VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

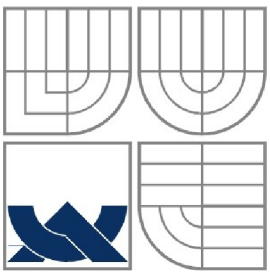
WEBOVÝ PROHLÍŽEČ PRO SQUEAK SMALLTALK

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

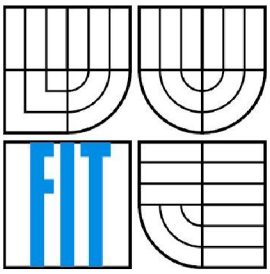
AUTOR PRÁCE
AUTHOR

Bc. MARTIN ŠLEMR

BRNO 2008



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

WEBOVÝ PROHLÍŽEČ PRO SQUEAK SMALLTALK

WEB BROWSER FOR SQUEAK SMALLTALK

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

BC. MARTIN ŠLEMR

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. Vladimír Janoušek, Ph.D.

BRNO 2008

Abstrakt

Práce se zabývá popisem webového prohlížeče Scamper v prostředí Squeak Smalltalk, jeho dosavadního vývoje, nového návrhu a implementace, která respektuje CSS box model a vizuální formátovací model včetně tabulek. Také objasňuje pojmy související s prohlížeči obecně a technologií kolem Internetu jako protokol HTTP a struktura pro popis dokumentů MIME. Další částí je popis systému Squeak Smalltalk a jeho grafického prostředí Morphic.

Klíčová slova

Internet, WWW, HTML, XHTML, CSS, HTTP, MIME, Smalltalk, Squeak, Morphic, Scamper.

Abstract

This Term Project is about web browser Scamper in Squeak Smalltalk system environment, it's actual progress, new design and implementation, which respect CSS box model and visual formatting model including tables. Also describe web browsers generally and Internet technologies such HTTP protocol, or structure MIME. Next part of this document is describing Squeak Smalltalk system and it's graphic environment Morphic.

Keywords

Internet, WWW, HTML, XHTML, CSS, HTTP, MIME, Smalltalk, Squeak, Morphic, Scamper.

Webový prohlížeč pro Squeak Smalltalk

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením Ing. Vladimíra Janouška, Ph.D.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Martin Šlemr
15.05.2008

Poděkování

Děkuji Ing. Janouškovi za odborné vedení a ochotě k řešení problémů při konzultacích.

Obsah

Obsah ... 1
1 Úvod ... 5
2 Historie ... 6
2.1 ARPANET ... 6
2.2 Hypertext ... 6
2.3 Statistika připojení k Internetu ... 7
2.4 Shrnutí ... 7
3 Webové prohlížeče ... 8
3.1 Historické prohlížeče ... 8
3.1.1 WorldWideWeb ... 9
3.1.2 Mosaic ... 9
3.1.3 Arena a Amaya ... 10
3.1.4 Ostatní historické prohlížeče ... 11
3.2 Renderovací jádra ... 11
3.2.1 Trident ... 12
3.2.2 Gecko ... 12
3.2.3 Presto ... 13
3.2.4 KHTML ... 13
3.3 Grafické prohlížeče ... 14
3.3.1 Microsoft Internet Explorer ... 14
3.3.2 Mozilla Firefox ... 15
4 MIME struktura ... 16
4.1 Typy zpráv MIME ... 16
4.1.1 Typ application ... 16
4.1.2 Typ audio ... 16
4.1.3 Typ image ... 16
4.1.4 Typ text ... 17
4.1.5 Typ multipart ... 17
5 HTTP - HyperText Transfer Protocol ... 18
5.1 Syntaxe HTTP ... 18
5.2 Požadavky - metody ... 18
5.2.1 Metoda GET ... 18
5.2.2 Metoda HEAD ... 18
5.2.3 Metoda POST ... 19

5.2.4 Metoda PUT 19
5.2.5 Metoda DELETE 19
5.3 Odpovědi 19
5.3.1 Stavové kódy 19
5.4 Shrnutí 20
6 HTML – HyperText Markup Language 21
6.1 DTD pro HTML 4.01 21
6.2 HTML elementy, tagy a atributy 21
6.3 Speciální značky 22
6.3.1 Html Latin-1 22
6.3.2 HTML Symbolové entity 23
6.4 Struktura dokumentu 23
6.5 Tok textu 23
6.6 Rozdělení elementů podle funkce 24
6.6.1 Tagy v hlavičce dokumentu 24
6.6.2 Základní tagy 24
6.6.3 Formátování písma 25
6.6.4 Výstupní tagy 25
6.6.5 Odkazy 26
6.6.6 Rámce 26
6.6.7 Formuláře 27
6.6.8 Seznamy 29
6.6.9 Obrázky 29
6.6.10 Tabulky 30
6.6.11 Styly 31
6.6.12 Programování 31
6.7 Standardní atributy 33
6.7.1 Základní atributy 33
6.7.2 Jazykové elementy 33
6.7.3 Klávesové elementy 33
6.8 Atributy událostí 33
6.8.1 Události oken 33
6.8.2 Události formulářů 33
6.8.3 Události klávesnice a myši 34
6.9 Barvy 34
6.10 Shrnutí 34
7 XHTML 35

7.1 DTD pro XHTML 1.0 ... 35
7.2 Struktura dokumentu ... 35
7.3 XHTML vs. HTML ... 35
7.4 Další pravidla XHTML syntaxe ... 36
8 Cascading Style Sheet ... 37
8.1 Selektory, pseudotřídy a pseudoelementy ... 37
8.1.1 Selektory ... 37
8.1.2 Pseudoelementy ... 37
8.1.3 Pseudotřídy ... 37
8.1.4 Syntaxe selektorů ... 37
8.2 CSS Box model ... 38
8.3 Shrnutí ... 39
9 Squeak Smalltalk ... 40
10 Morphic – grafické prostředí Squeaku ... 41
10.1 TableLayout ... 42
10.2 ProportionalLayout ... 42
10.3 Shrnutí ... 42
11 Scamper – původní stav ... 43
11.1 Dostupné funkce ... 43
11.2 Architektura Scamperu ... 44
11.3 Vrstva Model ... 44
11.3.1 Třída Scamper ... 44
11.3.2 Třída ScamperWorker ... 46
11.3.3 Třída BrowserUrl ... 46
11.4 Vrstva Parser ... 46
11.5 Vrstva Formatter ... 47
11.6 Vrstva View ... 48
11.7 Formulářové prvky ... 49
11.8 Shrnutí ... 50
12 Další vývoj Scamperu ... 51
12.1 Přehled změn ... 51
12.1.1 Nová architektura ... 51
12.2 Načtení HTML stránky ... 52
12.3 Parser ... 54
12.4 Načtení CSS pravidel do boxu ... 55
12.5 ScamperHtmlRenderer ... 56
12.5.1 Diagramy tříd CSS boxů ... 57

12.5.2 Určení typu netabulkového boxu ... 57
12.5.3 Určení typu tabulkového boxu ... 59
12.5.4 Úprava rodičovského boxu ... 59
12.5.5 Vytvoření HTML elementu ... 59
12.6 Renderování CSS boxů ... 60
12.6.1 Základní popis ... 60
12.6.2 Blokové boxy ... 61
12.6.3 Inline boxy ... 63
12.6.4 Tabulky ... 65
12.6.5 Převod jednotek na pixely ... 67
12.7 Zpracování obrázků ... 68
12.8 Hypertextové odkazy ... 70
12.9 Nedostatky a návrhy na rozšíření ... 70
12.9.1 Optimalizace ... 70
12.9.2 Možná rozšíření ... 70
12.10 Shrnutí ... 71
13 Závěr ... 72
Literatura ... 73
Seznam použitých zkratek a symbolů ... 75

1 Úvod

Webové prohlížeče jsou dnes téměř povinnou výbavou osobního počítače. V některých verzích Microsoft Windows dokonce tvoří základní součást operačního systému. V různých podobách a kvalitách jsou webové prohlížeče dostupné téměř na každé, ne-li na všech, platformách. Je to počítačový program, který slouží k prohlížení World Wide Webu (WWW). Program umožňuje komunikaci s http serverema zpracování přijatého kódu (HTML, XHTML, XML apod.), který podle daných standardů zformátuje a zobrazí webovou stránku. Textové prohlížeče zobrazují stránky jako text, obvykle velmi jednoduše formátovaný. Grafické prohlížeče umožňují složitější formátování stránky včetně zobrazení obrázků[1].

Cílem této práce je rozebrat současné techniky a standardy v zobrazování webových stránek. Dále popsat webový prohlížeč Scamper a navrhnout jeho úpravy tak, aby se stal co možná nejvíce plnohotnotným prohlížečem podporujícím nejnovější normy.

V další kapitole se podíváme na historii internetu a webových prohlížečů. Následovat bude kapitola o současných webových prohlížečích, z nichž budou popsány ty nejvýznamnější grafické, používající jádra *Trident*, *Gecko* a *KHTML*, či textové *Links* a *Lynx*. Ve zkratce uvedu i několik dalších. Také rozebereme ***HTTP*** *(Hyper Text Transfer Protokol)*, protokol používaný pro výměnu informací. Další kapitola rozebere ***HTML*** *(Hyper Text Markup Language)* podle standardu 4.01. Poté se přesuneme do jazyka Smalltalk a vývojového prostředí Squeak. Popíšeme jeho základní vlastnosti, chování a odlišnost od dnes běžně rozšířených programovacích jazyků.

Ovšem nejpodstatnější části této práce se budou týkat webového prohlížeče Scamper. Nejprve se podíváme na stav, v jakém se nachází v současné době. Další kapitola pak bude pojednávat o jeho rozšíření a přepracování prohlížeče tak, aby se vývoj dostal ze slepé uličky a dokázal zpracovat a zobrazit stránku tak, aby co nejvíce odpovídala specifikaci CSS2.1 a HTML 4.01.

Na závěr shrneme tento dokument a rozebereme možnosti případného rozšíření funkcionality.

2 Historie

Obsahem této kapitoly je pohled do historie internetu, shrnutí jeho vývoje a přechod k podobě, jak ji známe dnes.

První zmínky o využití počítačů ke komunikaci se objevily již roku 1945 v americkém časopise The Atlantic Monthly. Publikoval ho *Vannevar Bush*, jehož článek *As We May Think* bývá pokládán za jeden ze základních kamenů informační vědy. Ovšem první myšlenky na vytvoření sítě se začínaly objevovat až začátkem šedesátých let.

2.1 ARPANET

V roce 1962 vznikl v rámci úřadu ***ARPA*** *(the Advanced Research Projects Agency)* úřad **IPTO** pro vývoj počítačové sítě americké armády. Hned z počátku bylo požadavekem vytvořit decentralizovanou síť, která bude fungovat i po výpadku některého z uzlů. První testovací síť vznikla o devět let později, v roce 1969 ve Velké Británii. Měla název **ARPANET** a přepojovala pakety mezi čtyřmi uzly. Také byl navržen a realizován asymetrický protokol přenostu dat (telnet, ftp). Roku 1972 byl **ARPANET** rozšířen na cca 20 směrovačů a 50 počítačů. Komunikace probíhala protokolem *NCP*. Až do osmdesátých let 20. století se Internet příliš nerozvíjel. Významným počinem bylo až experimentální nasazení protokolu *TCP/IP* a *DNS protokol* *(Domain Name System)* v roce 1980. Jako standard pro **ARPANET** byl uznán až v roce 1983.

První komerční služby se váží k síti **NFSNet**, která byla spuštěna roku 1985 a tvořila opravdovou páteř počátku Internetu. Spojovala pět nejdůležitějších amerických počítačových supercenter.

2.2 Hypertext

Hypertext je informační systém, který zobrazuje informace v textu, který obsahuje návěstí odkazující na upřesnění nebo zdroje uváděných informací tzv. hyperlinky, neboli česky (hypertextové) odkazy. Rovněž odkazuje i na jiné informace v systému a umožňuje snadné publikování, údržbu a vyhledávání těchto informací. Nejznámějším takovým systémem je World Wide Web.

Z povahy hypertextu jakožto média vyplývá, že v něm neexistuje centrální, hlavní text, kterému jsou jiné texty podřazeny, jak je tomu v prostorové koncepci tištěné stránky. Princip organizace, hierarchie a nakládání s odkazy závisí na čtenáři [2].

Jako první myšlenku hypertextu vyjádřil ve výše uvedeném článku *Vannevar Bush* jako zařízení nazvané „Memex“. Definici pojmu publikoval až roku 1965 *Theodor Holm Nelson*.

Za zmínku určitě stojí rok 1980 a *Tim Berners-Lee*, který přišel s myšlenkou hypertextu ve švýcarském institutu pro jaderný výzkum CERN, nicméně ji oživil až v roce 1989. Také publikoval návrh vývoje www. Již v listopadu 1990 předvedl CERN první prototyp WWW serveru, jenž pojmenoval **httpd**. S příchodem hypertextu a WWW serveru se také zrodil první webový prohlížeč s názvem **WorldWideWeb** (viz kapitola 3.1).

Roku 1992 byly položeny základy grafického prohlížeče **Mosaic**, který díky svému grafickému uživatelskému prostředí a multimediálním možnostem nastartoval internetový boom a tím ho rozšířil do komerční sféry a k běžným lidem. Ten nastal v USA roku 1993, např. byl v té době připojen Bílý dům. Následovalo vyvinutí WWW standardu.

Instituce, která se stará o rozvoj WWW služby, se jmenuje **W3C Consorcium** a funguje od poliviny roku 1994. Konsorcium sdružuje lidi, kteří se podíleli v ústavu CERN na prvních krůčcích fenoménu jménem WWW, techniky z MIT a z francouzského institutu INRIA. Ředitelem konsorcia není nikdo jiný, než **tvůrce WWW Tim Berners-Lee**.

2.3 Statistika připojení k Internetu

Po dlouhou dobu byl Internet sítí pro malý počet uživatelů. V *roce 1983* bylo připojeno teprve *1000 počítačů*. O čtyři roky později už bylo propojeno více než *27000 počítačů*. V té době však šlo pouze o sdílení souborů a komunikaci.V *říjnu 1992* to bylo již *více než milion* počítačů. O půl roku později to bylo o půl milionu více. S nástupem WWW se však začal počet rapidně zvyšovat, v roce 1993 bylo v provozu *zhruba 50 WWW serverů*.

Rok 1994 znamenal jednak komercionalizaci Internetu, jednak vyvinutí prohlížeče Netscape Navigator. O dva roky později (*1996*) se počet uživatelů zvýšil na neuvěřitelných *55 milionů*, za další čtyři (*2000*) *250 milionů uživatelů*. V roce *2003* byl počet uživatelů vyčíslen na *600 milionů* a **v současné době** udává statistika **přes jednu miliardu uživatelů** Internetu.

2.4 Shrnutí

Tato kapitola si kladla za cíl stručně seznámit čtenáře s dosavadním vývojem Internetu a prohlížečů obecně. Následující kapitola rozebere dva již zmiňované historické prohlížeče a pokusí se rozčlenit a představit prohlížeče, které byly a jsou vyvinuty a používány.

3 Webové prohlížeče

Kapitola o prohlížečích Internetu představuje přehled všech významnějších prohlížečů, které kdy byly používány. Rozhodl jsem se je rozčlenit do několika kategorií podle [6]:

- Historické prohlížeče
  - Historicky významné prohlížeče
  - Ostatní rané prohlížeče
- Grafické prohlížeče
  - Založené na jádrech Trident i Gecko
  - Založené na jádru Trident
  - Založené na jádru Gecko
  - Založené na jádrech KHTML a WebKit
  - Založené na Presto
  - Pro Java platformu
  - Speciální (současné)
  - Speciální (ukončené)
  - Ostatní
  - Mobilní
- Textové prohlížeče

3.1 Historické prohlížeče

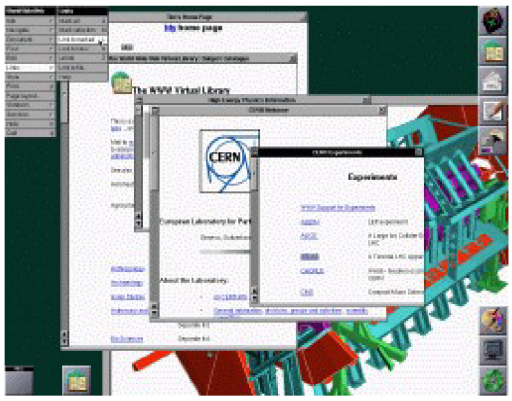
Z prvních prohlížečů není možné nezmínit *WorldWideWeb*, první prohlížeč pro hypertext. Také podrobněji popíši *Mosaic*, browser odpovědný za internetový boom. Mezi historicky významné patří bezpochyby i Internet Explorer, Mozilla Firefox, Opera, či Netscape Navigator, některé z nich budou zmíněny v jiných kapitolách. Další rané prohlížeče jsou tyto:

- Erwise, ViolaWWW
- AMosaic
- Arena
- Cello
- CyberDog
- Grail
- IBM Web Explorer
- Lynx
- MacWeb
- MidasWWW
- Oracle PowerBrowser
- SlipKnot
- WebRouser

3.1.1 WorldWideWeb

První webový prohlížeč byl představen 26. února 1991, běžel na platformě NeXTSTEP. Zároveň sloužil jako WYSIWYG HTML editor (uživatel tvoří stránky přímo tak, jak mají vypadat, nikoliv html značkami). Jako první nevyužíval pouze **File Transfer Protocol**, ale i **HyperText Transfer Protocol**. Později byl přejmenován na *Nexus* kvůli pojmu internetové sítě World Wide Web. Obsahoval navigační menu s navigačními tlačítky „back“, „next“ a „previous“. Poslední dvě sloužily k prohlížení seznamu odkazů. Jejich funkce se dá popsat jako „běž zpět a nahraj dokument v dalším (předchozím) odkazu“. Kromě klasických menu měl zajímavé, v dnešních prohlížečích nepodporované vlastnosti. Zajímavá byla položka „style“, která dovolila načíst šablonu stylu, která se má na dokument aplikovat, případně prvky jako seznam ručně upravovat.

Jelikož byl WorldWideWeb zároveň prohlížeč i editor, musely se nějak označit upravené, ale neuložené dokumenty. NeXT prostředí to řešilo elegantně. V pravém horním rohu mělo tlačítko s „X“ tak jak ho známe dnes z Windows (které ho dle [3] právě odtud zkopírovalo), které se při editaci dokumentu porušilo tím, že symbolu zmizel střed. Screenshot prohlížeče-editoru ukazuje Kresba 3.1.



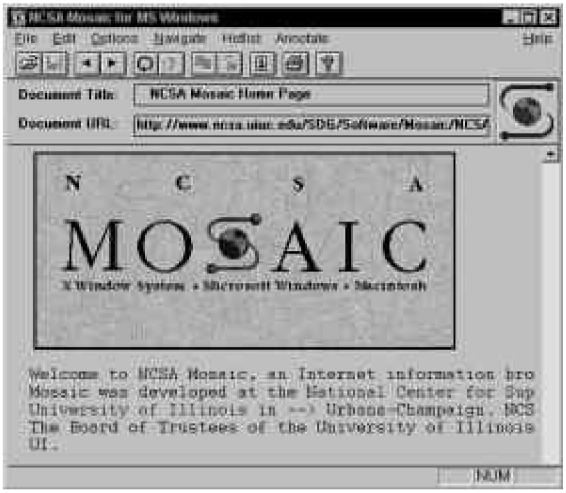
Kresba 3.1: Prohlížeč WorldWideWeb

- **Protokoly:** HTTP, FTP, NNTP
- **Webové technologie:** ne (standardy ještě neexistovaly)

3.1.2 Mosaic

Počátek vývoje se datuje do roku 1992. Tento grafický prohlížeč, který byl vyvíjen na půdě *NCSA* (National Center for Supercomputing Applications) byl zdarma uvolněn 22. dubna 1993. Stal se prvním široce oblíbeným webovým prohlížečem. V podstatě je jedním z důležitých faktorů, které zapříčinily masivní rozšíření Internetu v devadesátých letech. Společně s browserem Mosaic vznikla společnost *Mosaic Communications*.

Prohlížeč fungoval jak na Unixu, tak o pár měsíců později na Windows a Mac. Také byl portován na Amigu pod názvem *Amosaic*. Kromě prohlížení *World Wide Webu* byl i *Gopher* klientem (vylepšená forma anonymního FTP). Jako první multimediální prohlížeč uměl procházet web, prohlížet obrázky, přehrávat zvuky, či video skrz grafické uživatelské rozhraní. Na rozdíl od *WorldWideWeb*u (kapitola 3.1) uměl pracovat s formuláři a tedy odesílat příkaz HTTP PUT (od alfa verze Mosaic 2.0), dále měl podporu HTML 3 tabulek. Vzhled Mosaic se velmi podobal dnešním prohlížečům (Kresba 3.2).



Kresba 3.2: Mosaic

Vývoj byl oficiálně ukončen 7. ledna 1997, resp. se pokračovalo ve vývoji pod názvem Netscape Navigator. Spolu s browserem se přejmenovala společnost *Mosaic Communications* na **Netscape Communications**. Také se z Mosaic vyvinul Internet Explorer.

Klady	*Nedostatky*
✓ Správce záložek	× Správa hesel × Správce stahování × Správa formulářů × Kontrola pravopisu × Panely × Inkrementální hledání v textu × Ad filtrování

- **Protokoly:** HTTP (částečně), FTP, NNTP, Gopher
- **Webové technologie:** ne (standardy ještě neexistovaly)

3.1.3 Arena a Amaya

Nejprve si přiblížíme *Arenu.* Byl vyvinut **W3C** jako podpora testování pro *HTML 3* a *CSS* (*Cascading Style Sheets*). Platforma, na které fungovala, je X11/Unix. Práce na tomto prohlížeči byla zastavena kvůli přechodu na *Amayu*, která se stala novým testovacím subjektem.

Amaya je opensource browser vytvořený ve francouzském *Národním centru věděckého výzkumu*, později adoptovaný **W3C**. Je to přímý potomek *WYSIWYG SGML* editoru **Grif**. Původně byl navržen jako strukturovaný textový editor, později rozšířen na HTML/CSS editor. Zobrazuje otevřené formáty obrázků jako PNG, nebo SVG.

I dnes se používá jako podpora pro testování nových technologií před tím, než jsou nasazeny do majoritních prohlížečů. Podporuje Unix, Windows i Mac OS X.

Klady	*Nedostatky*
✓ Správa hesel	× Správce záložek
✓ Kontrola pravopisu	× Správce stahování
✓ Panely	× Správa formulářů
✓ CSS 2.1	× Ad filtrování
✓ XHMTL 1.1	× Inkrementální hledání v textu
	× Rámce

- **Protokoly**: HTTP, E-mail
- **Webové technologie:** CSS 2.1, XHTML 1.1, MathML

3.1.4 Ostatní historické prohlížeče

Z dalších browserů stojí za zmínku **Cello** (1993), prohlížeč pro Windows 3.1, dále **CyberDog** pro MacOS (1996), IBM **WebExplorer** pro OS/2 (1994). Pro Mac OS byl vyvinut MacWeb (1994-1996), který vynikal svými nízkými požadavky na hardware. Stačilo mu 680kB diskového prostoru a méně než 1MB operační paměti. Zajímavou variantou byl **Oracle PowerBrowser** (1996), který umožňoval režim webového serveru, čili uživatelé si mohli svoje stránky hostovat na svých počítačích. Posledním z historických, zde zmíněných prohlížečů je **WebRouser** (1995) podporující pluginy, client-side obrázkové mapy, stránkou definované menu stromy a stránkou definované tlačítkové lišty.

3.2 Renderovací jádra

Nejprve si ujasníme základy, na kterých jsou moderní grafické prohlížeče postaveny. Je to politika rozmístění objektů a jejich vykreslení. Řeč je o renderovacích jádrech. Je to software, který vezme obsah webové stránky, případně další formátovací informace jako CSS a zobrazí výsledný naformátovaný obsah stránky do okna prohlížeče. Mezi nejznámější patří jistě *Trident*, *Gecko* a *Presto*.

3.2.1 Trident

Trident (také známý jako MSHTML) je renderovací jádro použité v aplikaci Microsoft Internet Explorer. První verze byla představena roku 1997 a používá se ve vylepšené verzi dodnes. Je to softwarová komponenta založená na COM technologi (tzn. poskytuje COM rozhraní pro jednotné zacházení s komponentami). To znamená, že má podporu pouze na platformě Windows. Je navrženo tak, aby ho mohli programátoři využívat ve svých aplikacích. Obsahuje funkce pro vytvoření libovolného dialogového okna, nebo okna vlastností, které zobrazí HTML stránku. Také má desítky rozhraní pro objekty na stránce i pro události, neboli pro přístup k DHTML objektovému modelu. Všechny jsou odvozeny z rozhraní IDispatch. Základem je rozhraní IHtmlDocument2 pro celý dokument. Detailní popis MSHTML je na [8].

Podporuje následující standardy:

- XHTML 1.1
- XHTML 1.0 (Strict, Transitional, Frameset)
- HTML 4.1 (Strict, Transitional, Frameset)
- HTML 4.0 (Strict, Transitional, Frameset)
- XML 1.0
- CSS 1, 2.1 (částečně)
- DOM Level 1, 2 (málo)
- Javascript Edition 3 (JScript)
- JPEG, GIF, PNG (částečně – ne průhlednost pozadí)

Podporuje všechny HTML elementy, funkce elementu object je ale chybná. Další zvláštností je počítání rozměrů bloků, které není podle standardu (alespoň do verze 6.0).

3.2.2 Gecko

Toto jádro je open source a je napsáno v C++ pod MPL/GPL/LGPL tri-licenci. Je spravován a vyvíjen organizací *Mozilla Foundation*. Je navrženo jako podpora otevřených internetových standardů. Stejně jako Trident (kapitola 3.2.1) poskytuje API pro vývoj rozličných internetových aplikací jako webové prohlížeče, prezentace aj. Primárně je určen aplikacím, které jsou odvozeny z *Mozilly* jako *Firefox, Camino, SeaMonkey, K-Meleon a Netscape.*

Podpora standardů:

- XHTML 1.1, HTML 4.0
- XML 1.0
- CSS Level 1 (částečně CSS 2 a CSS 3)
- JavaScript 1.7 (SpiderMonkey)
- DOM Level 1 a 2 (částečně Level 3)
- MathML, XML 1.0, XForms, SVG (částečně), RDF

Dokumentace k jádru se nachází na [9]. Dokumenty neobsahující element DOCTYPE, nebo mají uveden starší typ fungují v tzv. quirk módu, které emuluje nestandardní chování Netscape Navigatoru. Také podporuje některé nestandardní tagy Internet Exploreru, jako <marquee> a vlastnost document.all, ovšem s omezenou funkčností.

Funguje jak na windowsech, tak na systémech unix, bsd, či linux a také na Mac OS X.

3.2.3 Presto

Vyvinuto *Opera Software* slouží jako jádro pro prohlížeč *Opera Browser*. Nicméně ho využívá třeba i HTML editor Macromedia Dreamweaver. Nicméně je dostupné pouze jako součást Opery, či spřízněných produktů. Zdrojový kód, nebo DLL knihovny nejsou volně stažitelné.

Vyznačuje se vysokou kompatibilitou se standardy:

- XHTML 1.1, HTML 4.01
- XML 1.1
- CSS Level 1, 2, 3 (největší podpora ve srovnání s ostatními jádry)
- DOM Level 1, 2 (většinu), 3 (minimálně)
- Javascript 1.5 (linear b/futhark)
- JPEG, GIF, PNG, SVG (největší podpora ve srovnání s ostatními jádry)

Stejně jako Gecko (kapitola 3.2.2) je multiplatformní.

3.2.4 KHTML

Jádro bylo vyvinuto v rámci projektu **KDE**. Používají ho prohlížeče *Konqueror* a *Safari* od Apple. Je šířen pod LGPL licencí.

Podporovány jsou následující standardy:

- HTML 4.01
- XML 1.0 i 1.1 částečně
- CSS Level 1, 2.1 (částečně), 3 (částečně)
- DOM Level 1, 2 (částečně), 3 (minimálně)
- ECMA-262/Javascript 1.5 (KJS)
- PNG, MNG, JNG, JPEG, GIF
- SVG částečně

I KHTML je multiplatformní záležitostí.

3.3 Grafické prohlížeče

V této části bude popsáno několik nejdůležitějších moderních prohlížečů. Začneme na jádru *Trident (kapitola 3.2.1)* a *Microsoft Internet Explorer*. Avšak dříve se podíváme na současnou statistiku v používání prohlížečů z listopadu 2007 [10]:

- Internet Explorer - 77.37%
- Mozilla Firefox - 16.01%
- Safari - 5.14%
- Opera - 0.65%
- Netscape - 0.60%
- Mozilla - 0.09%
- Ostatní - 0.16%

3.3.1 Microsoft Internet Explorer

Až do verze Internet Exploreru (dále jen IE) 4.0 hrály předchozí verze na internetovém poli pouze druhořadou úlohu. Jako mnohem propracovanější se jevil Netscape, zvláště od verze 2.0, ve které představil Javascript, rámce a plug-in technologii. Až IE 4.0 (integrován ve Windows 98) dosáhl úrovně, která Netscape v mnoha ohledech předčila a situace v tzv. „válce browserů“ se obrátila.

Má několik specifických rysů spjatých s operačním systémem. Za prvé nelze odinstalovat, za druhé není možné provést downgrade (přechod na dřívější verzi) a za třetí není ani možné mít současně nainstalováno několik verzí.

Využívá renderovací jádro *Trident* (kapitola 3.2.1) a v současné době je dostupná verze 7.0. Funguje pouze na operačním systému Windows. Nicméně existuje i varianta pro Macintosh a v letech 1997 až 2003 byl díky investici 150 milionů dolarů do Apple výchozím prohlížečem na MacOS.

Logo prohlížeče ve verzi 7.0 ukazuje následující obrázek:

Kresba 3.3: Internet Explorer 7 logo

Klady	*Nedostatky*
✓ Správa záložek	× Správce stahování
✓ Správa hesel	× Kontrola pravopisu
✓ Panely (až IE 7.0)	× Inkrementální hledání v textu
✓ Správa formulářů	
✓ Ad filtrování	
✓ Pop-up blokování	

3.3.2 Mozilla Firefox

Mozilla Firefox je projekt vycházející ze souboru Mozilla aplikací spravovaných *Mozilla Corporation*. Používá open-source renderovací jádro Gecko (kapitola 3.2.2).

Původní název prohlížeče bylo *Phoenix*, což byla verze 0.1-0.5 s jádrem Gecko 1.2 a 1.3. První release byl uvolněn 23 září 2002. Od verze 0.6 a Gecko 1.5 se browser přejmenoval na Mozilla Firebird. Od verze 0.8 s jádrem ve verzi 1.6 byl konečně nazván Mozilla Firefox a oficiální release verze 1.0 byla vypuštěna dne 9. listopadu 2004. V současné době je aktuální verze 2.0.0.11 a testovací verze Mozilla Firefox 3.0 Beta 2.

Kromě některých operačních systémů pro MacIntosh podporuje Windowsové, Unixové, Linuxové a některé Mac operační systémy. Logo prohlížeče znázorňuje Kresba 3.4.

Kresba 3.4: Mozilla Firefox logo

Mezi jeho přednosti patří bezesporu lepší dodržování standardů a zároveň možnost přizpůsobit se nestandardním, nutno říci někdy chybným, vlastnostem Internet Exloreru, a to zejména v oblasti stylů. Dovoluje psát CSS vlastnosti s podtržítkem na začátku a tím přepisovat ty bez podtržítka, čímž dovoluje programátorům psát stránky i pro chybné vlastnosti Internet Exloreru.

Další výhodou je obrovské množství doplňků, kterými lze Firefox rozšířit. Z nejznámějších uvedu ***Mouse Gestures*** – doplněk umožňující ovládání například navigace, oken, panelů a velikosti písma pomocí gest myši. Dovoluje také uživatelsky definovaná gesta. Dalším velmi užitečným doplňkem je ***Foxmarks Bookmarks Synchronizer***, který automaticky odesílá záložky na internetový server, který může uživatel sám definovat a zpřístupňuje tak záložky odkudkoliv.

Následující tabulka shrnuje klady a zápory Firefoxu:

Klady	*Nedostatky*
✓ Správa záložek	× Zoom stránek (bez Mouse Gestures)
✓ Správa hesel	× Web Forms 2.0
✓ Správa stahování	
✓ Panely	
✓ Správa formulářů	
✓ Kontrola pravopisu	
✓ Vyhledávací toolbary	
✓ Inkrementální hledání v textu	
✓ Ad filtrování	
✓ Pop-up blokování	

4 MIME struktura

Podle dokumentu RFC-822 je MIME (Multipurpose Internet Mail Extension) struktura pro definici datových typů zpráv, které lze přenášet protokolem *smtp*. Nicméně je tato struktura používána šířeji, a to zejména v protokolu *HTTP (*kapitola 5*)*, což je důvod, proč byla zařazena do tohoto textu.

Dává možnost přenášet kromě textových také binární data. Definuje jednak jaké typy souborů mohou být přenášeny a způsob jejich kódování.

4.1 Typy zpráv MIME

Počet typů zpráv se postupně rozšiřuje, nicméně několik základních typů existuje dlouhodobě a s vysokou pravděpodobností tomu tak bude i nadále. Jsou to typy:

- **application** – aplikace
- **audio** – zvukové zprávy
- **video** – video zprávy
- **text** – obyčejné textové zprávy
- **image** – obrázky
- **message** – zprávy elektronické pošty
- **model** – jiné grafické a vícerozměrné zprávy
- **multipart** – zpráva složená z více jednoduchých typů

Každý z typů má libovolný počet podtypů a zapisuje se v notaci ***typ/podtyp***.

4.1.1 Typ application

Formát dat u typu application je závislý na formátu, který definuje příslušná aplikace, jako např. program z kolekce Microsoft Office. Univerzálním podtypem je **octet-stream** sdělujícím, že s daty není žádná aplikace asociována.

4.1.2 Typ audio

Typ basic obsahuje podtypy **basic** (základní formát), **midi, mpeg, wav** (pro odpovídající formáty) a **x-pn-realaudio**, která definuje formát pro přenos internetového rozhlasu.

4.1.3 Typ image

Obrázkový typ podporuje podtypy významných formátů jako **bmp, gif, jpg, tiff,** nebo **jpeg**.

4.1.4 Typ text

Tento typ popisuje přenos textové informace a jeho podtypy jsou použity pro webovou technologii. Základní podtypy jsou **plain** (prostý text), **rtf** (formátovaný text), **HTML, XML, CSS** (použití protokolem HTTP (kapitola 5)).

Textový typ obsahuje také užitečnou informací o kódování textu a zapisuje se ve formátu: ***typ/podtyp; charset=.....***.

4.1.5 Typ multipart

Strukturovaný typ pro data skládající se z více typů. Má hlavní podtypy **alternative** (příjemce si může z dostupných typů vybrat) a **mixed** (zpráva složená z různých typů).

5 HTTP - HyperText Transfer Protocol

Protokol http je momentálně nejpoužívanějším protokolem v Internetu pro přenos dat mezi klientem a serverem a do značné míry nahrazuje protokol pro přenos souborů FTP. Je to poměrně malý protokol, který pracuje s adresami URL a má pouze několil příkazů, nazývaných metody. Jedním z rozdílů od FTP je skutečnost, že je anonymní (tzn. bezstavový). Používá standardu MIME (kapitola 4), čili může zpracovávat a být rozšiřován o různé druhy dat.

Je protokolem typu klient/server a pracuje na principu požadavek/odpověď. Datový přenos se skládá z dat a metadat. Data představují tělo zprávy a metadata její hlavičku. Nejnovější verze je 1.1.

5.1 Syntaxe HTTP

Protokol http je textový a v záhlaví jsou pole hlavičky zprávy ve formátu ***jméno pole : hodnota pole*** ukončené znaky ***CR*** a ***LF*** pro konec řádku. Polí záhlaví může být libovolný počet. Za nimi je prázdný řádek a vlastní zpráva podle protokolu MIME.

5.2 Požadavky - metody

Jak již bylo uvedeno výše, protokol pracuje na principu požadavek/odpověď, kde klient zadává požadavky a server na ně odpovídá. Metody říkají, co s daty na daný požadavek provést. Jsou umístěny v požadavku na prvním řádku nazývaném *Request-Line*. Mezi hlavní metody patří **„GET“, „PUT“, „POST“, „HEAD“, „OPTIONS“, „DELETE“, „TRACE“, „CONNECT“**. Metody „GET“ a „HEAD“ jsou povinné pro každou implementaci protokolu, ostatní jsou volitelné. Podívejme se na některé metody detailněji.

5.2.1 Metoda GET

Nejčastěji používaná metoda pro komunikaci. Server, který obdrží GET načte zdrojová data dle zvoleného URL a pošle je zpět klientovi.

Syntaxe příkazu je **GET <url> HTTP/1.0**.

5.2.2 Metoda HEAD

Metoda HEAD se velmi podobá metodě GET, ale místo zdrojového kódu stránky vrací pouze metainformace o této adrese. Používá se pro test dostupnosti, případně změny na dané URL adrese.

Syntaxe příkazu je **HEAD <url> HTTP/1.0**.

5.2.3 Metoda POST

POST umožňuje přenos dat směrem od klienta k serveru. Typicky se používá pro formuláře. Délka odesílaných dat není omezena. I příkazem GET se dá odesílat informace na server a to tím, že se připojí k URL. Toto řešení je pouze pro jednoduchá, krátká a nepříliš podstatná data, neboť délka URL adresy je omezena.

5.2.4 Metoda PUT

Příkaz PUT taktéž odesílá data z klienta na server. Není však již mezi povinně implementovanými příkazy.

5.2.5 Metoda DELETE

Tato metoda požádá server o odstranění souboru na zadané URL. Již z podstaty příkazu není běžně podporován kvůli zřejmým bezpečnostním rizikům.

5.3 Odpovědi

Odpověď serveru na požadavek klienta se skládá z řádku *Status-Line*, následovaná poli hlavičky oddělených koncem řádku tvořených posloupností znaků ***CR*** a ***LF***. Následuje (stejně jako u požadavku) prázdý řadek a za ním tělo zprávy.

Status-Line má následující syntax:

<HTTP verze> <status kód číselně> <status kód textově> CRLF.

5.3.1 Stavové kódy

Stavové kódy se dělí do několika kategorií, jejichž význam je popsán následující tabulkou:

Kategorie stavových kódů	*Čísla stavových kódů*	*Popis*
informační	100-199	Aplikací definované zprávy
úspěšné	200-299	Úspěšné zpracování požadavku
přesměrování	300-399	Klient musí vykonat další činnost pro další zpracování požadavku. Klientem se rozumí prohlížeč, nikoliv uživatel.
chyba klienta	400-499	Problémy na straně klienta
chyba serveru	500-599	Problémy na straně serveru

Každá kategorie má jeden nebo více nejvýznamnějších stavových kódů, uvedeme si některé nejdůležitější dle standardu protokolu HTTP:

Kat.	*Číslo kódu*	*Název kódu*	*Popis*
Informační	100	Continue	Přijata byla pouze část požadavku
	101	Switching Protocols	Server zaměňuje protokoly
Úspěšné	200	OK	Bezchybná operace
	201	Created	Požadavek typu POST byl úspěšně splněn
	202	Accepted	Byl přijat asynchronní požadavek. Není však zaručeno, že odpovídající činnost byla provedena
	204	No Content	Požadavek byl úspěšný, ale klient nemá co zobrazit
Přesměrování	300	Multiple Choices	Zdroj je možné získat z několika míst. Doporučení serveru je v poli `Location`
	301	Moved permanently	Zde určuje pole `Location` novou URL, kam se zdroj natrvalo přesunul. Všechny další odkazy na tento zdroj musí používat novou URL
	302	Moved Temporarily	Pole `Location` opět určuje novou URL, ale zdroj se přesunul dočasně. Všechny další odkazy na tento zdroj mohou používat dosavadní URL
	304	Not Modified	Podmíněný GET se správně zpracoval. Dokument však do doby definované polem `If-Modified-Since` nebyl modifikován.
Chyba klienta	400	Bad Request	Server nerozumí požadavku, požaduje opakování
	401	Unauthorized	Pokud byl požadavek anonymní, musí být autentizován, jinak je zamítnut
	403	Forbidden	Neoprávněný přístup ke zdroji
	404	Not Found	Server nenašel požadovanou URL
Chyba serveru	500	Internal Server Error	Na serveru došlo k neočekávané chybě
	501	Not Implemented	Tento požadavek server neimplementuje
	502	Bad Gateway	Proxy server, nebo brána obdržely od serveru neplatnou odpověď
	503	Service Unavailable	Server dočasně nemůže, nebo nechce zpracovat požadavek

5.4 Shrnutí

Kapitola o protokolu HTTP popsala jeho nejdůležitější aspekty, jakým způsobem probíhá komunikace klienta (resp. prohlížeče) a serveru, možné zprávy klienta a stavové kódy odpovědí. Následující kapitola se bude zabývat jazykem HTML.

6 HTML – HyperText Markup Language

Jazyk HTML je značkovacím jazykem. Značkovací jazyk (markup language) je speciální typ programovacího jazyka, který slouží k popisu dokumentů. Vychází z metajazyka SGML, je jakousi jeho aplikací v praxi. Popisuje hypertextové dokumenty, které musí mít příponu **.htm,** nebo **.html.** Ty mimo pouhý popis značkováním HTML mohou obsahovat další formální jazyky v rámci speciálních značek, a to třeba JavaScript, nebo CSS. Výhodou tvorby stránek v HTML je, že je lze programovat v běžném textovém editoru.

6.1 DTD pro HTML 4.01

Existují tři oficiální varianty definice. První je striktní, kde W3C postupně stahuje prezentační atributy a prvky. Doporučuje se ho používat co nejvíce. Druhou variantou je přechodné DTD s podporou prezetačních prvků a atributů. Konečně třetí varianta zajišťuje podporu rámců (Frameset).

Definice se píše na první řádek HTML dokumentu a má následující tvar:

<!DOCTYPE HTML PUBLIC <formal public identifier> <URL definice DTD> >

V tomto pořádí uvedeme také konkrétní případy:

- DTD Strict
  - `"-//W3C//DTD HTML 4.01//EN"`
  - `http://www.w3.org/TR/html4/strict.dtd`
- DTD Transitional
  - `"-//W3C//DTD HTML 4.01 Transitional//EN"`
  - http://www.w3.org/TR/1999/REC-html401-19991224/loose.dtd
- DTD Frameset
  - `"-//W3C//DTD HTML 4.01 Frameset//EN"`
  - http://www.w3.org/TR/1999/REC-html401-19991224/frameset.dtd

Mimo DTD existují i jiné definice parametrových entit, které reprezentují formáty jazyků. Říká se jim souhrně importovaná jména. Jejich popis však je nad rámec tohoto textu. Některé jsou popsány v protokolu o MIME (kapitola 4), jako *ContentType*, nebo *Charset*, jiné, jako např. *LanguageCode* jsou popsány RFC dokumenty.

6.2 HTML elementy, tagy a atributy

Tagy se používají k vyznačení HTML elementů. Jsou uzavřeny mezi znaky < a >. Obvykle je použito párových značek, jako např. **<table>** a **</table>**, startovní tag je povinný, nicméně ukončující tag *není vždy* povinný. Pak se značky nazývají *nepárové*. Tagy nejsou *case-sensitive* (citlivé na velkost písmen). Dále mohou mít **atributy**, které poskytují elementům přídavné informace nejčastěji použité na formátování. Specifikují se ve strartovním tagu elementu, jako oddělovač slouží mezera. Tag může mít obecně více atributů najednou. Atribut se skládá z dvojice jméno/hodnota a zapisuje se ve tvaru jméno="hodnota". Doporučuje se jméno psané malými písmeny. Existují standardní atributy, které se dají aplikovat na každý element (více v kapitole 6.9), dále existují atributy událostí (kapitola 6.10).

HTML element se skládá z povinného startovního tagu, který může obsahovat atributy, dále ne vždy povinného těla, neboli obsahu elementu a ne vždy povinného ukončovacího tagu. Příklad HTML elementu:

```
<i>Text psaný kurzívou</i>
```

6.3 Speciální značky

Jelikož je některé znaky, jako například ohraničující závorky tagů, nemožné obyčejným způsobem zapsat v tělu elementu, používají se zástupné značky, které se vyjadřují jako znak ampérsand (&) následovaný buď mříží (#), číslem a středníkem, nebo jménem a středníkem Mezi nejpoužívanější patří Html Latin-1, Html symbolové entity a HtmlSpecial.

6.3.1 Html Latin-1

Tato znaková sada je také označena jako ***ISO-8859-1***. Obsahuje 255 znaků, z nichž prvních 127 tvoří originální sedmibitová ASCII tabulka. Z ní uvedeme ty znaky, které mají svoje jméno:

Značka		**Jméno**		**Číslo**
" (uvozovky)	-	&quot;	-	&#34;
& (ampérsand)	-	&amp;	-	&#38;
' (apostrof)	-	' (ne IE)	-	&#39;
< (menší než)	-	&lt;	-	&#60;
> (větší než)	-	&gt;	-	&#62;

Ze znaků s vyššími čísly vybereme některé důležité:

nedělitelná mezera	-		-	
© (copyright)	-	&copy;	-	&#169;
® (reg. obchodní zn.)	-	&reg;	-	&#174;

6.3.2 HTML Symbolové entity

Symbolové entity se dělí do třech kategorií. Matematické symboly jako suma, odmocnina, nebo nekonečno (celkem 38 symbolů), dále symboly řecké abecedy (55 znaků) a ostatní symboly (např. znak promile, euro aj. - 50 znaků).

6.4 Struktura dokumentu

První řádek dokumentu tvoří značka **!doctype**. V HTML není povinná, nicméně její používání se doporučuje. Specifikuje DTD (kapitola 6.1). Značka není párová, tzn. nemá ukončující tag.

Zbytek dokumentu tvoří stromovou strukturu. Základní strom vypadá následovně:

- <html>
    - <head>
    - </head>
    - <body>
    - </body>
- </html>

Značka <**html**> je párová, povinná a je kořenovým prvkem stromové struktury. Její první potomek je značka <**head**>, která se v textu nezobrazuje, nicméně je povinná a může obsahovat tagy <**title**>, <**meta**>, <**base**>, <**link**>, <**script**> a <**style**> (kapitola 6.6.1).

Za hlavičkou následuje tělo dokumentu <**body**>, párová značka, jehož definice není povinná. Viditelný obsah stránky tvoří právě obsah elementu s těmito tagy.

Alternativou k tělu <**body**> jsou rámce <**frameset**>, ovšem pouze v definici *DTD Frameset*. Umožňuje rozdělit stránku na libovolný počet rámců, jejichž obsah jsou další dokumenty.

6.5 Tok textu

HTML elementy se dělí do dvou kategorií. *Řádkové elementy* (také nazývané *inline*) nezpůsobují změnu toku textu, další elementy mohou být na stejném řádku před i po něm. *Blokové elementy* naproti tomu způsobují změnu toku textu, tzn. před a za elementem se zalomí řádek. Nutno říci, že zde se každý prohlížeč chová mírně odlišně. Správné zobrazení elementů na svých místech je velkou nepříjemností každého vývojáře, resp. grafika webových stránek. Oba druhy značek mají tu vlastnost, že obsah uvnitř se přizpůsobuje velikosti okna, tzn. nevejde-li se slovo na konec řádku, automaticky se přemístí na další. Pokud obsahuje element slovo, které je samo o sobě delší než řádek, nezalamuje se.

V HTML dokumentu nemá vliv odřádkovávání ve zdrojovém kódu. Stejně tak nemá vliv na text počet mezer mezi slovy, zobrazí se pouze jedna (výjimkou je nedělitelná mezera ** **).

6.6 Rozdělení elementů podle funkce

Nyní se podíváme na tagy z hlediska jejich významu. Budou to tagy v hlavičce, základní tagy, formátovací, odkazové, rámce, tabulky, seznamy, formuláře a obrázky.

6.6.1 Tagy v hlavičce dokumentu

- **<title>** - Párový tag, je titulkem stránky. Může obsahovat pouze obyčejný text bez dalších tagů. Zobrazuje se v záhlaví okna jako nadpis aplikace a jako nadpis ve vyhledávačích.
- **<base>** - Nepárový tag, určuje základní URL pro ostatní odkazy na stránce.
  - ***href*** – povinný atribut s hodnotou popisující základní URL.
  - ***target*** – nepovinný atribut. Definuje okno, kam se budou defaultně otevírat odkazy. Hodnoty *_blank* (nové okno), *_parent* (rodičovský frameset), *_self* (aktuální frame), *_top* (celé aktuální okno). O rámcích bude pojednáno v kapitole 6.6.6.
- **<link>** - Nepárový tag, propojení s jiným souborem, nejčastěji CSS. Všechny atributy volitelné.
  - ***rel*** – vztah mezi aktuálním a cíleným dokumentem. Nejčastěji *stylesheet* (CSS), příp. *alternate* aj.
  - ***href*** – odkaz na URL cílového dokumentu.
  - ***type*** – MIME typ souboru (viz kapitola 4)
  - ***media*** – zařízení pro zobrazení dokumentu. *Screen, tty, tv,…*
  - ***charset, hreflang, rev, target***
- **<meta>** - Nepárový tag s informacemi o dokumentu. Obsahuje buď atribut *name*, nebo *http-equiv*, ne oba najednou. S nimi asociované metainformace jsou v atributu *content*.
  - ***name*** – uživatelské informace jako *author*, *keywords* aj.
  - ***http-equiv*** – ekvivalent http hlavičky. Hodnoty *content-type* (MIME specifikace + kódování), nebo *refresh* (přesměrování po čase), *expires* a *set-cookie*.
  - ***content*** – povinný atribut, definuje metainformace spojené s *name*, nebo *http-equiv*.

6.6.2 Základní tagy

Základní tagy jsou také blokové (pokud se zobrazují).

- **<!DOCTYPE>** - viz kapitoly 6.1, 6.4
- **<html>** - viz kapitola 6.4
- **<body>** - viz kapitola 6.4
- **<h1>** - **<h6>** - Párová značka. Nadpisy jsou v podstatě zvláštní druhy odstavců. Zobrazují se automaticky s vertikálními mezerami, tučně s následujícími velikostmi písma:
  - **h1** – 24px, **h2** – 18-19px, **h3** – 16px, **h4** – 13px, **h5** – 10px, **h6** – 9px

- **<p>** - Nepovinně párová značka, znamená odstavec. Obsahuje před i za sebou prázdný řádek. U dvou odstavců za sebou se mezery nesčítají.
- **
** - Nepárová značka. Způsobuje zalomení stránky, resp. přechod na nový řádek.
- **<hr>** - Nepárová značka. Vodorovná čára, výchozí barva šedá. Všechny volitelné atributy jsou podle normy zastaralé
    - *align, noshade, size, width*
- **<!-- ... -->** - Nepárová značka, komentář zdrojového kódu. Prohlížečem ignorován.

6.6.3 Formátování písma

Všechny formátovací značky jsou párové.

- **<b>, <i>, <u>** - Tučný text, kurzíva a podtržení textu (poslední je zastaralé).
- **<font>** - Zastaralé. Definuje font, jeho velikost a barvu.
- **, ** - Zvýraznění textu – první tučně, druhé kurzívou.
- **<small>. <big>** - Zmenšení (zvětšení) textu o jeden stupeň.
- **<sup>, <sub>** - Horní (dolní) index.
- **<bdo>** - Směr textu. Má atribut:
    - ***dir*** – hodnoty *rtl* (zprava doleva) a *ltr* (zleva doprava)
- **<acronym>, <abbr>** - Vysvětlení zkratky. Zobrazení tečkovaným podtržením.
- **<address>** - Blokový element pro zobrazení adresy, obsah obyčejně kurzívou.
- **<q>** - Krátká citace. Firefox a Opera dělají kolem textu uvozovky, IE ne.
    - ***cite*** – nepovinný atribut definuje zdroj citace
- **<cite>** - Inline element. Krátká citace, zobrazuje se kurzívou.
- **<blockquote>** - Blokový element. Dlouhá citace. Text je odsazen zleva i zprava o 40px. Pokud se tag opakuje vícekrát, odsazení se zvětšuje.
    - ***cite*** – nepovinný atribut definuje zdroj citace
- **<ins>, <del>** - Vložený (smazaný text). Vložený se zobrazí podtrhnutím, smazaný přeškrtnutím. Mají 2 stejné volitelné atributy:
    - ***cite*** – URL na vysvětlení, proč byl text vložen (smazán)
    - ***datetime*** – datum a čas vložení (smazání) ve tvaru *RRRRMMDD*.
- **<s>, <strike>** - Zastaralé. Přeškrtnutí textu.
- ** ** - Zastaralé. Horizontální vycentrování textu.

6.6.4 Výstupní tagy

Tagy pro výstup jsou párové.

- **<pre>** - Předformátovaný text. Uvnitř elementu neplatí obvyklá HTML syntaxe, ale zobrazuje se zalomení řádku a více mezer za sebou tak, jak je ve zdrojovém kódu. Případné

konce řádků na začátku a konci obsahu elementu jsou zanedbány. Má jeden nepovinný atribut:

- ***width*** – definuje počet písmen na řádek (nefunguje vždy – např. IE)

- **<code>, <samp>** - Používá se pro výpis nějakého zdrojového kódu, resp. jeho ukázky. Zobrazuje se neproporcionálním písmem.
- **<tt>, <kbd>** - Teletypový text, vstup z klávesnice. Zobrazuje se neproporcionálním písmem.
- **<var>, <dfn>** - Výpis proměnné kurzívou, výpis definice tučnou, nebo netučnou kurzívou.

6.6.5 Odkazy

- **<a>** - Párová značka, inline. Definice odkazu. Používá se buď pro odkaz na jiný dokument, nebo záložku v rámci dokumentu definovanou atributem *id*, nebo *name*.
  V těle elementu nelze použít další odkaz (tag **<a>**), formulář (tag **<form>**), nebo tabulku (tag **<table>**). Má následující atributy:
  - ***href*** – URL na odkazovaný dokument
  - ***rel, rev, target, type*** – viz **<link>** (kapitola 6.6.1)
  - ***name*** – záložka v rámci dokumentu, na kterou lze odkazovat pomocí ***href=***"*#name*"
  - ***shape*** – definování tvaru oblasti v rámci tagu **<area>**. Používá se ve spojení s atributem ***coords***. Možné hodnoty *rect, rectangle, circ, circle, poly, polygon.*
  - ***coords*** – definuje souřadnice odpovídající atributu ***shape*** v rámci obrázku, nebo obrázkové mapy. Hodnoty pro
    - ***shape***="*rect*"->***coords***="*left,top,right,bottom*"
    - ***shape***="*circ*"->***coords***="*centerx,centery,radius*"
    - ***shape***="*poly*"->***coords***="*x1,y1,x2,y2,...,xn,yn*"
  - ***hreflang*** – základní jazyk cílového URL.
  - ***charset*** – jazyková sada cílového URL.

  Text odkazu je podtržen včetně mezer. Pokud text mezerou začíná, mezera se podtrhne, pokud mezerou končí, mezera se nepodtrhne.
- **<link>** - viz kapitola 6.6.1

Odkazy lze také vytvořit obrázky s klikacími mapami (kapitola 6.6.9), potvrzovacím tlačítkem formuláře (kapitola 6.6.7), nebo skriptem pomocí události ***onclick***.

6.6.6 Rámce

Rámce je možné definovat pouze v DTD Frameset (kapitola 6.1). Kořenovým uzlem je element s tagem **<frameset>**, který se píše za hlavičku dokumentu (**<head>**) místo těla dokumentu (**<body>**).

- **<frameset>** - Párová značka. Rozděluje stránku do několika rámů po řádcích, nebo sloupcích (jen jedna z možností). Může obsahovat pouze tagy **<frameset>, <frame>** a **<noframes>**. Mezi jeho parametry patří:
  - ***cols, rows*** – rozdělení stránky na sloupcové (řádkové) rámy.Použit může být pouze jeden z atributů. Hodnota je tvořena čárkami oddělenými délkami, procenty, nebo * (zbytek stránky).
  - ***frameborder*** – zobrazení rámečku (yes | no | 0 | 1)
  - ***framespacing*** – další místo mezi rámy (pouze IE)
- **<frame>** - Nepárová značka. Představuje rám, jehož odkazem je samostatný dokument. Musí být vložen v elementu **<frameset>**. Má tyto nepovinné atributy:
  - ***src*** – URL na dokument, který má být zobrazen
  - ***name*** – jméno rámu
  - ***noresize*** – může mít pouze hodnotu *noresize* a jejím definováním nemůže uživatel měnit velikost rámu.
  - ***scrolling*** – zobrazení skrolovacích lišt. Hodnoty auto | no | yes.
  - ***marginheight, marginwith*** – prázdné místo na okraji rámu v pixelech.
- **<noframes>** - Párová značka. Pro prohlížeče, které neumějí rámy zobrazit. V jeho těle lze definovat tělo alternativního dokumentu (**<body>**).
- **<iframe>** - Párová značka. Součástí všech DTD. Nemusí být uvnitř **<frameset>**. Význam a funkce stejná jako u **<frame>**, ale jedná se o ***inline prvek*** ! Oproti němu má dva nepovinné atributy:
  - ***widht, height*** – šířka a výška vloženého rámu.

6.6.7 Formuláře

Elementy formuláře dovolují zadat uživateli vstupní informace a odeslat je serveru.

- **<form>** - Párový tag, který obaluje skupinu ovládacích polí do jednoho formuláře a určuje, kam budou data odeslána.
  - ***action*** – povinný atribut, jehož hodnota je URL, kam se mají data odelat. Pokud není uveden, odešle se téže stránce.
  - ***method*** – může být *get* | *post*. První metoda předává data jako součást URL, druhá nezávisle. Více v kapitole 5 o protokolu HTTP.
  - ***name*** – jedinečné jméno formuláře. Není ve striktním DTD.
  - ***target*** - viz kapitola 6.6.1 (tag **<link>**).
  - ***enctype*** - způsob kódování dat dle MIME (kapitola 5).
  - ***accept*** – tečkami oddělený seznam typů, které může server správně zpracovat.
  - ***accept-charset*** – tečkami oddělený seznam možných znakových sad pro data formuláře.

- **<input>** - Nepárový tag. Základní vstupní pole. Má velkou spoustu volitelných atributů, které ovlivňují jeho vzhled i funkcionalitu:
  - ***type*** – typ vstupního prvku. Možné hodnoty *button|checkbox|file|hidden|image|password|radio|reset|submit|text*. Defaulní hodnota je *text*.
  - ***value*** – pro tlačítka definuje nápis, pro checkboxy a radiobuttony jejich číslo, resp. hodnotu, pro skrytá pole, obyčejná pole a pole pro heslo obsahují defaultní hodnotu.
  - ***name*** – jedinečné jméno prvku.
  - ***alt*** – alternativní text – pouze pro typ *image*.
  - ***disabled*** – jediná hodnota *disabled*. Zneplatní prvek, takže ho nejde vybrat, nebo do něj psát text.
  - ***checked*** – jediná hodnota *checked*. Pouze pro typ *checkbox, radio*. Tyto pole jsou defaultně vybrány.
  - ***readonly*** – jediná hodnota *readonly*. Pouze pro typ *text*. Pole nemůže být modifikováno.
  - ***size*** – hodnota udává velikost elementu danou počtem znaků. Nelze pro typ *hidden*.
  - ***src*** – pouze pro typ *image*. Obsahuje URL obrázku k načtení.
- **<textarea>** - Párový tag. Určuje víceřádkové vstupní pole.
  - ***cols, rows*** – povinné atributy. Počet znaků na sloupec/řádek.
  - ***disabled*** - jediná hodnota *disabled*. Zneplatní prvek, takže ho nejde vybrat, nebo do něj psát text.
  - ***readonly*** jediná hodnota *readonly*. Pouze pro typ *text*. Pole nemůže být modifikováno.
  - ***name*** – jedinečné jméno prvku.
- **<button>** - Párový tag. Lze do něj vložit libovolný HTML kód, který se zobrazí na tlačítku.
  - ***type*** – implicině hodnota *button*. Dále může být odesílací *(submit)* a resetovací (*reset*).
  - ***name*** – jméno tlačítka
  - ***value*** – hodnota tlačítka. IE chybně posílá místo value obsah elementu.
- **<select>** - Párový tag. Roletové menu, nebo obdélník s výběrem. Obsahuje pouze tagy **<option**> a **<optgroup>**. Jeho šířka je dána nejširším potomkem.
  - ***name*** – jméno pole
  - ***multiple*** – jediná hodnota *multiple*. Umožňuje hromadný výběr.
  - ***size*** – počet viditelných hodnot
  - ***disabled*** - jediná hodnota *disabled*. Zneplatní prvek, takže ho nejde vybrat, nebo do něj psát text.
- **<optgroup>** - Párový tag. Seskupuje elementy **<option>** a bývá zobrazen tučnou kurzívou.

- **<option>** - Nepárový prvek. Položka výběru v rámci elementu **<select>**. Jeho obsahem je text až do následujícího tagu.
    - ***label*** – štítek při použití **<optgroup>**.
    - ***selected*** – jediná hodnota *selected*. Výběr je implicitně zvolen.
    - ***value*** – hodnota pole pro odeslání.
    - ***disabled*** – jediná hodnota *disabled*. Zneplatní prvek, takže ho nejde vybrat, nebo do něj psát text.
- **<label>** - Párový tag. Popisek jiného prvku, obvykle se uvádí před ním. Prvek formuláře je vybrán i při kliknutí na popisek.
    - ***for*** – volitelný atribut, jeho hodnotou je ***id*** pole, ke kterému se popisek vztahuje.
- **<fieldset>** - Párový tag. Rámeček, kterým lze jiné prvky opticky seskupit.
- **<legend>** - Zastaralé. Párový tag. Nadpis rámečku **<fieldset>**.
- **<isindex>** - Zastaralé. Jednořádkové vstupní pole.

6.6.8 Seznamy

- **<ul>** - Párový tag. Nečíslovaný seznam, položky definovány prvky **<li>**. Atributy zastaralé.
- **<ol>** - Párový tag. Číslovaný seznam, položky definovány prvky **<li>**. Atributy zastaralé.
- **<li>** - Párový tag. Položka seznamu, může se vyskytovat uvnitř **<ul>**, **<ol>**, **<dir>**, **<menu>**. Každá položka se zobrazuje na novém řádku, text je buď za odrážkou, číslem, nebo písmenem. Atributy zastaralé.
- **<dl>** - Párový tag. Seznam definic. Obsahuje střídavě prvky **<dt>** a **<dd>**.
- **<dt>** - Párový tag. Definuje název termínu.
- **<dd>** - Párový tag. Vysvětluje termín definovaný **<dt>**. Odsazení zleva o 40 pixelů.
- **<dir>, <menu>** - Zastaralé. Funkce stejná jako **<ul>**.

6.6.9 Obrázky

- **<img>** - Nepárový tag. Představuje obrázek, který se načte ze zadané adresy. Má množství atributů, z nichž jsou dva povinné:
    - ***src*** – povinný atribut. Zdrojové URL obrázku. Podporuje obrázky typu BMP, GIF, JPG a PNG (v IE omezeně).
    - ***alt*** – povinný atribut. Alternativní text, který se zobrazuje pokud není obrázek načten a také jako bublinový text při přejetí myší přes obrázek (ne ve Firefoxu).
    - ***longdesc*** – URL k dokumentu, který popisuje obrázek.
    - ***height, width*** – šířka a výška obrázku definovaná v pixelech, nebo procentech. Doporučuje se definovat kvůli optimalizaci načítání stránky. Pokud je uveden jen jeden z atributů, druhý se dopočítá tak, aby byla zachována proporce obrázku.

- ***ismap*** – definice obrázkové mapy na straně serveru. Pokud je obrázek odkazem, při kliknutí se za adresu připojí otazník a souřadnice oddělené čárkou.
- ***usemap*** – definice obrázkové mapy na straně klienta. Hodnotou atributu je mříž (#) a jméno mapy, která je vyznačena tagem **<map>**.
- ***align, border, hspace, vspace*** – Zastaralé. Místo těchto atributů se používají styly.

- **<map>** - Párový tag. Definice obrázkové mapy. Obsahuje seznam aktivních oblastí, které jsou reprezentovány prvky **<area>**. Těch může mít libovolný počet. Po kliknutí na oblast se aktivuje příslušný odkaz. Tento element se nezobrazuje.
  - ***id*** – povinný atribut je jedinečným jménem mapy.
  - ***name*** – nepovinný atribut, který také definuje jméno mapy. Je některými prohlížeči vyžadován.
- **<area>** - Nepárový tag. Vždy musí být vnořen v elementu **<map>**. Definuje souřadnice na klikací mapě na obrázku.
  - ***alt*** – povinný atribut. Alternativní text pro oblast.
  - ***shape, coords, href, target*** – viz odkaz **<a>** (kapitola 6.6.5).

6.6.10 Tabulky

Tabulky slouží pro strukturované zobrazení dat, případně pro rozvržení vzhledu stránky. Nicméně tato možnost je spíše zastaralá, neboť prohlížeče zobrazují obsah tabulky až po načtení všech buněk.

- **<table>** - Párový tag. Kořenový element tabulky.
  - ***border*** – šířka čar ohraničujících buňky tabulky. Pokud nastaveno na nulu, buňky budou bez rámečku.
  - ***cellpadding, cellspacing*** – vnitřní a vnější okraj tabulky.
  - ***frame*** – definuje, které čáry rámečku se mají zobrazit.
  - ***rules*** – další pravidla pro vykreslení čar.
  - ***width*** – šířka tabulky v pixelech, nebo procentech.

  Ačkoliv jsou tyto atributy i ve striktní definici, je vhodnější je nahrazovat CSS styly.
- **<tr>** - Párový tag. Řádek tabulky. Smí obsahovat pouze **<th>** a **<td>**.
  - ***align*** – zarovnání textu v buňkách. Hodnoty *left, right, center, justify, char*.
  - ***valign*** – vertikální zarovnání textu v buňce. Hodnoty *top, middle, bottom, baseline*.
  - ***char, charoff***
- **<th>, <td>** - Párové tagy, nicméně ukončovací se často neuvádí. Definují buňku tabulky. První z tagů je nadpisem a text uvnitř je tučný a vystředěný.
  - ***abbr*** – stručný textový popis buňky.
  - ***align, valign, char, charoff*** – viz tag **<tr>**.
  - ***axis, scope, headers*** – informace pro automatické zpracování.
  - ***rowspan, colspan*** – sloučení buněk přes *n* řádků (sloupců).

- ***width, height, nowrap, bgcolor*** – zastaralé.

- **<thead>, <tbody>, <tfoot>** - Párové tagy. Seskupování řádků tabulky. Tagy představují hlavičku, tělo a zápatí. Musí být použity buď všechny najednou, nebo žádný. Tag **<thead>** musí obsahovat řádek **<tr>**.
- **<caption>** - Nadpis tabulky. Měl by být prvním potomkem elementu **<table>**. Každá tabulka může mít pouze jeden nadpis. Zobrazuje se jako vycentrovaný nápis nad tabulkou.
- **<col>** - Nepárový tag. Element nemá obsah, pouze definuje atributy pro všechny buňky v jednom sloupci. Vztahuje se ke sloupcím tak, v jakém pořadí je napsán.
  - ***span*** – určuje, pro kolik sloupců vedle sebe má platit.
  - ***align, valign, char, charoff, width***
- **<colgroup>** - Nepárový tag. Určuje vlastnosti pro více sloupců jako element **<col>**, které mají větší prioritu.

6.6.11 Styly

- **<style>** - Párový tag. Zápis stylu dokumentu v jazyce CSS. Měl by být součástí hlavičky **<head>**.
  - ***type*** – jediná možná hodnota *text/css.* Definuje, že bude použit jazyk CSS.
  - ***media*** – definuje výstupní zařízení, na které se bude zobrazovat. Hodnoty *screen|tty|tv|projection|handheld|print|braille|aural|all*.
- ** ** - Párový tag. Označuje se jako oddíl.Může obsahovat libovolné elementy. Používá se pro formátování prvků uvnitř něho. Dále používá pro rozložení prvků na stránce a v tomto smyslu nahradil zastaralou metodu rozložení tabulkami. Před a za oddílem se zalomí řádek, jde tedy o blokový prvek. Narozdíl od odstavce <p> se není obklopen prázdými řádky. Má pouze standardní atributy (kapitola 6.7) a zastaralý ***align***, který je nahrazen CSS stylem.
- **** - Párový tag. Jde o prvek se stejným významem jako , rozdíl je v tom, že tento prvek je in-line, tzn. nezalamuje před a po sobě řádek. Může obsahovat pouze inline elementy.

6.6.12 Programování

- **<script>** - Párový tag. Obsahuje zápis skriptu v jiném jazyce, nejčastěji JavaScriptu. Může být kdekoliv v dokumentu, začne se zpracovávat okamžitě, jakmile jsou instrukce načteny ze zdrojového textu. Skript může být v těle elementu, nebo jako externí soubor.
  - ***type*** – povinný atribut. Označuje MIME typ textu (kapitola 4). Možné hodnoty *text/javascript, text/ecmascript, application/javascript, application/ecmascript, text/vbscript*.
  - ***src*** – URL adresa na externí skript

    - ***defer*** – pro urychlení načítání. Indikace, že skript negeneruje žádný obsah dokumentu, který by se zobrazoval a může být načten až později.
    - ***charset*** – znaková sada skriptu. Zvláště starší prohlížeče nepodporují.
- **<noscript> -** Párový tag. Alternativní obsah, pokud není vykonán kód skriptu. Pro prohlížeče, které umějí tag <script> rozpoznat, ale ne zpracovat jeho obsah. Pokud je skript zpracován, tento element je ignorován.
- **<applet> -** Zastaralé. Párový tag. Definuje vložený Java applet. Jeho obsahem mohou být pouze tagy <param>.
    - ***height, width*** – povinné atributy. Výška a šířka appletu v pixelech.
    - ***code*** – URL, které ukazuje na soubor s třídou appletu (soubor .class)
    - ***codebase*** – URL, které ukazuje na pracovní adresář appletu.
    - ***align, alt, archive, hspace, name, object, title, vspace***
- **<object> -** Párový tag. Definuje vložený objekt podobně jako <applet>, nicméně jeho možnosti jsou širší. Také může obsahovat pouze prvky <param>, ostatní je považováno za alternativní text. Může být uvnitř hlavičky i těla dokumentu. Používá se pro přidání multimédií do stránky. Zabalení objektu do odkazu nefunguje.
    - ***data*** – URL adresa na vkládaný soubor.
    - ***type*** – MIME typ objektu (viz kapitola 4).
    - ***name*** – unikátní jméno objektu.
    - ***codebase*** – URL na základní (pracovní adresář) objektu.
    - ***classid*** – jedinečné identifikační ID programu .
    - ***standby*** – text, který se bude zobrazovat během načítání objektu.
    - ***archive*** – mezerami oddělené adresy URL k archivům. Ty definují zdroje důležité pro objekt.
    - ***codetype*** – MIME typ kódu, který je odkazován atributem ***classid***.
    - ***declare*** – definuje, že objekt má být vytvořen až ve chvíli, kdy je ho třeba.
    - ***height, width*** – výška a šířka objektu.
    - ***usemap*** – URL na obrázkovou klikací mapu.
- **<param> -** Nepárový tag. Může být vložen pouze v elementech <applet>, nebo <object>. Specifikuje run-time nastavení objektu.
    - ***name*** – povinný atribut. Jedinečné jméno parametru.
    - ***value*** – hodnota parametru.
    - ***valuetype*** – MIME typ atributu ***value***.
    - ***type*** – MIME typ parametru.

6.7 Standardní atributy

Standardní atributy mohou být (až na pár výjimek) použity jako atributy každého elementu. Dělí se na základní (*core attributes*), jazykové a klávesové.

6.7.1 Základní atributy

- ***class*** – třída elementu. V dokumentu může být obecně více elementů se stejnou třídou. Význam pro CSS
- ***id*** – jedinečný identifikátor elementu. V dokumentu může být jenom jednou. Dále také jako náhrada atributu ***name***.
- ***style*** – definice CSS stylu platná jen pro element, v němž je definován (a jeho obsah). Aplikovatelné na zabrazitelné elementy.
- ***title*** – titulek elementu zobrazovaný jako bublinový text.

6.7.2 Jazykové elementy

- ***dir*** – nastavení směru textu. Buď zleva doprava (hodnota *ltr*), nebo zprava doleva (hodnota *rtl*).
- ***lang*** – jazyk elementu.

6.7.3 Klávesové elementy

- ***accesskey*** – jeho hodnotou je znak, který je použit jako klávesová zkratka pro přístup k elementu. Aplikovatelné na aktivní elementy.
- ***tabindex*** – číslo, které určuje pořadí elementu při výběru tabulátorem. Aplikovatelné na aktivní elementy.

6.8 Atributy událostí

Tyto atributy umožňují zachytit a zpracovat nastalé události, jako provedení JavaScriptového kódu když uživatel klikne na HTML element. Dělí se a události oken, formulářů, klávesnice a myši.

6.8.1 Události oken

Jsou dva a mají název ***onload*** a ***onunload***. Mohou být použity pouze v elementech <body>, nebo <frameset>. Aktivují se v případě načtení, resp. opuštění stránky.

6.8.2 Události formulářů

Již z názvu je patrné, že je lze aplikovat pouze na elementy spojené s formulářem (kapitola 6.6.7).

- ***onchange, onselect*** – skript spuštěn při změně (výběru) elementu.
- ***onsubmit, onreset*** – skript spuštěn po stisknutí tlačítka submit (reset).
- ***onfocus, onblur*** – skript spuštěn při nabytí (ztrátě) focusu.

6.8.3 Události klávesnice a myši

Je použitelné na všechny elementy kromě <base>, <bdo>, <br>, <frame>, <frameset>, <head>, <html>, <iframe>, <meta>, <param>, <script>, <style>, <title>.

Aktivace skriptu je patrná z názvů událostí: ***onkeydown, onkeyup, onkeypress, onclick, ondblclick, onmousedown, onmousemove, onmouseout, onmouseover, onmouseup***.

6.9 Barvy

V HTML se barvy určují ze složek červené, modré a zelené (RGB). K jejich vyjádření se používá buď hexadecimálního zápisu, nebo jméno, kterými jsou definovány základní barvy. Při použití validátoru stránky je možné použít jména pouze na 16 základních barev definovaných W3C (*aqua, black, blue, fuchsia, gray, green, lime, maroon, navy, olive, purple,red, silver, teal, white* a *yellow*). Prohlížeče jich ovšem definují mnohem více.

Hexadecimální zápis má tvar #RRGGBB, nebo zkráceně #RGB.

6.10 Shrnutí

Kapitola o HTML popsala všechny důležité rysy jazyka HTML, z čeho vychází a většinu jeho syntaxe. Následující kapitola velmi krátce popíše jazyk XHTML a rozdíly mezi ním a HTML.

7 XHTML

Jak již název napovídá, jedná se o rozšíření HTML, písmenko X znamená **eXtensible**. Jazyk HTML vznikl jako aplikace SGML, jehož rozšířením je XML a z něho vznikl právě XHTML. Většina jazyka je podobná normě HTML 4.01. V podstatě se jedná o čistší a více striktní verzi HTML 4.01, s níž je kompatibilní.

7.1 DTD pro XHTML 1.0

Stejně jako HTML 4.01 existují tři definice typů: STRICT, TRANSITIONAL a FRAMESET.

Definice se píše na první řádek XHTML dokumentu a má následující tvar:

<!DOCTYPE html PUBLIC <formal public identifier> <URL definice DTD> >

V tomto pořádí uvedeme také konkrétní případy:

- DTD Strict
  - `"-//W3C//DTD XHTML 1.0 Strict//EN"`
  - `http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd`
- DTD Transitional
  - `"-//W3C//DTD XHTML 1.0 Transitional//EN"`
  - `http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd`
- DTD Frameset
  - `"-//W3C//DTD XHTML 1.0 Frameset//EN"`
  - `http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd`

7.2 Struktura dokumentu

Oproti HTML 4.01 se dokument skládá ze povinně tří částí, které jsou:

- DOCTYPE
- <head>
- <body>

7.3 XHTML vs. HTML

Základní rozdíly oproti HTML jsou následující:

1. XHTML elementy musí být řádně vnořeny
2. XHTML elementy musí být uzavřeny
3. XHTML elementy musí být psány malými písmeny

4. XHTML dokument musí mít jeden kořenový element.

Řádné vnoření bylo možné v HTML v některých případech porušit, např. `<b><i>text</b></i>`. Uzavření elementů znamená, že všechny neprázdné elementy musí mít koncový tag. Prázdné elementy musí mít buď koncový tag, nebo startovní tag musí končit znaky />, např.
. Jeden kořenový element znamená, že všechny elementy XHTML dokumentu musí být vnořeny v <html> elementu.

7.4 Další pravidla XHTML syntaxe

- Názvy atributů musí být psány malými písmeny.
- Hodnoty atributů musí být ohraničeny uvozovkami.
- Minimalizace atributů není povolena (musí být `nazev="hodnota"`).
- Atribut ***id*** nahrazuje atribut ***name***.
- Pokud má element atribut ***lang***, musí mít také atribut ***xml:lang***.
- Element <html> musí být ve tvaru <html xmlns="http://www.w3.org/1999/xhtml">.

8 Cascading Style Sheet

Kaskádové styly (CSS) reprezentují jazyk, který popisuje vzhled HTML elementů a umožňuje tudíž oddělit programování obsahu a vzhledu. Zdrojový kód CSS se skládá z množiny pravidel, které se skládají ze dvou částí: selektoru a deklarace. Deklarace se skládá z množiny vlastností a jejich hodnot, oddělených středníkem. V této kapitole se stručně seznamíme se syntaxí a CSS box modelem. Visuálním formátovacím modelem se zde zabývat nebudu, neboť bude rozebrán v rámci popisu implementace Scamperu v kapitole 11. Tyto informace jsem čerpal převážně z CSS specifikace[22].

8.1 Selektory, pseudotřídy a pseudoelementy

Selektory, pseudotřídy a pseudoelementy určují, na které HTML elementy, případně v jaké situaci se budou aplikovat deklarační bloky vlastností následující za selektorem (jsou uzavřeny do složených závorek { a }).

8.1.1 Selektory

Jednoduchý selektor se skládá z typového, nebo univerzálního selektoru, který může být následován jedním, nebo více atributovými selektory, ID selektory, nebo pseudotřídami. Je možné je seskupovat, tzn. jeden deklarační blok přiřadit více selektorům oddělením čárkou.

Univerzální selektor se označuje znakem „*“ a bez dalších informací platí pro všechny HTML elementy. Typový selektor označuje konkrétní element a jeho název odpovídá názvu elementu.

8.1.2 Pseudoelementy

Pseudoelementy nabízejí možnost výběr jinak nedostupných částí dokumentu, jako je část obsahu elementu (první písmeno, první řádka). Dále umožňují definovat styl pro obsah, který není ve zdrojovém dokumentu.

8.1.3 Pseudotřídy

Pseudoelementy vybírají elementy podle charakteristik, které nejsou určitelné ze stromu dokumentu. Mohou to být i dynamické vlastnosti jako pozice kurzoru nad elementem.

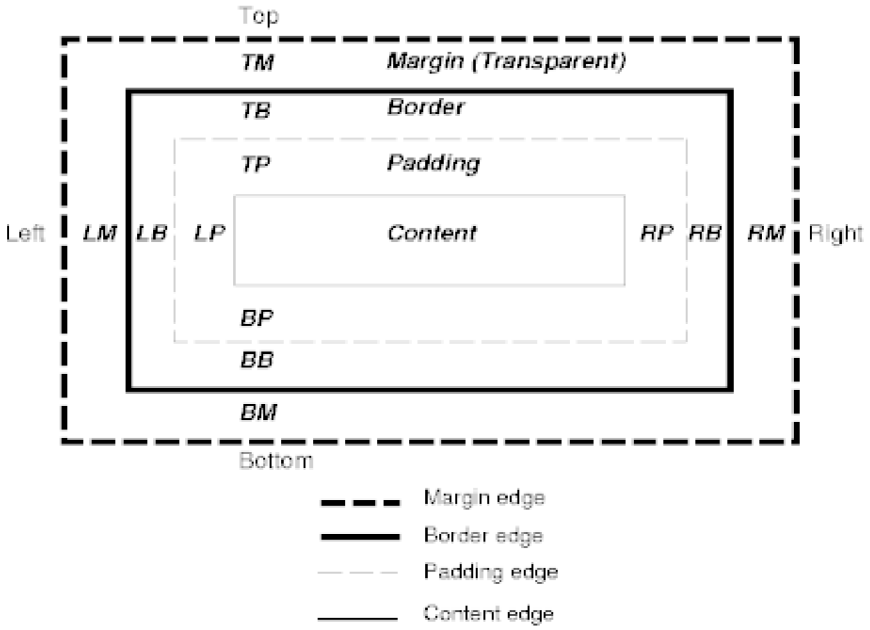
8.1.4 Syntaxe selektorů

Následující tabulka zobrazuje možné kombinace selektorů, pseudoelementů a pseudotříd a jejich význam:

Vzor	*Význam*
*	Vyhovuje každému elementu
E	Vybere element typu E
E F	Vybere element F, který je potomkem elementu E (nemusí být přímý)
E > F	Vybere element F, který je přímým potomkem E
E:first-child	Vybere prvního přímého potomka elementu E
E:link	Vybere element, který je odkazem. Tento odkaz ještě nebyl navštíven
E:visited	Vybere element, který je odkazem. Tento odkaz již byl navštíven
E:active	Vybere element, který byl právě aktivován (například kliknutím myší)
E:hover	Vybere element, nad který byl umístěn kurzor myši
E:focus	Vybere element E s fokusem
E:lang (cz)	Vybere element E, který je je v daném jazyce cz (zde v češtině)
E + F	Vybere element F, jehož prevSibling je E
E[foo]	Vybere element E, který má definován atribut foo
E[foo="warning"]	Vybere element E, který má definován atribut foo s hodnotou warning
E[foo~="warning"]	Vybere element E, který má definován atribut foo s hodnotou jinou než warning
E[lang|="en"]	Vybere element E, mezi jehož jazyky patří en
E.warning	Vybere všechny elementy třídy warning.
E#myID	Vybere element E s ID="myID"

8.2 CSS Box model

CSS box model popisuje obdélníkovou oblast (box), kterou generuje každý element ve stromu dokumentu. Každý box obsahuje plochu obsahu (content area) a nepovinné padding, border a margin plochy. Popis boxu a jeho ploch popisuje Kresba 8.1 převzatá z [23]:



Kresba 8.1: Plochy boxu

- Content box – uvnitř tohoto boxu jsou renderovány vnitřní elementy. Je definován vlastnostmi *width* a *height*

- Padding box – vnitřní prázdná plocha boxu
- Border box – rámeček boxu
- Margin box – vnější prázdná plocha boxu.

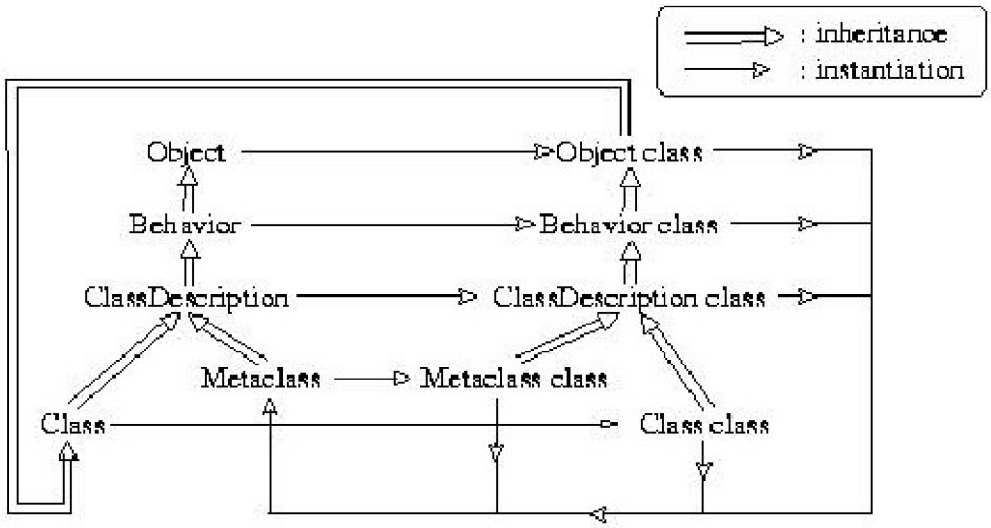
K definování šířky boxů slouží vlastnosti s odpovídajícím názvem: *margin-top, margin-bottom, margin-left, margin-right, padding-top, padding-bottom, padding-left, padding-right, border-top-width, border-bottom-width, border-left-width, border-bottom-width*. Každý box lze definovat i jednou, zkrácenou vlastností: *margin, padding, border-width*. Mohou mít určeny jednu až čtyři hodnoty. Pokud je definována jedna hodnota, platí pro všechny čtyři strany. Pokud dvě hodnoty, je první hodnota pro vrchní a spodní hranu, druhá pro levou a pravou hranu. Pokud tři hodnoty, první patří vrchní straně, druhá levé a pravé a třetí spodní straně. Více o box modelu se lze dočíst ve [23].

8.3 Shrnutí

Tato kapitola nastínila některé hlavní části jazyka kaskádových stylů. Nemá sloužit jako reference jazyka, spíše jako vysvětlení hlavních částí pro další kapitoly. V těch se již podíváme do jazyka Smalltalk a prostředí Squeak. Více o CSS se lze dočíst ve [22].

9 Squeak Smalltalk

V jazyce Smalltalk je vše objektem. Tedy i třídy jsou vlastně objekty, které mají za své třídy metatřídy. Metatřídy jsou také objektem a mají své vlastní třídy. Tím vzniká jakási cyklická dědičnost, která bude nejlépe zřejmá z obrázku Kresba 9.1.



Kresba 9.1: Základ tříd Squeak Smalltalku

Jednou z možností je využívání dědičnosti a psaní odvozených tříd, nebo použití nového konceptu ***Traitů***. O této možnosti pojednává [16]. Veškerá komunikace mezi objekty probíhá pomocí zpráv.

Základní třídou je **Object**, ze které dědí všechny ostatní třídy. **Object** dále dědí sám z třídy **ProtoObject** a jelikož musí být hierarchie nějak ukončená a zároveň musí platit pravidlo, že všechny třídy jsou následovníkem jiné, dědí **ProtoObject** sám ze sebe.

Smalltalk je jazyk interpretovaný a podobně jako Java (která z něho čerpá) běží na virtuálním stroji, který je rozšířen na mnoho platforem. Tím je vlastní prostředí Squeak Smalltalk zcela platformně nezávislé. Dokonce může fungovat jako samostatný operační systém, prakticky se používá pouze jako nástavba na linuxovém jádře.

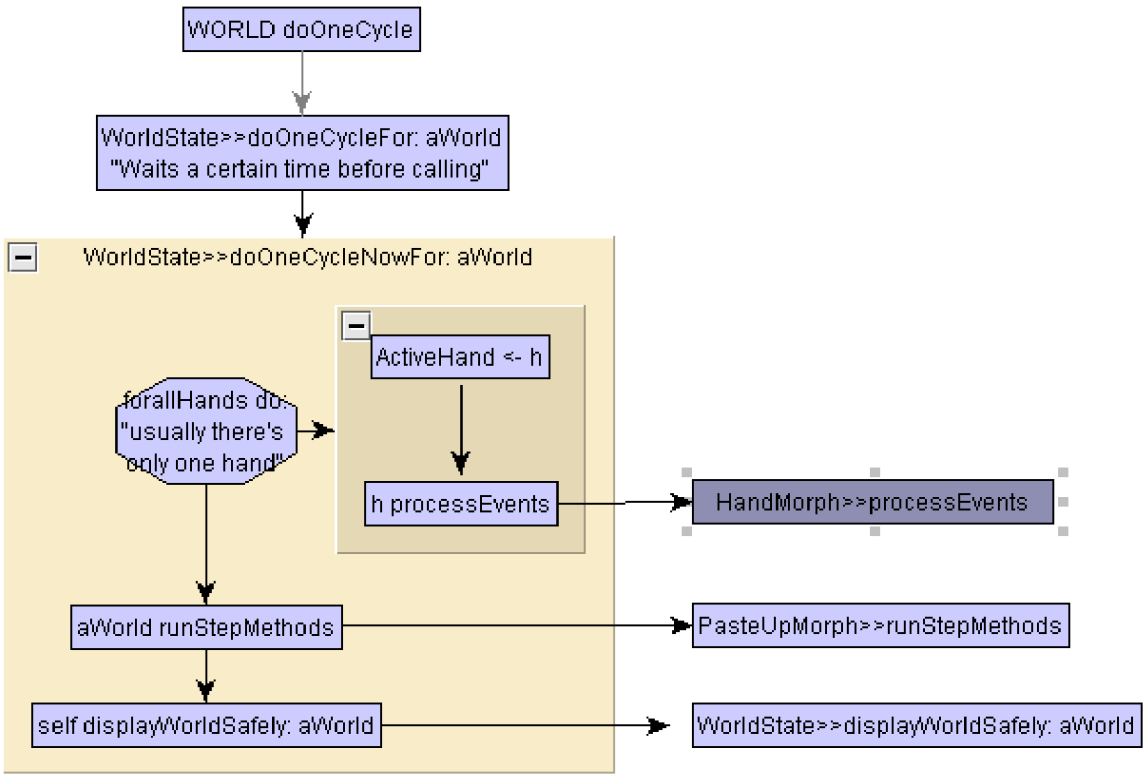
Squeak má dvě grafická prostředí. První – původní – se jmenuje ***MVC*** a jeho výhodou jsou jeho extrémně nízké nároky. Druhé bylo převzato ze systému *Self* a jmenuje se ***Morphic***. O něm stručně pojedná následující kapitola.

10 Morphic – grafické prostředí Squeaku

Podobně jako třídy Smalltalku vychází z koncepce, že vše dědí ze třídy Object, Morphic vychází z představy, že vše je Morph.

Morph může být hierarchicky komponován, reaguje na události (myš, klávesnice). kompozice morphů se může dynamicky měnit. Kořenem hierarchie je World. Je to morph, který se oproti ostatním umí vykreslit do framebufferu (a tedy zobrazit se na obrazovce). World obsahuje jeden, nebo více Hands. Hand reprezentuje kurzor myši a čte ze systému události, které přeposílá dalším morphům[17].

Veškeré zpracování Morphicu probíhá v jedné smyčce, což je velká nevýhoda tohoto prostředí a při neopatrné manipulaci nezřídka způsobí „zamrznutí“ celého GUI. Smyčka se nazývá WORLD doOneCycle. Situaci ilustruje Kresba 10.1:



Kresba 10.1: Morphic - World doOneCycle

Rozmístění morphů uvnitř jiného je možné automatizovat pomocí třídy LayoutPolicy a z ní odvozených. Následující podkapitoly popíšou tyto politiky.

10.1 TableLayout

Umožňuje automatizované rozmístění submorphů po řádcích, případně sloupcích a automatizovanou úpravu velikosti nadřazeného morphu v závislosti na celkové velikosti ostatních. Výpočet umístění probíhá ve čtyřech krocích:

- Prvním krokem je výpořet minimálních rozměrů každého submorphu.
- Je vypočítán počet buněk na každý řádek (sloupec) v závislosti na velikosti buněk.
- V závislosti na velikosti řádku (sloupce) je dopočítáno extra místo, které má být k němu přidáno:
    - #leftFlush/#topFlush – extra místo se přidá na konec
    - #rightFlush/#bottomFlush – extra místo se přidá na začátek
    - #centering – na každou stranu se přidá ½ extra místa
    - #justified – extra místo je rozděleno rovnoměrně mezi morphy.
- Morphy jsou umístěny do buněk.

10.2 ProportionalLayout

Tato politika rozmísťuje všechny potomky nějakého morphu do daného LayoutFrame. Tento frame má hodnoty, kterými lze určit relativní, či pevná pozice:

- ***leftFraction, topFraction, rightFraction, bottomFraction*** – Float hodnoty, určující poměr (mezi 0 a 1) k umístění Morphu do jeho vlastníka.
- ***leftOffset, topOffset, rightOffset, bottomOffset*** – Integer hodnoty, určující pevný offset v pixelech, který se má aplikovat po frakcionálním rozmístění (např. „10 pixelů napravo od středu vlastníka“).

Vlastností tohoto rozmístění je, že submorphy mění automaticky velikost podle velikosti svého vlastníka.

10.3 Shrnutí

Kapitola o grafickém prostředí Morphic ukázala jeho základní cyklus a především byla zaměřena na popis rozmísťovacích politik, což bude teoreticky důležité při přepracování zobrazovacích vlastností Scamperu (kapitola 11).

11 Scamper – původní stav

Scamper je jednoduchým webovým prohlížečem pro prostředí Squeak Smalltalk. Jeho základy vytvořil programátor IBM ***Alexander Spoon*** (http://www.lexspoon.org). Na zpracování DOM (Document Object Model – http://w3c.org/DOM/) se podílel ***David J. Pennel***. Jako vlastník je uváděn ***Tansel Ersavas*** (tansel@squeakonline.com). Další lidé uvádění jako spolupracovníci jsou ***Marcus Denker*** (denker@iam.unibe.ch) a ***Göran Krampe*** (goramSPAM_BLOCK@krampe.se). Scamper je zveřejněn pod licencí SqueakL. Je nutné dodat, že literatura, ani manuál ke Scamperu nejsou dostupné a zpracování komentářů je slabé. Všechny následující poznatky kromě kapitoly 10.1 jsem odvodil pozorováním a zkoušením a je možné, že budou obsahovat drobné nepřesnosti.

11.1 Dostupné funkce

V současné době podporuje Scamper následující:

- nadpisy **(title)**
- hlavičky **(h1-h6)**
- odstavce **(p)**
- odkazy **(a)**
- seznamy **(ul,ol)**
- tagy pro zvýraznění textu
- HTTP přesměrování
- obrázky **(image)**

Naopak neumí následující:

- tabulky **(table)**
- cookies
- rámy **(frameset, frame, iframe)**
- některé formuláře
- obrázkové mapy
- přesměrování a znovunačtení pomocí tagu <**meta**>
- Java, JavaScript
- CSS styly

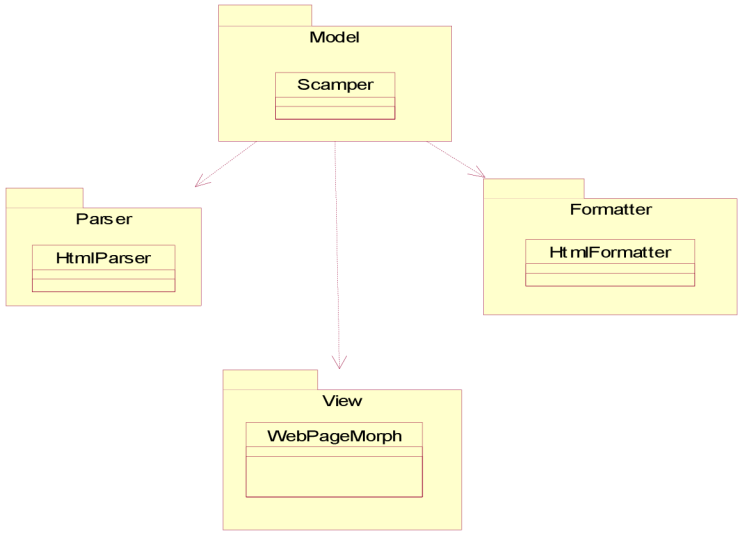
Oficiální TODO:

- podpora pozadí stránky (rozpracováno)
- rámy
- cíle odkazů (***target***)
- podpora tabulek/TableMorph (GridMorph)

- manažer stahování
- vylepšená navigace, podpora záložek
- integrace Squeak-Postscript pro tisk stránek
- podpora CSS stylů
- integrace možností XML
- integrace Celeste, Telnet, PWS, News klienta...
- podpora SqueakScript

11.2 Architektura Scamperu

Na architekturu je možné pohlížet jako na kolekci vrstev Model-Parser-Formatter-View.



Kresba 11.1: Architektura prohlížeče Scamper

11.3 Vrstva Model

Hlavní třídou prohlížeče je **Scamper**, která dědí ze třídy **Model**. Ta poskytuje rozhraní pro rychlou a účinnou správu závislých objektů. Do této vrstvy patří také třídy **ScamperWorker**, jenž pracuje jako vlákno na pozadí a stahuje nekompletní morphy jako třeba obrázky, a **BrowserURL,** třída obsahující URL a spolupracující s browserem.

11.3.1 Třída Scamper

Výchozí metoda pro start prohlížeče je ***Scamper class***>>*openOnUrl:*. Postup při vytváření stránky s HTML obsahem ukazuje následující sekvenční diagram:

Kresba 11.2: Sekvence načtení HTML dokumentu

Z diagramu je podstatné zmínit, že ve frontě objektu **SharedQueue** se nacházejí objekty typu **MIMEDocument** s informací o MIME typu dokumentu a jeho obsahem. Jakmile se do fronty přidá dokument, asynchronně se zavolá metoda ***Scamper***>>*stepAt:in:*, ve které se podle typu dokumentu zavolá příslušná metoda pro zpracování, zde pro HTML dokument. Metoda ***Scamper***>>*displayHtmlTextPage:* pak vytváří parser a formatter, které vytvoří strom dokumentu a zobrazí jej graficky.

11.3.2 Třída ScamperWorker

Tato třída slouží ke stahování dalších objektů, jako například obrázků, které ještě nebyly načteny. Pro každé stahování je vytvořen nový proces. Objekty ScamperWorker jsou vytvářeny v metodě ***Scamper***>>*startDownloadingMorphState* (viz Kresba 11.2).

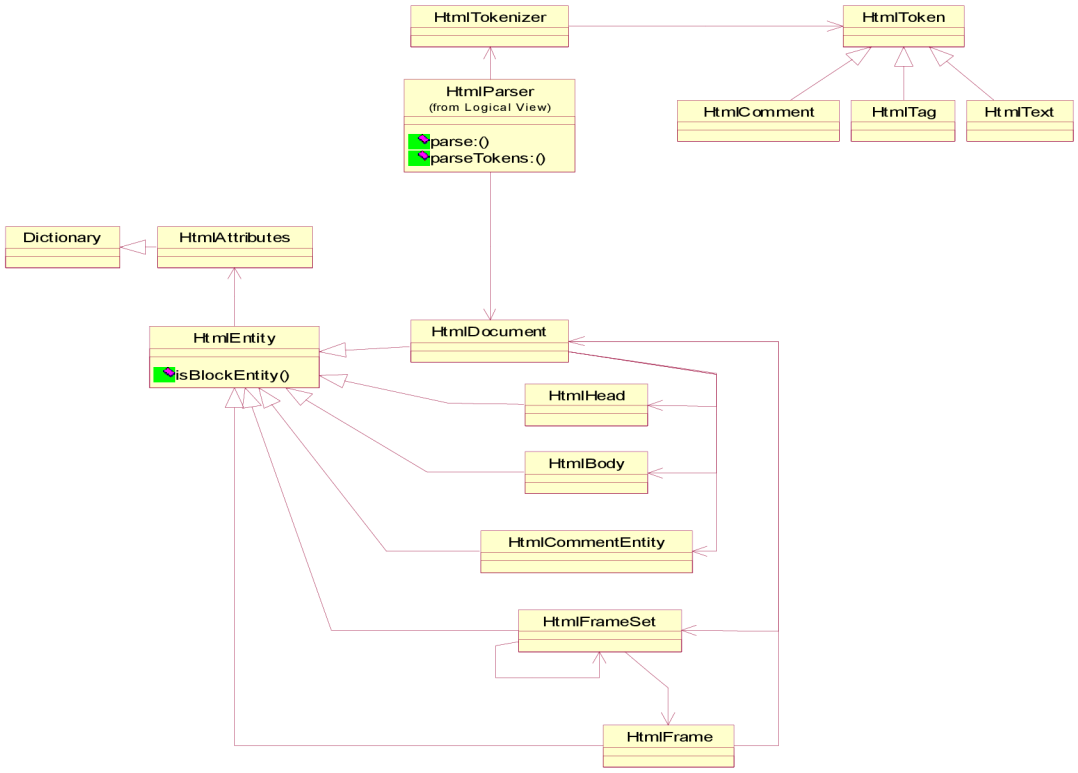
11.3.3 Třída BrowserUrl

Třída **BrowserUrl** je objektem URL adresy, která dokáže Scamperu kromě klasické URL oznámit, že má zobrazit startovní stránku, nebo stránku *about*.

Hierarchie třídy je **Object**->**Url**->**GenericUrl**->**BrowserUrl**.

11.4 Vrstva Parser

Parsing dokumentu začíná metodou ***HtmlParser class***>>*parse:*. Vstupní data tvoří **ReadStream** objekt, který se přetypuje na stream **HtmlTokenizer**. S ním pak parser pracuje a vytváří stromovou strukturu dokumentu, kde každý rozpoznaný element má vlastní třídu, každý nerozpoznaný je evidován jako komentář. Základ návrhového diagramu tříd vypadá takto:



Kresba 11.3: Parser - diagram návrhových tříd

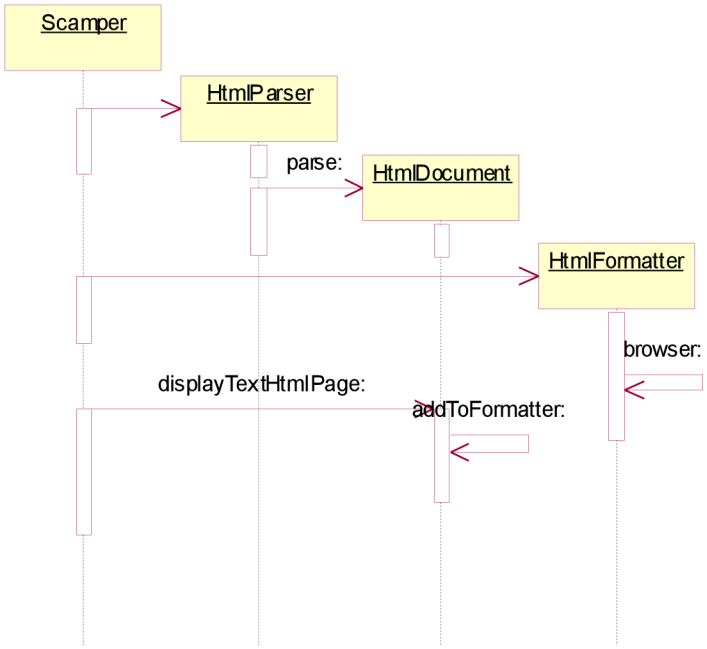
Tříd je ovšem mnohem více, v diagramu jsou jen hlavní třídy dokumentu.

- Z **HtmlEntity** dědí třídy: **HtmlArea**, **HtmlBlockEntity**, **HtmlBody**, **HtmlDefinitionListElement**, **HtmlDocument**, **HtmlForm**, **HtmlFormEntity**, **HtmlFrame**, **HtmlFrameSet**, **HtmlHead**, **HtmlHeadEntity**, **HtmlHeader**, **HtmlHorizontalRule**, **HtmlListItem**, **HtmlMap**, **HtmlParagraph**, **HtmlPreformattedRegion**, **HtmlTable**, **HtmlTableDataItem**, **HtmlTableRow**, **HtmlTextualEntity**.
    - Z **HtmlBlockEntity** dědí třídy: **HtmlBlockQuote**, **HtmlDefinitionList**, **HtmlList**.
        - Z **HtmlList** dědí třídy: **HtmlOrderedList**, **HtmlUnorderedList**.
    - Z **HtmlDefinitionListElement** dědí třídy: **HtmlDefinitionDefinition**, **HtmlDefinitionTerm**.
    - Z **HtmlFormEntity** dědí třídy: **HtmlButton**, **HtmlInput**, **HtmlOption**, **HtmlOptionGroup**, **HtmlSelect**, **HtmlTextArea**.
    - Z **HtmlHeadEntity** dědí třídy: **HtmlMeta**, **HtmlStyle**, **HtmlTitle**.
    - Z **HtmlTableDataItem** dědí třídy: **HtmlTableHeader**.
    - Z **HtmlTextualEntity** dědí třídy: **HtmlCommentEntity**, **HtmlFontChangeEntity**, **HtmlSpecialEntity**, **HtmlTextEntity**.
        - Z **HtmlFontChangeEntity** dědí třídy: **HtmlBiggerFontEntity**, **HtmlBoldEntity**, **HtmlFixedWidthEntity**, **HtmlFontEntity**, **HtmlItalicsEntity**, **HtmlSmallerFontEntity**, **HtmlStrikeEntity**, **HtmlSubscript**, **HtmlSuperScript**, **HtmlUnderlineEntity**.

Do vrstvy Parser by se daly jistě zařadit i formulářové prvky, nicméně jim je věnována samostatná část v kapitole 10.7.

11.5 Vrstva Formatter

Tato vrstva je tvořena hlavně třídou **HtmlFormatter** a z ní odvozené **DHtmlFormatter**. Snaží se vytvořit hezky vypadající text, a to hlavně pomocí **TextMorphu**, což je hlavní slabinou Scamperu, jelikož jeho rozšíření o CSS styly a tabulky je pak velmi obtížné. Základní sekvenční diagram formátování ukazuje Kresba 11.4. Naformátovaný text je pak přístupný metodou ***HtmlFormatter***>>*text*.

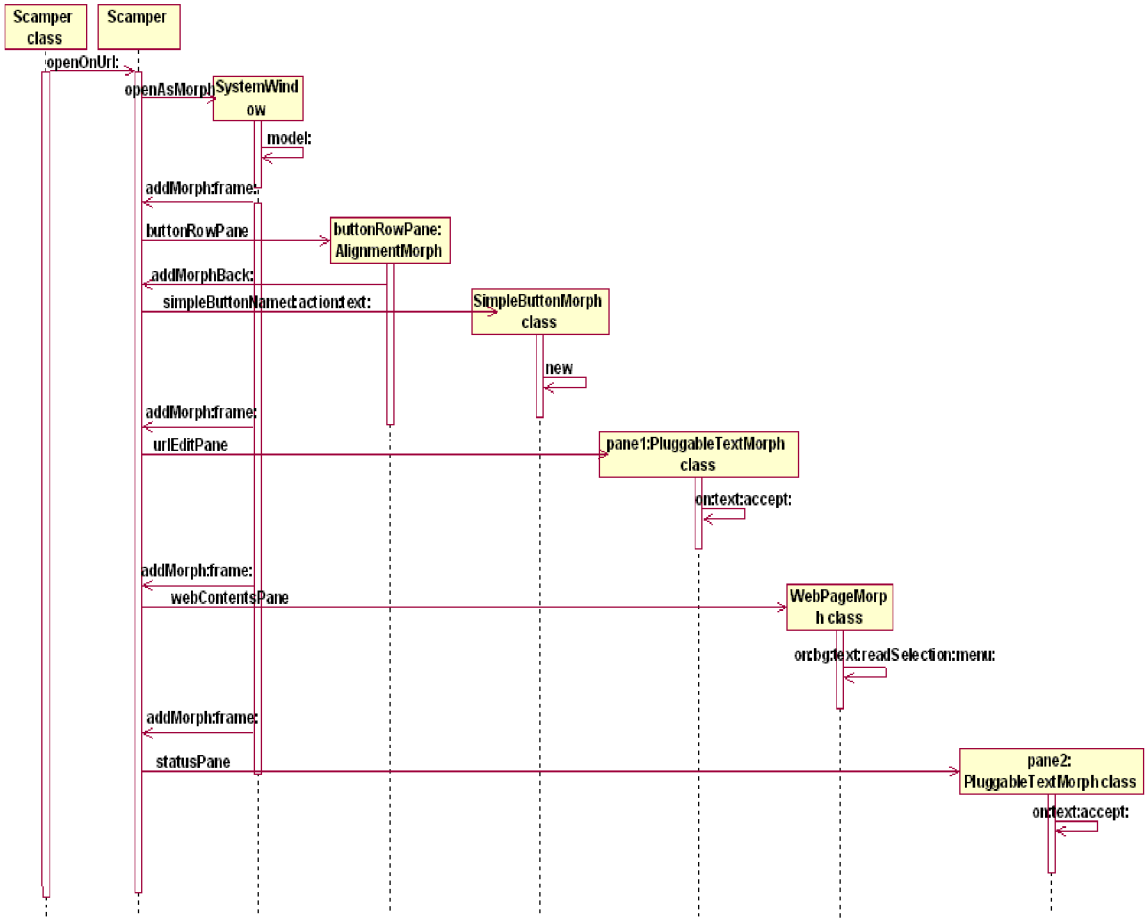


Kresba 11.4: Sekvenční diagram formátování

11.6 Vrstva View

Druhou věcí, která je při načítání dokumentu potřeba udělat, je jeho grafické prostředí. Opět se vychází z metody ***Scamper class***>>*openOnUrl:* a situaci popisuje sekvenční diagram Kresba 11.5.

Hlavní okno je tvořeno morphem **SystemWindow**, které jako model přijímá objekt **Scamper**. Do tohoto okna jsou postupně přidávány další morphy. Nejprve lišta s navigačními tlačítky, poté textové pole pro zadávání URL adresy, posléze vlastní okno dokumentu tvořené morphem **WebPageMorph** a nakonec status bar lišta.



Kresba 11.5: Sekvence vytvoření GUI

11.7 Formulářové prvky

K části formulářů pouze zmíním fakt, že Scamper zvládá teoreticky jak metodu GET, tak POST. Třídy formulářových prvků jsou **FormInput** (objekty pro tag **<input>** (kapitola 6.6.7)) a z něj odvozené **HiddenInput**, **RadioButtonInput**, **RadioButtonSetInput**, **SelectionInput**, **TextInput** a **ToggleButtonInput**.

Samostatnou částí je třída FormInputSet, která obsahuje odkazy na všechny vstupní elementy HTML formuláře. Má odkazy na browser, formulář i vstupy HtmlInput ze stromu dokumentu.

Metodou ***FormInputSet***>>*submit* se zavolá metoda ***Scamper***>>*submitFormWithInputs:url: method:encoding:*. Ta následně podle zadané metody formulář odešle serveru.

11.8 Shrnutí

Tato verze Scamperu má již poměrně pokročilé funkce pro načítání a zobrazování textu. Jeho nevýhodou je formátování HTML dokumentu pomocí textových morphů. To víceméně znemožňuje implementovat CSS box model a tím splnit zobrazovací nároky na současné prohlížeče. Výhodou je velká renderovací rychlost. Další kapitola podrobně rozebere moji práci na prohlížeči Scamper, představí některé zásadní změny v architektuře a pokusí se vysvětlit funkci kódu tak, aby byl pro případného zájemce o pokračování co nejlépe srozumitelný.

12 Další vývoj Scamperu

Prioritou mé práce bylo upravit Scamper tak, aby podporoval CSS box model, CSS visuální formátovací model, tabulky, formátovaný text a obrázky. Při analýze dosavadního kódu a CSS specifikace jsem došel k závěru, že vrstvy parser a formatter nelze použít. Nejdříve tedy proberu přehled změn, kterými prohlížeč prošel.

12.1 Přehled změn

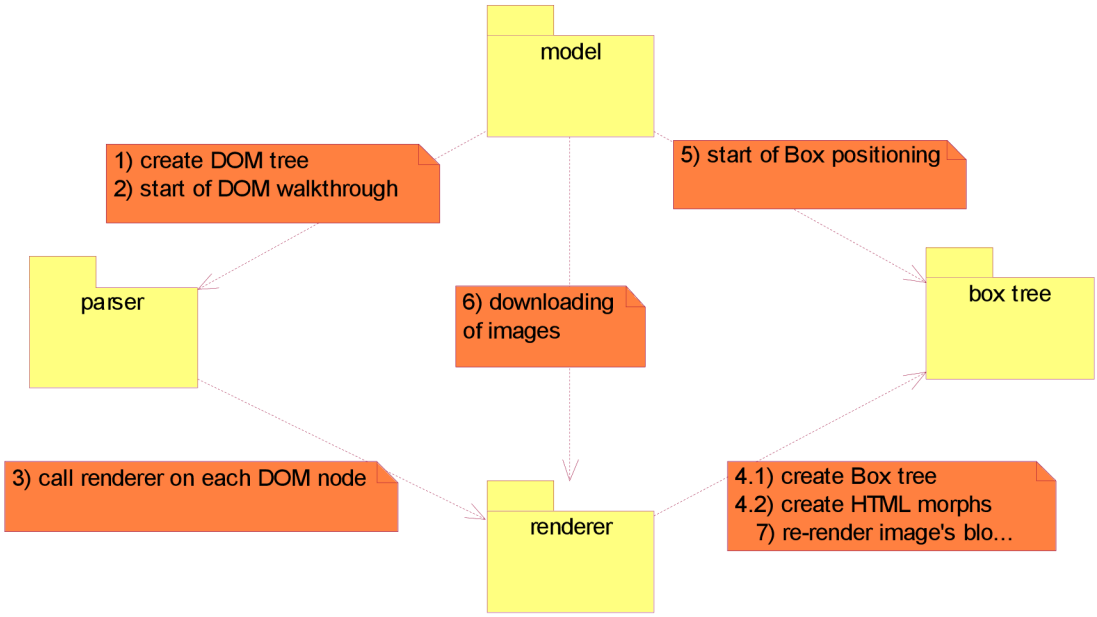
V první řadě si představíme novou architekturu Scamperu, jeho vrstvy a funkce. Dále letmo nahlédneme na obsah každé vrstvy

12.1.1 Nová architektura

Nejméně změn prodělala vrstva Model. Její hlavní třída **Scamper** se významně nezměnila.

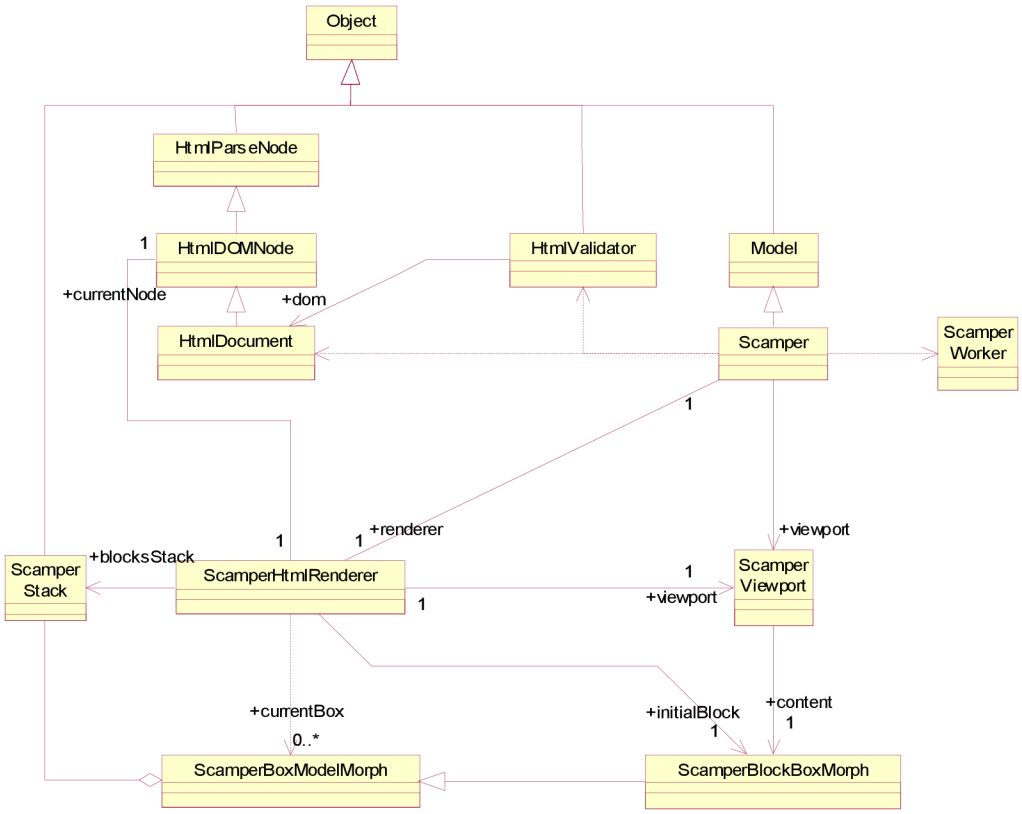
Dále obsahuje třídu **TableMatrix** (dědí ze třídy **Matrix**). Implementuje dynamické přidávání řádků a sloupců k matici.

ScamperDictionary, která dědí také ze tříd kolekcí, konkrétně ze třídy **Dictionary**. Obsahuje několik speciálních metod usnadňující práci při renderování. Další změny si vysvětlíme za pomoci schématu na Kresbě 12.1.



Kresba 12.1: Nová architektura Scamperu

Vrstva parser byla zcela nahrazena novým HTML&CSS parserem (viz kapitola 12.3). Vrstva formatter bylá také zcela nahrazena a její funkci přebírá nový renderer, o kterém pojednává kapitola 12.5. Celkový přehled o provázání objektů představuje Kresba 12.2



Kresba 12.2: Diagram objektů

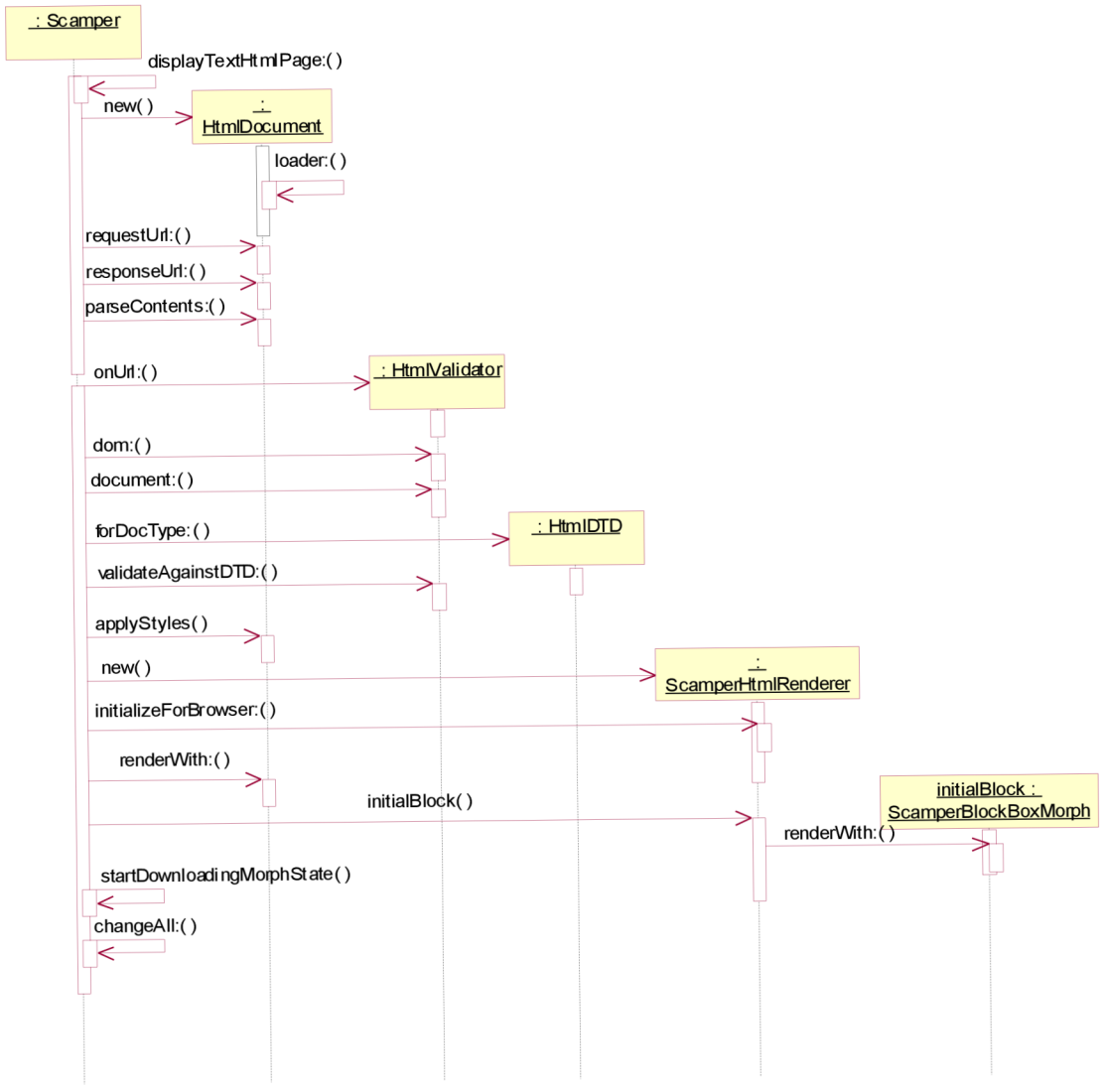
12.2 Načtení HTML stránky

Základní schéma načtení HTML stránky zůstává stejné jako na obr. Kresba 11.2. Výrazné změny jsou až v metodě ***Scamper***>>*displayTextHtmlPage:*

Činnost metody si vysvětlíme na sekvenčním diagramu (Kresba 12.3). V prvním kroku se vytvoří nový dokument, který naparsuje obsah MIME dokumentu v podobě čtecího streamu. Následně se dokument ověří proti DTD specifikaci. Dalším krokem je načtení a aplikování CSS stylů (v této fázi se neřeší zděděné a vícenásobné hodnoty).

Po tomto bodu lze přistoupit k první části renderování, tzn. je vytvořen renderovací objekt , který je poslán jako parametr kořeni DOM stromu. Tím se vytvoří strom CSS boxů (nemusí být 1:1). Kořeni tohoto nového stromu je opět poslán renderovací objekt.

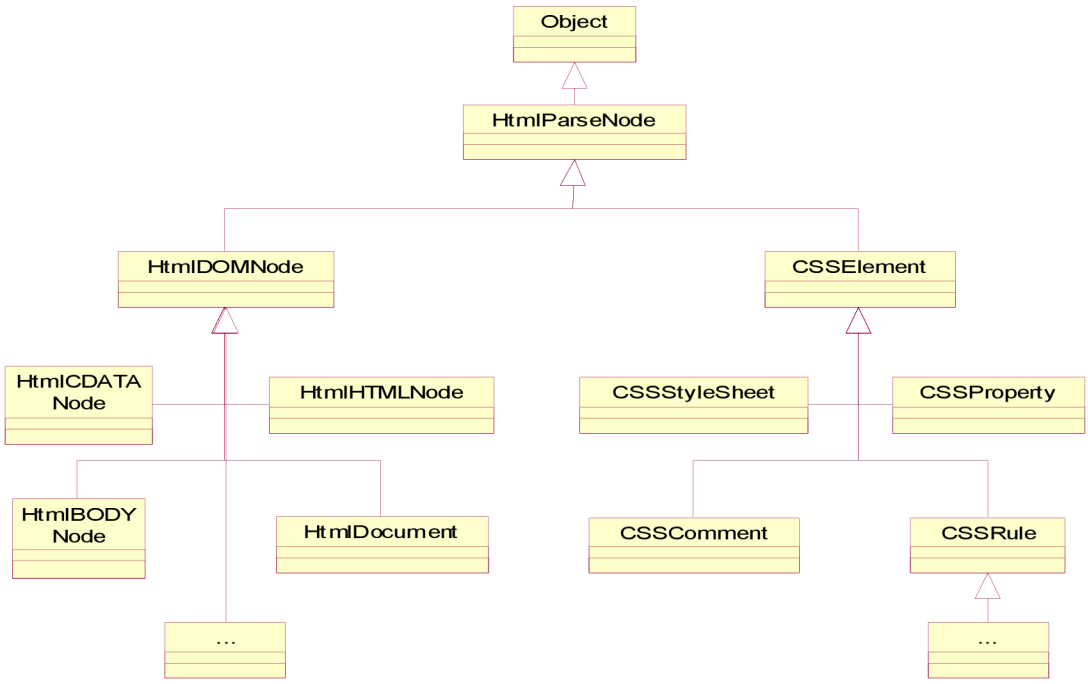
Nakonec jsou staženy obrázky a závislým objektům propagována změna.



Kresba 12.3: Hlavní renderovací metoda HTML stránky

12.3 Parser

Jak již bylo řečeno v úvodu této kapitoly, byl parser zcela nahrazen, a to balíkem dostupným ve *SqueakMap PackageLoaderu* pod názvem **HTML+CSS Validating Parser**[18]. Rozpoznává všechny HTML elementy, většinu elementů z CSS2.1 specifikace a také některé z CSS3. Zjednodušený diagram tříd ukazuje Kresba 12.3. Obsahuje dvě hlavní větve tříd, jedna pro DOM strom, kde každému HTML elementu odpovídá právě jedna třída. Každý z těchto DOM objektů obsahuje jako atribut kolekci **CSSRule** pravidel. Speciálním uzlem je **HtmlDocument**, který je kořenem DOM stromu a začíná na něm renderování (viz Kresba 12.4). Používá průchod stromem typu preorder.

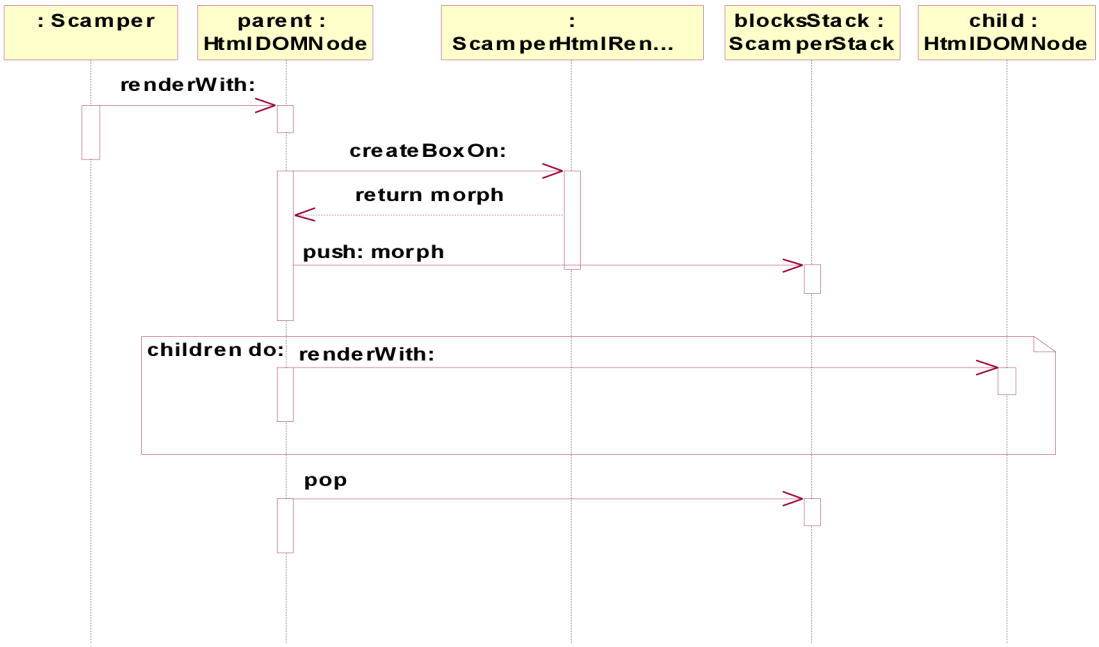


Kresba 12.4 Zjednodušený diagram tříd HTML+CSS Validating Parseru

Samotnou funkci vytváření stromu zde nebudu popisovat, neboť se přímo netýká mé práce. Zaměřím se především na průchod stromem a vytvoření stromu CSS boxů. Pro vysvětlení použiji opět sekvenční diagram (Kresba 12.5). Situace je zde poměrně jednoduchá. Objekt třídy **Scamper** zavolá metodu *renderWith: aRenderer* objektu **HtmlDocument**. Tento objekt použije objekt *aRenderer* k vytvoření morphu, určení jeho typu a načtení CSS vlastností. Následně tento morph uloží rendereru do zásobníku, který tím dostává možnost sledovat předky tohoto morphu (CSS boxu). K čemu zásobník slouží bude popsáno dále.

V dalším kroku volá objekt **HtmlDocument** stejnou metodu pro všechny své potomky, které používají buď defaultní metodu ze třídy **HtmlDOMNode**, nebo mají ve speciálních případech svoji přetíženou.

Na konci metody je box vytvořený pomocí tohoto objektu ze zásobníku opět vyjmut.



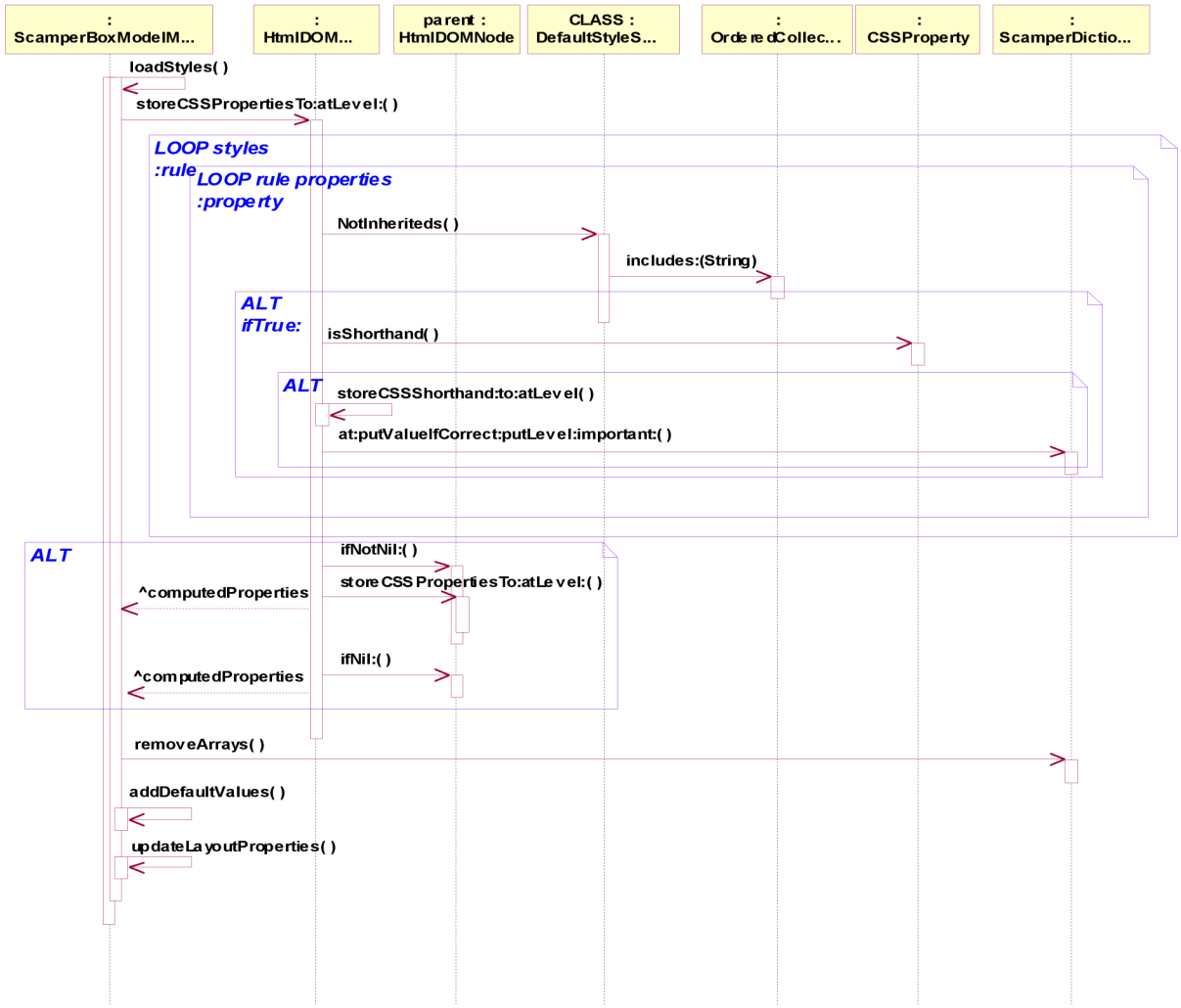
Kresba 12.5 Renderování DOM stromu

12.4 Načtení CSS pravidel do boxu

První akcí metody ***ScamperHtmlRenderer***>>*createBoxOn:* je vytvoření univerzálního CSS boxu. Ten ve své inicializaci provádí mimo jiné načtení CSS stylů ze svého DOM uzlu do slovníku computedProperties a chybějící vlastnosti doplní defaultními vlastnosti podle [19]. Odkaz na DOM uzel dostane při svém vytvoření. Situaci ukazuje Kresba 12.6.

DOM uzel provádí dvojitý cyklus. Vnější je na pravidla **CSSStyleRule**, které obsahují pole CSS vlastností **CSSProperty**. Vlastnost, která je přímo pro daný uzel, je přímo určena k uložení, je-li provedeno rekurzivní volání z nižšího uzlu, ověřuje se, zda je možné danou vlastnost zdědit. Dalším problémem jsou tzv. *shorthandy*, tedy vlastnosti, které sdružují víc hodnot (např *margin: 1px 2px* je složení vlastností *margin-top=margin-bottom=1px*; *margin-left=margin=right=2px*). Taková vlastnost je rozdělena a uložena jako několik dílčích vlastností. Vlastnosti jsou ukládány do speciálního slovníku **ScamperDictionary**, který mimo hodnoty vlastnosti uchovává také informaci o tom, zda nebyla vlastnost hodnoty **!important**, případně zda nebyla hodnota v nižším levelu **inherit** a podle toho vytvoří nebo přepíše pravidlo ve slovníku. Za dvojitým cyklem se rekurzivně volá stejná metoda (*storeCSSPropertiesTo:AtLevel:*) pro rodiče, pokud existuje.

Nakonec ještě odstraní z hodnot ve slovníku computedProperties pole a nechá jenom hodnoty. Na závěr načítání stylů ještě upraví vztahy hodnot *display, position* a *float* podle [20].



Kresba 12.6 Načtení CSS vlastností

Další kapitola se bude zabývat popisem vytváření stromu CSS box morphů ve třídě **ScamperHtmlRenderer**.

12.5 ScamperHtmlRenderer

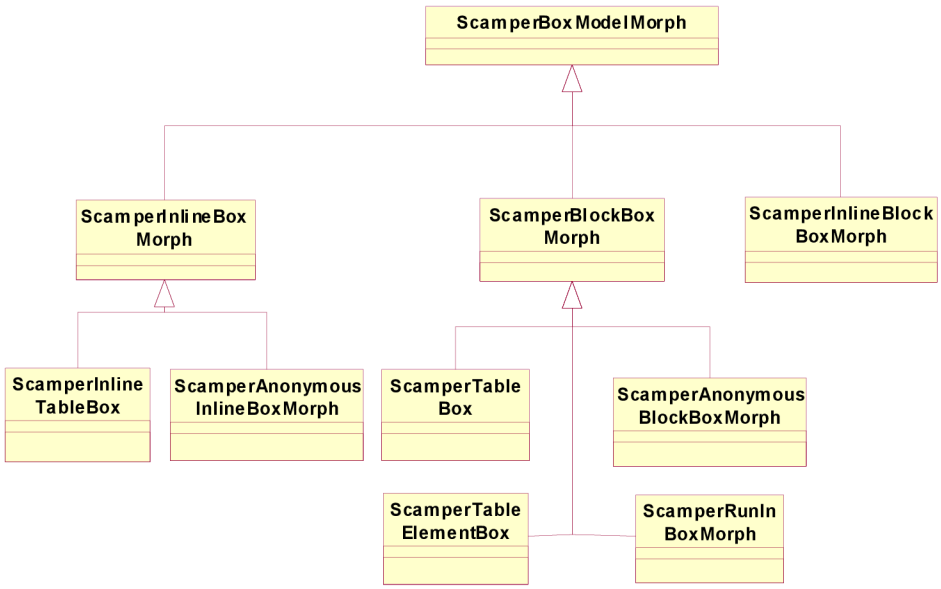
Tato třída má několik hlavních funkcí. V první řadě je to tvorba stromu CSS box morphů a pro listové elementy tvorba odpovídajícího HTML obsahu. Druhou funkcí je implementace kritické sekce pro přerenderování stránky po načtení obrázků. První funkci se budu věnovat v této kapitole, druhá bude popsána v kapitole 11.7.

Výchozí metodou rendereru je metoda ***ScamperHtmlRenderer***>>*createBoxOn: aDOMNode* (viz Kresba 12.5). Ta vytvoří jako první krok obecný CSS box morph, který se inicializuje a načte CSS vlastnosti z DOM uzlu (viz Kresba 12.6). Druhým krokem je konkretizace typu boxu v závislosti na mnoha parametrech, které jsou určeny specifikací dle [20] a [21]. Je to poměrně složitá procedura,

která se dělí na dvě základní části podle hodnoty *display*: Buď se pracuje s tabulkovými elementy, nebo s ostatními. Popis obou částí bude následovat později, nejprve si uvedeme diagram tříd CSS boxů.

12.5.1 Diagramy tříd CSS boxů

Jedná se o rozsáhlý diagram, který bylo nutné rozdělit na několik částí



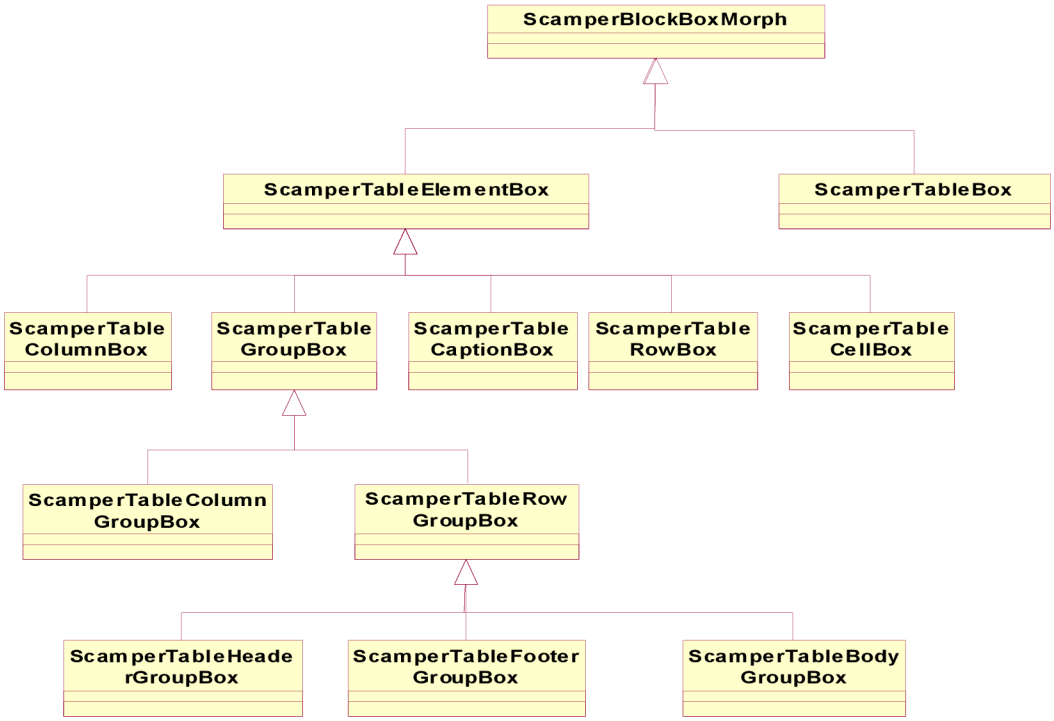
Kresba 12.7 Diagram tříd boxů 1

Na Kresbě 12.7 vidíme základní hierarchii boxů od obecného boxu. Hlavní třídy tvoří boxy typu *inline, inline-block* a *block*. Každý z těchto typů boxů se renderuje podle vlastních typů pravidel. Speciální případ tvoří tabulky a jejich elementy, které fungují z velké části jinak, nicméně některými svými vlastnostmi určují svůj druh také jako blokové (kromě inline-tabulek). Jejich diagram tříd ukazuje Kresba 12.8. Zde odpovídá třídá příslušnému HTML elementu, skupinové elementy mají speciální nadtřídu.

12.5.2 Určení typu netabulkového boxu

Zde je zcela zásadní pravidlo, že blokové elementy se formátují podle *blokového formátovacího kontextu* a inline elementy podle *inline formátovacího kontextu*. Blokový kontext znamená, že box může obsahovat buď **pouze** blokové, nebo **pouze** inline boxy. V inline kontextu by se měly do boxu přidávat pouze inline boxy (nicméně specifikace definuje chování, pokud tomu tak není).

Výchozí metodou pro netabulkové boxy je ***ScamperHtmlRenderer***>>*setBoxTypeNormalOn: aDOMNode*. Základním kritériem pro určení typu boxu je typ přímého předka, který se získá ze zásobníku (Kresba 12.5). V úvahu připadají následující možnosti **předka**:



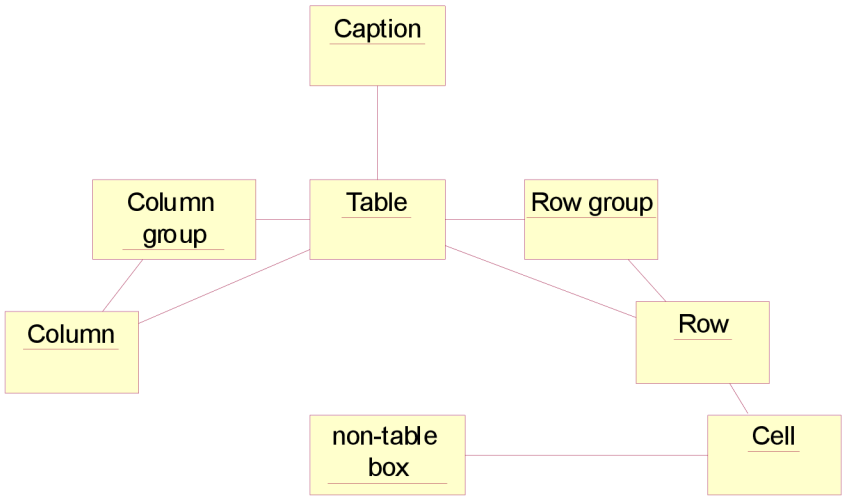
Kresba 12.8 Diagram tříd boxů 2

- Na zásobníku **žádný** není - vloží se do bloku **initialBlock**, což je výchozí blok pokrývající celou stránku a vytváří se při inicializaci **ScamperViewportu**.
- **Run-in** – může obsahovat pouze inline-elementy, pokud je aktuální box blokový (tzn. má hodnotu *display* blokovou*)*, je předek předělán na blokový.
- **Blokový** – záleží také na hodnotě *display* aktuálního bloku:
    - **Blokový** – pokud existuje prevSibling a ten je typu „run-in“, stane se prvním inline elementem tohoto bloku, ovšem pouze pokud už tento blok run-in potomka neobsahuje. Dále zkontroluje, jestli je dodrženo pravidlo blokového kontextu, v případě, že předek obsahuje inline boxy, jsou obaleny do tzv. *„anonymního blokového boxu*“.
    - **Inline** – kontroluje se pravidlo blokového kontextu. V případě, že předek obsahuje bloky, je obalen anonymním blokovým boxem.
    - **Čistý text** – je obalen *„anonymním inline boxem“.*
- **Inline** – čistý text je obalen anonymním inline boxem, ostatním je typ určen podle hodnoty *display*.
- **Table-element** – v tomto případě je element vložen do buňky tabulky podle algoritmu popsaném v následující kapitole.

Nakonec je box propojen se svým předkem a prevSiblingem, pokud ten existuje.

12.5.3 Určení typu tabulkového boxu

Vysvětlení tabulkových boxů je mnohem snažší. Logiku si vysvětlíme na Kresbě 12.9.



Kresba 12.9 Tabulková hierarchie

Nejvyšší vrstvou je tabulka, která může obsahovat boxy typu skupina sloupců, sloupec, nadpis, skupina řádků, nebo řádek, tzn. ty, se kterými je spojena čarou. Řádek může mít jako přímého předka buď skupinu řádků, nebo přímo tabulku atd. Je-li aktuální box takový, že pro aktuálního předka na zásobníku neexistuje žádná z těchto cest, je potřeba mezičlánky automaticky vytvořit. Jako příklad uvedu kód:

```
<table>text</table>
```

V tomto případě je mezi boxem pro tabulku a anonymním inline boxem pro text potřeba vytvořit minimálně boxy typu „cell“ a „row“.

12.5.4 Úprava rodičovského boxu

Změna rodičovského boxu přichází v případě, že je box vyjmut z normálního toku vlastností *position* nastavenou na *absolute*, nebo *fixed*. Pro hodnotu *fixed* se nastaví rodič **viewport**, což v důsledku znamená, že na něj nemá vliv scrollování. U hodnoty *absolute* se hledá nejbližší předek s hodnotou *position*={*absolute, relative, fixed*}. V případě, že se nenajde, je jako rodič určen **initialBlock**.

12.5.5 Vytvoření HTML elementu

V současné době podporuje tyto vizuální HTML elementy (VC):

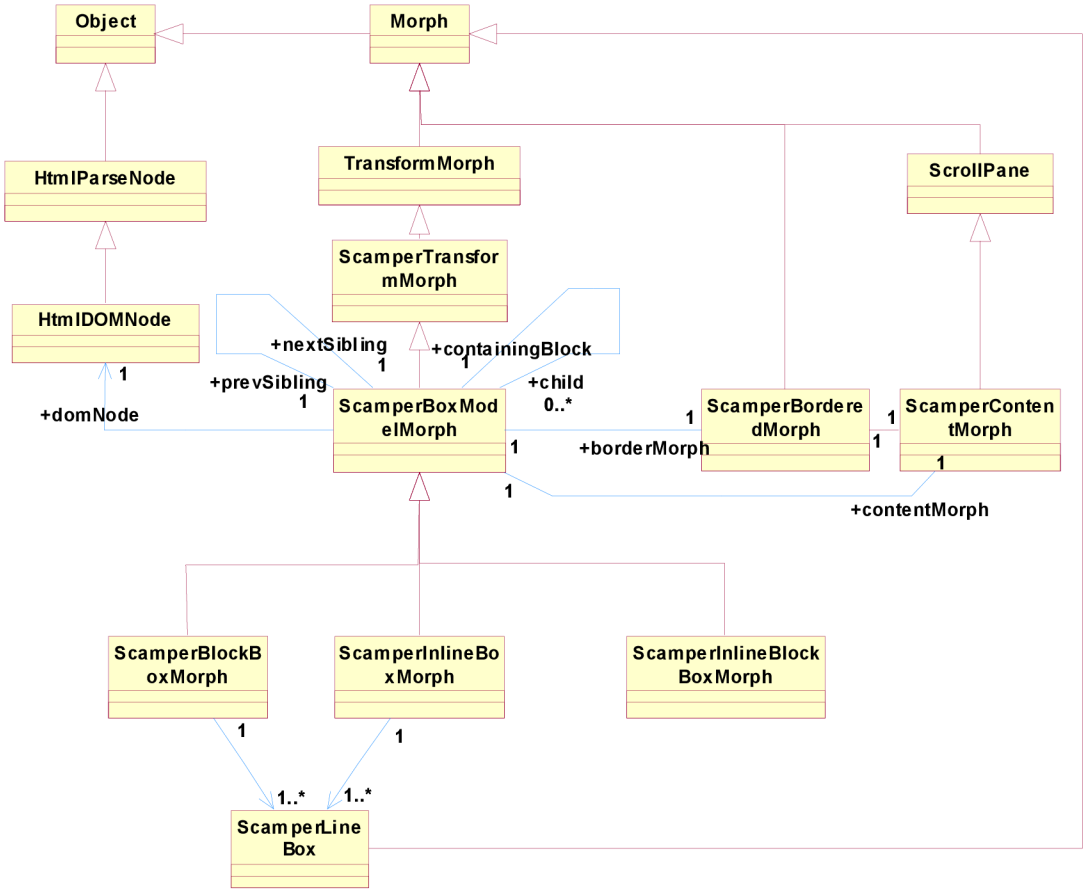
- **CDATA** – formátovaný text, objekt **TextMorph**
- **<img>** - obrázek, objekt **DownloadingImageMorph**

12.6 Renderování CSS boxů

Po dokončení minulé kapitoly jsme dostali strom boxů. Renderování probíhá z výchozího **initialBlock**u průchodem stromu typu preorder. Nejdříve se podíváme na diagramy tříd.

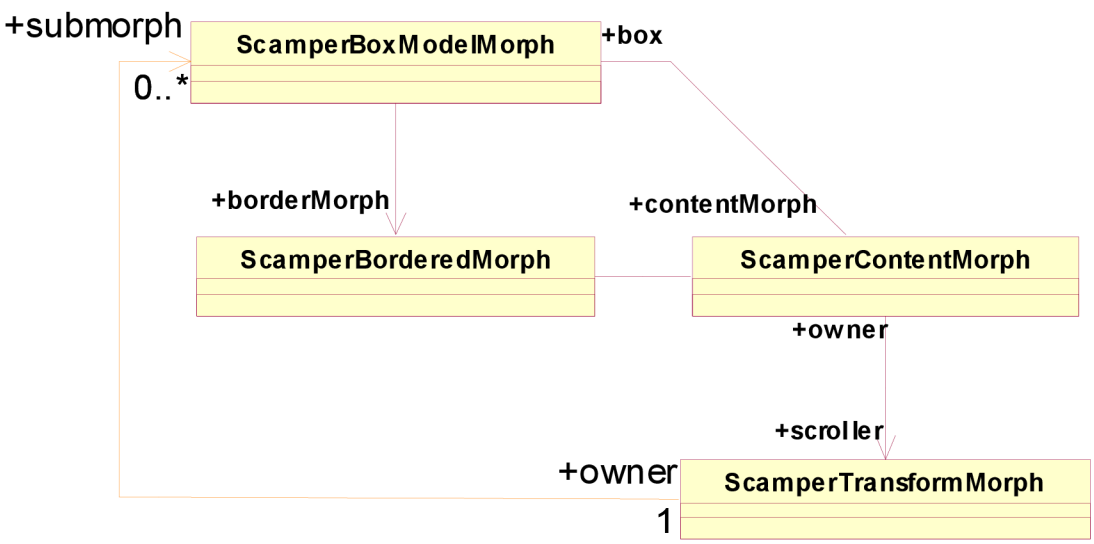
12.6.1 Základní popis

Prvním diagramem je diagram tříd (Kresba 12.10). Ukazuje dědičnosti základního boxu a jeho hlavních atributů až ke třídě **Object**. Modré šipky znázorňují vazby mezi objekty.



Kresba 12.10 Diagram tříd 3

Každý box má svůj DOM uzel, odkaz na předchozího a následujícího sourozence, rodiče a kolekci svých potomků.Dále má svůj morph na zobrazení rámečku a obsahový morph, což je scrollovací okno pro vnitřní obsah boxu (možnost scrollování je kvůli CSS vlastnosti *overflow*). Navíc blokové a inline-blokové boxy obsahují navíc kolekci tzv. line-boxů, které určují šířku a pozici vnitřních inline-boxů, pokud nějaké obsahuje. Detail hierarchie ukazuje diagram objektů na Kresbě 12.11.

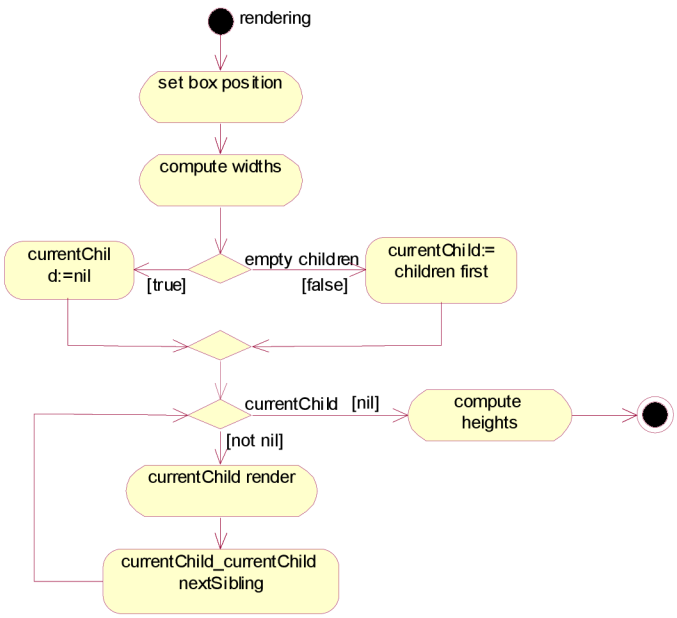


Kresba 12.11 Hierarchie objektů

Vzhledem k tomu, že **ScamperBoxModelMorph** a **ScamperTransformMorph** jsou třídy odvozené z **TransformMorph**, jsou souřadnice jejich potomků automaticky přepočítávány, tzn. platí pro ně, že jejich počátek souřadnic je umístěn v levém horním rohu rodičovského content morphu.

12.6.2 Blokové boxy

Hlavní funkce renderování blokových boxů si ukážeme na diagramu aktivit (Kresba 12.12):



Kresba 12.12: Metoda renderWith:

Tato funkce je v podstatě stejná i pro renderování inline boxů a spočívá ve čtyřech krocích:

- Nastavení pozice boxu vzhledem k rodičovskému boxu
- Vypočtení šířky bloku.
- renderování všech potomků
- vypočtení výšky bloku v závislosti na výšce potomků.

Určení pozice bloku

Blokové boxy jsou umístěny pod sebou jeden za druhým, tzn. horní hrana bloku je určena spodní hranou předchozího sourozence. Nemá-li takového sourozence, je umístěn na hodnoty PL@PT rodičovského content morphu.

Výpočet šířky bloku

1 Načtení vlastností *margin-left (ML)*, *margin-right (MR)*, *border-left-width (BLW)*, *border-right-width (BRW)*, *padding-left (PL)*, *padding-right (PR)* a *width (W)*. První dotaz je na kolekci **usedProperties**, tzn. už vypočtené hodnoty. Není-li tam hodnota nalezena, hledá se v **computedProperties**, která obsahuje všechny vlastnosti CSS 2.1 a jejich hodnoty. Hodnoty jsou přepočítány na pixely (viz. kapitola 12.6.5).

2 Načtení šířky nejbližšího blokového předka.

3 Další krok závisí na hodnotě vlastnosti *width:*

- *W='auto'*
  pokud jsou *ML,* nebo *MR* hodnoty *'auto',* je nastavena hodnota 0.
  W=(šířka předka) - (*ML+BLW+PL+PR+BRW+MR*)
- *W <> 'auto'*
  - *ML<>'auto' a MR<>'auto'*
    zde se nastaví jeden z marginů na 'auto', který to má být, závisí na orientaci bloku (CSS vlastnost *direction*)
  - hodnoty automatických marginů se dopočítají odečtením ostatních vlastností z bodu 1 od šířky předka. Jsou-li oba marginy automatické, je vnitřní blok vycentrován, tzn.*ML* a *MR* si hodnotu rozdělí napůl.

4 Vypočtené hodnoty jsou uloženy do **usedProperties** pro příští použití.

5 Šířka boxu se rovná hodnotě *ML+BLW+PL+W+PR+BRW+MR*.

6 X-ová pozice morphu pro rámeček se rovná hodnotě *ML*, jeho šířka *BLW+PL+W+PR+BRW*.

7 X-ová pozice morphu pro obsah se rovná hodnotě *ML+BLW*, jeho šířka *PL+W+PR*.

8 Pozice LineBoxu je PL@PT, šířka *W*.

LineBox reprezentuje jeden řádek a jsou podle něho umísťovány inline boxy potomků.

Výpočet výšky bloku

1 Načtení vlastností *margin-top (MT)*, *margin-bottom (MB*, *border-top-width (BTW)*, *border-bottom-width (BBW)*, *padding-top (PT)*, *padding-bottom (PB)* a height *(H)*. První dotaz je na kolekci **usedProperties**, tzn. už vypočtené hodnoty. Není-li tam hodnota nalezena, hledá se v **computedProperties**, která obsahuje všechny vlastnosti CSS 2.1 a jejich hodnoty. Hodnoty jsou přepočítány na pixely (viz. kapitola 11.6.5).
2 Pokud se některá z hodnot *MT*, *MB* rovná '*auto*', počítá se jako 0.
3 Výška, která není určena (tzn. *H='auto'*) se počítá následovně:
   - Když má box VC, pak je *H*=výška VC.
   - Když obsahuje inline boxy, je výška určena spodní hranou posledního LineBoxu.
   - Když obsahuje blokové boxy, je výška určena spodní hranou posledního blokového potomka.
4 Vypočtené hodnoty jsou uloženy do **usedProperties** pro příští použití.
5 Výška boxu se rovná hodnotě *MT+BTW+PT+H+PB+BBW+MB*.
6 Y- ová pozice morphu pro rámeček se rovná hodnotě *MT*, jeho výška *BTW+PT+H+PB+BBW*.
7 Y-ová pozice morphu pro obsah se rovná hodnotě *MT+BTW*, jeho šířka *PT+H+PB*.

12.6.3 Inline boxy

Hlavní renderovací funkce je totožná jako u blokových boxů (Kresba 12.12). Vnitřní funkce jednotlivých aktivit se však zásadně liší. Explicitně zadaná šířka bloku se ignoruje.

Určení pozice bloku

- Pokud byl box vložen na nový řádek, je jeho pozice určena pozicí posledního LineBoxu.
- Jinak záleží na tom, zda je jeho prevSibling prázdný:
   - Když ano, testuje se rodič boxu:
      - Inline rodič – pozice je 0@0.
      - Blokový rodič – pozice je dána pozicí posledního LineBoxu rodiče.
   - Když ne, měl by být inline a v tom případě je box vložen do stejného LineBoxu za prevSibling.

Výpočet šířky bloku

1 Načtení *ML, BLW, PL, PR, PRW, MR*. Hodnota *W* se **ignoruje**.
2 Hodnoty *ML* a *MR* nastavené na '*auto*' se ignorují (hodnota se rovná nula).

3 Výpočet W se provádí, pokud box obsahuje VC (tzn. je listovým elementem). Je-li tomu tak, porovná se šířka boxu s místem zbývajícím v aktuálním LineBoxu. Pokud se do něho nevejde, mohou nastat dvě možnosti:
   - VC lze rozdělit (tzn. je to text). V tom případě je box i s obsahem a všemi inline předky rozdělen na dvě části, první část vyplní zbytek místa v aktuálním LineBoxu a pro zbývající část je vytvořen nový LineBox. Platí pravidlo, že rozdělené boxy nemají okraje ani rámečky na konci starého a začátku nového řádku (LineBoxu).
   - VC nelze rozdělit. V tomto případě je vytvořena kopie tohoto boxu a rozděleni všichni inline předci, vytvořen nový LineBox, do kterého je nová hierarchie vložena. Současný box je následně rodičem smazán a výpočet zde končí.
4 Vypočtené hodnoty jsou uloženy do **usedProperties** pro příští použití.
5 X-ová pozice morphu pro rámeček se rovná hodnotě *ML*, X-ová pozice morphu pro obsah se rovná *ML*+*BLW*.
6 Pokud je box listem ve stromové hierarchii (tzn. má VC), počítají se šířky následovně:
   - Šířka boxu: *ML*+*BLW*+*PL*+šířka VC +*PR*+*BRW*+*MR*.
   - Šířka morphu pro rámeček: *BLW*+*PL*+šířka VC +*PR*+*BRW*.
   - Šířka morphu pro obsah: *PL*+šířka VC +*PR*.
   - Všem inline předkům je zvětšena šířka o šířku VC.

   Pokud není box listem, počítají se šířky obdobně, pouze se nezapočítává hodnota šířky VC (změna je propagována od listu).

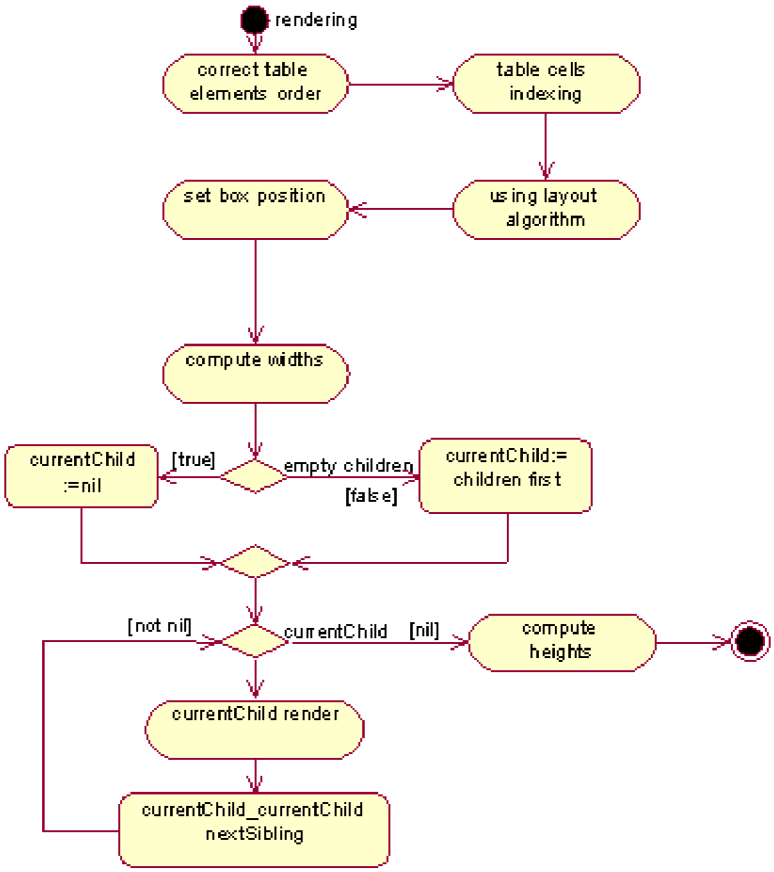
Výpočet výšky bloku

1 Načtení *BTW, BBW, PT, PB*, **ignoruje** se *MT, H, MB*.
2 Vypočtené hodnoty jsou uloženy do **usedProperties** pro příští použití.
3 Y-ová pozice boxu je dána aktuální pozicí mínus (*PT*+*BTW*)
4 Pokud je box listem ve stromové hierarchii (tzn. má VC), počítají se výšky následovně:
   - Výška boxu se rovná (výška VC)+*BTW*+*PT*+*PB*+*BBW*.
   - Y-pozice morphu pro rámeček je 0, jeho výška pak stejná jako výška boxu.
   - Y-pozice morphu pro obsah se rovná *BTW*, jeho výška *PT*+ (výška VC)+*PB*.
   - Pokud je třeba, zvětší se výška LineBoxu, ve kterém blok leží.
   - Všem inline předkům je zvětšena výška o výšku VC.
   - Pokud je třeba, jsou posunuty všechny následující LineBoxy.

   Pokud není box listem, počítají se výšky obdobně, pouze se nezapočítává hodnota výšky VC (změna je propagována od listu).

12.6.4 Tabulky

Renderování tabulek je zcela odlišné od ostatních boxů a nelze ho provést jedním průchodem skrz tabulkovou hierarchii. Diagram aktivit pro metodu ***ScamperTableBox***>>*renderWith*: znázorňuje Kresba 12.13 Ostatní tabulkové elementy odvozené ze třídy **ScamperTableElementBox**. používají buď původní metodu ***ScamperBlockBoxMorph***>>*renderWith:*, nebo vlastní mírně upravenou.



Kresba 12.13: Metoda renderWith: pro tabulky

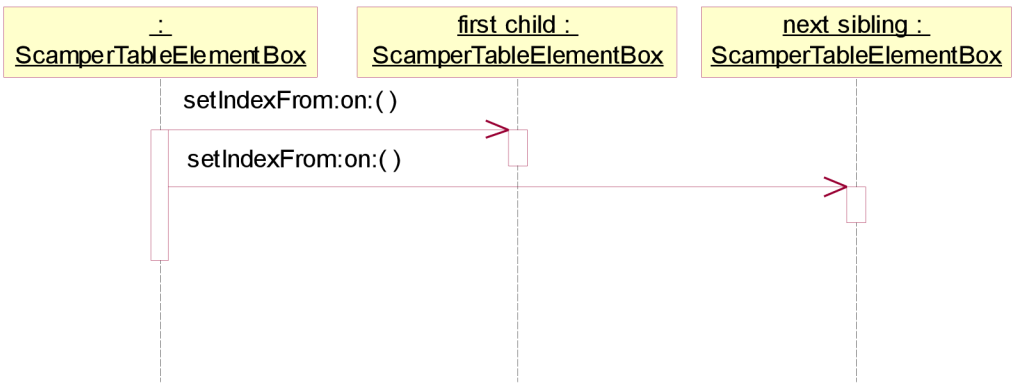
Nastavení správného pořadí elementů v tabulce

První aktivitou je zajištění správného pořadí elementů. Z historických důvodů je v tabulce napřed záhlaví, pak zápatí a nakonec tělo tabulky. Dále nemusí tabulka obsahovat některou, nebo žádnou řádkovou skupinu a nadpis tabulky může být zobrazen nad, nebo pod tabulkou. Úkolem je tedy nastavit pořadí tak, aby platilo pořadí: Nadpis (pokud má být nad tabulkou – CSS vlastnost *caption-side*)->sloupce->záhlaví->tělo tabulky->zápatí->nadpis (pokud má být pod tabulkou).

Indexace buněk

Indexování pokrývá první průchod stromovou strukturou tabulkových elementů. Informace o indexech buněk je ukládána do matice, která je atributem objektu tabulky a je reprezentována

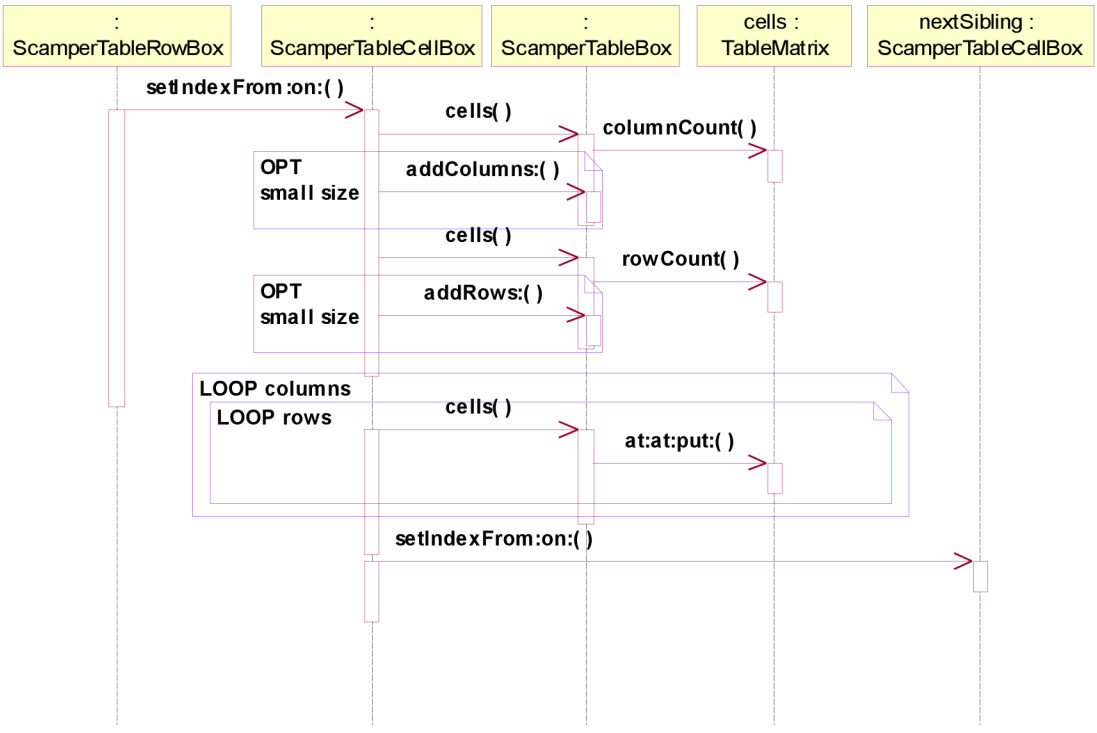
objektem třídy **TableMatrix**. Zjednodušenou sekvenci volání při indexování nebuňkových elementů ukazuje Kresba 12.14.



Kresba 12.14: Indexování nebuňkových elementů

V podstatě se jedná o to, aby se indexovácí funkce dostala ke všem buňkovým elementům. Je tedy volán první potomek a nextSibling. Parametry metody jsou bod, který určuje první index boxu v tabulce a ukazatel na objekt tabulky.

Zajímavější je samotná indexace buněk, tedy <td> a <th> elementů. Interakce mezi objekty je o něco složitější a zhruba ho popisuje Kresba 12.15.



Kresba 12.15: Indexování buněk

Pro přehlednost zde nejsou uvedeny všechny volání. V první řadě se kontroluje, jestli je daný index volný. V případě, že není, je možné posunout jednosloupcové buňky (atribut buňky *colspan*) na první volný sloupec ve stejném řádku. Následně je provedena kontrola velikosti matice a v

případě, že indexy buňky přesahují počet sloupců, nebo řádků matice, je matice zvětšena na velikost nejvyššího indexu buňky. Poté jsou odkazy na buňku uloženy do příslušných buněk matice a nakonec je zavolána stejná metoda pro indexaci sousední buňky.

Výpočet šířek

Pro šířky sloupců a celé tabulky je možné použít dva algoritmy (CSS vlastnost *table-layout*). První z nich je fixní, kde se šířka sloupců počítá pouze ze sloupcových elementů a prvního řádku tabulky. Tento algoritmus není plně implementován.

Druhým algoritmem, který je nastaven defaultně je automatický výpočet. Jeho přesná definice není v CSS specifikaci popsána, nicméně popisuje nezávazné doporučení, které jsem se snažil implementovat. Rozdělil jsem ho do pěti základních kroků:

1. Určení minimální (MinW) a maximální (MaxW) šířky každé buňky. Minimální šířka buňky se vypočítá podle nejširšího VC, jaký buňka obsahuje, s libovolným dělením, tzn. s libovolným počtem LineBoxů. V případě textu se hledá nejdelší slovo (co do počtu pixelů). Maximální šířka je dána nejdelší inline posloupností VC. V případě, že box buňky obsahuje inline boxy, počítá se součet všech VC na jednom LineBoxu, v případě, že obsahuje blokové boxy, počítá se nejširší blok pouze s jedním LineBoxem.
2. Určení minimální a maximální šířky každého sloupce. Určují se z maximálních MinW a MaxW všech buněk v daném sloupci. Používá matici **TableMatrix**.
3. Je-li šířka skupiny sloupců <colgroup> neautomatická, zvýší se MinW sloupců, které pokrývá tak, aby šířky souhlasily (v případě, že je větší, než dosavadní součet).
4. Zjištění součtů MinW a MaxW všech sloupců s ohledem na vlastnost *border-collapse*.
5. Nakonec se určí celková šířka tabulky a výsledné pozice a šířky všech sloupců. Algoritmus se liší na hodnotě W u tabulky:
   - *W='auto'* – celková šířka tabulky je určena jako větší z hodnot: šířka rodiče a součet minimálních šířek sloupců. Nicméně pokud je součet maximálních šířek sloupců menší, než šířka rodiče, použije se tato šířka
   - *W<>'auto'* – celková šířka tabulky je určena jako větší z hodnot: *W* a součet minimálních šířek sloupců.

   V případě nesrovnalostí je extra šířka rovnoměrně distribuována mezi sloupce.

Výpočet výšek

Výška tabulky se rovná součtu výšek všech řádků plus případné mezery mezi buňkami a rámečky. Výška buňky je určena nejvyšší buňkou v daném řádku.

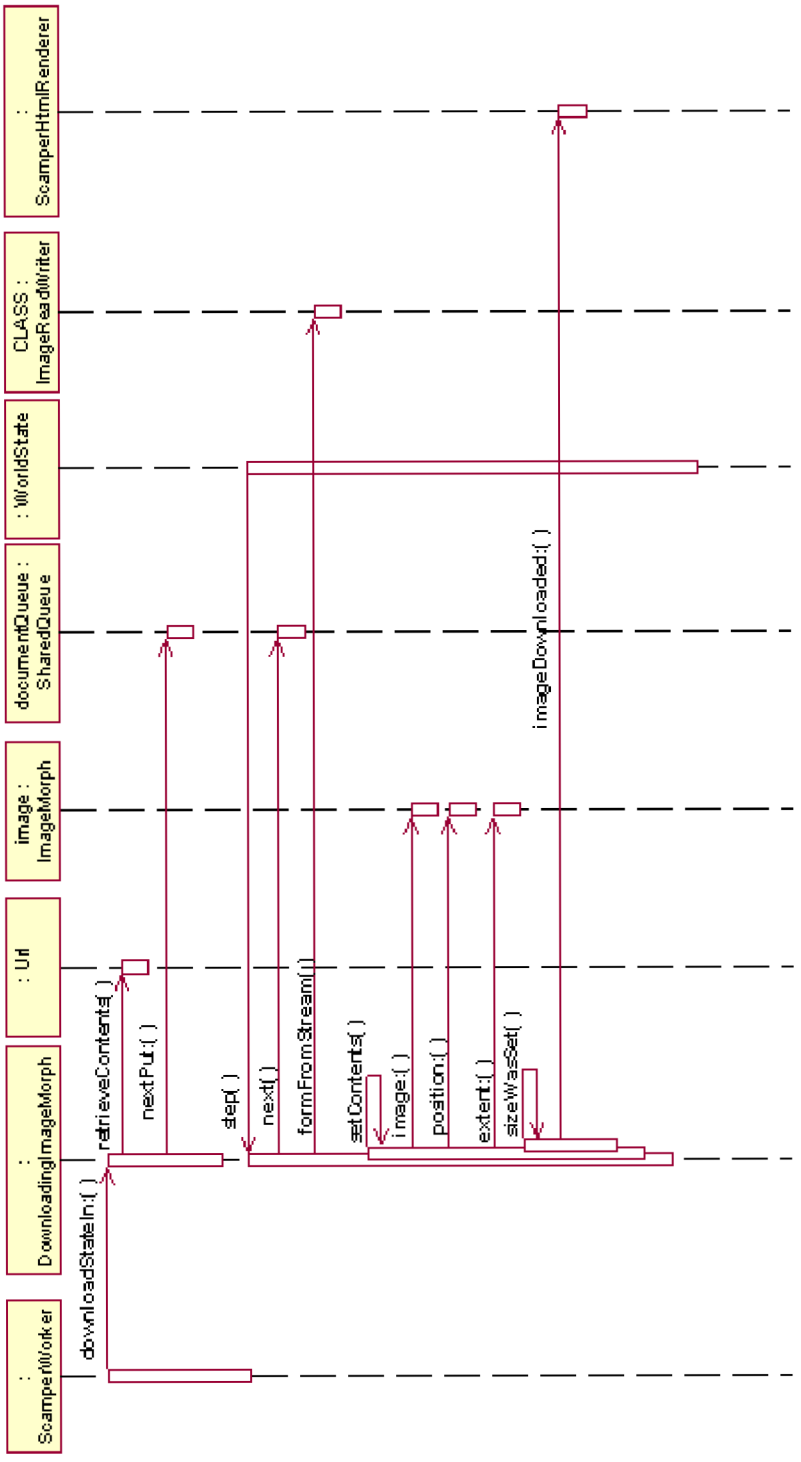
12.6.5 Převod jednotek na pixely

Jednotky použitelné v CSS pro zadání velikost jsou:

- **procenta** – vrácena je poměrná šířka k hodnotě W rodičovského bloku.
- **px** (pixely)
- **pt** (points), **in** (inches), **pc** (picas), **cm**, **mm** – hodnoty jsou přepočteny na pixely pomocí metod třídy **TextStyle**
- **em**, **ex** – poměrná velikost z šířky velkého M, příp.malého x aktuálního fontu. Velikost fontu se vypočítává pouze poprvé, používá se tzv. *lazy initialization*.

12.7 Zpracování obrázků

Obrázky jsou předem zakomponovány do stránky objektem třídy **DownloadingImageMorph**. Pokud nemá element explicitně nastavenou velikost obrázku, jemu nastavena velikost 20@20. Samotné stahování začíná až po skončení renderování stránky v metodě ***Scamper***>>*startDownloadingMorphState:*. Tam je pro každý obrázek vytvořen samostatný proces a zahájeno stahování obrázku. Seznam obrázků pro stažení obsahuje kolekce ***ScamperHtmlRenderer***>*incompleteMorphs*. Na Kresbě 12.16 si ukážeme sekvenci stahování jednoho obrázku. Nejdříve se stáhne MIMEDocument s obrázkem, který je dán do sdílené fronty. Tuto frontu pravidelně kontroluje metoda *step*, z níž pak dokument vyzvedne a stáhne samotný obrázek. Následně je obrázek upraven a pomocí objektu **ScamperHtmlRenderer** přerendrován blok, ve kterém obrázek leží. Bloky ležící za ním jsou pouze posunuty. Aby se vyhnulo kolizím, může přerenderovací akci provádět současně pouze jeden proces obrázku. K tomu účelu obsahuje metoda ***ScamperHtmlRenderer***>>*imageDownloaded:* inicializovaný semafor pro vzájemné vyloučení a kritickou sekci.



Kresba 12.16: Stahování obrázků

12.8 Hypertextové odkazy

Implementace hypertextových odkazů byla až překvapivě jednoduchá. Ve výchozím nastavení se pozice a tlačítka myši nesledují u žádného boxu, morphu pro rámeček, ani morphu pro obsah (ten pouze sleduje nastavení boxu). Je-li box odkazem, nastaví se mu atribut *url* a tím začne automaticky sledovat činnost myši, když se pohybuje uvnitř něho. Jsou to metody:

- ***ScamperBoxModelMorph***>>*handlesMouseDown:*
- ***ScamperBoxModelMorph***>>*handlesMouseOver:*

První metoda sleduje kliknutí a při něm se vyvolá metoda ***Scamper***>>*jumpToUrl: url*. Druhá metoda změní při přejetí nad odkazem ukazatel myši na ukazující ruku.

12.9 Nedostatky a návrhy na rozšíření

Jelikož se jedná o první verzi takto implementovaného prohlížeče, je zde nedostatků celá řada. Některé jsou ovšem poměrně zásadní. Prvním z nich je rychlost načítaní stránky. V případě rozsáhlých HTML stránek se obsah načítá a renderuje v řádu minut, což je zcela neúnostné. S tím souvisí i druhý problém, který je díky tomu také dosti závažný. Celé renderování probíhá v jenom cyklu Morphicu, což znamená, že po dobu načítání GUI Squeaku „zamrzne“.

12.9.1 Optimalizace

Zvýšení rychlosti půjde jistě docílit tím, že obsah blokových boxů na stejné úrovni půjde jistě renderovat souběžně v paralelních procesech. Co se týká vytváření stromové struktury boxů, která zabírá více než polovinu výpočetního času, bude zde situace nepoměrně složitější, nicméně jednou z možností je předělat přístup ke všem položkám ve slovníku ze stringu na symbol, další možností je nepřidávání všech defaultních vlastností do každého boxu, ale v případě absence se bude prohledávat slovník ve třídě **DefaultStyleSheet**, taktéž s přístupem k prvkům přes symboly. Dále se vytváří každý box v podstatě 2x, což jistě nemalou měrou zpomaluje. Optimalizace tohoto druhu nicméně bude vyžadovat rozsáhlou refaktorizaci kódu.

Třetím rozsáhlým krokem musí nutně být provádění renderování ve více cyklech Morphicu. K tomu slouří metoda *step*, možným řešením bude použití sdílených front typu **SharedQueue.**

12.9.2 Možná rozšíření

V první řadě bude třeba dopsat podporu všem implementovaným funkcím třídy Scamper. Těmi jsou například metody na odesílání formulářů, tzn. implementovat formulářové komponenty. Dále je možné jen mírnou modifikací zprovoznit metody na zobrazení stránek s flash obsahem.

Co se týká rozšíření v oblasti CSS specifikace, chybí zde podpora plovoucích bloků, které jsou nyní renderovány jako blokové. Dále zde chybí podpora dynamických vlastností jako např. pseudoelement *:hover*. A nakonec v oblasti HTML je potřeba doplnit implementaci chybějících elementů.

12.10 Shrnutí

V této kapitole jsem detailně rozebral fungování nové implementace Scamperu s návazností na původní verzi. Oproti stávající verzi používá nový parser podporující specifikaci CSS 2.1 a částečně CSS 3. Ten vytvoří stromovou strukturu DOM uzlů, jichž se používá k vygenerování stromové struktury morphů představujících CSS boxy. Dále je renderován strom boxů, tzn. každému boxu jsou přiřazeny odpovídající šířky včetně marginů, rámečků atd. Nakonec jsou staženy obrázky a na základě jejich velikosti se přerenderovává blok, ve kterém se nachází.

13 Závěr

V této práci byly rozebrány historické i dnešní prohlížeče a jádra, na kterých jsou ty dnešní postaveny. Ukázali jsme si základní vědomosti, které jsou nutné pro realizaci webového prohlížeče, jako je protokol HTTP a struktura MIME a také stručnou syntaxi jazyka HTML verze 4.01 a CSS2.1. Dále byl vysvětlen rozdíl mezi jazykem HTML a jeho nástupcem XHTML. V další části tohoto dokumentu byl probrán současný stav webového prohlížeče Scamper, tzn. jeho funkcionalita, ale také jeho nedostatky. Poslední kapitola se detailně věnovala přepracování prohlížeče, které jsem navrhnul a implementoval v rámci této diplomové práce

Zároveň byly rozebrány nedostatky této implementace, zejména v oblasti rychlosti a efektivity renderování a navrhnuta řešení pro jejich odstranění. Také jsem navrhl pro případné zájemce některé možnosti rozšíření prohlížeče, i když jich je nepochybně celá řada.

Tento projekt jsem si vybral z důvodu mých sympatií k vývojovému prostředí Squeak Smalltalku a přání nějak přispět do jeho vývoje. Vybral jsem si webový prohlížeč z důvodu, že stávající verze zakomponovaná do image nebere v potaz CSS model a zobrazované stránky zdaleka neodpovídají požadovanému vzhledu. Oproti jiným prohlížečům má Scamper tu výhodu, že jednotlivé elementy jsou reprezentovány standardními morphy, se kterými lze jinými aplikacemi libovolně manipulovat a využívat pro vlastní účely. Například aplikace mohou generovat HTML výstup, rovnou je zobrazovat a s výsledkem dále pracovat.

Literatura

[1] Wikipedie: Otevřená encyklopedie: *Webový prohlížeč* [online]. c2007 [citováno 24. 12. 2007]. Dostupný z WWW: <http://cs.wikipedia.org/w/index.php?title=Webov%C3%BD_prohl%C3%AD%C5%BEe%C4%8D&oldid=2043445> .

[2] Wikipedie: Otevřená encyklopedie: *Hypertext* [online]. c2007 [citováno 24. 12. 2007]. Dostupný z WWW: <http://cs.wikipedia.org/w/index.php?title=Hypertext&oldid=2027354>

[3] BERNERS-LEE, Tim. *The WorldWideWeb browser* [online]. [cit. 2007-12-25]. Dostupný z WWW: <http://www.w3.org/People/Berners-Lee/WorldWideWeb.html>

[4] WILSON, Brian. *Browser History: Mosaic* [online]. c1996-2003 [cit. 2007-12-25]. Dostupný z WWW: <http://www.eskimo.com/~bloo/indexdot/history/mosaic.htm>

[5] KODÝTEK, Pavel. *Historie Internetu* [online]. 2006 [cit. 2007-12-25]. Dostupný z WWW: <http://www.webdesign.paysoft.cz/clanky/2006/historie-internetu/>

[6] Wikipedia, The Free Encyclopedia: *List of web browsers* [online]. c2007 [cit. 2007-12-25]. Dostupný z WWW: <http://en.wikipedia.org/w/index.php?title=List_of_web_browsers&oldid=179801950

[7] Wikipedia, The Free Encyclopedia: *Comparison of web browsers* [online]. c2007 [cit. 2007-12-25]. Dostupný z WWW: <http://en.wikipedia.org/w/index.php?title=Comparison_of_web_browsers&oldid=180108810>.

[8] MICROSOFT CORPORATION. *MSHTML Reference* [online]. c2007 [cit. 2007-12-25]. Dostupný z WWW: <http://msdn2.microsoft.com/en-us/library/aa741317(VS.85).aspx>

[9] *Gecko:Home Page* [online]. 2007 , 19:00, 26 July 2007 [cit. 2007-12-26]. Dostupný z WWW: <http://wiki.mozilla.org/Gecko:Home_Page>

[10] *Market share for browsers, operationg systems and search engines* [online]. [2007] [cit. 2007-12-26]. Dostupný z WWW: <http://marketshare.hitslink.com/report.aspx?qprid=0>

[11] *HTML 4.01 / XHTML 1.0 Reference* [online]. c1999-2007 [cit. 2007-01-01]. Dostupný z WWW: <http://www.w3schools.com/tags/ref_byfunc.asp>

[12] JANOVSKÝ, Dušan. *HTML příručka, přehled HTML tagů* [online]. 2007 [cit. 2007-01-01]. Dostupný z WWW: <http://www.jakpsatweb.cz/html/>

[13] HRUŠKA, Tomáš, Prof. Ing., CSc. *Internetové aplikace (WAP) I. : část Internet a WWW*. FIT VUT Brno : [s.n.], 2006. 148 s

[14] HRUŠKA, Tomáš, Prof. Ing., CSc. *Internetové aplikace (WAP) II. : část SGML, HTML, CSS, DOM*. FIT VUT Brno : [s.n.], 2006. 204 s

[15] *Scamper* [online]. [2006] , 5:03 pm UTC on 31 October 2006 [cit. 2008-01-03]. Dostupný z WWW: <http://wiki.squeak.org/squeak/14>

[16] Čáň, Jakub, ŠLEMR, Martin. *Squeak Traits*. FIT VUT Brno : [s.n.], 2007. 9 s

[17] JANOUŠEK, Vladimír, Ing., Ph.D. *Smalltalk IST : studijní opora*. FIT VUT Brno : [s.n.], 2006. 65 s

[18] BLANCHARD, Todd. *Squeak Source : HTML & CSS Validating Parser* [online]. 15 April 2006 7:02:15 am [cit. 2008-04-23]. Dostupný z WWW: http://www.squeaksource.com/@agC5lE22ScFYD_LD/GhDs-yl1

[19] BOS, Bert, et al. *Full property table* [online]. MIT, ERCIM, Keio, c2006 [cit. 2008-04-23]. Dostupný z WWW: http://www.w3.org/TR/CSS21/propidx.html

[20] BOS, Bert, et al. *Visual formatting model* [online]. MIT, ERCIM, Keio, c2007 [cit. 2008-04-23]. Dostupný z WWW: http://www.w3.org/TR/CSS21/visuren.html

[21] BOS, Bert, et al. *Tables* [online]. MIT, ERCIM, Keio, c2007 [cit. 2008-04-23]. Dostupný z WWW: http://www.w3.org/TR/CSS21/tables.html

[22] BOS, Bert, et al. *Cascading Style Sheets Level 2 Revision 1 (CSS 2.1) Specification : W3C Candidate Recommendation 19 July 2007*., c2006. Dostupný z WWW: <http://www.w3.org/TR/CSS21/css2.zip>

[23] BOS, Bert, et al. *Box model* [online]. MIT, ERCIM, Keio, c2007 [cit. 2008-04-23]. Dostupný z WWW: http://www.w3.org/TR/CSS21/box.html

Seznam použitých zkratek a symbolů

Zkratka	*Význam*	*Zkratka*	*Význam*
ML	margin-left	**MT**	margin-top
MR	margin-right	**MB**	margin-bottom
BLW	border-left-width	**BTW**	border-top-width
BRW	border-right-width	**BBW**	border-bottom-width
PL	padding-left	**PT**	padding-top
PR	padding-right	**PB**	padding-bottom
W	width	**H**	height
VC	visual content (kap. 12.5.5)		

- nextSibling – element ve stromu bezprostředně následující po aktuálním
- prevSibling – element ve stromu bezprostředně předcházející aktuální