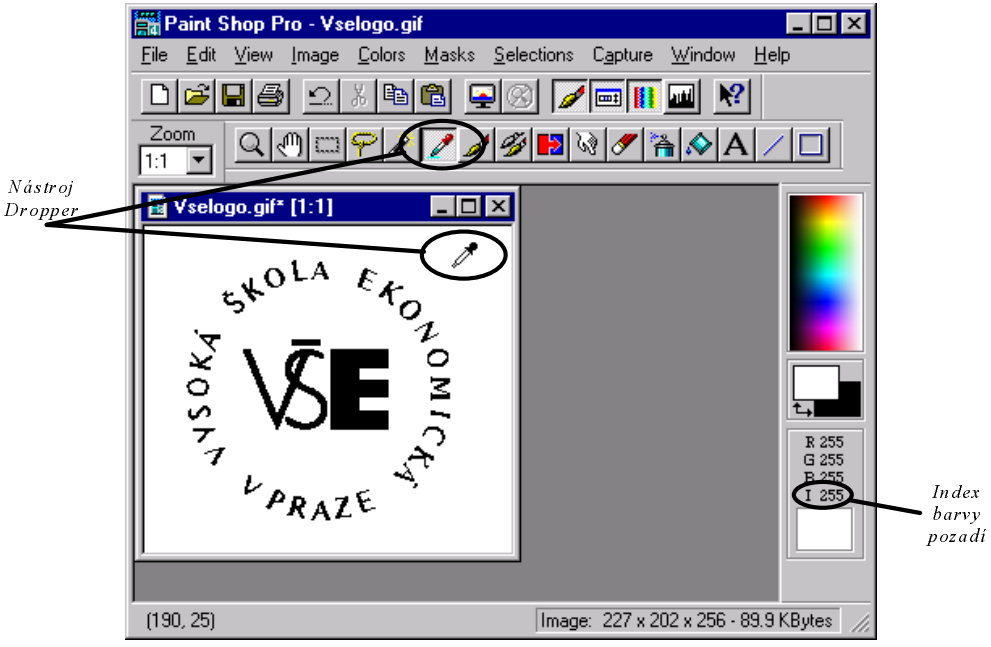
Když známe index transparentní barvy, stačí obrázek uložit a při ukládání nastavit transparenci. Vybereme proto z menu příkaz File ▷ Save As.... Jako formát samozřejmě vybereme GIF a podtyp Version 89a – Interlaced (viz obrázek 4-6 na následující straně).

Před uložením pomocí tlačítka Save ještě stiskneme tlačítko Options. V dialogovém okně nastavíme index transparentní barvy na hodnotu již dříve zjištěnou (obrázek 4-7 na straně 68). Potvrdíme stiskem OK a obrázek uložíme pomocí Save.

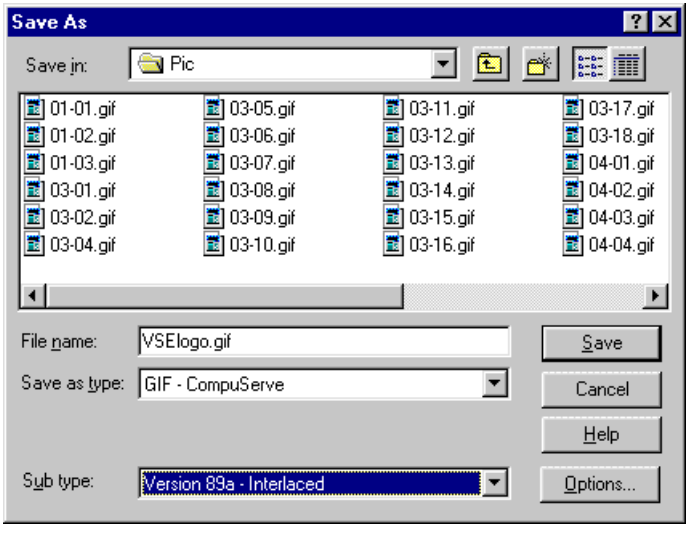
Když logo zařadíme na stránku pomocí následujícího zcela běžného zápisu, bude výsledek v prohlížeči již o poznání lepší (obrázek 4-8 na straně 68).

```
<H1><IMG SRC="VSElogo.gif" ALT="Logo VŠE" ALIGN=MIDDLE>
Vysoká škola ekonomická</H1>
```

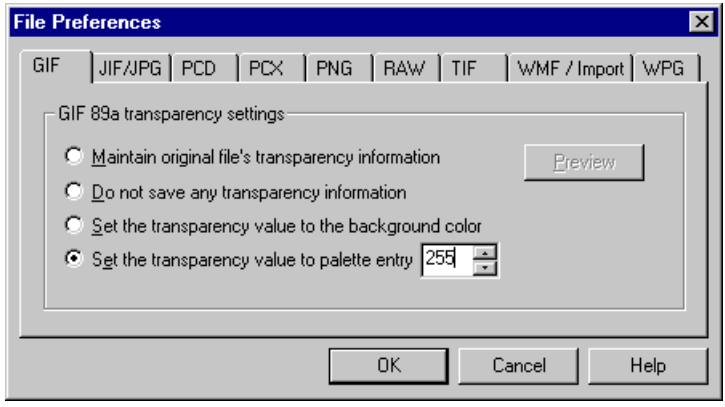
[2] Stáhnout si jej můžete na adrese `http://www.jasc.com/pspdl.html`.



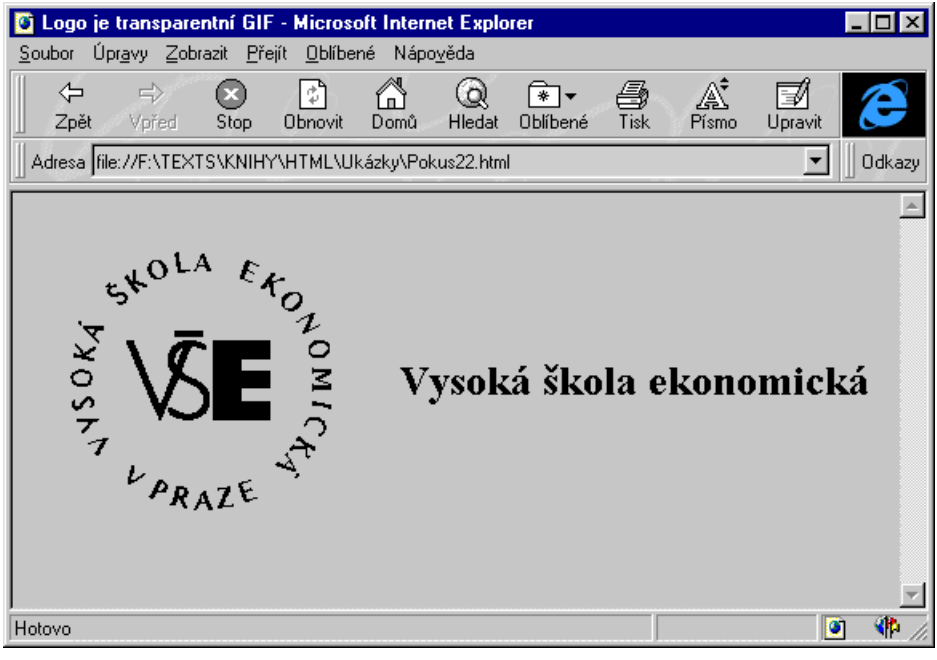
Obr. 4-5: Zjištění indexu barvy v Paint Shop Pro



Obr. 4-6: Nastavení správného typu souboru při ukládání



Obr. 4-7: Nastavení indexu transparentní barvy



Obr. 4-8: Jako transparentní GIF vypadá logo již mnohem lépe

✍ *Teď si uděláme malou vsuvku a testík. Jaké chyby jsme se dopustili při vytváření transparentního obrázku s logem VŠE?*[3]

Na obrázku obrazovky PSP (4-5 na straně 67) je vidět, že obrázek má 256 barev. Jelikož je ve skutečnosti použita pouze černá a bílá barva, měli jsme počet barev v obrázku snížit na 2 (barevná hloubka v tomto případě bude 1).

4.4 Animované obrázky

Animované obrázky jsou na webu poměrně novinkou, ale přesto se s nimi setkáme téměř na každém kroku. Proužková reklama na různých seznamech a vyhledávacích serverech, rotující čudlíčky místo normálních odrážek seznamů — to vše jsou animované obrázky.

Animovaný obrázek je speciální případ obrázku GIF, kdy je v jednom souboru uloženo několik obrázků. Tím, že se tyto obrázky postupně střídají na obrazovce, dochází ke vzniku pohybu vnímanému jako animace. Pro jednotlivé obrázky lze nastavit dobu, po které se změní, způsob přechodu mezi obrázky a samozřejmě i transparentní barvu.

K vytvoření animovaného GIFu musíme použít specializovaný program. Mezi takovéto programy patří *Microsoft GIF Animator* a *GIF Construction Set* firmy Alchemy Mindworks.

Oba umí uživatelem dodané obyčejné GIFy poskládat do animace a nastavit všechny potřebné parametry — dobu jednotlivých fází animace, způsob přechodu atd.

GIF Construction Set navíc nabízí možnost konverze souboru s videem ve formátu `.avi` na animovaný GIF. Rovněž obsahuje wizarda,[4] který umožňuje snadné vytvoření skrolujícího textu ve tvaru animovaného GIFu.

Při tvorbě animovaného obrázku máme v podstatě tři možnosti. Tou první je získat někde již hotový animovaný obrázek. Pokud na některé stránce v Internetu vidíme pěkný animovaný obrázek, technicky není problém jej stáhnout a použít na vlastních stránkách. Měli bychom se však vždy ujistit o tom, že tímto činem neporušujeme autorská práva.

Druhá možnost spočívá ve vytvoření animace v nějakém specializovaném programu na vytváření animací: *3D Studio Max*, *Autodesk Animator*, *SoftImage* atd. Tyto programy buď umí animaci přímo uložit jako animovaný GIF, anebo ji můžeme uložit jako soubor `.avi`. Ten následně pomocí *GIF Construction Set* převedeme do animovaného GIFu (převod nalezneme v menu pod File ▷ Movie to GIF).

[3] Správné řešení se dozvíte po otočení knihy vzhůru nohama.

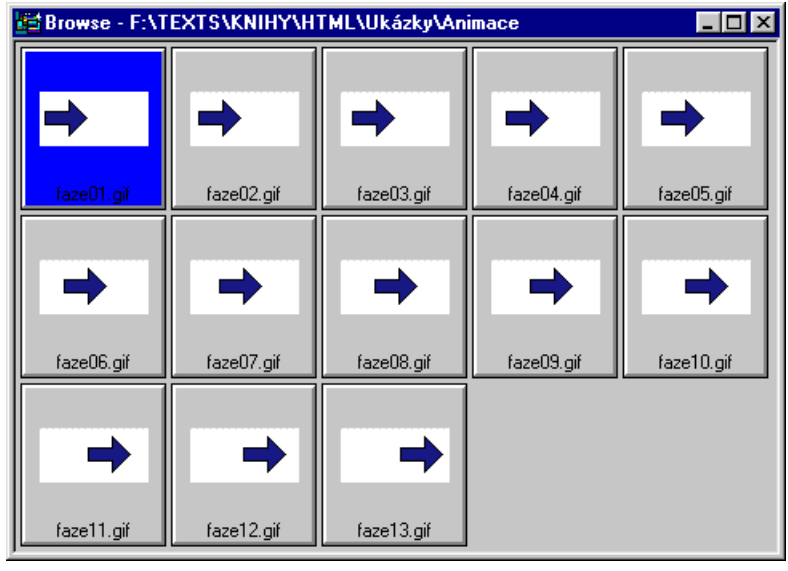
[4] Česky se tomuto nástroji říká kouzelník nebo šaman. Oficiální překlad zní průvodce.

Při generování souboru `.avi` musíme nastavit malé rozlišení, jinak dostaneme nepříjemně objemný animovaný GIF. V jednom animovaném GIFu je uloženo mnoho obyčejných GIFů. Jestliže obrázek jedné fáze animace zabere 10 KB a animace se skládá z 10 kroků, vznikne nám 100KB obluda.

Poslední možnost vytvoření animovaného obrázku je asi nejpracnější. Ručně rozkreslíme jednotlivé fáze animace a uložíme je do samostatných GIF obrázků. Pomocí *Microsoft GIF Animatoru* nebo *GIF Construction Set* poskládáme jednotlivé GIFy do výsledné animace.

My si ukážeme vytvoření jednoduché animované šipky. Šipka se bude pohybovat zleva doprava, na chvíli se zastaví a pak se bude celý pohyb opakovat. Využití nalezne v místech, kde budeme chtít upozornit na nějakou důležitou informaci na naší stránce.

Nejprve musíme vytvořit obrázky s jednotlivými fázemi animace. Výsledkem této nepříliš záživně a zdlouhavé práce je 13 obrázků s různě umístěnou šipkou (viz obr. 4-9).



Obr. 4-9: Jednotlivé fáze animace

Když máme animaci rozkreslenu, zbývá ji složit dohromady. My si tento postup ukážeme v prostředí programu *GIF Construction Set* (dále jen GCS).[5] V *Microsoft GIF Animatoru* lze sice získat stejný výsledek, ale bylo by to v našem případě 13krát pracnější. GCS totiž umožňuje nastavit parametry společně pro celou skupinu obrázků v animaci.

[5] GCS si můžete stáhnout na `http://www.mindworkshop.com/alchemy/alchemy.html`.

Spustíme GCS a vytvoříme nový animovaný obrázek pomocí File ▷ New. Do něj vložíme obrázky jednotlivých fází. Stiskneme tlačítko Insert a Image. Objeví se dialog pro výběr souboru. Nyní můžeme přidat jednotlivé fáze animace. Lze označit všechny obrázky najednou a přidat je v jediném kroku. Myší klikneme na soubor `faze01.gif`. Podržíme SHIFT a klikneme myší na `faze13.gif`. Označeny by měly být všechny fáze. Výběr potvrdíme stiskem tlačítka OK.

Měl by se objevit dialog, ve kterém vybereme způsob převodu palety vkládaných obrázků. Ponecháme standardní volbu Remap this image to global palette a zaškrtneme volbu Use this selection for subsequent images. Vše potvrdíme stiskem OK.

Po vložení obrázků musíme nastavit parametry animace. Parametry animace jsou uloženy v kontrolních blocích mezi jednotlivými fázemi. Nejsnazší postup pro přidání kontrolních bloků je stisk tlačítka Manage. V dialogovém okně, které se objeví, vybereme všechny fáze (stiskem tlačítka Select All) a stiskneme horní tlačítko Apply.

Objeví se dialog pro nastavení přechodu mezi jednotlivými obrázky (viz obrázek 4-10 na následující straně). Můžeme zde velice jednoduše nastavit transparentní barvu použitím kapátka. Do pole Delay vyplníme délku zobrazení jedné fáze animace v setinách sekundy. My zvolíme pět setin sekundy. Pokud používáme transparentní barvu, musíme ještě v poli Remove By vybrat položku Background. Kdybychom tak neučinili, jednotlivé fáze animace by se překreslovaly přes sebe. Nastavení potvrdíme stiskem OK. Ukončíme okénko Block Management a vrátíme se zpět na hlavní obrazovku programu.

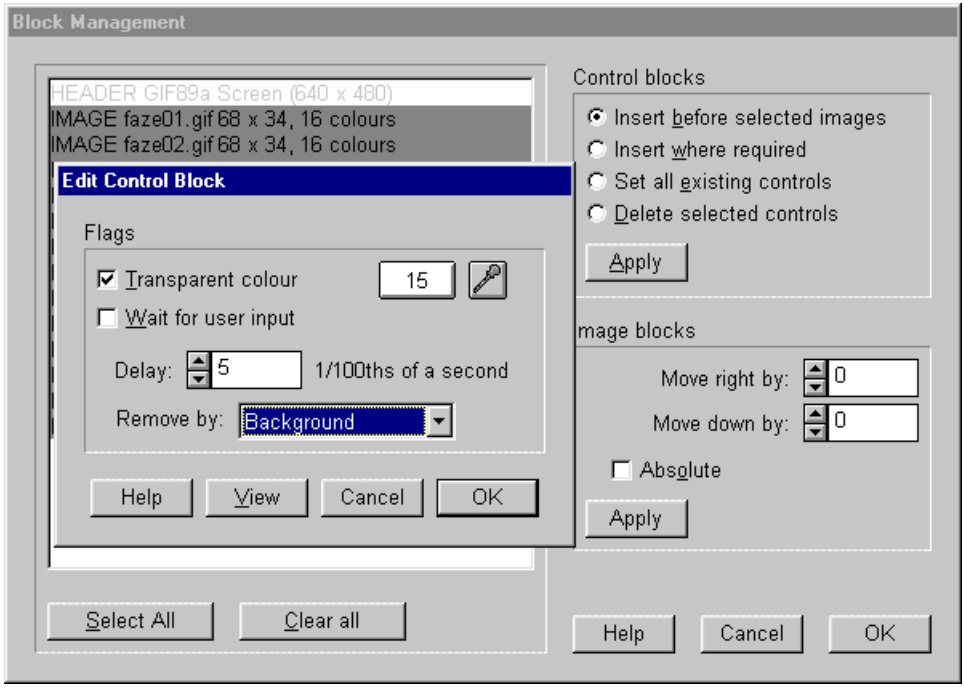
V seznamu vybereme poslední kontrolní blok a nastavíme u něj větší hodnotu Delay. Šipka pak zůstane v poslední fázi pohybu o něco déle.

Takto vytvořenou animaci můžeme vyzkoušet stiskem tlačítka View. Vše by mělo fungovat, animace však proběhne pouze jednou. My bychom chtěli, aby se animace opakovala pořád dokola. Stiskneme tedy tlačítko Insert a poté Loop. Nyní by již měla animace probíhat podle našich představ.

Před uložením animace do souboru zbývá pouze upravit rozměry animace tak, aby odpovídaly rozměrům jednotlivých fází. Rozměry obrázku se mění v první položce seznamu — HEADER (viz obr. 4-11 na straně 73).

4.5 Výběr vhodného grafického formátu

Předtím, než umístíme obrázek finálně na stránku, stojíme před rozhodnutím, který grafický formát použít. Dnes jsou široce podporovány dva formáty — GIF a JPEG. Podpora formátu PNG zatím není moc široká — údajně by jej měl podporovat *Internet Explorer 4.0*. Zatím jej podporuje pouze experimentální prohlížeč *Amaya* vyvíjený konsorciem W3C. Ostatní prohlížeče PNG podporují pouze pomocí plug-in modulů (obrázek pak ale nemůže být vkládán pomocí tagu `<IMG>`).



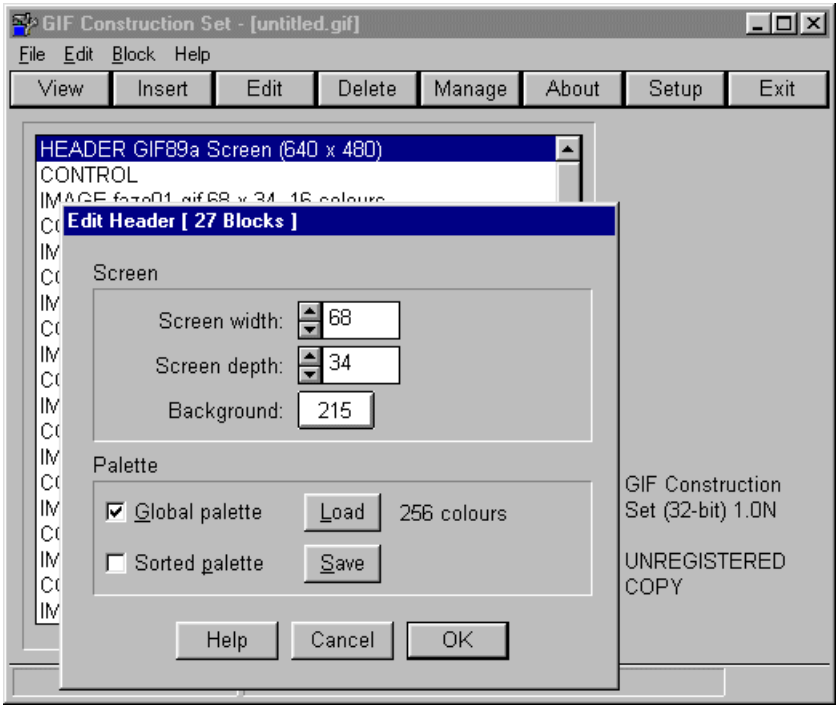
Obr. 4-10: Nastavení parametrů animace v GIF Construction Set

GIF

Formát GIF (Graphics Image Format) byl v 80. letech původně vyvinut pro použití v síti BBS. Byl tedy optimalizován pro přenos souborů pomocí pomalých modemů (modem 2 400 bit/s byl tehdy nejmodernější technologií). Tato optimalizace spočívá zejména v možnosti používat prokládané obrázky, které se mohou hrubě zobrazit ještě před přenesením celého obrázku.

Formát GIF samozřejmě podporuje kompresi. Komprese umožňuje různými metodami snížit velikost potřebnou pro uložení dat obrázku. Využívá se zejména toho, že v obrázku se často opakují stejné sekvence bodů. Tuto sekvenci stačí uložit pouze jednou a na místě jejího výskytu se pak ukládá pouze odkaz na tuto sekvenci. Tento odkaz je samozřejmě menší než původní sekvence a dojde tak ke snížení místa potřebného pro uložení obrázku.

V GIFu se používá kompresní algoritmus LZW. V době vzniku bylo používání tohoto algoritmu zcela volné. Na počátku devadesátých let však firma Unisys, majitel patentu na LZW, politiku zpřísnila. Používání algoritmu LZW je zdarma pouze v případě dekompresní rutiny. Pokud v nějakém programu použijeme kompresní rutinu (např. při ukládání obrázku ve formátu GIF), musíme zaplatit licenční poplatky. To se vývojářům samozřejmě nelíbilo, a tak vznikl prostor



Obr. 4-11: Nastavení velikosti animovaného obrázku

pro vývoj nového formátu, který by obsahoval kompresní metody neomezené nějakými licencemi. Tímto formátem je překvapivě PNG.

V době, kdy GIF vznikal, byly grafické adaptéry ještě v plenkách. A tak autoři formátu předpokládali, že 256 barev bude dostačujících. Dnes se zdá 256 barev skutečně málo, protože téměř každý provozuje *Windows* v režimu HiColor (65 tisíc barev) nebo TrueColor (16 miliónů barev). Omezení maximálního počtu barev na 256 bylo dalším z důvodů, proč vznikl nový formát PNG odstraňující toto omezení.

GIF se samozřejmě vyvíjel a v roce 1989 byla uvolněna jeho novější specifikace (GIF89a). Ta zavedla podporu transparentnosti, bez které bychom si dnes jen stěží představili kvalitní webovskou stránku.

GIF již od počátku umožňoval do jediného souboru uložit několik obrázků. Tato vlastnost nebyla příliš využívána. To se však změnilo s příchodem GIF89a. Ten do formátu přidal rozšíření, které umožnilo specifikovat dobu zobrazení a způsob přechodu mezi obrázky. Na světě byl animovaný GIF, který se začal masově používat na webovských stránkách.

GIF se vzhledem k jeho vlastnostem hodí zejména pro zobrazení různých ikon, obrázků nakreslených na počítači a pérovek. Uplatnění nalezne i v případech, kdy požadujeme transparenci nebo animaci.

JPEG

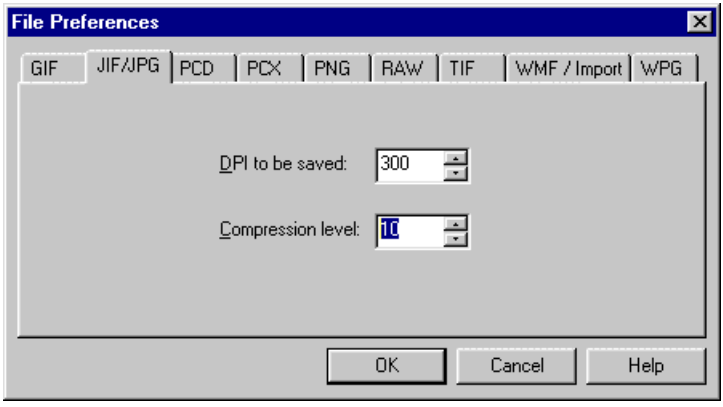
Hned ze začátku si neodpustím malé terminologické ujasnění názvu. JPEG není přímo grafickým formátem. JPEG je zkratka pro skupinu Joint Photographic Experts Group, která se zabývá vývojem algoritmů pro kompresi obrazových dat. Kompresní metoda, kterou vyvinuli, se nazývá JPEG. Tato metoda se používá v grafickém formátu JFIF (JPEG File Interchange Format). Tento formát definuje standardní způsob, kterým se data zkomprimovaná metodou JPEG poskládají do souboru. Správně bychom tedy měli hovořit o formátu JFIF, ale zažitý termín pro tento formát je JPEG. Budeme se jej tedy držet, abychom nevyvolávali zbytečné zmatky snahou o formální dokonalost.

Největší rozdíl mezi JPEGem a GIFem je v použitém způsobu komprese. GIF používá tzv. bezztrátovou kompresi. Zkomprimovaná a následně dekomprimovaná data jsou zcela shodná s originálem. JPEG naopak používá ztrátovou kompresi. Obrázek uložený ve formátu JPEG nemusí být zcela totožný s předlohou. Tyto změny jsou patrné zejména na ostrých přechodech jako jsou rovné hrany. Formát JPEG se proto nehodí pro běžné obrázky kreslené na počítači či vzniklé raytracingem. Obrázek pak obsahuje rušivé mapy. Naopak pokud JPEG aplikujeme na obrázek rozkvetlé louky, nějakou tu malou změnu ani nepoznáme — JPEG využívá nedokonalosti lidského zraku, který nerozezná jemnou změnu v odstínu barvy.

Vidíme, že JPEG se hodí pouze pro uchovávání obrázků vzniklých naskenováním fotografií. Pro tyto obrázky dosahuje vynikajících kompresních poměrů. Navíc lze u JPEG komprese nastavit tzv. Q-koeficient, který určuje kvalitu obrázku. Čím vyšší Q, tím je obrázek kvalitnější (menší odlišnosti od originálu) a větší. Naopak nižší Q dosahuje výrazně lepších kompresních poměrů při zhoršení kvality obrázku.

Novější prohlížeče umí pracovat i s tzv. progresivním JPEGem. Jedná se o upravený formát JPEG, kde je obrázek ukládán prokládaně podobně jako u GIFu. Grafické editory umožňují při ukládání obrázku ve formátu JPEG použít prokládání.

Formát JPEG se hodí pro umisťování fotografií na webovské stránky. Díky zaměření na fotografie podporuje pouze jedinou barevnou hloubku, a tou je režim TrueColor. Při přípravě obrázků ve formátu JPEG si můžeme vyhrát s parametrem Q a získat malý, ale ještě kvalitní obrázek.



Obr. 4-12: Nastavení míry komprese JPEGu $(100 - Q)$ *v PSP*

PNG

PNG (Portable Network Graphics) vznikl jako náhrada za formát GIF. GIF zůstal daleko pozadu za dnešními multimediálními aplikacemi, které pracují v režimu TrueColor. Navíc bylo použití kompresní metody GIFu licencováno.

Podrobnou specifikaci PNG můžeme nalézt v [6]. My si zde nastíníme alespoň základní vlastnosti PNG.

PNG přejímá od GIFu vše dobré a osvědčené. PNG podporuje transparentní i prokládané obrázky. U prokládaných obrázků je použita metoda, která ještě zrychlí zobrazení prvního náhledu.

PNG samozřejmě nabízí různá další rozšíření. Mezi ně patří podpora truecolorových obrázků s barevnou hloubkou 24 nebo 48 bitů a obrázků s odstíny šedi s 16bitovou barevnou hloubkou. U těchto druhů obrázků může být použit alfa-kanál. Alfa-kanál je v podstatě vylepšením transparentnosti. Pokud použijeme alfa-kanál, můžeme pro každý bod obrázku specifikovat jeho průhlednost plynule od 0 % (zcela neprůhledný) do 100 % (úplně transparentní). Této vlastnosti půjde využít k různým efektům zejména ve spojení s dynamickým HTML, o kterém si více povíme ve 14. kapitole.

Jediné, co PNG nemá oproti GIFu, jsou animované obrázky. PNG slouží k uložení pouze jednoho obrázku do jednoho souboru.

Formát PNG zatím není podporován všemi prohlížeči. V době, kdy čtete tuto knihu, však může být situace již zcela jiná. PNG se hodí všude tam, kde se hodil GIF. Navíc nalezne uplatnění u obrázků vzniklých ray-tracingem či jinou počítačovou vizualizací — zde se totiž převážně používá barevná hloubka TrueColor.

Tak tedy, který?

Z předchozích znalostí bychom již měli umět vybrat vhodný grafický formát pro naše obrázky. PNG se zatím raději vyhneme. Pro fotografie použijeme JPEG a pro vše ostatní GIF. Až se PNG více rozšíří, můžeme obrázky ve formátu GIF převést na PNG. PNG totiž dosahuje o něco lepší komprese než GIF. To neplatí pro ikony a malé obrázky, kdy výslednou velikost grafického souboru negativně ovlivní delší hlavička formátu PNG. V tabulce 4-1 přehledně porovnáváme vlastnosti jednotlivých grafických formátů.

	GIF	JPEG	PNG
Barevná hloubka	1–8	24	1–48
Maximální počet barev	256	16 miliónů	$2{,}8 \times 10^{14}$
Podpora prokládaných obrázků	ano	ano	ano
Transparentní obrázky	ano	ne	ano
Animované obrázky	ano	ne	ne
Komprese	bezztrátová	ztrátová	bezztrátová
Určení	ikony, obrázky nakreslené na počítači, pérovky	fotografie	jako GIF + obrázky vzniklé ray-tracingem a počítačovou vizualizací
Podpora	Netscape, Explorer, Amaya	Netscape, Explorer, Amaya	Explorer 4.0, Amaya

Tab. 4-1: Srovnání klíčových vlastností grafických formátů

4.6 Klikací mapy

Zatím jsme si ukázali, jak udělat z celého obrázku odkaz. Na Webu však často potkáme stránky, které obsahují obrázek s nějakými navigačními tlačítky či mapu. Kliknutí na různé části obrázku vyvolá různé odkazy. Obrázku, za jehož částmi se skrývají odkazy, říkáme *klikací mapa*.

Než se pustíme do tvorby klikací mapy, musíme si rozmyslet, k čemu ji budeme potřebovat. My si ukážeme vytvoření jednoduché navigační lišty pro pohyb po skupině dokumentů. Lišta by měla umožnit pohyb na předcházející a následující kapitolu, na obsah, na úvodní stránku a na stránku sloužící pro vyhledávání.

Nejprve nakreslíme navigační lištu v nějakém grafickém editoru. Výsledek bude záležet na našem kreslířském umu. Naše lišta sice není nic moc, ale pro demonstraci použití klikacích map úplně postačí:

Nyní se musíme rozhodnout, které části obrázku budou aktivní — budou sloužit jako odkazy a reagovat na kliknutí myši. V našem případě je volba jednoduchá: aktivní oblasti budou jednotlivá tlačítka. První tlačítko bude odkazem na stránku `kapitola2.html`, druhé na `obsah.html` a třetí na `kapitola4.html`. Tlačítko s domečkem povede na naši domovskou stránku. Dejme tomu, že se k ní dostaneme pomocí relativního URL `../index.html`. Poslední tlačítko povede na stránku nějaké vyhledávací služby. My využijeme službu Kompas a URL posledního odkazu tedy bude `http://kompas.seznam.cz/`.

Takto definovaným aktivním oblastem porozumí člověk, ale počítač asi těžko. Jemu musíme jednotlivé oblasti popsat geometricky. V HTML můžeme každou aktivní oblast popsat jako obdélník, kruh nebo mnohoúhelník. V našem případě vystačíme s obdélníky.

Každou mapu, kterou vytváříme, musíme nějak pojmenovat. Pomocí tohoto jména se pak spojí obrázek s příslušnou mapou. Definice mapy má následující tvar:

```
<MAP NAME=«jméno mapy»>
    «definice aktivních oblastí»
</MAP>
```

Každá aktivní oblast je definována pomocí elementu `AREA`:

```
<AREA HREF=«URL» ALT=«popis odkazu»
      SHAPE=«tvar oblasti» COORDS=«souřadnice oblasti»>
```

Tvar oblasti přitom může být `RECT` (obdélník), `CIRCLE` (kruh) nebo `POLY` (mnohoúhelník). U obdélníku obsahuje atribut `COORDS` čárkami oddělené souřadnice levého horního a pravého dolního rohu obdélníku. U kruhu jsou zde uvedeny souřadnice středu a poloměr. U mnohoúhelníku se postupně uvádí souřadnice jednotlivých vrcholů. Pokud nespecifikujeme tvar oblasti pomocí atributu `SHAPE`, předpokládá se, že souřadnice v `COORDS` popisují obdélník (`SHAPE=RECT`).

Atribut `HREF` slouží k zadání URL, které se vyvolá po kliknutí na určitou aktivní oblast. Atribut `ALT` je opět pomůckou pro prohlížeče pracující v textovém režimu — místo klikací mapy mohou zobrazit alespoň seznam odkazů s jejich popisem. Definice naší klikací mapy tedy vypadá takto:

```
<MAP NAME="mapa">
    <AREA HREF="kapitola2.html" ALT="Zpět"
          SHAPE=RECT COORDS="11,11,49,49">
```

```
    <AREA HREF="obsah.html" ALT="Obsah"
          SHAPE=RECT COORDS="63,11,102,49">
    <AREA HREF="kapitola4.html" ALT="Další"
          SHAPE=RECT COORDS="115,11,153,49">
    <AREA HREF="../index.html" ALT="Domů"
          COORDS="167,11,205,49">
    <AREA HREF="http://kompas.seznam.cz/" ALT="Hledej"
          COORDS="219,11,255,49">
</MAP>
```

Když máme definovánu mapu, zbývá do stránky vložit samotný obrázek klikací mapy. Obrázek se vkládá zcela standardně pomocí tagu `<IMG>`, pouze se pomocí atributu `USEMAP` určí mapa:


```
<IMG SRC=«URL obrázku» USEMAP="#«jméno mapy»">
```

Uvedený zápis předpokládá, že je s definicí mapy v jednom souboru. Teoreticky je možné načíst definici mapy z jiného souboru a u atributu `USEMAP` uvést úplné URL včetně fragmentu. Většina prohlížečů si s mapou v jiném souboru neporadí a proto bychom mapu měli umístit vždy do stejného souboru jako tag pro vložení klikací mapy. V našem případě bude vložení klikací mapy do stránky vypadat následovně:

```
<IMG SRC="lista.gif" ALT="Navigační lišta" USEMAP="#mapa">
```

Je jedno, zda je mapa nejdříve definována a pak teprve použita nebo naopak. Definici mapy však musíme vždy uvádět v elementu `BODY`.

Aby naše exkurze do klikacích map byla úplná, uvedeme ještě zbývající fakta. V hodnotě atributu `COORDS` mohou být kromě souřadnic uvedena čísla zakončená znakem procento (`%`). V tomto případě se souřadnice spočítá jako poměrná procentuální část z šířky resp. výšky obrázku.

Může se stát, že se definice jednotlivých aktivních oblastí překrývají. V tomto případě má vyšší prioritu dříve uvedená oblast.

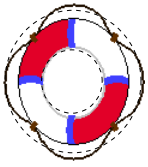
Místo atributu `HREF` můžeme použít `NOHREF`. V tomto případě již neuvádíme URL odkazu. `NOHREF` prohlížeči říká, že z uvedené oblasti nevede žádný odkaz.

Zkombinováním předchozích dvou vlastností můžeme vytvářet zajímavé klikací mapy. Pomocí oblasti s `NOHREF`, kterou uvedeme na začátku definice mapy, můžeme vytvořit neaktivní oblast uvnitř nějaké větší aktivní. Kdybychom třeba chtěli definovat aktivní oblast ve tvaru záchranného kruhu, můžeme to udělat následovně:

```
<MAP NAME="mapakruhu">
   <AREA NOHREF SHAPE=CIRCLE COORDS="100,100,30">
   <AREA HREF="http://www.baywatch.com/"
```

```
        SHAPE=CIRCLE COORDS="100,100,90">
</MAP>
```

První oblast, která je tvořena malým kroužkem, překryje druhou oblast (velký kruh — viz obr. 4-13). Výsledkem bude, že vnitřek záchranného kruhu nebude fungovat jako aktivní oblast.



Obr. 4-13: Přerušované čáry naznačují polohu oblastí

☺ Aktivní oblasti udělejte vždy trochu větší než grafický prvek, ke kterému se váží. Uživatel, který je unaven prací nebo omámen nějakou drogou, jen těžko pohybuje myší zcela přesně.

MapEdit

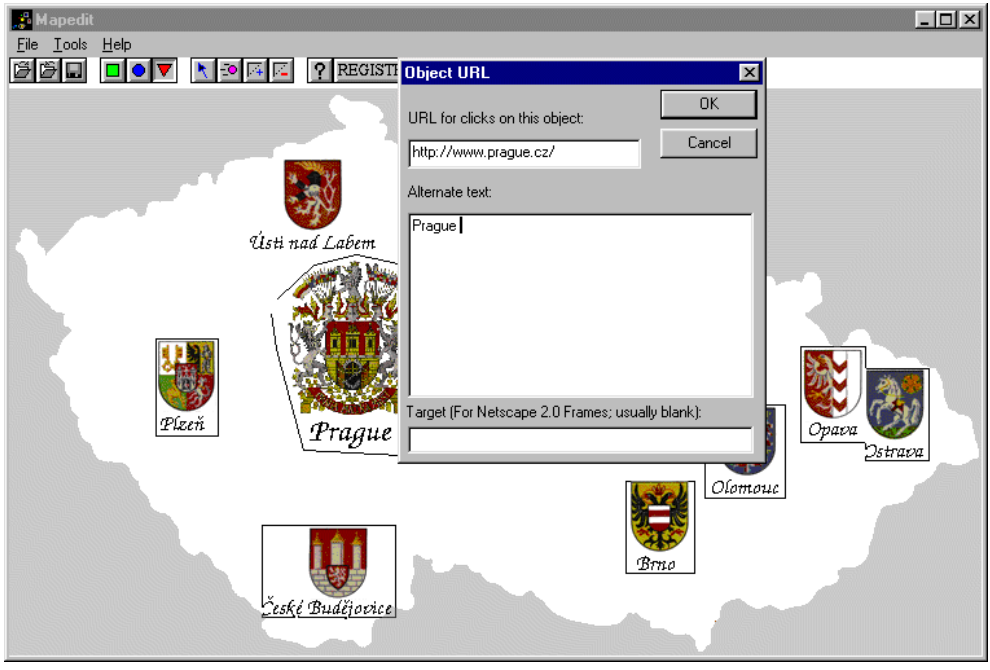
Ruční vytváření map aktivních oblastí je poměrně zdlouhavé. Usnadnit si jej lze pomocí programu *MapEdit*, který umožňuje pohodlné vizuální definování aktivních oblastí. Po spuštění programu vybereme stránku, která obsahuje obrázky. *MapEdit* nám nabídne výběr všech obrázků na stránce. Vybereme obrázek, pro který chceme vytvořit mapu. Přímo v obrázku pak můžeme myší definovat aktivní oblasti jednoho ze tří základních tvarů a pro každou oblast definovat i URL a alternativní text. Vytvořená mapa se pak uloží přímo do stránky. V programu lze upravovat i již hotové mapy. *MapEdit* si můžeme stáhnout na adrese `http://www.boutell.com/mapedit/`.

4.7 Bonus track

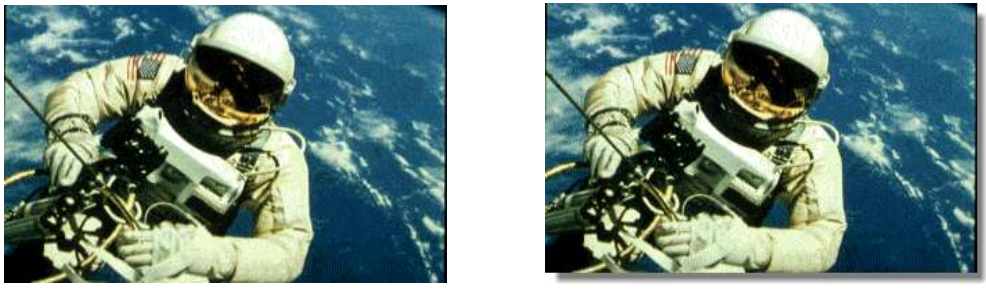
Tímto názvem bývají na CD-albech označeny skladby, které autoři přidali jaksi navíc pro potěšení posluchače. Podobný význam má i tato sekce. Nepřináší nic nového co se týče HTML, pouze obsahuje návod na vylepšení vzhledu obrázků na našich stránkách.

Obrázky, které obsahují fotografie, vypadají většinou mnohem profesionálněji, když jsou doplněny malým stínem. Rozdíl mezi obrázky se stínem a bez něj můžeme porovnat na obrázku 4-15 na následující straně.

Nyní si ukážeme, jak vyrobit u obrázku stín v nám už známém programu PSP. Nejprve si musíme zjistit rozměry obrázku, který chceme doplnit o stín. Náš



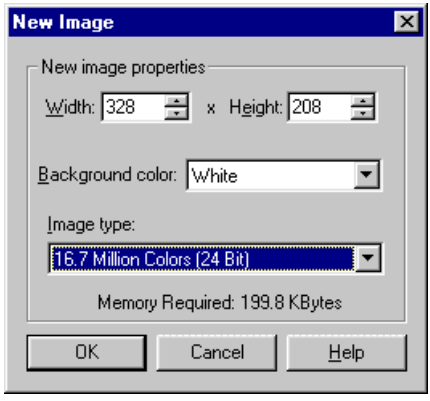
Obr. 4-14: Definice aktivní oblasti v MapEditu



Obr. 4-15: Stín dokáže fotografii opravdu vylepšit

obrázek s kosmonautem má rozměry 320×200. Nyní vytvoříme nový obrázek, který je v obou směrech o 8 bodů větší. V PSP tedy zvolíme příkaz File ▷ New. Barvu pozadí nastavíme na bílou a počet barev na 16 miliónů (viz obr. 4-16 na následující straně).

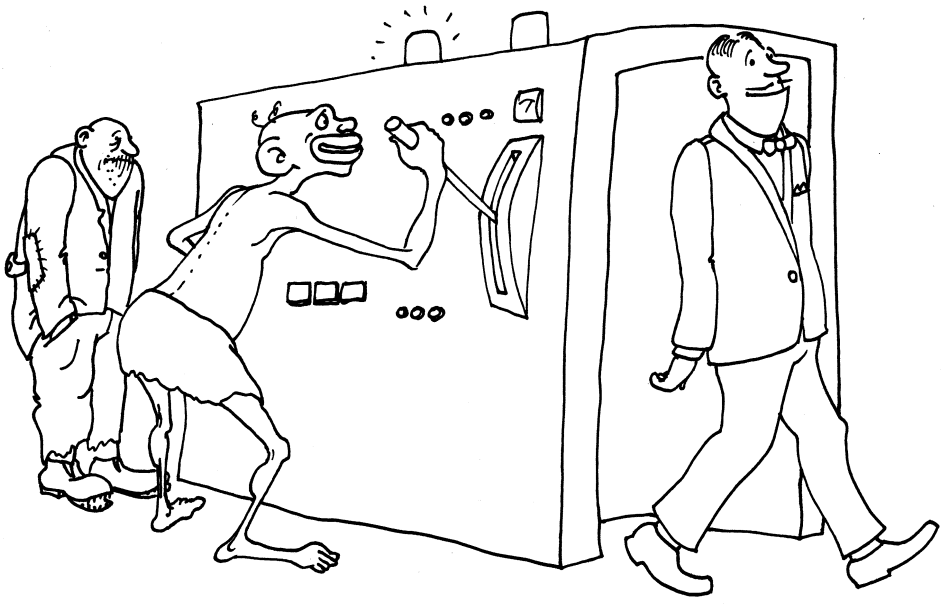
Doprostřed nově vzniklého obrázku nakreslíme černý nebo tmavě šedivý obdélník o rozměrech originálního obrázku, tj. 320×200. K nakreslení obdélníku použijeme nástroj . Nyní u obdélníku trošičku rozostříme hrany pomocí filtru Image ▷ Normal Filters ▷ Blur More.



Obr. 4-16: Nastavení parametrů nového obrázku

Obdélník, který bude tvořit stín, je potřeba posunout do pravého dolního rohu obrázku. Nejlépe je označit celý obrázek pomocí SHIFT + A a posunout ho myší.

Nyní stačí nad stín umístit původní obrázek. Zkopírujeme jej do schránky pomocí CTRL + C a do nového obrázku jej vložíme příkazem Edit ▷ Paste ▷ As New Selection (CTRL + E). Myší provedeme přesné umístění obrázku do levého horního rohu a náš výsledek se stínem uložíme. Stránky, kde jej budeme používat, by měly mít stejné pozadí, jaké jsme použili při vytváření stínu. V našem případě bylo pozadí bílé a proto barvu pozadí stránky nastavíme na bílou pomocí `<BODY BGCOLOR=WHITE>`.



5. Pokročilé formátování dokumentů

V této kapitole se naučíme přizpůsobit vzhled stránky našim grafickým požadavkům. Naučíme se určovat způsob zarovnání textu, měnit barvu textu i podkladu celé stránky a mnoho dalšího.

Několikrát jsme již naznačili, že HTML je značkovací jazyk. Jednotlivé značky určují pouze význam textu a nikoliv jeho vzhled. Tím dosáhneme dobré nezávislosti na platformě i na médiu, kde je HTML prezentováno. HTML dokument můžeme zobrazit na PC stejně tak dobře jako na pracovní stanici Silicon Graphics. Můžeme jej vytisknout na tiskárně nebo převést hlasovým syntetizátorem na řeč. Vidíme tedy, že prostor pro určování vizuálních atributů jako zarovnání zde opravdu není. Jak by třeba hlasový syntetizátor odlišil text zarovnaný doleva od textu centrovaného?

Přesto do HTML pronikly atributy a elementy umožňující lepší kontrolu nad grafickým vzhledem. Byl to důsledek rozšíření Webu do komerční sféry. Firmy chtěly své stránky vytvářet atraktivní a barevné a navíc chtěly, aby dokument vypadal ve všech prohlížečích stejně. Softwarové firmy vyvíjející prohlížeče jejich touhu uspokojovaly rozšířením HTML o možnost lepší kontroly nad formátováním. Velká část těchto rozšíření se stala součásti HTML 3.2 a tím se jejich používání zároveň standardizovalo.

5.1 Zarovnávání textu

Mnohé výzkumy uvádějí, že počítače jsou nejčastěji používány ke zpracování textu — tedy jako superinteligentní psací stroj. Můžeme tedy předpokládat, že každý čtenář této knihy pracoval někdy s nějakým textovým procesorem. Jednou ze základních funkcí těchto programů je možnost měnit způsob zarovnání odstavce. Většinou jsou k dispozici tři možnosti: zarovnání doleva, doprava či na střed.[1]

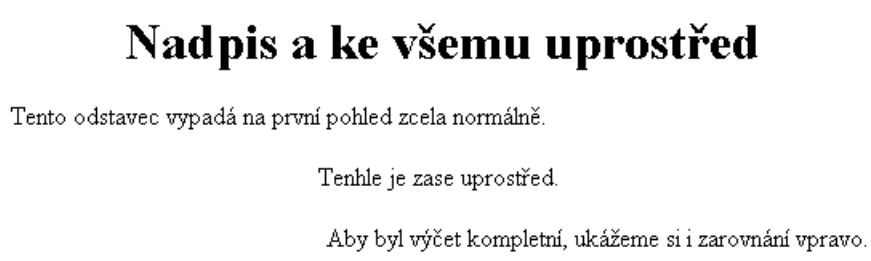
V HTML lze způsob zarovnání ovlivňovat pomocí atributu `ALIGN`. Ten může nabývat tří hodnot: `LEFT`, `RIGHT` a `CENTER`. Atribut lze použít u elementů pro nadpisy (`H1`–`H6`) a odstavce (`P`). Pokud jej neuvedeme, text se bude zarovnávat vlevo (`ALIGN=LEFT`). K vycentrování nadpisu stránky stačí opravdu málo:

```
<H1 ALIGN=CENTER>Nadpis a ke všemu uprostřed</H1>
```

Použití u odstavců je zcela obdobné:

[1] Typograf by vám uvedené tři způsoby pojmenoval jako zarovnání na levý praporek, na pravý praporek a centrování.

```
<P ALIGN=LEFT>Tento odstavec vypadá na první pohled zcela
normálně.
<P ALIGN=CENTER>Tenhle je zase uprostřed.
<P ALIGN=RIGHT>Aby byl výčet kompletní, ukážeme si i zarovnání
vpravo.
```



To, že můžeme měnit zarovnání pro jednotlivé odstavce, je pěkné. V případě, že bychom takto chtěli měnit zarovnání pro více odstavců nebo pro celý dokument, se však upíšeme k smrti. U každého tagu `<P>` budeme ručně přidávat atribut `ALIGN`. Navíc, když budeme chtít zarovnání změnit, budeme ho muset upravit u každého odstavce. Tyto problémy nám pomůže vyřešit element `DIV`. Do tohoto elementu můžeme uzavřít libovolně dlouhý text a nastavit mu některé společné atributy, jako je právě způsob zarovnání. Budeme-li chtít do stránky zařadit tři odstavce zarovnané doprava, můžeme použít tuto konstrukci:

```
<DIV ALIGN=RIGHT>
<P>První odstavec...
<P>Druhý odstavec...
<P>Třetí odstavec...
</DIV>
```

Nesmíme zapomenout na to, že `<DIV>` je potřeba uzavřít použitím párového tagu `</DIV>`. Když na to zapomeneme, bude nastaveným zarovnáním postižen zbytek dokumentu a navíc dokument nebude korektním HTML dokumentem.

Pokud uvnitř elementu `DIV` použijeme u elementů `P` nebo `H1`–`H6` atribut `ALIGN`, bude se odstavec resp. nadpis zarovnávat podle něj.

☺ Místo `<DIV ALIGN=CENTER>...</DIV>` můžeme použít kratší `<CENTER>...</CENTER>`. Element `CENTER` byl původně rozšířením HTML od firmy Netscape. Jeho používání se však tak rozšířilo, že byl zařazen do HTML 3.2 i přesto, že element `DIV` s patřičným atributem nabízí zcela stejnou funkčnost.

Atribut `ALIGN` je použitelný i u mnoha dalších elementů. Upozorníme na to při výkladu o nich.

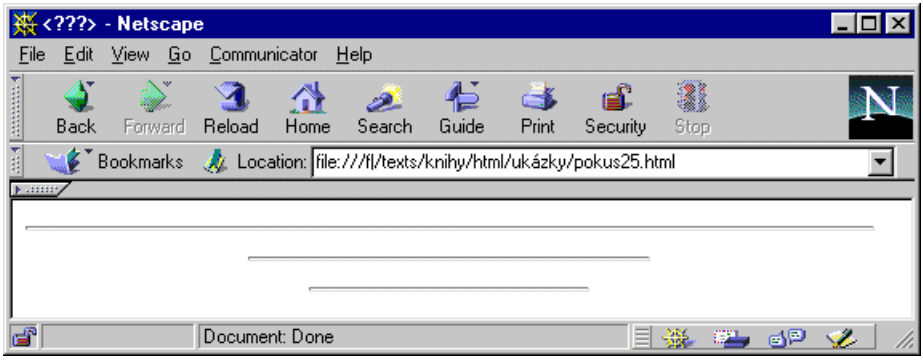
5.2 Optické členění dokumentu

Již víme, že k oddělení jednotlivých částí stránky můžeme použít `<HR>`. Na místě tohoto tagu se zobrazí čára přes celou šířku okna prohlížeče. Vlastnosti této čáry lze měnit pomocí několika atributů.

Atributem `WIDTH` můžeme měnit šířku čáry. Hodnotou atributu může být buď šířka udaná přímo v pixelech nebo procento. Procento udává šířku čáry jako poměrnou část šířky okna prohlížeče. Použití si hned ukážeme:

```
<HR>
<HR WIDTH=250>
<HR WIDTH="33%">
```

Na obrázku 5-1 vidíme, že poslední čára zabírá opravdu třetinu okna prohlížeče.



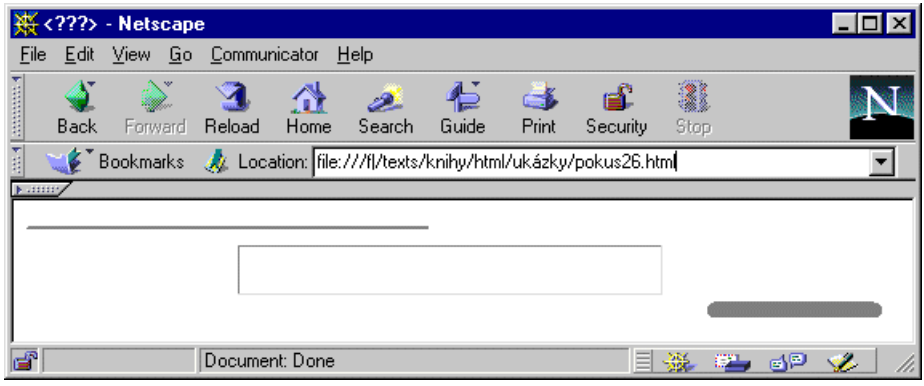
Obr. 5-1: Ovlivnění šířky čáry pomocí atribut WIDTH

Čára je normálně umisťována na střed stránky. Toto zarovnání můžeme změnit pomocí atributu `ALIGN` s obvyklým významem. Pokud se nám nelíbí, že jsou čáry zobrazeny se stínem, můžeme použít atribut `NOSHADE`.

Poslední atribut, který lze u `<HR>` použít, je `SIZE`. Ten určuje výšku čáry. Výška se zadává jako číslo vyjadřující počet pixelů.

Na obrázku 5-2 na následující straně si můžeme prohlédnou výsledek použití různých kombinací atributů:

```
<HR WIDTH=250 ALIGN=LEFT NOSHADE>
<HR WIDTH="50%" SIZE=30>
<HR NOSHADE SIZE=10 ALIGN=RIGHT WIDTH=100>
```



Obr. 5-2: HR s různými atributy

5.3 Seznamy

V druhé kapitole jsme se naučili pomocí elementů `OL`, `UL` a `LI` vytvářet jednoduché seznamy. Teď si ukážeme, jak lze pomocí atributů měnit vzhled seznamů.

Začneme s nečíslovaným seznamem `UL`. U elementu `UL` lze použít atribut `TYPE`. Ten ovlivňuje typ odrážky (puntíku), která je zobrazena vlevo před jednotlivými položkami seznamu. Na výběr máme ze tří druhů odrážek: `DISC` (plné kolečko), `SQUARE` (čtvereček) a `CIRCLE` (kroužek). Pokud atribut použijeme u `UL`, bude daný druh odrážky použit pro celý seznam. Pokud uvedeme `TYPE` u konkrétní položky seznamu `LI`, použije se daný druh od této položky do konce seznamu. Malá ukázka:

```
<UL>
    <LI TYPE=DISC>První
    <LI TYPE=SQUARE>Druhý
    <LI TYPE=CIRCLE>Třetí
</UL>

<UL TYPE=SQUARE>
    <LI>Praha
    <LI>Hradec Králové
    <LI TYPE=CIRCLE>Brno
    <LI>Bratislava
</UL>
```

- První
- Druhý
- Třetí

- Praha
- Hradec Králové
- Brno
- Bratislava

Atribut `TYPE` lze použít i u číslovaných seznamů (`OL`), zde však určuje způsob číslování. Význam jednotlivých hodnot atributu v tomto případě uvádí tabulka 5-1.

Hodnota TYPE	Způsob číslování	Ukázka
`1`	arabské číslice	1, 2, 3, 4, …
`a`	malá písmena	a, b, c, d, …
`A`	velká písmena	A, B, C, D, …
`i`	malé římské číslice	i, ii, iii, iv, …
`I`	velké římské číslice	I, II, III, IV, …

Tab. 5-1: Význam atributu TYPE u číslovaných seznamů

U tagu `<OL>` lze použít i atribut `START`, který určuje hodnotu první odrážky. Hodnota odrážky může být měněna i průběžně pomocí atributu `VALUE` u elementu `LI`. Použití atributů u číslovaných seznamů si shrneme na následující ukázce:

```
<OL TYPE=I START=3>
    <LI>Svačina
    <LI>Spacák
    <LI VALUE=10>Nůž
    <LI TYPE=1>Mapa
    <LI>Dobrá nálada
</OL>
```

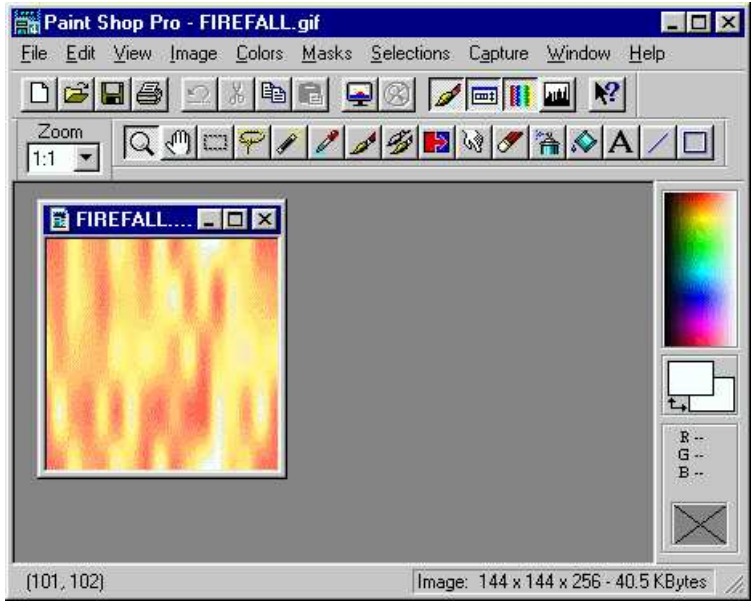
III. Svačina
IV. Spacák
X. Nůž
11. Mapa
12. Dobrá nálada

U tagů `<UL>`, `<OL>` a `<DL>` lze použít atribut `COMPACT`. Jeho uvedení by mělo způsobit kompaktnější zobrazení seznamu — s menšími mezerami mezi jednotlivými položkami. Praxe ukazuje, že ve většině prohlížečů se seznam zobrazí nezávisle na hodnotě tohoto atributu.

5.4 Pozadí stránky

Pokud chceme naši stránku zpestřit nebo učinit zajímavější, můžeme místo jednobarevného pozadí stránky určit obrázek, který se použije jako podklad pod celou stránkou. URL obrázku se zapisuje pomocí atributu `BACKGROUD` elementu `BODY`.

Většinou se jako pozadí stránky používá relativně malý obrázek, kterým se podle potřeby „vykachlíčkuje“ celá plocha pod stránkou. Je proto záhodno, aby obrázek navazoval sám na sebe a nevznikaly tak nepříjemné přechody mezi jednotlivými opakováními obrázku. Na obrázku 5-3 si můžeme prohlédnout obrázek, který za chvilku použijeme jako podklad naší stránky.

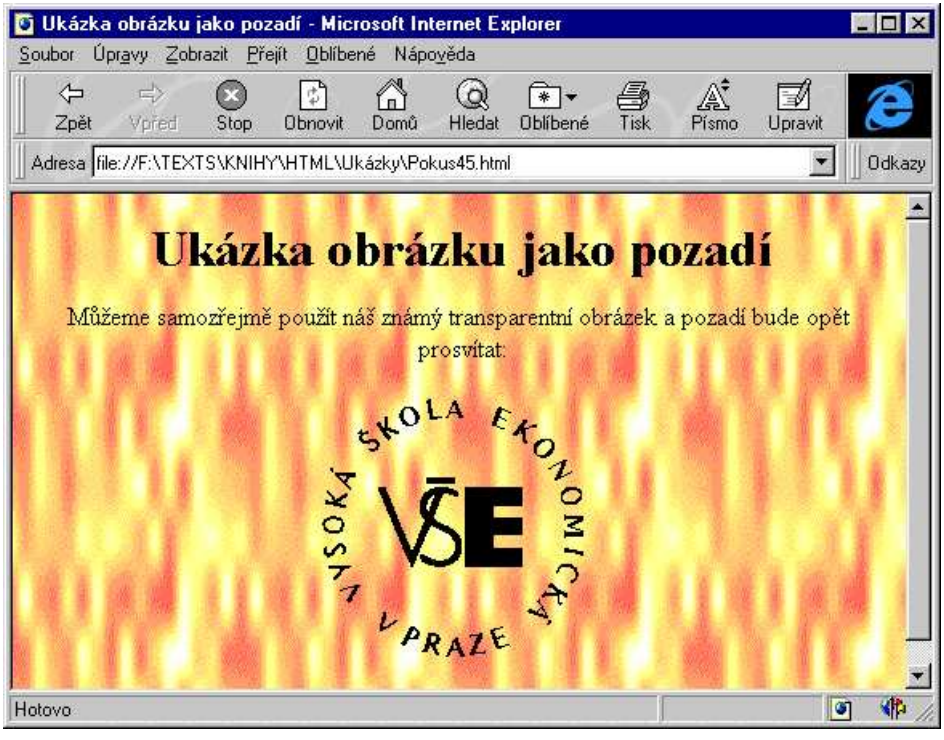


Obr. 5-3: Obrázek, který svým opakováním vytvoří ohnivé pozadí stránky

U všech obrázků, které používáme jako podklad stránky, je důležité, aby nebyly příliš kontrastní. Musíme si vždy ověřit, že samotný text stránky je proti pozadí dobře čitelný. Když máme vhodný obrázek, nic nebrání tomu vytvořit stránku s pěkným pozadím:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">
<HTML>
<HEAD>
    <TITLE>Ukázka obrázku jako pozadí</TITLE>
</HEAD>
```

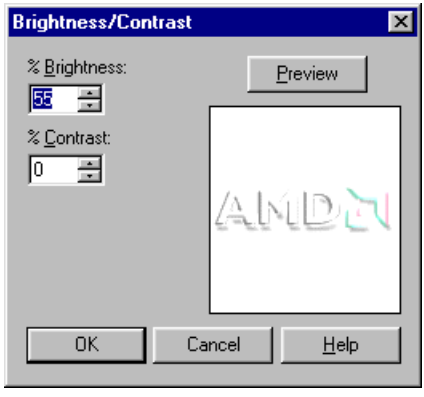
```
<BODY BACKGROUND="firefall.gif">
<DIV ALIGN=CENTER>
<H1>Ukázka obrázku jako pozadí</H1>
Můžeme samozřejmě použít náš známý transparentní obrázek
a pozadí bude opět prosvítat:<BR>
<IMG SRC="vselogo.gif">
</DIV>
</BODY>
</HTML>
```



Obr. 5-4: Stránka s obrázkem místo normálního pozadí

Obr. 5-5: Postupná úprava loga pro použití jako pozadí

Poměrně často se stává, že firma chce mít jako podklad svých stránek vlastní logo. Abychom zachovali čitelnost stránky oproti pozadí, je většinou potřeba logo upravit. My si postup ukážeme na logu firmy AMD (obrázek 5-6). Osvědčenou metodou je nejprve vytvoření jakéhosi reliéfu původního loga. K tomu nám v PSP výborně poslouží filtr Emboss. Před tím než jej aplikujeme, musíme obrázek převést do barevné hloubky TrueColor. K tomu nám poslouží příkaz Colors ▷ Increase Color Depth ▷ 16 Million Colors. Poté aplikujeme filtr Image ▷ ▷ Spacial Filters ▷ Emboss. U takto upraveného obrázku ještě upravíme jas a kontrast. Z menu vyvoláme funkci Colors ▷ Adjust ▷ Brightness/Contrast. V dialogu se pokoušíme nastavit co nejvyšší jas a co nejmenší kontrast tak, aby bylo logo ještě rozeznatelné. Když jsme s obrázkem spokojeni, stačí jej uložit a použít na stránkách.



Obr. 5-6: Nastavení jasu a kontrastu

5.5 Barvy

Dobrá zpráva: HTML umožňuje vykročit našim stránkám z šedého průměru. Můžeme měnit barvu pozadí, barvu textu a barvu odkazů. Ke změně barvy slouží několik atributů, které lze aplikovat na některé elementy. Než se však pustíme do konkrétních ukázek změny barvy dokumentu, musíme se seznámit se způsobem definování barev v HTML.

V HTML lze použít dva způsoby určení barvy. Jednak HTML obsahuje několik předdefinovaných barev, které mají své jméno. Pro červenou barvu máme v HTML jméno `red` pro bílou zase `white`.

Druhou možností definice barvy je její popsání RGB-modelem. Pro pochopení tohoto modelu budeme muset zavzpomínat na školní škamna a hodiny fyziky. Tam nás učili, že složením červené, zelené a modré barvy v různých poměrech lze získat i odstíny dalších barev. Chceme-li tedy získat žlutou barvu,

stačí ve stejném poměru namíchat červenou a zelenou barvu. Pro konkrétní barvu tedy potřebujeme rozhodnout, jak se na ní podílí jednotlivé základní barvy. V HTML lze podíl každé základní barvy určit číslem od 0 do 255. Hodnota 255 odpovídá maximální intenzitě barvy a 0 naopak minimální intenzitě. Vezmeme-li tedy červenou, zelenou i modrou v intenzitách 255, dostaneme barvu bílou. Pokud bude intenzita všech základních barev nulová, výsledná barva bude černá.

Barvu, kterou potřebujeme získat, tedy můžeme vyjádřit jako trojici čísel, jejíž jednotlivé členy odpovídají intenzitě červené, zelené a modré složky barvy. V HTML se pak barva zapisuje jako `#ččzzmm`, kde `čč` je hexadecimální zápis intenzity červené složky, `zz` je hexadecimální zápis intenzity zelené složky a `mm` je hexadecimální zápis intenzity modré složky. Červenou barvu tedy můžeme zapsat buď jako `red` nebo jako `#ff0000`. Tabulka 5-2 obsahuje přehled předdefinovaných barev společně s jejich vyjádřením v RGB notaci.

Název barvy	RGB-notace	Český název
AQUA	#00FFFF	jasně modrozelená
BLACK	#000000	černá
BLUE	#0000FF	modrá
FUCHSIA	#FF00FF	anilínová červeň
GRAY	#808080	šedivá
GREEN	#008000	zelená
LIME	#00FF00	citrónově zelená
MAROON	#800000	kaštanová
NAVY	#000080	tmavá modř
OLIVE	#808000	olivová
PURPLE	#800080	purpurová
RED	#FF0000	červená
SILVER	#C0C0C0	stříbrná
TEAL	#008000	tmavá modrozelená
WHITE	#FFFFFF	bílá
YELLOW	#FFFF00	žlutá

Tab. 5-2: Předdefinované barvy a jejich RGB-notace

Jelikož se barvy zapisují jako hodnoty atributů, nezáleží na velikosti písmen. Červenou barvu dostaneme tedy jako `red`, `RED`, `#FF0000` i `#ff0000`.

Barvy pro celý dokument

Většina z nastavení barev platí pro celý dokument. Proto nás nepřekvapí, že tato nastavení provádíme pomocí atributů elementu `BODY`. Co tedy můžeme měnit?

Jako první se nabízí barva pozadí celé stránky. Tu můžeme určit pomocí atributu `BGCOLOR`. Jako hodnotu uvedeme buď jednu z předdefinovaných barev, nebo si namícháme barvu vlastní. Budeme-li tedy vytvářet stránku pro americké námořnictvo, můžeme použít modrý podklad. K realizaci tohoto záměru stačí na začátku stránky místo `<BODY>` použít `<BODY BGCOLOR=NAVY>` nebo `<BODY BGCOLOR="#000080">`.

Pokud zároveň použijeme obrázek na pozadí (`BACKGROUND`*) a barvu pozadí (*`BGCOLOR`*), nejprve se vykreslí pozadí barvou určenou v* `BGCOLOR`*. Přes toto pozadí se podle potřeby zopakuje obrázek určený v* `BACKGROUND`*. Pokud bude mít obrázek nastavenu transparentnost, barva pozadí* `BGCOLOR` *bude obrázkem prosvítat.*

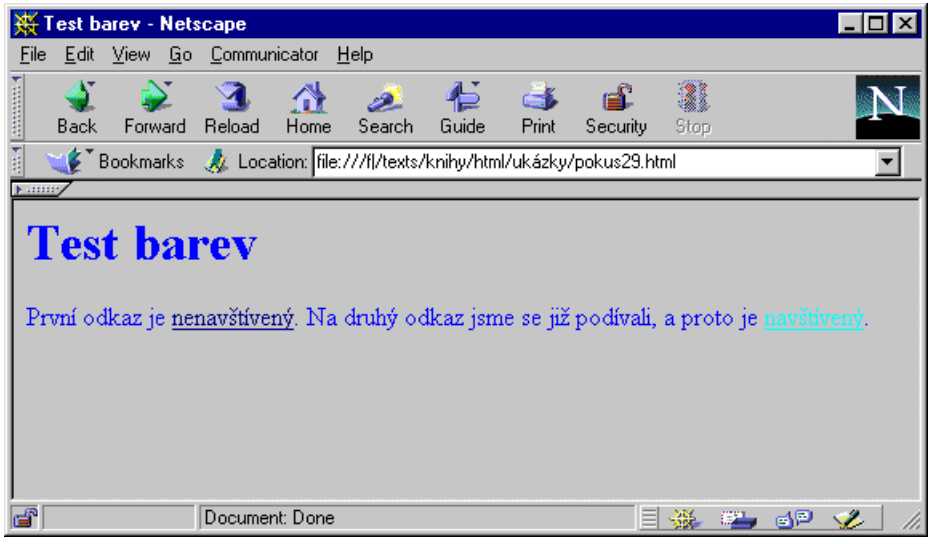
Samozřejmě, že obdobným způsobem lze nastavit barvu textu. K tomu slouží atribut `TEXT`. Pokud bychom chtěli používat bílý text na černém pozadí stránky, můžeme použít `<BODY BGCOLOR=BLACK TEXT=WHITE>`.

Když měníme barvu pozadí a textu, měli bychom volit kontrastní dvojice barev, aby byl text na pozadí dobře čitelný.

Kromě barvy textu můžeme ovlivnit i barvu odkazů. U odkazů lze barvu nastavovat zvlášť pro nenavštívené odkazy (atribut `LINK`), pro navštívené odkazy (atribut `VLINK`) a pro právě aktivovaný odkaz (atribut `ALINK`):

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">
<HTML>
<HEAD>
    <TITLE>Test barev</TITLE>
</HEAD>
<BODY BGCOLOR="#C0C0C0" TEXT=BLUE LINK=NAVY ALINK=YELLOW
      VLINK=AQUA>

<H1>Test barev</H1>
První odkaz je <A HREF="xxx">nenavštívený</A>. Na druhý odkaz
jsme se již podívali, a proto je <A HREF="yyy">navštívený</A>.
</BODY>
</HTML>
```

Jak změna barev dopadne v prohlížeči ukazuje obrázek 5-7 na následující straně. (Doufám, že na černobílém obrázku vidíte ty správné barvy.)



Obr. 5-7: Autorem definované barvy dokumentu

Barva pouze pro část textu

Pokud chceme barevně zvýraznit pouze určitou část textu, můžeme ji uzavřít do elementu `FONT`. Můžeme pak použít atribut `COLOR` k určení barvy označeného textu. Element `FONT` můžeme použít ve stejných případech jako elementy pro změnu stylu písma (`B`, `I`, `STRONG` apod.). Budeme-li chtít vyvést nadpis stránky v červené, stačí použít konstrukci:

```
<H1><FONT COLOR=RED>Červený nadpis</FONT></H1>
```

Opět musíme dávat pozor, aby se nám jednotlivé elementy nekřížily. Často se setkáme na Webu se zápisem

```
<H1><FONT COLOR=RED>Červený nadpis</H1></FONT>
```

který je zcela chybný.

5.6 Velikost písma

Pomocí atributů elementu `FONT` lze kromě barvy měnit i velikost písma. Slouží k tomu atribut `SIZE`. Jako jeho hodnotu můžeme uvést buď absolutní nebo relativní velikost písma.

Absolutní velikost písma je číslo od jedné do sedmi. Vyšší číslo odpovídá větší velikosti. Ukázku jednotlivých velikostí písma si můžeme prohlédnout na obrázku 5-8 na straně 95. Přiložíme i zdrojový kód stránky.

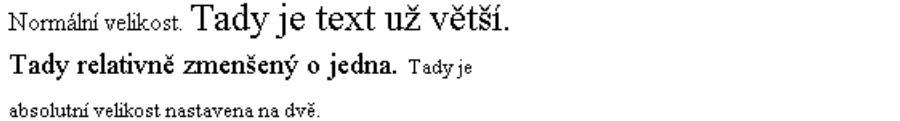
```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">
<HTML>
<HEAD>
    <TITLE>Písma rozličných velikostí</TITLE>
</HEAD>
<BODY>
<H1>Písma rozličných velikostí</H1>
<P><FONT SIZE=1>Písmo velikosti 1</FONT>
<P><FONT SIZE=2>Písmo velikosti 2</FONT>
<P><FONT SIZE=3>Písmo velikosti 3</FONT>
<P><FONT SIZE=4>Písmo velikosti 4</FONT>
<P><FONT SIZE=5>Písmo velikosti 5</FONT>
<P><FONT SIZE=6>Písmo velikosti 6</FONT>
<P><FONT SIZE=7>Písmo velikosti 7</FONT>
</BODY>
</HTML>
```

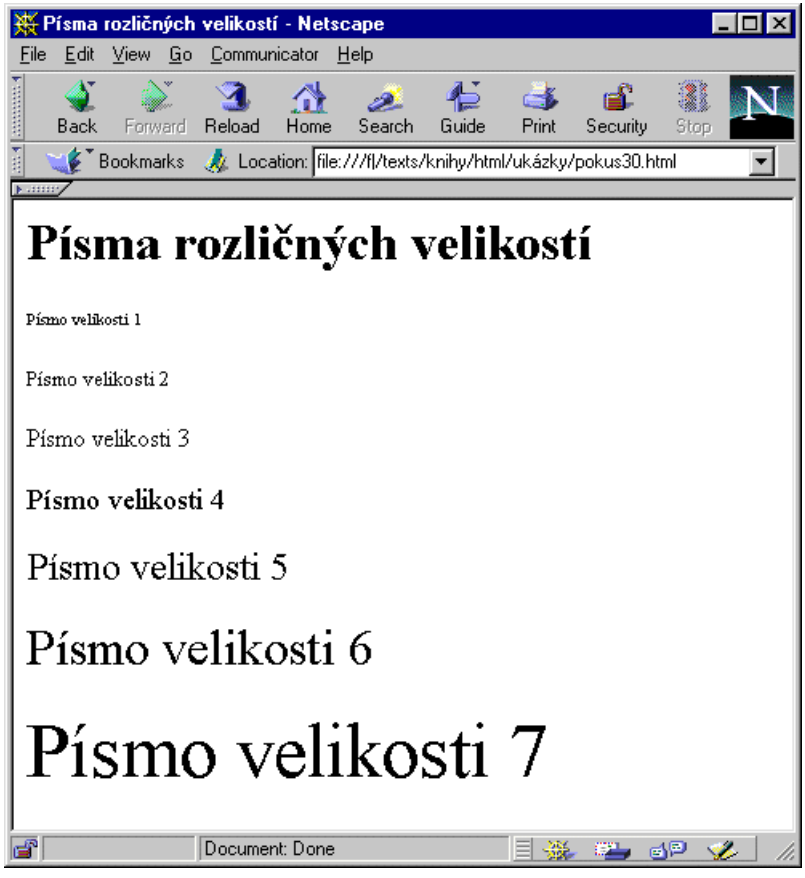

Při relativním určování velikosti písma používáme jako hodnotu atributu `SIZE` číslo se znaménkem plus (‘+’) nebo minus (‘–’). Použijeme-li `SIZE="-2"`, použité písmo se o dva stupně zmenší. Naopak při použití `SIZE="+1"` se použité písmo o jeden stupeň zvětší. Výsledná absolutní velikost písma se určí jako součet základní velikosti písma prohlížeče s relativní velikostí. Základní velikost písma je u každého prohlížeče jiná, ale většinou se jedná o hodnotu 3 nebo 4. Tuto hodnotu můžeme změnit použitím:

`<BASEFONT SIZE=`*«základní velikost»*`>`

Jako základní velikost můžeme samozřejmě použít jen absolutní určení velikosti od jedné do sedmi. Písmo této velikosti se použije pro text, který následuje za místem uvedení tagu `<BASEFONT>`. Od této velikosti se rovněž odvozuje výsledná velikost písma, pokud použijeme relativní udání velikosti písma atributem `SIZE` elementu `FONT`. Použití `<BASEFONT>` a `<FONT>` si demonstrujeme na malé ukázce:

```
Normální velikost.
<BASEFONT SIZE="5">Tady je text už větší.
<FONT SIZE="-1">Tady relativně zmenšený o jedna.</FONT>
<FONT SIZE="2">Tady je absolutní velikost nastavena na
dvě.</FONT>
```





Obr. 5-8: Absolutní velikosti písma

Nastavení základní velikosti písma pomocí `<BASEFONT>` a pomocí absolutního udání velikosti elementem `FONT` se nevztahuje na nadpisy. U nich lze použít pouze relativní změnu velikosti `<FONT SIZE="±«n»">...</FONT>`.

Pokud potřebujeme změnit velikost písma pouze o jeden stupeň, lze místo `<FONT SIZE="±1">...</FONT>` použít kratší zápis `<BIG>...</BIG>` resp. `<SMALL>...</SMALL>`.

Zmenšením písma o jeden stupeň lze velice jednoduše simulovat kapitálky.[2] Kapitálky se často používají pro zápis různých jmen a názvů:

[2] Vy, kdož nevíte, co to jsou kapitálky, vězte, že jimi byla zapsána věta, za kterou je odkaz k této poznámce pod čarou.

```
J<SMALL>ETHRO</SMALL> T<SMALL>ULL</SMALL><BR>
A<SMALL>LOIS</SMALL> J<SMALL>IRÁSEK</SMALL>
```

JETHRO TULL
ALOIS JIRÁSEK

Kapitálky vypadají pěkně v nadpisu nebo v jednom či dvou slovech po sobě. Delší text zobrazený kapitálkami je však špatně čitelný.

Některé prohlížeče podporují u elementu `FONT` *atribut* `FACE`*. Jako jeho hodnotu lze zapsat seznam čárkami oddělených názvů fontů. Pro zobrazení označené části textu se pak použije font, který co nejvíce odpovídá uvedeným fontům. Pokud chceme na stránku umístit nějaký text bezpatkovým písmem, můžeme použít* `<FONT FACE="Arial, Helvetica">`*. Atribut* `FACE` *není součástí HTML 3.2, v další verzi se s ním však počítá. Pro jeho větší použitelnost bude ještě zapotřebí definovat nějaký standard v pojmenovávání písem.*

5.7 Komentáře

Komentáře slouží k vkládání poznámek do HTML stránky. Tyto poznámky se však při zobrazování stránky v prohlížeči ignorují a nijak se nezobrazují. Obecný tvar komentáře je:

`<!--` *«text komentáře»* `-->`

Jako text komentáře můžeme uvést prakticky cokoliv, výjimku tvoří trojice znaků '`-->`' vyhrazená pro uzavření komentáře. V praxi poslouží komentáře ke vložení poznámek, podle kterých se můžeme lépe orientovat v HTML kódu. Komentářem můžeme obalit kusy HTML kódu, které nechceme dočasně zobrazovat. Komentáře se rovněž používají pro vkládání příkazů pro různé programy pracující s HTML dokumenty — to si ukážeme v kapitole „Dynamicky generované dokumenty".

6. Zpřístupnění stránek světu

Na svém počítači již máme vytvořeny první webovské stránky. Můžeme se jimi chlubit návštěvám, ale zatím nejsou přístupné všem přes Internet. V této kapitole si ukážeme, jak své stránky vystavit pro celý svět. Naučíme se je přenést z disku našeho počítače na WWW-server. Kromě samotného technického postupu si ukážeme, jak svou stránku zařadit do různých seznamů na Internetu. O naší stránce se pak dozví nejen naši přátelé či obchodní partneři, kterým sdělíme URL naší domovské stránky, ale i všichni ostatní.

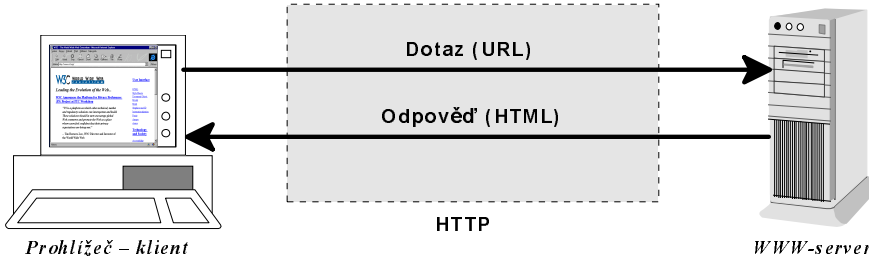
☺ Pokud vás právě netrápí problém, jak stránky ukázat celému světu, a radši byste studovali další možnosti jazyka HTML, můžete tuto kapitolu bez obav přeskočit. V budoucnu se k ní můžete kdykoliv vrátit.

6.1 WWW-server

Než se pustíme do výkladu konkrétních postupů, jejichž výsledkem je zpřístupnění našich stránek celému světu, objasníme si podrobněji pojem WWW-server. Pojem WWW-server v sobě totiž skrývá dva významy:

- WWW-server je program, který nabízí své služby jiným programům (těm se říká klienti). Když si chceme prohlížet určitou webovskou stránku, vyžádá si ji náš prohlížeč (ten zde vystupuje v roli klienta) od serveru. Jelikož klient i server jsou programy, nemohou spolu komunikovat lidskou řečí, ale nějakým jiným, předem daným způsobem. V Internetu se těmto pravidlům komunikace říká protokoly. Pro přenos webovských stránek se používá protokol HTTP (HyperText Transfer Protocol).
- WWW-server je počítač, který nabízí své služby ostatním počítačům. Na pevném disku serveru jsou jednotlivé webovské stránky uloženy ve formě souborů. Na WWW-serveru–počítači je spuštěn WWW-server–program. Server––program komunikuje s prohlížečem a na základě požadavků mu protokolem HTTP zasílá webovské stránky.

Protože počítač bez programu je jako auto bez benzínu, budeme použitím termínu WWW-server mít na mysli obvykle oba dva významy. Grafické znázornění



Obr. 6-1: Průběh komunikace mezi prohlížečem a WWW-serverem

průběhu komunikace mezi prohlížečem a WWW-serverem si můžeme prohlédnout na obrázku 6-1.

Jako hardware se pro WWW-servery používají nejčastěji obyčejné počítače PC. Pro náročnější aplikace se pak používají počítače zvučných jmen jako Sun nebo IBM. Tyto počítače pracují nejčastěji pod operačními systémy *Unix* nebo *Windows NT*.

Jako software pro WWW-server se v prostředí Unixu nejčastěji používá program *Apache*. Výhodou tohoto WWW-serveru je jeho cena — *Apache* je zdarma, a přitom se jedná o velice kvalitní produkt. Kromě *Apache* je k dispozici mnoho komerčních serverů. Nejúspěšnější na tomto poli jsou produkty firem Netscape Communications a Microsoft.

6.2 Umístění stránek na server

Konkrétních postupů, jak umístit stránky na server, je mnoho. Postupy se samozřejmě liší podle toho, zda k Internetu přistupujeme z domova pomocí komutované linky nebo zda máme Internet k dispozici v našem zaměstnání. Větší firmy a instituce totiž bývají k Internetu připojeny pevnou linkou a mají proto vlastní server — postup se pak samozřejmě liší.

Předtím než se na jednotlivé varianty podíváme podrobněji, učiníme několik společných předpokladů:

- Naši domovskou stránku včetně dalších navazujících stránek a obrázků budeme mít uloženu v jednom adresáři (a případně v jeho podadresářích). V tomto adresáři nebudeme mít žádné další soubory — zabráníme tak možným zmatkům.
- Odkazy mezi našimi vlastními stránkami budou pouze relativní (viz sekce *Relativní URL* na straně 30).

Pokud máme splněny předpoklady, můžeme se pustit do zveřejňování svých stránek. O tom, jak to udělat, bychom se měli poradit se správcem sítě v naší firmě nebo s naším poskytovatelem připojení. Jenom on totiž zná přesné detaily

jako uživatelská jména, hesla, jména adresářů, které slouží pro uložení stránek apod. Společný princip však zůstává a my se do něj pustíme.

Domovská stránka ve firmě, která je připojena k Internetu

Většina firem má již dnes vybudovanou lokální počítačovou síť (LAN), která spojuje počítače jednotlivých zaměstnanců. LAN jim umožňuje sdílet soubory, tiskárny, informační systém podniku apod. Servery sítě LAN pracují nejčastěji pod síťovým operačním systémem Novell Netware, v poslední době se uplatňují i Windows NT a Linux. Každý uživatel má na serveru obvykle vyhrazen určitý diskový prostor, kam může ukládat své soubory. Tento prostor se uživateli nejčastěji jeví jako nový disk (má tedy svoje písmenko — např. `H:`).

LAN firmy bývá do Internetu připojena pevnou linkou a firma má přímo ve svém sídle umístěn počítač, který zprostředkovává komunikaci počítačů v LAN se zbytkem Internetu. Pokud má firma svoje webovské stránky, znamená to, že některý z jejích počítačů musí pracovat jako WWW-server. Čeká na požadavky uživatelů Internetu, kteří si chtějí prohlížet firemní stránky, a jako odpověď jim posílá příslušné stránky.

Pokud WWW-server běží na počítači, který je využíván i jako file-server (uživatelé na něm mají uloženy své soubory), nastala asi nejlepší situace, jakou jsme si mohli přát. Budeme předpokládat, že tento server má doménovou adresu `server.firma.cz`. Většinu WWW-serverů pracujících pod Novellem, Windows NT i Linuxem lze nastavit tak, že při požadavku na stránku s URL `http://server.firma.cz/~jméno` se server podívá do adresáře uživatele `jméno` a hledá zde podadresář, který se nejčastěji jmenuje `html`, `public_html` či `public.www` (konkrétní nastavení záleží na správci serveru). V tomto adresáři pak hledá soubor s názvem `index.html` či `default.html` (opět záleží na správci serveru). V případě, že soubor existuje, vrátí jej jako odpověď. Překlad adresářů je samozřejmě prováděn i pro všechna další URL. Tak např. URL `http://server.firma.cz/~jméno/linky/cestovani.html` způsobí, že WWW-server vrátí stránku uloženou v souboru `cestovani.html`, který je uložen v adresáři `html\linky` (`public_html\linky`) uživatele `jméno`.

Za této situace stačí pro zpřístupnění našich stránek vytvořit v domovském adresáři podadresář daného jména (např. `h:\html`) a do něj překopírovat domovskou stránku, obrázky a naše další stránky. Domovskou stránku pojmenujeme `index.html` (`default.html`), protože pak si na své vizitky můžeme napsat pouze krátké URL bez jména souboru domovské stránky.

Dejme tomu, že jsme si vytvořili adresář `html` a do něj nahráli dva soubory: naši domovskou stránku `index.html` a druhou stránku `linky.html`, která obsahuje naše oblíbené odkazy. Naše uživatelské jméno v podnikové LAN je `jnovak`. URL naší domovské stránky, které můžeme rozdávat zájemcům o naši stránku, může mít dvě podoby:

```
http://server.firma.cz/~jnovak
http://server.firma.cz/~jnovak/index.html
```

Doporučuji vám používat první podobu, protože je kratší a snadněji zapamatovatelná.

URL naší stránky s odkazy bude: `http://server.firma.cz/~jnovak/linky.html`. Součástí URL nikdy není jméno adresáře (složky), ve kterém jsou uloženy všechny HTML dokumenty. Zápis URL, které obsahuje i jméno adresáře vyhrazeného pro stránky (`html`), `http://server.firma.cz/~jnovak/html/linky.html` je zcela špatný a při pokusu o načtení stránky s touto adresou nám prohlížeč ohlásí pouze chybové hlášení.

Server směruje všechny požadavky na naše stránky do speciálního adresáře, aby znepřístupnil ostatní adresáře před zvědavýma očima uživatelů Internetu. Kdyby pro HTML dokumenty nebyl vytvořen speciální adresář, dalo by se pomocí URL přistoupit k libovolnému souboru v domovském adresáři uživatele.

Pokud vaše firma WWW-server má, ale provozuje jej na jiném počítači než jsou vaše domovské adresáře, je postup publikování stejný, jako když jste z domova připojeni komutovanou linkou. Vaše firma bude v podobném postavení jako poskytovatel připojení — samozřejmě pouze technicky, nikoliv finančně. Jediný rozdíl je v rychlosti — komunikace s WWW-serverem nebude probíhat po pomalé telefonní lince, ale po relativně rychlé lokální síti.

Domovská stránka u poskytovatele připojení

Pokud jsme připojeni komutovanou linkou, musí být naše domovská stránka uložena na serveru poskytovatele — jinak by nebyla přístupná v době, kdy nejsme připojeni modemem k Internetu. Abychom mohli na server poskytovatele nahrát svoje stránky a zpřístupnit je tak světu, musíme se nejprve s poskytovatelem dohodnout.

Poskytovatel nám většinou za finanční úplatu poskytne jistý prostor na svém WWW-serveru. Zároveň by nám měl sdělit, jakým způsobem můžeme na WWW-serveru aktualizovat naše stránky.

Nejčastěji se používá přístup pomocí služby FTP (File Transfer Protocol). Tato služba umožňuje přenášet soubory na vzdálený počítač. V tomto případě se budou přenášet soubory se stránkami z našeho počítače na WWW-server poskytovatele.

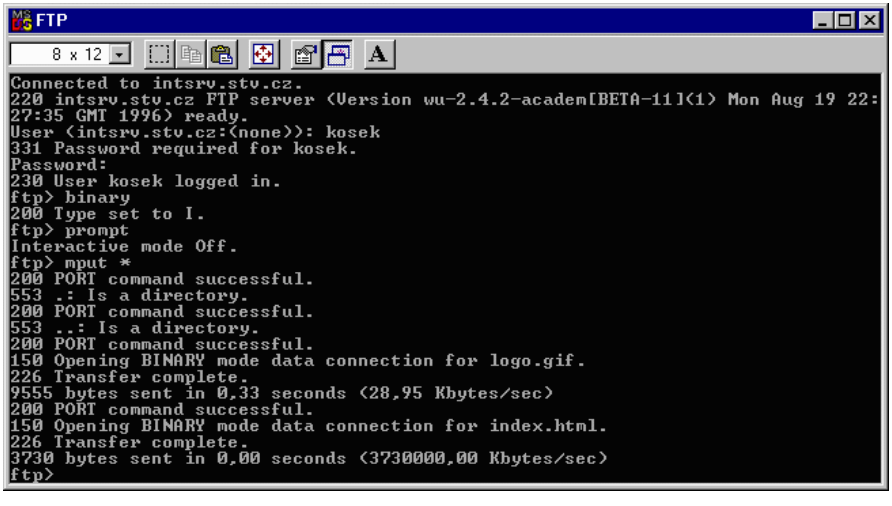
Abychom mohli využívat FTP, musí nám poskytovatel přidělit uživatelské jméno a heslo, kterým je chráněn přístup k vyhrazenému prostoru na WWW--serveru. Dejme tomu, že nám bude přiděleno jméno `kosek`.

Pro přenesení stránek se musíme spojit s WWW-serverem poskytovatele pomocí protokolu FTP. K tomu můžeme využít buď specializovaného FTP-klienta nebo třeba klidně *Netscape Navigator*.

Předpokládejme, že doménová adresa WWW-serveru poskytovatele bude `intsrv.stv.cz`. Pro spojení se serverem pomocí klasického FTP je nutné provést následující kroky:

1. Musíme být v adresáři, kde máme uloženu naši domovskou stránku, obrázky a případně další stránky.
2. Pomocí příkazu `ftp intsrv.stv.cz` se spojit se serverem poskytovatele.
3. Na výzvu `User name:` vložit naše uživatelské jméno `kosek`.
4. Na výzvu `Password:` vložit naše heslo.
5. Pokud vše proběhlo v pořádku, měl by se objevit prompt ve tvaru `ftp>`.
6. Zadat příkaz `binary`, kterým se přepneme do binárního režimu přenosu.
7. Příkazem `mput *` přenést všechny soubory z aktuálního adresáře našeho lokálního disku na server. Přenos jednotlivých souborů potvrdíme odesláním ‘y’ (tomu se můžeme vyhnout zadáním příkazu `prompt` ještě před příkazem `mput`).
8. Ukončit spojení se serverem pomocí příkazu `bye`.

Praktický průběh celého procesu si můžeme prohlédnout na obrázku 6-2. Tento postup lze samozřejmě nahradit použitím grafických FTP-klientů nebo již zmíněným *Navigatorem* a jeho funkcí File ▷ Upload File.



Obr. 6-2: Přenesení stránky na WWW-server pomocí FTP

Sami vidíme, že postup není zrovna pohodlný. Život si zkomplikujeme ještě více v případě, že jsou naše stránky rozčleněny do více podadresářů — pro každý adresář budeme muset soubory přenášet zvlášť. V tomto případě je lepší se s poskytovatelem domluvit na nějakém jiném řešení. Možností je opět několik:

- Přenést pomocí FTP všechny stránky jako jeden zapakovaný soubor; pak se k serveru připojit pomocí telnetu a soubor dekomprimovat do celé struktury adresářů, které obsahují stránky a obrázky.
- Připojit si disk u poskytovatele pomocí některého ze síťových souborových systémů jako je NFS nebo SMB a přistupovat k němu jako k lokálnímu disku.

První řešení je celkem často používané. Jediný problém spočívá ve výběru vhodného formátu komprimovaných souborů. V prostředí DOSu a Windows se nejčastěji používají ZIP-soubory. WWW-servery však nejčastěji pracují pod Unixem a zde se používá program *tar* společně s programem *gzip*. *Tar* umožňuje více souborů uložit do jednoho a *gzip* zase umí velice účinně zkomprimovat jeden soubor.

Pokud je na serveru poskytovatele program *unzip*, máme vyhráno. Doma si pohodlně pomocí programu *WinZip*[1] připravíme zapakovaný soubor (např. `html.zip`), který bude obsahovat všechny potřebné soubory s HTML dokumenty a obrázky. Výše uvedeným způsobem pomocí FTP přeneseme tento soubor na server poskytovatele. Pak se pomocí telnetu připojíme k serveru. Telnet spustíme příkazem `telnet intsrv.stv.cz`. Zadáme uživatelské jméno a heslo a příkazem `unzip html.zip` vytvoříme na serveru všechny potřebné soubory s HTML dokumenty a obrázky. Poté ještě můžeme pomocí příkazu `rm html.zip` smazat zapakovaný soubor a uvolnit tak trochu místa na disku poskytovatele.

V případě, že na serveru poskytovatele program *unzip* není, nezbude nám nic jiného než si sehnat *tar* a *gzip* ve verzích pro DOS/Windows.[2] Na našem počítači se přepneme do adresáře s webovskými stránkami a pomocí

```
tar cvf html.tar .
gzip html.tar
```

vytvoříme archivní soubor `html.tar.gz` (na některých systémech `html.tgz`). Tento soubor opět přeneseme pomocí FTP a k jeho rozbalení použijeme:

```
gunzip html.tar.gz
tar xvf html.tar
```

Ať už příslušné soubory dostaneme na WWW-server jakkoliv, URL naší domovské stránky bude nejčastěji ve tvaru `http://intsrv.stv.cz/kosek` nebo `http://intsrv.stv.cz/~kosek`.

---

[1] *WinZip* si můžete stáhnout na adrese `http://www.winzip.com/`.

[2] *tar* – `ftp://sunsite.mff.cuni.cz/Utils/TeX/CTAN/tools/tar/msdos/tar.exe`
gzip – `ftp://ftp.eunet.cz/pub/simtelnet/msdos/compress/gzip124.zip`

☺ Vždy si u správce serveru zjistíme, jaké je implicitní jméno pro soubor s domovskou stránkou. Nejčastěji se používá `index.html` nebo `default.html`. Naši domovskou stránku pojmenujeme tímto způsobem a můžeme pak používat kratší URL bez udání jména souboru.

6.3 Vědět, jak být viděn

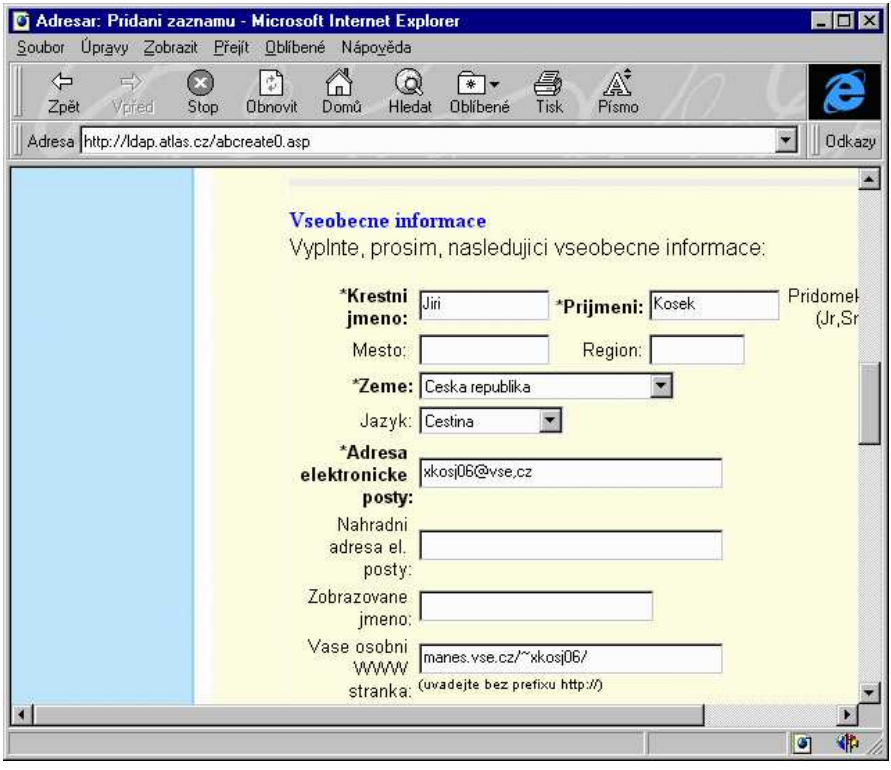
Parafráze známé zálesácké poučky nám něžně naznačuje obsah následujících řádků. S informacemi na Internetu se roztrhl pytel a uživatel Webu se může velice snadno cítit informacemi zavalen. V jisté fázi nezbývá uživateli nic jiného než naučit se pracovat se službami Internetu, které mu pomohou porvat se s jeho obrovským informačním potenciálem. Nejčastější jsou dvě základní podoby těchto služeb:

1. Tematicky členěné seznamy odkazů na další stránky. Mezi zástupce těchto služeb patří např. Seznam — `http://www.seznam.cz/`, U zdroje — `http://www.uzdroje.cz/` a Yahoo — `http://www.yahoo.com/`.
2. Vyhledávací stroje, mezi něž patří například zahraniční AltaVista firmy Digital — `http://altavista.digital.com/` a české Kompas — `http://kompas.seznam.cz/` a Atlas — `http://www.atlas.cz/`.

Aby se o našich stránkách dozvěděl zbytek světa, je dnes již nezbytně nutné oznámit to právě pomocí výše zmíněných služeb. O tom, jak a proč naše stránky zařadit do seznamů a indexů těchto služeb, si povíme na následujících stránkách.

Adresářové služby

Člověk má rád ve věcech řád, a proto s rozvojem komunikačních sítí vznikaly i systémy, které evidují a umožňují dále využívat běžné informace o uživatelích připojených k sítí. Běžnými informacemi máme v tomto případě na mysli jméno, adresu elektronické pošty a případně běžnou poštovní adresu. Služby, které umožňují tyto informace uchovávat a hlavně je efektivně prohlédávat, se nazývají adresářové služby. Dnes se ponejvíce používají adresářové služby založené na standardu X.500. Se servery, kde jsou uloženy informace o uživatelích, se komunikuje pomocí protokolu LDAP (Lightweight Directory Access Protocol). Jedná se o poměrně jednoduchý protokol, který umožňuje do adresáře serveru



Obr. 6-3: Přidání záznamu do LDAP adresáře

přidávat nové objekty, měnit stávající, ale hlavně prohledávat celý adresář. Někdy se podle jména protokolu říká i serverům s adresáři — označují se jako LDAP-servery.

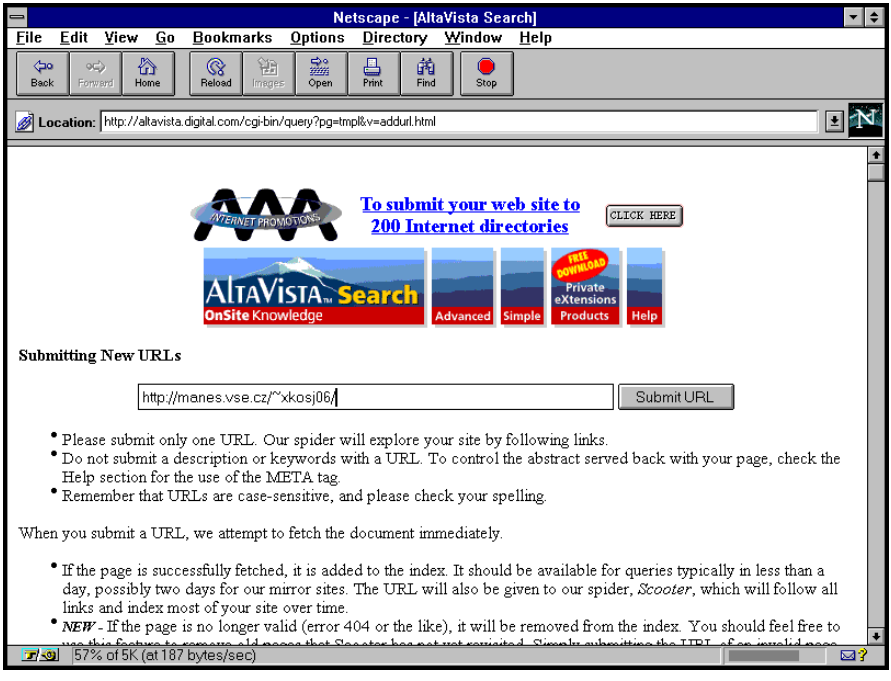
Zaregistrování do adresáře odpovídá zařazení našeho telefonního čísla do Zlatých stránek. Každý, kdo nám bude chtít poslat poštu, si při znalosti našeho jména může vyhledat naši e-mailovou adresu. Tím, že je adresář uložen v elektronické podobě, je v něm možné prohledávat i na základě mnoha dalších kritérií (můžeme třeba podle e-mailové adresy hledat skutečné jméno).

V Čechách je LDAP-server přístupný pomocí WWW-rozhraní na adrese `http://ldap.atlas.cz/`. Z této stránky lze rovnou provádět prohledávání adresáře. Vybereme-li odkaz Přidání adresy, můžeme se sami přidat do adresáře. Zápis údajů se provádí do připraveného formuláře (obr. 6-3).

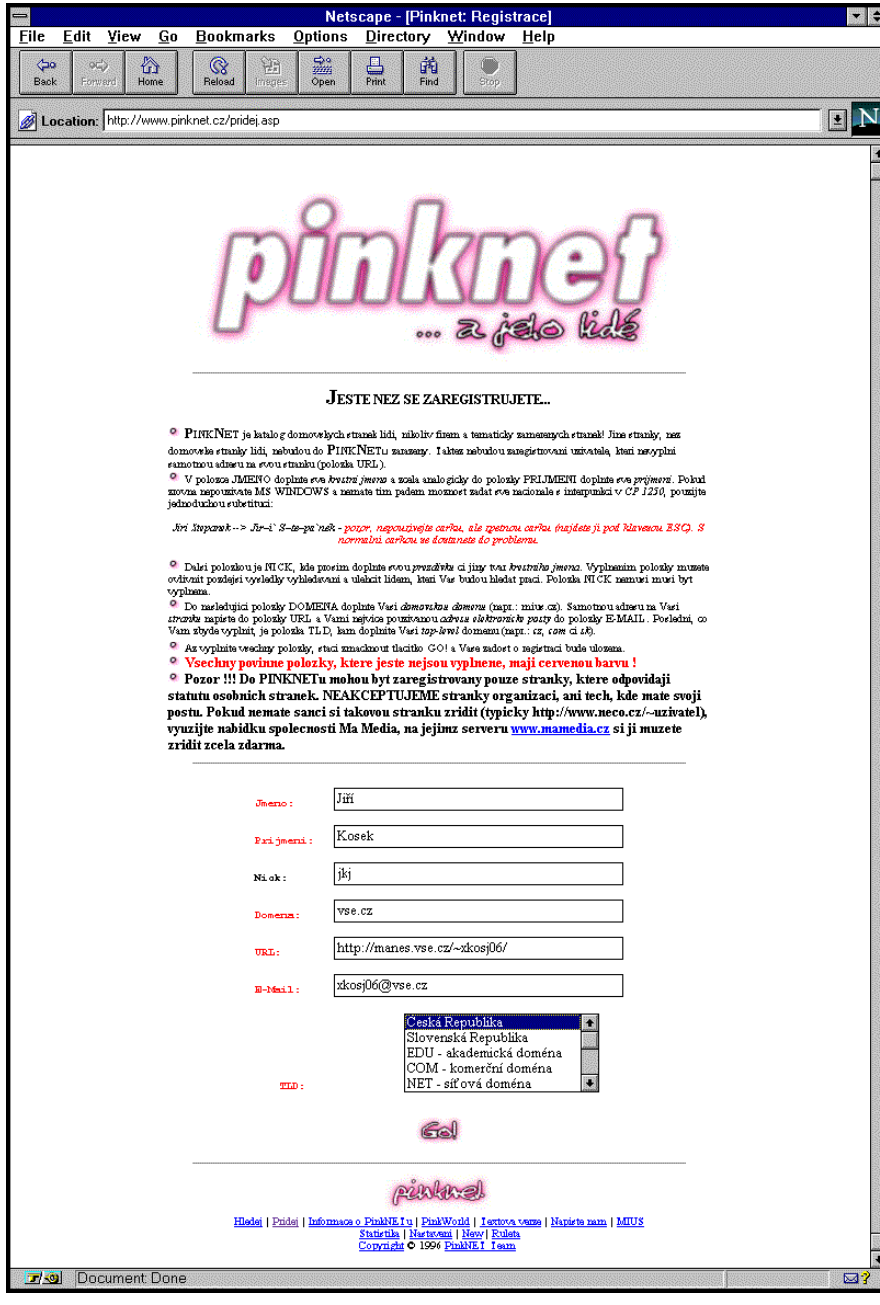
Seznamy a vyhledávací servery

Zatímco LDAP-servery slouží zejména pro vyhledávání lidí v Internetu, různé seznamy odkazů a vyhledávací servery se používají zejména k hledání webovských stránek.

Seznamy vznikají tak, že několik lidí pečlivě sbírá a do tematických kategorií třídí odkazy na stránky. Většina seznamů má na své úvodní stránce odkaz, kterým můžeme vyvolat formulář pro přidání naší stránky do seznamu. Odkaz bývá většinou pojmenován jako „Přidání URL“ nebo „Add URL“ na anglicky mluvících serverech. Tím, že naši stránku zaregistrujeme, dostane se do rukou správci seznamu. Ten si stránku prohlédne a v případě, že se mu jeví vhodná pro zařazení do seznamu, ji tam přidá. Díky tomu, že proces probíhá manuálně, objeví se stránka v seznamu s jistým spožděním. Obsah seznamů však bývá poměrně kvalitní.



Obr. 6-4: Předání naší domovské stránky AltaVistě k zaindexování



Obr. 6-5: Přidání záznamu do adresáře PinkNetu

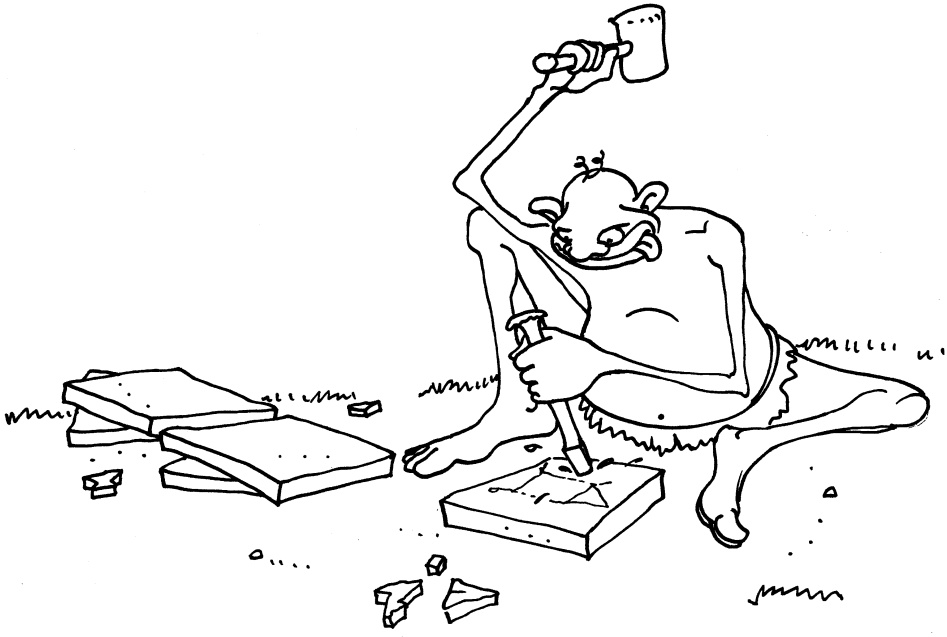
Vyhledávací servery pracují na trochu odlišném principu. Procházejí všechny stránky dosažitelné na Webu a u každé si pamatují všechna slova, která obsahuje. Tomuto procesu se říká indexování. Když pak hledáme nějakou informaci, zadáme několik klíčových slov, a server nám vrátí seznam stránek, které obsahují daná slova. Vyhledávací servery tak mají pod palcem mnohem větší prostor. Jejich používání je však složitější než u seznamů — pro nalezení relevantních dokumentů je potřeba umět klást dobré dotazy. A to je oblast, o které by šla napsat další a ještě mnohem tlustší kniha než tato.

Všechny vyhledávací servery obsahují funkci pro přidání nové stránky do indexu. Obvykle stačí přidat pouze domovskou stránku. Vyhledávací server automaticky zaindexuje i všechny stránky, na které vedou z domovské stránky odkazy.

Aby naše stránky byly k nalezení, je dobré jejich přidání do nejpoužívanějších seznamů a do indexů nejnavštěvovanějších prohledávacích serverů. Dnes mezi ty nejvíce „in“ servery patří `http://www.seznam.cz/`, `http://www.uzdroje.cz/`, `http://www.atlas.cz/` a `http://altavista.digital.com/`. Je možné, že v době čtení knihy budou existovat další nové služby. Neváhejte a svoje stránky přidejte i tam. Zajímavou aktivitou je i adresář soukromých domovských stránek na PinkNetu `http://www.pinknet.cz/`.

☺ Pokud své stránky píšeme česky, nemá smysl je zařazovat do zahraničních seznamů (to neplatí pro vyhledávací stroje).

Obr. 6-6: Vyhledávání v adresáři PinkNetu



7. Tabulky

V této kapitole se seznámíme s tvorbou tabulek. Tabulky můžeme použít jako efektivní nástroj při prezentaci informací, které je možno přehledně uspořádat do řádek a sloupců. Kromě tohoto primárního účelu lze tabulky využít i pro poměrně přesnou kontrolu nad grafickým rozvržením celé stránky (layoutem).

7.1 Základy tabulek

Tabulky se v HTML vytvářejí pomocí elementu `TABLE`. Tento element může obsahovat element `CAPTION`, který slouží k zadání nadpisu tabulky, a několik elementů `TR`, které obsahují jednotlivé řádky tabulky. Tabulky se v HTML zadávají po řádcích. (Název elementu `TR` je zkratkou Table Row — řádka tabulky.)

Každá řádka se pak skládá z jednotlivých buněk, které jsou zadány pomocí elementů `TD` nebo `TH`. `TD` se používá pro obyčejné buňky a `TH` pro buňky se záhlavím tabulky. Nejlepší bude malá ukázka tabulky, která zachycuje obraty dvou výrobků A a B v několika letech:

```
<TABLE>
<TR>
    <TH>Rok</TH>
    <TH>Obrat A</TH>
    <TH>Obrat B</TH>
</TR>
<TR>
    <TD>1994</TD>
    <TD>12,6 mil.</TD>
    <TD>3,6 mil.</TD>
</TR>
<TR>
    <TD>1995</TD>
    <TD>11,7 mil.</TD>
    <TD>5,9 mil.</TD>
</TR>
<TR>
    <TD>1996</TD>
    <TD>8,3 mil.</TD>
    <TD>9,7 mil.</TD>
</TR>
</TABLE>
```

Rok Obrat A Obrat B
1994 12,6 mil. 3,6 mil.
1995 11,7 mil. 5,9 mil.
1996 8,3 mil. 9,7 mil.

Jelikož všechny ukončovací tagy `</TR>`, `</TD>` i `</TH>` si prohlížeč snadno domyslí, není potřeba je uvádět. Předchozí tabulku můžeme zapsat tedy mnohem kratším a přehlednějším způsobem:

```
<TABLE>
<TR><TH>Rok     <TH>Obrat A     <TH>Obrat B
<TR><TD>1994    <TD>12,6 mil.   <TD>3,6 mil.
<TR><TD>1995    <TD>11,7 mil.   <TD>5,9 mil.
<TR><TD>1996    <TD>8,3 mil.    <TD>9,7 mil.
</TABLE>
```

☺ Tabulku je vhodné přehledně formátovat se sloupci pod sebou i v zápisu stránky. Usnadní nám to orientaci v tabulce při jejích opravách.

Naše tabulka je sice pěkně vyrovnaná, ale něco jí chybí. Mnohem lépe by vypadala, kdyby obsahovala mřížku. K tomu stačí uvést u `<TABLE>` atribut `BORDER`:

```
<TABLE BORDER>
<TR><TH>Rok     <TH>Obrat A     <TH>Obrat B
<TR><TD>1994    <TD>12,6 mil.   <TD>3,6 mil.
<TR><TD>1995    <TD>11,7 mil.   <TD>5,9 mil.
<TR><TD>1996    <TD>8,3 mil.    <TD>9,7 mil.
</TABLE>
```

Rok	Obrat A	Obrat B
1994	12,6 mil.	3,6 mil.
1995	11,7 mil.	5,9 mil.
1996	8,3 mil.	9,7 mil.

Atributu `BORDER` lze přiřadit hodnotu, která udává šířku rámečku okolo tabulky v pixelech. Samotné `<TABLE BORDER>` v předchozí ukázce je totožné s `<TABLE BORDER=1>`. Na obrázku 7-1 na následující straně si můžeme prohlédnout, jak dopadne naše tabulka, když použijeme šířku rámečku 8 (`<TABLE BORDER=8>`).

Rok	Obrat A	Obrat B
1994	12,6 mil.	3,6 mil.
1995	11,7 mil.	5,9 mil.
1996	8,3 mil.	9,7 mil.

Obr. 7-1: Tabulka s osmibodovým rámečkem

Horizontální zarovnání

Na naší ukázce s dvěma výrobky vidíme, že normální buňky (`TD`) jsou zarovnány vlevo a buňky záhlaví (`TH`) jsou vycentrovány. Standardní způsob zarovnání můžeme u každé buňky změnit pomocí atributu `ALIGN`. Ten může nabývat jedné ze tří hodnot: `LEFT` (zarovnání vlevo), `RIGHT` (zarovnání vpravo) a `CENTER` (centrování). Naši tabulku jednoduše upravíme tak, aby byl obsah buněk s čísly zarovnán vpravo (vypadá to mnohem lépe):

```
<TABLE BORDER>
<TR><TH>Rok  <TH>Obrat A                <TH>Obrat B
<TR><TD>1994 <TD ALIGN=RIGHT>12,6 mil. <TD ALIGN=RIGHT>3,6 mil.
<TR><TD>1995 <TD ALIGN=RIGHT>11,7 mil. <TD ALIGN=RIGHT>5,9 mil.
<TR><TD>1996 <TD ALIGN=RIGHT>8,3 mil.  <TD ALIGN=RIGHT>9,7 mil.
</TABLE>
```

Rok	Obrat A	Obrat B
1994	12,6 mil.	3,6 mil.
1995	11,7 mil.	5,9 mil.
1996	8,3 mil.	9,7 mil.

Pokud máme tabulku s mnoha sloupci, ve kterých chceme změnit zarovnání stejným způsobem, můžeme atribut `ALIGN` použít u tagu `<TR>`. V tomto případě bude zarovnání platné pro všechny buňky řádky tabulky. Pokud uvedeme zároveň `ALIGN` i u buňky, budeme mít takto určený způsob zarovnání přednost před zarovnáním pro celou řádku.

Vertikální zarovnání

Zcela obdobně jako horizontální zarovnání můžeme používat i vertikální. K jeho ovládání je zde atribut `VALIGN`. Ten může nabývat tří hodnot: `TOP` (obsah buňky je zarovnán s horním okrajem buňky), `BOTTOM` (obsah buňky je zarovnán se spodním okrajem buňky) a `MIDDLE` (obsah buňky je uprostřed buňky).

Atribut `VALIGN` je možno použít u elementů `TR`, `TD` a `TH`. Nastavení u `TR` je společné pro celou řádku. Způsob zarovnání určený přímo u `TD` a `TH` má větší váhu než společně určený pro celou řádku. Použití si ukážeme na vytvoření malého slovníčku pomocí tabulky. Půjde o stejný slovníček, kterým jsme demonstrovali použití elementu `DL` (definiční seznam) na straně 49.

```
<TABLE>
<TR VALIGN=TOP><TD><STRONG>HTTP</STRONG>
    <TD>Transportní protokol využívaný k přenosu souborů
        obsahujících popis WWW-stránek v jazyce HTML.
<TR VALIGN=TOP><TD><STRONG>FTP</STRONG>
    <TD>Transportní protokol používaný k přenosu souborů.
<TR VALIGN=TOP><TD><STRONG>NNTP</STRONG>
    <TD>Transportní protokol používaný k přenosu news.
</TABLE>
```

HTTP Transportní protokol využívaný k přenosu souborů obsahujících popis WWW-stránek v jazyce HTML.
FTP Transportní protokol používaný k přenosu souborů.
NNTP Transportní protokol používaný k přenosu news.

Slučování buněk

V některých tabulkách je potřeba občas několik buněk sloučit v jednu (např. máme společný nadpis nad více sloupci). Pokud chceme sloučit několik buněk v jednom řádku za sebou, stačí u buňky použít atribut `COLSPAN`. Jako jeho hodnotu uvedeme počet buněk, které se mají sloučit.

Zcela obdobně jako `COLSPAN` funguje `ROWSPAN`. Jediný rozdíl je v tom, že nyní se slučují buňky, které leží pod sebou a ne vedle sebe. Použití si ukážeme na vylepšené verzi naší původní tabulky:

```
<TABLE BORDER>
<TR><TH ROWSPAN=2>Rok<TH COLSPAN=2>Obrat z prodeje výrobků
<TR>          <TH>             A          <TH>             B
<TR><TD>1994 <TD ALIGN=RIGHT>12,6 mil. <TD ALIGN=RIGHT>3,6 mil.
<TR><TD>1995 <TD ALIGN=RIGHT>11,7 mil. <TD ALIGN=RIGHT>5,9 mil.
<TR><TD>1996 <TD ALIGN=RIGHT>8,3 mil.  <TD ALIGN=RIGHT>9,7 mil.
```

```
</TABLE>
```

Rok	Obrat z prodeje výrobků	
	A	B
1994	12,6 mil.	3,6 mil.
1995	11,7 mil.	5,9 mil.
1996	8,3 mil.	9,7 mil.

Velikost buňky

Velikost buněk je určována automaticky, tak aby se do nich vešel jejich obsah a aby se tabulka vešla do okna prohlížeče. Velikost buňky můžeme určit i ručně. Pomocí atributu `WIDTH` můžeme určit šířku buňky v pixelech. Atribut `HEIGHT` slouží k určení výšky opět v pixelech. Toto nastavení není pro prohlížeč závazné, pokud se mu obsah buňky do vymezeného prostoru nevejde, prostě buňku zvětší. Specifikace HTML 3.2 pouze říká, že by prohlížeč neměl buňku zobrazit menší než jsou zadané rozměry.

✍ *Většina prohlížečů dnes akceptuje u atributu* `WIDTH` *i hodnotu zadanou jako procento. Šířku buňky pak bere jako poměrnou část z šířky celé tabulky. Tato funkce je velice užitečná, ale není součástí standardu. Měli bychom se jí tudíž vyhnout.*

Mnohem praktičtější využití má poslední atribut buněk, o kterém jsme se dosud nezmínili. Pokud použijeme u buňky atribut `NOWRAP`, nebude se její obsah zalamovat do řádků. Použití je nasnadě zejména u delších textů v buňce, které však z nějakého důvodu nechceme rozdělit do více řádků.

✍ *Podobného efektu jako atribut* `NOWRAP` *můžeme dosáhnout používáním nedělitelné mezery* ` ` *místo běžné mezery mezi slovy. Sami uznáte, že použití* `NOWRAP` *je o poznání pohodlnější.*

Atributy elementu TABLE

Již jsme si řekli, že šířku rámečku okolo tabulky lze ovládat pomocí atributu BORDER. Mezi další užitečné atributy patří CELLPADDING. Pomocí něj můžeme určit vzdálenost obsahu buňky od okrajů buňky (opět v pixelech). Této vlastnosti využijeme poměrně často, protože jinak jsou údaje v tabulce příliš namačkány:

```
<TABLE BORDER=1 CELLPADDING=5>
<TR><TH>Rok  <TH>Obrat A                <TH>Obrat B
<TR><TD>1994 <TD ALIGN=RIGHT>12,6 mil. <TD ALIGN=RIGHT>3,6 mil.
<TR><TD>1995 <TD ALIGN=RIGHT>11,7 mil. <TD ALIGN=RIGHT>5,9 mil.
<TR><TD>1996 <TD ALIGN=RIGHT>8,3 mil.  <TD ALIGN=RIGHT>9,7 mil.
</TABLE>
```

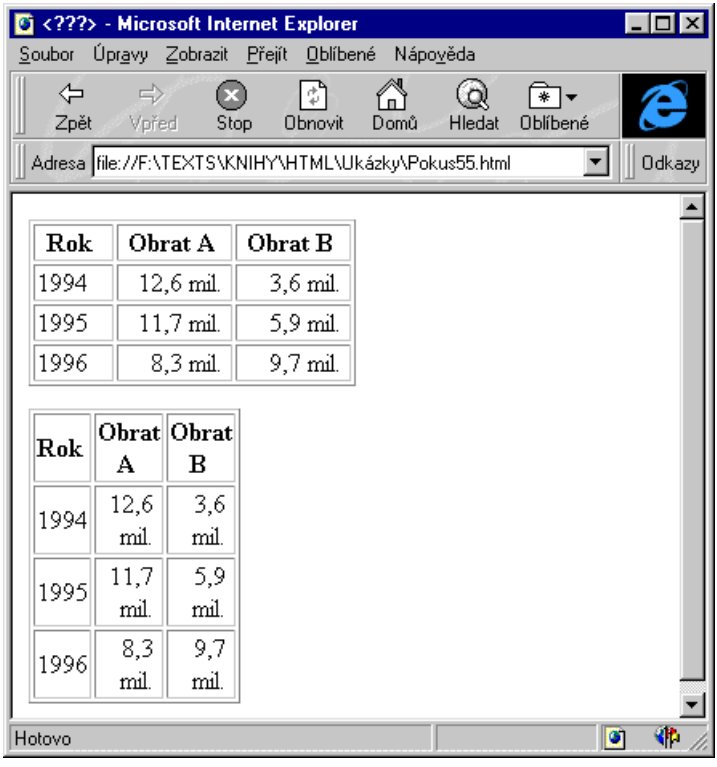
Rok	Obrat A	Obrat B
1994	12,6 mil.	3,6 mil.
1995	11,7 mil.	5,9 mil.
1996	8,3 mil.	9,7 mil.

Atribut CELLSPACING naopak zvětšuje mezeru mezi jednotlivými buňkami:

```
<TABLE BORDER=1 CELLSPACING=10>
<TR><TH>Rok  <TH>Obrat A                <TH>Obrat B
<TR><TD>1994 <TD ALIGN=RIGHT>12,6 mil. <TD ALIGN=RIGHT>3,6 mil.
<TR><TD>1995 <TD ALIGN=RIGHT>11,7 mil. <TD ALIGN=RIGHT>5,9 mil.
<TR><TD>1996 <TD ALIGN=RIGHT>8,3 mil.  <TD ALIGN=RIGHT>9,7 mil.
</TABLE>
```

Rok	Obrat A	Obrat B
1994	12,6 mil.	3,6 mil.
1995	11,7 mil.	5,9 mil.
1996	8,3 mil.	9,7 mil.

U tabulky lze určit požadovanou šířku pomocí atributu WIDTH. Jeho hodnotou může být buď šířka v pixelech nebo procento. V případě procenta se šířka tabulky



Rok	Obrat A	Obrat B
1994	12,6 mil.	3,6 mil.
1995	11,7 mil.	5,9 mil.
1996	8,3 mil.	9,7 mil.

Rok	Obrat A	Obrat B
1994	12,6 mil.	3,6 mil.
1995	11,7 mil.	5,9 mil.
1996	8,3 mil.	9,7 mil.

Obr. 7-2: Explicitní určení šířky tabulky

určí jako poměrná část šířky okna prohlížeče. Na obrázku 7-2 si můžeme prohlédnout naši tabulku nejprve s šířkou udanou absolutně `<TABLE WIDTH=200>` a poté i se šířkou zadanou relativně jako `<TABLE WIDTH="30%">`.

Tabulky lze obtékat textem podobně jako obrázky. Pokud u tabulky použijeme atribut `ALIGN` s hodnotou `LEFT` nebo `RIGHT`, bude tabulka umístěna vlevo (resp. vpravo) a text stránky bude obtékat okolo ní. Použijeme-li `ALIGN=CENTER`, umístí se tabulka doprostřed řádky.

✍ *Prohlížeč Microsoft Internet Explorer 3.01 u atributu* `ALIGN` *hodnotu* `CENTER` *ignoruje.*

Nadpis tabulky

Každé tabulce můžeme určit její nadpis pomocí elementu `CAPTION`. `CAPTION` musíme uvést ještě před definicemi vlastních řádek (`TR`) tabulky. U `CAPTION` můžeme použít atribut `ALIGN`. Pokud uvedeme hodnotu `TOP`, bude nadpis nad tabulkou. Hodnota `BOTTOM` umístí nadpis pod tabulku. Pokud neuvedeme nic, předpokládá se `ALIGN=TOP`.

```
<TABLE BORDER=1>
<CAPTION>Vývoj obratu z prodeje výrobků A a B</CAPTION>
<TR><TH>Rok  <TH>Obrat A               <TH>Obrat B
<TR><TD>1994 <TD ALIGN=RIGHT>12,6 mil. <TD ALIGN=RIGHT>3,6 mil.
<TR><TD>1995 <TD ALIGN=RIGHT>11,7 mil. <TD ALIGN=RIGHT>5,9 mil.
<TR><TD>1996 <TD ALIGN=RIGHT>8,3 mil.  <TD ALIGN=RIGHT>9,7 mil.
</TABLE>
```

Vývoj obratu z prodeje výrobků A a B

Rok	Obrat A	Obrat B
1994	12,6 mil.	3,6 mil.
1995	11,7 mil.	5,9 mil.
1996	8,3 mil.	9,7 mil.

Záludnosti tabulek

Možnosti tabulek v HTML 3.2 jsme již probrali kompletně. Nyní se podíváme na některé věci, které by nás při rutinním použití tabulek mohli překvapit.

První zajímavou věcí je, co se stane, když obsah některé buňky bude prázdný. Nejlepší bude vytvořit malou ukázku:

```
<TABLE BORDER=1>
<TR><TD>A<TD>B
<TR><TD>C<TD>
</TABLE>
```

A	B
C	

Na místě prázdné buňky se ani nenakreslila mřížka. Pokud nám tento efekt připadá nežádoucí, můžeme jako obsah buňky uvést nedělitelnou mezeru ` `. Ta se viditelně nijak nezobrazí, ale buňka již nebude prázdná a nakreslí se okolo ní rámeček.

```
<TABLE BORDER=1>
<TR><TD>A<TD>B
<TR><TD>C<TD> 
</TABLE>
```

A	B
C	

Druhé ošemetnosti jste si možná také všimli. V naší poslední ukázce je druhý sloupec tabulky širší. Za písmenem `B` je v buňce ještě mezera, kdežto za `A` a `C` není. Tabulka je zobrazena správně, když se podíváme do jejího zápisu, vidíme, že za `B` skutečně mezery jsou. Pokud nám takto vzniklá mezera kazí grafický dojem, můžeme buňku těsně uzavřít mezi tagy `<TD>` a `</TD>` (resp. `<TH>` a `</TH>`). Druhou možností je namačkat buňky v jedné řádce těsně za sebou a na konci uvést tag `</TR>`:

```
<TR><TD>A<TD>B</TR>
```

Nyní ovšem ztrácíme přehledný zápis, kdy máme pod sebou údaje patřící do stejného sloupce tabulky. Vyřešit to můžeme využitím skutečnosti, že mezery na začátku obsahu buňky se ignorují a můžeme je tedy využít k přehlednému zarovnání do sloupečků:[1]

```
<TABLE BORDER=1>
<TR><TH>              Rok<TH>     Obrat A<TH>     Obrat B</TR>
<TR ALIGN=RIGHT><TD>1994<TD>    12,6 mil.<TD>   3,6 mil.</TR>
<TR ALIGN=RIGHT><TD>1995<TD>    11,7 mil.<TD>   5,9 mil.</TR>
<TR ALIGN=RIGHT><TD>1996<TD>    8,3 mil.<TD>    9,7 mil.</TR>
</TABLE>
```

Rok	Obrat A	Obrat B
1994	12,6 mil.	3,6 mil.
1995	11,7 mil.	5,9 mil.
1996	8,3 mil.	9,7 mil.

Porovnáme-li obrázek této tabulky s předchozími, zjistíme, že pouze v této tabulce jsou nadpisy sloupečků perfektně vycentrovány.

7.2 Praktické použití tabulek

Zatím jsme tabulky použili pouze k prezentování suchých ekonomických čísel. Tabulky mají však mnohem širší uplatnění. Obsahem buňky může být i několik odstavců, obrázky, další tabulky apod. Nyní si ukážeme několik složitějších tabulek a ilustrujeme na nich některé z možností jejich uplatnění.

[1] Jestli vám zápis vizuálně připomíná zápis tabulek v TEXu, trefili jste do černého.

Stejně široké sloupečky

Pokud v nějaké tabulce potřebujeme z estetických důvodů stejně široké sloupce, můžeme s výhodou použít atribut `WIDTH` jako v naší ukázce:

```
<TABLE BORDER=1>
<TR ALIGN=CENTER><TH ROWSPAN=2>Tab. 1<TD>
        <VAR>x</VAR><TD>
        1<TD>
        2<TD>
        3<TD>
        4<TD>
        5<TD>
        6<TD>
        7<TD>
        8<TD>
        9<TD>
        10</TR>
<TR ALIGN=CENTER><TD>
        <VAR>x<SUP>3</SUP></VAR><TD WIDTH=30>
        1<TD WIDTH=30>
        8<TD WIDTH=30>
        27<TD WIDTH=30>
        64<TD WIDTH=30>
        125<TD WIDTH=30>
        216<TD WIDTH=30>
        343<TD WIDTH=30>
        512<TD WIDTH=30>
        729<TD WIDTH=30>
        1000</TR>
</TABLE>
```

Tab. 1	x	1	2	3	4	5	6	7	8	9	10
	x^3	1	8	27	64	125	216	343	512	729	1000

Zarovnávání desetinných čísel

Tabulky podle HTML 3.2 neumožňují zarovnání podle desetinné čárky. Jediné, co můžeme udělat, je zvolit zarovnání vpravo a u všech čísel doplnit nuly zprava tak, aby měly stejný počet cifer za desetinnou čárkou.

Text do více sloupečků

Pomocí tabulek můžeme velice snadno zařídit vícesloupcový text. Vytvoříme tabulku s jednou řádkou a dvěma buňkami. Do každé buňky dáme text jednoho sloupce a výsledkem bude text ve dvou sloupcích. Pokud takto zpřístupňujeme vícejazyčnou verzi nějakého textu, je dobré pro každý odstavec vytvořit samostatnou řádku tabulky a v této řádce dát do jednotlivých buněk jazykové mutace příslušného odstavce. Text ve sloupcích pak bude vzájemně synchronizován a bude přehlednější. Při tomto způsobu využití tabulek musíme obvykle změnit vertikální zarovnání u každé řádky tabulky pomocí `VALIGN=TOP`.

Komplexní tabulka

Nakonec si ukážeme vytvoření složitější tabulky v HTML. Výsledné zobrazení v prohlížeči si můžeme prohlédnout na obrázku 7-3.

Měsíc	Prodej zboží	
	A	B
Leden	865	16
Únor	917	8
Březen	1036	18
Shrnutí	V případě zboží A vidíme poměrně rychlý nárůst obratu zvláště v březnu. Zdá se, že to bude tím, že se v tu dobu oteplilo.	Pokud jde o zboží B, je to mizerné.
Závěr	**Ponechat a dále sledovat vývoj.**	**Zrušit výrobu**

Analýza prodeje

Obr. 7-3: Zobrazení složitější tabulky

```
<TABLE BORDER=1>
<CAPTION ALIGN=BOTTOM>Analýza prodeje</CAPTION>
<TR><TH ROWSPAN=2>Měsíc<TH COLSPAN=2>Prodej zboží</TR>
<TR>             <TH>           A<TH>           B</TR>
<TR ALIGN=CENTER><TD>Leden<TD>  865<TD>         16</TR>
<TR ALIGN=CENTER><TD>Únor<TD>   917<TD>         8</TR>
<TR ALIGN=CENTER><TD>Březen<TD> 1036<TD>        18</TR>
<TR VALIGN=BOTTOM><TD><TABLE>
                      <TR><TH HEIGHT=150>Shrnutí</TR>
                      <TR><TH><HR></TR>
                      <TR><TH VALIGN=TOP HEIGHT=50>Závěr</TR>
                      </TABLE>
                  </TD>
                  <TD WIDTH=150><TABLE WIDTH="100%">
                      <TR><TD ALIGN=CENTER HEIGHT=150>
                          V případě zboží A vidíme poměrně
                          rychlý nárůst obratu zvláště
                          v březnu. Zdá se, že to bude tím,
                          že se v tu dobu oteplilo.</TR>
                      <TR><TH><HR></TR>
                      <TR><TH VALIGN=TOP HEIGHT=50>Ponechat
                          a dále sledovat vývoj.</TR>
                      </TABLE>
                  </TD>
                  <TD WIDTH=150><TABLE WIDTH="100%">
                      <TR><TD ALIGN=CENTER HEIGHT=150>
                          Pokud jde o zboží B, je to mizerné.
                          </TR>
                      <TR><TH><HR></TR>
                      <TR><TH VALIGN=TOP HEIGHT=50>Zrušit
                          výrobu</TR>
                      </TABLE>
                  </TD>
</TABLE>
```

Ač tabulka vypadá celkem pěkně, není její zápis v HTML úplně čistý. Některé rozměry jsou v tabulce nastaveny natvrdo (`HEIGHT=50`, `HEIGHT=150`) a tabulka kvůli tomu není zcela nezávislá na výstupním zařízení.

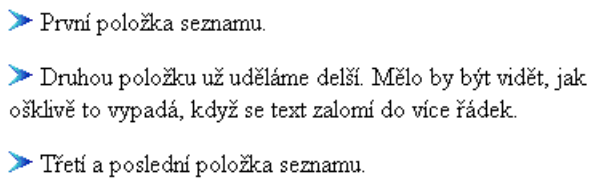
7.3 Praktické zneužití tabulek

Zatím jsme tabulky používali k zobrazení dat, která logicky do tabulky patří. Tabulky nám však poskytnou výbornou službu i v jiných případech. Rozmístěním různých grafických prvků do tabulky můžeme vytvořit zajímavé efekty. Tato oblast použití se těžko vysvětluje nějak příliš teoreticky. Ukážeme si tedy několik praktických příkladů. Váženému čtenáři mohou být návodem a inspirací při vytváření vlastních grafických kreací. Předtím, než některé z následujících řešení použijeme, bychom si měli rozmyslet, zda je to nezbytně nutné. Naše stránky sice mohou nabýt na atraktivnosti pro čtenáře, na druhou stranu porušujeme základní myšlenku HTML — vyznačování textu podle jeho významu a ne podle požadovaného grafického vzhledu.

Seznamy s grafickými odrážkami

Mnoha autorům webovských stránek nestačí puntíky a kolečka, které se používají jako odrážky v nečíslovaných seznamech. Místo nich používají malé obrázky, které mají stránce dodat osobitost a švih. Dejme tomu, že jako odrážku chceme používat šipku  , kterou máme uloženu v souboru `arrow.gif`. Většina autorů pak seznam vytvoří zhruba takto:

```
<P><IMG SRC=arrow.gif> První položka seznamu.
<P><IMG SRC=arrow.gif> Druhou položku už uděláme delší. Mělo by
být vidět, jak ošklivě to vypadá, když se text zalomí do více
řádek.
<P><IMG SRC=arrow.gif> Třetí a poslední položka seznamu.
```



Vidíme, že druhá položku seznamu je příliš dlouhá a musela se rozdělit do dvou řádek. Druhá řádka položky pak není odsazena, ale nepěkně zasahuje pod obrázek odrážky. Pomocí tabulek můžeme dosáhnout mnohem lepší výsledek. Vytvoříme tabulku se dvěma sloupci. V prvním bude obrázek šipky a ve druhém vlastní text položky seznamu. Tímto způsobem zabráníme tomu, aby se text přelil pod obrázek šipky. Nesmíme zapomenout na nastavení vhodného způsobu zarovnání buněk tabulky. V ukázce ještě u obrázku explicitně určíme jeho velikost, aby se tabulka mohla vykreslit rovnou. Upravený seznam zapíšeme tedy jako:

```
<TABLE WIDTH="100%">
<TR VALIGN=TOP ALIGN=LEFT>
    <TD><IMG SRC=arrow.gif WIDTH=15 HEIGHT=13>
    <TD>První položka seznamu.
<TR VALIGN=TOP ALIGN=LEFT>
    <TD><IMG SRC=arrow.gif WIDTH=15 HEIGHT=13>
    <TD>Druhou položku už uděláme delší. Mělo by být vidět, jak
    ošklivě to vypadá, když se text zalomí do více řádek.
<TR VALIGN=TOP ALIGN=LEFT>
    <TD><IMG SRC=arrow.gif WIDTH=15 HEIGHT=13>
    <TD>Třetí a poslední položka seznamu.
</TABLE>
```

- První položka seznamu.
- Druhou položku už uděláme delší. Mělo by být vidět, jak ošklivě to vypadá, když se text zalomí do více řádek.
- Třetí a poslední položka seznamu.

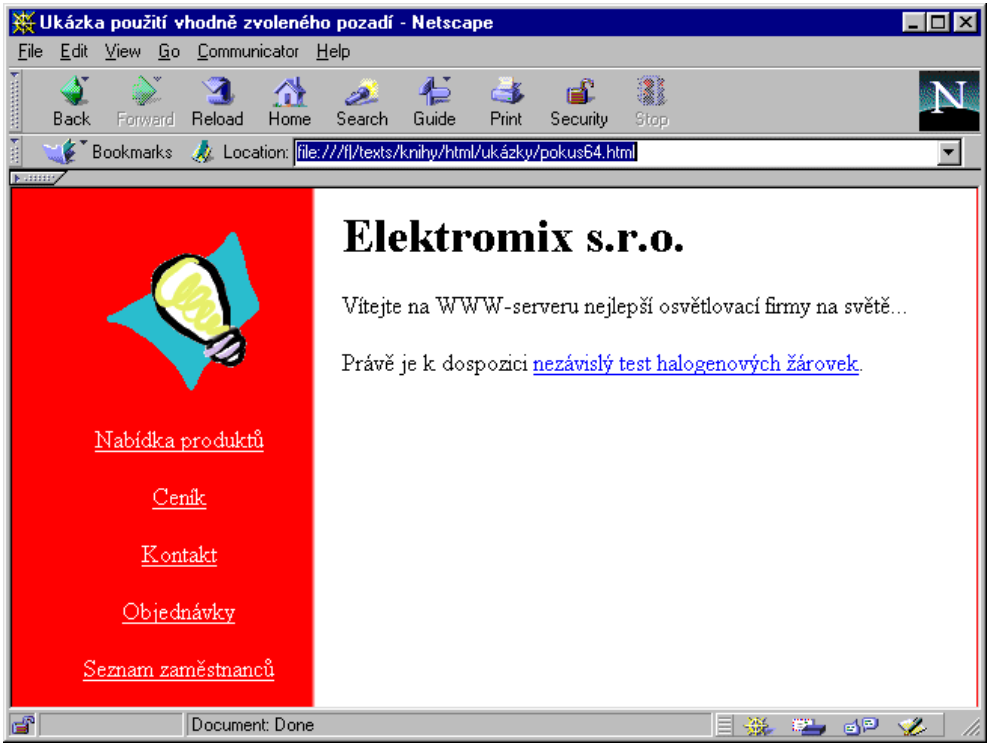
✍ *Celý seznam vypadá ještě o chlup lépe, když místo* `VALIGN=TOP` *použijeme zarovnání na účaří textu*[2] `VALIGN=BASELINE`. *Tato hodnota atributu je součástí rozšířené definice tabulek podle RFC1942 a bude i součástí další verze HTML s kódovým označením Cougar.*

Frame-like design

Mnoho stránek na Internetu má grafickou úpravu podobnou jako na obrázku 7-5 na následující straně. Na první pohled by se mohlo zdát, že výsledný efekt — navigační odkazy v levé části a informace v pravé — je dosažen použitím rámů (viz strana 204). Ve skutečnosti tomu tak není. Optické rozdělení stránky na dvě části bylo dosaženo použitím vhodného obrázku na pozadí. Jednalo se o úzký proužek, který byl v levé části červený a v pravé bílý (viz obr. 7-4). Tento

Obr. 7-4: Obrázek je jen úzký proužek

[2] Účaří je pomyslná linka, na kterou se skládají písmena v jedná řádce (typografové mi doufám prominou nepatrné zjednodušení).



Obr. 7-5: I tato stránka je dílem tabulek

podklad se opakuje pod sebou podle potřeby, takže vyplní tolik místa, kolik je potřeba.

Abychom mohli obsah stránky umístit přesně nad červenou a bílou část stránky, použili jsme tabulku. Celá stránka je vlastně tvořena jednou tabulkou, která má dva sloupce. U obou sloupců je šířka pomocí atributu `WIDTH` nastavena tak, aby jejich obsah vyšel nad červenou resp. bílou část stránky. Naší ukázkové stránce tedy odpovídá následující zápis v HTML:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">
<HTML>
<HEAD>
    <TITLE>Ukázka použití vhodně zvoleného pozadí</TITLE>
</HEAD>
<BODY BACKGROUND="zadek.gif">

<TABLE WIDTH=600>
<TR>
```

```
<!-- PRVNÍ SLOUPEC -- NA ČERVENÉM -->
<TD WIDTH=200 VALIGN=TOP>
<DIV ALIGN=CENTER>
<IMG SRC="logo.gif">
<P><A HREF="xxx">Nabídka produktů</A>
<P><A HREF="xxx">Ceník</A>
<P><A HREF="xxx">Kontakt</A>
<P><A HREF="xxx">Objednávky</A>
<P><A HREF="xxx">Seznam zaměstnanců</A>
</DIV>

<!-- DRUHÝ SLOUPEC -- NA BÍLÉM -->
<TD WIDTH=400 VALIGN=TOP>
<H1>Elektromix s.r.o.</H1>
Vítejte na WWW-serveru nejlepší osvětlovací firmy na světě...
<P>
Právě je k dispozici <A HREF="test.html">nezávislý test
halogenových žárovek</A>.

</TABLE>
</BODY>
</HTML>
```


Jak jste jistě vytušili, obrázek s proužkem, který vytváří pozadí, je uložen v souboru `zadek.gif`. První sloupeček naší tabulky má šířku 200 pixelů a druhý 400. Z toho nám logicky vyjde, že šířka obrázku pro podklad by měla být 600 pixelů. V praxi je však potřeba proužek udělat širší (asi 640 bodů), protože tabulka je vždy posunuta kousek vpravo od levého okraje a šířka celé stránky pak zabere více než 600 pixelů. Krátký proužek podkladu by mohl způsobit, že se podklad bude opakovat i vedle sebe a na pravém okraji stránky se objeví úzký svislý červený pruh.

✍ *Ty z vás, kteří znají rámy, možná napadne, proč požadovaného efektu dosahovat takto složitě — stačí přece použít rámy. Důvody pro zde popsané řešení jsou dva. Za prvé elementy a atributy pro práci s rámy nejsou zatím přijaty jako žádný standard (budou až v Cougaru). Druhým důvodem může být, že při troše práce a kouzlení s buňkami tabulky lze dosáhnout toho, že nějaký obrázek přesahuje do obou částí stránky (červené i bílé). Vypadá to mnohdy velmi efektně a při použití rámů si o takovém výsledku můžeme nechat jen zdát.*

Layout stránky pod palcem

Pokud chceme vytvořit stránku, která má speciální požadavky na rozmístění jednotlivých grafických prvků, mohou nám v tom pomoci tabulky. Dejme tomu, že chceme vytvořit domovskou stránku fiktivního serveru zaměřeného na sport. Vzhled stránky, kterou chceme vytvořit, si můžeme prohlédnout na obrázku 7-6.

Obr. 7-6: Stránka s náročným layoutem

Rozmístění jednotlivých obrázků a odkazů je dosaženo šikovně použitou tabulkou. Celá stránka (kromě nadpisu) je tabulka s vhodně sloučenými buňkami (viz 7-7 na následující straně).

Podíváme-li se na zdrojový kód stránky, neobsahuje až tak nic speciálního:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">
<HTML>
<HEAD>
```

Obr. 7-7: Umístění objektů v buňkách tabulky

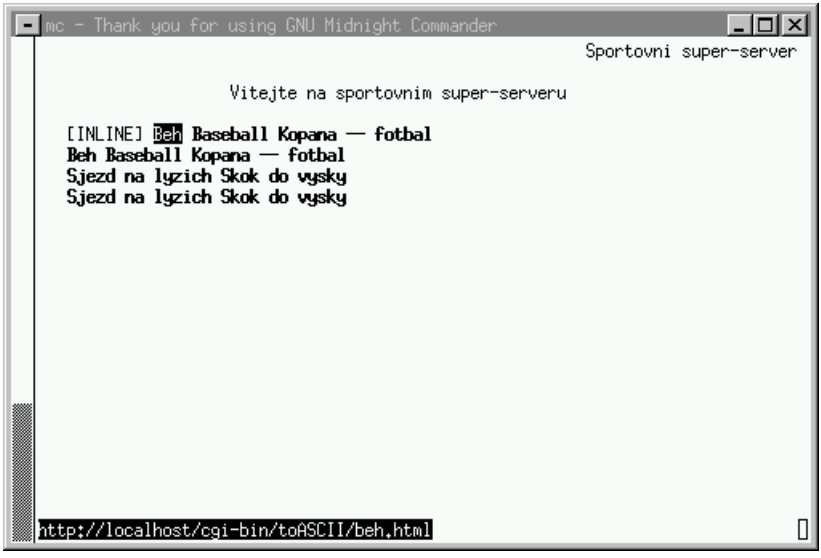
```
<TITLE>Sportovní super-server</TITLE>
</HEAD>

<BODY>
<H1 ALIGN=CENTER>Vítejte na sportovním super-serveru</H1>

<TABLE WIDTH="100%" ALIGN=CENTER>
<TR ALIGN=CENTER VALIGN=TOP>
    <TD ROWSPAN=4><IMG SRC="sportlogo.gif">
    <TD WIDTH=150><A HREF="beh.html"><IMG
        SRC="beh.gif" ALT="Běh" BORDER=0></A>
    <TD WIDTH=150><A HREF="baseball.html"><IMG
     SRC="baseball.gif" ALT="Baseball" BORDER=0></A>
    <TD WIDTH=150><A HREF="kopana.html"><IMG
     SRC="kopana.gif" ALT="Kopaná -- fotbal" BORDER=0></A>
```

```
<TR ALIGN=CENTER VALIGN=TOP>
    <TD HEIGHT=40><A HREF="beh.html">Běh</A>
    <TD><A HREF="baseball.html">Baseball</A>
    <TD><A HREF="kopana.html">Kopaná -- fotbal</A>
<TR ALIGN=CENTER VALIGN=TOP>
    <TD COLSPAN=2><A HREF="sjezd.html"><IMG
        SRC="sjezd.gif" ALT="Sjezd na lyžích" BORDER=0></A>
    <TD><A HREF="skok.html"><IMG
     SRC="skok.gif" ALT="Skok do výšky" BORDER=0></A>
<TR ALIGN=CENTER VALIGN=TOP>
    <TD COLSPAN=2><A HREF="sjezd.html">Sjezd na lyžích</A>
    <TD><A HREF="skok.html">Skok do výšky</A>
</TABLE>

</BODY>
</HTML>
```

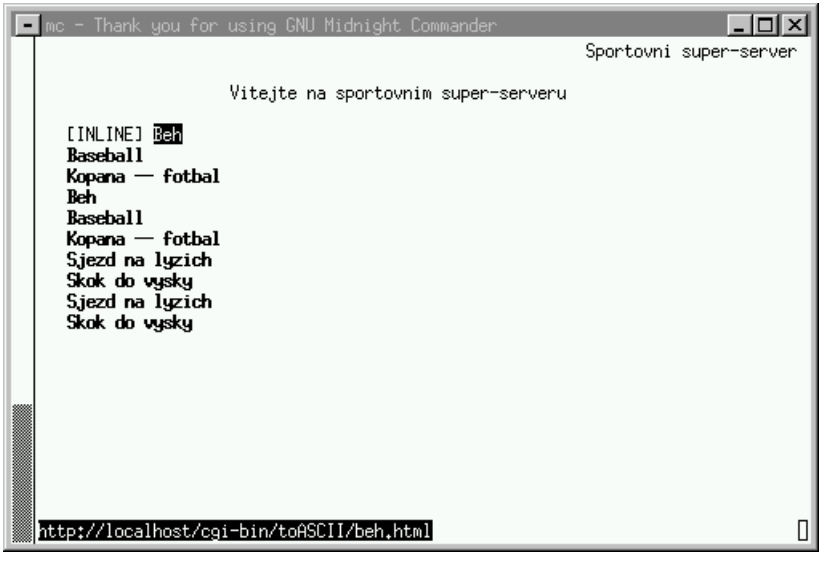
Pomocí tabulky jsme informace velice přehledně uspořádali na obrazovce.

Co se však stane v prohlížeči, který tabulkám nerozumí? Můžeme se na to podívat na obrázku 7-8 na následující straně. Tam je výsledek naší stránky zobrazen ve znakovém prohlížeči *Lynx*. Obrázky zmizely, tabulky jakbysmet a přehledné formátování je fuč. Přesto situace není zcela ztracená, vše by v *Lynxu* vypadalo mnohem lépe, kdyby odkazy byly uspořádány pod sebou. To lze zařídit celkem snadno. Na konec obsahu každé buňky přidáme tag `<BR>`. Ten způsobí v *Lynxu*, který ignoruje tabulky, přechod na nový řádek. Grafické prohlížeče tento tag vůbec neovlivní, protože v buňce není za `<BR>` nic, co by se mělo zobrazit. Zápis stránky upravíme a výsledek si prohlédneme na obrázku 7-9 na následující straně:


```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">
<HTML>
<HEAD>
<TITLE>Sportovní super-server</TITLE>
</HEAD>

<BODY>
<H1 ALIGN=CENTER>Vítejte na sportovním super-serveru</H1>

<TABLE WIDTH="100%" ALIGN=CENTER>
<TR ALIGN=CENTER VALIGN=TOP>
    <TD ROWSPAN=4><IMG SRC="sportlogo.gif">
    <TD WIDTH=150><A HREF="beh.html"><IMG
        SRC="beh.gif" ALT="Běh" BORDER=0></A><BR>
```



Obr. 7-8: Naše stránka v Lynxu nedopadla moc dobře



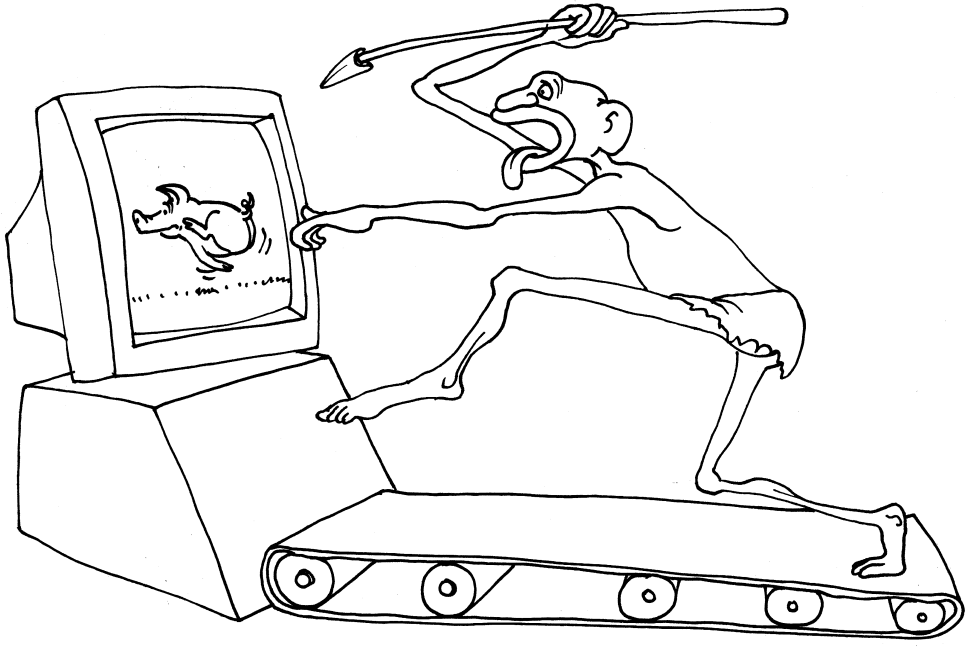
Obr. 7-9: Po úpravě je naše stránka již použitelná

```
<TD WIDTH=150><A HREF="baseball.html"><IMG
    SRC="baseball.gif" ALT="Baseball" BORDER=0></A><BR>
<TD WIDTH=150><A HREF="kopana.html"><IMG
 SRC="kopana.gif" ALT="Kopaná -- fotbal" BORDER=0></A><BR>
```

```
<TR ALIGN=CENTER VALIGN=TOP>
    <TD HEIGHT=40><A HREF="beh.html">Běh</A><BR>
    <TD><A HREF="baseball.html">Baseball</A><BR>
    <TD><A HREF="kopana.html">Kopaná -- fotbal</A><BR>
<TR ALIGN=CENTER VALIGN=TOP>
    <TD COLSPAN=2><A HREF="sjezd.html"><IMG
        SRC="sjezd.gif" ALT="Sjezd na lyžích" BORDER=0></A><BR>
    <TD><A HREF="skok.html"><IMG
        SRC="skok.gif" ALT="Skok do výšky" BORDER=0></A><BR>
<TR ALIGN=CENTER VALIGN=TOP>
    <TD COLSPAN=2><A HREF="sjezd.html">Sjezd na lyžích</A><BR>
    <TD><A HREF="skok.html">Skok do výšky</A><BR>
</TABLE>

</BODY>
</HTML>
```

Ponaučení, které plyne z následující ukázky, je jednoznačné. Než pustíme naše stránky do světa, měli bychom si je prohlédnout alespoň v těch nejpoužívanějších prohlížečích a zkontrolovat, zda je výsledné zobrazení podle našich představ. Dnes jsou asi nejpoužívanějšími prohlížeči *Internet Explorer*, *Netscape Navigator* a znakový prohlížeč *Lynx*.

✍ *Z poslední ukázky si můžeme odnést zajímavý poznatek. Všechny prohlížeče jsou naprogramovány tak, aby neznámé tagy a atributy ignorovaly. Proto nám taky Lynx stránku zobrazil tak, jako by tam žádná tabulka nebyla.*



8. Dynamicky generované dokumenty

Až dosud byly všechny naše stránky uloženy kdesi na disku WWW-serveru a ten je v případě požadavku klienta v nezměněné podobě odeslal. Obsah těchto stránek je poměrně stálý — jejich autor je občas pouze aktualizuje. V této kapitole si ukážeme, jak vytvářet stránky, které obsahují informace v čase proměnlivé. Příkladem nám mohou být HTML stránky obsahující údaje z nějaké databáze, aktuální čas serveru, seznam přihlášených uživatelů. Do stejné kategorie spadají i známá počítadla přístupů, kterými se pyšní mnohé stránky.

Aby mohly být dosaženy výše uvedené vlastnosti, je potřeba pro každý dotaz klienta provést určité akce, které povedou k požadované stránce. V současnosti existují dvě cesty, jak toho dosáhnout:

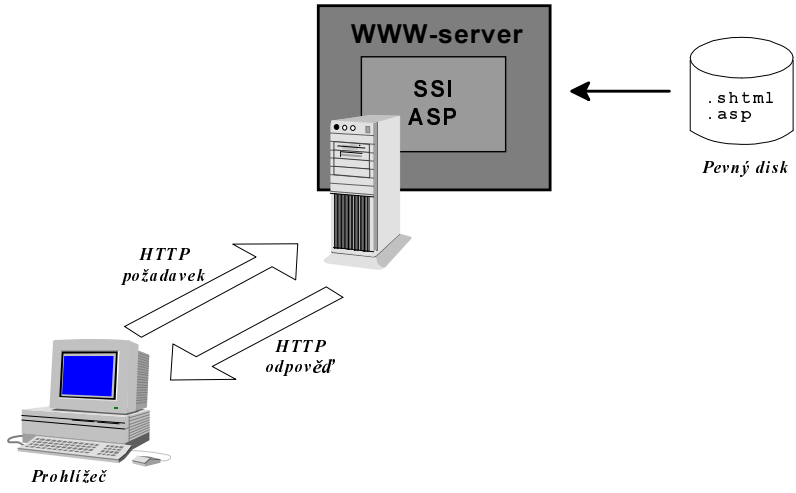
- *Serverem vkládané vsuvky*. V tomto případě jsou HTML dokumenty doplněny speciálními příkazy, které WWW-server nahradí částmi HTML kódu. Server tedy musí v každém dokumentu vyhledat příkazy, provést je a jejich výsledek vložit do dokumentu. Teprve takto upravený dokument je zaslán klientovi jako odpověď na jeho požadavek.

  Dnes se běžně používá několik druhů serverem vkládaných vsuvek. Nejobvyklejší jsou *Server Side Includes (SSI)*. Jedná se o velice rozšířenou a poměrně dobře standardizovanou metodu. Mezi další patří např. *Active Server Pages (ASP)*. Jde o novou technologii firmy Microsoft použitou v jejich *Internet Information Serveru*. Pracuje na stejném principu jako SSI, nabízí však mnohem větší možnosti. Na obdobném principu pracují další systémy — např. LiveWire od Netscapu nebo volně šiřitelný PHP/FI.

- *CGI-skripty*. CGI-skript je program, jehož výstupem je HTML dokument. Když server zjistí, že požadavek klienta směřuje k CGI-skriptu, spustí jej a jeho výsledek předá klientovi jako odpověď na požadavek.

  Možnosti CGI-skriptů jsou takřka neomezené, záleží na schopnostech programátora. Chování CGI-skriptů může být ovlivňováno parametry zaslanými spolu s požadavkem klienta. Pro psaní CGI-skriptů lze využít v podstatě libovolný programovací jazyk.

Výše uvedené způsoby dynamicky generovaných stránek jsou prováděny na serveru. Existuje však i další možnost — zařazení krátkých programů (tzv. skriptů) přímo do stránky. Tyto skripty jsou prováděny až prohlížečem v našem počítači. O tomto způsobu tvorby interaktivních stránek si více povíme v sekci „Skripty“ na straně 221 a v kapitole věnované dynamickému HTML.



Obr. 8-1: Princip serverem vkládaných vsuvek

8.1 Server side includes

SSI jsou příkazy vložené do HTML dokumentu, které se provádějí před zasláním dokumentu klientovi. SSI se do dokumentu vkládají jako komentář se speciálním tvarem:

`<!--#`*«příkaz»* *«parametr»*`="`*«hodnota»*`"-->`

Tím, že příkaz obalíme komentářem, máme zaručeno, že pokud server SSI nepodporuje a příkaz neprovede, bude celý komentář v prohlížeči ignorován a neporuší zbytek stránek. Valná většina serverů podporuje alespoň následující příkazy:

`#include`
SSI bude nahrazeno souborem, jehož jméno je definováno parametrem `file`. Tento příkaz se využívá zejména pro vkládání standardizovaných patiček do všech stránek na serveru. Taková patička pak obsahuje informace o firmě a správci serveru. Praktická ukázka:

```
<!--#include file="signature.inc"-->
```

Cesta k souboru zadaná parametrem `file` může být jen relativní. Pokud potřebujeme použít cestu absolutní, můžeme místo `file` použít parametr `virtual`. Kořenový adresář v tomto případě odpovídá kořenovému adresáři s dokumenty na daném WWW-serveru.

`#fsize`
Tento příkaz má stejné parametry jako `#include`, ale výsledkem jeho expanze je číslo, udávající velikost daného souboru.

`#flastmod`
Tento příkaz do stránky vloží datum poslední modifikace souboru zadaného stejně jako u příkazu `#include`.

`#echo`
Tento příkaz se nahradí obsahem proměnné specifikované v parametru `var`. Mezi proměnné, jejichž obsah můžeme do stránky vložit, patří:

`DATE_GMT`	datum a čas na serveru v Greenwichském čase
`DATE_LOCAL`	datum a čas na serveru
`DOCUMENT_NAME`	jméno dokumentu, ve kterém je SSI
`DOCUMENT_URI`	URL adresa dokumentu
`LAST_MODIFIED`	datum a čas poslední změny
`QUERY_STRING_UNESCAPED`	případný dotaz zaslaný klientem

Následující SSI do dokumentu vloží datum jeho poslední modifikace:

```
Stránka byla naposledy změněna:
<!--#echo var="LAST_MODIFIED"-->
```

`#exec`
Vloží do dokumentu výstup z programu. Program, který se má provést, se zadává pomocí parametru `cmd`. Pokud takto chceme spustit CGI-skript, použijeme parametr `cgi` a jako jeho hodnotu zadáme URL adresu CGI-skriptu. Na serveru běžícím pod Unixem můžeme snadno zařídit výpis právě přihlášených uživatelů pomocí příkazu `who`. Pokud chceme tento výpis zařadit do stránky, použijeme následující SSI:

```
<PRE>
<!--#exec cmd="who"-->
</PRE>
```

Příkaz jsme zabalili do elementu `PRE`, aby bylo zachováno řádkování výstupu z programu `who`.

`#config`
Pomocí tohoto příkazu můžeme nastavit formát výstupu ostatních příkazů. Pokud například jako parametr uvedeme `sizefmt="bytes"`, budou všechny výpisy velikosti souborů v bytech. Pokud použijeme `sizefmt="abbrev"`, použijí se zkratky KB a MB. Ovlivnit lze samozřejmě více parametrů, jejich výčet bychom měli nalézt v dokumentaci k příslušnému WWW-serveru.

Abychom mohli SSI na svých stránkách používat, musíme si zjistit, zda je náš server podporuje. Kromě toho si musíme zjistit, zda jejich používaní není zakázáno správcem serveru. SSI totiž umožňují provádění příkazů, které mohou poskytnout mnoho cenných informací pro hackery, a správci serverů proto často možnost používání SSI vypínají.

Stránky s SSI bývá zvykem ukládat do souborů s příponou `.shtml`. WWW-server pak hledá SSI příkazy pouze v těchto souborech a nezpomaluje se prohlížením běžných stránek.

8.2 Active Server Pages

ASP nabízí mnohem větší možnosti než SSI. V dokumentech můžeme kromě HTML-tagů používat příkazy speciálního skriptovacího jazyka. Před odesláním stránky klientovi jsou provedeny všechny příkazy a posílá se již obyčejná HTML stránka.

Jako skriptovací jazyk se v ASP standardně používá VisualBasic Script. Toto nastavení je však možno změnit v konfiguraci Internet Information Serveru a použít třeba JavaScript.

Příkazy skriptovacího jazyka se uzavírají mezi sekvence znaků `<%` a `%>`. To je drobná nevýhoda oproti SSI, které jsou v komentářích. Pokud si ASP stránku prohlédneme v prohlížeči přímo ze souboru (neinterpretovanou serverem), uvidíme i všechny skriptovací příkazy a naší stránce to na kráse a přehlednosti zrovna nepřidá. Podívejme se tedy, jak ASP vypadají prakticky.

Přiřadit nějakou hodnotu do proměnné můžeme velice snadno:

```
<% pozdrav = "Buďte zdráv!" %>
```

Pokud chceme obsah nějaké proměnné nebo výrazu vložit do stránky, stačí použít následující konstrukci:

`<%=` *«výraz»* `%>`

K vložení textu „Buďte zdráv!“ do naší stránky můžeme tedy použít příkaz `<%= pozdrav %>`. VBScript obsahuje i mnoho užitečných předdefinovaných funkcí. Např. funkce `Now` vrací aktuální čas serveru:

```
Stránku jste si stáhli v <%= Now %>
```

V ASP lze používat i podmínky. Snadno vytvoříme stránku, která nás přivítá jinak dopoledne a jinak odpoledne:

```
<BIG>
<% If Time >= #12:00:00 AM# And Time <= #12:00:00 PM# Then %>
Dobré ráno!
<% Else %>
Dobré odpoledne!
<% End If %>
</BIG>
```

V případě, že čas je někdy mezi půlnocí a polednem, vloží server do stránky pro klienta text „Dobré ráno“. V opačném případě odešle obligátní „Dobré odpoledne!“.

Kromě podmínek nám VBScript nabídne i cykly. Typickou ukázku použití cyklu, předváděnou na všech prezentacích firmy Microsoft, si samozřejmě nenecháme ujít. Následující ASP stránka

```
<% For I = 1 to 7 %>
<FONT SIZE=<%= I %>>Ukázkový text</FONT><BR>
<% Next %>
```

dorazí do prohlížeče takto:

```
<FONT SIZE=1>Ukázkový text</FONT><BR>
<FONT SIZE=2>Ukázkový text</FONT><BR>
<FONT SIZE=3>Ukázkový text</FONT><BR>
<FONT SIZE=4>Ukázkový text</FONT><BR>
<FONT SIZE=5>Ukázkový text</FONT><BR>
<FONT SIZE=6>Ukázkový text</FONT><BR>
<FONT SIZE=7>Ukázkový text</FONT><BR>
```

ASP stránky by měly být ukládány v souborech s příponou `.asp`. Dokumentace k příkazům VBScriptu a k dalším technologiím souvisejícím s ASP je součástí microsoftího produktu *Visual InterDev*, který je určen pro vývoj dynamických webovských stránek.

8.3 HTTP — Hypertext Transfer Protocol

Než se pustíme do výkladu principů CGI-skriptů, uděláme si krátkou exkurzi do protokolu HTTP. Bude se nám to pak hodit. Protokol HTTP se používá při komunikaci klienta (prohlížeče) s WWW-serverem. Celý proces komunikace se skládá ze dvou hlavních kroků:

- Klient naváže spojení ze serverem a zašle mu svůj požadavek.
- Server pošle klientovi odpověď na jeho požadavek.

Protokol HTTP definuje, jak bude vypadat požadavek klienta i odpověď serveru. Aby to nebylo tak jednoduché, používají se dnes tři verze HTTP — 0.9, 1.0 a 1.1. Pro nás bude zajímavé seznámit se s některými rysy verzí 0.9 a 1.0. Verze 1.1 přináší výrazná vylepšení v podobě zrychlení přenosu složitějších stránek, ale v pro nás podstatném se od verze 1.0 neliší. Předem ještě dodám, že neprobereme zdaleka všechny možnosti, které HTTP nabízí. Zaměříme se pouze účelově na ty pro nás nejdůležitější.

HTTP verze 0.9

V HTTP/0.9 je syntaxe požadavku velice jednoduchá. Skládá se z jednoho řádku, kde je za klíčovým slovem `GET` uvedena cesta k požadovanému dokumentu. Například:

```
GET /produkty/cenik.html
```

Odpovědí na takovýto dotaz je obsah samotného souboru `cenik.html`.

HTTP verze 1.0

V HTTP/1.0 je už dotaz o něco složitější. Jeho tvar je následující:

`GET` *«cesta k dokumentu»* `HTTP/1.0`
«hlavičky»
«CRLF»

Oproti verzi 0.9 přibyla identifikace verze protokolu. Dále je možné požadavek blíže upřesnit pomocí různých hlaviček. Každá řádka hlaviček má syntaxi *«jméno hlavičky»*: *«hodnota»*. Prohlížeče v hlavičkách posílají různé doplňující informace jako svoji identifikaci apod. Podrobný popis všech hlaviček nalezneme v popisu HTTP v RFC 1945 [4].

Důležité je, že požadavek je ukončen prázdnou řádkou (*«CRLF»*). Nejjednodušší požadavek v HTTP/1.0 se tedy musí skládat z řádky s příkazem `GET` a z prázdné řádky.

Odpověď v HTTP/1.0 je také složitější. Její obecný tvar je:

`HTTP/1.0` *«stavový kód»* *«stavové hlášení»*
«hlavičky»
«CRLF»
«obsah odpovědi»

Stavový kód vypovídá o tom, nakolik se podařilo splnit požadavek klienta. Stavové kódy jsou třímístná čísla a jsou rozděleny do pěti skupin. Kódy začínající jedničkou jsou rezervovány pro použití v budoucnu. Dvojka na začátku indikuje úspěšné provedení operace. Pokud nám server vrátí stavový kód začínající trojkou, jde o přesměrování — k vyřízení požadavku se klient musí obrátit jinam. Čtyřka je vyhrazena pro chyby klienta (špatná syntaxe požadavku, neexistující dokument apod.) a pětka pro chyby serveru.

Stavové hlášení slovně popisuje chybový kód. Nejčastější tvar první řádky odpovědi je

```
HTTP/1.0 200 OK
```

Tím nám server dává najevo, že vše je O.K.

Hlavičky mohou opět obsahovat různé užitečné informace. Asi nejdůležitější z nich je hlavička `Content-type`. Ta udává druh dat, která se vracejí jako obsah odpovědi. Druhy dat se uvádějí v souladu s typy MIME. Pro HTML existuje typ `text/html`, pro čistý ASCII-text pak `text/plain`. Obrázku GIF přísluší typ `image/gif`.

Po hlavičkách je zde prázdná řádka a pak již následují sama data vraceného objektu. Typ objektu je určen právě podle hlavičky `Content-type`. Představují-li vracená data webovskou stránku, bude v hlavičce řádka:

```
Content-type: text/html
```

Aby vše fungovalo, musí se server chovat inteligentně. Pokud mu přijde požadavek pomocí HTTP/0.9, musí i odpověď poslat ve tvaru definovaném protokolem HTTP/0.9. Pokud mu přijde požadavek v HTTP/1.0, odpoví rovněž HTTP/1.0.

8.4 CGI — Common Gateway Interface

CGI definuje rozhraní, pomocí něhož může WWW-server komunikovat s libovolným programem. Na program jsou kladeny pouze dva požadavky:

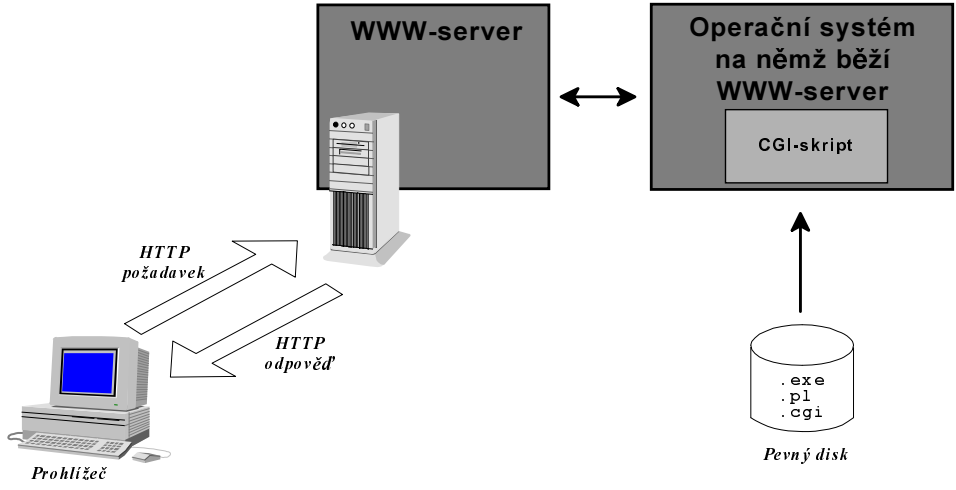
- musí umět přebírat parametry předané podle definice CGI;
- výsledkem jeho činnosti je odpověď ve formátu HTTP (bez první stavové řádky) zapsaná na standardní výstup programu.

V praxi se pro psaní těchto programů používají nejčastěji příkazové interprety *Unixu* nebo skriptový jazyk Perl. Odtud se pro ně také vžilo označení CGI-skript. CGI-skript však může být i program napsaný v nějakém klasickém programovacím jazyce jako Pascal, C, C++ nebo Java.[1]

CGI-skripty se používají téměř ke všemu — k objednání pizzy, pro přístup k rešeršním systémům, pro diskusní skupiny s WWW-rozhraním, pro počítadla přístupů, jako prostředek pro elektronické nakupování atd. Možnosti jsou opravdu téměř neomezené a vše závisí pouze na schopnosti programátora a správce WWW-serveru.

CGI-skripty jsou programy, a proto jejich vytváření vyžaduje znalost nějakého programovacího jazyka. My se v našich ukázkách budeme vyhýbat nějakým složitým a nepochopitelným konstrukcím. Pokud se však necítíte být programátorem (či sličnou programátorkou) a CGI-skripty vás nechávají chladnými, klidně přeskočte až na poslední část kapitoly věnovanou počítadlům (strana 150).

[1] Java nabízí kromě možnosti psaní krátkých Java-apletů pro zpestření WWW-stránek i vlastnosti moderních programovacích jazyků pro psaní běžných programů.



Obr. 8-2: Princip CGI-skriptu

Jak tedy probíhá vlastní spolupráce WWW-serveru s CGI-skriptem?

1. Nejprve klient pošle serveru požadavek, jehož URL směřuje na některý z CGI-skriptů. Kromě toho může klient serveru předat i parametry ovlivňující chování CGI-skriptu.
2. Server zjistí, že požadované URL je potřeba obsloužit CGI-skriptem. To server nejčastěji pozná podle toho, že na začátku cesty v URL je odkaz na adresář `cgi-bin`. Server spustí skript a předá mu parametry buď přes rozhraní CGI nebo na standardní vstup.
3. CGI-skript proběhne a zpracuje předaná data. Na svůj standardní výstup zapíše odpověď ve tvaru HTTP (bez první stavové řádky). Tento výstup je zachycen WWW-serverem.
4. WWW-server doplní další položky do hlavičky HTTP odpovědi a odešle klientovi odpověď. (Server se zároveň postará i o to, aby odpověď byla ve správné verzi HTTP. Pokud klient vznesl požadavek HTTP/0.9, server vrátí odpověď podle HTTP/0.9 a odstraní tedy všechny hlavičky vygenerované CGI-skriptem a pošle pouze samotné tělo odpovědi — HTML stránku.)

První CGI-skript

Pokud se chceme pustit do psaní vlastních CGI-skriptů, musíme si nejprve ověřit, zda jsou na našem serveru správcem povoleny. Rovněž musíme zjistit, do jakého adresáře můžeme naše skripty ukládat a kam je tento adresář mapován WWW-serverem. Na unixových serverech se skripty nejčastěji ukládají do adresáře `/usr/local/etc/httpd/cgi-bin/`. Náš první skript pojmenujeme např. `test.cgi`. Server provádí přemapování adresářů, a tak URL našeho skriptu bude `http://`*«server»*`/cgi-bin/test.cgi`.

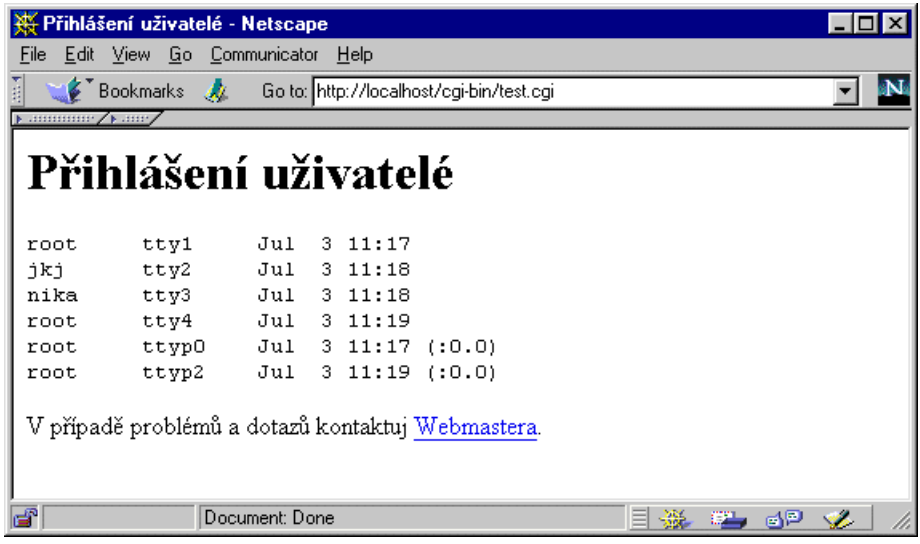
Jako ukázka principu CGI nám postačí následující jednoduché zadání: vytvoříme skript, který nám v prohlížeči zobrazí seznam uživatelů přihlášených k serveru. Skript vytvoříme pro příkazový interpret Unixu `sh`, činnosti všech použitých příkazů si vysvětlíme.

V Unixu existuje příkaz `who`, který vypisuje seznam přihlášených uživatelů. Úkolem našeho CGI-skriptu bude zavolat tento příkaz a zabalit do HTML tak, aby se v prohlížeči pěkně zobrazil. CGI-skript by měl před generovaný HTML dokument uložit ještě hlavičky HTTP. Ve většině případů úplně postačí použití hlavičky `Content-type: text/html`, kterou říkáme, že bude následovat HTML stránka. Nesmíme zapomenout, že mezi hlavičkami a stránkou musí být v HTTP prázdná řádka:

```
#!/bin/sh
echo 'Content-type: text/html'
echo
echo '<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">'
echo '<HTML>'
echo '<TITLE>Přihlášení uživatelé</TITLE>'
echo '</HEAD>'
echo '<BODY>'
echo '<H1>Přihlášení uživatelé</H1>'
echo '<PRE>'
/usr/bin/who
echo '</PRE>'
echo 'V případě problémů a dotazů kontaktuj'
echo '<A HREF="mailto:webmaster@server.cz">Webmastera</A>.'
echo '</BODY>'
echo '</HTML>'
```

První magická řádka skriptu pouze říká, že skript se má interpretovat programem `sh`. Příkaz `echo` zajistí vypsání svého argumentu na standardní výstup. Zapíšeme tedy hlavičku, prázdnou řádku a kostru HTML dokumentu. Dovnitř HTML dokumentu bude skriptem vložen výstup programu `who` — tedy seznam uživatelů. Výsledek volání našeho skriptu si můžete prohlédnout na obrázku 8-3 na následující straně.

Aby skript správně fungoval, musíme u něj nastavit příznak spouštění pro všechny uživatele pomocí příkazu `chmod +x test.cgi`.



Obr. 8-3: Výstup z našeho prvního CGI-skriptu

8

Předávání parametrů

Náš první CGI-skript byl opravdu hodně jednoduchý. Bylo to mimo jiné způsobeno tím, že nepotřeboval ke své činnosti žádné informace od uživatele. To však není typický příklad. Mnoho CGI-skriptů slouží např. pro vyhledávání v různých databázích. Uživatel tedy musí zadat klíčová slova, ta se musí nějak předat CGI-skriptu a ten se podle nich již nějak zařídí. Předtím, než se podíváme, jak si CGI-skript může vyzvednout parametry, ukážeme si, jak klient může nějaká data poslat WWW-serveru.

Klient má v podstatě dvě možnosti, jak data na server odeslat. První z nich se používá pro kratší informace a označuje se jako metoda GET. Na konec URL se za otazník připojí dotaz. V dotazu musí být všechny mezery nahrazeny znakem '+' a provedeno případné překódování některých vyhrazených znaků (viz strana 25).

Provádět ruční zápis kryptopřepisů však není pro uživatele zrovna příjemné, a tak existují možnosti, jak přimět prohlížeč k automatickému vygenerování celého URL i s dotazem. Existují tři možnosti, kdy prohlížeč vyvolá URL a sám zakóduje dotaz do patřičné formy:

- Stránka obsahuje ve svém záhlaví tag `<ISINDEX>`. V tomto případě si prohlížeč vyžádá od uživatele zadání textu do jednoho vstupního pole. Obsah tohoto pole příslušně zakóduje a připojí jej jako dotaz za URL. URL přitom bývá nejčastěji specifikováno pomocí `<BASE HREF=`*«URL»*`>`. URL by mělo ukazovat na skript, který bude umět zadaný dotaz zpracovat.

- U obrázku, který je odkazem, je použit atribut `ISMAP`. V tomto případě po kliknutí myší na obrázku odešle klient serveru požadavek. Požadavek se bude skládat z URL specifikovaného v elementu `A`, který obrázek obaluje, a ze souřadnic kliknutí myší. Bude-li tedy na stránce zařazen následující obrázek:

  ```
  <A HREF="http://server.cz/image">
  <IMG SRC="mapa.gif" ISMAP></A>
  ```

  a uživatel na obrázek klikne na souřadnicích (100, 89), prohlížeč vytvoří požadavek `http://server.cz/image?100,89`. Odpovědí na takovýto požadavek bývá nejčastěji adresa nějaké stránky, která se skrývá za danou oblastí obrázku.

  Tento postup byl dříve používán pro vytváření klikacích map. Dnes však jeho použití ztrácí význam, protože všechny prohlížeče podporují klientské klikací mapy (atribut `USEMAP`), které jsou mnohem šetrnější k přenosovým kapacitám sítě.

- Stránka obsahuje formulář (element `FORM`) jehož přenosová metoda je stanovena na `GET`. O formulářích si více povíme v další kapitole, prozatím nám bude stačit, že formulář slouží ke vstupu několika položek od uživatele. V tomto případě jsou za URL doplněné o otazník přidány jména jednotlivých položek i s jejich obsahem. Jednotlivé položky jsou odděleny znakem '`&`'.

Druhá metoda, jak může klient poslat data na server, se použije v případech, kdy stránka obsahuje formulář s metodou `POST`. V tomto případě jsou data kódována stejně jako v předchozím případě, ale jsou odesílaná zvlášť (nestávají se částí URL).

Když víme, jak může klient poslat data na server, vraťme se zpět k problému, jak je server předá našemu skriptu. Existují tři možnosti:

1. data jsou předána jako parametry na příkazovém řádku;
2. data jsou předána v proměnné prostředí;
3. data jsou předána na standardní vstup skriptu.

Poslední uvedená metoda se uplatní pouze v případech, kdy jsou data odesílána z formuláře metodou POST.

Kromě dat, která předá server CGI-skriptu, má skript vždy k dispozici několik užitečných údajů v různých proměnných prostředí. Podle definice CGI ve verzi 1.1 se jedná o následující proměnné:

`REQUEST_METHOD`	určuje způsob předávání informací — `GET` nebo `POST`
`QUERY_STRING`	obsahuje data přenášená metodou `GET`
`PATH_INFO`	cesta, která má být zpracována skriptem; nejčastěji jde o část cesty v URL za jménem skriptu
`PATH_TRANSLATED`	cesta ke stejnému souboru jako `PATH_INFO`; v tomto případě však byla cesta přemapována podle konfigurace serveru
`CONTENT_TYPE`	MIME typ dat zasílaných metodou POST
`CONTENT_LENGTH`	délka dat zasílaných metodou POST
`SCRIPT_NAME`	URL právě prováděného skriptu
`SERVER_NAME`	jméno serveru
`SERVER_PORT`	číslo portu
`SERVER_SOFTWARE`	jméno a verze programu pracujícího jako WWW-server
`SERVER_PROTOCOL`	jméno a verze protokolu, kterým přišel požadavek (typicky `HTTP/1.0` nebo `HTTP/1.1`)
`GATEWAY_INTERFACE`	označení a verze použitého rozhraní ke spuštění skriptu (typicky `CGI/1.1`)
`REMOTE_HOST`	doménová adresa počítače, z nějž přišel požadavek
`REMOTE_ADDR`	IP-adresa počítače, z nějž přišel požadavek
`AUTH_TYPE`	způsob použité autorizace uživatele
`REMOTE_USER`	v případě, že byl uživatel autorizován, obsahuje tato proměnná jeho jméno
`REMOTE_IDENT`	informace o identitě získaná zpětným dotazem u klienta; tuto vlastnost příliš mnoho serverů nevyužívá

Obsah těchto a některých dalších proměnných si ukážeme v tabulce 8-1 na následující straně. Ukážeme si jejich obsah pro tři odlišné dotazy:

A. `http://localhost/cgi-bin/test.cgi`

B. `http://localhost/cgi-bin/test.cgi?parametry`

Proměnná	A	B	C
QUERY_STRING		parametry	
REQUEST_METHOD	GET	GET	GET
SCRIPT_NAME	/cgi-bin/test.cgi	/cgi-bin/test.cgi	/cgi-bin/test.cgi
PATH_INFO			/adresar/soubor.html
SERVER_NAME	localhost	localhost	localhost
SERVER_SOFTWARE	Apache/1.1.1	Apache/1.1.1	Apache/1.1.1
SERVER_PROTOCOL	HTTP/1.0	HTTP/1.0	HTTP/1.0
GATEWAY_INTERFACE	CGI/1.1	CGI/1.1	CGI/1.1
REMOTE_HOST	localhost	localhost	localhost
REMOTE_ADDR	127.0.0.1	127.0.0.1	127.0.0.1

Tab. 8-1: Význam některých proměnných předávaných CGI-skriptu

C. `http://localhost/cgi-bin/test.cgi/adresar/soubor.html`

Při praktickém vytváření CGI-skriptu již záleží jen na tom, jak jsme dobří programátoři. Podle proměnné `REQUEST_METHOD` zjistíme, zda máme příchozí parametry načíst ze standardního vstupu nebo z proměnné `QUERY_STRING`. Příchozí data musíme rozkódovat (rozložit na položky oddělené znakem '`&`', převést plus na mezery atd.) a pak již s nimi můžeme snadno pracovat.[2] My si zde žádnou praktickou ukázku složitějšího CGI-skriptu neuvedeme. Má to hned dva důvody. Za prvé naše knížka není zaměřena na jejich tvorbu, zmiňujeme se o nich pouze v souvislosti s jazykem HTML. Druhým problémem zůstává v jakém jazyce psát ukázkový skript — v Perlu, v C++ nebo snad jako dávkový soubor pro *Windows 95*. Motat čtenáři hlavu ještě nějakým neznámým programovacím jazykem (protože podle zákona schválnosti vyberu jistě ten, který neznáte), to se mi opravdu nechce. Myslím, že nám HTML úplně bude stačit.

8.5 Cookies aneb server si u nás nechal koláček

Pomocí CGI lze vytvářet opravdu zajímavé aplikace. Ovšem i CGI mají jistá nepříjemná omezení vyplývající z principu protokolu HTTP. Protokol HTTP je tzv. nestavový. Znamená to, že pro přenos každé stránky se otevírá nové zvláštní HTTP spojení, které se ihned po přenosu uzavře. Server a potažmo CGI-skript nemá tedy moc šancí zjistit, zda jej nějaký uživatel spouští poprvé nebo podesáté.

[2] Práci nám může usnadnit skript *UnCGI*. Ten se umístí do cesty před náš skript, kterému předá obsah všech parametrů v proměnných prostředí `WWW_`*«parametr»*. Ušetříme si tak spoustu problémů spojených s analýzou parametrů. Náš skript se pak místo `/cgi-bin/`*«skript»* bude volat pomocí `/cgi-bin/uncgi/`*«skript»*. *UnCGI* je možno získat na adrese `http://www.mindwinter.com/~koreth/uncgi.html`.

Částečnou možností řešení tohoto problému je ukládání pomocných stavových informací do skrytých polí formuláře a nebo tyto informace přidávat do cesty v URL za jméno CGI-skriptu. Tyto techniky jsou využívány poměrně často a mnoho problémů uspokojivě vyřeší. Ovšem ani tyto dva způsoby neřeší problém trvalého uložení nějakých informací. Když totiž prohlížeč příští den spustíme znovu, server už o tom, že jsem po něm včera brouzdali, vůbec neví. Tento problém řeší *koláčky — cookies*, což je rozšíření protokolu HTTP z dílny firmy Netscape.[3] Nejprve se podíváme na to, co nám cookies nabídnou z uživatelského hlediska. Pak si ukážeme, jak cookies využít v našich CGI-skriptech a ASP.

Pokud si chceme prohlédnout stránku uloženou na některém serveru, může server v odpovědi na náš požadavek zaslat i informaci, kterou má klient (prohlížeč) uložit pro další použití. Pokud je pak někdy v budoucnosti navazováno spojení se stejným serverem, jsou mu tyto informace zaslány zpět. Informace jsou ukládány v textovém souboru, který se nejčastěji jmenuje `cookies.txt` a bývá uložen ve stejném adresáři jako prohlížeč. Jde o obyčejný textový soubor, a proto si jej můžeme prohlédnout téměř libovolným textovým editorem.

Cookies nejsou vázány na naše jméno ani e-mailovou adresu. Jsou společné pro jednu instalaci prohlížeče. Server se naše jméno ani další soukromé údaje z cookies nemůže dozvědět. Jedinou možností by bylo vyplnění těchto údajů do nějakého formuláře na serveru a zaslání těchto údajů jako cookies zpět prohlížeči pomocí CGI-skriptu, který obsluhuje formulář. Cookies tedy neumožňují přenos uživatelského jména a dalších osobních údajů bez vědomí uživatele.

Server si může například uchovávat informace o tom, jaké stránky jsme navštívili a kolik času jsme strávili jejich prohlížením. Tyto informace mohou být využity při statistickém vyhodnocování návštěvnosti jednotlivých stránek serveru. Mohly by být využity i nějakou marketingovou společností, pokud by stránky obsahovaly např. nabídky zboží a služeb či inzerci. V tomto případě bychom již mohli diskutovat o tom, zda je využívaní těchto informací legitimní. Vzhledem k tomu, že bez našeho vědomí však nelze získat naši adresu, nemůže se nám stát, že by z ničeho nic naši poštovní schránku zavalily nabídky sekaček na trávu.

Mnohem užitečnější se jeví využití cookies při ukládání konfiguračních informací. Server si tak může zjistit naše posledně použité nastavení těch WWW-stránek, které byly generovány dynamicky.

Jako reakce na tlak uživatelů je ve většině dnešních prohlížečů možno podporu cookies vypnout, a to jak napořád, tak i jen pro jednu relaci.

---

[3] Cookies je speciální druh sušenek, které při troše dobré vůle mohou připomínat i koláčky. Koláček je pěkné libozvučné slůvko, a proto jej občas použijeme.

Poněkud problematičtější je zajištění informací uložených pomocí cookies před servery, které k nim nemají mít přístup (tj. nevytvořily je). I když prohlížeč vrátí cookies pouze tomu serveru, který je uložil, existují metody, jejichž použitím se server může prokazovat jako nějaký jiný server.

Cookies mohou být vázány i na konkrétní podadresáře serveru, nemusí tedy být společné pro jeden server (doménu). Cookies se přenáší pomocí protokolu HTTP. Specifikace vyžaduje, aby byl klient schopen uložit alespoň 300 cookies po 4 KB a alespoň 20 cookies pro jeden server, případně doménu. Velikost souboru `cookies.txt` tedy obvykle nebude větší než 1,2 MB.

Jak vidíme, při používání cookies je potřeba provádět dvě operace: ukládat si cookies na klientovi a číst cookies zaslané klientem spolu s požadavkem na nějaké URL.

Uložení koláčku na klientovi

Cookies se klientovi posílají v HTTP hlavičce. Obecný formát je

`Set-Cookie:` *«jméno»*`=`*«hodnota»*`; expires=`*«datum»*`; path=`*«cesta»*`;`
`domain=`*«doménová adresa»*`; secure`

Jedinou povinnou částí je přitom *«jméno»*`=`*«hodnota»*. Ta slouží k nastavení koláčku se jménem *«jméno»* na hodnotu *«hodnota»*.

«jméno»`=`*«hodnota»*
Slouží k nastavení cookie na určitou hodnotu. Pokud později přijde od serveru cookie se stejným jménem ale jinou hodnotou, má cookie tuto novou hodnotu. Celý řetězec nesmí obsahovat středník a mezery. V případě, že je potřebujeme, musíme je překódovat pomocí procentové notace podobně jako v URL. Nastavení koláčku `Jmeno` na hodnotu `Jan Novák` tedy provedeme následovně:

```
Set-Cookie: Jmeno=Jan%20Novák
```

protože hexadecimálně vyjádřený kód znaku mezera je 20.

`expires=`*«datum»*
Atribut `expires` určuje, dokdy má cookie platnost. Po určeném datu se cookie neukládá ani se neposílá serveru. Formát data si ukážeme na čase oběda v pondělí 20. ledna 1997:

```
Mon, 20-Jan-97 12:00:00 GMT
```

Tento atribut lze použít i k vymazání už nepotřebné cookie. Pokud zašleme cookie s dobou platnosti v minulosti, prohlížeč na cookie zapomene.

`domain=`*«doménová adresa»*
Pokaždé, když se prohlížeč chystá odeslat cookies, porovnává doménovou adresu z URL, které má být vyvoláno, s atributem `domain`. Při porovnávání

stačí, aby `domain` bylo částí doménové adresy. Pokud by mohla být cookie zaslána díky shodě domén, je provedeno ještě porovnání cesty (atribut `path`). Pokud bude tedy `domain=acme.com`, vyhoví adresy `anvil.acme.com` i `shipping.create.acme.com` a může se přistoupit k porovnání cesty.

Pokud atribut `domain` nenastavíme, použije se doménová adresa serveru, který cookie prohlížeči poslal.

`path=`*«cesta»*
Pomocí tohoto atributu určujeme tu část URL v doméně, pro kterou je cookie platná. Jestliže cookie vyhověla porovnání domén, provede se porovnání cest. Obsah atributu `/kolo` vyhoví např. těmto cestám v URL `/kolo/author.html`, `/kolowrat`. Pokud atribut nastavíme na `/`, vyhoví to všem URL, kde se shoduje doména.

Pokud atribut `path` nespecifikujeme, použije se cesta z URL dokumentu, s nímž byla cookie zaslána.

`secure`
Pokud byla cookie zaslána prohlížeči s tímto atributem, bude na server poslána zpět pouze v případě, že spojení je bezpečné. To prakticky znamená, že spojení probíhá pomocí SSL (Secure Socket Layer), což je specifikace zabezpečené komunikace mezi klientem a serverem. Většinou ji podporují pouze komerční servery.

Pro zaslání cookies v CGI-skriptu tedy stačí přidat do HTTP-hlavičky jednu nebo více položek `Set-Cookie`.

V ASP máme dvě možnosti. Jednak můžeme použít obecnou metodu `Response.AddHeader` k přidání cookie do hlavičky odpovědi:

```
<% Response.AddHeader "Set-Cookie", "Jmeno=Jan%20Novák" %>
```

Druhou, většinou používanější metodou, je využití kolekce `Response.Cookies`. K přidání naší známe cookie s dobou expirace 5. července 1998 můžeme použít následující kód:

```
<% Response.Cookies("Jmeno") = "Jan Novák"
   Response.Cookies("Jmeno").Expires =
                              "Sat, 05-Jul-98 00:00:00 GMT" %>
```

Ochutnání koláčku

Klient s každým požadavkem na dokument o určitém URL posílá i všechny vyhovující cookies. Ty jsou posílány jako součást HTTP hlavičky v následujícím tvaru:

`Cookie:` *«jméno1»*`=`*«hodnota1»*`;` *«jméno2»*`=`*«hodnota2»*`; ...`

Pro nás je však důležitější, že takto zaslanou hlavičku předá server našemu CGI-skriptu v proměnné `HTTP_COOKIE`. Odtud si můžeme cookies přečíst a naložit s nimi dle vlastního uvážení.

Pokud používáme ASP, máme obsah proměnné `HTTP_COOKIE` uložen v kolekci `Request.ServerVariables`. Dostaneme se k němu pomocí konstrukce `Request.ServerVariables("HTTP_COOKIE")`. Kromě toho ASP nabízí i pohodlnější přístup. V kolekci `Request.Cookies` jsou jednotlivé cookie dostupné pomocí svého jména. Výraz `Request.Cookies("`*«jméno2»*`")` tedy vrátí *«hodnota2»*.

Pečeme koláčky

Použití cookies si samozřejmě ukážeme i prakticky. Vytvoříme CGI-skript, který bude vytvářet speciální stránku. Tato stránka bude ukazovat, kolikrát jsme ji již navštívili.[4] Navíc nás při prvním navštívení uvítá trošku jiným textíkem. CGI-skript opět napíšeme v příkazovém interpretu `sh`:


```
#!/bin/sh

pocet=‘echo $HTTP_COOKIE | sed ’s/pocet=//’‘
if [ -z $pocet ]; then
  pocet=0
fi
pocet=‘expr $pocet + 1‘

echo ’Content-type: text/html’
echo "Set-Cookie: pocet=$pocet"
echo
echo ’<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">’
echo ’<HTML>’
echo ’<TITLE>Koláčkový server</TITLE>’
echo ’</HEAD>’
```

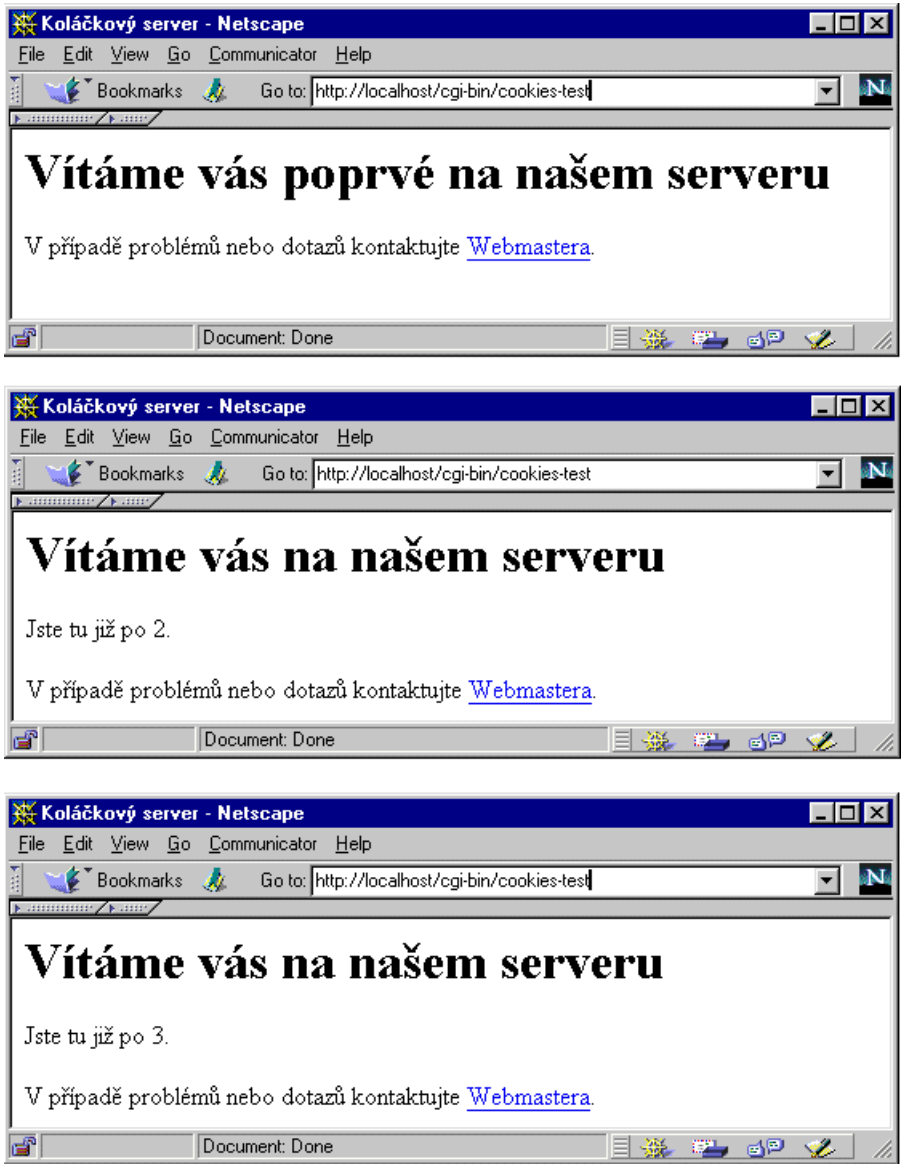
[4] Pozor! Nejedná se o známe počítadlo přístupů. To ukazuje celkový počet přístupů všech uživatelů na stránku. Naše stránka bude ukazovat počet přístupů každému uživateli zvlášť. Klasickými počítadly přístupů se budeme zabývat v následující sekci.

```
echo '<BODY>'

if [ $pocet = 1 ]; then
  echo '<H1>Vítáme vás poprvé na našem serveru</H1>'
else
  echo '<H1>Vítáme vás na našem serveru</H1>'
  echo "Jste tu již po $pocet."
fi

echo '<P>V případě problémů nebo dotazů kontaktujte'
echo '<A HREF="mailto:webmaster@server.cz">Webmastera</A>.'
echo '</BODY>'
echo '</HTML>'
```

Celý skript jistě zaslouží malé vysvětlení. Třetí šílená řádka nedělá nic jiného, než že z proměnné `HTTP_COOKIE` vezme pouze hodnotu koláčku `pocet`. Následující podmínka testuje, zda cookie byla klientem zaslána. V případě že ne, nastaví se počet dosavadních přístupů na 0. Pak tento počet aktualizujeme — zvýšíme jej o jedničku. Pak již generujeme odpověď. Kromě obvyklé hlavičky `Content-type` zašleme i aktualizovanou hodnotu cookie `pocet`. V následně generované stránce máme ještě podmínku, která vygeneruje trošku odlišný dokument při prvním přístupu na stránku. Na obrázku 8-4 na následující straně si můžeme prohlédnout, jak dopadnou první tři požadavky na URL, které ukazuje na náš skript.

Stejný efekt lze samozřejmě dosáhnout i pomocí jednoduché ASP:

```
<%
   If IsEmpty(Request.Cookies("pocet")) Then
     pocet = 0
   Else
     pocet = CInt(Request.Cookies("pocet"))
   End If
   pocet = pocet + 1
   Response.Cookies("pocet") = CString(pocet)
%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">
<HTML>
<TITLE>Koláčkový server</TITLE>
</HEAD>
<BODY>

<% If pocet = 1 Then %>
<H1>Vítáme vás poprvé na našem serveru</H1>
```



Obr. 8-4: Výstup CGI-skriptu využívajícího cookies

```
<% Else %>
<H1>Vítáme vás na našem serveru</H1>
Jste tu již po <%= pocet %>.
<% End If %>
```

```
<P>V případě problémů nebo dotazů kontaktujte
<A HREF="mailto:webmaster@server.cz">Webmastera</A>.
</BODY>
</HTML>
```

Naše skripty bychom měli navrhnout tak, aby fungovaly (klidně s omezeným komfortem) i když klient cookies nepodporuje. Uživatelé, kteří mají cookies vypnuty nebo mají starší prohlížeč, by neměli být diskriminováni.

8.6 Počítadla přístupů

Velice oblíbeným doplňkem každé stránky je počítadlo přístupů. To ukazuje, kolikrát již byla stránka navštívena. Tento údaj je zajímavý jak pro návštěvníky stránky, tak i pro samotného autora. My si teď ukážeme dva přístupy k integraci počítadla přístupu do našich stránek.

Počítadlo jako serverem vkládaná vsuvka

Pěkným příkladem počítadla založeného na SSI je program *access_counts* od Chucka Muscianiho. Program v podobě zdrojového textu v jazyce C si můžeme stáhnout na adrese


```
http://members.aol.com/htmlguru/access_counts.html
```

Program stačí zkompilovat libovolným kompilátorem jazyka C a umístit do vhodného adresáře — např. `/usr/local/http/bin`. Do stránek v místě, kde chceme mít zobrazen počet přístupů, pak vložíme SSI pro vyvolání programu:

```
Jste
<!--#exec cmd="/usr/local/http/bin/access_count"-->.
návštěvník naší stránky.
```

Celý program pracuje na velmi jednoduchém principu. V souboru má pro každou stránku uložen počet přístupů. Při svém vyvolání tento počet o jedničku zvýší a zároveň na svůj výstup zapíše tuto novou hodnotu.

Toto řešení má dvě velké výhody. Za prvé je velmi úsporné k přenosové kapacitě sítě — počet přístupů se přenáší jako pár znaků. Druhou výhodu ocení především uživatelé používající textový prohlížeč *Lynx*. Číslo vyjadřující počet přístupů je přímo součástí stránky a lze jej zobrazit i v textovém režimu.

Nutným předpokladem pro používaní tohoto počítadla je povolení provádění SSI správcem serveru. Pokud máme počítadlo přístupu umístěno na většině stránek, je asi výhodnější nakonfigurovat server tak, aby SSI hledal ve všech `.html` souborech a ne jen v `.shtml`.

Počítadlo jako CGI-skript

Tato metoda je dnes mnohem používanější než ta předchozí. Počet přístupů je do stránky vkládán jako obrázek. Tento obrázek je však generován pomocí CGI-skriptu a odráží tedy vždy aktuální počet přístupů. Když prohlížeč narazí na tag `<IMG>` snaží se stáhnout obrázek, jehož URL je určeno atributem `SRC`. Toto URL však neukazuje na soubor s obrázkem, ale na CGI-skript, který zvýší čítač a následně jeho obsah převede do obrázku, který je vrácen klientovi.

Mezi zástupce této skupiny patří program *WWWcounter* od Muhammada A. Muquita. Získat jej lze na adrese

`http://www.fccc.edu/users/muquit/Count.html`

Program je dodáván jako zdrojové texty v jazyce C a kromě Unixu jej lze použít i ve Windows NT a OS/2.

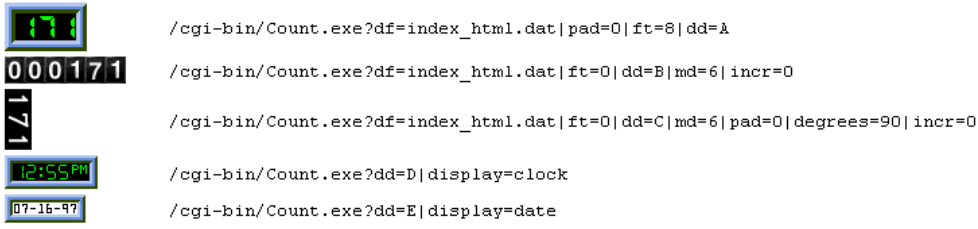
Program pracuje tak, že pro každé počítadlo má ve stanoveném adresáři uložen datový soubor, ve kterém je zapsán dosavadní počet přístupů. Ke vložení čítače do stránky pak stačí použít následující konstrukci:[5]

`<IMG SRC="/cgi-bin/Counter.exe?df=`*«datový soubor»*`">`

«datový soubor» je přitom jméno souboru, ve kterém je uložena hodnota čítače. Datový soubor musíme vytvořit v příslušném adresáři programu (nejčastěji `data`) a uložit do něj počáteční počet přístupů. Datové soubory lze nechat vytvářet i automaticky, ale v případě překlepu ve jménu datového souboru budou počty přístupů zkresleny.

☺ Jména datových souborů je vhodné vytvářet jednotným způsobem, který zaručí, že různé stránky nebudou používat stejné datové soubory. Bude-li počítadlo používáno pouze na jednom serveru, což je asi nejlepší řešení, stačí jednotně převádět cestu a jméno souboru. Datový soubor pro počítadlo stránky s adresou `http://www.server.cz/~jnovak/info/cenik.html` by tedy mohl mít jméno `jnovak_info_cenik_html.dat`. Vidíme, že všechny tečky a lomítka převedeme na podtržítko a znak vlnka (`~`) vynecháme. Pokud by počítadlo bylo využíváno i pro stránky z jiných serverů, do jména datového souboru přidáme i adresu serveru (`www_server_cz_jnovak_info_cenik_html.dat`).

[5] Pokud je čítač umístěn na jiném serveru než stránka, musíme použít kompletní URL i se jménem protokolu a serveru. Rovněž se některých systémech může skript jmenovat `Count.cgi` nebo jen `Count`.



Obr. 8-5: Modifikace vzhledu počítadla pomocí parametrů

Za jméno datového souboru lze umístit několik parametrů, které ovlivňují chování a vzhled počítadla. Jednotlivé parametry se oddělují buď rouřítkem (`|`) nebo ampersandem (`&`). Kompletní přehled parametrů nalezneme v dokumentaci počítadla. My se seznámíme alespoň s těmi nejdůležitějšími.

Pomocí parametru `ft` lze určovat šířku rámečku okolo počítadla. Použijeme-li `ft=0`, rámeček okolo počítadla zmizí. Vizuální vzhled počítadla řídí i parametr `md`. Ten určuje maximální počet zobrazených číslic. Pokud zároveň použijeme `pad=1`, počet přístupů se zleva doplní nulami tak, aby výsledná délka odpovídala hodnotě parametru `md`.

Zajímavé možnosti nabízí parametr `display`. Použijeme-li `display=date`, místo počtu přístupů se zobrazí aktuální datum na serveru. Zobrazit lze i čas — `display=clock`.

Pokud chceme, aby se počítadlo zvýšilo, ale nezobrazilo se na stránce, můžeme použít `sh=0`. Pokud naopak chceme, aby se počítadlo zobrazilo, ale počet přístupů se nezvyšoval, použijeme `incr=0`.

Pomocí parametru `dd` lze určovat podadresář adresáře `digits`, ze kterého se budou brát obrázky jednotlivých číslic. Standardně je s programem dodáváno pět sad (`A`–`E`). Další sady číslic lze získat na serveru Digit Mania — `http://www.digitmania.holowww.com/`.

Veřejná počítadla

Počítadla jsou atraktivní, ale pro jejich používání je potřeba správně nainstalovat a nakonfigurovat CGI-skript. Může se tedy snadno stát, že na serveru, kde máme naše stránky, není žádné počítadlo k dispozici. V těchto případech můžeme použít některé z veřejných počítadel. Veřejná počítadla jsou implementována CGI-skriptem, který je zpřístupněn všem uživatelům Internetu. U nás nalezneme veřejné počítadlo na adrese `http://pocitadlo.pinknet.cz/`. Stránka obsahuje i český návod k používání tohoto počítadla na našich stránkách.

9. Formuláře

Formuláře slouží k získání informací od uživatele. Získané informace jsou následně odeslány na server ke zpracování. Ke zpracování údajů z formuláře se používají CGI-skripty a ASP. Z toho vyplývá, že vytváření formulářů není už taková hračka. Musíme vytvořit stránku s formulářem a ještě skript, který správným způsobem zpracuje zaslaná data. V této kapitole se naučíme do stránky vkládat formuláře a povíme si, v jakém formátu jsou data z formuláře předávána CGI-skriptu.

9.1 Definice formuláře

Formulář je část stránky, která se běžnému uživateli jeví podobně jako dialogové okno. Formulář může obsahovat různá vstupní pole, tlačítka a přepínače.

Do stránky se formulář vkládá pomocí elementu `FORM`. Formulářů může být na jedné stránce více, ale nesmí být do sebe vnořeny. Pro správnou práci formuláře jsou důležité atributy `ACTION` a `METHOD`. První z atributů slouží k určení URL CGI-skriptu, který bude obsah formuláře zpracovávat. `METHOD` určuje metodu, která se při odeslání dat použije. Metody existují dvě (známe je z předchozí kapitoly) — `GET` a `POST`. Pro větší formuláře bychom měli používat výhradně metodu `POST`. Torzo formuláře tedy vypadá takto:

```
<FORM ACTION="http://«server»/cgi-bin/«skript»" METHOD=«metoda»>
«definice prvků formuláře»
</FORM>
```


Pokud nemáme možnost vytvářet CGI-skripty a přesto toužíme po formuláři, který nám občas někdo odešle, můžeme v atributu `ACTION` použít URL se schématem `mailto`.[1] V tomto případě bude obsah formuláře zaslán elektronickou poštou na zadanou adresu:

```
<FORM ACTION="mailto:xkosj06@vse.cz" METHOD=POST>
«definice prvků formuláře»
</FORM>
```

Posledním atributem, který lze u elementu `FORM` použít, je `ENCTYPE`. Ten slouží k určení způsobu kódování obsahu formuláře. Jeho standardní hodnota je `application/x-www-form-urlencoded` a pokud se nesetkáme s mimozemšťany, asi není důvod, proč tuto hodnotu měnit.

[1] Bohužel tato vlastnost, která je součástí standardu HTML 3.2, nefunguje v *Internet Exploreru 3.x* správně.

V takto vytvořeném torzu formuláře můžeme míchat běžné elementy s elementy pro jednotlivé prvky formuláře a vytvořit tak požadovaný formulář. K dispozici máme následující tři elementy pro prvky formuláře:

- `INPUT` slouží pro většinu prvků — vstupní pole, pole pro zadání hesla, zaškrtávací pole (checkboxes), přepínací tlačítka (radio buttons), tlačítka pro odeslání a smazání formuláře, skrytá pole, odeslání souboru a tlačítka s obrázkem;
- `SELECT` umožňuje vytvořit seznamy, ze kterých je možno vybírat jednu i více položek;
- `TEXTAREA` slouží k vytvoření vstupního pole pro víceřádkový text.

9.2 Element INPUT

Tento element ve formuláři použijeme nejčastěji. Víme, že slouží k vytvoření různých druhů vstupních prvků. Druh prvku určíme pomocí atributu `TYPE`. Každý prvek formuláře musí mít své jedinečné jméno, které je určeno atributem `NAME`. Toto jméno je později využito při odesílání obsahu formuláře serveru. Data jsou totiž odesílána v následujícím formátu:

«jméno1»=*«hodnota1»*&*«jméno2»*=*«hodnota2»*&...

«jméno1» a *«jméno2»* jsou přitom jména vstupních prvků určená právě atributem `NAME`. Podívejme se tedy na jednotlivé druhy vstupních prvků.

Vstupní pole pro krátký text (TYPE=TEXT)

Vstupní pole slouží k zadání krátkého textu, např. klíčových slov, jména či adresy elektronické pošty. Pokud u elementu `INPUT` neuvedeme atribut `TYPE`, předpokládá se, že jde o tento druh vstupního prvku.

Maximální délka zadávaného textu není omezena, můžeme ji však omezit použitím atributu `MAXLENGTH`. Velikost vstupního pole lze určit atributem `SIZE` — jako hodnota se udává počet znaků, které má pole pojmout. Pokud je `MAXLENGTH` větší `SIZE` nic se neděje — zadávaný text bude ve vstupním poli rolovat. Pokud chceme, aby v poli byla nějaká počáteční hodnota, můžeme ji specifikovat pomocí atributu `VALUE`. Jednoduchý formulář pro zadání jména tedy vytvoříme velice snadno:

```
<FORM ACTION="/cgi-bin/obsluha.cgi" METHOD=GET>
Jméno: <INPUT TYPE=TEXT NAME=Jmeno
              VALUE="Sem napište své jméno">
</FORM>
```

Jméno: Sem napište své jméno

Vidíme, že popisky k jednotlivým polím formuláře je potřeba zapsat zcela klasicky. Vstupní pole vyhradí pouze prostor na samotné zapsání hodnoty. Možnost míchat dohromady prvky formuláře s ostatními elementy lze šikovně využít. Pokud chceme mít jednotlivá políčka formuláře pěkně zarovnaná, můžeme použít tabulku:

```
<FORM ACTION="/cgi-bin/obsluha.cgi" METHOD=GET>
<TABLE>
<TR><TD>Jméno:     <TD><INPUT TYPE=TEXT NAME=Jmeno SIZE=30>
<TR><TD>Příjmení: <TD><INPUT TYPE=TEXT NAME=Prijmeni SIZE=30>
<TR><TD>E-mail:    <TD><INPUT TYPE=TEXT NAME=Email SIZE=20>
</TABLE>
</FORM>
```

Jméno:
Příjmení:
E-mail:

Heslo (TYPE=PASSWORD)

Použití a chování tohoto prvku je stejné jako u textového pole (`TYPE=TEXT`). Pouze při psaní do pole jsou místo znaků zobrazovány hvězdičky. Nikdo nám tak nemůže přes rameno přečíst heslo.

9

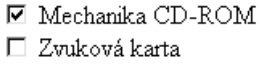
Zaškrtávací pole (TYPE=CHECKBOX)

Tento prvek slouží pro vstup logických hodnot. Prohlížečem bývá zobrazován jako okénko, které je buď prázdné nebo zaškrtnuté. Kromě jména (`NAME`) musíme vždy určit i atribut `VALUE`. Pokud je pole zaškrtnuté, pošle se jeho jméno společně s hodnotou atributu `VALUE` serveru. Pokud pole není zaškrtnuto, neposílá se z tohoto prvku formuláře nic.

Pokud chceme, aby bylo pole zaškrtnuto ihned po načtení stránky, použijeme atribut `CHECKED`.

V jednom formuláři můžeme použít více zaškrtávacích polí se stejným jménem. V tomto případě může uživatel zaškrtnout libovolné z těchto polí nezávisle na sobě. Na server jsou posílány dvojice jméno a hodnota všech zaškrtlých polí (se stejným jménem tedy může dorazit více položek). Malá ukázka: Dejme tomu, že formulář obsahuje dvě následující pole:

```
<INPUT TYPE=CHECKBOX NAME=konfigurace VALUE=CD-ROM CHECKED>
Mechanika CD-ROM<BR>
<INPUT TYPE=CHECKBOX NAME=konfigurace VALUE=SOUND>
Zvuková karta
```



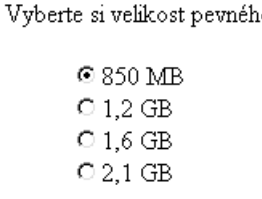
První pole je již zaškrtnuto a druhé zaškrtne uživatel. Pokud v tomto okamžiku uživatel odešle formulář, součástí dat odesílaných na server bude i následující řetězec:

```
konfigurace=CD-ROM&konfigurace=SOUND
```

Přepínací tlačítka (TYPE=RADIO)

Tento typ vstupního prvku použijeme v případě, kdy chceme uživateli nabídnout možnost výběru právě jedné z několika variant. Všechny varianty musí mít stejné jméno (`NAME`) a rozdílnou hodnotu (`VALUE`). Jedna z variant musí být označena pomocí atributu `CHECKED`. Serveru se posílá jen dvojice jméno a hodnota vybrané varianty.

```
Vyberte si velikost pevného disku:
<BLOCKQUOTE>
<INPUT TYPE=RADIO NAME=disk VALUE=850 CHECKED>850 MB<BR>
<INPUT TYPE=RADIO NAME=disk VALUE=1200>1,2 GB<BR>
<INPUT TYPE=RADIO NAME=disk VALUE=1600>1,6 GB<BR>
<INPUT TYPE=RADIO NAME=disk VALUE=2100>2,1 GB
</BLOCKQUOTE>
```



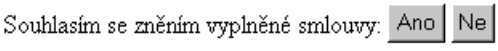
Tlačítko pro odeslání formuláře (TYPE=SUBMIT)

Použijeme-li element `INPUT` ve spojení s atributem `TYPE=SUBMIT`, vytvoříme tlačítko, jehož stisk bude sloužit k odeslání formuláře serveru. Na tlačítku se objeví nápis, který specifikujeme pomocí atributu `VALUE`:

```
<INPUT TYPE=SUBMIT VALUE="Odeslání dotazníku">
```

Pokud u odesílacího tlačítka definujeme i atribut `NAME`, můžeme v jednom formuláři použít několik odesílacích tlačítek. S daty formuláře se totiž odešle pár jméno/hodnota odpovídající stisknutému tlačítku.

```
Souhlasím se zněním vyplněné smlouvy:
<INPUT TYPE=SUBMIT NAME="Potvrzeni" VALUE="Ano">
<INPUT TYPE=SUBMIT NAME="Potvrzeni" VALUE="Ne">
```



Tlačítko pro vynulování (TYPE=RESET)

Po stisku tohoto tlačítka se všechna pole formuláře nastaví na původní hodnoty. Popis tlačítka lze určit pomocí atributu `VALUE`. Toto tlačítko nikdy není posíláno zpět serveru jako součást dat formuláře.

Tlačítka s obrázkem (TYPE=IMAGE)

Toto tlačítko slouží k odeslání formuláře podobně jako tlačítko `SUBMIT`. Místo textu je však na tlačítku umístěn obrázek, jehož URL zadáme pomocí atributu `SRC`. Zarovnání obrázku s okolím můžeme ovlivnit atributem `ALIGN`, který má stejný význam jako u elementu `IMG` (viz strana 59).

Tlačítko s obrázkem funguje podobně jako klikací mapa — s ostatními daty formuláře jsou odeslány i informace o místě, kde došlo ke kliknutí. Místo kliknutí je odesláno ve dvou položkách. Za jméno tlačítka je doplněno `.x` respektive `.y` a jako hodnota se odešle souřadnice x resp. y.

```
<INPUT TYPE=IMAGE NAME=obr SRC="mapa.gif">
```

Byl-li formulář odeslán kliknutím na pozici $x = 25$ a $y = 32$, pak součástí dat posílaných serveru bude i `obr.x=25&obr.y=32`.

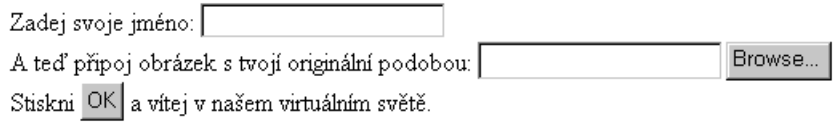
☺ Není-li nezbytně nutné, používání těchto tlačítek ve formulářích se raději vyhneme. V textových prohlížečích je totiž takový formulář úplně k ničemu.

Odeslání souboru (TYPE=FILE)

Tento ovládací prvek použijeme v případech, kdy chceme uživateli umožnit odeslání souboru společně se zbytkem formuláře. Obvykle se prvek zobrazí jako vstupní textové pole a tlačítko. Do pole můžeme přímo vepsat jméno souboru. Můžeme proto použít atributy `SIZE` a `MAXLENGTH`, které mají stejný význam jako u polí s typem `TEXT`. Tlačítko slouží k vyvolání dialogu, který nám umožní pohodlné vybrání souboru.

Pomocí atributu `ACCEPT` můžeme určit MIME typy souborů, které je přípustné vybrat. Následující malý formulář může sloužit třeba pro přihlášení k nějakému chatu — zadáme svoje jméno a spolu s ním pošleme naši podobenku. Ta přitom může být buď obrázkem (`image/*`) nebo pseudoobrázkem poskládaným z ASCII-znaků (`text/plain`):

```
<FORM ACTION="/cgi-bin/logon" METHOD=POST
      ENCTYPE="multipart/form-data">
Zadej svoje jméno: <INPUT NAME=Jmeno SIZE=20><BR>
A teď připoj obrázek s tvojí originální podobou:
<INPUT TYPE=FILE NAME=Foto ACCEPT="image/*,text/plain"><BR>
Stiskni <INPUT TYPE=SUBMIT VALUE="OK"> a vítej v našem
virtuálním světě.
</FORM>
```



A je to tady, jsou tu mimozemšťané. Pro přenos souborů je klasická metoda kódování dat formuláře `application/x-www-form-urlencoded` nevhodná. Musíme tedy použít novější `multipart/form-data`. Každé pole formuláře je zde přenášeno jako jedna část MIME zprávy.

Skrytá pole (TYPE=HIDDEN)

Tato pole se ve formuláři neobjeví. Slouží k uchování stavové informace, která je odeslána s vyplněným formulářem zpět serveru. U pole použijeme pouze atributy `NAME` a `VALUE`, které slouží k definovaní stavové proměnné a její hodnoty. Ještě o krůček lepší funkčnost v uchovávání stavové informace nabízejí cookies (viz strana 143).

Skrytá pole skrytě

A teď malý bonbónek pro osvěžení ducha a zpestření našich stránek.

Na mnoha domovských stránkách firem či jednotlivců jsou odkazy na oblíbenou prohledávací službu autora stránky. Vytvořit takovýto odkaz není nic složitého. Pro AltaVistu můžeme použít následující kód:

```
<A HREF="http://altavista.digital.com/">Prohledávací služba
AltaVista</A>
```

Toto řešení sice funguje, ale mnohem hezčí by bylo, kdyby na naší stránce bylo rovnou políčko pro zadání dotazu a po jeho odeslání bychom dostali odpověď od AltaVisty. Odpadlo by tak natahování stránky pro zadání dotazu AltaVistě.

K vyřešení tohoto problému stačí, když se podíváme na to, jak se AltaVistě dotaz posílá: je předáván jako parametr v URL — vše, co je v URL uvedeno za znakem '?', je považováno za parametr dotazu. Dotaz „skalní lezení"[2] je do URL zakódován takto:

«začátek URL»`?pg=q&what=web&fmt=.&q=rock+climbing`

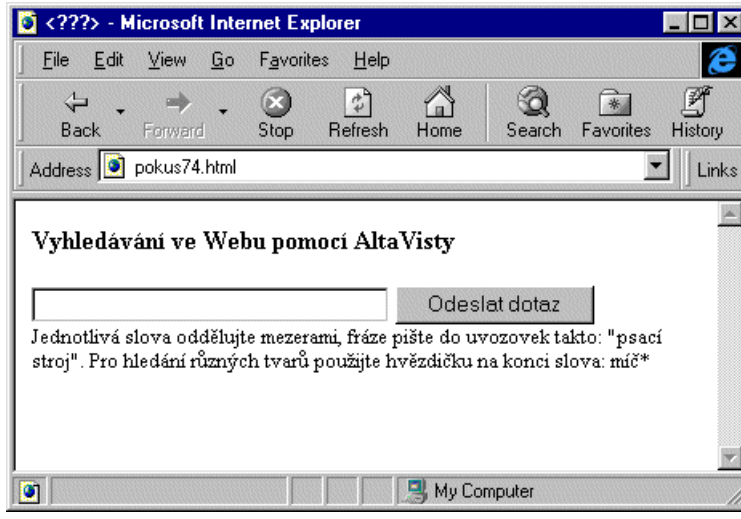
První přiřazení indikuje, že se jedná o stránku s dotazem; druhé určuje, co se bude prohledávat — zda Web či diskusní skupiny. Třetí přiřazení určuje formát zobrazení výsledků — v našem případě standardní. Za `q=` následuje již samotný dotaz. Abychom mohli takovéto URL vytvořit, musíme použít formulář. U jeho atributu `ACTION` nastavíme URL na AltaVistu. Pole, která určují druh dotazu, vytvoříme jako skrytá (typ `HIDDEN`). Formulář bude tedy obsahovat pouze jediné viditelné pole pro zadání hledaných slov (a samozřejmě tlačítko pro odeslání dotazu). Nebudeme tedy muset složitě volit druh dotazu a co chceme prohledávat. Konkrétní zápis v HTML může být následující:

```
<B>Vyhledávání ve Webu pomocí AltaVisty</B><BR>
<FORM ACTION="http://altavista.digital.com/cgi-bin/query"
      METHOD=GET>
<INPUT TYPE=HIDDEN NAME="pg" VALUE="q">
<INPUT TYPE=HIDDEN NAME="what" VALUE="web">
<INPUT TYPE=HIDDEN NAME="fmt" VALUE=".">
<INPUT TYPE=TEXT NAME="q" SIZE=30>
<INPUT TYPE=SUBMIT VALUE="Odeslat dotaz">
<BR><SMALL>
Jednotlivá slova oddělujte mezerami, fráze pište do uvozovek
takto: "psací stroj". Pro hledání různých tvarů použijte
hvězdičku na konci slova: míč*
```

[2] Angl. rock climbing — pozn. aut.

```
</SMALL>
</FORM>
```

Výsledek našeho „nečestného“[3] použití skrytých polí je na obrázku 9-1.



Obr. 9-1: Prohledávání Webu z vlastní stránky

9.3 Seznamy (element SELECT)

9

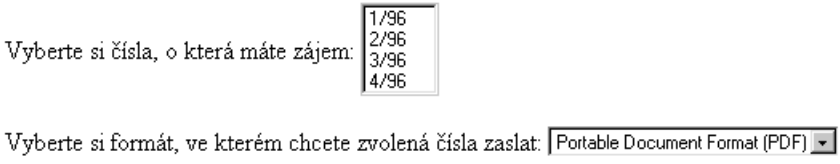
Tento element lze použít v případech, kdy chceme uživateli nabídnout výběr jedné z několika položek. Položky se uvádějí mezi tagy `<SELECT>` a `</SELECT>`. Pokud použijeme atribut `MULTIPLE`, může být ze seznamu najednou vybráno i více položek. Jako u většiny ostatních prvků formuláře, i zde musíme uvést jméno pomocí atributu `NAME`. Počet nejednou zobrazených řádek seznamu můžeme nastavit pomocí atributu `SIZE`.

Jednotlivé položky seznamu se uvádějí jako obsah elementu `OPTION`. Ukončovací tag však můžeme vždy vynechat. Položky, u nichž použijeme atribut `SELECTED`, budou již předem vybrány. Jako hodnota vybrané položky se se jménem seznamu přenáší obsah položky. Pokud je příliš dlouhý, můžeme pomocí atributu `VALUE` nastavit hodnotu, která bude identifikovat položku seznamu. Použití si ukážeme na malé ukázce:

```
Vyberte si čísla, o která máte zájem:
<SELECT NAME="Cisla" SIZE=4 MULTIPLE>
```

[3] Nečestného a nesportovního — vždyť Mirek Dušín by to nikdy neudělal.

```
<OPTION VALUE="1">1/96
<OPTION VALUE="2">2/96
<OPTION VALUE="3">3/96
<OPTION VALUE="4">4/96
</SELECT>

<P>Vyberte si formát, ve kterém chcete zvolená čísla zaslat:
<SELECT NAME="Format" SIZE=1>
<OPTION VALUE="HTML">HTML
<OPTION VALUE="PDF" SELECTED>Portable Document Format (PDF)
<OPTION VALUE="PS">PostScript
<OPTION VALUE="ASCII">obyčejný text
</SELECT>
```

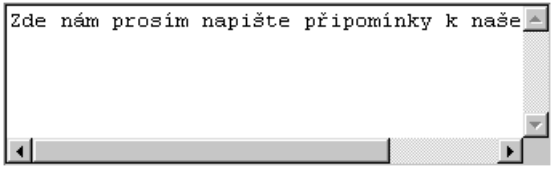


Seznam pro výběr formátu se po kliknutí na šipku dolů rozbalí do celé své krásy:

9.4 Víceřádkový text (element TEXTAREA)

Element `TEXTAREA` slouží pro zadávání delších textů ve formuláři. Jako všechny ostatní elementy má atribut `NAME`. Pomocí atributu `ROWS` můžeme určit počet řádek vstupního pole a pomocí `COLS` počet sloupců. Obsahem elementu je text, který se ve vstupním poli objeví na začátku editace (ukončovací tag tedy nesmí být vynechán):

```
<TEXTAREA NAME="Komentar" ROWS=5 COLS=40>
Zde nám prosím napište připomínky k našemu programu.
</TEXTAREA>
```



10. Design WWW-stránek

V předchozích kapitolách jsme se důkladně seznámili s jazykem HTML. To však bohužel ještě neznamená, že máme všechny předpoklady pro vytvoření perfektních WWW-stránek. HTML je jen prostředkem k zpřístupnění informací. Důležité je mít zajímavé informace a umět je uspořádat tak, aby byly pro čtenáře atraktivní a snadno přístupné. V této kapitole si ukážeme některé recepty, které naše stránky mohou vylepšit. Nejde však v žádném případě o přesný návod, který lze použít vždy. Jednotlivé myšlenky je potřeba přizpůsobit konkrétní situaci a požadavkům.

10.1 Organizace informací

Vždy, když zpřístupňujeme nějaké informace stojíme před základní otázkou, jak tyto informace logicky uspořádat do jednotlivých stránek a jak mají být tyto stránky velké.

Struktura stránek

Hypertext nám umožňuje informace uspořádat do téměř libovolné struktury. Měli bychom se však držet na zemi a nevytvářet džungli informací zamotanou liánami tak, že se z ní už nikdo nikdy nevyhrabe. Lidé jsou na některé způsoby strukturování informací zvyklí. Nejběžnější strukturou je hierarchie — známe ji z většiny knížek. Kniha se skládá z kapitol. Kapitola se zase skládá z jednotlivých sekcí atd.

Držet se tohoto členění i v našich stránkách se většinou jeví jako dobrý nápad. Uživatelé jsou na tento druh členění informací zvyklí. Vytvoříme tedy jednu stránku s obsahem, z které povedou odkazy na jednotlivé kapitoly a sekce. Každá kapitola a sekce by pak měla obsahovat odkazy na předcházející a následující kapitolu a sekci a odkaz na obsah.

Výše uvedené členění informací přivítají zejména uživatelé, kteří prezentované informace studují systematicky. Pro uživatele, kteří pouze občas vyžadují přístup k nějaké konkrétní informaci, můžeme vytvořit rejstřík, který bude obsahovat odkazy na jednotlivé sekce podle jejich obsahu — něco jako referenční příručku.

Uživatel se budé také snáze orientovat, budou-li mít všechny spolu související stránky stejný grafický design.

Délka stránek

Délka stránek by měla v první řadě vycházet ze struktury poskytovaných informací. Jedna stránka by měla obsahovat logicky související informace. Musíme však dát pozor na to, aby takováto stránka nebyla příliš dlouhá. Jednak se pak dlouho přenáší po síti (musíme započítat i dobu potřebnou na přenesení grafiky). Druhak je orientace v dlouhém textu pro čtenáře obtížná. V těchto případech se pokusíme jednu stránku rozdělit do několika menších.

Obecné pravidlo, které by rozlišovalo mezi dlouhým a krátkým dokumentem, neexistuje. Přesto — stránka s odkazy na další stránky by se měla vejít celá na obrazovku, aby uživatel pohromadě viděl všechny možnosti dalšího putování za informacemi. Dokumenty, jež obsahují text, by neměly být delší než několik obrazovek.

Na tomto místě si neodpustím ještě poznámku. Každý autor chce, aby jeho stránky byly atraktivní. Přidá na stránku různou grafiku, a později i animace. Animace bychom však měli používat velmi opatrně a to zejména na stránkách, které obsahují již nějaké delší úseky textu. Při čtení textu různá skotačící loga a rotující nesmysly rozptylují čtenáře a prodlouží čas potřebný k absorbování informací.

10.2 Odkazy

Odkazy jsou největším rozdílem publikování na Webu oproti běžným postupům na papír. Mnoho začátečníků s nimi také má problémy. My si ukážeme ty nejčastější chyby a omyly při jejich používaní. Samozřejmě nevynecháme ani příklady správného použití.

Klikněte zde

Na mnoha stránkách se setkáme s odkazy typu:

> Pro informace o nové verzi našeho programu *klikněte zde*.

Od takovýchto slovních hříček bychom se měli oprostit. Třeba proto, že někdo používá znakový prohlížeč *Lynx* a těžko v něm bude někde klikat myší. Nebo když si takovou stránku vytiskneme na papír, už si taky ani neklikneme. Lepší je použít něco jako:

> Od června je dostupná *nová verze programu*.

Vůbec bychom se na stránkách měli oprostit od používání termínů, které souvisejí s technologiemi Webu a Internetu. Běžného uživatele Webu nezajímá, co je to odkaz, WWW-server nebo FTP-archiv.

Relativní vs. absolutní odkazy

Při vytváření odkazu stojíme před problémem, zda jej vytvořit jako relativní nebo absolutní. Naštěstí na to existuje velice jednoduché pravidlo. Mezi vlastními souvisejícími stránkami bychom měli používat relativní odkazy. Nebude pak problém celou skupinu stránek přenést na jiný server. Oproti tomu odkazy na cizí zdroje a stránky by měly být absolutní, aby fungovaly i po případném přesunu našich stránek na jiný server.

Odkazy nebo kopie

Občas na našich stránkách máme informaci, ke které existují další podrobnosti někde jinde. Máme tedy dvě možnosti: (1) vytvořit na tuto informaci odkaz; (2) zkopírovat cizí stránku s informací k našim a vytvořit odkaz pouze na tuto kopii.

Pokud vytvoříme kopii, přiděláme si tím spoustu práce, protože musíme neustále kontrolovat, zda se originál nezměnil. Někdy můžeme kopii vytvořit v dobré víře, že z našeho serveru bude rychleji přístupná. Musíme si však uvědomit, že Internet je celosvětová síť a že někdo jiný má k danému zdroji třeba mnohem rychlejší přístup než my.

Samozřejmě, že existují situace, kdy se pořizování kopií vyplatí. Je to v případech, kdy se odkazujeme na dočasné informace (např. novinové články nebo dokument, který se vyvíjí, ale nearchivují se starší verze).

10.3 Dokument

Nyní si shrneme to, na co bychom u žádného dokumentu neměli zapomenout.

V první řadě bychom měli každý dokument podepsat, aby bylo zřejmé, kdo je autorem prezentovaných informací. Úplně postačí, když na konec dokumentu připojíme naše iniciály, které budou odkazem na naši domovskou stránku, kde případný zájemce nalezne naši adresu a e-mail.

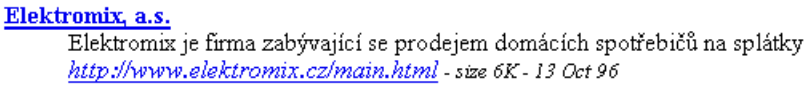
U každého dokumentu by rovněž byla být informace o tom, zda je úplný, kdy byl naposledy modifikován a dokdy platí. Některé z těchto informací lze do stránky nechat vložit serverem automaticky pomocí SSI.

Každý dokument by měl obsahovat odkaz na jemu nadřazený dokument, aby i náhodný návštěvník dokázal stránku zařadit do správného kontextu.

Znakem kvalitní stránky je i dobře zvolený název. Pokud chceme naši stránku lépe připravit pro vyhledávací stroje, můžeme ji doplnit o stručný popis a o klíčová slova. Využívá se k tomu element `META`, který se smí použít v záhlaví stránky pro připojení libovolné metainformace. Pomocí atributu `NAME` určíme jméno metainformace a atributem `CONTENT` pak samotný obsah metainformace:

```
<HEAD>
<TITLE>Elektromix, a.s. </TITLE>
<META NAME="description"
      CONTENT="Elektromix je firma zabývající se prodejem
               domácích spotřebičů na splátky">
<META NAME="keywords" CONTENT="Elektromix, prodej, elektrické
                               spotřebiče, leasing">
</HEAD>
```

Při zobrazení výsledků prohledávání nějakou vyhledávací službou bude naše stránka prezentována zadaným popisem místo mnohdy nic neříkajícího začátku textu stránky:



Kromě metainformací `description` a `keywords` se často používá i `author` — slouží k určení autora stránky.

```
<META NAME="Author" CONTENT="Jiří Kosek">
```

11. Čeština a Web

Ti z nás, kteří pamatují doby, kdy MS-DOS byl nejpoužívanějším operačním systémem na našich PC-286, si ještě pamatují na problémy s předáváním textových souborů v češtině. Tehdy se používala nejméně tři různá kódování češtiny a většinou nás to stálo trochu práce, než jsme z nečitelného dokumentu vyrobili krásný text hemžící se nádhernými českými znaky s diakritikou.

Tato chaotická situace vznikla celkem snadno. Původně MS-DOS podporoval pouze angličtinu a pár západoevropských jazyků, jejichž speciální znaky obsahovala druhá půlka kódové tabulky. Tento nevyhovující stav pak zaplnilo několik nezávisle vzniklých kódování češtiny. Podobný chaotický vývoj se nevyhnul ani češtině ve Webu. Podívejme se tedy, jak to s podporou češtiny vypadá.

11.1 Cestina bez hacku a carek

Asi nejjednodušší, i když zdaleka ne optimální řešení problematiky české diakritiky na Webu je její nepoužívání. Ve všech textech budeme používat pouze znaky anglické abecedy a diakritická znaménka budeme ignorovat.

Je až s podivem, kolik stránek na českém Internetu nalezneme pouze v ASCII. Obliba tohoto řešení spočívá v jeho naprosté technické nenáročnosti a bezproblémovosti. My bychom se však měli poohlédnout po nějakém lepším řešení, protože Web by měl usnadnit přístup k informacím a ne sloužit ke komolení jazyka.

11.2 Kódování ISO Latin 1

HTML 2.0 zavedlo jako standardní kódování ISO Latin 1 (ISO 8859-1). Kódování se ve své první půlce zcela shoduje s ASCII. V horní části pak nalezneme všechny speciální znaky, které jsou potřeba v nejrůznějších západoevropských jazycích. Bohužel z tohoto hlediska Čechy ještě do západní Evropy nepatří. ISO Latin 1 obsahuje např. znaky: ‘í’, ‘é’, ‘á’, ‘ý’, ale marně bychom zde hledali např. ‘ř’ a ‘š’.

Použijeme-li v dokumentu tedy kódování ISO Latin 1, můžeme mít v textu *některé* znaky i s diakritickými znaménky. Abychom mohli toto kódování využít, musíme mít editor, který s ním umí pracovat. Pokud takový editor nemáme,

můžeme znaky zapisovat jako znakové entity. Přehled entit a jim odpovídajících znaků nalezneme v tabulce 11-1 na následující straně.

Pokud do textu stránky budeme chtít zapsat slovo 'Jirásek', můžeme použít jeden z následujících dvou zápisů:

```
Jir&aacute;sek
Jir&#225;sek
```

Takovýto způsob zápisu je velmi nepohodlný, zvláště pro delší texty. Obejít to lze pomocí jednoduchého konverzního programku, který české znaky nahradí patřičnou znakovou entitou.

Povšimněme si ještě, že v ISO Latin 1 nalezneme i mnoho zajímavých znaků. Např. pomocí `&copy;` můžeme do textu stránky vložit znak pro vymezení autorských práv — ©. Znaková entita ` ` slouží k zápisu nedělitelné mezery. Nedělitelnou mezeru bychom měli psát například za neslabičné jednopísmenné předložky, aby nezůstávaly na koncích řádků:

```
Hurvínek našel Máničku v lese.
```

11.3 Čeština bez kompromisů

Předchozí dvě řešení byla sice technicky bezproblémová, ale český uživatel, který chtěl pomocí Webu informace ve své nezprzněné mateřštině, zůstal neuspokojen. Samozřejmě, že existují řešení, která umožní v dokumentech používat všechny české znaky s diakritickými znaménky. Jediným problémem zůstává velký počet různých používaných kódování češtiny. Podívejme se alespoň na ty nejrozšířenější.

CP 1250
Kódová stránka 1250 je pro psaní českých znaků používána ve všech verzích *Windows*. V tomto kódu jsou uloženy všechny fonty, jejichž název končí na 'EE' a 'CE' (ve *Windows 3.x*) nebo na 'středoevropský' či 'central european' (ve *Windows 95/NT*).

ISO 8859-2
Toto kódování je mezinárodně standardizováno. V praxi se používá zejména v operačních systémech založených na Unixu. Poslední verze *Netscape Navigatoru* a *Internet Exploreru* pro *Windows* si poradí i s tímto kódováním a zobrazí text správně česky.

V návrhu HTML 3.0 bylo ISO 8859-2 uvedeno jako kódování, které se má používat pro střední Evropu. Bohužel tento standard nebyl nikdy oficiálně přijat. Návrh standardu HTML 4.0 umožňuje používaní jakékoliv znakové sady uvedené v RFC-dokumentu 2045.

Kód	Entita	Znak
160		
161	&iexcl;	¡
162	&cent;	¢
163	&pound;	£
164	&curren;	¤
165	&yen;	¥
166	&brvbar;	¦
167	&sect;	§
168	&uml;	¨
169	&copy;	©
170	&ordf;	ª
171	&laquo;	«
172	&not;	¬
173	&shy;	-
174	&reg;	®
175	&macr;	¯
176	&deg;	°
177	&plusmn;	±
178	&sup2;	²
179	&sup3;	³
180	&acute;	´
181	&micro;	µ
182	&para;	¶
183	&middot;	·
184	&cedil;	¸
185	&sup1;	¹
186	&ordm;	º
187	&raquo;	»
188	&frac14;	¼
189	&frac12;	½
190	&frac34;	¾
191	&iquest;	¿
192	&Agrave;	À
193	&Aacute;	Á
194	&Acirc;	Â
195	&Atilde;	Ã
196	&Auml;	Ä
197	&Aring;	Å
198	&AElig;	Æ
199	&Ccedil;	Ç
200	&Egrave;	È
201	&Eacute;	É
202	&Ecirc;	Ê
203	&Euml;	Ë
204	&Igrave;	Ì
205	&Iacute;	Í
206	&Icirc;	Î
207	&Iuml;	Ï

Kód	Entita	Znak
208	&ETH;	Ð
209	&Ntilde;	Ñ
210	&Ograve;	Ò
211	&Oacute;	Ó
212	&Ocirc;	Ô
213	&Otilde;	Õ
214	&Ouml;	Ö
215	&times;	×
216	&Oslash;	Ø
217	&Ugrave;	Ù
218	&Uacute;	Ú
219	&Ucirc;	Û
220	&Uuml;	Ü
221	&Yacute;	Ý
222	&THORN;	Þ
223	&szlig;	ß
224	&agrave;	à
225	&aacute;	á
226	&acirc;	â
227	&atilde;	ã
228	&auml;	ä
229	&aring;	å
230	&aelig;	æ
231	&ccedil;	ç
232	&egrave;	è
233	&eacute;	é
234	&ecirc;	ê
235	&euml;	ë
236	&igrave;	ì
237	&iacute;	í
238	&icirc;	î
239	&iuml;	ï
240	&eth;	ð
241	&ntilde;	ñ
242	&ograve;	ò
243	&oacute;	ó
244	&ocirc;	ô
245	&otilde;	õ
246	&ouml;	ö
247	&divide;	÷
248	&oslash;	ø
249	&ugrave;	ù
250	&uacute;	ú
251	&ucirc;	û
252	&uuml;	ü
253	&yacute;	ý
254	&thorn;	þ
255	&yuml;	ÿ

Tab. 11-1: Znakové entity pro kódování ISO Latin 1

PC Latin 2
Toto kódování je standardem Microsoftu v DOSu (známe též jako kódová stránka 852). Stejné kódování používá i operační systém OS/2 od IBM.

Kódování bratrů Kamenických
Jedná se o původní český kód, často označovaný jako KEYBCS2. Používá se zejména v prostředí MS-DOSu.

Mac CE
Takto se často označuje kódování používané pro češtinu na počítačích Apple Macintosh.

Vidíme, že co operační systém, to jiné kódování češtiny. Většina prohlížečů umí správně zobrazit jen ty stránky, které přijdou v kódu používaném v operačním systému. Slušný autor stránek by tedy měl uživateli nabídnout možnost výběru požadovaného kódování češtiny. Kromě výše uvedených možností by měl nabídnout i verzi dokumentu bez háčků a čárek[1] v ASCII.

Pro každý kód jeden soubor

Asi první řešení, které člověka napadne, je pro každé kódování vytvořit jednu verzi dokumentu. Vystačíme si tedy s programem, který dokáže převádět soubor z jednoho kódování do druhého. Odlišné kódování souboru můžeme rozlišit buď použitím přípony nebo umístění každého kódování do zvláštní větve adresářového stromu.

Toto řešení má několik problémů. Prvním z nich jsou odkazy. Ve všech dokumentech musíme zajistit, aby odkazy vedly na verzi dokumentu ve správném kódování. Tuto práci nám může usnadnit program Jiřího Kvardy NTCPCONV, který je k dispozici na adrese:

```
http://web.cvut.cz/ascii/cc/icsc/software/
```

Druhou nevýhodou jsou vysoké nároky na diskovou kapacitu — každý dokument je uložen několikrát v různých kódováních. Poslední velkou nevýhodou je náročnost na správu dokumentů. Pokud v jednom dokumentu uděláme změnu, nesmíme zapomenout aktualizovat i verze téhož dokumentu v jiných kódováních.

Abychom nekončili tak pesimisticky — toto řešení má i jednu výhodu. Vzhledem k tomu, že dokumenty v jednotlivých kódováních jsou obyčejné stránky, mohou je proxy-servery uchovávat ve svých vyrovnávacích pamětech a urychlit k nim opakovaný přístup. Tento mechanismus bohužel nefunguje u dynamicky překódovaných dokumentů.

[1] a kroužků

Dynamická změna kódu

Největší nevýhodou předchozího řešení byla nutnost udržovat pro každý kód zvláštní verzi dokumentu. Pokud použijeme dynamickou změnu kódu, je na serveru dokument uložen pouze jednou. Pokud si uživatel vyžádá dokument v určitém kódu, je dokument pomocí CGI-skriptu do tohoto kódu převeden.

Nevýhodou tohoto řešení je větší zátěž serveru — ten musí pro každou přenášenou stránku spustit CGI-skript, který zajistí převod. Tím, že jsou dokumenty generovány dynamicky, nemohou být uchovávány ve vyrovnávacích pamětích proxy-serverů.

Programových balíků, které slouží pro dynamický převod kódu, existuje několik. Jejich přehled uvádí tabulka 11-2. My si ukážeme, jak se používá balík *SaCzech* od Pavla Satrapy.

Název	Adresa
C-SaCzech 1.23	`http://www.fi.muni.cz/~dolecek/c-saczech/`
compose 2.0	`http://www.cuni.cz/cgi-bin/cgiwrap/obo/compose/++CP1250/files/compose.html.iso-8859-2`
CzechWeb 1.1	`ftp://ftp.lepton.cz/PUB/UPLOAD/CzechWeb%201.1%20Demo.sit.hqx`
gw_convert.pl	`http://infox.eunet.cz/www/CONVERT.HTM`
kod 1.1	`ftp://cech.cesnet.cz/pub/local/`
SaCzech 2.0	`http://www.kin.vslib.cz/cgi-bin/whichcode/~satrapa/sw/saczech/saczech.html`
WWWdia	`ftp://service.felk.cvut.cz/pub/export/software/`

Tab. 11-2: Balíky podporující dynamickou změnu kódu

SaCzech aneb pytlík v akci

Podivný název tohoto odstavce je způsoben přáním autora programu *SaCzech*. Ten si přál, aby se jméno programu vyslovovalo jako „sáček“ nebo „pytlík“.

Informace o požadovaném kódování se v *SaCzechu* ukládá na začátek URL. *SaCzech* je tvořen několika skripty, které slouží k převodu dokumentu do určitého kódu. Tyto skripty mají název `to`*«kód»*. *SaCzech* podporuje následující kódy:

«kód»	kódování
`ASCII`	ASCII
`ISO-8859-2`	ISO Latin 2
`ISO-8859-1`	ISO Latin 1
`MAC`	Macintosh CE
`CP1250`	CP 1250 — Windows
`KEYBCS2`	Kameničtí

`CP852`	PC Latin 2
`KOI8-CS`	KOI8-CS — standard zemí RVHP

Předpokládejme, že skripty jsou uloženy v adresáři `/cgi-bin`. Dokument, jehož URL bylo `http://`*«server»*`/`*«cesta»*, je nyní v různých kódech dostupný přes URL:

`http://`*«server»*`/cgi-bin/to`*«kód»*`/`*«cesta»*

Při požadavku na toto URL spustí server skript `to`*«kód»*. Skript provede konverzi dokumentu určeného *«cestou»* do kódu *«kód»* a odešle jej zpět klientovi. Pokud tento dokument obsahuje nějaký relativní odkaz, změní se ve výsledném absolutním URL pouze část *«cesta»* a kódování stránky zůstane zachováno. Další důvod navíc pro používání relativních odkazů!

V dokumentech se musíme vyhnout určení základního URL dokumentu pomocí tagu `<BASE>`. V tomto případě by se informace o kódování ztratila.

Relativní odkazy můžeme používat i pro obrázky. Skript se chová inteligentně a binární soubor obrázku nepřekóduje. Místo toho pošle klientovi v HTTP hlavičce `Location` správné URL souboru s obrázkem. Prohlížeč si obratem požádá o obrázek znovu a tentokráte jej již skutečně dostane.

Pokud předem nevíme v jakém kódu uživatel bude chtít dokument doručit, můžeme použít URL ve tvaru:

`http://`*«server»*`/cgi-bin/whichcode/`*«cesta»*

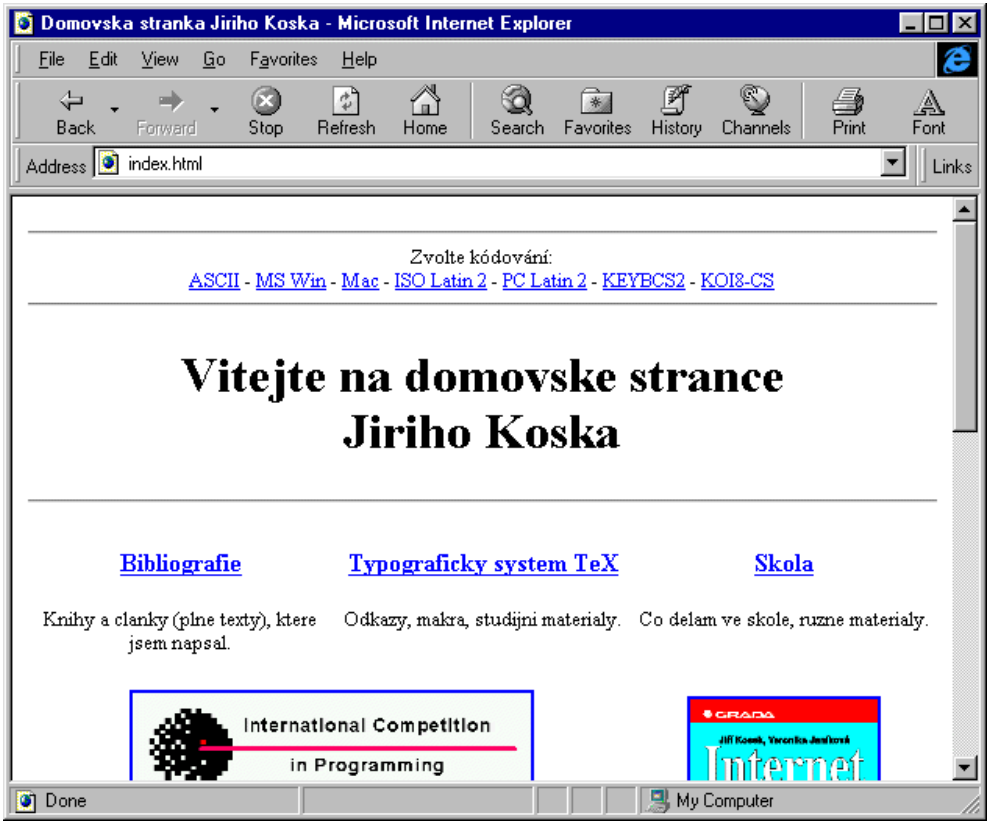
Skript `whichcode` nechá uživateli vybrat jedeno z dostupných kódování a poté mu přenese zadaný dokument již se správným kódováním.

Pokud chceme, aby se na některém místě stránky objevila lišta s nabídkou pro výběr kódování, stačí na samostatném řádku uvést speciální komentář:

```
<!--BAR-->
```


Při zpracování překódovacím skriptem se namísto tohoto komentáře vloží několik HTML odkazů, vedoucích k různým kódováním téhož dokumentu. Ještě lépe vypadá, když nabídku kódování vycentrujeme a oddělíme od zbytku stránky horizontálními čarami:

```
<DIV ALIGN=CENTER><HR>
<!--BAR-->
<HR></DIV>
```



Obr. 11-1: Lišta pro výběr kódování

Jak taková lišta vypadá v praxi na webovské stránce si můžeme prohlédnout na obrázku 11-1.

SaCzech předpokládá, že všechny dokumenty na serveru jsou uloženy právě v jednom z podporovaných kódů. Tento kód se vybírá při instalaci programu. Pokud chceme na server uložit dokument v jiném kódování a zajistit jeho správný převod do ostatních kódování, stačí na začátku dokumentu uvést speciální komentář

```
<!--MYCHARSET=«kód»-->
```

Pokud tedy vytváříme stránky pod *Windows*, měly by začínat zhruba takto:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">
<!--MYCHARSET=CP1250-->
```

Poslední vlastností *SaCzechu*, kterou zde zmíníme, je možnost zachovat kódování i při přechodu na jiný server. *SaCzech* při překódování dokumentu převádí

všechny výskyty řetězce `__CHARSET__` na jméno aktuálního kódu. Pokud tedy víme, že na serveru, kam se odkazujeme, používají rovněž *SaCzech*, můžeme v odkazu použít URL:

```
http://www.někde.cz/cgi-bin/to__CHARSET__/Home.html
```

Pokud si stránku s tímto odkazem budeme prohlížet například ve *Windows*, dorazí k nám již odkaz správně upravený:

```
http://www.někde.cz/cgi-bin/toCP1250/Home.html
```

Vidíme, že používání *SaCzechu* nás téměř nic nestojí (stačí pouze určit kódování stránky pomocí `<!--MYCHARSET=`*«kód»*`-->` a případně do stránky zařadit lištu pro výběr kódu `<!--BAR-->`). Získáme tím spokojeného uživatele, který si může pohodlně zvolit kódování.

Poslední otázka, kterou zbývá vyřešit, je tvar URL, které budeme poskytovat zájemcům o naši stránku. Předpokládejme, že URL naší domovské stránky je `http://server.cz/index.html`. Zájemcům o naši stránku pak můžeme dát jednu ze dvou adres `http://server.cz/cgi-bin/whichcode/index.html` a `http://server.cz/cgi-bin/toASCII/index.html`. V prvním případě si uživatel před vstupem na naši stránku musí vybrat požadované kódování. Druhý případ je pro uživatele příjemnější — stránku vidí rovnou (ASCII je čitelné ve všech operačních systémech). V tomto případě by však stránka měla hned na samém začátku obsahovat lištu pro výběr kódování, aby si uživatel mohl přepnout na verzi s háčky a čárkami.

☺ Situace se nám poněkud zkomplikuje, pokud jsme naše stránky dosud poskytovali pouze v jednom kódu (třeba v ASCII) a teď se rozhodneme pro používání *SaCzechu*, aby si každý uživatel mohl vybrat požadovaný kód. První řešení, které se naskýtá, je vytvoření jakési falešné domovské stránky, která nahradí původní domovskou stránku. Tato stránka bude sloužit pouze k výběru kódování a poté nahraje původní domovskou stránku uloženou v souboru s jiným jménem.

Toto řešení je sice funkční, ale pro mnoho uživatelů nepohodlné. Nabízí se lepší alternativa. Originální domovskou stránku `index.html` přejmenujeme na `indexa.html`. Ve všech našich dalších stránkách opravíme odkazy na domovskou stránku z `index.html` na nové `indexa.html`. Do dokumentu `indexa.html` doplníme lištu pro výběr kódování.

Nyní si naši novou domovskou stránku vyvoláme pomocí adresy `http://`*«server»*`/cgi-bin/toASCII/indexa.html`. Měla by se nám objevit naše domovská stránka bez háčků a čárek a s lištou pro výběr kódování. Tuto stránku uložíme do souboru `index.html` ve stejném adresáři jako je `indexa.html`.

V souboru `index.html` nyní musíme upravit všechny relativní odkazy, které vedou na naše další stránky. Před tyto odkazy musíme zařadit volání skriptu pro překódování stránky do ASCII. Pokud tedy domovská stránka obsahuje např. odkaz `clanky/index.html`, musíme z něj udělat odkaz `/cgi-bin/toASCII/clanky/index.html`. Všechny odkazy můžeme opravit jedinou operací, pokud využijeme funkci editoru Najdi a nahraď.

Pokud se nyní někdo dostane na naši stránku, objeví se mu v ASCII. To je výborné, protože ASCII umí správně zobrazit každý prohlížeč. Pokud ASCII uživateli vyhovuje, může se rovnou pohybovat po odkazech — kódování je již ve všech relativních URL správně zachyceno. Pokud by uživatel dostal chuť na něco lepšího, může si kódování s podporou diakritiky zvolit výběrem příslušného odkazu na liště pro výběr kódování.

Jediné, na co nesmíme zapomenout, je vždy po změně `indexa.html` aktualizovat i `index.html`.

Automatický výběr kódu

Řešení popsané v předchozích sekci uspokojí většinu požadavků. Jedinou nevýhodou jsou vysoké nároky na uživatele, který si musí vybrat správné kódování.[2] V poslední době se objevilo několik způsobů, jak zajistit správný výběr kódu bez účasti uživatele.

Prvním krokem bylo přesunutí samotného programu pro překódování z CGI-skriptů přímo do samotného WWW-serveru. Důvodem byla snaha snížit zatížení serveru způsobené opakovaným zaváděním CGI-skriptu do paměti a jeho spouštěním. Pokud rutiny pro převod kódu zahrneme do samotného WWW-serveru, zavedou se do paměti pouze jednou při startu serveru. Poskytování stránek se správným kódováním se tak nesmírně zrychlí. Pokud vím, existuje takováto úprava pro server *Apache* a pro *Internet Communication Server* firmy IBM.

To by však ještě nestačilo. Podstatným krokem k možnosti automatického výběru kódu byla nové verze protokolu HTTP/1.1. Ta pro HTTP požadavek zavedla novou hlavičku `Accept-charset`. Jejím obsahem je seznam kódování, která klient podporuje. Serveru tedy stačí požadovaný dokument převést do jednoho z podporovaných kódování a odeslat jej klientovi zpět. Aby klient věděl, které kódování je ve skutečnosti použito, pošle server zpět v hlavičce `Content-type` i informaci o použitém kódování. Pokud server odešle dokument v ISO Latin 2, bude hlavička vypadat takto:

```
Content-type: text/html; charset=iso-8859-2
```

[2] V minulé kapitole jsme naznačili, že před uživateli by měla být samotná technologická stránka Webu co nejvíce skryta. Proč by běžný uživatel počítače měl vědět, co je to nějaké kódování?

Odtud prohlížeč zjistí použité kódování a dokument správně zobrazí. Jediným problémem tohoto geniálního řešení je teprve pomalu vznikající podpora HTTP/1.1 v prohlížečích a nutnost zachovat funkčnost i se staršími prohlížeči.[3] Vypadá to, že toto řešení však bude nejlepší do té doby, než se v Čechách dohodneme na jediném společném kódování, které budou všichni používat.[4]

11.4 Tak takhle tedy ne!

Zatím jsme si ukazovali, jak *máme* řešit problém češtiny ve Webu. Nyní si povíme, jak jej řešit *nemáme*. Mnoho stránek je dnes na českém Internetu dostupných pouze v kódování pro *Windows*. Uživatelé, kteří používají jiné operační systémy, pak na svých obrazovkách vidí pouze změť podivných znaků. Autoři těchto stránek by se nad sebou měli zamyslet a uvědomit si, že ne každý už skočil strýčkovi Billovi na lep. Každá stránka by měla být dostupná i v ASCII bez diakritiky třeba pro ty, kteří se po Internetu brouzdají z nějakého starého unixové terminálu.

Další problém je obecnější, i když s *Windows* má také mnoho společného. Některé HTML-editory pracující pod *Windows*, např. *FrontPage*, do hlavičky HTML dokumentu uloží informaci o použitém kódování pomocí elementu `META`. Využívají k tomu atribut `HTTP-EQUIV`, který má podobný účel jako `NAME`, ale navíc je prohlížečem interpretován, jako by daný údaj byl součástí hlavičky HTTP odpovědi:

```
<META HTTP-EQUIV="Content-type"
      CONTENT="text/html; charset=windows-1250">
```

Pokud takovouto stránku používáme s některými systémy pro dynamickou změnu kódu a s některými prohlížeči, můžeme se dočkat nepěkných výsledků. Pokud víme, že server může stránku překódovat do jiného kódování a identifikaci tohoto kódování nepřipojí do HTTP hlavičky `Content-type`, radši řádku z HTML dokumentu vymažeme. Mohla by totiž zmást některé novější prohlížeče.

[3] To lze poměrně úspěšně obejít zvolením kódování podle identifikace prohlížeče uvedené v HTTP-hlavičce `User-Agent`. Kódování je správně určeno téměř ve sto procentech požadavků.

[4] Něco mi říká, že se stejně nedohodneme. Zachrání nás možná až nějaké kódování, které pokryje všechny znaky všech na světě používaných jazyků. Návrh HTML 4.0 již dnes počítá s používáním 32bitového kódování ISO 10 646 (to je kompatibilní s Unicode 2.0). Bohužel se jaksi nedostává programů, které by toto kódování podporovaly a byly běžně dostupné.

12. Kaskádové styly dokumentů

Původně byl jazyk HTML navržen tak, aby pomocí něj šlo snadno vyznačit jednotlivé logické části dokumentu. Nikdy přitom nebyl kladen důraz na přesnou definici výsledného vzhledu dokumentu. Web se však rozšířil i do komerční sféry a vznikly tak požadavky na lepší kontrolu nad grafickým vzhledem dokumentu. Tyto požadavky byly uspokojeny přidáním atributů jako je `ALIGN` a elementů jako `FONT` pro lepší ovládání vzhledu a formátování dokumentu. Kromě toho mnohé prohlížeče nabízely nepřeberné množství proprietárních rozšíření HTML, která umožnila lepší kontrolu nad výsledným vzhledem stránky.

Použití těchto technik mělo dva neblahé účinky. Za prvé se v HTML dokumentech místo jejich struktury začal vyznačovat spíše grafický vzhled. Druhá nevýhoda spočívala ve velké pracnosti a neflexibilitě tohoto řešení — způsob formátování se musel nastavit jednotlivě pro každý element dokumentu.

Obě tyto nevýhody odstraňují *káskádové styly* (CSS — Cascading Style Sheets [9]). Pomocí stylu lze jednoduše definovat druh písma, způsob zarovnání, barvu a další vlastnosti elementu. Tato definice se pak použije jednotně v celém dokumentu. V dokumentu se již zaměříme pouze na strukturu informace — grafický vzhled je definován stylem.

V jednom dokumentu může být použito několik stylů, které se navzájem doplňují. A naopak. Jeden styl může být použit společně s neomezeným počtem stránek, které tak získají jednotný vzhled.

V této kapitole se naučíme využívat styly pro ovládání zobrazení našich stránek.

12.1 Základy práce se styly

Práce se styly je velmi jednoduchá. Ukážeme si jednoduchý styl, který zajistí, že všechny nadpisy `H1` budou zobrazeny modře:

```
H1 { color: blue }
```

Tento jednoduchý styl se skládá z jednoho *pravidla*. Každé pravidlo má dvě části — *selektor* (`H1`) a *deklaraci* (`color: blue`). Deklarace se skládá ze dvou částí — z vlastnosti (`color`) a z její hodnoty (`blue`).

Selektor nám zajišťuje vazbu na odpovídající HTML element. Jako selektor mohou být samozřejmě použity všechny elementy HTML. Vlastností, které je u každého elementu možno použít, je několik desítek. Společně s hodnotami, které mohou nabývat, jsou shrnuty v tabulkách na následujících stránkách. Na ty důležitější z nich se kromě toho podíváme i samostatně.

Připojení stylu k HTML dokumentu

Aby mohl při zobrazování stránky použít prohlížeč styl, musí o něm mít informaci. V HTML jsou čtyři možnosti, jak styl k HTML dokumentu připojit. Následující ukázka obsahuje všechny čtyři najednou:

```
<HTML>
  <HEAD>
    <TITLE>Titulek stránky</TITLE>
    <LINK REL=STYLESHEET TYPE="text/css"
          HREF="http://style.com/super" TITLE="Super styl">
    <STYLE TYPE="text/css">
      @import url(http://style.com/zakladni);
      H1 { color: blue }
    </STYLE>
  </HEAD>
  <BODY>
    <H1>Nadpis je krásně modrý</H1>
    <P STYLE="color: green">Ekologický paragraf je zelený.
  </BODY>
</HTML>
```

První dva způsoby pracují se stylem uloženým v separátním souboru. Tento soubor se stylem lze k dokumentu připojit buď použitím elementu `LINK` s odpovídajícími atributy nebo pomocí příkazu `@import` v samotné definici stylu.

Styl může být definován i přímo v dokumentu mezi tagy `<STYLE>` a `</STYLE>`. Za pozornost stojí atribut `TYPE`. Pomocí něj určujeme druh použitého stylového jazyka. V budoucnu možná vzniknou i jiné stylové jazyky a proto bychom měli tento atribut používat k určení použitého druhu stylů. Pro styly CSS se používá MIME typ `text/css`.

Poslední možností je definice stylu pouze pro jeden konkrétní element. Toho lze dosáhnout použitím atributu `STYLE`, který můžeme použít u všech elementů. Tento způsob však poněkud odporuje samotné filosofii stylů — míchá totiž obsah dokumentu s jeho grafickou prezentací.

Aby byla zachována kompatibilita s prohlížeči, které nepodporují styly, je výhodné samotnou definici stylu uzavřít do komentáře. Starší prohlížeče ji pak ignorují:

```
<STYLE><!--
  H1 { color: blue }
--></STYLE>
```

Slučování definic

Abychom ušetřili místo, můžeme použít pro několik selektorů stejnou deklaraci. Selektory v tomto případě oddělujeme čárkou:

```
H1, H2, H3 { color: blue }
```

Sloučit můžeme i několik deklarací — ty se však oddělují středníkem:

```
H1 { font-weight: bold;
     font-size: 12pt;
     line-height: 14pt;
     font-family: sans-serif;
     font-variant: normal;
     font-style: normal }
```

Některé vlastnosti lza nastavit společně pomocí jediné. Předchozí příklad můžeme zkráceně zapsat jako

```
H1 { font: bold 12pt/14pt sans-serif }
```

Dědění vlastností

V naší první ukázce stylu jsme barvu elementu `H1` nastavili na modrou. Co se stane, když uvnitř elementu `H1` použijeme například element pro zvýraznění?

```
<H1>Tato kapitola <EM>je</EM> důležitější než ostatní</H1>
```

Pokud jsme stylem neurčili barvu elementu `EM`, bude slovo „je“ zobrazeno modře. Tuto barvu zdědí z nadřazeného elementu (`H1`), který je modrý. Většina vlastností se dědí podobně jako barva. U každé vlastnosti je v přehldu uvedeno, zda se dědí či ne.

Pokud chceme nastavit nějakou vlastnost jako základní pro celý dokument, můžeme použít deklaraci se selektorem `BODY`.

```
BODY { color: black;
       background-color: white }
```

Takto definované vlastnosti zdědí elementy, které neobsahují ve stylu vlastní deklaraci použitých vlastností. Dokument tedy bude zobrazen černě na bílém podkladu.

Třída jako selektor

Zatím jsme si ukázali, jak nastavit vzhled určitého elementu pro celý dokument společně. Někdy však potřebujeme stejný element zobrazit v různých výskytech rozdílně. Proto můžeme u každého elementu, který patří do těla dokumentu (`BODY`), určit jeho třídu pomocí atributu `CLASS`. Na jméno třídy definované tímto atributem samozřejmě můžeme odkazovat i v definici stylu. Dejme tomu, že naše stránka obsahuje poznámky, které jsou zapisovány následujícím způsobem:

```
<DIV CLASS=footnote>
Tady je text poznámky
</DIV>
```

V definici stylu lze selektor doplnit o název třídy. Název třídy se od jména elementu odděluje tečkou:

```
DIV.footnote { font-size: smaller;
               margin-left: 2em;
               margin-right: 2em }
```

Jako selektor můžeme použít i samotný název třídy. Pak deklarace platí pro všechny elementy, kterým je atributem `CLASS` přiřazena daná třída.

```
.footnote { font-size: smaller;
            margin-left: 2em;
            margin-right: 2em }
```

Použití tagu `<DIV>` způsobí zalomení textu a text elementu pokračuje až na další řádce. Pokud chceme třídu přiřadit nějakému textu, který je součástí odstavce, použijeme k jeho značení element `SPAN` společně s atributem `CLASS`. `SPAN` je nový element, který byl do HTML přidán právě kvůli stylům.

Pomocí tříd můžeme libovolný element nahradit nějakým jiným elementem, který je určen pro zcela jiné účely. Tuto vlastnost bychom však neměli příliš využívat, aby si HTML dokumenty zachovaly svoji strukturu. Třídy použijeme pouze tehdy, kdy potřebujeme ještě jemnější rozčlenění struktury dokumentu, než nám nabízí HTML.

Identifikátor elementu jako selektor

Pro použití se styly bylo HTML rozšířeno ještě o jeden atribut. U každého elementu můžeme použít atribut `ID`. Ten slouží k definici jedinečného jména elementu v rámci dokumentu. Na takto pojmenované elementy jednak můžeme vytvářet odkazy (v URL použijeme fragment) a jednak pro ně lze ve stylu uvést zvláštní deklaraci.

```
#L027 { letter-spacing: 1pt }
H1#L027 { letter-spacing: 2pt }
```

První deklarace vyhoví všem elementům, jejichž atribut `ID` je nastaven na hodnotu `L027`. Např.:

```
<P ID=L027>Odstavec prostrkaného textu
```

Druhá deklarace stylu se použije pouze v případech, kdy je `ID=L027` použito u elementu pro nadpis první úrovně.

Vidíme, že pomocí této konstrukce můžeme nastavit styl pro každý element zvlášť. Tomu bychom se však měli vyhnout — styl by měl být jednotný pro všechny elementy. Pokud potřebujeme u nějakého elementu rozlišit několik jeho různých významů, můžeme použít rozlišení pomocí tříd (atribut `CLASS`).

Kontextové selektory

To, že se vlastnosti ve stylech dědí, nám při vytváření stylů ušetří mnoho práce. Místo pracného nastavování všech vlastností stačí nastavit jejich základní hodnoty a poté pouze vyjmenovat výjimky. Pro nastavování výjimek je velice výhodné použít *kontextové selektory*. Kontextový selektor může vypadat třeba takto: `H1 EM`. Vyhovuje všem elementům `EM`, které jsou uvnitř elementu `H1`. Uvedenou vlastnost můžeme elegantně použít pro nastavení menšího písma pro položky hlouběji vnořených seznamů:

```
LI OL, LI UL { font-size: smaller }
```


Tato deklarace nám neříká nic jiného, než že v číslovaných a nečíslovaných seznamech obsažených uvnitř položky jiného seznamu bude použita menší velikost písma.

V kontextových selektorech můžeme kromě elementů používat i třídy (`CLASS`), názvy elementů (`ID`) a jejich kombinace:

```
.preface BLOCKQUOTE { font-style: italic }
#x81a CODE { color: yellow }
```

Komentáře

V definici stylu můžeme používat komentáře, na které jsme zvyklí z programovacího jazyka C:

```
EM { color: red } /* všechna zvýraznění budou červená */
```

Pseudotřídy a pseudoelementy

Výše popsaný způsob práce se selektory nám dovoluje opravdu mnoho. Existují však některé speciální případy, které je potřeba ošetřit samostatně. Pro tyto účely slouží speciální selektory. Jejich specialita spočívá v tom, že se v HTML stránce nikde neobjeví. V HTML zápisu stránky jim neodpovídá žádný element. Jsou pouze myšleně doplněny prohlížečem na potřebná místa.

Pseudotřídy existují tři. Slouží pro nastavení barvy odkazů, navštívených odkazů a aktivovaných odkazů:

```
A:link          { color: blue }   /* nenavštívený odkaz */
A:visited       { color: red }    /* navštívený odkaz */
A:active        { color: yellow } /* aktivní odkaz */
```

Pseudotřídy mohou být použity i v kontextových selektorech.

Pomocí pseudoelementů můžeme dosáhnout velice mocných grafických efektů. Bohužel je zatím většina prohlížečů nepodporuje. Pseudoelementy existují dva, `first-line` a `first-letter`, a slouží k nastavení vlastností první řádky resp. prvního písmene textu v daném elementu. Malá ukázka:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">
<HTML>
<HEAD>
<TITLE>Ukázka pseudoelementu</TITLE>
<STYLE TYPE="text/css"><!--
  P.iniciala:first-letter { font-size: 300%; float: left }
--></STYLE>
</HEAD>
<BODY>
<P CLASS=iniciala>"V té lednici už zase nic není," pomyslel si
pan J. a smutně si namazal krajíc chleba se sádlem, když v tom
se ozvalo zaklepání. Přerušil pohyb ruky, která směřovala
spolu s jeho večeří k ústům, a šel ke dveřím.
</BODY>
</HTML>
```

"V té lednici už zase nic není," pomyslel si pan J. a smutně si namazal krajíc chleba se sádlem, když v tom se ozvalo zaklepání. Přerušil pohyb ruky, která směřovala spolu s jeho večeří k ústům, a šel ke dveřím.

Všimněme si zajímavé vlastnosti. Kromě prvního písmene se změna nastavení dotkla i uvozovek. Jedná se o ustálený typografický zvyk.

Skládání stylů

Pokud je na stránce použito několik stylů, skládají se podle poměrně složitých pravidel. Vyšší váhu mají později uvedené styly. Pokud chceme některou deklaraci učinit důležitější, stačí za ní uvést `! important`:

```
H1 { color: blue ! important }
```

12.2 Vlastnosti

Vlastností, které lze pomocí stylů nastavit, je přes padesát. Jejich kompletní přehled nalezneme v tabulkách, kde jsou tematicky rozčleněny. Nejdůležitější z těchto vlastností probereme i samostatně.

Informace v tabulkách jsou uspořádány jednotně. První sloupec obsahuje název vlastnosti. Druhý sloupec přináší seznam přípustných hodnot. V tomto seznamu jsou použity některé operátory se speciálním významem (viz tabulka 12-1).

Konstrukce	Význam
A `|` B	Právě jedna z entit A a B
A `||` B	Alespoň jedna z entit A a B v libovolném pořadí
`[ ... ]`	Skupiny
`?`	Předcházející entita je nepovinná
`{`A`,`B`}`	Předcházející entita se opakuje nejméně A-krát a nejvíce B-krát

Tab. 12-1: Přehled syntaktických operátorů

Třetí sloupec uvádí implicitní hodnotu — tj. tu, která se pro danou vlastnost použije, pokud není ve stylu změněna. Další sloupec vymezuje elementy, na které se daná vlastnost vztahuje. Většinou se jedná o všechny, ale existují jisté výjimky. Pro tyto účely si elementy rozdělíme do tří skupin:

- *Blokové elementy* jsou ty elementy, před i za kterými je zalomena řádka (např. `H1` a `P`).
- *Inline elementy* jsou běžnou součástí textu na řádce. Nemají okolo sebe žádné zalomení řádek (např. `STRONG`).
- *Nahrazované elementy* jsou ty, které jsou nahrazeny nějakým obsahem a pro jejichž zařazení do okolního textu stránky jsou důležité pouze jejich rozměry (např. `IMG` a `OBJECT`).

Pátý sloupec nás informuje o tom, zda se vlastnost dědí i pro vnořené elementy. Následuje sloupec s informací o způsobu interpretace procentních hodnot. Ty se totiž u každého elementu mohou vztahovat k něčemu jinému. Poslední sloupec stručně popisuje danou vlastnost.

Písmo

Přehled vlastností, kterými můžeme ovlivnit vzhled písma, nalezneme v tabulce 12-3 na následující straně.

Název	Popis
`serif`	patkové písmo (např. Times Roman)
`sans-serif`	bezpatkové písmo (např. Arial)
`cursive`	ozdobná kurzíva (např. Zapf-Chancery)
`fantasy`	ozdobné písmo
`monospace`	neproporcionální písmo (např. Courier)

Tab. 12-2: Obecné rodiny písem

K nastavení použité rodiny písma slouží vlastnost `font-family`. Jako její hodnota se uvádí seznam písem.

```
BODY { font-family: Arial, Helvetica, sans-serif }
```

Prohlížeč se v tomto případě pokusí pro zobrazení dokumentu použít písmo Arial. Pokud jej nemá k dispozici, zkusí Helveticu. Pokud ani u té neuspěje, použije libovolné bezpatkové písmo — `sans-serif` je totiž obecným označením rodiny bezpatkového písma. Obecných rodin písma existuje několik (viz tab. 12-2). Obecnou rodinu písma bychom měli vždy uvést na konci seznamu

Vlastnost	Možné hodnoty	Implicitní hodnota	Aplikuje se na	Dědí se?	Interpretace procentních hodnot	Popis
font-family	*«seznam písem»*	záleží na prohlížeči	všechny elementy	ano	–	rodina písma
font-style	`normal | italic | oblique`	`normal`	všechny elementy	ano	–	styl písma: `normal` = normální, `italic` = kurzíva, `oblique` = skloněné
font-variant	`normal | small-caps`	`normal`	všechny elementy	ano	–	varianta písma: normální nebo kapitálky
font-weight	`normal | bold | bolder | lighter | 100 | 200 | 300 | 400 | 500 | 600 | 700 | 800 | 900`	`normal` = 400	všechny elementy	ano	–	duktus písma (tj. „jak bude písmo silné“)
font-size	`xx-small | x-small | small | medium | large | x-large | xx-large | larger | smaller |` *«délka»* `|` *«procento»*	`medium`	všechny elementy	ano	relativně k rodičovské velikosti písma	velikost písma
font	`[`*«font-style»* `||` *«font-variant»* `||` *«font-weight»*`]?` *«font-size»* `[/` *«line-height»*`]?` *«font-family»*	implicitní hodnoty jednotlivých vlastností	všechny elementy	ano	použitelné pro *«font-size»* a *«line-height»*	komplexní nastavení vlastností písma

Tab. 12-3: Vlastnosti písma

písem pro případ, že předchozí písma nebude mít prohlížeč k dispozici. Použije pak to, které svými vlastnostmi odpovídá obecnému písmu.

Po výběru rodiny písma můžeme výsledný řez písma ještě dále ovlivňovat. Jednak lze volit styl písma vlastností `font-style`. Kromě normálního (`normal`) písma můžeme použít kurzívu (`italic`) a skloněné písmo (`oblique`). Skloněné písmo přitom vzniká prostou *geometrickou transformací normálního písma*, oproti tomu *kurzíva je jedinečný řez písma*.

Rovněž můžeme volit variantu písma (`font-variant`). Tato vlastnost může nabývat pouze dvou hodnot `normal` a `small-caps`. Druhá varianta způsobí, že text bude ZOBRAZEN KAPITÁLKAMI.

Duktus (sílu) písma řídí vlastnost `font-weight`. Nejčastěji použijeme hodnoty `normal` pro normální písmo a `bold` pro písmo tučné. Zajímavou variantou je

hodnota `bolder`, která udělá písmo o něco silnější než momentálně je. Obdobně použitím `lighter` dostaneme o chlup světlejší písmo.

K ovládání velikosti písma slouží vlastnost `font-size`. Velikost můžeme nastavit absolutně — např. `12pt`. Zajímavou možnost nabízí použití procent. Použijeme-li `font-size: 200%`, velikost písma se zdvojnásobí oproti aktuální velikosti. Písmo lze nastavit i o kousek menší (`smaller`) a větší (`larger`). Všechny další možnosti jsou uvedeny v tabulce.

Vlastnost `font` nám umožňuje nastavit všechny vlastnosti písma najednou. Zápis

```
BLOCKQUOTE { font: bold italic 12pt/14pt "Times Roman", serif }
```

je ekvivalentní s

```
BLOCKQUOTE { font-weigth: bold;
             font-style: italic;
             font-size: 12pt;
             line-height: 14pt;
             font-family: "Times Roman", serif }
```

✍ *Předchozí vlastnosti nám umožní na jedné stránce použít téměř neomezený počet různých písem. V opojení těmito možnostmi pak mnoho autorů na svých stránkách hýří písmy. Praxe však ukazuje, že čím méně různých písem použijeme, tím lépe. Velký počet použitých písem jen rozptyluje čtenáře od čtení a od obsahu dokumentu.*

Pro běžné písmo odstavce bychom měli vždy volit patkové písmo. Patky totiž usnadňují orientaci v textu a vedou oko čtenáře po řádce. Při použití bezpatkového písma trvá čtenáři mnohem delší dobu, než přejde na další řádku textu. Tyto vlastnosti patkového písma se ještě zdůrazní po vytištění stránky na papír.

Barva a pozadí

Asi nejdůležitější vlastností této skupiny je vlastnost `color`. Ta slouží k nastavení barvy textu. Pro udání barvy máme několik možností. Jednak můžeme použít libovolnou z šestnácti předdefinovaných barev jako v HTML (tabulka 5-2 na straně 91). K nastavení barvy na modro můžeme použít `color: blue`.

Druhou možností je použít zápis `#ččmmzz`, kde pomocí dvouciferného šestnáctkového čísla zadáme postupně intenzity červené, modré a zelené složky barvy. Tmavě červené barvě odpovídá zápis `#330000`. Použít lze i zkrácenou notaci `#čmz`, kde je pro každou složku vyhrazena jen jedna cifra. Takto určená barva se automaticky převede na předchozí zápis tak, že se všechny cifry zdvojí. Tmavě červenou tedy můžeme zapsat i jako `#300`.

Styly nabízejí ještě další dvě možnosti určení barvy. Pokud chceme zadat intenzity jednotlivých složek v procentech, můžeme použít funkcionální zápis: `rgb(25%, 25%, 25%)`. V tomto případě dostaneme tmavě šedou barvu. Funkcionální zápis lze použít i v případech, kdy jednotlivé složky zadáváme jako čísla od 0 do 255. Předchozí šedivou tedy dostaneme také jako `rgb(63, 63, 63)`.

K nastavení barvy pozadí slouží vlastnost `background-color`. Pokud chceme mít na pozadí obrázek, použijeme `background-image`. Hodnotou této vlastnosti je URL. To se ve stylech zapisuje také speciálním způsobem:

```
BODY { background-image: url(http://server.cz/img/logo.gif) }
```

URL lze použít i relativní, ale nesmíme zapomenout, že jako základní URL se bere URL stylu. Pokud tedy máme k dokumentu připojen styl z vnějšku a je v jiném adresáři či na jiném serveru, musíme si uvědomit, kde je ve skutečnosti obrázek s pozadím. Pokud chceme do URL zapsat nějaké speciální znaky jako uvozovky, závorky či čárky, musíme jim předřadit zpětné lomítko (‘`\"`’, ‘`\(`’ apod.).

Pokud specifikujeme obě vlastnosti `background-color` i `background-image`, dosáhneme tím speciálního chování prohlížeče. Pokud se mu nepodaří získat obrázek s pozadím, bude pozadí ve specifikované barvě.

Přehled všech vlastností týkajících se barev a pozadí nalezneme v tabulce 12-4 na následující straně. Za zmínku ještě stojí vlastnost `background`, kterou můžeme nastavit více parametrů najednou:

```
BODY { background: blue url(http://server.cz/img/logo.gif) }
```

V tomto případě se použije jako pozadí obrázek, a pokud je nedostupný, bude pozadí modré.

Text

Souhrn vlastností, které ovlivňují formátování textu, je v tabulce 12-5 na straně 189. Mezi nejpoužívanější bude asi patřit vlastnost `text-align`, která určuje způsob zarovnání. Kromě klasických hodnot `left`, `right` a `center`, které známe z HTML od atributu `ALIGN`, se nabízí i hodnota `justify`. V tomto případě bude text zarovnán do bloku — rovné budou oba dva okraje textu.

Další důležitou vlastností je `line-height`. Ta určuje velikost řádkování. Můžeme ji zadat buď absolutně (`14pt`) nebo relativně jako násobek či procento velikosti písma. Aby se text dobře četl, mělo by být řádkování alespoň o 2 body větší než písmo. V tabulce 12-6 na straně 190 jsou uvedeny všechny jednotky, které můžeme používat pro určování délky. Jednotky můžeme rozdělit do dvou základních skupin — na relativní a absolutní. První se vztahují k velikosti písma nebo rozlišení obrazovky. Druhé jsou pevně vázány na běžně používané jednotky délky.

Vlastnost	Možné hodnoty	Implicitní hodnota	Aplikuje se na	Dědí se?	Interpretace procentních hodnot	Popis
color	*«barva»*	záleží na prohlížeči	všechny elementy	ano	–	barva
background-color	*«barva»* | `transparent`	`transparent`	všechny elementy	ne	–	barva pozadí
background-image	*«URL»* | `none`	`none`	všechny elementy	ne	–	obrázek na pozadí
background-repeat	`repeat` | `repeat-x` | `repeat-y` | `no-repeat`	`repeat`	všechny elementy	ne	–	směry, ve kterých se bude obrázek na pozadí opakovat
background-attachment	`scroll` | `fixed`	`scroll`	všechny elementy	ne	–	pozadí se pohybuje se stránkou (`scroll`) nebo je jako přibité (`fixed`)
background-position	[*«procento»* | *«délka»*] {1,2} | [`top` | `center` | `bottom`] || [`left` | `center` | `right`]	`0% 0%` (odpovídá `top left`)	blokové a nahrazované elementy	ne	vztahují se k velikosti vlastního elementu	poloha obrázku na pozadí (udává se X- a Y-pozice)
background	*«background-color»* || *«background-image»* || *«background-repeat»* || *«background-attachment»* || *«background-position»*	implicitní hodnoty jednotlivých vlastností	všechny elementy	ne	pouze u *«background-position»*	komplexní nastavení vlastností pozadí

Tab. 12-4: Vlastnosti pro nastavení barev a pozadí

✍ *Internet Explorer 3.x a 4.0 považuje jednotky* `em`, `ex` *za stejně velké jako* `px`. *Při jejich použití se v IE dočkáme nepěkných výsledků.*

12

Když můžeme ovlivnit výšku řádky, můžeme ovlivnit i to, jak se na této řádce budou jednotlivé elementy vertikálně zarovnávat. K tomu slouží vlastnost `vertical-align`. Z HTML již známe hodnoty `top`, `bottom` a `middle`. Zajímavou možností je `sub` a `super`. Ty zajistí, že element bude umístěn jako dolní resp. horní index. Standardní hodnotou je `baseline`, která zajistí vyrovnání účaří elementu s účařím okolí. Při vertikálním zarovnání můžeme použít i procenta. Interpretují se jako posunutí elementu nahoru o poměrnou část výšky

Vlastnost	Možné hodnoty	Implicitní hodnota	Aplikuje se na	Dědí se?	Interpretace procentních hodnot	Popis
word-spacing	`normal` | *«délka»*	`normal`	všechny elementy	ano	–	o kolik se má zvětšit mezislovní mezera
letter-spacing	`normal` | *«délka»*	`normal`	všechny elementy	ano	–	o kolik se má zvětšit mezera mezi jednotlivými písmeny
text-decoration	`none` | `[underline` || `overline` || `line-through` || `blink]`	`none`	všechny elementy	ne	–	„ozdoba“ textu: `underline` = podtržení, `overline` = čára nad, `line-through` = čára přes, `blink` = blikání
vertical-align	`baseline` | `sub` | `super` | `top` | `text-top` | `middle` | `bottom` | `text-bottom` | *«procento»*	`baseline`	inline elementy	ne	vztahují se k hodnotě `line-height`	vertikální zarovnání
text-transform	`capitalize` | `uppercase` | `lowercase` | `none`	`none`	všechny elementy	ano	–	převod textu na: `capitalize` = kapitálky, `uppercase` = velká písmena, `lowercase` = malá písmena
text-align	`left` | `right` | `center` | `justify`	záleží na prohlížeči	blokové elementy	ano	–	zarovnání textu: `left` = na levý praporek, `right` = na pravý praporek, `center` = centrování, `justify` = do bloku
text-indent	*«délka»* | *«procento»*	0	blokové elementy	ano	vztahují se k šířce rodičovského elementu	velikost odstavcové odrážky (odsazení první řádky odstavce)
line-height	`normal` | *«číslo»* | *«délka»* | *«procento»*	`normal`	všechny elementy	ano	relativně k velikosti písma elementu	výška řádky; *«číslo»* udává násobek velikosti písma (většinou by měl být alespoň 1.2)

Tab. 12-5: Vlastnosti textu

Jednotka	Popis
em	Výška aktuálního písma. Odpovídá šířce písmene 'M'.
ex	Výška písmene 'x'.
px	Pixely. 1px odpovídá jednomu bodu obrazovky.
in	Palce. 1in = 2,54 cm = 72 pt
cm	Centimetry.
mm	Milimetry. 10mm = 1 cm
pt	Body. 1pt = 1/72 in = 1/12 pc
pc	Pica. 1pc = 12 pt

Tab. 12-6: Přehled délkových jednotek

řádky. Pokud chceme, aby byl nějaký element posunut o čtvrtinu řádky dolů, použijeme deklaraci `vertical-align: -25%`.

Další užitečnou vlastností je `text-indent`. Umožňuje nastavit velikost odstavcové zarážky — tj. o kolik bude první řádka odstavce odsazena. Můžeme použít buď určení délky nebo procentní část šířky nadřazeného elementu (tím je běžně šířka okna prohlížeče).

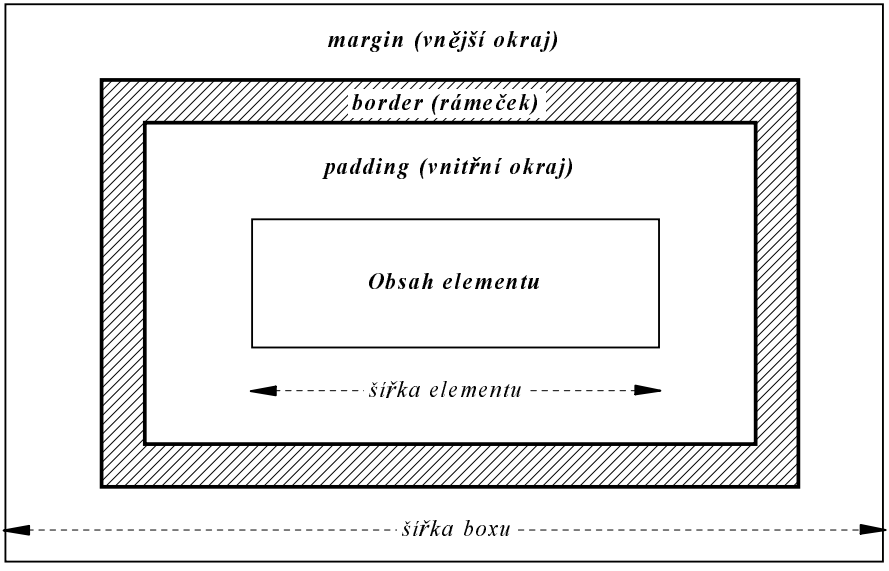
Formátování

Abychom správně pochopili všechny vlastnosti, které ovlivňují formátování stránky, musíme se seznámit s formátovacím modelem, který je v CSS používán. V tomto modelu je výsledkem zobrazení každého elementu jeden nebo několik obdélníkových boxů. Všechny boxy se skládají ze samotného obsahu elementu. Kromě toho mohou obsahovat vnitřní okraj, rámeček a vnější okraj (viz obr. 12-1 na následující straně).

K nastavení velikosti vnějšího okraje slouží vlastnosti začínající na `margin`. Obdobně k nastavení velikosti vnitřního okraje slouží vlastnosti začínající na `padding`. U rámečku lze kromě jeho velikosti nastavit i barvu (`border-color`) a styl (`border-style`). Přehled všech těchto vlastností přinášejí tabulky 12-7 na straně 192, 12-8 na straně 193 a 12-9 na straně 194.

Mnoho z výše uvedených vlastností lze nastavit jedinou deklarací pro všechny čtyři strany (horní, pravá, dolní a levá). Počet hodnot přiřazených takovéto souhrnné vlastnosti může být od jedné do čtyř. Použijeme-li pouze jednu hodnotu, bude platit pro všechny čtyři strany. Při použití dvou hodnot bude první platit pro horní a dolní stranu a druhá pro levou a pravou stranu. Při použití tří hodnot budou postupně odpovídat horní, pravé a dolní straně. Pro levou stranu bude použita stejná hodnota jako pro pravou. Zápis

```
BODY { margin: 1em 2em }        BODY { margin: 1em 2em 3em }
```



Obr. 12-1: Formátovací model

je ekvivalentní s

```
BODY { margin-top: 1em;          BODY { margin-top: 1em;
       margin-bottom: 1em;               margin-bottom: 3em;
       margin-right: 2em;                margin-right: 2em;
       margin-left: 2em }                margin-left: 2em }
```

V této skupině ještě nalezneme vlastnosti `width` a `height`, které slouží k nastavení šířky a výšky elementu. Uplatní se zejména u nahrazovaných elementů (obrázky).

Pomocí vlastnosti `float` můžeme z elementu udělat plovoucí objekt. To znamená, že pak bude obtékán okolním textem zleva či zprava. Tato vlastnost odpovídá nastavení atributu `ALIGN` u obrázků na hodnotu `LEFT` nebo `RIGHT`. Svou obdobu má v HTML i vlastnost `clear`, která určuje, zda se element zobrazí až po skončení všech plovoucích objektů. Můžeme si vybrat, zda se bude čekat na skončení všech objektů (hodnota `both`) nebo jen na skončení objektů vpravo (`right`) či vlevo (`left`). Následující krátký styl umístí všechny obrázky jako plovoucí objekty vlevo. Všechny nadpisy první a druhé úrovně budou vždy začínat až pod obrázky.

```
IMG     { float: left }
H1, H2  { clear: both }
```

Vlastnost	Možné hodnoty	Implicitní hodnota	Aplikuje se na	Dědí se?	Interpretace procentních hodnot	Popis
margin-top	*«délka»* | *«procento»* | `auto`	0	všechny elementy	ne	vztahují se k šířce rodičovského elementu	velikost horního okraje
margin-right	*«délka»* | *«procento»* | `auto`	0	všechny elementy	ne	vztahují se k šířce rodičovského elementu	velikost pravého okraje
margin-bottom	*«délka»* | *«procento»* | `auto`	0	všechny elementy	ne	vztahují se k šířce rodičovského elementu	velikost spodního okraje
margin-left	*«délka»* | *«procento»* | `auto`	0	všechny elementy	ne	vztahují se k šířce rodičovského elementu	velikost levého okraje
margin	`[`*«délka»* | *«procento»* | `auto] {1,4}`	implicitní hodnoty jednotlivých vlastností	všechny elementy	ne	vztahují se k šířce rodičovského elementu	komplexní nastavení velikosti okrajů
padding-top	*«délka»* | *«procento»*	0	všechny elementy	ne	vztahují se k šířce rodičovského elementu	velikost vnitřního horního okraje
padding-right	*«délka»* | *«procento»*	0	všechny elementy	ne	vztahují se k šířce rodičovského elementu	velikost vnitřního pravého okraje
padding-bottom	*«délka»* | *«procento»*	0	všechny elementy	ne	vztahují se k šířce rodičovského elementu	velikost vnitřního spodního okraje
padding-left	*«délka»* | *«procento»*	0	všechny elementy	ne	vztahují se k šířce rodičovského elementu	velikost vnitřního levého okraje

Tab. 12-7: Vlastnosti boxů — I.

Vlastnost	Možné hodnoty	Implicitní hodnota	Aplikuje se na	Dědí se?	Interpretace procentních hodnot	Popis
padding	[*«délka»* | *«procento»*] {1,4}	0	všechny elementy	ne	vztahují se k šířce rodičovského elementu	komplexní nastavení velikostí vnitřního okraje
border-top-width	thin | medium | thick | *«délka»*	medium	všechny elementy	ne	–	šířka horní části rámečku
border-right-width	thin | medium | thick | *«délka»*	medium	všechny elementy	ne	–	šířka pravé části rámečku
border-bottom-width	thin | medium | thick | *«délka»*	medium	všechny elementy	ne	–	šířka spodní části rámečku
border-left-width	thin | medium | thick | *«délka»*	medium	všechny elementy	ne	–	šířka levé části rámečku
border-width	[thin | medium | thick | *«délka»*] {1,4}	implicitní hodnoty jednotlivých vlastností	všechny elementy	ne	–	komplexní nastavení šířky rámečku
border-color	*«barva»* {1,4}	hodnota vlastnosti color	všechny elementy	ne	–	barva rámečku
border-style	[none | dotted | dashed | solid | double | groove | ridge | inset | outset] {1,4}	none	všechny elementy	ne	–	nastavení tvaru rámečku
border-top	*«border-top-width»* || *«border-style»* || *«barva»*	implicitní hodnoty jednotlivých vlastností	všechny elementy	ne	–	nastavení vlastností horní části rámečku
border-right	*«border-top-width»* || *«border-style»* || *«barva»*	implicitní hodnoty jednotlivých vlastností	všechny elementy	ne	–	nastavení vlastností pravé části rámečku

Tab. 12-8: Vlastnosti boxů — II.

Vlastnost	Možné hodnoty	Implicitní hodnota	Aplikuje se na	Dědí se?	Interpretace procentních hodnot	Popis
border-bottom	*«border-top-width»* || *«border-style»* || *«barva»*	implicitní hodnoty jednotlivých vlastností	všechny elementy	ne	–	nastavení vlastností spodní části rámečku
border-left	*«border-top-width»* || *«border-style»* || *«barva»*	implicitní hodnoty jednotlivých vlastností	všechny elementy	ne	–	nastavení vlastností levé části rámečku
border	*«border-width»* || *«border-style»* || *«barva»*	implicitní hodnoty jednotlivých vlastností	všechny elementy	ne	–	komplexní nastavení vlastností rámečku
width	*«délka»* | *«procento»* | `auto`	`auto`	blokové a nahrazované elementy	ne	vztahují se k šířce rodičovského elementu	šířka
height	*«délka»* | `auto`	`auto`	blokové a nahrazované elementy	ne	–	výška
float	`left` | `right` | `none`	`none`	všechny elementy	ne	–	umístění plovoucího objektu: `left` = vlevo, `right` = vpravo, `none` = normální objekt
clear	`none` | `left` | `right` | `both`	`none`	všechny elementy	ne	–	čekání na skončení plovoucích elementů: `left` = vlevo, `right` = vpravo, `both` = na obou stranách

Tab. 12-9: Vlastnosti boxů — III.

Vlastnost	Možné hodnoty	Implicitní hodnota	Aplikuje se na	Dědí se?	Interpretace procentních hodnot	Popis
display	`block | inline | list-item | none`	`block`	všechny elementy	ne	–	druh elementu
white-space	`normal | pre | nowrap`	`normal`	blokové elementy	ano	–	způsob práce s mezerami: `normal` = normální, `pre` = mezery a kance řádek se zachovají, `nowrap` = text se nebude zalamovat do řádek
list-style-type	`disc | circle | square | decimal | lower-roman | upper-roman | lower-alpha | upper-alpha | none`	`disc`	elementy, které mají `display: list-item`	ano	–	druh odrážek v seznamech
list-style-image	*«URL»* | `none`	`none`	elementy, které mají `display: list-item`	ano	–	obrázek použitý jako odrážka v seznamu
list-style-position	`inside | outside`	`outside`	elementy, které mají `display: list-item`	ano	–	umístění odrážky: `outside` = před textem, `inside` = v textu položky seznamu
list-style	*«list-style-type»* || *«list-style-position»* || *«list-style-image»*	implicitní hodnoty jednotlivých vlastností	elementy, které mají `display: list-item`	ano	–	komplexní nastavení vzhledu položek seznamu

Tab. 12-10: Klasifikační vlastnosti

Klasifikace elementů

Vlastnosti uvedené v tabulce 12-10 můžeme rozdělit do dvou skupin. První slouží k určení druhu elementu a způsobu jeho zobrazení. Do druhé skupiny patří vlastnosti, které určují druh zobrazení seznamů.

Pomocí vlastnosti `display` můžeme změnit druh elementu. Z blokového elementu tak můžeme udělat třeba inline element. Tato možnost v praxi asi uplatnění nenajde. Užitečná je však hodnota `none`, která způsobí nezobrazování daného elementu. Chceme-li tedy, aby se v dokumentu nezobrazovaly obrázky, můžeme použít:

```
IMG { display: none }
```

Pomocí vlastnosti `list-style-type` si vybíráme způsob odrážek a jejich číslování. Na výběr máme stejné možnosti, které nám nabízí atribut `TYPE` v HTML u seznamů. O poznání zajímavější je vlastnost `list-style-image`. Slouží ke specifikování URL obrázku, který se použije místo odrážky. Snadno tak vyřešíme náš problém ze sekce „Seznamy s grafickými odrážkami“ na straně 121. Stačí používat běžně seznamy a styl

```
UL LI { list-style-image: url(arrow.gif) }
```

Pokud není obrázek z nějakého důvodu k dispozici, použije se odrážka nastavená pomocí `list-style-type`. Vlastnost `list-style-position` určuje pozici odrážky vůči položce seznamu:

- Tato položka seznamu má `list-style-position: inside`. Když ji uděláme dostatečně dlouhou, můžeme ji porovnat s druhou položkou.
- Tato položka seznamu má `list-style-position: outside`. Když ji uděláme dostatečně dlouhou, můžeme ji porovnat s první položkou.

Vlastnost odrážek můžeme nastavit najednou pomocí vlastnosti `list-style`:

```
list-style: disc outside url(arrow.gif)
```

12.3 Styly v praxi

Z předchozí části kapitoly vidíme, že styly jsou opravdu mocný a výkonný nástroj pro zlepšení grafického vzhledu našich stránek. Jedinou nevýhodou je, že zatím ne všechny prohlížeče podporují všechny vlastnosti, které jsme zde popsali. *Netscape Communicator* podporuje styly až od verze 4.0. *Internet Explorer* je podporuje již od verze 3.0. Bohužel ani jeden z prohlížečů nepodporuje všechny vlastnosti stylů. Naštěstí každá nová verze přináší výrazná zlepšení v podpoře stylů. Styly tedy může hojně používat, ale stránky musíme psát tak, aby svůj význam neztratily ani v prohlížečích, které styly nepodporují.

Nyní si na jednoduché stránce prakticky ukážeme použití stylů. Nejprve si do souboru `mujstyl.css` uložíme definici stylu:

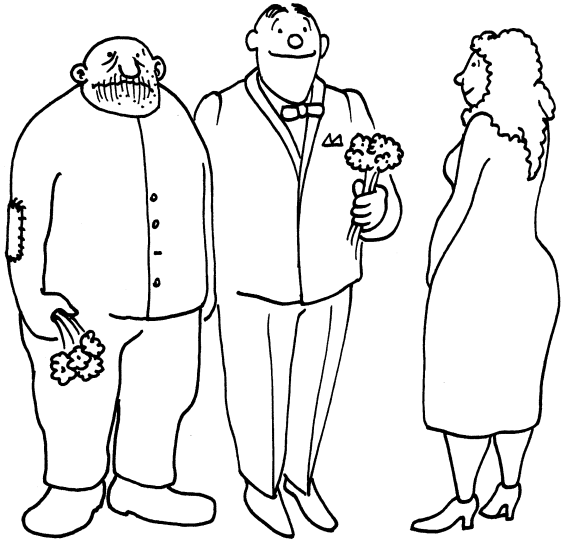
```
BODY            { text-align: justify;
                  background-color: white;
                  color: black }

A:link          { color: blue }
A:visited       { color: gray }
A:active        { text-decoration: blink }


P               { text-indent: 20pt;
                  margin-top: 0pt }

.nadpis         { text-align: center }

H1              { text-transform: uppercase;
                  background-color: blue;
                  color: yellow;
                  padding: 4px }

H2              { font-family: Arial, sans-serif }

.autor          { font-variant: small-caps }
```

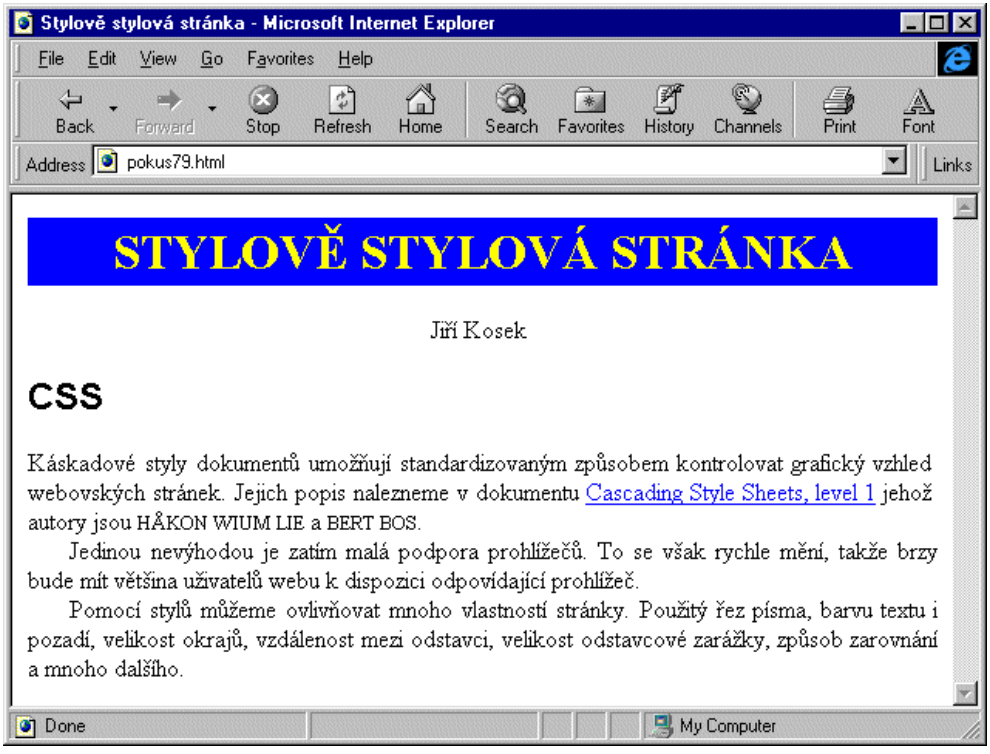
Styl ke stránce připojíme s využitím elementu `LINK`.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">
<HTML>
<HEAD>
<TITLE>Stylově stylová stránka</TITLE>
<LINK REL=StyleSheet HREF="mujstyl.css" TYPE="text/css">
</HEAD>
<BODY>

<DIV CLASS=nadpis>
<H1>Stylově stylová stránka</H1>
Jiří Kosek
</DIV>

<H2>CSS</H2>

Káskádové styly dokumentů umožňují standardizovaným způsobem
kontrolovat grafický vzhled webovských stránek. Jejich popis
nalezneme v dokumentu
```

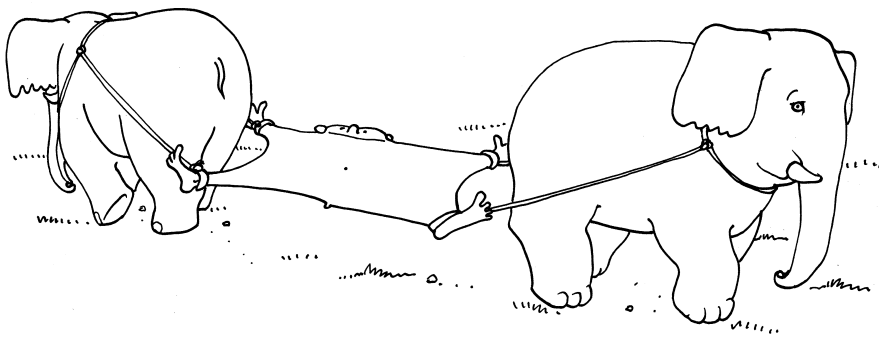
```
<A HREF="http://www.w3.org/pub/WWW/TR/REC-CSS1.html">Cascading
Style Sheets, level 1</A> jehož autory jsou
<SPAN CLASS=autor>H&aring;kon Wium Lie</SPAN>
a <SPAN CLASS=autor>Bert Bos</SPAN>.

<P>Jedinou nevýhodou je zatím malá podpora prohlížečů. To se
však rychle mění, takže brzy bude mít většina uživatelů webu
k dispozici odpovídající prohlížeč.

<P>Pomocí stylů můžeme ovlivňovat mnoho vlastností stránky.
Použitý řez písma, barvu textu i pozadí, velikost okrajů,
vzdálenost mezi odstavci, velikost odstavcové zarážky, způsob
zarovnání a mnoho dalšího.

</BODY>
</HTML>
```

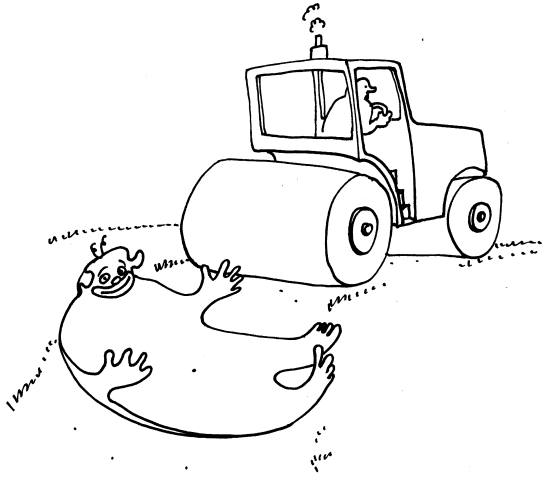
Výsledné zobrazení naší stránky v prohlížeči *Internet Explorer 4.0* si můžeme prohlédnout na obrázku 12-2 na následující straně.

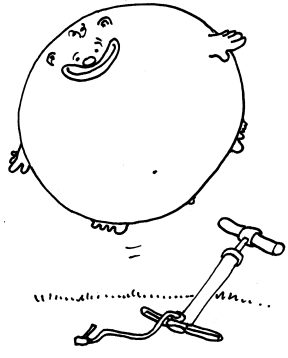




Obr. 12-2: Stránka využívající styly







13. Rozšíření HTML

Tato kapitola nebude tvořit tak kompaktní celek jako ty předchozí. V prvé řadě se podíváme na vlastnosti HTML 3.2, o kterých jsme se dosud nezmínili. Budou jimi Java-aplety a rozšířené možnosti tvorby odkazů.

Poté se podíváme na nové elementy a atributy, které jsou již podporovány mnohými prohlížeči a jsou i součástí návrhu HTML 4.0. Jedná se o rámy, rozšířený model tabulek podle RFC 1942 a o používání skriptů na stránkách.

Následující velká část kapitoly bude věnována popisu novinek, s kterými přichází návrh standardu HTML 4.0 — podpora vícejazyčných dokumentů, vkládání obecných objektů do stránek, vylepšení v oblasti formulářů a některé další změny.

Na závěr kapitoly si ukážeme, jaké máme dnes možnosti, pokud chceme do stránky vložit matematický výraz či chemický vzorec.

13.1 Java-aplety

Java-aplet je program v jazyce Java, který je součástí stránky. Nejčastěji se aplety využívají pro vkládání animací, interaktivní grafiky apod. Můžeme je však použít v podstatě k čemukoliv — záleží na schopnostech programátora, který aplet psal.

Programy v Javě mají příponu `.java`. Pro jejich použití na stránkách je musíme zkompilovat do speciálního tvaru — tzv. *bajtového kódu (byte-code)*. Zkompilované aplety mají příponu `.class`. Bajtový kód není spouštěn přímo operačním systémem počítače, ale speciálním interpretem — virtuálním strojem (virtual machine). Ten odstiňuje rozdíly jednotlivých platforem a umožňuje tak programy napsané v Javě spouštět na libovolném počítači.

Do stránek se aplety vkládají pomocí elementu `APPLET`. Nejdůležitějším atributem je `CODE`. Ten určuje jméno apletu, který chceme spustit. Stejně důležité jsou i atributy `WIDTH` a `HEIGHT`, které určují rozměry apletu v pixelech. Toto místo se pak na stránce vyhradí pro potřeby apletu — aplet zde může psát, kreslit, zjišťovat pozici myší apod.

Jelikož pro samotnou stránku je aplet pouhá obdélníková oblast, můžeme podobně jako u obrázků řídit zarovnání s okolním textem pomocí atributu `ALIGN`. Vzdálenost apletu od okolního textu ovlivňují atributy `HSPACE` a `VSPACE`.

Jednoduchý aplet můžeme do stránky vložit následovně:

```
<APPLET CODE="rotujici_kedluben.class" WIDTH=400 HEIGHT=350>
Na tomto místě je Java aplet, na kterém se kolem své
osy otáčí specifický druh kořenové zeleniny.
</APPLET>
```

Vidíme, že obsahem elementu je text pro prohlížeče, které aplety nepodporují. K podobným účelům lze použít i atribut `ALT`. Ten však prohlížeče, které nepodporují aplety, budou ignorovat, takže pro vložení textu pro „neapletové“ prohlížeče se výhradně používá text mezi `<APPLET>` a `</APPLET>`.

Každý aplet na stránce můžeme pojmenovat pomocí atributu `NAME`. Pokud je na jedné stránce více apletů, mohou toto jméno používat pro vzájemnou komunikaci.

Posledním atributem použitelným pro aplety je `CODEBASE`. Slouží k určení základního URL apletu. Pokud jej nepoužijeme, bude se za základní URL považovat URL stránky, která aplet obsahuje.

Appletům je možné předávat ze stránky parametry. K tomu slouží element `PARAM`, který se vkládá ihned za tag `<APPLET>`. U každého parametru musíme zadat jeho jméno a hodnotu:

`<PARAM NAME=`*«jméno parametru»* `VALUE=`*«hodnota»*`>`

✍ *Před odesláním hodnoty parametru apletu jsou nahrazeny všechny znakové entity příslušným znakem. Pokud tedy chceme jako hodnotu parametru předat znak ‘`&`’, musíme jej zapsat jako ‘`&`’.*

Obr. 13-1: Ukázka Java-apletu na stránce

Na obrázku 13-1 na straně 202 je zobrazena stránka s apletem, který pohybuje zadaným textem. K vložení apletu byl použit následující kód:

```
<DIV ALIGN=CENTER>
<APPLET CODE="NervousText.class"
        WIDTH=350 HEIGHT=50
        ALT="Stránka veselých vtipů">
<PARAM NAME=text VALUE="Stránka veselých vtipů">
<H1>Stránka veselých vtipů</H1>
</APPLET>
</DIV>
```

Vidíme, že v prohlížečích, které Javu nepodporují, se místo poskakujícího textu zobrazí normální nadpis. To je důležité, protože ne všechny prohlížeče Javu podporují. (Ty textové ani nemohou, protože Java-aplet může zobrazovat pouze na grafické obrazovce.)

13.2 Odkazy s určením typu

V předešlé kapitole jsme se zabývali kaskádovými styly. Jednou z možností, jak styl připojit ke stránce, je použití elementu `LINK`. Kromě připojení stylu může být tento element využit i k dalším účelům:

- určení vzájemných vztahů mezi skupinou stránek;
- ovlivnění způsobu tisku souvisejících dokumentů;
- poskytnutí dokumentu v jiném formátu.

Informace potřebné pro dosažení výše popsaných cílů se specifikují pomocí atributů elementu. Nám již známým atributem je `HREF`, který určuje URL zdroje, na který odkaz ukazuje. Rovněž známe atribut `TITLE`, který slouží pro zadání popisu připojeného zdroje. Popis je zde pouze pro informaci uživatele, je na prohlížeči, jak jej uživateli zpřístupní.

Důležitou roli hraje atribut `REL`. Pomocí něj určujeme typ informace, která je pomocí `LINK` k dokumentu připojena. Když jsme připojovali styl, použili jsme `REL=StyleSheet`. Tím jsme prohlížeč informovali o tom, že atribut `HREF` obsahuje definici stylu, který chceme použít při formátování stránky. Přehled nejčastěji používaných typů vztahů přináší tabulka 13-1 na následující straně.

Místo atributu `REL` můžeme použít atribut `REV`. Ten určuje typ opačného vztahu. Jestliže mezi dokumenty A a B existuje vztah `REL=`*«vztah»*, pak mezi B a A je vztah `REV=`*«vztah»*. Typ vztahu `Made` se společně s tímto atributem někdy využívá pro určení autora dokumentu:

```
<LINK REV=Made HREF="mailto:Jiri.Kosek@stv.cz">
```

Typ vztahu	Popis
`Top, Start`	Odkaz na hlavní úroveň hierarchie dokumentů. V praxi to znamená odkaz na úvodní stránku skupiny stránek.
`Contents`	Odkaz na dokument s obsahem.
`Index`	Odkaz na rejstřík k aktuálnímu dokumentu.
`Glossary`	Odkaz na slovníček použitých termínů.
`Copyright`	Odkaz na stránku, která obsahuje vymezení autorských práv pro aktuální stránku.
`Next`	Odkaz na další dokument ve skupině.
`Previous`	Odkaz na předcházející dokument ve skupině.
`Help`	Odkaz na stránku, která uživatele informuje o významu a kontextu aktuální stránky.
`Search`	Odkaz na stránku, která umožňuje prohledávat skupinu stránek.
`StyleSheet`	Odkaz na definici stylu, který se má použít při formátování dokumentu.
`Bookmark`	Odkaz na záložku definovanou v dokumentu.
`Alternate`	Odkaz na alternativní verzi dokumentu (např. připravenou pro tisk). MIME typ této verze dokumentu se zapisuje do atributu `TYPE`.

Tab. 13-1: Typy vztahů

Asi nejtypičtější použití ukazuje následující příklad. V něm definujeme odkazy na další a předcházející dokument a na obsah:

```
<LINK REL=Contents HREF="obsah.html">
<LINK REL=Next HREF="kap4.html">
<LINK REL=Previous HREF="kap2.html">
```

Ve zcela stejném významu můžeme atributy `REL`, `REV` a `TITLE` použít u elementu `A`.

13.3 Rámy

13

Rámy do HTML přidaly možnost rozdělit okno prohlížeče na několik částí. V každé části přitom může být zobrazována jiná stránka. (Těmto částem tedy říkáme rámy.) Rámy lze nastavit i tak, že zvolení odkazu v jednom rámu způsobí načtení a zobrazení stránky v jiném rámu. Není pak problém vytvořit stránku, kde jeden rám obsahuje navigační menu, pomocí kterého můžeme volit zobrazovanou stránku v druhém rámu.

Rozložení rámů na stránce

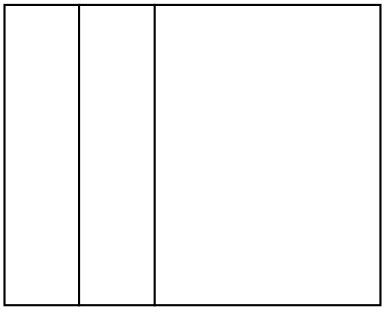
Struktura dokumentu, který definuje rozložení rámů na stránce, se od struktury běžné stránky trochu liší. Stránka s rámy musí obsahovat záhlaví (`HEAD`) a definici rozložení rámů (`FRAMESET`). Tělo dokumentu (`BODY`) je nepovinné a zobrazuje se pouze v těch prohlížečích, které rámy nepodporují. Tímto způsobem lze poskytnou alternativu pro přístup k informacím uloženým v rámech pro starší prohlížeče.

Definice rozložení rámů na stránce se provádí pomocí párového elementu `FRAMESET`. U něj lze použít dva atributy `ROWS` a `COLS`, které určují, na kolik řádek či sloupců se obrazovka rozdělí. Pokud použijeme oba dva atributy najednou, vytvoří se mřížka.

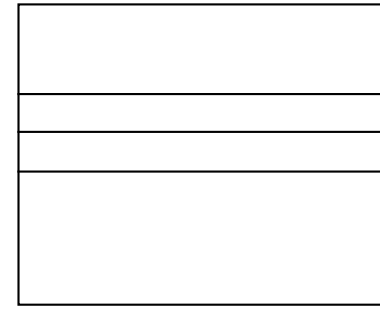
Jako hodnota atributů se uvádí čárkami oddělený seznam velikostí řádek (sloupců). Velikost můžeme určit buď absolutně nebo relativně. Absolutní velikost se zadává jako počet pixelů nebo jako procento z celkové výšky (šířky) obrazovky. Relativní velikost ze zadává ve tvaru '`i*`', kde '`i`' je celé číslo. Samotná hvězdička '`*`' odpovídá zápisu '`1*`'. Při určování výšky (šířky) jednotlivých rámů se nejprve dostupné místo rozdělí mezi rámy s absolutně udanou velikostí. Zbylé místo se poměrně rozdělí mezi rámy s relativně určenou velikostí. Popsaná pravidla si ukážeme na čtyřech ukázkách. Budeme předpokládat, že mezi jednotlivé rámy se má rozdělit část obrazovky o rozměrech 1000×800 bodů.

```
<FRAMESET ROWS="50%, 50%">
    «definice obsahu rámů»
</FRAMESET>
```

```
<FRAMESET COLS="*, 200, 3*">
    «definice obsahu rámů»
</FRAMESET>
```



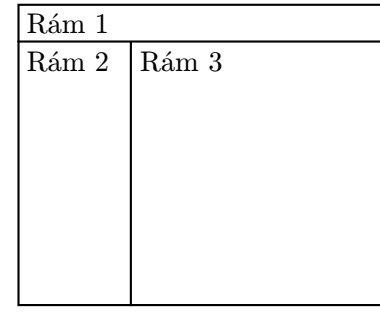
```
<FRAMESET ROWS="30%, 100, 1*, 3*">
    «definice obsahu rámů»
</FRAMESET>
```



```
<FRAMESET ROWS="100, *, *"
          COLS="50%, 50%">
    «definice obsahu rámů»
</FRAMESET>
```

Pokud chceme nějaký takto vzniklý rám dále dělit, můžeme použít vnořený element FRAMESET. Následující ukázka vytvoří tři rámy. V horní části obrazovky bude úzký rám přes celou šířku. Zbylý prostor se rozdělí mezi dva svislé rámy:

```
<FRAMESET ROWS="100, *">
  «definice obsahu 1. rámu»
  <FRAMESET COLS="30%, 70%">
    «definice obsahu 2. rámu»
    «definice obsahu 3. rámu»
  </FRAMESET>
</FRAMESET>
```



Definice obsahu rámu

Když máme definováno rozložení rámů pomocí elementu FRAMESET, můžeme pomocí elementu FRAME určit, které dokumenty se v jednotlivých rámech zobrazí. URL dokumentu se určuje atributem SRC. Poslední ukázku rozšíříme již do praktické podoby:

```
<HTML>
<HEAD>
  <TITLE>Stránka s rámy</TITLE>
</HEAD>
<FRAMESET ROWS="100, *">
  <FRAME SRC="logo.html">
  <FRAMESET COLS="30%, 70%">
    <FRAME SRC="obsah.html">
    <FRAME SRC="uvod.html">
  </FRAMESET>
</FRAMESET>
</HTML>
```

Spolu s elementem FRAME lze použít mnoho atributů, které ovlivňují chování rámu. Pokud nechceme, aby šla velikost rámu změnit, můžeme použít atribut NORESIZE. Uplatnění nalezne především v případech, kdy je rám tvořen obrázkem, u něhož známe přesné rozměry. V těchto případech můžeme rozložení rámů definovat tak, aby byl obrázek celý vidět, a můžeme tedy uživateli zakázat možnost měnit velikost rámu. V ostatních případech bychom to dělat neměli, protože uživatel si velikost rámů přizpůsobí svým potřebám a velikosti obrazovky.

Další atribut, ovlivňující chování rámů, je SCROLLING. Pokud jeho hodnotu nastavíme na NO, nepůjde se po obsahu rámu pohybovat. Bude-li text v rámu příliš dlouhý, nebude čtenáři celý přístupný. Hodnota YES naopak vždy rám doplní o posuvník, takže pohyb po delší stránce nebude žádným problémem. Standardně má tento atribut hodnotu AUTO — posuvník se zobrazí pouze pokud se stránka celá nevejde do rámu.

Rámy jsou od sebe normálně odděleny úzkým rámečkem. Pokud tento rámeček mezi rámy nechceme, použijeme atribut FRAMEBORDER=0.

Vzdálenost obsahu rámu od jeho okrajů na levé a pravé straně můžeme určit pomocí atributu MARGINWIDTH. Jako hodnotu je možno použít počet pixelů nebo procento z šířky rámu. Obdobně lze nastavit i vzdálenost od horního a spodního okraje rámu pomocí atributu MARGINHEIGHT.

Velice důležitým atributem je NAME. Pomocí něj definujeme jméno rámu. Jméno rámu musí začínat písmenem anglické abecedy (a–z, A–Z); dále pak může obsahovat libovolné znaky. Na toto jméno se pak můžeme odvolávat v odkazech a docílit tak toho, že stránka schovaná za odkazem se zobrazí v jiném rámu.

Určení cílového rámu

U elementů, které definují odkazy (`A`, `LINK`, `AREA` a `FORM`), můžeme použít nový atribut `TARGET`, který určuje jméno rámu, ve kterém se má dokument zobrazit.

Použití si ukážeme na příkladě. Máme v HTML vytvořenu elektronickou příručku, která je rozdělena po kapitolách do několika stránek. Tyto stránky mají jména `kap1.html` až `kap7.html`. Obrazovku rozdělíme na dva rámy. V levém bude obsah příručky (`obsah.html`). Pokud v obsahu vybereme nějakou kapitolu, zobrazí se v pravém rámu. Na začátku bude pravý rám obsahovat první kapitolu. Nejprve musíme definovat rozložení rámů a určit počáteční dokumenty. Pravý rám rovnou pojmenujeme.

```
<HTML>
<HEAD>
  <TITLE>Elektronická příručka</TITLE>
</HEAD>
<FRAMESET COLS="30%, 70%">
  <FRAME SRC="obsah.html">
  <FRAME SRC="kap1.html" NAME="hlavni">
</FRAMESET>
</HTML>
```

Soubor `obsah.html` pak bude v jednotlivých odkazech obsahovat určení cílového rámu pomocí atributu `TARGET`:

```
<HTML>
<HEAD>
  <TITLE>Obsah příručky</TITLE>
</HEAD>
<BODY>
Obsah:
<OL>
  <LI><A HREF="kap1.html" TARGET="hlavni">Úvod</A>
  <LI><A HREF="kap2.html" TARGET="hlavni">Tvrdý alkohol</A>
  <LI><A HREF="kap3.html" TARGET="hlavni">Vína</A>
  <LI><A HREF="kap4.html" TARGET="hlavni">Pivo</A>
  <LI><A HREF="kap5.html" TARGET="hlavni">Koktejly</A>
  <LI><A HREF="kap6.html" TARGET="hlavni">Bowle</A>
  <LI><A HREF="kap7.html" TARGET="hlavni">Jak pít?</A>
</OL>
</BODY>
</HTML>
```

Pokud dokument obsahuje mnoho odkazů se stejnou hodnotou atributu TARGET, můžeme si práci ušetřit tím, že jméno cílového rámu specifikujeme u elementu BASE:

```
<HTML>
<HEAD>
  <TITLE>Obsah příručky</TITLE>
  <BASE TARGET="hlavni">
</HEAD>
<BODY>
Obsah:
<OL>
  <LI><A HREF="kap1.html">Úvod</A>
  <LI><A HREF="kap2.html">Tvrdý alkohol</A>
  <LI><A HREF="kap3.html">Vína</A>
  <LI><A HREF="kap4.html">Pivo</A>
  <LI><A HREF="kap5.html">Koktejly</A>
  <LI><A HREF="kap6.html">Bowle</A>
  <LI><A HREF="kap7.html">Jak pít?</A>
</OL>
</BODY>
</HTML>
```

Použití atributu TARGET u odkazu má větší váhu než u elementu BASE. Pokud cílové okno není specifikováno, dokument se nahraje do stejného rámu, v jakém na něj byl odkaz. Pokud odkaz obsahuje jako cíl rám, který neexistuje, prohlížeč otevře nové okno a přiřadí mu toto jméno.

Jako jméno cílového rámu můžeme použít i některé ze čtyř předdefinovaných jmen _blank, _self, _parent a _top. Jejich význam je popsán v tabulce 13-2.

Jméno	Popis
_blank	Dokument se zobrazí v novém nepojmenovaném okně.
_self	Dokument se zobrazí ve stejném okně. (To se může hodit, pokud jsme použili <BASE TARGET=*«jméno»*>.)
_parent	Dokument se zobrazí v rámu nebo okně, které obsahuje nejbližší nadřazený element FRAMESET.
_top	Dokument se nahraje do celého okna prohlížeče; všechny rámy se zruší.

Tab. 13-2: Speciální jména rámů

Alternativa pro staré prohlížeče

Jelikož stránka, která definuje rozložení rámů, je jinak prázdná, v prohlížečích, které nepodporují rámy, se nic nezobrazí. Tomuto jistě nežádoucímu efektu lze snadno zabránit vložením alternativního textu do elementu BODY.

```
<HTML>
<HEAD>
  <TITLE>Elektronická příručka</TITLE>
</HEAD>
<FRAMESET COLS="30%, 70%">
  <FRAME SRC="obsah.html">
  <FRAME SRC="kap1.html" NAME="hlavni">
</FRAMESET>
<BODY>
Váš prohlížeč nepodporuje rámy. To je velká škoda, protože
s nimi je prohlížení naší příručky úplná hračka. I přesto se
však podívejte na <A HREF="obsah.html">její obsah</A>, odkud
se dostanete na jednotlivé kapitoly.
</BODY>
</HTML>
```

Kromě této možnosti můžeme v libovolném dokumentu použít element NOFRAMES. Jeho obsah se zobrazí pouze v prohlížečích, které nepodporují rámy. Do jednoho dokumentu tak můžeme doplnit kousíčky HTML, které stránku zpřístupní i starším prohlížečům.

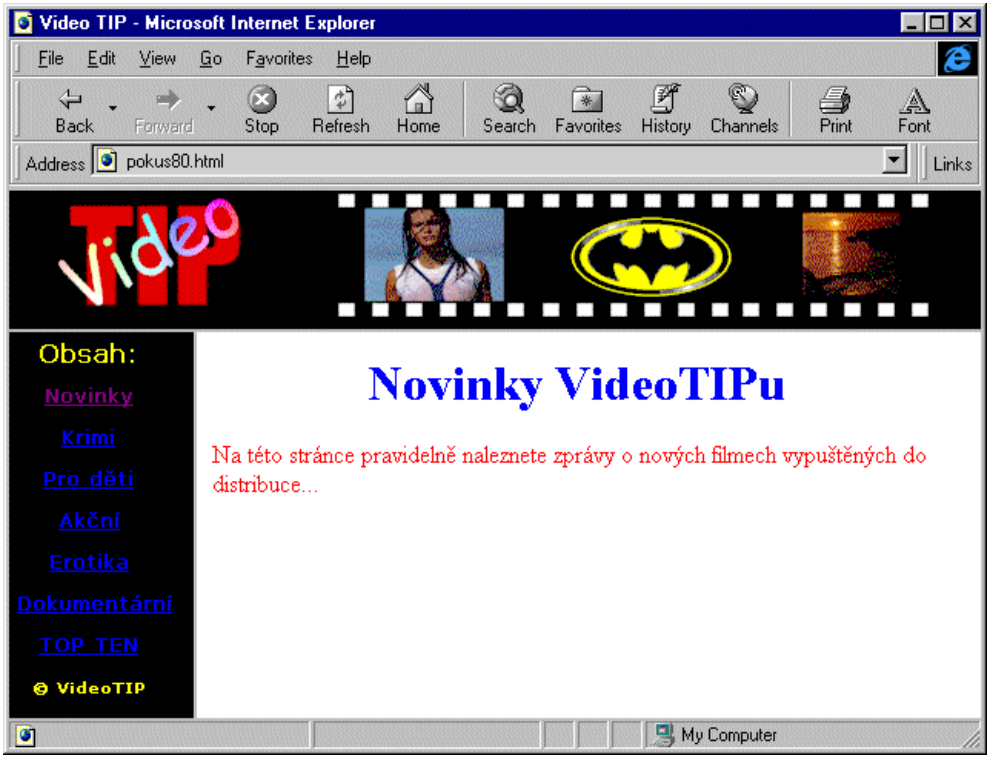
Typickým příkladem je situace, kdy máme dlouhou stránku, a proto k ní přidáme rám, který bude obsahovat odkazy na její jednotlivé části. V tomto případě je užitečné obsah zařadit i přímo na začátek dlouhé stránky mezi tagy <NOFRAMES> a </NOFRAMES>.

☺ Věrni heslu Internetu „Buď konzervativní v tom, co poskytuješ, a tolerantní k tomu, co přijímáš“, bychom měli vždy své stránky s rámy upravovat tak, aby byly přístupné i pro prohlížeče bez podpory rámů.

Praktická ukázka použití rámů

Nyní si prakticky ukážeme nejčastější způsob použití rámů. Pomocí rámů rozdělíme stránku na tři části — v horní bude proužek s logem, vlevo bude obsah a v pravo bude největší část okna vyhrazena pro vlastní stránky (viz obr. 13-2 na následující straně).

☺ Dělit stránku na více než tři části již není příliš vhodné. Jednotlivé rámy jsou pak tak malé, že se do nich pomalu nic nevejde (zvláště na obrazovkách s rozlišením 640×480). Ve velkém počtu rámů se navíc špatně orientuje i uživatel — zvláště pokud se po výběru odkazu změní obsah více než jednoho rámu.



Obr. 13-2: Stránka s rámy

Počáteční stránka, která rozdělila okno prohlížeče na rámy, je vskutku jednoduchá a krátká:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Draft//EN">
<HTML>
<HEAD>
<TITLE>Video TIP</TITLE>
</HEAD>
<FRAMESET ROWS="90,*">
  <FRAME SRC="banner.html" MARGINWIDTH=0 MARGINHEIGHT=0
         NORESIZE SCROLLING=NO FRAMEBORDER=0>
```

```
    <FRAMESET COLS="120,*">
      <FRAME SRC="obsah.html" MARGINWIDTH=5 FRAMEBORDER=0>
      <FRAME SRC="novinky.html" NAME="hlavni" FRAMEBORDER=0>
    </FRAMESET>
  </FRAMESET>
  <BODY>
  <H1>Video TIP</H1>
  Váš prohlížeč nepodporuje rámy. Pokračujte
  <A HREF="obsah.html">obsahem</A>.
  </BODY>
  </HTML>
```

Stránka s obsahem `obsah.html` vypadá takto:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Draft//EN">
<HTML>
<HEAD>
<TITLE>Obsah Video TIPu</TITLE>
<BASE TARGET="hlavni">
</HEAD>
<BODY BGCOLOR=BLACK>
<DIV STYLE="color: yellow; font: bold 10pt/20pt Verdana, Arial;
            text-align: center">
<BIG>Obsah:</BIG><BR>
<A HREF=novinky.html>Novinky</A><BR>
<A HREF=krimy.html>Krimi</A><BR>
<A HREF=deti.html>Pro děti</A><BR>
<A HREF=akcni.html>Akční</A><BR>
<A HREF=ero.html>Erotika</A><BR>
<A HREF=dokumenty.html>Dokumentární</A><BR>
<A HREF=top.html>TOP TEN</A><BR>
<SMALL>&copy; VideoTIP</SMALL>
</DIV>
</BODY>
</HTML>
```

A nakonec kostra stránek se samotnými informacemi vypadá takto (soubor `novinky.html`):

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Draft//EN">
<HTML>
<HEAD>
<TITLE>Novinky VideoTIPu</TITLE>
</HEAD>
```

```
<BODY BGCOLOR=WHITE TEXT=RED>
<H1 STYLE="color: blue; text-align: center">
Novinky VideoTIPu</H1>
Na této stránce pravidelně naleznete zprávy o nových filmech
vypuštěných do distribuce...
</BODY>
</HTML>
```

Inline rámy

Zatím jsme se zabývali rámy, které si rozdělí celou plochu okna prohlížeče. Microsoft přišel s dalším druhem rámů, které se staly součástí návrhu HTML 4.0. Tyto rámy se chovají podobně jako obrázky — na stránce se zobrazí jako obdélníkový prostor. V tomto prostoru pak může být zobrazena jiná stránka. Tyto inline rámy se do stránky vkládají pomocí elementu `IFRAME`. Můžeme u něj použít stejné atributy jako u `FRAME` s výjimkou `NORESIZE`. Navíc můžeme, podobně jako u obrázků, určit výšku `HEIGHT`, šířku `WIDTH` a způsob zarovnání `ALIGN`.

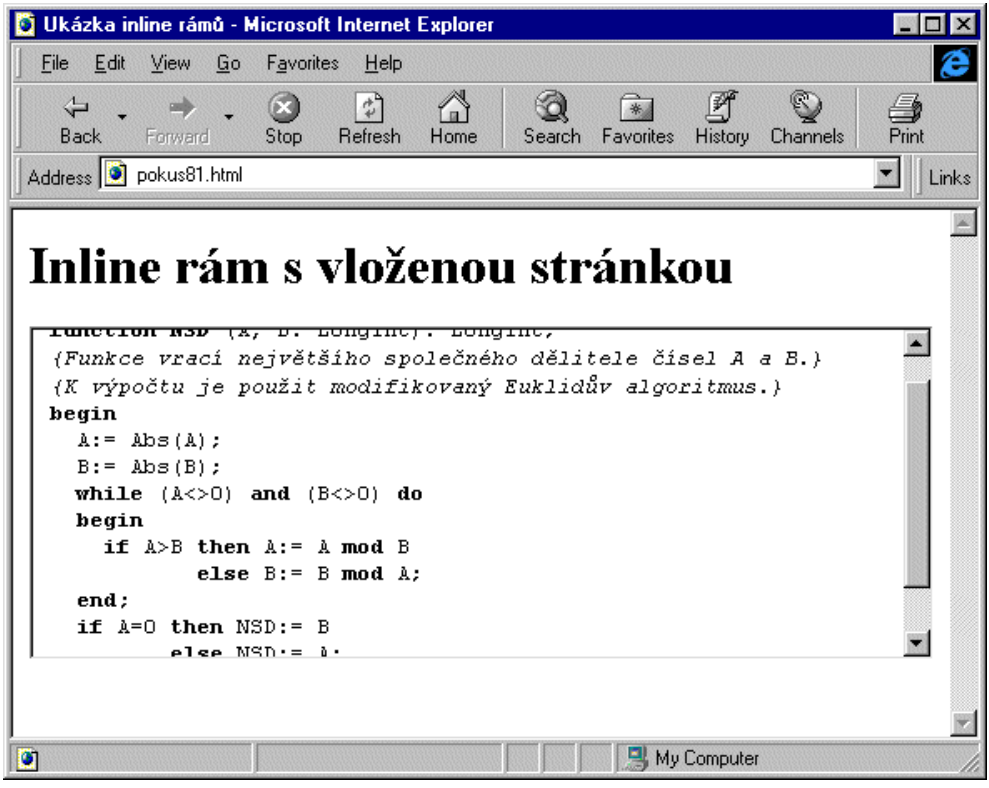
Text mezi `<IFRAME>` a `</IFRAME>` se zobrazuje pouze v prohlížečích, které inline rámy nepodporují. Je to to pravé místo pro umístění normálního odkazu na text v rámu pro prohlížeče, které rámy neumí:

```
<IFRAME SRC="soubor.html" WIDTH="100%" HEIGHT=200
        FRAMEBORDER=1 SCROLLING=AUTO>
Váš prohlížeč nepodporuje rámy. V rámu je zobrazena stránka
s <A HREF="soubor.html">ukázkou programu</A>.
</IFRAME>
```

13.4 Rozšířený model tabulek

Tabulky, tak jak je známe z HTML 3.2, jsou pouze zjednodušenou verzí tabulek podle specifikace v RFC 1942. Rozšířený model tabulek nabízí vylepšení zejména v možnosti formátování tabulek, rychlosti vykreslení dlouhých tabulek a ve snazší navigaci uživatele po dlouhých tabulkách. Tato rozšíření jsou součástí návrhu HTML 4.0 a již dnes je podporuje například *Internet Explorer*.

Rozšířený model tabulek vychází ze stejných principů jako ten, který už známe. Navíc je zpětně kompatibilní s HTML 3.2 — naše tabulky tedy vyhovují i tomuto novému modelu — pouze nevyužívají všechny jeho možnosti.



Obr. 13-3: Zobrazení inline rámu v prohlížeči

Skupiny řádek

První zlepšení přichází v možnosti rozčlenit tabulku na tři části: záhlaví `THEAD`, patu `TFOOT` a tělo `TBODY`. To má význam zejména u dlouhých tabulek. Záhlaví a pata, které obsahují důležité identifikační informace, mohou být pořád zobrazeny, zatímco samotné tělo tabulky roluje. Při tisku může být u tabulky, která zabere několik stran, na každé straně zopakováno záhlaví a pata. Malá ukázka dělení tabulky do skupin řádek:

```
<TABLE>
<THEAD>
  <TR><TH>Výrobek<TH>Cena<TH>Prodané ks<TH>Tržba
</THEAD>
<TFOOT>
  <TR><TH COLSPAN=3>Celkem<TH>123 567,20
</TFOOT>
<TBODY>
```

```
  <TR><TD>A<TD>23,50<TD>3<TD>70,50
  <TR><TD>B<TD>12,50<TD>12<TD>150
  «zde pokračuje několik desítek dalších řádek tabulky»
</TBODY>
</TABLE>
```

Při zápisu nemusíme používat ukončovací tagy `</THEAD>`, `</TFOOT>` a `</TBODY>`; prohlížeč si je sám domyslí:

```
<TABLE>
<THEAD>
  <TR><TH>Výrobek<TH>Cena<TH>Prodané ks<TH>Tržba
<TFOOT>
  <TR><TH COLSPAN=3>Celkem<TH>123 567,20
<TBODY>
  <TR><TD>A<TD>23,50<TD>3<TD>70,50
  <TR><TD>B<TD>12,50<TD>12<TD>150
  «zde pokračuje několik desítek dalších řádek tabulky»
</TABLE>
```

Velice důležité je pořadí, v jakém jsou jednotlivé skupiny použity. První musí být záhlaví, poté pata a až na konci následuje samotné tělo tabulky. To může být rozděleno na několik částí opakovaným použitím elementu `TBODY`.

Skupiny sloupců

Při zobrazování dlouhých tabulek je největším problémem zjištění počtu sloupců a jejich šířek. Aby prohlížeč nemusel čekat na přenesení celé tabulky, můžeme mu tyto údaje sdělit pomocí elementů `COLGROUP` a `COL`. Musíme je však použít ještě před první řádkou tabulky.

Element `COLGROUP` definuje skupinu sloupců. Obvykle pomocí ní označíme sloupce, které mají něco společného. Počet sloupců ve skupině se určuje pomocí atributu `SPAN`. Atributem `WIDTH` můžeme určit šířku sloupců ve skupině. Jako šířku je možno kromě počtu pixelů a procenta zadat speciální šířku '`0*`'. Ta prohlížeči říká, že šířka sloupec má mít minimální šířku, do které se ještě vejde obsah buněk ve sloupci. Pro skupinu sloupců můžeme určit společný způsob horizontálního (atribut `ALIGN`) i vertikálního zarovnání (atribut `VALIGN`).

Následující tabulka obsahuje dvě skupiny sloupců. První skupina má tři sloupce o šířce 100 pixelů. Druhá skupina obsahuje čtyři sloupce, které budou zobrazeny co nejužší a jejich obsah bude vycentrován.


```
<TABLE>
<COLGROUP SPAN=3 WIDTH=100>
<COLGROUP SPAN=4 WIDTH="0*" ALIGN=CENTER>
```

```
<THEAD>
  ....
</TABLE>
```

Pokud chceme mít každý sloupec ve skupině jinak široký nebo zarovnaný odlišným způsobem, nepoužijeme u tagu `<COLGROUP>` atribut `SPAN`. Místo toho počet sloupců ve skupině určíme počtem elementů `COL`, které budou obsaženy v `COLGROUP`. Následující příklad definuje v tabulce dvě skupiny sloupců. V první skupině budou dva sloupce o šířce 200 pixelů a o minimální šířce. Druhá skupina bude obsahovat tři sloupce se shodnou šířkou 50 pixelů:

```
<COLGROUP>
  <COL WIDTH=200>
  <COL WIDTH="0*">
<COLGROUP SPAN=3 WIDTH=50>
```

Jako hodnotu atributu `WIDTH` u elementu `COL` můžeme použít i výraz ve tvaru '`i*`', kde '`i`' je celé číslo. Sloupcům s takto určenou šířkou se jejich šířka určuje až na konec. Zbylé místo si rozdělí tak, aby poměry šířek sloupců odpovídaly poměrům čísla '`i`'. Hodnota '`*`' odpovídá hodnotě '`1*`'.

Pro sloupec můžeme rovněž určit společný způsob zarovnání pomocí atributů `ALIGN` a `VALIGN`.

Pokud chceme pro více sloupců nastavit šířku a zarovnání společné, můžeme použít atribut `SPAN` i u `COL`. V tomto případě však nedefinujeme novou skupinu sloupců jako u `COLGROUP`, ale pouze říkáme, že několik sloupců ve skupině bude mít stejné parametry. Rozdělení na skupiny pomocí `COLGROUP` je důležité pro správné vykreslení mřížky uvnitř tabulky, jak si za chvíli ukážeme.

Pokud jako hodnotu `SPAN` uvedeme nulu, použije se šířka a způsob zarovnání pro všechny zbývající sloupce tabulky. Tuto vlastnost bychom však měli používat pouze společně s atributem `COLS` elementu `TABLE`, protože jinak se prohlížeč na začátku tabulky nedozví skutečný počet sloupců. Atribut `COLS` totiž určuje celkový počet sloupců tabulky.

Označování buněk tabulky

Při převodu obsahu tabulky do mluvené řeči nebo do databázové tabulky se teoreticky může hodit jednotlivé buňky tabulky lépe označit. Proto nyní u všech buněk (`TH` a `TD`) můžeme použít dva nové atributy `AXIS` a `AXES`.

`AXIS` definuje jméno buňky. Pokud jej nepoužijeme, jméno buňky je shodné s jejím obsahem.

Obsahem atributu `AXES` je čárkami oddělený seznam jmen definovaných pomocí `AXIS`, ke kterým se obsah buňky váže.

Abychom lépe porozuměli této nepochopitelné a v praxi asi neupotřebitelné vlastnosti, zde je malá ukázka:

```
<TABLE BORDER=1>
<CAPTION>Spotřeba kávy jednotlivých senátorů</CAPTION>
<TR><TH>Jméno   <TH>Počet šálků turka  <TH>Počet šálků espresa
<TR><TD AXIS="Novák" AXES="Jméno">Karel Novák<TD>5<TD>10
<TR><TD AXIS="Procházka" AXES="Jméno">Jan Procházka<TD>1<TD>7
 ...
</TABLE>
```

Zarovnání buněk

Možnost ovlivnit zarovnání buněk je v novém modelu tabulek také rozšířeno. Vertikální zarovnání se ovládá atributem `VALIGN`. Kromě nám již známých hodnot `TOP`, `MIDDLE` a `BOTTOM` můžeme použít i hodnotu `BASELINE`. Pokud mají v jedné řádce tabulky některé buňky nastaveno `VALIGN=BASELINE`, budou účaří prvních řádek obsahu těchto buněk ve stejné výšce.

Rozšířeny jsou i možnosti atributu `ALIGN`. Nově lze použít dvě nové hodnoty atributu — `JUSTIFY` a `CHAR`. Použijeme-li `JUSTIFY`, bude obsah buňky zarovnán do bloku.

Způsob zarovnání `CHAR` využijeme zejména ve sloupcích, které obsahují číselné hodnoty. Pomocí nového atributu `CHAR` můžeme v tomto případě vybrat znak, který bude v buňkách pod sebou lícovat. Pokud nastavíme tento znak na desetinnou čárku, budou čísla vyrovnána pod sebou tak, jak jsme na to zvyklí. Malá ukázka:

```
<TABLE BORDER=1>
<CAPTION>Přehled průměrných cen vybraného ovoce</CAPTION>
<COLGROUP SPAN=1>
<COLGROUP SPAN=1 ALIGN=CHAR CHAR=",">
<THEAD>
<TR><TH>Ovoce<TH ALIGN=CENTER>Cena v Kč
<TBODY>
<TR><TD>Jablka<TD>25,30
<TR><TD>Hroznové víno<TD>46
<TR><TD>Banány<TD>23,90
<TR><TD>Jahody<TD>46,20
<TR><TD>Bílý meloun<TD>151,80
</TABLE>
```

Přehled průměrných cen vybraného ovoce

Ovoce	Cena v Kč
Jablka	25,30
Hroznové víno	46
Banány	23,90
Jahody	46,20
Bílý meloun	151,80

Vidíme, že čísla jsou zarovnána přehledně pod sebou podle desetinné čárky. Pomocí atributu `CHAR` můžeme určit libovolný znak, jehož první výskyt bude ve všech buňkách sloupce vždy pod sebou. Pokud nějaká buňka daný znak neobsahuje, zarovná se v pravo k místu, kde je v ostatních buňkách znak určený pomocí `CHAR`.

Pozici znaku pro zarovnání v buňce můžeme ovlivňovat pomocí atributu `CHAROFF`. Jako hodnota atributu se uvádí vzdálenost znaku od levého okraje buňky. Může použít buď pixely nebo procento. Pokud tedy chceme, aby desetinná čárka byla uprostřed sloupce, použije něco jako:

```
ALIGN=CHAR CHAR="," CHAROFF="50%"
```

Způsob zarovnání `ALIGN=CHAR` zatím ani *Netscape* ani *Explorer* neumí. Pokud se nám tabulka tedy nezobrazí tak, jak jsme chtěli, nemusí to ještě být naše chyba.

Výše popsané atributy řídící způsob zarovnání můžeme použít u následujících elementů: `TD`, `TH`, `COL`, `COLGROUP`, `TR`, `THEAD`, `TFOOT` a `TBODY`. Elementy uvedené v tomto výčtu dříve mají větší váhu. Nastavení způsobu zarovnání u nich překryje nastavení provedené u jiného elementu.

Rámečky a mřížka

U elementu `TABLE` nyní můžeme použít nový atribut `FRAME`, který určuje způsob vykreslení rámečku okolo tabulky. Jeho přípustné hodnoty shrnuje tabulka 13-3 na následující straně.

O tom, kde a jak se v tabulce vykreslí mřížka, rozhoduje atribut `RULES`. Přípustné hodnoty nalezneme v tabulce 13-4 na následující straně.

Sílu rámečku lze nastavit opět pomocí atributu `BORDER`. Pokud použijeme u tagu `<TABLE>` pouze atribut `BORDER=0` a nepoužijeme `RULES` a `FRAME`, tabulka se zobrazí, jako by bylo nastaveno `FRAME=VOID` a `RULES=NONE`. Pokud použijeme pouze nenulovou hodnotu atributu `BORDER`, použije se `FRAME=BORDER` a `RULES=ALL`.

FRAME	Popis
VOID	Tabulka je bez rámečku. (Standardní hodnota.)
ABOVE	Rámeček je pouze na horní straně tabulky.
BELOW	Rámeček je pouze na spodní straně tabulky.
HSIDES	Rámeček je pouze na horní a spodní straně tabulky.
VSIDES	Rámeček je pouze na levé a pravé straně tabulky.
LHS	Rámeček je pouze na levé straně tabulky.
RHS	Rámeček je pouze na pravé straně tabulky.
BOX	Rámeček je okolo celé tabulky.
BORDER	Rámeček je okolo celé tabulky.

Tab. 13-3: Hodnoty atributu FRAME

RULES	Popis
NONE	Tabulka je bez mřížky. (Standardní hodnota.)
GROUPS	Mřížka bude pouze mezi skupinami řádek (THEAD, TFOOT a TBODY) a mezi skupinami sloupců (COLGROUP).
ROWS	Mřížka bude pouze mezi řádky tabulky.
COLS	Mřížka bude pouze mezi sloupci tabulky.
ALL	Mřížka bude mezi všemi buňkami.

Tab. 13-4: Hodnoty atributu RULES

Barva pozadí tabulky a buněk

U elementů TABLE, TR, TD a TH lze nyní použít atribut BGCOLOR, který slouží k určení barvy pozadí tabulky, řádky či jen jedné buňky tabulky.

Příkladná tabulka

Praktické využití novinek, které nový model tabulek přináší, si ukážeme na příkladě. Výsledné zobrazení v prohlížeči přináší obrázek 13-4 na následující straně.

```
<TABLE FRAME=HSIDES RULES=GROUPS CELLPADDING=2>
<CAPTION>Formáty papíru podle ČSN 50 0040</CAPTION>
<COLGROUP>
        <COL ALIGN=LEFT>
        <COL ALIGN=CHAR CHAR="&times;" CHAROFF="50%">
<COLGROUP>
        <COL ALIGN=LEFT>
```

Formáty papíru podle ČSN 50 0040

Základní řada		Doplňkové řady			
A	čisté rozměry v mm	**B**	čisté rozměry v mm	**C**	čisté rozměry v mm
A0	841 × 1189	B0	1000 × 1414	C0	917 × 1297
A1	594 × 841	B1	707 × 1000	C1	648 × 917
A2	420 × 594	B2	500 × 707	C2	458 × 648
A3	297 × 420	B3	353 × 500	C3	324 × 458
A4	210 × 297	B4	250 × 353	C4	229 × 324
A5	148 × 210	B5	176 × 250	C5	162 × 229
A6	105 × 148	B6	125 × 176	C6	114 × 162
A7	74 × 105	B7	88 × 125	C7	81 × 114
A8	52 × 74	B8	62 × 88	C8	57 × 81
A9	37 × 52	B9	44 × 31	C9	40 × 57
A10	26 × 37	B10	31 × 44	C10	28 × 40

Obr. 13-4: Ukázka zformátované tabulky

```
        <COL ALIGN=CHAR CHAR="&times;" CHAROFF="50%">
<COLGROUP>
        <COL ALIGN=LEFT>
        <COL ALIGN=CHAR CHAR="&times;" CHAROFF="50%">
<THEAD>
<TR><TH COLSPAN=2 ALIGN=CENTER>Základní řada
    <TH COLSPAN=4 ALIGN=CENTER>Doplňkové řady
<TBODY>
<TR><TH>A<TD ALIGN=CENTER>čisté rozměry v mm
    <TH>B<TD ALIGN=CENTER>čisté rozměry v mm
    <TH>C<TD ALIGN=CENTER>čisté rozměry v mm
<TBODY>
<TR><TD>A0<TD>841 &times; 1189
    <TD>B0<TD>1000 &times; 1414
    <TD>C0<TD>917 &times; 1297
<TR><TD>A1<TD>594 &times; 841
    <TD>B1<TD>707 &times; 1000
    <TD>C1<TD>648 &times; 917
<TR><TD>A2<TD>420 &times; 594
    <TD>B2<TD>500 &times; 707
    <TD>C2<TD>458 &times; 648
<TR><TD>A3<TD>297 &times; 420
```

```
    <TD>B3<TD>353 &times; 500
    <TD>C3<TD>324 &times; 458
<TR><TD>A4<TD>210 &times; 297
    <TD>B4<TD>250 &times; 353
    <TD>C4<TD>229 &times; 324
<TR><TD>A5<TD>148 &times; 210
    <TD>B5<TD>176 &times; 250
    <TD>C5<TD>162 &times; 229
<TR><TD>A6<TD>105 &times; 148
    <TD>B6<TD>125 &times; 176
    <TD>C6<TD>114 &times; 162
<TR><TD>A7<TD>74 &times; 105
    <TD>B7<TD>88 &times; 125
    <TD>C7<TD>81 &times; 114
<TR><TD>A8<TD>52 &times; 74
    <TD>B8<TD>62 &times; 88
    <TD>C8<TD>57 &times; 81
<TR><TD>A9<TD>37 &times; 52
    <TD>B9<TD>44 &times; 31
    <TD>C9<TD>40 &times; 57
<TR><TD>A10<TD>26 &times; 37
    <TD>B10<TD>31 &times; 44
    <TD>C10<TD>28 &times; 40
</TABLE>
```

13.5 Skripty

Poprvé zabudovala skripty do svého prohlížeče firma Netscape. Ta umožnila přímo do stránky vkládat programy — *skripty* v jazyce JavaScript. Tyto programy však narozdíl od CGI-skriptů vykonával až prohlížeč. Odezva na akce uživatele tedy byla okamžitá v porovnání s pomalou komunikací s WWW-serverem. Skripty mohly být použity k mnoha účelům:

- modifikování dokumentu v průběhu jeho nahrávání;
- kontrolování správnosti dat vkládaných do formulářů;
- inteligentnímu chování formulářů, které mohou automaticky doplňovat některé chybějící hodnoty;
- vytváření stránek, které mají interaktivnější uživatelské rozhraní (odkazy mění po přejetí barvu apod.).

Skripty se ukázaly býti natolik užitečnou pomůckou, že do HTML byla přidána podpora pro vkládání skriptů nezávislá na použitém skriptovacím jazyce. Kromě JavaScriptu se dnes totiž na stránkách dnes používá i VBScript, což je speciálně upravená verze Visual Basicu od Microsoftu. Existují i další skriptovací jazyky — např. Tcl. Ovšem ne všechny prohlížeče podporují všechny skriptovací jazyky. *Netscape* podporuje JavaScript a *Explorer* podporuje jak VBScript tak JavaScript.[1] Pokud se rozhodneme používat skripty na našich stránkách, je v současné době nejlepším řešením JavaScript, protože je podporován oběma leadery na poli prohlížečů.

Podívejme se však na prostředky, které nám HTML nabízí pro vkládání skriptů do stránek. Skripty můžeme v zásadě rozdělit na dvě skupiny:

- Skripty, které jsou vykonány při nahrání stránky do prohlížeče. Tyto skripty se do stránky vkládají pomocí elementu `SCRIPT`.
- Skripty, které jsou vykonány vždy, když se vyskytne určitá událost. Událostí existuje několik a spojují se vždy s určitým elementem. Událostí může být například nastavení ukazovátka myši nad odkaz.

Vložení skriptu do stránky

Skript se do stránky vkládá pomocí párového elementu `SCRIPT`. Skript můžeme vložit do stránky buď přímo nebo uvést pouze odkaz na URL, které obsahuje program v příslušném skriptovacím jazyce. Pro určení typu skriptovací jazyka bychom měli použít atribut `LANGUAGE`.[2] Vložení jednoduchého skriptu může vypadat např. takto:

```
<SCRIPT LANGUAGE="JavaScript">
  window.status = "Právě si prohlížíš moji stránku."
</SCRIPT>
```

Na probrání všech možností JavaScriptu zde bohužel nemáme prostor. Jen pro informaci čtenáře — výše uvedený skript zobrazí zadaný text ve stavové řádce.

Druhou možností je vložit skript pouze pomocí odkazu. I v tomto případě však musíme použít ukončovací tag `</SCRIPT>`:

```
<SCRIPT LANGUAGE="VBScript"
        SRC="http://microsoft.com/exmpl/vbdemo">
</SCRIPT>
```


[1] *Explorer* ve skutečnosti nepodporuje JavaScript, ale JScript. JScript je rozšířená implementace JavaScriptu z dílny Microsoftu. Naštěstí je s JavaScriptem zpětně kompatibilní.

[2] Návrh HTML 4.0 doporučuje místo atributu `LANGUAGE` používat obecnější určení druhu skriptovacího jazyka pomocí atributu `TYPE`. Jeho hodnotou pak bude např. `text/javascript` nebo `text/vbscript`.

Skripty mohou být na stránce umístěny kdekoliv — v záhlaví i v těle dokumentu. Do záhlaví bychom měli umístit vše, co musí být vykonáno ještě dříve, než začne uživatel pracovat s dokumentem — např. definice funkcí, které se vyvolávají různými událostmi.

Při vkládání skriptů je rovněž vhodné jejich obsah skrýt před staršími prohlížeči pomocí nám již známého triku s komentáři:

```
<HEAD>
<TITLE>Stránka, která vám popíše stavovou řádku</TITLE>
<SCRIPT LANGUAGE="JavaScript"><!--
  window.status = "Právě si prohlížíš mojí stránku."
--></SCRIPT>
```

Pokud někde používáme skripty pro dynamické vygenerování části dokumentu, můžeme použít element `NOSCRIPT` a umístit do něj alternativní text pro prohlížeče, které nepodporují skripty:

```
<BODY>
<SCRIPT LANGUAGE="JavaScript"><!--
  document.write("Používáte prohlížeč " + navigator.appName
                                  + " " + navigator.appVersion)
--></SCRIPT>
<NOSCRIPT>
Používáte prohlížeč, který nepodporuje JavaScript.
</NOSCRIPT>
</BODY>
```

Můj prohlížeč tuto stránku zobrazil následovně:

Používáte prohlížeč Microsoft Internet Explorer 4.0 (compatible; MSIE 4.0b2; Windows NT)

Události a skripty

Aby byly naše dokumenty opravdu interaktivní, je potřeba, aby provedení určité části skriptu bylo vyvoláno *událostí*, kterou způsobil uživatel. Událostí může být přejetí myší přes určitý element, kliknutí myší apod.

U každého elementu můžeme použít několik nových atributů, které odpovídají jednotlivým událostem. Jako jejich hodnota se uvádí příkazy, které se po vyvolání dané události mají provést — tzv. *obsluha události* (angl. event-handler). Obecně se pak vše zapisuje takto:

`<TAG` *«událost»=«obsluha události»*`>`

Událost	Popis
onLoad	Událost je vyvolána po natažení dokumentu do okna prohlížeče nebo do všech rámů v rámci jednoho FRAMESET. Atribut může být použit u elementů BODY a FRAMESET.
onUnLoad	Událost je vyvolána po odstranění dokumentu z okna nebo rámu. Atribut může být použit s elementy BODY a FRAMESET.
onClick	Událost je vyvolána po kliknutí myší na element. Atribut může být použit u většiny elementů.
onDblClick	Událost je vyvolána po dvojitém kliknutí myší na element. Atribut může být použit u většiny elementů.
onMouseDown	Událost je vyvolána po stisknutí tlačítka myši nad elementem. Atribut může být použit u většiny elementů.
onMouseUp	Událost je vyvolána po uvolnění tlačítka myši nad elementem. Atribut může být použit u většiny elementů.
onMouseOver	Událost je vyvolána při přesunutí myši nad element. Atribut může být použit u většiny elementů.
onMouseMove	Událost je vyvolána při pohybu myši nad elementem. Atribut může být použit u většiny elementů.
onMouseOut	Událost je vyvolána po odsunutí myši z elementu. Atribut může být použit u většiny elementů.
onFocus	Událost je vyvolána v okamžiku, kdy je element aktivován myší nebo pomocí tabulátoru. Atribut je možno použít u elementů LABEL, INPUT, SELECT, TEXTAREA a BUTTON.
onBlur	Událost je vyvolána v okamžiku, kdy element přestává být aktivní. Atribut je možno použít u elementů LABEL, INPUT, SELECT, TEXTAREA a BUTTON.
onKeyPress	Událost je vyvolána po stisku a následném uvolnění tlačítka na klávesnici. Atribut může být použit u většiny elementů.
onKeyDown	Událost je vyvolána po stisku tlačítka na klávesnici. Atribut může být použit u většiny elementů.
onKeyUp	Událost je vyvolána po uvolnění tlačítka na klávesnici. Atribut může být použit u většiny elementů.
onSubmit	Událost je vyvolána při odesílání formuláře. Atribut může být použit pouze u elementu FORM.
onReset	Událost je vyvolána po vynulování formuláře. Atribut může být použit pouze u elementu FORM.
onSelect	Událost je vyvolána po označení textu ve vstupním poli. Atribut může být použit u elementů INPUT a TEXTAREA.
onChange	Událost je vyvolána pokud se změnila hodnota vstupního pole formuláře. Atribut může být použit u elementů INPUT, SELECT a TEXTAREA.

Tab. 13-5: Události, které lze obsloužit skriptem

Přehled událostí, které můžeme použít, uvádí tabulka 13-5 na straně 224. Krátkou exkurzi do světa skriptů ukončíme malou ukázkou, která využívá události. Po najetí myší nad odkaz se ve stavové řádce prohlížeče objeví vysvětlující text:

```
O rozvoj Webu se stará konsorcium
<A HREF="http://www.w3.org"
   onMouseOver="window.status='Kliknutím na odkaz se dostanete'
   + ' na domovskou stránku konsorcia W3C'; return true">
W3C</A>
```

Poznamenejme, že pro zachování přehlednosti HTML dokumentů je vhodné jako obsluhu události uvést pouze vyvolání dříve definované funkce, která zajistí všechny potřebné akce. Toto řešení nám kromě toho usnadní pozdější změny v chování obsluhy události.

13.6 Nové vlastnosti HTML 4.0

HTML 4.0 navazuje na předchozí verzi 3.2 a je s ní zpětně kompatibilní. Přesto však celý návrh specifikace prostupují údaje o atributech a elementech, které by se neměly používat. Jedná se zejména o atributy a elementy určující vzhled dokumentu (`ALIGN`, `<FONT>`). Návrh standardu HTML 4.0 se snaží dosáhnout původního záměru a představit HTML jako jazyk pro *vyznačování významu* jednotlivých částí dokumentu. Veškeré atributy, které ovlivňují *pouze vzhled*, by měly být definovány pomocí připojených stylů.

Kromě HTML 3.2 v sobě nová verze zahrnuje i rámy, rozšířený model tabulek podle RFC 1942 a podporu pro skripty a další novinky, na které se teď podíváme.

Podpora vícejazyčných dokumentů

Dosud v HTML chyběl nějaký standardní způsob, jak sdílet po celém světě dokumenty, které obsahují informace v různých jazycích. Přítrž tomuto nevyhovujícímu stavu chce učinit právě HTML 4.0. U každého elementu lze použít atributy `LANG` a `DIR`.

Atribut `LANG` slouží pro určení jazyka, kterým je zapsán obsah daného elementu. K určení jazyka se používají kódy uvedené v normě ISO 639. Pro angličtinu tak existuje kód `en`, pro americkou angličtinu `en-US` apod. Jazyk se dědí z nadřazeného elementu na všechny elementy v něm obsažené. Pokud by tedy dokument v angličtině obsahoval jednu citaci ve francouzštině, mohla by jeho kostra vypadat zhruba takto:

```
<BODY LANG="en">
....
<BLOCKQUOTE LANG="fr">
  «citace ve francouzštině»
</BLOCKQUOTE>
....
</BODY>
```

Atribut DIR určuje směr textu na řádce. Může nabývat dvou hodnot LTR a RTL. LTR určuje směr zleva doprava (ten je běžný i pro češtinu). V arabských jazycích se však používá opačný směr — zprava doleva, a v těchto případech použijeme hodnotu RTL. Směr textu se také dědí na podřazené elementy.

Použití atributu DIR u elementů říká, v jakém směru se má text zobrazit. Uložen však musí být vždy ve směru zleva doprava. To je nepohodlné pro autory, kteří zapisují text v jazyce, v němž se píše zprava doleva. Pro tyto případy lze použít nový element BDO. U něj atribut DIR určuje směr, jakým je text zapsán na stránce, a zároveň i směr, v jakém se text zobrazí v prohlížeči.

Zatímco atribut DIR slouží k určení směru toku textu, nemusí nám být praktický význam atribut LANG zřejmý. Podle jeho hodnoty by se však měly použít správné vzory dělení slov, používat specifické typografické zvyklosti daného jazyka jako závorky, uvozovky, ligatury atd.

Vyznačování textu

Různé elementy, které slouží pro vyznačování významu textu (STRONG, CITE, VAR apod.), jsou důležité, protože umožňují mnohem lepší prohledávání textů a lepší konverzi do kvalitní tištěné podoby. HTML 4.0 tyto možnosti dále rozšiřuje.

Přibyl zde nový element ACRONYM, který slouží k označování zkratek použitých v textu. Pomocí atributu TITLE můžeme zkratku doplnit o její úplné znění.

```
O masové rozšíření Internetu se zasloužila především služba
<ACRONYM TITLE="World-Wide Web">WWW</ACRONYM>.
```

Pokud jsme citovali delší pasáže textu, používali jsme k tomu element BLOCKQUOTE. Pro kratší citáty, jež mají být součástí běžného textu v odstavci, je vyhrazen nový element Q. U BLOCKQUOTE i u Q lze nyní použít atribut CITE, který obsahuje URL citovaného zdroje.

Nové jsou rovněž elementy INS a DEL, které slouží k označení textu, který byl do stránky přidán resp. z ní vymazán. Obsah elementu DEL se tedy v prohlížeči normálně nezobrazuje. U obou elementů lze použít atribut CITE, který má obsahovat URL dokumentu s vysvětlením důvodu provedené změny ve stránce. Rovněž můžeme použít atribut DATETIME, který obsahuje čas provedení změny. Čas provedení změny se udává ve formátu RRRR-MM-DDThh:mm:ssTZD. TZD je určení časové zóny (Z odpovídá univerzálnímu času, jinak se udává časový posuv

v hodinách a minutách). Poledne 19. září 1997 univerzálního času tedy zapíšeme jako `1997-09-19T12:00:00Z`.

Formátování textu

Ačkoliv HTML 4.0 upřednostňuje použití stylů pro řízení formátování, standardizuje několik běžně používaných atributů a jejich hodnot.

U atributu `ALIGN` lze nyní používat hodnotu `JUSTIFY`, která text zarovná do bloku. U `VALIGN` lze použít hodnotu `BASELINE`, která vyrovná účaří jednotlivých bloků textu.

U elementu `FONT` lze nyní použít atribut `FACE`, který určuje použitou rodinu písma:

```
<FONT FACE="Arial, Helvetica">Toto je bazpatkové písmo</FONT>
```

Vkládání objektů

Do HTML se postupně přidávaly elementy pro vkládání různých druhů objektů — obrázků, apletů apod. Protože množství objektů, které lze do stránky vložit, vzrůstá, celkem logicky verze 4.0 přináší jednotný způsob pro vkládání objektů libovolného typu. Element, který slouží k těmto účelům, je `OBJECT`. Jeho chování ovlivňuje mnoho atributů, my si popíšeme nejdůležitější z nich.

`OBJECT` lze využít pro vložení libovolného objektu — obrázku, Java-apletu, videosekvence, komponenty ActiveX, klikací mapy či další HTML stránky. Důležité je, že můžeme specifikovat jak program, který se pro zobrazení objektu použije, tak i data, která se programem zobrazí. K určení programu, který se postará o zobrazení objektu, slouží atribut `CLASSID`. Pro vložení Java-apletu můžeme použít následující zápis:

```
<OBJECT CLASSID="java:Earth.class">
Váš prohlížeč nepodporuje Javu.
</OBJECT>
```

Samotný obsah elementu se zobrazí pouze v případě, že prohlížeč neumí zobrazit objekt specifikovaný pomocí atributů elementu `OBJECT`.

Pokud potřebujeme určit základní URL, ke kterému se vztahuje `CLASSID`, použijeme podobně jako u elementu `APPLET` atribut `CODEBASE`. Pomocí atributu `CODETYPE` můžeme určit typ dat, která program určený pomocí `CLASSID` očekává. Pokud se typy dat neshodují, nemusí se prohlížeč zdržovat nahráváním programu pro zobrazení objektu.

Pro určení URL dat, která se mají zobrazit, slouží atribut `DATA`. Typ dat můžeme specifikovat pomocí atributu `TYPE`. Při vkládání obvyklých typů dat nemusíme určovat program pro jejich zobrazení. Mnoho prohlížečů si samo poradí s videem ve formátu MPEG a pro vložení videosekvence pak stačí:

```
<OBJECT DATA="http://www.mtv.com/clips/u2.mpeg"
        TYPE="application/mpeg">
Protože Váš prohlížeč nepodporuje MPEG, nevidíte
videoklip skupiny U2.
</OBJECT>
```

To, že se obsah elementu OBJECT zobrazuje pouze v případech, kdy prohlížeč neumí zobrazit objekt určený pomocí atributů, lze použít pro poskytnutí několika alternativních podob objektu:

```
<OBJECT CLASSID="java:Earth.class">
  <OBJECT DATA="Earth.mpeg" TYPE="application/mpeg">
    <OBJECT DATA="Earth.gif" TYPE="image/gif">
      Planeta Země tak, jak ji viděl první kosmonaut.
    </OBJECT>
  </OBJECT>
</OBJECT>
```

Tento kód spustí v prohlížečích, které podporují Javu, aplet s rotující planetou Zemí. Pokud prohlížeč neumí Javu, ale umí přehrávat MPEG, spustí se video s rotující planetou. Pokud ani s MPEGem neuspějeme, zobrazí se obrázek planety. Na znakových prohlížečích pak uvidíme pouze popis toho, co je na obrázku vidět.

Mezi další atributy, které lze u OBJECT použít, patří STANDBY. Jeho obsah prohlížeč zobrazí do té doby, než se přenese celý program a data příslušející k objektu.

Pomocí atributu DECLARE můžeme objekt pouze deklarovat. Jeho samotné vložení do stránky se provede následovným použitím elementu OBJECT s odkazem na deklarovaný objekt.

Zarovnání objektu s okolním textem lze ovlivňovat pomocí atributu ALIGN. Přípustné jsou hodnoty TEXTTOP, MIDDLE, TEXTMIDDLE, BASELINE, TEXTBOTTOM, LEFT, CENTER a RIGHT.

Stejný význam jako u obrázků mají i atributy HEIGHT, WIDTH, BORDER, HSPACE a VSPACE.

Parametry lze objektu předávat pomocí elementu PARAM. Ten má stejný význam a použití jako u Java-apletů. Přibyl zde pouze atribut VALUETYPE, který určuje typ hodnoty předávané parametrem. VALUETYPE=DATA znamená, že se obsah atributu VALUE předá přímo vloženému objektu. Hodnota VALUETYPE=REF nám říká, že VALUE obsahuje URL, které ukazuje na data, která se mají objektu předat. Typ těchto dat je určen atributem TYPE. Poslední možností je VALUETYPE=OBJECT. V tomto případě VALUE obsahuje z URL pouze fragment a odkazuje se na dříve deklarovaný objekt.

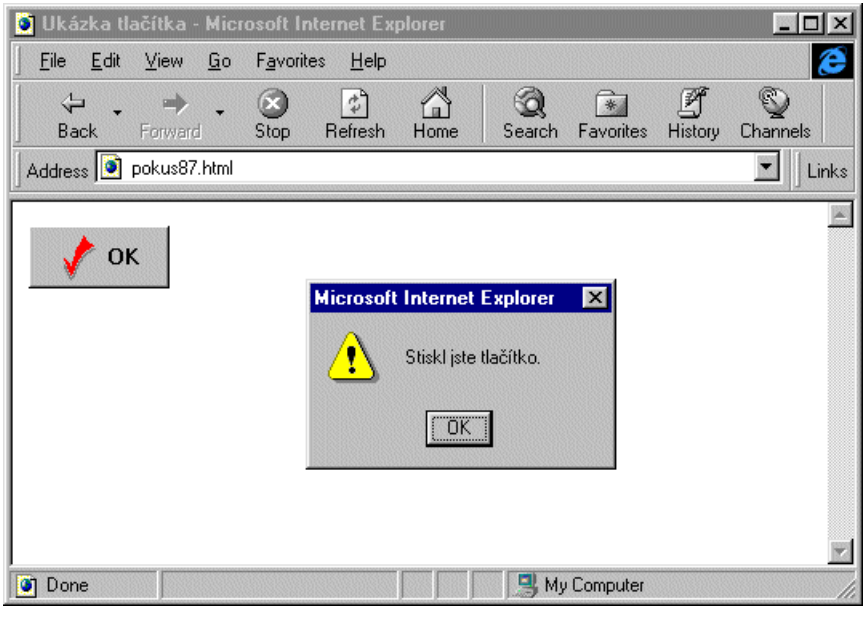
Formuláře

Na poli formulářů přináší HTML 4.0 jednak drobná rozšíření a jednak zcela novou funkčnost pomocí elementů `LABEL`, `FIELDSET` a `LEGEND`.

Mezi ty drobnější úpravy patří přidání nového typu vstupního pole. Použijeme-li `<INPUT TYPE=BUTTON ...>`, získáme tlačítko, které je možné pomocí události spojit s nějakým skriptem.

Ke stejným účelům slouží i nový element `BUTTON`. Ten definuje tlačítko. Obsah elementu je zobrazen jako popis tlačítka, takže zde můžeme bez problémů kombinovat text i obrázky. Typ tlačítka můžeme nastavit pomocí atributu `TYPE` na `BUTTON`, `SUBMIT` nebo `RESET` se zcela stejným významem jako u elementu `INPUT`.

```
<BUTTON TYPE="BUTTON" onClick="alert('Stiskl jste tlačítko.')">
<IMG SRC="check.gif" ALIGN=MIDDLE> <B>OK</B> 
</BUTTON>
```

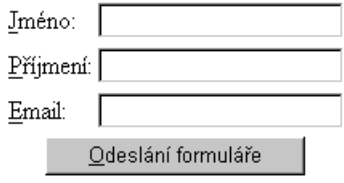


Obr. 13-5: Ukázka tlačítka s obrázkem

Nově nyní můžeme popis vstupního pole označit elementem `LABEL` a pomocí atributu `FOR` jej s polem spojit. Jako hodnota atributu `FOR` se uvádí `ID` vstupního pole. Toto spojení popisu se vstupním polem se využije při převodu do mluvené řeči.

Mnohem větší uplatnění nalezne element `LABEL` ve spojení s atributem `ACCESSKEY`. Jako hodnota tohoto atributu se uvádí písmeno, které bude použito pro klávesovou zkratku. Použijeme-li u `LABEL ACCESSKEY=N`, pomocí stisku Alt + N se kurzor okamžitě přesune do odpovídajícího vstupního pole. Malá ukázka:

```
<FORM ACTION="..." METHOD=POST>
<TABLE>
<TR><TD><LABEL FOR=Jmeno ACCESSKEY=J><U>J</U>méno:</LABEL>
    <TD><INPUT TYPE=TEXT ID=Jmeno>
<TR><TD><LABEL FOR=Prijmeni ACCESSKEY=P><U>P</U>říjmení:
    </LABEL>
    <TD><INPUT TYPE=TEXT ID=Prijmeni>
<TR><TD><LABEL FOR=email ACCESSKEY=E><U>E</U>mail:</LABEL>
    <TD><INPUT TYPE=TEXT ID=email>
<TR><TH COLSPAN=2><LABEL FOR=Send ACCESSKEY=O></LABEL>
    <BUTTON TYPE=SUBMIT ID=Send><U>O</U>deslání formuláře
    </BUTTON>
</TABLE>
</FORM>
```



V tomto formuláři se mezi jednotlivými vstupními poli můžeme pohybovat pomocí Alt + J, Alt + P a Alt + E. Stiskem Alt + O formulář odešleme. U popisů vstupních polí je dobré vyznačit horkou klávesu podtržením. Abychom mohli podtrhnout i písmeno klávesové zkratky pro odeslání formuláře, použili jsme pro vytvoření tlačítka `<BUTTON>` místo `<INPUT TYPE=Submit>`.

U rozsáhlejších formulářů můžeme vstupní pole, která spolu souvisí, sdružit dohromady pomocí elementu `FIELDSET`, do kterého je uzavřeme. Ihned za tagem `<FIELDSET>` můžeme použít element `LEGEND`, kterým můžeme určit název celé skupiny vstupních polí. Toto rozčlenění formuláře se uplatní zejména při převodu do mluvené řeči.

Atribut `ACCESSKEY` lze kromě `LABEL` použít i u elementů `A`, `CAPTION` a `LEGEND`. Může tak zpříjemnit práci s formuláři a s odkazy, které je potřeba často vyplňovat a aktivovat.

Podobný cíl má i atribut `TABINDEX`, který lze použít u elementů `A`, `AREA`, `OBJECT`, `INPUT`, `SELECT`, `TEXTAREA` a `BUTTON`. Hodnotou tohoto atributu je celočíselná hodnota. Pokud se pak po jednotlivých výše uvedených elementech pohybujeme pomocí tabulátoru (Tab), aktivují se v pořadí určeném pomocí `TABINDEX`. Pokud `TABINDEX` nepoužijeme, aktivují se postupně tak, jak byly zařazeny do stránky.

13.7 Vkládání matematických vzorců

Původně Web vznikl pro potřeby vědců, a to zejména fyziků. Je tedy celkem s podivem, že poměrně dlouho jazyk HTML neobsahoval žádné prostředky pro tvorbu složitějších matematických formulí. Jistě nebude v HTML problém zapsat rovnici $3 \times 5 = 15$:

```
3 &times; 5 = 15
```

Problémy však nastanou se složitějšími matematickými výrazy. Tak například následující vzorec na první pohled dalece převyšuje možnosti, které nám HTML nabízí:

$$\lim_{n\to\infty}\left(1+\frac{1}{n}\right)^n = e$$

Pro sazbu dokumentů, u nichž byl požadován pouze tištěný výstup a které obsahovaly hodně matematický výrazů, se již téměř dvacet let používá výhradně typografický systém TEX. K vysázení výše uvedeného vztahu stačí v TEXu následující jednoduchý zápis:

```
$$\lim_{n\to\infty}\left(1+{1\over n}\right)^n=e$$
```

První podporu pro zařazení matických výrazů na stránku přineslo až HTML 3.0 v roce 1995. Jelikož TEX je pro sazbu matematiky de facto standardem, zahrnovalo i HTML 3.0 elementy odvozené od makropříkazů TEXu a k vykouzlení našeho vztahu na HTML stránce stačilo použít následující zápis:

```
<MATH>&lim;_n&rarr;&inf;_(<LEFT>1+{1<OVER>n}<RIGHT>)^n^=e
</MATH>
```

Vidíme, že logika zápisu je zcela stejná jako v TEXu. Bohužel HTML 3.0 se nikdy nestalo standardem, protože ho nepodporoval žádný prohlížeč. Proto nám tento způsob zápisu matematických vzorců vůbec nepomůže. Podívejme se tedy na způsoby zápisu matematických vzorců, které můžeme použít dnes.

Vzorec jako obrázek

Pokud naše stránka bude obsahovat pouze pár složitějších vzorců, není použití obrázků až tak špatný nápad. Vzorec vysázíme v lepším případě v TEXu, v horším v editoru rovnic od Microsoftu. Z takto vysázeného vzorce vyrobíme obrázek ve formátu GIF. Je vhodné obrázek udělat transparentní a snížit v něm počet barev na dvě. Vzorec pak do stránky zařadíme pomocí tagu `<IMG>`. Neměli bychom zapomenout v atributu `ALT` uvést zápis vzorce například pomocí TEXové syntaxe pro uživatele, kteří nemají grafický prohlížeč.

Kouzla a podvody s tabulkami

Pokud máme v dokumentu hodně vzorců, je řešení s vloženými obrázky poměrně náročné na přenosovou kapacitu. V těchto případech se můžeme pokusit pomocí šikovně vytvořené tabulky poskládat symboly, které máme k dispozici, do výsledného vzorce. Pokud nám nějaký symbol chybí, můžeme ho nahradit malým obrázkem. Naši oblíbenou rovnici můžeme vytvořit pomocí následující tabulky:

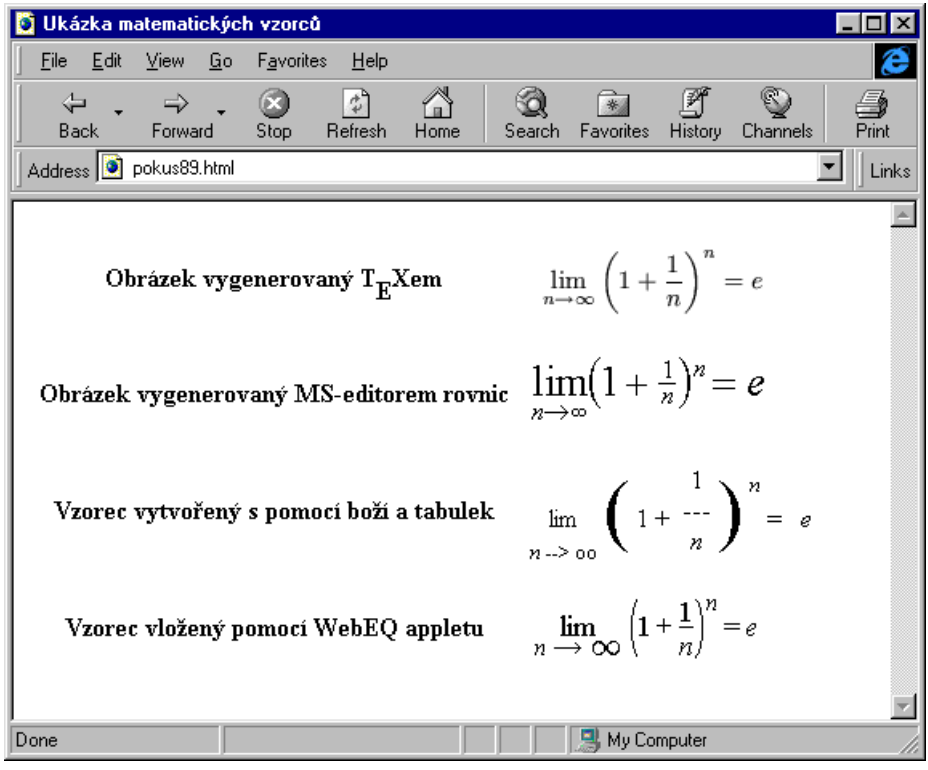
```
<TABLE>
<TR><TD>
    <TD ROWSPAN=3><FONT SIZE="+4">(</FONT>
    <TD>
    <TD VALIGN=MIDDLE ALIGN=CENTER ROWSPAN=3>
        1<BR>---<BR><I>n</I>
    <TD ROWSPAN=3><FONT SIZE="+4">)</FONT>
    <TD><SUB><I>n</I></SUB>
<TR><TD ALIGN=CENTER>lim
    <TD>1 + 
    <TD>
    <TD>=   <I>e</I>
<TR><TD><SMALL><I>n</I> --&gt; oo</SMALL>
</TABLE>
```

Výsledek je spolu s předchozími dvěma možnosti zachycen na obrázku 13-6 na následující straně. Proč pomocí této na první pohled zmatené tabulky dostaneme „celkem slušný“ výsledek, nechám laskavému čtenáři na rozmyšlení.

Pokud bychom chtěli mít ve vzorci lepší zlomkové čáry, můžeme použít zajímavý trik. Vyrobíme si obrázek, který se bude skládat z jednoho černého bodu. Pokud jej pak do stránky zařadíme pomocí

```
<IMG SRC=blackdot.gif WIDTH=50 HEIGHT=1>
```

dostaneme krásnou zlomkovou čáru o šířce 50 bodů.



Obr. 13-6: Různé způsoby zařazení matematického vzorce na stránku

WebEQ

WebEQ je sada Java-apletů, které umožňují do stránky jednoduše vkládat vzorce zapsané v podobné syntaxi, jakou má TEX. Navíc lze uvnitř vzorců používat odkazy a barvy. Autorem WebEQ je Robert R. Miner a systém je možno získat na adrese `http://www.geom.umn.edu/`.

Podívejme se však, jak se WebEQ používá. Pro vložení naší rovnice použijeme následující zápis:

```
<APPLET CODE="geom.webeq.app.mdraw" HEIGHT=50 WIDTH=150>
<PARAM NAME=eq VALUE="\displaystyle{\lim_{n\to\infty}
      \left(1+\frac{1}{n}\right)^n=e}">
</APPLET>
```

Pomocí elementu `APPLET` vložíme do stránky aplet, který umí zobrazovat matematické výrazy. Pokud máme aplet nainstalován v jiném adresáři než je stránka, musíme ještě pomocí `CODEBASE` určit URL, kde sídlí potřebné aplety z WebEQ. Pomocí parametru `eq` pak apletu předáme vzorec k vykreslení.

První natažení apletu trvá poměrně dlouho, pro další vzorce je však již aplet uložen ve vyrovnávací paměti a vykreslení je tak velmi rychlé. WebEQ tedy můžeme použít i pro stránky, které obsahují vetší množství matematických vzorců.

WebEQ je doplněn i řadou pomocných utilit, které například automaticky na stránce upraví velikost všech apletů tak, aby vyhovovala velikosti vkládaných vzorců.

Podobné a ještě větší možnosti nabízí TechExplorer. Jedná se plug-in modul od firmy IBM. Ten rovněž umožňuje na stránku zařadit vzorce zapsané v syntaxi odvozené od TEXu. Jelikož se však jedná o modul plug-in, je potřeba, aby si jej každý uživatel, který si prohlíží naši stránku, nejprve nainstaloval.[3]

MathML

MathML (Mathematical Markup Language) [10] je značkovací jazyk určený speciálně pro zápis matematických výrazů. Oproti předchozím způsobům zápisu matematických vzorců přináší jednu novou vlastnost. Předchozí notace vycházely z TEXu a vzorce byly zapisovány tak, aby mohly být jednoduše zobrazeny. V HTML 3.0 například výraz $(x+y)^2$ zapíšeme jako `(x+y)^2^`. Výraz se vysází správně, ale jeho skutečný význam notace nepostihuje. Exponent 2 se váže pouze k uzavírací závorce. Zápis tedy nijak nezachycuje to, že mocnina se ve skutečnosti vztahuje k celému obsahu závorky. Výraz tedy nemůžeme snadno překopírovat do nějakého programu, který by nám např. vykreslil jeho graf.

MathML jako novinku přináší možnost vzorce zapisovat tak, aby byl postihnut jejich přesný význam a mohly být dále zpracovány nějakým matematickým programem. Kromě toho MathML obsahuje i elementy, které slouží k čistě prezentačnímu zápisu vzorce.

Zápis naší rovnice si nejprve ukážeme v MathML v čistě prezentační formě, kdy není postižen přesný význam celého výrazu:

```
<MROW>
  <MROW>
    <MSUB>
      <MO>&lim;</MO>
      <MROW>
        <MI>n</MI>
        <MO>&RightArrow;</MO>
        <MI>&inf;</MI>
      </MROW>
    </MSUB>
    <MO>&ApplyFunction;</MO>
```


[3] S instalací je však problém, protože i přes opakované pokusy se mi nepodařilo plug-in ze serveru IBM stáhnout.

```
      <MSUP>
        <MROW>
          <MF>(</MF>
          <MROW>
            <MN>1</MN>
            <MO>+</MO>
            <MFRAC>
              <MN>1<MN>
              <MI>n</MI>
            </MFRAC>
          </MROW>
          <MF>)</MF>
        </MROW>
        <MI>n</MI>
      </MSUP>
    </MROW>
    <MO>=</MO>
    <MI>e</MI>
  </MROW>
```

Zápis je to poměrně zdlouhavý a autoři proto předpokládají, že vzniknou programy, které budou převádět vzorce například ze stručné TEXové notace do výřečnějšího MathML. V MathML se v prezentačním způsobu zápisu vyjadřuje význam jednotlivých atomů vzorce. Element `MN` vyznačuje číslo, `MI` vyznačuje identifikátor nebo proměnnou, `MO` pak slouží k označení operátorů. `MROW` sdružuje podle potřeby jednotlivé podvýrazy celé rovnice. `MF` slouží k zápisu závorek — ty se mimo jiné automaticky zvětšují podle potřeby, aby byly vysoké jako mezi nimi uzavřený výraz. `MSUP` a `MSUB` slouží k zápisu horního a dolního indexu. Tyto elementy obsahují vždy dva další elementy — první z nich je argument, ke kterému se index vztahuje, a druhý je samotný index. Zcela obdobně dva elementy obsahuje i `MFRAC`. První element je čitatel a druhý jmenovatel zlomku.

I když je předchozí zápis mnohem lépe strukturovaný než dříve uvedené jazyky pro zápis matematických vzorců, přesný význam výrazu zachycují v MathML elementy pro vyznačování obsahu:

```
<EXPR>
  <LIMIT>
    <LOWLIMIT> n <TENDSTO/> &inf; </LOWLIMIT>
    <EXPR>
     <EXPR>
       1 <PLUS/> <EXPR> 1 <OVER/> n </EXPR>
     </EXPR>
     <POWER/>
```

```
      n
     </EXPR>
   </LIMIT>
   <EQ/>
   e
</EXPR>
```

Tento zápis lze již přímo převést do sémantického stromu, který reprezentuje význam celého výrazu. Element `EXPR` slouží k označení jednotlivých podvýrazů. `LIMIT` slouží pro zápis limity a obsahuje dva další elementy. `LOWLIMIT` obsahuje výraz, který obsahuje podmínky limity. Následující element `EXPR` obsahuje výraz, z něhož se limita počítá. Element `POWER` slouží k vyznačení mocniny, `EQ` k vyznačení rovnosti.

Poněkud podivný nám může připadat zápis nepárových elementů, které mají na konci svého jména lomítko (`<PLUS/>`, `<POWER/>`, `<TENDSTO/>` apod.). Tato syntaktická jemnůstka je příčinou toho, že elementy použitelné v MathML jsou definovány pomocí jazyka XML (o něm si více povíme v poslední kapitole na straně 268). Tento jazyk vyžaduje, aby byly nepárové elementy v dokumentu jednoznačně odlišeny od těch párových. K tomuto účelu se využívá již zmíněné lomítko.

MathML umožňuje míchat prezentační i obsahové vyznačování dohromady. Poslední zápis tedy můžeme doplnit i o prezentační elementy pro zobrazení závorek, abychom měli jistotu, že se závorky zobrazí:

```
<EXPR>
  <LIMIT>
    <LOWLIMIT> n <TENDSTO/> &inf; </LOWLIMIT>
    <EXPR>
     <EXPR>
     <MF>(</MF>
       <EXPR>
         1 <PLUS/> <EXPR> 1 <OVER/> n </EXPR>
       </EXPR>
     <MF>)</MF>
     </EXPR>
     <POWER/>
     n
    </EXPR>
  </LIMIT>
  <EQ/>
  e
</EXPR>
```

MathML umožňuje jeden výraz zapsat i oběma způsoby zároveň. K tomu slouží element SEMANTIC. Měl by obsahovat dva elementy. První z nich se použije při zobrazování výrazů a druhý při jeho interpretaci.

K vložení výrazů zapsaných pomocí MathML do stránky budou sloužit dva nové elementy MATH a MATHDISP. První z nich vloží výraz do běžného textu odstavce.[4] MATHDISP výraz zobrazí na samostatném řádku.[5]

MathML je nový (pochází z května 1997) a komplexní značkovací jazyk. O tom, jak se ujme v praxi, rozhodne teprve budoucnost. Již nyní je však MathML podporován prohlížečem *Amaya*, který je k dispozici pro Unix, a od konce září by měla být uvolněna verze pro *Windows 95/NT*. Autoři systému WebEQ a TechExlorer plánují přidat do dalších verzí svých produktů podporu pro MathML. O podpoře v *Navigatoru* a *Exploreru* bych si žádné přílišné iluze nevytvářel. Microsoft a Netscape bojují o běžného uživatele, který (údajně) požaduje barvami hýřící, pohybující se a zvuky vyluzující stránky. Jejich úsilí je tedy upřeno směrem multimediálním a nikoli matematickým.

Skutečné rozšíření MathML však nezáleží pouze na prohlížečích. Z ukázek jsme viděli, že zápis výrazů v MathML je poměrně zdlouhavý. Pokud tedy nevzniknou editory, které jednoduchým způsobem umožní zadat vzorec a sami vytvoří jeho zápis v MathML, bude MathML používat asi málokdo.

13.8 Vkládání chemických vzorců

Pro vkládání vzorců chemických reakcí většinou vystačíme se stejnými prostředky jako pro vkládání matematických vzorců. Tak například rovnici fotosyntézy můžeme zapsat pomocí následujícího poměrně jednoduchého kódu:

```
<TABLE>
<TR>
<TD>6 CO<SUB>2</SUB> + 6 H<SUB>2</SUB>O
<TD ALIGN=CENTER><SMALL>světlo</SMALL><BR>
    <IMG SRC=blackdot.gif WIDTH=40 HEIGHT=1 ALIGN=MIDDLE>&gt;
    <BR> 
<TD>C<SUB>6</SUB>H<SUB>12</SUB>O<SUB>6</SUB> + 6 O<SUB>2</SUB>
</TABLE>
```

$$6\,CO_2 + 6\,H_2O \xrightarrow{\text{světlo}} C_6H_{12}O_6 + 6\,O_2$$

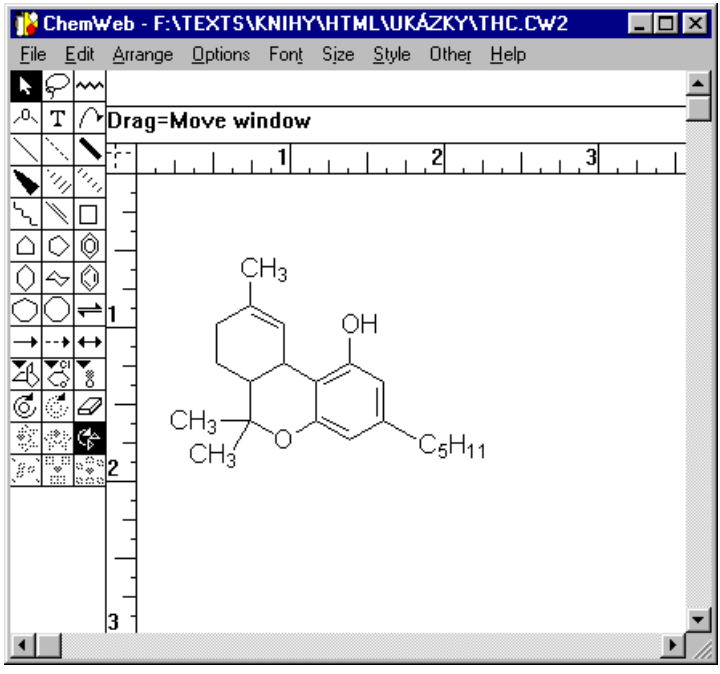
Složitější situace nastane, pokud chceme do stránky vložit strukturní vzorec. Ty jsou většinou tak složité, že nám nezbyde nic jiného než je vložit jako obrázek. K vytvoření obrázku můžeme s výhodou použít program *ChemWeb* od firmy

[4] MATH se tedy chová jako inline element.

[5] MATHDISP je tedy blokový element.

SoftShell. Jedná se o zjednodušenou verzi programu *ChemWindow* od téže firmy. Zjednodušení spočívá v tom, že *ChemWeb* umí vzorec uložit pouze jako obrázek ve formátu GIF. Neumí tedy vytvořit například obrázek ve formátu WMF, který lze použít pro tisk s vyšším rozlišením. To nám však vůbec nevadí, GIF je přesně to, co na webovských stránkách potřebujeme. Navíc je *ChemWeb* k dispozici zdarma a lze jej získat na adrese `http://www.softshell.com/`.

K vytvoření vzorce použijeme poměrně pohodlné grafické prostředí programu (viz obr. 13-7).

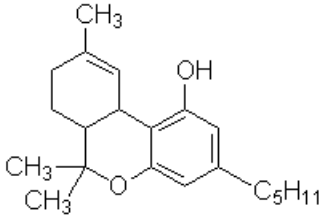


Obr. 13-7: Prostředí programu ChemWeb

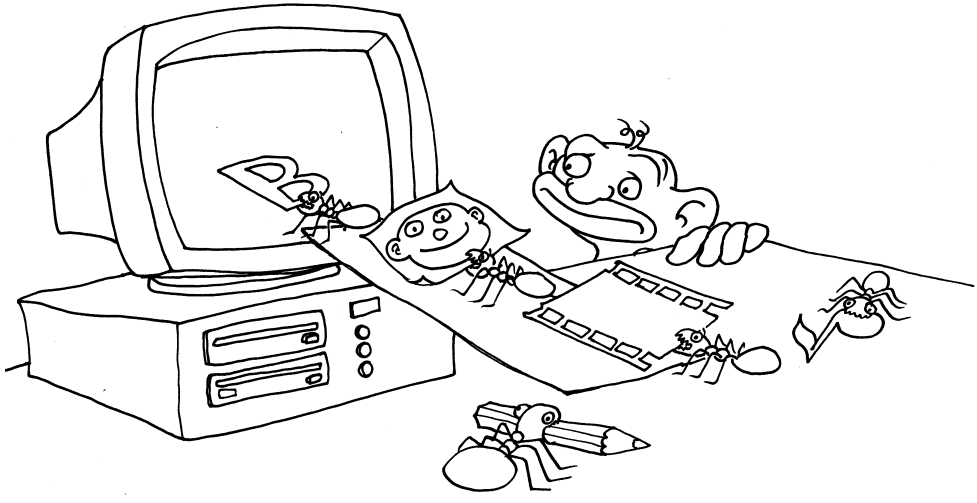
Vytvořený vzorec pomocí File ▷ Save as... uložíme do souboru ve formátu GIF. Před uložením tedy v poli Save file as type vybereme volbu Compuserve GIF89.

Do stránky pak vzorec vložíme jako běžný obrázek. Pokud chceme vzorec doplnit i popisem, můžeme obrázek vložit do tabulky s jednou buňkou a popis vytvořit pomocí `CAPTION`:


```
<TABLE>
<CAPTION ALIGN=BOTTOM>Tetrahydrocannabinol</CAPTION>
<TR><TH><IMG SRC=thc.gif>
</TABLE>
```



Tetrahydrocannabinol



14. Dynamické HTML

Na vývoji HTML je vidět, že neustále vzrůstaly požadavky na podporu multimédií v HTML. Nejprve byly přidány obrázky, pak možnosti vkládání Java-apletů. Zhruba ve stejné době začalo být možné oživení stránek díky JavaScriptu. Dalším krokem, který umožní vytvářet ještě interaktivnější stránky, je *dynamické HTML*. Dynamické HTML poprvé představila firma Microsoft ve svém prohlížeči *Internet Explorer 4.0*. Velkou výhodou dynamického HTML je, že do samotného jazyka HTML nepřidává nic nového. Pouze rozvíjí možnosti, které je možno dosáhnout pomocí stylů a skriptů. V této kapitole si vysvětlíme myšlenku, na které je dynamické HTML vystavěno. Nevynecháme ani praktické ukázky těch částí dynamického HTML, které mají podle mého názoru šanci na to stát se standardem. Naopak se zdaleka vyhneme těm zákoutím dynamického HTML, které jsou postaveny na proprietárních řešeních Microsoftu jako je např. ActiveX. I tak však bude tato kapitola velice zajímavá.

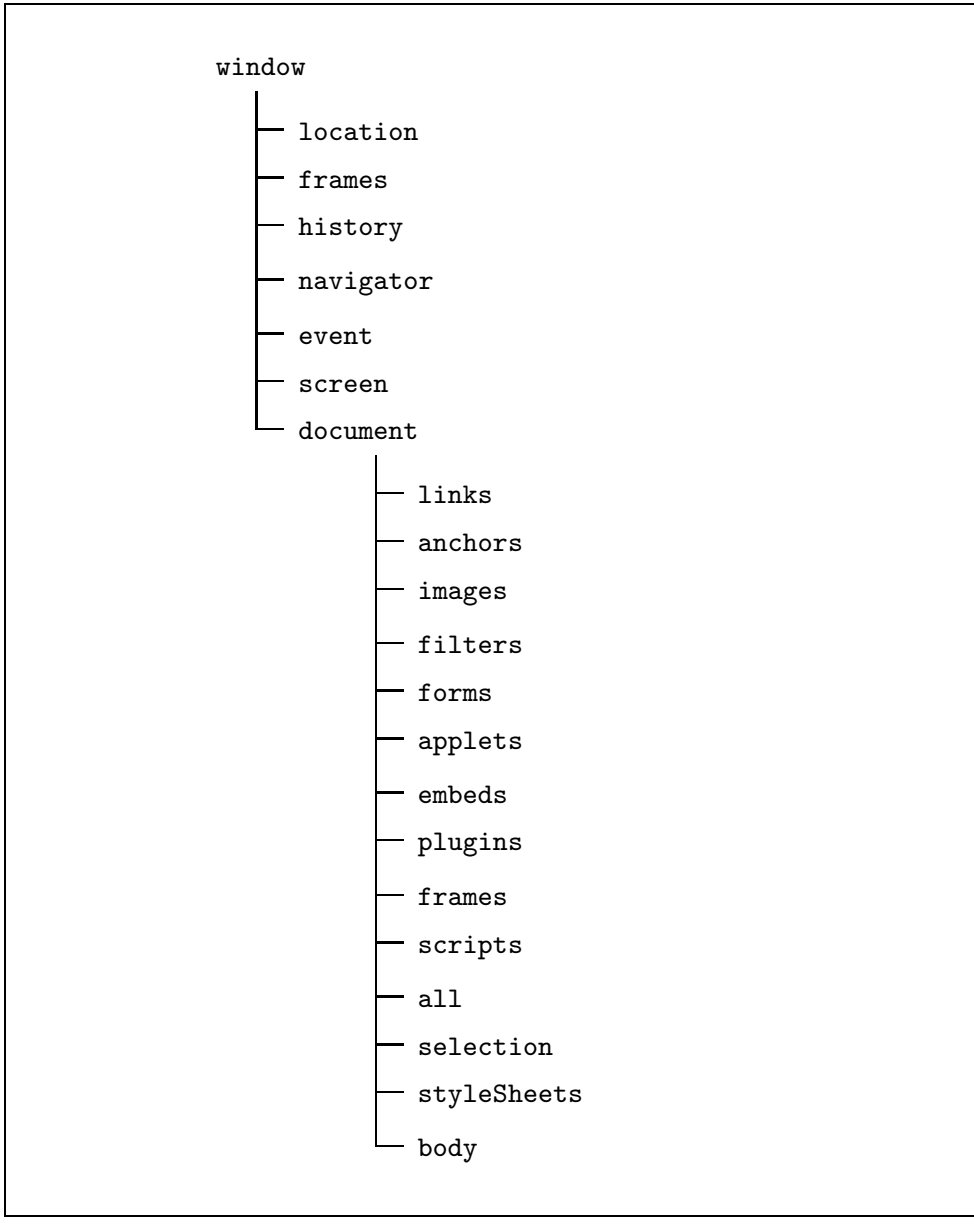
14.1 Objektový model dokumentu

Stránky, které jsme dosud vytvářeli, byly ve své podstatě statické. Sice je mohl automaticky vygenerovat server nebo před jejich zobrazením byly upraveny skriptem, ale během prohlížení v prohlížeči se neměnily. Dynamické HTML umožňuje během prohlížení stránky měnit obsah jednotlivých elementů a atributů, přidávat a ubírat elementy. Kromě toho lze během prohlížení měnit i styly, což se na výsledném zobrazení dokumentu může výrazně promítnout. Aby byly možnosti už „téměř dokonalé“, byla definice stylů rozšířena o možnost libovolně měnit viditelnost a umístění každého elementu v 2,5D prostoru. O tomto rozšíření si samozřejmě také povíme.

Aby mohly být pomocí skriptů prováděny všechny výše uvedené činnosti, bylo zapotřebí definovat jednotný způsob pro přístup k jednotlivým elementům, atributům dokumentu a vlastnostem stylu. Teoreticky tyto požadavky definuje konsorcium W3C v *DOM (Document Object Model) — objektovém modelu dokumentu*.

Explorer 4.0 jde však dál a tyto požadavky i prakticky implementuje. Staví přitom na objektovém modelu Netscapu použitém v JavaScriptu, který dále rozšiřuje.

Základním objektem je `window`, přes který jsou dostupné všechny další objekty. Nejdůležitějším objektem je `document`, který v sobě zahrnuje všechny objekty a vlastnosti vztahující se k aktuálnímu dokumentu. Na jednotlivé objekty v hierarchii (obrázek 14-1 na následující straně) se odvoláváme pomocí tečkové notace: `window.screen`, `window.document`, `window.document.body`. Protože je



Obr. 14-1: Základní hierarchie objektů

objekt `document` nejpoužívanější, můžeme před ním vynechat specifikaci nadřazeného objektu `window`. Zkráceně tedy můžeme psát `window.screen`, `document`, `document.body`.

Na dalších nižších úrovních v objektové hierarchii mohou být buď další objekty a nebo přímo vlastnosti a metody jednotlivých objektů. Vše si ukážeme na příkladě. Objekt `document.all` obsahuje všechny elementy obsažené v dokumentu. Přístupné jsou pomocí svého jména definovaného atributem `ID`:

```
<H1 ID=Nadpis>Pokusný nadpis</H1>
<P ID=Odstavec1>Text odstavce teď není důležitý
```

Nyní můžeme ve skriptech na stránce používat objekty `document.all.Nadpis` a `document.all.Odstavec1`, které zastupují element pro nadpis a odstavec. U těchto objektů máme k dispozici vlastnosti, které odpovídají atributům použitelným u odpovídajících elementů. Pomocí vlastnosti `document.all.Nadpis.align` můžeme měnit způsob zarovnání nadpisu. Následující stránka obsahuje skript, který po kliknutí myší na nadpis změní jeho zarovnání zleva doprostřed:

```
<HTML>
<HEAD>
<TITLE>Ukázka dynamického HTML 1.</TITLE>
<SCRIPT LANGUAGE="JavaScript"><!--
function ChangeAlign()
{
  document.all.Nadpis.align = "center";
}
--></SCRIPT>
</HEAD>
<BODY>
<H1 ID=Nadpis ALIGN=LEFT
    onClick="ChangeAlign()">Pokusný nadpis</H1>
<P ID=Odstavec1>Text odstavce teď není důležitý
</BODY>
</HTML>
```

Z principu práce s dynamickým HTML vidíme, že stránka bez problémů bude pracovat i s prohlížeči, které dynamické HTML nepodporují. Jen po kliknutí myší na nadpis se nic nestane, protože tyto prohlížeče neumějí dynamicky měnit obsah stránky po jejím zobrazení. To však nijak dramaticky nesníží informační hodnotu stránky.

U objektů, které odpovídají elementům stránky, máme k dispozici vlastnost i objekt `style`. Pomocí vlastnosti `style` můžeme přistupovat k definici stylu elementu stejně jako pomocí atributu `STYLE`. Následující kód by změnil barvu a velikost nadpisu:

```
document.all.Nadpis.style = "color: red; font-size: 30px";
```

Oproti tomu vlastnosti objektu `style` odpovídají jednotlivým vlastnostem stylů. Poslední efekt tedy můžeme dosáhnout i následujícím kódem:

```
document.all.Nadpis.style.color = "red";
document.all.Nadpis.style.fontSize = "30px"
```

Zajímavá je vlastnost `innerText`. Ta obsahuje text uvnitř elementu. Zápisem do této vlastnosti můžeme text změnit. Zkuste následující legrácku:

```
<HTML>
<HEAD>
<TITLE>Ukázka dynamického HTML 2.</TITLE>
<SCRIPT LANGUAGE="JavaScript"><!--
function ChangeText()
{
  document.all.Nadpis.innerText = "Poslouchás mě pěkně!!!";
  document.all.Nadpis.style.color = "blue";
  document.all.Nadpis.style.fontSize = "50px";
}
--></SCRIPT>
</HEAD>
<BODY>
<H1 ID=Nadpis onClick="ChangeText()">
Klikni na tenhle nadpis a uvidíš, co se stane.</H1>
</BODY>
</HTML>
```

Dokonce lze měnit celý element, ne jen jeho obsah. Použijeme vlastnost `outerHTML` a předchozí ukázku upravíme tak, aby se místo nadpisu po kliknutí objevil obrázek:

```
<HTML>
<HEAD>
<TITLE>Ukázka dynamického HTML 3.</TITLE>
<SCRIPT LANGUAGE="JavaScript"><!--
function ChangeText()
{
  document.all.Nadpis.outerHTML = "<IMG SRC=astro.gif>"
        + "<P>Kromě obrázku přidám i krátký odstavec textu.";
}
--></SCRIPT>
</HEAD>
<BODY>
<H1 ID=Nadpis onClick="ChangeText()">
```

```
Klikni na tenhle nadpis a uvidíš, co se stane.</H1>
</BODY>
</HTML>
```

14.2 Rozšíření stylů o řízení pozice elementů

Po krátkém úvodu do dynamického HTML si popíšeme nová rozšíření stylů dokumentů. Toto rozšíření je dnes pracovní verzí konsorcia W3C [21] a podle mne se brzy stane doporučením (a pro nás tedy i standardem).

Rozšíření přináší nové vlastnosti, které umožňují přesně řídit vzájemné umístění jednotlivých elementů na stránce. Mezi nejdůležitější nové vlastnosti patří `left` a `top`, které určují posunutí elementu. Jejich konkrétní interpretace však závisí na hodnotě vlastnosti `position`. Ta může nabývat dvou hodnot — `absolute` a `relative`.

Absolutní určení pozice

Pokud použijeme absolutní určení pozice elementu, je tento element zobrazen zcela nezávisle na svém původním umístění. Jeho levý horní roh se posune do místa zadaného pomocí `left` a `top`. Souřadnice se vztahují k souřadnému systému nadřazeného elementu a tím je v tomto případě nejčastěji celý dokument. Jeho levý horní roh má souřadnice $(0, 0)$.

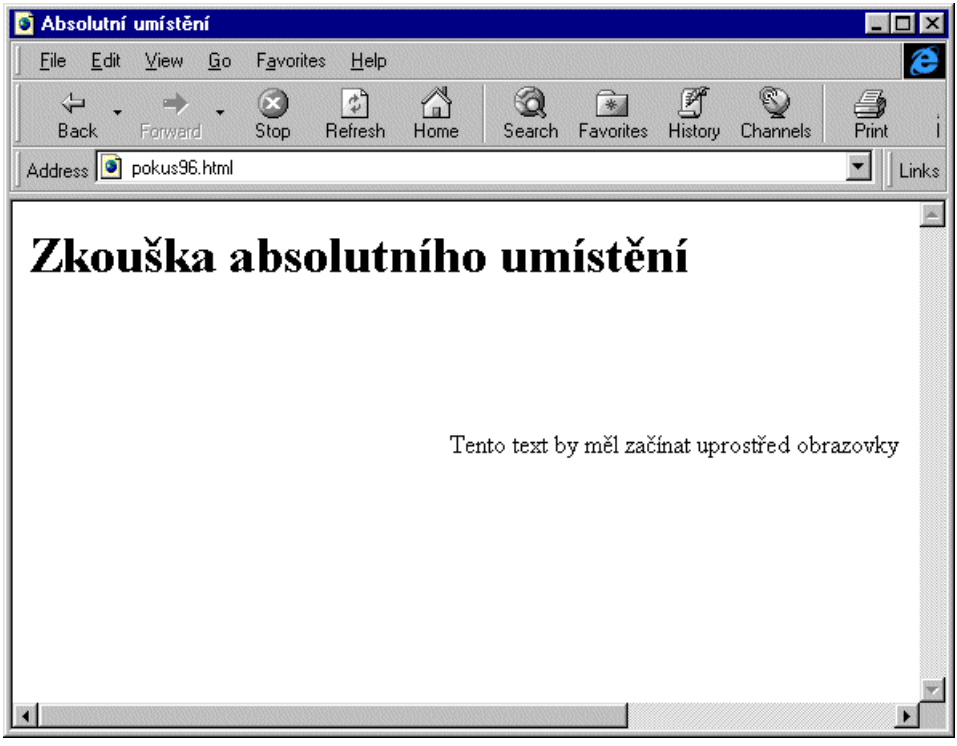
Jako hodnotu `left` a `top` můžeme použít buď pixely (`px`) nebo procento, které se vztahuje k šířce a výšce rodičovského elementu:

```
<BODY>
<H1>Zkouška absolutního umístění</H1>
<SPAN STYLE="position: absolute; left: 50%; top: 50%">
Tento text by měl začínat uprostřed obrazovky</SPAN>
</BODY>
```

Výsledek našeho prvního pokusu o změnu umístění elementu na stránce si můžeme prohlédnout na obrázku 14-2 na následující straně.

Šířku, do které se bude formátovat obsah elementu, můžeme určit pomocí vlastnosti `width`. Výška se pak upraví podle potřeby. I tu však lze nastavit pomocí vlastnosti `height`.

Absolutně umístěné elementy se mohou překrývat. O tom, který element bude vidět, rozhoduje vlastnost `z-index`. Element, který ji má nastavenu na větší hodnotu, překryje ty ostatní. To si demonstrujeme na následující malé ukázce:



Obr. 14-2: Stránka s absolutně umístěným textem

```
<HTML>
<HEAD>
<TITLE>Absolutní umístění a z-index</TITLE>
<STYLE TYPE="text/css"><!--
  .nadpis { font-size: 50px;
            font-style: bold }
--></STYLE>
</HEAD>
<BODY>
<DIV CLASS=nadpis>
<SPAN STYLE="position: absolute; left: 1; top: 1;
 color: #001; z-index: 1">Efektní nápis</SPAN>
<SPAN STYLE="position: absolute; left: 2; top: 2;
 color: #002; z-index: 2">Efektní nápis</SPAN>
<SPAN STYLE="position: absolute; left: 3; top: 3;
 color: #003; z-index: 3">Efektní nápis</SPAN>
<SPAN STYLE="position: absolute; left: 4; top: 4;
```

```
 color: #004; z-index: 4">Efektní nápis</SPAN>
<SPAN STYLE="position: absolute; left: 5; top: 5;
 color: #005; z-index: 5">Efektní nápis</SPAN>
<SPAN STYLE="position: absolute; left: 6; top: 6;
 color: #006; z-index: 6">Efektní nápis</SPAN>
<SPAN STYLE="position: absolute; left: 7; top: 7;
 color: #007; z-index: 7">Efektní nápis</SPAN>
    :
    .
<SPAN STYLE="position: absolute; left: 13; top: 13;
 color: #00d; z-index: 13">Efektní nápis</SPAN>
<SPAN STYLE="position: absolute; left: 14; top: 14;
 color: #00e; z-index: 14">Efektní nápis</SPAN>
<SPAN STYLE="position: absolute; left: 15; top: 15;
 color: #00f; z-index: 15">Efektní nápis</SPAN>
</DIV>
</BODY>
</HTML>
```

Relativní určení pozice

Pokud použijeme relativní určení pozice, element se pouze posune ze své přirozené pozice o délku určenou pomocí `left` a `top`. Jeho formátování přitom zůstane zachováno. Text v nadřazeném elementu zůstane zformátován tak, jako by posunutý element nebyl vůbec posunut. Malá ukázka:

```
Nějaký text
<SPAN STYLE="position: relative; left: 10px; top: 10px">
Relativní text</SPAN>
a zase původní text.
```

Nějaký text Relativní text a zase původní text.

Element se stylem `position: relative` opět definuje nový souřadný systém s počátkem ve svém levém horním rohu. To lze využit například pro vytvoření stínu:

```
<DIV STYLE="font: bold 30px Arial">Ukázka
<SPAN STYLE="position: relative">stínovaného textu
```

Vlastnost	Možné hodnoty	Implicitní hodnota	Aplikuje se na	Dědí se?	Interpretace procentních hodnot	Popis
position	`absolute` | `relative` | `static`	`static`	všechny elementy	ano	–	způsob umístění elementu; `static` odpovídá běžnému formátování
left	*«délka»* | *«procento»* | `auto`	`auto`	všechny elementy	ne	–	posunutí elementu vpravo; záporná hodnota posune vlevo
top	*«délka»* | *«procento»* | `auto`	`auto`	všechny elementy	ne	–	posunutí elementu dolů; záporná hodnota posune nahoru
width **height**	*«délka»* | *«procento»* | `auto`	`auto`	elementy blokové, nahrazované a s `position: absolute`	ne	vztahují se k šířce (výšce) rodičovského elementu	šířka a výška elementu
clip	`auto` | `rect([`*«délka»* | `auto] [`*«délka»* | `auto] [`*«délka»* | `auto] [`*«délka»* | `auto])`	`auto`	elementy s `position: absolute`	ne	–	definice obdélníkové části elementu, která bude viditelná (standardně je viditelný celý element)
overflow	`none` | `clip` | `scroll`	`none`	elementy s relativní nebo absolutní pozicí	ne	–	způsob zobrazení elementů, jejichž obsah se nevejde do vyhrazeného prostoru: `none` = obsah elementu přeteče, `clip` = obsah elementu bude oříznut, `scroll` = po obsahu půjde rolovat
z-index	`auto` | *«číslo»*	`auto`	elementy s relativní nebo absolutní pozicí	ne	–	pozice elementu na pseudoose *z*
visibility	`inherit` | `visible` | `hidden`	`inherit`	všechny elementy	pro `inherit`	–	viditelnost elementu: `visible` = viditelný, `hidden` = neviditelný;

Tab. 14-1: Vlastnosti pro určení pozice

```
<SPAN STYLE="position: absolute; left: 2px; top: 2px;
            color: green">stínovaného textu
</SPAN></SPAN>uvnitř jiného textu.</DIV>
```

Další důležitou vlastností je `visibility`. Může nabývat dvou hodnot — `visible` a `hidden`. První z nich říká, že element bude vidět. Druhá, že vidět nebude.

Tuto vlastnost musíme odlišit od vlastnosti `display`. Její hodnota `display: none` způsobí, že element se bude zcela ignorovat, jakoby na stránce vůbec nebyl. Oproti tomu `visibility: hidden` vynechá pro element místo, element pouze nebude vidět. To lze využít pro různé efekty, pokud máme několik elementů zobrazených přes sebe.

Přehled všech nových vlastností shrnuje tabulka 14-1 na straně 248.

14.3 Praktické ukázky dynamického HTML

Když jsme se seznámili se základy, na kterých dynamické HTML staví, je pravá chvíle pro několik větších ukázek jeho možností.

Zobrazování částí dokumentu na požadavek uživatele

Tato ukázka je pro předvedení možností dynamického HTML přímo typická. Nahraje se stránka, které obsahuje několik článků, ale zobrazí se pouze jejich nadpisy. Samotný text článků bude skryt nastavením vlastnosti `display` na hodnotu `none`. Celý článek se zobrazí po najetí myší na jeho název. Navíc se nadpis zobrazeného článku barevně zvýrazní. Podobný mechanismus můžeme použít i pro několikaúrovňové seznamy, které umožníme postupně rozbalovat do nižších úrovní. Zobrazení stránky v prohlížeči s podporou dynamického HTML přináší obrázek 14-3 na straně 252.

```
<HTML>
<HEAD>
<TITLE>Vybrané články z dnešního tisku</TITLE>
<SCRIPT LANGUAGE="JavaScript"><!--
function Display(elementID)
{
var id;
```

```
    id = "Clanek" + elementID;
    document.all.item(id).style.display = "";
    id = "Nadpis" + elementID;
    document.all.item(id).style.color = "red";
}

function Undisplay(elementID)
{
var id;

    id = "Clanek" + elementID;
    document.all.item(id).style.display = "none";
    id = "Nadpis" + elementID;
    document.all.item(id).style.color = "black";
}
--></SCRIPT>
</HEAD>
<BODY>

<DIV ALIGN=CENTER>
Myší najeďte nad titulek článku, který si chcete přečíst.
</DIV>

<DIV onMouseOver="Display('1')" onMouseOut="Undisplay('1')">
<H1 ID=Nadpis1>Stodola shořela do základů</H1>
<DIV ID=Clanek1 STYLE="display: none">
<P><EM>MF Dnes -- 2. září 1997</EM>
<P>Krasíkovice (tš) -- Půlmiliónovou škodu způsobil oheň
majiteli zemědělské usedlosti na Pelhřimovsku. V sobotu po
třetí hodině odpoledne sice zasahoval u požáru stodoly se
senem pelhřimovský záchranný hasičský sbor spolu s místními
dobrovolníky, ale živel zničil po chvíli celý objekt. Příčinu
požáru zatím vyšetřovatelé nezjistili.
</DIV>
</DIV>

<DIV onMouseOver="Display('2')" onMouseOut="Undisplay('2')">
<H1 ID=Nadpis2>Shořel kombajn za čtyři miliony korun</H1>
<DIV ID=Clanek2 STYLE="display: none">
<P><EM>MF Dnes -- 2. září 1997</EM>
<P>Uhřinčice (ČTK) -- Škodu dosahující čtyř milionů korun
způsobil požár kombajnu značky Holmer, který propukl z pátku
```

```
na sobotu na poli u obce Uhříněice na Přerovsku. Při rozsáhlém
požáru stroje nebyl nikdo zraněn. Policisté uvedli, že požár
zavinila technická závada na kombajnu.
</DIV>
</DIV>

<DIV onMouseOver="Display('3')" onMouseOut="Undisplay('3')">
<H1 ID=Nadpis3>Mladíci nepřežili náraz do stromu</H1>
<DIV ID=Clanek3 STYLE="display: none">
<P><EM>MF Dnes -- 2. září 1997</EM>
<P>Vysoká u Holice (soc) -- Dva mladíci z Vysokého Mýta zemřeli
při nehodě Škody 105 v noci ze soboty na neděli na silnici mezi
Hradcem Králové a Holicemi. "Při předjíždění vybočili doleva
a narazili do stromu. Zemřeli na místě," řekl operační
důstojník pardubické policie. Policisté zatím nezjistili, zda
řidič před jízdou pil alkohol.
</DIV>
</DIV>

</BODY>
</HTML>
```

Všimněme si, že jednotlivé články jsou obsaženy v elementu `DIV`, aby pro ně šlo nastavit společné reagování na události. Na ukázce je rovněž vidět, jaké úspory přináší používání funkcí a jejich parametrizování.

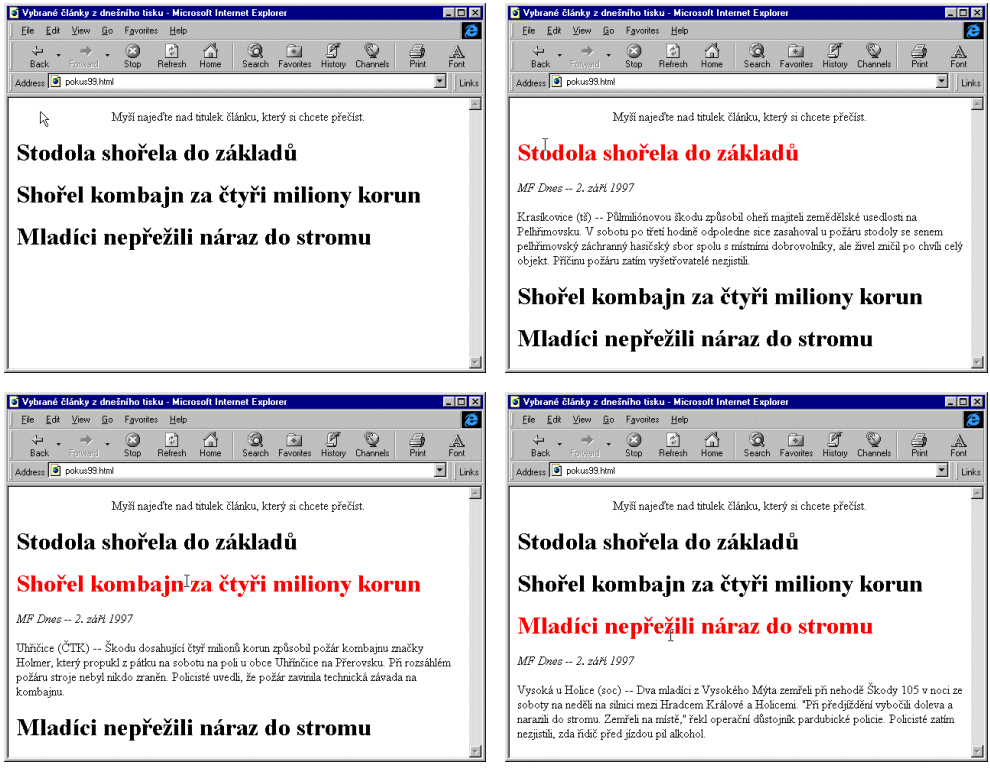
Definovali jsme dvě funkce — `Display()` a `Undisplay()`. Obě jako parametr očekávají číslo článku. `Display()` zobrazí obsah článku a zvýrazní nadpis — ty musí mít přiřazeny správné `ID`. `Undisplay()` se volá po opuštění elementu myší a článek schová a barvu nadpisu nastaví zpět na černou.

Při vytváření dynamických stránek bychom měli mít na paměti, že ne všichni používají k prohlížení naší stránky nejnovější výstřelky browserové technologie. Stránku s dynamickým HTML navrhujeme tak, aby se korektně zobrazila i v prohlížečích bez podpory dynamického HTML. Není to nic těžkého. Naše ukázková stránka tyto vlastnosti splňuje (viz obr. 14-4 na straně 253).

Zvýraznění odkazů po přejetí myší

Pokud chceme upoutat pozornost uživatele na naše odkazy, můžeme použít malý trik. Vytvoříme takovou stránku, kde se barva odkazu změní z modré na červenou vždy, když bude myš nad odkazem.

Abychom nemuseli předělávat celý dokument a u každého odkazu psát obsluhu události, využijeme další zajímavý objekt z objektové hierarchie. Objekt



Obr. 14-3: Jedna stránka má čtyři podoby podle polohy myši

`window.event.srcElement` odpovídá elementu, ke kterému se váže aktuálně zpracovávaná událost.

Události vyvolávající zvýraznění odkazů nastavíme pro celý dokument u elementu `BODY`. V obsluze události si pak zkontrolujeme, zda element, nad kterým je myš, je odkaz (`A`). Pokud ano, odkaz zvýrazníme, případně mu vrátíme původní barvu. V dokumentu pak můžeme mít odkazů kolik chceme, barevně zvýrazňovat se budou všechny.

```
<HTML>
<HEAD>
<TITLE>Stránka s obarvenými odkazy</TITLE>
<STYLE TYPE="text/css"><!--
    :link { color: blue }
--></STYLE>
<SCRIPT LANGUAGE="JavaScript"><!--
function HighlightAnchor()
{
```

Obr. 14-4: Zobrazení stránky s dynamickým HTML v prohlížeči, který ho nepodporuje

```
    if (window.event.srcElement.tagName == "A")
        window.event.srcElement.style.color = "red";
}
```

```
function LowlightAnchor()
{
    if (window.event.srcElement.tagName == "A")
        window.event.srcElement.style.color = "blue";
}
--></SCRIPT>
</HEAD>
<BODY onMouseOver="HighlightAnchor()"
      onMouseOut="LowlightAnchor()">
    «zde může být zcela libovolný dokument»
</BODY>
</HTML>
```

Animace na stránce

Abychom na stránku mohli umístit nějakou animaci — tj. element, který se pohybuje — musíme v pravidelných intervalech měnit jeho pozici. K tomu můžeme s výhodou použít metodu `window.setInterval("«funkce»", «ms»)`. Funkce zadaná prvním parametrem se vyvolá každých *«ms»* milisekund.

My vytvoříme stránku, na které se bude od okrajů obrazovky odrážet obrázek zeměkoule. O pohyb obrázku zeměkoule a jeho odrážení od okrajů se stará funkce `Move()`, která je volaná v pravidelných intervalech. Funkce skriptů by měla být zřejmá z komentářů:

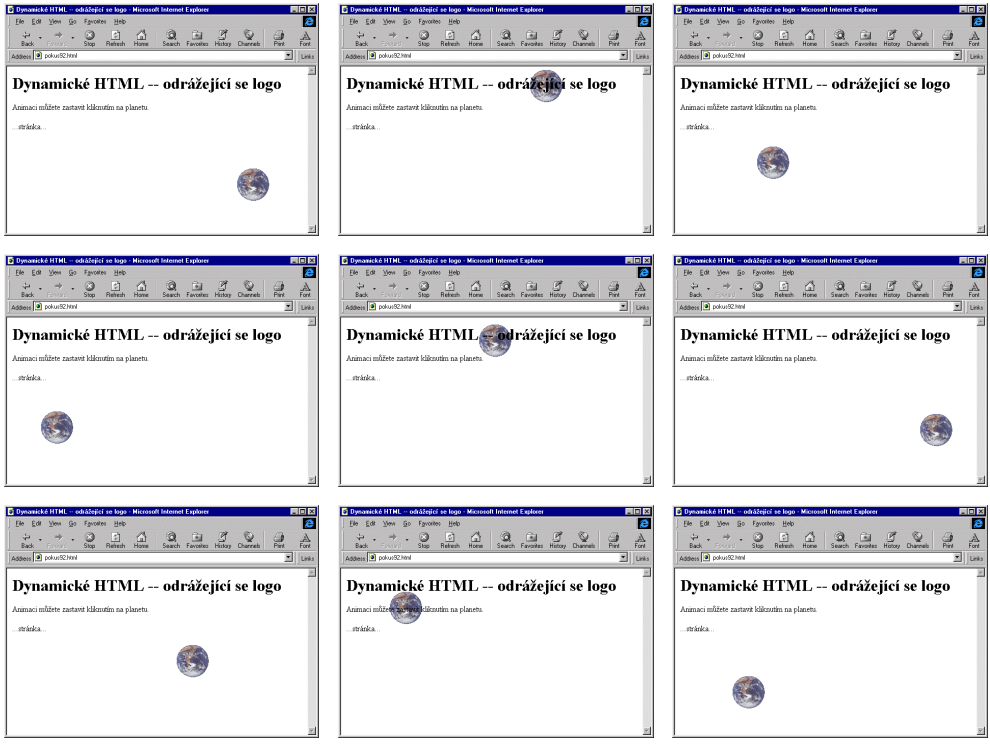
```
<HTML>
<HEAD><TITLE>Dynamické HTML -- odrážející se logo</TITLE>
<SCRIPT LANGUAGE="JavaScript"><!--
var id,                  // pomocná proměnná pro časovač
    stepX, stepY; // krok v X a Y-směru

function Start() // spuštění pohybu
{
    // umístění obrázku doprostřed obrazovky
    document.all.Logo.style.pixelLeft =
        document.body.offsetWidth / 2;
    document.all.Logo.style.pixelTop =
        document.body.offsetHeight / 2;

    // obrázek uděláme viditelný
    document.all.Logo.style.visibility = "visible";

    // náhodná inicializace směru a rychlosti pohybu
```

```
    stepX = (Math.random()+5) * 2 - 5;
    stepY = (Math.random()+5) * 2 - 5;

    // nastavení časovače
    id = window.setInterval("Move()",50);
}

function Stop() // ukončení pohybu
{
    // vypnutí časovače
    window.clearInterval(id);

    // "schování obrázku"
    document.all.Logo.style.visibility = "hidden";
}

function Move() // posun loga
{
    // posunutí obrázku
    document.all.Logo.style.pixelLeft += stepX;
    document.all.Logo.style.pixelTop += stepY;

    // odražení od levého okraje
    if (document.all.Logo.style.pixelLeft <= 0) stepX = -stepX;

    // odražení od pravého okraje
    if ( document.all.Logo.style.pixelLeft >=
         (document.body.offsetWidth - document.all.Logo.width
          - stepX - 22) ) stepX = -stepX;
                // 22 je magické číslo šířky scroll-baru

    // odražení od horního okraje
    if (document.all.Logo.style.pixelTop <= 0) stepY = -stepY;

    // odražení od dolního okraje
    if ( document.all.Logo.style.pixelTop >=
         (document.body.offsetHeight - document.all.Logo.height
          - stepY) ) stepY = -stepY;
}
--></SCRIPT>
<BODY onload="Start()">
<IMG ID="Logo"
```

```
        STYLE="visibility: hidden; position: absolute; z-index:-1"
        SRC="earth.gif" onClick="Stop()">
<H1>Dynamické HTML -- odrážející se logo</H1>
Animaci můžete zastavit kliknutím na planetu.
        «libovolný obsah stránky»
</BODY>
</HTML>
```

U obrázku jsme vlastnost `z-index` nastavili na -1, aby se obrázek pohyboval pod textem stránky.

☺ Stránky, které jsou moc živé, mohou některým uživatelům vadit, protože je ruší při čtení. Měli bychom proto používat krátké animace nebo nabídnout viditelný způsob, jak animace ukončit.



Obr. 14-5: Těžko se to na papír zachycuje, ale Země se opravdu pohybuje

15. Nástroje podporující tvorbu stránek

Zatím jsme se zabývali ruční tvorbou stránek přímo v jazyce HTML. Dnes je k dispozici mnoho programů, které nám mohou vytváření stránek usnadnit. V této kapitole se stručně seznámíme s některými z programů, které nám mohou pomoci.

15.1 HTML editory

My jsme pro vytváření stránek používali obyčejný textový editor. Existuje však několik dalších druhů editorů, které nám mohou práci s HTML dokumenty usnadnit.

Textové editory s podporou HTML

Do této kategorie patří editory, které většinou v menu nabízejí příkazy pro vložení jednotlivých tagů do textu. Text stránky v nich vidíme zapsaný přímo v HTML. Samozřejmostí u těchto programů je barevné odlišení tagů od běžného textu, což usnadňuje orientaci v zápisu stránky.

Oblíbeným představitelem této skupiny programů jsou editory *HotDog* a *HomeSite*. Ty byly vytvořeny speciálně pro potřeby jazyka HTML. Podobnou funkčnost nabízejí klasické editory jako *Emacs* či *JED*, pokud je doplníme patřičnou makronadstavbou, která je rozšíří o potřebné funkce.

Strukturní editory

Tyto editory na první pohled vypadají podobně jako editory z předchozí skupiny. Oproti nim však nabízejí důležitou vlastnost navíc — neumožní nám vytvořit syntakticky nesprávnou stránku. Do místa, které editujeme, můžeme vložit pouze tagy, které se zde mohou objevit. Nestane se nám tedy, že stránka obsahuje zkřížené elementy nebo že chybí ukončovací tagy.

Velice pěkným programem, který patří do této kategorie, je program *asWedit*. K dispozici je i jeho lokalizovaná česká verze.

WYSIWYG editory

Tato kategorie zahrnuje editory s jejichž pomocí můžeme vytvářet stránky bez znalosti HTML. WYSIWYG editor se ovládá podobně jako nějaký textový procesor (např. *Word*). Píšeme text, můžeme měnit použité písmo a zarovnání, vkládat obrázky apod. Výsledek se však ukládá ve formátu HTML.

WYSIWYG editory již během editace zobrazují stránku přibližně tak, jak bude zobrazena v prohlížeči. Problémem těchto programů je, že zaostávají za vývojem HTML a tak mnohdy nepodporují jeho novější rysy jako rámy apod. S podivem je, že některé z těchto editorů generují HTML kód, který není správný a navíc je pro následnou ruční editaci nepřehledně zformátován.

Asi nejznámějším programem této skupiny je *FrontPage Editor* od Microsoftu. Podobnou funkčnost nabízí *Netscape Composer*, což je editor dodávaný s *Netscape Communicatorem*.

Do této skupiny patří i editor *HoTMetaL* od SoftQuadu. Ten však kromě WYSIWYG režimu nabízí i speciální režim, kdy je WYSIWYG podoba doplněna o tagy. Tento produkt nabízí výborné nástroje na kontrolu správné struktury dokumentu a na práci s touto strukturou. To nepřekvapí ty z nás, kteří vědí, že firma SoftQuad se již pěknou řádku let zabývá vývojem produktů podporujících standard SGML.

15.2 Správa celého serveru

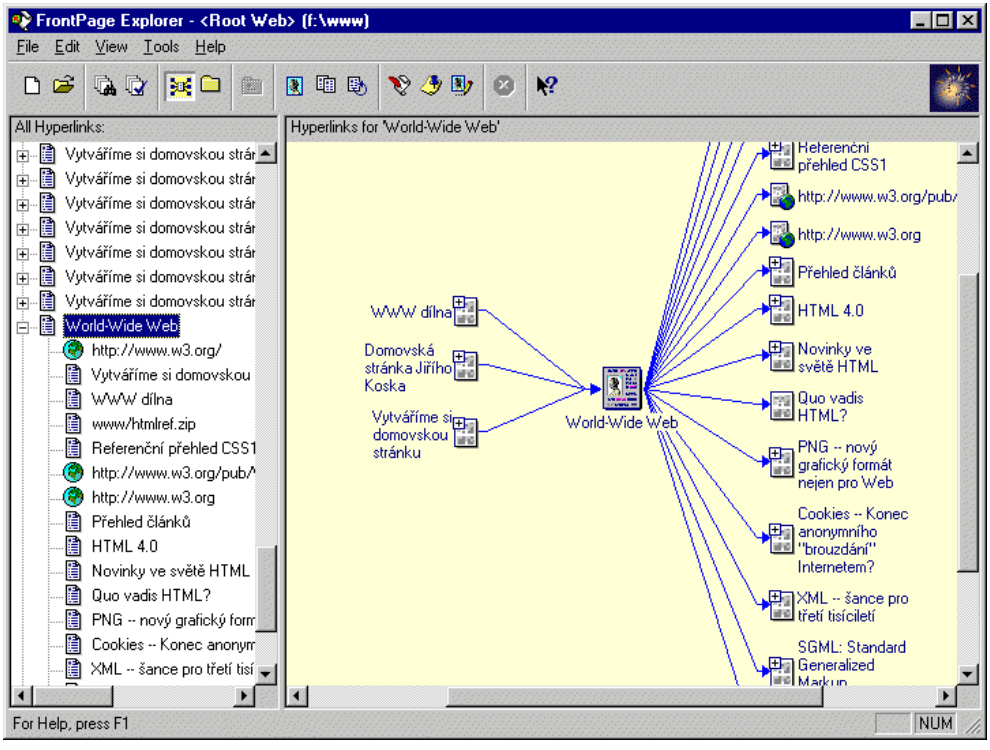
U velkých serverů, které obsahují velké množství provázaných stránek, je uměním udržovat všechny odkazy aktuální. Tuto práci nám mohou usnadnit speciální programy, které mají ulehčit správu rozsáhlých skupin stránek.

Typickým zástupcem této skupiny je *FrontPage Explorer*. Ten graficky znázorňuje strukturu odkazů mezi stránkami, umí tyto odkazy i automaticky kontrolovat. Dvojitým kliknutím na stránku pak lze vyvolat její editaci. Tento program můžeme používat i když se nám nelíbí *FrontPage Editor* — lze si nastavit vlastní program, který se použije pro editaci stránek.

15.3 Kontrola správnosti stránky

HTML stránka je textový soubor doplněný o tagy, které mohou obsahovat různé atributy. Definice HTML přitom jasně říká, kdy a které tagy a atributy můžeme použít. Existují programy, které nám dovolí zkontrolovat, zda stránka vyhovuje definici HTML.

Specializovaným programem, který slouží pro tyto účely, je *weblint*. Pomocí parametrů můžeme určit, která verze HTML se při kontrole našeho dokumentu použije.



Obr. 15-1: FrontPage Explorer

Jelikož je HTML definováno pomocí jazyka SGML, lze použít ke kontrole syntaktické správnosti stránky libovolný *parser*. Parser je program, který umožňuje zkontrolovat, zda daný soubor obsahuje text označkovaný podle určitého DTD. Parserů existuje několik, zajímavý je např. volně šiřitelný *NSGLMS*.

15.4 Konverze

Při rutinním spravování stránek na nějakém serveru se poměrně často dostaneme do situace, kdy je potřeba již existující dokument umístit na Web jako HTML-stránku. Dokument je přitom k dispozici v různých formátech textových procesorů, tabulkových kalkulátorů či databází. Někdy máme i opačný problém. Jak HTML-stránku načíst do našeho oblíbeného textového editoru. My si zde ukážeme, jak si poradit s konverzí těch nejběžnějších formátů.

Kancelářské aplikace

Na osobních počítačích PC je dnes asi nejpoužívanější kancelářskou aplikací *Microsoft Office*. V jeho poslední verzi *Office 97* s konverzí není žádný problém. Poslední verze *Wordu* i *Excelu* umějí bez větších problému HTML stránky načítat i ukládat. Pokud HTML stránku vytvoříme takto, měli bychom si ji před finálním vystavením na Webu prohlédnout a provést v ní případné ruční úpravy. Takto zkonvertované stránky totiž často obsahují mnoho prázdných a nepotřebných elementů. Ty je možno smazat a zmenšit tak velikost stránky.

Starší verze *Wordu* umějí HTML pouze načítat a nebo jej nepodporují vůbec. Lze je však o tyto možnosti rozšířit pomocí nadstavby *Internet Assistant*, která je k dispozici zdarma na serverech Microsoftu.

Podobné možnosti nabízejí i ostatní kancelářské balíky. (Textový procesor *Lotus WordPro* v sobě obsahoval podporu HTML téměř o rok dříve než *MS-Word*.)

Textové soubory

Z HTML lze vytvořit obyčejný textový soubor poměrně snadno. Většina prohlížečů nabízí možnost uložení stránky v tomto formátu. Stačí vybrat File ▷ Save As... a jako typ souboru zvolit textový soubor (přípona `.txt`).

Pokud chceme z HTML získat text doplněný i o formátování, můžeme použít nějaký novější textový editor, který to umí. Pokud jej nemáme k dispozici, můžeme použít program *Jade* od Jamese Clarka. *Jade* (James' DSSSL Engine) je program, který umí převádět SGML dokumenty na základě formátování předepsaného stylem DSSSL do několika formátů. Mezi těmito formáty je i formát RTF, se kterým si poradí většina editorů. *Jade* si je možno stáhnout na adrese `http://www.jclark.com/`.

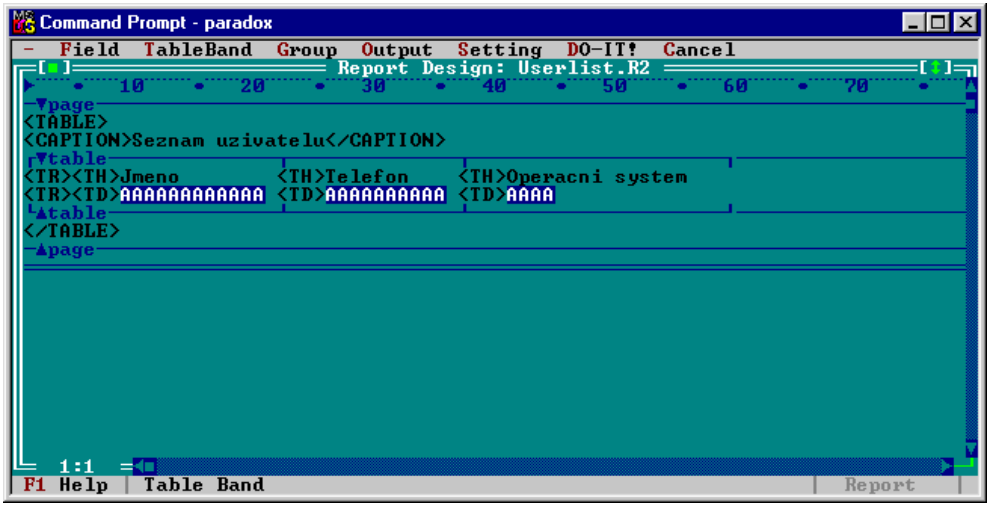
Pokud chceme konvertovat textový soubor do HTML, máme opět několik možností. Pokud se jedná o čisté ASCII a pospícháme, stačí textový soubor vložit mezi tagy `<PRE>` a `</PRE>`. Výsledek sice nebude graficky nejlepší, zato jej získáme velice rychle.

Pokud do HTML chceme konvertovat dokument z formátu RTF, můžeme použít program *rtftohtml*.

Pro konverzi dokumentů z LaTeXu do HTML můžeme použít program *LaTeX2html*. Konverze je pro mnoho případů celkem snadná. LaTeX je nadstavba nad typografickým systémem TeX, která definuje makra pro logické označování dokumentů na poměrně vysoké úrovni. Tato makra mnohdy odpovídají přímo HTML elementům, mají však odlišnou syntaxi. Pokud dokument v LaTeXu obsahuje něco, co překračuje možnosti HTML, automaticky se zavolá LaTeXa z problematického místa se vytvoří obrázek, který se vloží do stránky — takto se převedou např. matematické vzorce.

Databáze

V případě databází je většinou potřeba převést nějakou datovou tabulku do formátu HTML. To v principu není problém, protože HTML nám umožňuje používat tabulky a data lze tedy přehledně zformátovat. Pokud databázový program neumí provést konverzi sám, není proč se rmoutit. V databázovém programu vytvoříme jednoduchou výstupní sestavu, ve které se jednotlivé položky databáze doplní o tagy, které ve výsledku vytvoří HTML kód tabulky. Tuto výstupní sestavu stačí vytisknout do souboru a máme vyhráno.



Obr. 15-2: Vytvoření výstupní sestavy pro HTML v db-systému Paradox

Při konverzí databází si však vždy musíme ověřit, jak jsou data dlouhá. Pro dlouhé tabulky trvá dlouho přenos i zobrazení. Rychlost zobrazení lze zlepšit tak, že z každého záznamu databáze uděláme jednu samostatnou tabulku. U jednotlivých buněk `TD` a `TH` nastavíme šířku `WIDTH` na pevnou hodnotu, aby pod sebou lícovaly hodnoty jednotlivých položek.

Jelikož jsou však databáze většinou velké a nezřídka se v čase mění, je lepší je zpřístupnit pomocí systému stránek a CGI-skriptů, které uživateli umožní vybrat si z celé databáze jen ty záznamy, které ho zajímají.

Výpisy programů

Pokud chceme do stránky zařadit výpis zdrojového textu nějakého programu, můžeme ho uzavřít mezi tagy `<PRE>` a `</PRE>`. Kromě toho bychom měli nahradit znaky '`<`', '`>`' a '`&`' příslušnými znakovými entitami. Toto řešení je funkční, existují však lepší.

Zápis programu se zpřehlední, pokud použijeme zvýrazňovaní syntaxe. To znamená, že od sebe graficky odlišíme jednotlivé prvky programovacího jazyka — klíčová slova zobrazíme např. tučně a komentáře kurzívou. Tuto konverzi je samozřejmě dobré provést automaticky pomocí nějakého programu. S výhodou lze opět použít element `PRE` a uvnitř něj měnit použitý druh písma. Druhou možností je pro text programu použít proporcionální písmo a odsazení bloků programu řídit pomocí různého počtu tvrdých mezer (` `) na začátku řádky.

16. SGML aneb Web v dalším tisíciletí

V této kapitole si objasníme obsah pojmů SGML a DTD, které jsou pro dokonalé zvládnutí HTML potřebné. V mnoha jiných publikacích jim však není věnován žádný prostor a když už je, tak bývá vyplněn několika polopravdami a místy až lžmi.

Kromě toho si uděláme malou exkurzi do Webu příštího tisíciletí — povíme si něco o jazycích XML a DSSSL.

16.1 SGML

SGML (Standard Generalized Markup Language) je definován normou ISO 8879 z roku 1986. Jde o metajazyk, který slouží k definování různých značkovacích jazyků.

Této definici se říká *DTD (Document Type Definition)*. DTD mimo jiné obsahuje definici všech elementů, které lze v dokumentu použít, jejich přípustných vztahů a atributů, které lze u každého elementu použít. Tato definice je díky SGML zapsána ve zcela standardizované podobě a proto může být snadno zpracovávána počítači. Celé si to můžeme představit tak, že SGML je něco jako programovací jazyk. Každé DTD je pak program zapsaný v SGML.

DTD definuje pouze jakou může mít dokument *syntaxi* — jména elementů, jejich používání. *Význam (sémantiku)* jednotlivých elementů však musíme definovat jinak. Např. pro HTML obsahuje jeho specifikace popis použití všech elementů — autor tak ví, k čemu který element použít. Pro prohlížeč je zase naopak důležité vědět, jak se má obsah jednotlivých elementů zobrazit — tuto definici může mít v sobě prohlížeč zabudovanou napevno nebo může využívat např. styly, které definují vzhled jednotlivých elementů.

My můžeme vytvářet SGML-dokumenty, které vyhovují určitému DTD. SGML-dokument je textový soubor, který obsahuje text označkovaný pomocí tagů definovaných v DTD. Existují programy, tzv. *parsery*, které kontrolují, zda SGML-dokument vyhovuje DTD. Vyhoví mu v případě, že v dokumentu použijeme pouze ty elementy a atributy, které jsou v DTD definovány, a mezi elementy existují pouze přípustné vztahy. Aby parser věděl, které DTD jsme v dokumentu použili, musí každý SGML-dokument začínat *prologem*. Prolog má tvar

```
<!DOCTYPE «odkaz na DTD»>
```

«odkaz na DTD» můžeme mít několik tvarů a pro nás je zbytečné znát všechny možnosti. Pokud v našich HTML stránkách na začátku použijeme deklaraci

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">
```

můžeme naše stránky kontrolovat parserem. Pokud je parser správně nakonfigurován, nalezne DTD odpovídající HTML 3.2 a dokument zkontroluje. DTD jazyka HTML 3.2 je otištěno na konci této kapitoly. Předtím se však ještě podíváme na samotný jazyk SGML a jeho nejčastěji používané konstrukce při zápisu DTD.

16.2 Jak číst DTD?

Když chceme vytvářet dokumenty, které vyhovují určitému DTD — např. HTML dokumenty — musíme vědět, kdy a jaké elementy a atributy můžeme použít. To se můžeme dozvědět z nějakého návodu, který tvorbu dokumentů popisuje a přitom zatajuje nějaké SGML a DTD. V tomto duchu je koncipováno předchozích 15 kapitol této knihy. Druhou možností je přečíst si potřebné informace v DTD. Abychom věděli, k čemu jednotlivé elementy použít, bývají součástí DTD i komentáře, které čtenáři vysvětlují použití elementu. Komentáře se uzavírají mezi pár značek `--`. Nyní se podíváme na jednotlivé definice, které se v DTD mohou vyskytnout.

Definice entit

Definice entit jsou pouze pomůckou. Umožňují pro často používané řetězce definovat entitu, která je zastupuje. Je to něco jako makro v programovacích jazycích. Definice entity má tvar

`<!ENTITY %` *«jméno entity»* `"`*«řetězec»*`">`

V praxi můžeme mít definovánu entitu následovně:

```
<!ENTITY % heading "H1|H2|H3|H4|H5|H6">
```

Když nyní v DTD kdekoliv použijeme `%heading`, je to stejné, jako bychom na tomto místě použili řetězec `H1|H2|H3|H4|H5|H6`.

Definice elementů

Při vytváření dokumentu založeném na nějakém DTD — třeba v HTML — musíme znát elementy, které můžeme používat. Definice elementu se skládá ze tří částí:

1. jména elementu;
2. informace o tom, zda jsou počáteční a ukončovací tag povinné nebo volitelné;
3. přípustného obsahu elementu — které další elementy mohou být uvnitř definovaného elementu použity.

Definice elementu pro zápis tabulek v HTML je následující:

```
<!ELEMENT table - - (caption?, tr+)>
```

Definice nám říká, že definujeme element se jménem `table`. Protože za jménem následují znaky `- -`, musí být při použití elementu vždy použit počáteční i ukončovací tag. Pokud by mohl být vynechán ukončovací tag, použil by se zápis `- O`. Písmeno „O“ je zkratkou z anglického „Omit“ — což znamená, že něco můžeme opominout. `O` může být samozřejmě i na prvním místě, ale to není příliš obvyklé.[1]

Poslední část definice nám říká, jaké elementy můžeme uvnitř `table` použít. V tomto případě zápis `(caption?, tr+)` znamená, že uvnitř `table` můžeme použít jeden element `caption` následovaný několika elementy `tr`. Když jsi vzpomeneme na tabulky HTML, není to nic nového — tabulka může začínat definicí svého popisu a po něm následují její jednotlivé řádky.

Pokud element nemá obsahovat nic, použije se klíčové slovo `EMPTY`. Definice elementu `HR` je v HTML DTD tedy následující:

```
<!ELEMENT HR - O EMPTY>
```

Přehled všech symbolů se speciálním významem, které můžeme použít při definici obsahu elementu, je v tabulce 16-1.

Notace	Význam
`( ... )`	Uzavření skupiny.
A `|` *B*	Může se vyskytnout *A* i *B* v libovolném pořadí.
A`,` *B*	*A* se musí vyskytnout před *B*.
A `&` *B*	*A* a *B* se musí vyskytnout jednou v libovolném pořadí.
A`?`	*A* se může vyskytnout jednou nebo nevyskytnout.
A`*`	*A* se může vyskytnout libovolněkrát (i nulakrát).
A`+`	*A* se musí vyskytnout alespoň jednou (může i vícekrát).

Tab. 16-1: Symboly použitelné v definici obsahu elementu

Kromě přípustných elementů může definice obsahu elementu obsahovat klíčové slovo `#PCDATA` (parsed character data). Jedná se o libovolný text, ve kterém jsou rozeznávány komentáře a znakové entity. Pro procvičení se ještě podíváme na definici elementu `tr`:

```
<!ELEMENT tr - O (th|td)*>
```

Podle výše popsaných pravidel se element `tr` může skládat z libovolného počtu elementů `th` a `td`. U elementu `tr` nemusí být používán ukončovací tag `</tr>`.

[1] V HTML jsou takto definovány např. elementy `HTML`, `HEAD` a `BODY`, jejichž použití je nepovinné.

Definice atributů

Pro každý element můžeme v DTD definovat atributy, které lze používat. Definice atributů se skládá ze jména elementu a z definic všech atributů. Definice jednoho atributu se skládá ze:

1. jména atributu;
2. typu atributu nebo množiny přípustných hodnot;
3. definice,
   - zda se atribut musí vždy použít (`#REQUIRED`) nebo
   - zda je standardní hodnota atributu implicitní (`#IMPLIED`) nebo
   - zda je hodnota atributu určitá fixní hodnota (`#FIXED`).

My si ukážeme definici atributů na elementu `AREA`. Před samotnou definicí atributů jsou definovány ještě tři pomocné entity, které se využívají:

```
<!ENTITY % SHAPE "(rect|circle|poly)">
<!ENTITY % COORDS "CDATA" -- čárkami oddělený
                             seznam souřadnic -->
<!ENTITY % URL "CDATA"
        -- URL je atribut typu CDATA.
           Obsahuje adresu ve formě Uniform Resource Locator,
           definovanou v RFC 1808 a RFC 1738.
        -->

<!ATTLIST AREA
  shape   %SHAPE  rect
  coords  %COORDS #IMPLIED  -- souřadnice tvaru "shape" --
  href    %URL    #IMPLIED  -- oblast je aktivním odkazem --
  nohref (nohref) #IMPLIED  -- oblast je neaktivní --
  alt     CDATA   #REQUIRED -- nutné pro textové prohlížeče --
  >
```

Z definice mimo jiné vyplývá, že atribut `shape` může nabývat jedné z hodnot `rect`, `circle` a `poly`. Standardní je hodnota `rect`. Atribut `coords` může obsahovat data typu `CDATA` (character data). To je speciální datový typ — jedná se o řetězec, ve kterém jsou znakové entity (jako je např. `&`) před dalším zpracováním nahrazeny příslušnými znaky.

Atribut `href` obsahuje URL — to je znakový řetězec a komentář vymezuje, co by měl řetězec obsahovat. Atribut `nohref` je speciální. Slouží pouze jako jakýsi logický přepínač. Při použití elementu `AREA` v dokumentu se uvádí pouze jeho jméno (`<AREA NOHREF>`). Atribut `alt` by měl být přítomen u každého elementu `AREA`, protože ve své definici obsahuje klíčové slovo `#REQUIRED`.

Se znalostmi, které máme, by nám nyní nemělo činit potíže číst si v DTD jazyka HTML 3.2 uvedeném v poslední sekci této kapitoly. Do tohoto DTD se můžeme podívat vždy, když si nebudeme jisti nějakou kombinací elementů nebo hodnotou atributu.[2]

16.3 Web ve 21. století

Obrovské množství informací, kterými lidstvo disponuje, přináší mnoho problémů při jejich zpracování. Jedním z těchto problémů je sdílení informací po celém světě. Tento problém se poměrně úspěšně podařilo vyřešit pomocí Internetu a jeho služby WWW. Informace uložené na Webu jsou přístupné všem a vůbec nezávisí na tom, jaký používají počítač a operační systém (i když i toto se snaží Microsoft svou pseudotechnologií ActiveX změnit ;–).

Jazyk HTML, který se dnes používá pro tvorbu webovských stránek, má však některé rysy, které zabraňují používání efektivnějších prohledávacích metod. V HTML se jednotlivé části textu označují značkami, které textu přiřazují většinou čistě prezentační význam. Text lze označit jako nadpis, buňku tabulky nebo úsek, který bude zobrazen zvýrazněně. Pomocí SGML si však můžeme definovat vlastní sadu značek a těmito značkami můžeme mnohem výstižněji označit jednotlivé části textu. Text lze tedy označit např. jako název básně, jméno autora, kurz akcie, rodné číslo atd. — záleží pouze na tom, jaké značky definujeme v DTD. Takto označkovaný a strukturovaný text má mnohem vyšší informační hodnotu než stránka zapsaná v HTML.

Jaké jsou pak výhody? V dokumentech založených na SGML můžeme snadno vyhledávat odpovědi na dotazy typu: „Které básně napsal pan X?“. Při použití HTML můžeme klást pouze dotaz typu: „Které stránky obsahují slovo X?“. Výhody prvního způsobu dotazování (tzv. *kontextového dotazování*) v dnešní informacemi přehlcené době netřeba zdůrazňovat.

Samozřejmě, že předchozí výhody půjde používat pouze v případech, kdy prohledávané dokumenty budou vyhovovat stejnému DTD (budou používat stejnou množinu značek). Již dnes existují DTD, které používají určité skupiny uživatelů a v budoucnosti patrně vzniknou ještě další. Své DTD tak budou používat matematici (MathML), historici, chemici, elektrotechnici, ...

Tyto úvahy nad budoucností sdílení informací v celosvětovém měřítku znějí lákavě, ale mají jeden háček. SGML je velmi obsáhlý standard (my jsme si ukázali jen ty základní vlastnosti). Jeho implementace v programech je tedy časově a finančně nákladná.

[2] Pevně však věřím, že to nebude potřeba a že jsem váženého čtenáře srozumitelně provedl všemi zákoutími jazyka HTML.

Této nevýhody, která z SGML dělala prostředí pouze pro velké a bohaté firmy a nikoliv pro masové použití na Webu, si byli členové konsorcia W3C vědomi. Vytvořili proto nový jazyk *XML (eXtensible Markup Language)*. XML je zjednodušenou verzí SGML. Pomocí XML opět můžeme vytvářet DTD, která definují použitelné elementy a atributy pro určitou skupinu dokumentů. Tím zůstává zachována veliká flexibilita, kterou nabízelo SGML. XML naopak neobsahuje spoustu specialit a fines, které sice SGML obsahuje, ale v praxi se použijí jen zřídka. To umožňuje mnohem snazší vývoj programů než tomu bylo u SGML. Tím, že je XML podmnožinou SGML, lze navíc pro zpracování XML-dokumentů použít stávající programy pro práci s SGML-dokumenty.

XML obsahuje ještě další zjednodušení, která usnadní jeho používání. Pokud např. dodržíme určitá pravidla při zápisu XML-dokumentu, nemusí pro tento dokument existovat jeho definice DTD.

Ačkoliv je návrh standardu XML ještě mláďátko (narodil se v listopadu 1996), již dnes existují jeho aplikace. Zatím se s XML počítá především jako s prostředkem pro výměnu a sdílení různých informací a dat. Například Microsoft pro definici kanálů[3] vytvořil formát CDF (Channel Definition Format) — ten je definován pomocí jazyka XML v DTD.

Na XML je postaven i nový formát používaný pro distribuci softwaru po síti. Formát podporuje Microsoft a další velké softwarové společnosti.

V budoucnosti by se XML mohlo používat i pro přenos běžných dokumentů. K tomu je však zapotřebí standardizovat nějaký stylový jazyk, kterým by se pro každé DTD mohla vytvořit definice vzhledu obsahu jednotlivých elementů a atributů. Jedním z kandidátů na tuto pozici je *DSSSL (Document Style Semantics and Specification Language)*. Tento jazyk je definován ISO normou číslo 10 179 z roku 1996. Jazyk je to opravdu komplexní a jeho implementace není snadná. Postupuje se tedy stejně jako v případě SGML — z DSSSL se vybralo to nejpotřebnější a vznikl jednodušší stylový jazyk zvaný DSSSL-Online. Jeho implementace je mnohem snazší než u kompletního DSSSL a to zvyšuje šance na rozšíření této nadějné technologie do každodenního života.

16.4 HTML 3.2 DTD

V této sekci nalezneme definici HTML 3.2 ve formě DTD. Poslouží jako ukázka definice značkovacího jazyka v SGML. Její další a ještě důležitější význam spočívá v možnosti přesně zjistit, kdy lze jaký element a atribut na HTML stránce použít.

[3] Kanály automaticky stahují z Internetu aktuální obsah zajímavých serverů do počítače uživatele.

```
<!--
        W3C Document Type Definition for the HyperText Markup Language
        version 3.2 as ratified by a vote of W3C member companies.
        For more information on W3C look at  URL http://www.w3.org/

        Date: Tuesday January 14th 1997

        Author: Dave Raggett <dsr@w3.org>

        HTML 3.2 aims to capture recommended practice as of early '96
        and as such to be used as a replacement for HTML 2.0 (RFC 1866).
        Widely deployed rendering attributes are included where they
        have been shown to be interoperable. SCRIPT and STYLE are
        included to smooth the introduction of client-side scripts
        and style sheets. Browsers must avoid showing the contents
        of these element Otherwise support for them is not required.
        ID, CLASS and STYLE attributes are not included in this version
        of HTML.
-->

<!ENTITY % HTML.Version
        "-//W3C//DTD HTML 3.2 Final//EN"

        -- Typical usage:

            <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
            <html>
            ...
            </html>
        --
        >

<!--================== Deprecated Features Switch ========================-->

<!ENTITY % HTML.Deprecated "INCLUDE">

<!--================== Imported Names ====================================-->

<!ENTITY % Content-Type "CDATA"
        -- meaning a MIME content type, as per RFC1521
        -->

```

```
<!ENTITY % HTTP-Method "GET | POST"
        -- as per HTTP specification
        -->

<!ENTITY % URL "CDATA"
        -- The term URL means a CDATA attribute
           whose value is a Uniform Resource Locator,
           See RFC1808 (June 95) and RFC1738 (Dec 94).
        -->

<!-- Parameter Entities -->

<!ENTITY % head.misc "SCRIPT|STYLE|META|LINK" -- repeatable head elements -->

<!ENTITY % heading "H1|H2|H3|H4|H5|H6">

<!ENTITY % list "UL | OL |  DIR | MENU">

    <!ENTITY % preformatted "PRE | XMP | LISTING">
]]>

<!ENTITY % preformatted "PRE">

<!--================ Character mnemonic entities =========================-->

<!ENTITY % ISOlat1 PUBLIC
       "ISO 8879-1986//ENTITIES Added Latin 1//EN//HTML">
%ISOlat1;

<!--================ Entities for special symbols ========================-->
<!-- &trade and &cbsp are not widely deployed and so not included here -->

<!ENTITY amp    CDATA "&#38;"   -- ampersand         -->
<!ENTITY gt     CDATA "&#62;"   -- greater than      -->
<!ENTITY lt     CDATA "&#60;"   -- less than         -->

<!--================== Text Markup ======================================-->

<!ENTITY % font "TT | I | B  | U | STRIKE | BIG | SMALL | SUB | SUP">

<!ENTITY % phrase "EM | STRONG | DFN | CODE | SAMP | KBD | VAR | CITE">

```

```
<!ENTITY % special "A | IMG | APPLET | FONT | BASEFONT | BR | SCRIPT | MAP">

<!ENTITY % form "INPUT | SELECT | TEXTAREA">

<!ENTITY % text "#PCDATA | %font | %phrase | %special | %form">

<!ELEMENT (%font|%phrase) - - (%text)*>

<!-- there are also 16 widely known color names although
  the resulting colors are implementation dependent:

   aqua, black, blue, fuchsia, gray, green, lime, maroon,
   navy, olive, purple, red, silver, teal, white, and yellow

 These colors were originally picked as being the standard
 16 colors supported with the Windows VGA palette.
 -->

<!ELEMENT FONT - - (%text)*   -- local change to font -->
<!ATTLIST FONT
    size    CDATA   #IMPLIED  -- [+]nn e.g. size="+1", size=4 --
    color   CDATA   #IMPLIED  -- #RRGGBB in hex, e.g. red: color="#FF0000" --
    >

<!ELEMENT BASEFONT - O EMPTY  -- base font size (1 to 7)-->
<!ATTLIST BASEFONT
    size    CDATA   #IMPLIED  -- e.g. size=3 --
    >

<!ELEMENT BR    - O EMPTY    -- forced line break -->
<!ATTLIST BR
        clear (left|all|right|none) none -- control of text flow --
        >

<!--================== HTML content models ================================-->
<!--
    HTML has three basic content models:

        %text       character level elements and text strings
        %flow       block-like elements e.g. paragraphs and lists
        %bodytext   as %flow plus headers H1-H6 and ADDRESS
-->
```

```
127
128 <!ENTITY % block
129      "P | %list | %preformatted | DL | DIV | CENTER |
130       BLOCKQUOTE | FORM | ISINDEX | HR | TABLE">
131
132 <!-- %flow is used for DD and LI -->
133
134 <!ENTITY % flow "(%text | %block)*">
135
136 <!--================== Document Body =====================================-->
137
138 <!ENTITY % body.content "(%heading | %text | %block | ADDRESS)*">
139
140 <!ENTITY % color "CDATA" -- a color specification: #HHHHHH @@ details? -->
141
142 <!ENTITY % body-color-attrs "
143         bgcolor %color #IMPLIED
144         text %color #IMPLIED
145         link %color #IMPLIED
146         vlink %color #IMPLIED
147         alink %color #IMPLIED
148         ">
149
150 <!ELEMENT BODY O O  %body.content>
151 <!ATTLIST BODY
152         background %URL #IMPLIED  -- texture tile for document background --
153         %body-color-attrs;  -- bgcolor, text, link, vlink, alink --
154         >
155
156 <!ENTITY % address.content "((%text;) | P)*">
157
158 <!ELEMENT ADDRESS - - %address.content>
159
160 <!ELEMENT DIV - - %body.content>
161 <!ATTLIST DIV
162         align   (left|center|right) #IMPLIED -- alignment of following text --
163         >
164
165 <!-- CENTER is a shorthand for DIV with ALIGN=CENTER -->
166 <!ELEMENT center - - %body.content>
167
168 <!--================== The Anchor Element ================================-->
```

```
169
170 <!ELEMENT A - - (%text)* -(A)>
171 <!ATTLIST A
172         name    CDATA   #IMPLIED    -- named link end --
173         href    %URL    #IMPLIED    -- URL for linked resource --
174         rel     CDATA   #IMPLIED    -- forward link types --
175         rev     CDATA   #IMPLIED    -- reverse link types --
176         title   CDATA   #IMPLIED    -- advisory title string --
177         >
178
179 <!--================== Client-side image maps ============================-->
180
181 <!-- These can be placed in the same document or grouped in a
182      separate document although this isn't yet widely supported -->
183
184 <!ENTITY % SHAPE "(rect|circle|poly)">
185 <!ENTITY % COORDS "CDATA" -- comma separated list of numbers -->
186
187 <!ELEMENT MAP - - (AREA)*>
188 <!ATTLIST MAP
189     name    CDATA   #IMPLIED
190     >
191
192 <!ELEMENT AREA - O EMPTY>
193 <!ATTLIST AREA
194     shape   %SHAPE  rect
195     coords  %COORDS #IMPLIED  -- defines coordinates for shape --
196     href    %URL    #IMPLIED  -- this region acts as hypertext link --
197     nohref (nohref) #IMPLIED  -- this region has no action --
198     alt     CDATA   #REQUIRED -- needed for non-graphical user agents --
199     >
200
201 <!--================== The LINK Element ==================================-->
202
203 <!ENTITY % Types "CDATA"
204         -- See Internet Draft: draft-ietf-html-relrev-00.txt
205            LINK has been part of HTML since the early days
206            although few browsers as yet take advantage of it.
207
208            Relationship values can be used in principle:
209
210                 a) for document specific toolbars/menus when used
```

```
211                   with the LINK element in the document head:
212                b) to link to a separate style sheet
213                c) to make a link to a script
214                d) by stylesheets to control how collections of
215                   html nodes are rendered into printed documents
216                e) to make a link to a printable version of this document
217                   e.g. a postscript or pdf version
218 -->
219
220 <!ELEMENT LINK - O EMPTY>
221 <!ATTLIST LINK
222         href    %URL    #IMPLIED    -- URL for linked resource --
223         rel     %Types  #IMPLIED    -- forward link types --
224         rev     %Types  #IMPLIED    -- reverse link types --
225         title   CDATA   #IMPLIED    -- advisory title string --
226         >
227
228 <!--==================== Images ===============================================-->
229
230 <!ENTITY % Length "CDATA"   -- nn for pixels or nn% for percentage length -->
231 <!ENTITY % Pixels "NUMBER"  -- integer representing length in pixels -->
232
233 <!-- Suggested widths are used for negotiating image size
234      with the module responsible for painting the image.
235      align=left or right cause image to float to margin
236      and for subsequent text to wrap around image -->
237
238 <!ENTITY % IAlign "(top|middle|bottom|left|right)">
239
240 <!ELEMENT IMG    - O EMPTY --  Embedded image -->
241 <!ATTLIST IMG
242         src     %URL     #REQUIRED  -- URL of image to embed --
243         alt     CDATA    #IMPLIED   -- for display in place of image --
244         align   %IAlign  #IMPLIED   -- vertical or horizontal alignment --
245         height  %Pixels  #IMPLIED   -- suggested height in pixels --
246         width   %Pixels  #IMPLIED   -- suggested width in pixels --
247         border  %Pixels  #IMPLIED   -- suggested link border width --
248         hspace  %Pixels  #IMPLIED   -- suggested horizontal gutter --
249         vspace  %Pixels  #IMPLIED   -- suggested vertical gutter --
250         usemap  %URL     #IMPLIED   -- use client-side image map --
251         ismap   (ismap)  #IMPLIED   -- use server image map --
252         >
```

```
253
254 <!-- USEMAP points to a MAP element which may be in this document
255   or an external document, although the latter is not widely supported -->
256
257 <!--================== Java APPLET tag ===================================-->
258 <!--
259   This tag is supported by all Java enabled browsers. Applet resources
260   (including their classes) are normally loaded relative to the document
261   URL (or <BASE> element if it is defined). The CODEBASE attribute is used
262   to change this default behavior. If the CODEBASE attribute is defined then
263   it specifies a different location to find applet resources. The value
264   can be an absolute URL or a relative URL. The absolute URL is used as is
265   without modification and is not effected by the documents <BASE> element.
266   When the codebase attribute is relative, then it is relative to the
267   document URL (or <BASE> tag if defined).
268 -->
269 <!ELEMENT APPLET - - (PARAM | %text)*>
270 <!ATTLIST APPLET
271         codebase %URL     #IMPLIED   -- code base --
272         code     CDATA    #REQUIRED  -- class file --
273         alt      CDATA    #IMPLIED   -- for display in place of applet --
274         name     CDATA    #IMPLIED   -- applet name --
275         width    %Pixels  #REQUIRED  -- suggested width in pixels --
276         height   %Pixels  #REQUIRED  -- suggested height in pixels --
277         align    %IAlign  #IMPLIED   -- vertical or horizontal alignment --
278         hspace   %Pixels  #IMPLIED   -- suggested horizontal gutter --
279         vspace   %Pixels  #IMPLIED   -- suggested vertical gutter --
280         >
281
282 <!ELEMENT PARAM - O EMPTY>
283 <!ATTLIST PARAM
284         name    NMTOKEN   #REQUIRED  -- The name of the parameter --
285         value   CDATA     #IMPLIED   -- The value of the parameter --
286         >
287
288 <!--
289 Here is an example:
290
291     <applet codebase="applets/NervousText"
292         code=NervousText.class
293         width=300
294         height=50>
```

```
295     <param name=text value="Java is Cool!">
296     <img src=sorry.gif alt="This looks better with Java support">
297     </applet>
298 -->
299
300 <!--=================== Horizontal Rule ================================-->
301
302 <!ELEMENT HR    - O EMPTY>
303 <!ATTLIST HR
304         align (left|right|center) #IMPLIED
305         noshade (noshade) #IMPLIED
306         size  %Pixels #IMPLIED
307         width %Length #IMPLIED
308         >
309 <!--=================== Paragraphs=====================================-->
310
311 <!ELEMENT P     - O (%text)*>
312 <!ATTLIST P
313         align  (left|center|right) #IMPLIED
314         >
315
316 <!--=================== Headings ======================================-->
317
318 <!--
319   There are six levels of headers from H1 (the most important)
320   to H6 (the least important).
321 -->
322
323 <!ELEMENT ( %heading )  - -  (%text;)*>
324 <!ATTLIST ( %heading )
325         align  (left|center|right) #IMPLIED
326         >
327
328 <!--=================== Preformatted Text ==============================-->
329
330 <!-- excludes images and changes in font size -->
331
332 <!ENTITY % pre.exclusion "IMG|BIG|SMALL|SUB|SUP|FONT">
333
334 <!ELEMENT PRE - - (%text)* -(%pre.exclusion)>
335 <!ATTLIST PRE
336         width NUMBER #implied -- is this widely supported? --
```

```
337         >
338
339 <![ %HTML.Deprecated [
340
341 <!ENTITY % literal "CDATA"
342         -- historical, non-conforming parsing mode where
343            the only markup signal is the end tag
344            in full
345         -->
346
347 <!ELEMENT (XMP|LISTING) - -  %literal>
348 <!ELEMENT PLAINTEXT - O %literal>
349
350 ]]>
351
352 <!--================== Block-like Quotes ================================-->
353
354 <!ELEMENT BLOCKQUOTE - - %body.content>
355
356 <!--================== Lists ============================================-->
357
358 <!--
359     HTML 3.2 allows you to control the sequence number for ordered lists.
360     You can set the sequence number with the START and VALUE attributes.
361     The TYPE attribute may be used to specify the rendering of ordered
362     and unordered lists.
363 -->
364
365 <!-- definition lists - DT for term, DD for its definition -->
366
367 <!ELEMENT DL    - -  (DT|DD)+>
368 <!ATTLIST DL
369         compact (compact) #IMPLIED -- more compact style --
370         >
371
372 <!ELEMENT DT - O  (%text)*>
373 <!ELEMENT DD - O  %flow;>
374
375 <!-- Ordered lists OL, and unordered lists UL -->
376 <!ELEMENT (OL|UL) - -  (LI)+>
377
378 <!--
```

```
379        Numbering style
380     1   arabic numbers     1, 2, 3, ...
381     a   lower alpha        a, b, c, ...
382     A   upper alpha        A, B, C, ...
383     i   lower roman        i, ii, iii, ...
384     I   upper roman        I, II, III, ...
385
386     The style is applied to the sequence number which by default
387     is reset to 1 for the first list item in an ordered list.
388
389     This can't be expressed directly in SGML due to case folding.
390 -->
391
392 <!ENTITY % OLStyle "CDATA" -- constrained to: [1|a|A|i|I] -->
393
394 <!ATTLIST OL -- ordered lists --
395         type      %OLStyle   #IMPLIED   -- numbering style --
396         start     NUMBER     #IMPLIED   -- starting sequence number --
397         compact  (compact)   #IMPLIED   -- reduced interitem spacing --
398         >
399
400 <!-- bullet styles -->
401
402 <!ENTITY % ULStyle "disc|square|circle">
403
404 <!ATTLIST UL -- unordered lists --
405         type    (%ULStyle)   #IMPLIED   -- bullet style --
406         compact (compact)    #IMPLIED   -- reduced interitem spacing --
407         >
408
409 <!ELEMENT (DIR|MENU) - -  (LI)+ -(%block)>
410 <!ATTLIST DIR
411         compact (compact) #IMPLIED
412         >
413 <!ATTLIST MENU
414         compact (compact) #IMPLIED
415         >
416
417 <!-- <DIR>              Directory list                -->
418 <!-- <DIR COMPACT>      Compact list style            -->
419 <!-- <MENU>             Menu list                     -->
420 <!-- <MENU COMPACT>     Compact list style            -->
```

```
421
422 <!-- The type attribute can be used to change the bullet style
423      in unordered lists and the numbering style in ordered lists -->
424
425 <!ENTITY % LIStyle "CDATA" -- constrained to: "(%ULStyle|%OLStyle)" -->
426
427 <!ELEMENT LI - O %flow -- list item -->
428 <!ATTLIST LI
429         type    %LIStyle     #IMPLIED   -- list item style --
430         value    NUMBER      #IMPLIED   -- reset sequence number --
431         >
432
433 <!--================ Forms ===============================================-->
434
435 <!ELEMENT FORM - - %body.content -(FORM)>
436 <!ATTLIST FORM
437         action %URL #IMPLIED  -- server-side form handler --
438         method (%HTTP-Method) GET -- see HTTP specification --
439         enctype %Content-Type; "application/x-www-form-urlencoded"
440         >
441
442 <!ENTITY % InputType
443         "(TEXT | PASSWORD | CHECKBOX | RADIO | SUBMIT
444             | RESET | FILE | HIDDEN | IMAGE)">
445
446 <!ELEMENT INPUT - O EMPTY>
447 <!ATTLIST INPUT
448         type %InputType TEXT     -- what kind of widget is needed --
449         name  CDATA   #IMPLIED   -- required for all but submit and reset --
450         value CDATA   #IMPLIED   -- required for radio and checkboxes --
451         checked (checked) #IMPLIED -- for radio buttons and check boxes --
452         size CDATA    #IMPLIED   -- specific to each type of field --
453         maxlength NUMBER #IMPLIED -- max chars allowed in text fields --
454         src   %URL    #IMPLIED   -- for fields with background images --
455         align %IAlign #IMPLIED   -- vertical or horizontal alignment --
456         >
457
458 <!ELEMENT SELECT - - (OPTION+)>
459 <!ATTLIST SELECT
460         name CDATA #REQUIRED
461         size NUMBER #IMPLIED
462         multiple (multiple) #IMPLIED
```

```
463         >
464
465 <!ELEMENT OPTION - O (#PCDATA)*>
466 <!ATTLIST OPTION
467         selected (selected) #IMPLIED
468         value  CDATA  #IMPLIED -- defaults to element content --
469         >
470
471 <!-- Multi-line text input field. -->
472
473 <!ELEMENT TEXTAREA - - (#PCDATA)*>
474 <!ATTLIST TEXTAREA
475         name CDATA #REQUIRED
476         rows NUMBER #REQUIRED
477         cols NUMBER #REQUIRED
478         >
479
480 <!--======================== Tables ========================================-->
481
482 <!-- Widely deployed subset of the full table standard, see RFC 1942
483      e.g. at http://www.ics.uci.edu/pub/ietf/html/rfc1942.txt -->
484
485 <!-- horizontal placement of table relative to window -->
486 <!ENTITY % Where "(left|center|right)">
487
488 <!-- horizontal alignment attributes for cell contents -->
489 <!ENTITY % cell.halign
490         "align  (left|center|right) #IMPLIED"
491         >
492
493 <!-- vertical alignment attributes for cell contents -->
494 <!ENTITY % cell.valign
495         "valign  (top|middle|bottom)  #IMPLIED"
496         >
497
498 <!ELEMENT table - - (caption?, tr+)>
499 <!ELEMENT tr - O (th|td)*>
500 <!ELEMENT (th|td) - O %body.content>
501
502 <!ATTLIST table                         -- table element --
503         align     %Where;   #IMPLIED  -- table position relative to window --
504         width     %Length   #IMPLIED  -- table width relative to window --
```

```
505         border    %Pixels   #IMPLIED  -- controls frame width around table --
506         cellspacing %Pixels #IMPLIED  -- spacing between cells --
507         cellpadding %Pixels #IMPLIED  -- spacing within cells --
508         >
509
510 <!ELEMENT CAPTION - - (%text;)* -- table or figure caption -->
511 <!ATTLIST CAPTION
512         align (top|bottom) #IMPLIED
513         >
514
515 <!ATTLIST tr                        -- table row --
516         %cell.halign;               -- horizontal alignment in cells --
517         %cell.valign;               -- vertical alignment in cells --
518         >
519
520 <!ATTLIST (th|td)                   -- header or data cell --
521         nowrap (nowrap)  #IMPLIED   -- suppress word wrap --
522         rowspan NUMBER   1          -- number of rows spanned by cell --
523         colspan NUMBER   1          -- number of cols spanned by cell --
524         %cell.halign;               -- horizontal alignment in cell --
525         %cell.valign;               -- vertical alignment in cell --
526         width   %Pixels  #IMPLIED   -- suggested width for cell --
527         height  %Pixels  #IMPLIED   -- suggested height for cell --
528         >
529
530 <!--================ Document Head ======================================-->
531
532 <!-- %head.misc defined earlier on as "SCRIPT|STYLE|META|LINK" -->
533
534 <!ENTITY % head.content "TITLE & ISINDEX? & BASE?">
535
536 <!ELEMENT HEAD O O  (%head.content) +(%head.misc)>
537
538 <!ELEMENT TITLE - -  (#PCDATA)* -(%head.misc)
539           -- The TITLE element is not considered part of the flow of text.
540              It should be displayed, for example as the page header or
541              window title.
542           -->
543
544 <!ELEMENT ISINDEX - O EMPTY>
545 <!ATTLIST ISINDEX
546         prompt CDATA #IMPLIED -- prompt message -->
```

```
547
548 <!--
549     The BASE element gives an absolute URL for dereferencing relative
550     URLs, e.g.
551
552          <BASE href="http://foo.com/index.html">
553          ...
554          <IMG SRC="images/bar.gif">
555
556     The image is deferenced to
557
558          http://foo.com/images/bar.gif
559
560    In the absence of a BASE element the document URL should be used.
561    Note that this is not necessarily the same as the URL used to
562    request the document, as the base URL may be overridden by an HTTP
563    header accompanying the document.
564 -->
565
566 <!ELEMENT BASE - O EMPTY>
567 <!ATTLIST BASE
568         href %URL  #REQUIRED
569         >
570
571 <!ELEMENT META - O EMPTY -- Generic Metainformation -->
572 <!ATTLIST META
573         http-equiv  NAME    #IMPLIED  -- HTTP response header name  --
574         name        NAME    #IMPLIED  -- metainformation name       --
575         content     CDATA   #REQUIRED -- associated information     --
576         >
577
578 <!-- SCRIPT/STYLE are place holders for transition to next version of HTML -->
579
580 <!ELEMENT STYLE  - - CDATA -- placeholder for style info -->
581 <!ELEMENT SCRIPT - - CDATA -- placeholder for script statements -->
582
583 <!-- ELEMENT STYLE  - - (#PCDATA)*  -(%head.misc) -- style info -->
584 <!-- ELEMENT SCRIPT - - (#PCDATA)*  -(%head.misc) -- script statements -->
585
586 <!--================ Document Structure ===================================-->
587
588 <!ENTITY % version.attr "VERSION CDATA #FIXED '%HTML.Version;'">
```

```
589
590 <![ %HTML.Deprecated [
591     <!ENTITY % html.content "HEAD, BODY, PLAINTEXT?">
592 ]]>
593
594 <!ENTITY % html.content "HEAD, BODY">
595
596 <!ELEMENT HTML O O  (%html.content)>
597 <!ATTLIST HTML
598         %version.attr;
599         >
```

Literatura

[1] Alschuler, L.: *ABCD...SGML*. International Thomson Computer Press 1995. ISBN 1-850-32197-3

[2] Berners-Lee, T.: *Style Guide for online hypertext*. 1995
`http://www.w3.org/hypertext/WWW/Provider/Style/Overview.html`

[3] Berners-Lee, T. – Connolly, D.: *Hypertext Markup Language 2.0*. RFC 1866, 1995
`ftp://ftp.vse.cz/pub/docs/rfc/rfc1866.txt`

[4] Berners-Lee, T. – Fielding, R. – Frystyk, H.: *Hypertext Transfer Protocol – HTTP/1.0*. RFC 1945, 1996
`ftp://ftp.vse.cz/pub/docs/rfc/rfc1945.txt`

[5] Berners-Lee, T. – Masinter, L. – McCahill, M.: *Uniform Resource Locators (URL)*. RFC 1738, 1994
`ftp://ftp.vse.cz/pub/docs/rfc/rfc1738.txt`

[6] Boutell, T.: *PNG (Portable Network Graphics) Specification – Version 1.0*. W3C recommendation, 1996
`http://www.w3.org/TR/REC-png.html`

[7] Bray, T. – Sperberg-McQueen, C.M.: *Extensible Markup Language (XML)*. W3C working draft, 1996
`http://www.textuality.com/sgml-erb/WD-xml.html`

[8] Fielding, R.: *Relative Uniform Resource Locators*. RFC 1808, 1995
`ftp://ftp.vse.cz/pub/docs/rfc/rfc1808.txt`

[9] Håkon, W. L. – Bos, B.: *Cascading Style Sheets, level 1*. W3C recommendation, 1996
`http://www.w3.org/TR/REC-CSS1`

[10] Ion, P. – Miner, R.: *Mathematical Markup Language*. W3C working draft, 1997
`http://www.w3.org/TR/WD-math-970515`

[11] Korpela, J: *Learning HTML 3.2 by Examples*. 1997
`http://www.hut.fi/~jkorpela/HTML3.2/`

[12] Kosek, J. – Janíková, V.: *Internet – první kroky českého uživatele*. Grada 1996. ISBN 80-7169-421-5

[13] Murray, J. D. – vanRyper, W.: *Encyklopedie grafických formátů*. Computer Press 1995. ISBN 80-85896-18-4

[14] Netscape Communications Corp.: *Persistent Client State – HTTP Cookies*. 1996
http://home.netscape.com/newsref/std/cookie_spec.html

[15] Prescod, P.: *Introduction to DSSSL*. 1997
http://itrc.uwaterloo.ca/~papresco/dsssl/tutorial.html

[16] Ragget, D. – Hors, A. L. – Jacobs, I.: *HTML 4.0 Specification*. W3C working draft, 1997
http://www.w3.org/TR/WD-html40-970708

[17] Raggett, D.: *HTML 3.2 Reference Specification*. W3C recommendation, 1997
http://www.w3.org/TR/REC-html32.html

[18] Raggett, D.: *Hypertext Markup Language Specification Version 3.0*. W3C working-draft, 1995
http://www.w3.org/MarkUp/html3/

[19] Satrapa, P.: *World-Wide Web pro čtenáře, autory a misionáře*. Neokortext 1996

[20] Sollins, K. – Masinter, L.: *Functional Requirements for Uniform Resource Names*. RFC 1737, 1994
ftp://ftp.vse.cz/pub/docs/rfc/rfc1737.txt

[21] Stevahn, R.: *Positioning HTML elements with Cascading Style Sheets*. W3C working draft, 1997
http://www.w3.org/TR/WD-positioning-970131.html

Rejstřík

Jiří Kosek

HTML
tvorba dokonalých WWW stránek
podrobný průvodce

Návrh a grafická úprava obálky Stanislav Hudský
Ilustrace Ondřej Tůma
Počet stran 296

Vydala Grada Publishing, spol. s r. o.
U Průhonu 22, Praha 7

1998
Vydání 1.

Vytiskly Tiskárny Havlíčkův Brod, a.s.
Husova ulice 1881, Havlíčkův Brod