Euler - průvodce v češtině

Euler - Czech guide

Zuzana Kreizlová

Bakalářská práce
2007

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav aplikované informatiky
akademický rok: 2006/2007

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Zuzana KREIZLOVÁ**
Studijní program:	**B 3902 Inženýrská informatika**
Studijní obor:	**Informační technologie**
Téma práce:	**Euler – průvodce v češtině**

Zásady pro vypracování:

1. Proveďte literární rešerši týkající se tvorby web stránek.
2. Seznamte se s prací v programu Euler.
3. Vypracujte č4. eského průvodce ve formě www stránek.
5. Vytvořte také prezentaci pro MS PowerPoint, která bude stručně demonstrovat základní možnosti programu Euler.
7. Vámi vytvořené studijní podklady umístěte na samostatné CD-ROM jako přílohu bakalářské práce.

Rozsah práce:

Rozsah příloh:

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

1. Broža P. (2000): Programování www stránek pro úplné začátečníky. Computer Press, Praha, ISBN 80-7226-278-5
2. Kristián, P. (2001): Návrh webu a Frontpage 2000. UNIS Publishing, ISBN 80-86097-57-9
3. Kučera, M. (2001): HTML - tipy a triky od profesionálů. UNIS Publishing, ISBN 80-86097-64-1
4. Niederst, J. (1999): Web Design in a Nutshell. O'Reilly, USA, ISBN 1-56592-515-7.
5. http://mathsrv.ku-eichstaett.de/MGF/homes/grothmann/euler/german/index.html (Euler Dokumentation)

Vedoucí bakalářské práce: **Ing. Karel Perůtka**
Ústav řízení procesů

Datum zadání bakalářské práce: **13. února 2007**

Termín odevzdání bakalářské práce: **24. května 2007**

Ve Zlíně dne 13. února 2007

prof. Ing. Vladimír Vašek, CSc.
děkan

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky
L.S.

doc. Ing. Ivan Zelinka, Ph.D.
ředitel ústavu

ABSTRAKT

Tato bakalářská práce se zabývá realizací českého průvodce k programu Euler, který je volně dostupnou alternativou komerčního softwaru Matlab. Hlavním cílem je vytvořit informačně uspokojující a uživatelsky příjemné www stránky, které obsahují překlad větší části originální dokumentace k programu. Druhou částí je vytvoření prezentace pro Microsoft PowerPoint, jejímž účelem je představit program Euler a stručně demonstrovat jeho základní možnosti. Obě části mají sloužit případným zájemcům o práci s tímto programem k základní orientaci v programu a poskytnout dostatečný základ pro efektivní práci s tímto softwarem.

Klíčová slova: Euler, Matlab, internet, tvorba www, HTML, CSS, XML, XHTML, PHP

ABSTRACT

This bachelor thesis deals with the realization of Czech guide for Euler program. Euler is freely available alternative software for commercial MATLAB software. The main aim was to create user-friendly WWW pages with satisfactory amount of information which contain almost all information from original documentation for the program. In second part of this work, the MS PowerPoint presentation was created. Its purpose is to introduce the Euler program and shortly demonstrate its basic potential. Both parts are to be instrumental for appropriate persons interested in work with this program with basic orientation in the program and to afford sufficient base for effective work with this software.

Keywords: Euler, Matlab, Internet, programming of www pages, HTML, CSS, XML, XHTML, PHP

Na tomto místě bych ráda poděkovala Ing. Karlu Perůtkovi za cenné rady, podněty a připomínky, které nemalým dílem přispěly ke konečné podobě této práce. Také děkuji Ing. Tomáši Dulíkovi za odbornou výuku tvorby webových stránek, která mi pomohla rozšířit znalosti týkající se této problematiky a stala se základem, ze kterého jsem vycházela při tvorbě této práce.

Dále bych chtěla poděkovat svým přátelům za drobné, ale velmi užitečné rady týkající se tvorby webových stránek. V neposlední řadě děkuji také své rodině za všestrannou podporu, která rovněž přispěla ke vzniku této práce.

Motto

„

Překážky jsou ony obávané věci, které spatříte, když odvrátíte pohled od svého cíle

“

HENRY FORD (1863 – 1947)

Prohlašuji, že jsem na bakalářské práci pracovala samostatně a použitou literaturu jsem citovala. V případě publikace výsledků, je-li to uvolněno na základě licenční smlouvy, budu uvedena jako spoluautorka.

Ve Zlíně 28.5.2007 …….

Zuzana Kreizlová

OBSAH

OBSAH **6**

ÚVOD **8**

I. **9**

TEORETICKÁ ČÁST **9**

1 TVORBA WEBOVÝCH STRÁNEK **10**

1.1 ÚVOD 10

1.1.1 Technologie používané pro tvorbu webových stránek *10*

1.2 ZNAČKOVACÍ JAZYKY 14

1.2.1 HTML *14*

1.2.2 XML *22*

1.2.3 XHTML *28*

1.3 KASKÁDOVÉ STYLY 30

1.4 SKRIPTOVACÍ JAZYKY 35

1.4.1 JavaScript *36*

1.4.2 PHP *39*

II. **44**

PRAKTICKÁ ČÁST **44**

2 SEZNÁMENÍ S PROGRAMEM EULER **45**

2.1 INSTALACE 45

2.2 SPUŠTĚNÍ A UKONČENÍ PROGRAMU 46

2.3 PRÁCE S PROGRAMEM 47

3 UKÁZKOVÉ PŘÍKLADY PRÁCE V PROGRAMU EULER **49**

3.1 PŘIŘAZENÍ HODNOTY PROMĚNNÉ A JEDNODUCHÉ VÝPOČTY 49

3.2 OPERACE S VEKTORY 50

3.3 OPERACE S MATICEMI 51

3.4 VYKRESLOVÁNÍ GRAFŮ 52

3.4.1 2D grafy *52*

3.4.2 3D grafy *54*

3.5 STANDARDNÍ PŘÍKAZY A FUNKCE EULERU 56

3.5.1 Standardní příkazy *56*

3.5.2 Standardní funkce *57*

4 TVORBA PRŮVODCE K PROGRAMU EULER 58

4.1 PŘÍPRAVA PRÁCE 58

4.1.1 Výběr technologií k tvorbě stránek 58

4.1.2 Výběr softwarových prostředků 58

4.2 PRÁCE NA VLASTNÍ TVORBĚ HTML PRŮVODCE K PROGRAMU EULER 59

4.2.1 Grafický návrh stránek 60

4.2.2 Struktura stránek 60

4.2.3 Koncepce stránek 62

4.2.4 Ukázka zdrojového kódu stránky 63

4.2.5 Umístění stránek na serveru 67

ZÁVĚR 68

CONCLUSION 69

SEZNAM POUŽITÉ LITERATURY 70

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK 71

SEZNAM OBRÁZKŮ 73

ÚVOD

Technický pokrok provází vývoj lidské civilizace po celou dobu její existence. Už od pradávna hledá člověk způsoby, jak si zjednodušit práci a posunout hranice svých možností. Postupem času se tento vývoj přesouvá stále více z oblasti čistě fyzické do roviny duševní. Tak jako kdysi jednoduché pracovní nástroje, sloužící k usnadnění každodenních úkolů, vznikají dnes nástroje v podobě programů, určené k mnoha různým účelům ve všech oblastech lidské činnosti. Rozvoj těchto nástrojů je úzce spjatý s vývojem počítačů a také s internetem.

Na počátku vývoje sítě, která dnes spojuje celý svět, stál armádní projekt, jehož cílem bylo vytvořit fungující informační systém, který by nebyl řízen z jednoho centra. Tak vznikla základní myšlenka současného internetu, který funguje na principu existence mnoha na sobě nezávislých počítačů, z nichž každý je schopen samostatné existence. Současnou podobu internetu asi nejvíce ovlivnil vědec Tim Berners-Lee, který koncem 80. let 20. století začal hledat lepší způsob komunikace se svými kolegy prostřednictvím počítače. Výsledkem snažení jeho týmu bylo vytvoření definic hypertextového jazyka HTML a přenosového protokolu hypertextu HTTP, které jsou základy internetu, jak ho známe dnes.

Jednou z největších výhod internetu je to, že umožňuje komunikaci mezi lidmi z celého světa, nezávisle na tom, kde se nacházejí. Vzniká tak možnost spolupráce na různých projektech, z nichž mnohé jsou ve své konečné podobě dostupné zdarma každému, kdo hledá vhodné programové nástroje pro práci v dané oblasti. Tyto volně dostupné programy se nazývají freeware a jsou šířeny podle GNU General Public License (GPL), která mimo jiné umožňuje další vývoj těchto programů a jejich přizpůsobování pro konkrétní využití.

Takovým programem je i Euler, který je volně dostupnou variantou komerčního softwaru Matlab. Hlavním cílem této práce je vypracování informačně dostačujícího a esteticky příjemného průvodce ve formě www stránek, který případným zájemcům o práci s tímto programem usnadní orientaci v programu a poskytne dostatečný informační základ pro jeho efektivní využití. Dalším bodem mé práce je vytvoření prezentace pro MS PowerPoint, která bude stručně demonstrovat základní možnosti programu Euler.

I. TEORETICKÁ ČÁST

1 TVORBA WEBOVÝCH STRÁNEK

1.1 Úvod

Internet představuje v současnosti jedno z nejvýznamnějších médií. Tato síť, která dnes spojuje celý svět, se vyvinula v průběhu několika let z původního armádního projektu, který propojoval důležité počítače armády, vlády a několika univerzit ve Spojených Státech. K původní nepříliš rozsáhlé síti se postupně připojovaly další organizace na ostatních kontinentech a síť rostla rychlostí, kterou nikdo z tvůrců původního projektu nepředpokládal. Jen v letech 1983 až 1992 vzrostl počet počítačů připojených k síti na tisícinásobek původního počtu.

K rozvoji internetu značně přispěl značkovací jazyk HTML, který byl společně s protokolem HTTP vyvinut Timem Berners-Lee a jeho kolegy v roce 1989 ve švýcarském institutu pro jaderný výzkum CERN. Do roku 1993 byl internet doménou akademických a vědeckých pracovišť. S nástupem grafických prohlížečů začal být využíván také pro komerční a soukromé účely. Dnes představuje internet běžnou součást každodenního života a je využíván širokou veřejností.

1.1.1 Technologie používané pro tvorbu webových stránek

Internet je využíván jak ke komerčním účelům různými firmami, tak i k nekomerčním účelům, jako je například prezentace osobních stránek. K prezentaci informací na internetu je nezbytná znalost základních technologií tvorby webových stránek. V současné době existuje stále větší množství serverů zaměřených na tuto problematiku. Kromě základních informací, jakými jsou například možnosti jednotlivých jazyků, jejich syntaxe nebo způsob interpretace webovým prohlížečem, poskytují tyto servery také množství cenných rad od zkušenějších tvůrců webových stránek, které pomáhají začátečníkům vyhnout se zbytečným chybám, což značně usnadňuje práci.

Internetové stránky je možné rozdělit na dvě základní skupiny podle složitosti jejich koncepce a použitých technologií. První skupinou jsou jednoduché stránky, které jsou vytvořeny s využitím kombinace značkovacího jazyka HTML a kaskádových stylů CSS. Tyto jazyky postačí k tvorbě stránek menšího rozsahu (např. osobní stránky). Druhou skupinou jsou tzv. dynamické stránky, využívající kromě zmíněných jazyků také

skriptovací jazyky, nejčastěji JavaScript v kombinaci s jazykem PHP nebo ASP. Tyto technologie jsou vhodné k tvorbě složitějších a rozsáhlejších stránek, které vyžadují náročnější funkce, jako je spolupráce s databázemi a elektronickou poštou (např. internetové obchody).

Webové stránky je možné vytvářet dvojím způsobem. Buď „ručně“, tzn. tak, že píšeme jejich zdrojový kód přímo v textovém editoru, nebo s využitím editoru určeného k tvorbě webových stránek (např. Microsoft FrontPage, TSW WebCoder apod.). Výhodou prvního způsobu je maximální přehled o obsahu a struktuře stránek a souvislostech mezi zdrojovým kódem a zobrazením stránky v prohlížeči. Nevýhodou je zejména časová náročnost tohoto způsobu tvorby a úpravy stránek, která může být především u stránek většího rozsahu značně nepraktická. Proto je tento první způsob vhodný spíše pro tvorbu osobních stránek menšího rozsahu. Druhou variantou je práce v editoru pro tvorbu webových stránek, která je pohodlná a rychlá. Nevýhodou tohoto způsobu je přidávání různě dlouhých úseků zdrojového kódu, které jsou generovány editorem a mohou ovlivňovat vzhled a funkci stránek. Následující text je zaměřen na tvorbu webových stránek prvním ze zmíněných dvou způsobů.

1.1.1.1 Značkovací jazyky

K tvorbě jednoduchých webových stránek menšího rozsahu, které nevyžadují dynamické reakce na požadavky uživatele, slouží značkovací jazyky. Nejstarším a v současné době nejrozšířenějším z těchto jazyků je HTML. Tento jazyk umožňuje vytvoření základní struktury stránek a do jisté míry také určení vzhledu jednotlivých elementů. Možnosti ovlivnění vzhledu stránek pomocí jazyka HTML jsou však omezené a úpravy a změny vzhledu pouze pomocí tohoto jazyka jsou zbytečně zdlouhavé a složité. Jednodušší řešení představuje kombinace jazyka HTML a kaskádových stylů CSS. Styly umožňují upravovat vzhled jednotlivých prvků stránek snáze a efektivněji než jazyk HTML. Dalším jazykem používaným pro tvorbu jednoduchých stránek je XHTML, které vychází z HTML a standardu XML a jeho výhodou je především jednoduchost zpracování stránek webovými prohlížeči.

1.1.1.2 HTML a XHTML

HTML je značkovací jazyk, který určuje základní strukturu webové stránky a v omezené míře může určovat také vzhled jednotlivých elementů. Je nejstarším jazykem určeným pro tvorbu webových stránek, což má za následek řadu nevýhod. Patří k nim odlišná interpretace jednotlivých značek a jejich vlastností různými prohlížeči (některé značky jsou některými prohlížeči úplně ignorovány nebo jsou podporovány pouze jedním prohlížečem nebo jednou softwarovou firmou) a také různé způsoby zápisu jedné vlastnosti. To vede k rozdílnému zobrazení stránek v různých prohlížečích a ke zpomalení zpracovávání stránky prohlížeči. Aby byly stránky jednoznačné a snadno zpracovatelné pro prohlížeče, vznikl jazyk XHTML.

XHTML je značkovací jazyk, který je aplikací standardu XML vycházející z jazyka HTML 4.01. XML je velmi obecný jazyk, podobný jazykům určeným k programování databází, který slouží ke strukturovanému ukládání dat. Nedefinuje předem žádné značky, jeho význam spočívá ve strukturovaném zápisu informací podle jejich významové hierarchie. Vytváří tak standardní formát pro přenos dokumentů, který je ve spojení s jazykem HTML interpretován shodně všemi prohlížeči, kterými je podporován. Ke zjednodušení interpretace stránky prohlížečem přispívají také jednoduchá, ale přísná syntaktická pravidla, kterými se XHTML řídí.

1.1.1.3 CSS

Jazyk HTML je určen k vytvoření základní struktury webových stránek, proto neobsahuje dostatek značek pro nastavení jejich vzhledu. Vzhledové vlastnosti stránek určené pomocí HTML značek jsou jednotlivými prohlížeči interpretovány různým způsobem a úpravy vzhledu stránek pomocí HTML jsou náročné a zdlouhavé. Tyto nevýhody je možné eliminovat používáním CSS stylů.

Kaskádové styly slouží k ovlivňování vzhledu webových stránek. Umožňují hromadné změny vlastností celých skupin HTML prvků, což velmi zjednodušuje údržbu stránek. Mají také větší možnosti výběru vlastností jednotlivých prvků i celkového vzhledu stránky. Rozdíly v interpretaci CSS stylů jednotlivými prohlížeči nejsou tak velké, jako u některých HTML značek. Kaskádové styly se používají v kombinaci s jazykem HTML a řídí se jimi vzhled stránek, zatímco HTML určuje jejich obsah.

1.1.1.4 Skriptovací jazyky

Skriptovací jazyky slouží k obsluze událostí, které nastávají na webových stránkách (např. stisk tlačítka, přejetí odkazu myší, odeslání dat prostřednictvím webového formuláře atd.). V závislosti na druhu prováděné operace a dalších aspektech (např. zabezpečení zdrojového kódu proti zkopírování, potřebná rychlost reakce stránky na požadavek uživatele, spolupráce s databázemi apod.) jsou skripty vykonávány buď v prohlížeči v počítači klienta, nebo na webovém serveru. Podle toho se skripty dělí na klientské a serverové

1.1.1.5 Klientské skripty

Klientské skripty umožňují dynamicky reagovat na požadavky uživatele. Jsou vykonávány webovým prohlížečem v počítači klienta. Jejich výhodou je rychlost provádění, protože zde odpadá nutnost odesílání informací na server a zpět a vše se děje v jednom počítači. Nevýhodou je zejména to, že zdrojový kód skriptu se odesílá spolu s HTML kódem stránky do prohlížeče klienta a může tak být kýmkoliv kopírován a šířen bez vědomí jeho autora. Klientské skripty se využívají například pro kontrolu správnosti zadaných dat ve formuláři před jeho odesláním na server, změny vzhledu stránky, komunikaci s uživatelem pomocí různých hlášek apod. Nejrozšířenějšími skriptovacími jazyky pro realizaci klientských skriptů jsou v současné době JavaScript, Jscript a VBScript.

1.1.1.6 Serverové skripty

Serverové skripty jsou vykonávány webovým serverem. Umožňují např. odesílání dat z klientských počítačů na server. Princip fungování serverových skriptů je následující:

- klient odešle požadavek na server
- na serveru je prostřednictvím serverového skriptu zpracován požadavek odeslaný klientem
- server odešle klientovi výsledek skriptu

Výhodou serverových skriptů oproti klientským je to, že zdrojový kód není potřeba odesílat do počítače klienta a kód je tak zabezpečen proti kopírování. Nevýhodou, jak vyplývá z principu provádění skriptů, je nižší rychlost provádění, závislá na rychlosti internetového připojení. Serverové skripty se využívají např. pro elektronickou poštu a

různá využití vyžadující spolupráci s databázemi. Nejpoužívanějšími skriptovacími jazyky pro serverové skripty jsou dnes PHP, ASP a CGI.

1.2 Značkovací jazyky

Výraz „značkovací jazyky“ je volným překladem anglického výrazu „markup languages“ a svou strukturou pod něj spadají jazyky HTML a XHTML. Zmíněné dva jazyky jsou určeny k vytvoření základní struktury webových stránek a HTML také k základní úpravě jejich vzhledu ve webovém prohlížeči. Jejich společným znakem je používání značek, které se umísťují před a za prvkem nebo blokem prvků, jejichž význam nebo vlastnosti určují. Jazyku kaskádových stylů CSS je věnována samostatná kapitola, neboť je v tomto systému jakousi nadstavbou, protože ovlivňuje samotný význam používaných značek, na rozdíl od HTML, které využívá značek k ovlivnění vzhledu jednotlivých elementů stránky. V kapitole věnované značkovacím jazykům se budu zabývat také jazykem XML, na jehož principu je založeno XHTML.

1.2.1 HTML

Jazyk HTML je v současné době nejrozšířenějším značkovacím jazykem a nejpoužívanějším jazykem pro tvorbu jednoduchých webových stránek. Za zkratkou HTML se skrývá výraz „HyperText Markup Language“, což v překladu znamená „hypertextový značkovací jazyk“. Značkovací proto, že, jak už bylo nastíněno, využívá pro určování vlastností jednotlivých elementů značky, zvané tagy, z předem definované množiny. Tato množina tagů byla v průběhu vývoje jazyka postupně upravována podle potřeb tvůrců webu a vývoje webových prohlížečů, a tak vzniklo postupně několik verzí HTML. Dnešní verze HTML 4.01 je považována za konečnou, přičemž jejím nástupcem se má stát jazyk XHTML, který z této poslední verze vychází.

1.2.1.1 Historie a vývoj HTML

Jazyk HTML vzniknul jako aplikace standardu SGML (Standard Generalized Markup Language), který je definován v normě ISO 8879 z roku 1986. SGML je obecným značkovacím jazykem, který umožňuje definici dalších, podřízených značkovacích jazyků na základě vlastní definice typu dokumentu (DTD). Tyto definice popisují použití značek v jednotlivých verzích jazyka HTML.

Jak napovídá samotný název HyperText Markup Language, tento značkovací jazyk je úzce spojen s hypertextem, který v podstatě tvoří jeho základ. Odkazy, pomocí kterých jsou jednotlivé dokumenty na internetu provázány, jsou prakticky neodmyslitelné. Proto také na začátku historie jazyka HTML stojí právě vývoj hypertextových systémů. Ten začal už roce 1945, kdy Dr. Vannevar Bush (1890-1974) představil svůj fiktivní přístroj s názvem Memex, na němž dokázal jako první charakterizovat a popsat hypertextovou technologii. Jednalo se o přístroj založený na technologii mikrofilmu, se způsobem třídění a vyhledávání informací podobným principu asociací v lidském mozku.

Na jeho dílo navázal společně se svými kolegy Douglas Engelbart, který vytvořil začátkem 60 let ve Stanford Research Institute projekt s názvem On-Line System (NLS), který se stal prvním skutečně hypertextovým zařízením. Mimoto se Engelbart se svým týmem zasloužil o vznik počítačové myši, zavedení většiny základních prostředků, které jsou v současnosti nezbytné při práci s textem, grafikou, hypertextem a multimédii, a konečnou definici hypertextu, která spatřila světlo světa někdy v roce 1974.

Samotný pojem „hypertext“ zavedl Theodor Nelson, který vycházel z Bushových výzkumů nezávisle na Engelbartovi. Jeho životním dílem se stal projekt Xanadu, založený na realizaci nelineárního spojování dokumentů na počítačích, který však byl do jisté míry utopický a vedle World Wide Webu se neprosadil.

Další z výrazných osobností vývoje hypertextu byl Bill Atkinson, jehož jméno je spojováno s populárním systémem HyperCard z roku 1977, který byl později, v roce 1985, překonán novějším systémem NoteCard.

Jako vynálezce World Wide Webu (WWW) však bývá označován Tim Berners-Lee, který společně s Robertem Caillauem vyvinul v roce 1989 první neoficiální verzi HTML. Stalo se tak v CERNu, Evropském centru jaderného výzkumu, ve Švýcarsku. Tato první verze jazyka HTML vznikla společně s protokolem HTTP jako součást projektu WWW, který měl umožnit vědcům zabývajícím se fyzikou vysokých energií komunikaci a sdílení výsledků výzkumů po celém světě.

První neoficiální verze HTML je známá pod označením HTML 0.9. Umožňovala text rozčlenit do několika logických úrovní, ale nepodporovala grafiku. To u prvního prohlížeče, který pracoval pouze v textovém režimu, nebylo překážkou, po vzniku grafických prohlížečů se však tato vlastnost stala výrazným handicapem.

První neformální specifikace HTML 0.9 byla zveřejněna v roce 1992. V této době vznikla první veřejně dostupná verze webového prohlížeče, která pracovala pouze v textovém režimu.

V roce 1993 byl společností NCSA (National Centre for Supercomputer Applications) dokončen první grafický prohlížeč dokumentů s názvem Mosaic, který pracoval v prostředí X-Windows. Tento prohlížeč umožnil další rozvoj WWW. S postupujícím rozvojem však požadavky uživatelů na WWW vzrůstaly, a tak producenti různých prohlížečů obohacovali HTML o nové prvky. Aby byla zachována kompatibilita mezi jednotlivými modifikacemi HTML, vytvořil Berners-Lee pod hlavičkou internetové standardizační organizace IETF (Internet Engineering Task Force) návrh standardu HTML 2.0, který zahrnoval všechny v té době běžně používané prvky HTML.

V roce 1994 byla autorem prohlížeče Mosaic založena společnost Mosaic Communications Corp., která krátce nato uvedla prohlížeč Netscape. V září téhož roku byla založena webová standardizační organizace WWW Consortium (W3C). CERN předal vývoj WWW francouzskému institutu INRIA, který v roce 1995 vydal oficiální specifikaci HTML 2.0, která představovala první oficiální verzi HTML vůbec.

Verze HTML 2.0 rozšiřovala původní specifikaci především o grafické prvky a interaktivní formuláře. Tato verze měla dvě úrovně. První z nich, Level 1, pouze málo rozšiřovala předchozí verzi HTML. Level 2 navíc definovala práci s formuláři. HTML 2.0 již plně vyhovuje normě SGML (ISO 8879 z roku 1986). Nedostatkem verze 2.0 je však malé množství prvků, umožňujících vytváření příjemného grafického vzhledu.

Ve stejném roce, kdy byla vydána oficiální specifikace HTML 2.0, bylo tvůrcem prohlížeče Netscape vydáno neoficiální rozšíření HTML 3.0.

V roce 1996 byla konsorciem W3C vydána oficiální verze HTML 3.2, která však byla v porovnání s neoficiální verzí HTML 3.0 chudší. V tomto roce vydává firma Microsoft první prohlížeč Internet Explorer 3.0. V této době také začíná podpora první úrovně kaskádových stylů CSS.

Verze HTML 3.2 přidala k jazyku tabulky, zarovnávání textu a stylové elementy pro ovlivňování vzhledu. Nedostatkem této verze bylo to, že obsahovala mnohé prvky, které naprosto postrádají strukturální význam, a slouží pouze k definici vzhledu. To však nebylo v souladu se snahou konsorcia W3C vést web k tomu, aby byl přístupný z nejrůznějších

zařízení, nejen z PC. Tohoto cíle lze nejlépe dosáhnout používáním strukturálních prvků namísto zmiňovaných prvků vzhledových, které jsou specifické pro konkrétní zařízení, v tomto případě PC.

To bylo také hlavním důvodem, proč v roce 1997 vydalo konsorcium W3C další specifikaci, HTML 4.0.

Verze HTML 4.0 představuje již vcelku mocný jazyk pro definici struktury dokumentu, zatímco vzhled dokumentu se stává téměř výhradní záležitostí kaskádových stylů. Tato verze přidala do specifikace jazyka nové prvky pro tvorbu tabulek, formulářů a nově byly standardizovány také rámy.

Specifikace HTML 4.0 byla později revidována, byly opraveny chyby této verze a přidány některé tagy, čímž vznikla specifikace HTML 4.01. Tato verze obsahuje kompletní popis jazyka a všechny následující specifikace jazyků pro tvorbu webu, které dosud vyšly, se na ni odkazují. Verze HTML 4.01 je považována za verzi konečnou, od které by se jazyk HTML již neměl dále vyvíjet. Nástupcem jazyka HTML, který by měl postupem času zaujmout jeho místo, je XHTML, jazyk založený na standardu XML. Jazyku XHTML i standardu XML je věnována samostatná kapitola. [6]

1.2.1.2 Syntaxe jazyka HTML a struktura HTML dokumentu

Jazyk HTML je aplikací standardu SGML. To znamená, že každá verze jazyka HTML používá předem definovanou množinu značek. Tato množina je pro danou verzi určena definicí typu dokumentu (DTD). Typ dokumentu, kterým se daný dokument na definici typu odkazuje, je uveden v hlavičce každého HTML dokumentu.

V HTML souborech se setkáváme jednak s upravovanými elementy, což jsou nejčastěji úseky textu, dále se značkami, které upravují vlastnosti jednotlivých elementů, s atributy značek, které tyto vlastnosti upřesňují, a jejich hodnotami, s direktivami určenými pro prohlížeč, komentáři určenými pro potřeby tvůrců dokumentů, případně se zdrojovými kódy skriptovacích jazyků a kaskádových stylů.

Značky v HTML dokumentech se nazývají tagy. Upravují vlastnosti jednotlivých elementů nebo i celého dokumentu. Samotné tagy se uzavírají do úhlových závorek „<“ a „>“ a umisťují se před a za upravovaný element. Text psaný tučným písmem by ve zdrojovém kódu HTML vypadal takto:

```
<b> Tučný text </b>
```

Tagy se dělí na párové a nepárové. Párové tagy se vyskytují v párech počáteční-koncový. Počáteční tag obsahuje informace o konkrétní vlastnosti, kterou chceme danému elementu přiřadit, a umisťuje se před elementem. Koncový tag uzavírá element a sděluje prohlížeči, že pro následující elementy už požadovaná vlastnost neplatí. Tento tag se vždy označuje stejným slovem, jako počáteční, s tím rozdílem, že před tímto slovem následuje lomítko. U párových tagů platí pravidlo, že se tagy nesmějí křížit, tzn. ten, který první začal, musí skončit jako poslední. To je důležité mít na paměti především u složitějších celků, jako jsou například tabulky, kdy jeden tag ovlivňuje více vnořených elementů. Elementy, jejichž vlastnosti upravujeme, nemusí být pouze úseky textu, ale také obrázky, odkazy nebo cokoliv jiného. Následující příklad ukazuje, jak se do sebe tagy navzájem vnořují – nekříží se:

```
<b><h1><i> Tučný nadpis první úrovně kurzívou </i></h1></b>
```

První tři tagy v tomto příkladu jsou počáteční, druhé tři jsou koncové. Text mezi tagy se v prohlížeči zobrazí jako nadpis první úrovně, psaný tučně a kurzívou.

Nepárové tagy v HTML žádné ukončovací tagy nemají. Neovlivňují tedy úsek textu, neboť je nemožné ohraničit text jedním tagem ze dvou stran, ale používají se k jiným účelům. Jedním z nejtypičtějších příkladů nepárového tagu je tag **
** sloužící k zalomení řádku, někdy zvaný také „měkký enter“.

K upřesnění vlastností, které určujeme pomocí tagů, slouží atributy. Ty jsou stejně jako tagy součástí přesně definované množiny klíčových slov, určené definicí typu dokumentu, a stejně jako tagy se mohou v jednotlivých verzích HTML lišit. Hodnoty, kterých atributy nabývají, jsou také přesně vymezeny, a to buď výčtem, nebo intervalem. Udávají se slovně nebo číslem, v určených jednotkách nebo v procentech. Následující příklad ukazuje, jak se pomocí atributu color tagu font nastaví červená barva textu:

```
<font color=“red“> Červený text </font>
```

Direktivy jsou speciální informace, které jsou určeny pro potřeby prohlížeče a týkají se způsobu interpretace HTML dokumentu (verze HTML apod.). Začínají znaky „<!“ a končí pravou úhlovou závorkou stejně jako tagy. Umístěny bývají zpravidla na začátku dokumentu. Direktiva určující verzi HTML, ve které je dokument vytvořen, vypadá takto:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
```

Komentáře jsou naproti tomu informace určené pro uživatele a tvůrce webových stránek. Prohlížeč je ignoruje a na stránce je nezobrazuje. Komentáře začínají znaky „<!--“ a končí znaky „-->“. Mezi těmito znaky může být libovolný text i na více řádků. Komentáře se můžou vyskytovat kdekoliv ve zdrojovém kódu mezi HTML tagy. Následující příklad ukazuje, jak vypadá komentář v HTML kódu:

```
<!-- Toto je komentář. -->
```

Dokumenty HTML lze vytvářet v libovolném jednoduchém textovém editoru s tím rozdílem, že je ukládáme s příponou „*.htm*“ nebo „*.html*“. Zdrojovým kódem HTML dokumentu je tedy prostý, neformátovaný text, dodržující syntaxi HTML. Prohlížeč přitom nerespektuje konce řádků dané zdrojovým souborem a text zalamuje podle šířky prostoru, který je mu vymezen na webové stránce. Pokud chceme, aby další text začínal na novém řádku, musíme použít odpovídající tag, například výše zmíněný `<br>`. Podobně je to i s mezerami. Pokud jich ve zdrojovém souboru napíšeme více za sebou, prohlížeč respektuje pouze první mezeru a další mezery ignoruje. Tento problém lze řešit nahrazením mezery znakovou entitou „` `“, kterou ovšem prohlížeč chápe jako znak a proto nemůže mezi dvěma slovy, oddělenými touto tzv. pevnou mezerou, zalomit řádek. Podobně existují znakové entity pro nahrazení různých zvláštních znaků nebo znaků vyhrazených pro potřeby jazyka HTML.

Struktura HTML dokumentu se řídí několika pevně danými pravidly. Samotný soubor začíná a končí párovým tagem `<html>`, který oznamuje, že se jedná o HTML dokument. Před tagem `<html>` se nacházejí pouze direktivy určené pro prohlížeč, které specifikují údaje, jako je verze HTML, v níž byl dokument vytvořen, atd. Konkrétně specifikace DTD je uvozována slovem „`doctype`“ a je povinná pro HTML dokumenty od verze 4.0.

Mezi tagy `<html>` a `</html>` se nachází hlavička `<head>` a tělo `<body>` dokumentu. Tyto tagy jsou povinné. Hlavička obsahuje jednak element `<title>`, který představuje název stránky, tj. text, který se zobrazí na modrém pruhu v záhlaví okna prohlížeče, dále tag `<meta>` obsahující údaje o dokumentu, jako je například jazyk, kódování, klíčová slova, atd., tag `<base>` určující základ pro relativní odkazy, kódy CSS

stylů, skriptů a další informace. Údaje uvedené v hlavičce se, s výjimkou názvu stránky, nezobrazují a jsou určeny pouze pro potřeby prohlížeče.

Tělo dokumentu následuje bezprostředně za hlavičkou. Účelem tagu `<body>` je dnes pouze ohraničení vlastního obsahu HTML dokumentu. Dříve sloužily atributy tohoto tagu k předdefinování vlastností, jako je např. barva textu, barva pozadí apod., což jsou záležitosti, které je dnes možné mnohem efektivněji a jednodušeji řešit pomocí kaskádových stylů. Uvnitř těla dokumentu je prostor pro vlastní informace v podobě textů, obrázků, tabulek nebo čehokoliv dalšího.

1.2.1.3 Příklad zdrojového kódu v HTML

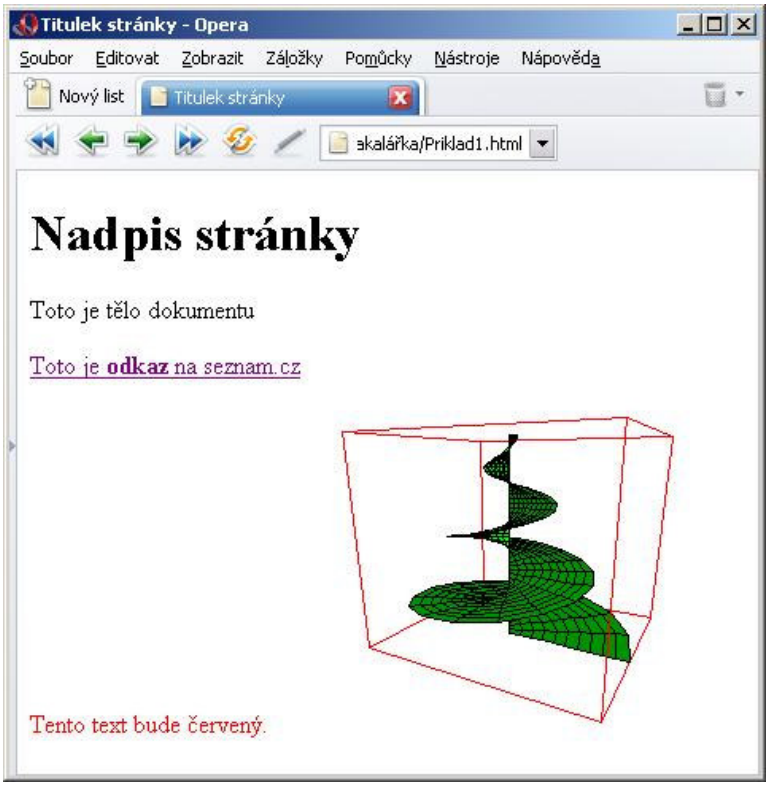
Zdrojový kód jednoduché HTML stránky, dodržující pravidla specifikace HTML 4.01 Strict, bude vypadat takto:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<!-- toto je komentář -->
   <head>
      <title>Titulek stránky</title>
   </head>
<!-- tělo dokumentu -->
   <body>
      <h1>Nadpis stránky</h1>
      <p>Toto je tělo dokumentu</p>
      <a href=http://www.seznam.cz>Toto je <b>odkaz</b> na
      seznam.cz</a>
     <br>
     <font color=red>Tento text bude červený.</font>
     <img src=euler-logo.gif>
```

```
    </body>
</html>
```

Za povšimnutí zde stojí odsazování jednotlivých řádků podle jejich hierarchie ve struktuře dokumentu. Prohlížeč toto odsazování ignoruje, ale ve zdrojovém kódu zůstává zachováno a značně zpřehledňuje celý kód, což jednak působí uspořádaně, ale především to usnadňuje orientaci, která může být zejména v rozsáhlejších kódech značně obtížná.

Kromě tagů zmíněných výše jsou v tomto příkazu použity také některé další. Tag `<p>` označuje odstavec, blok textu. Vedle možnosti ovlivnění vzhledu textu pomocí dalších atributů způsobuje sám o sobě zalomení řádku s vynecháním jednořádkové vertikální mezery nad i pod odstavcem. Tag `<a>` značí odkaz, atribut `href` pak adresu stránky, na kterou odkaz vede. Oba tyto tagy jsou párové, stejně jako tag `<font>`, použitý pro určení barvy dalšího řádku pomocí atributu color. Tag `<img>` je nepárový a slouží k vložení obrázku do stránky. Obrázek je umístěn vpravo, protože mezi ním a červeným textem není žádný tag pro zalomení řádku; obrázek tedy následuje na stejné řádce za tímto textem. Atribut `src` určuje adresu, na které je obrázek uložen. Narozdíl od adresy uvedené v atributu `href` u odkazu je adresa obrázku zadána relativně, tzn. pokud není určeno jinak, odvozuje se od umístění HTML dokumentu, v tomto případě bude prohlížeč hledat obrázek ve stejném adresáři, ve kterém je umístěn dokument. Naproti tomu adresa „*http://www.seznam.cz*“ je zadána absolutně, tzn. jednoznačně určuje umístění cíle odkazu na internetu. Na obrázku 1 je vidět, jak se výsledná stránka zobrazí v prohlížeči.



Obrázek 1: Zobrazení stránky v prohlížeči

Možnosti jazyka HTML a demonstrace využití všech tagů, jejich atributů dalece přesahující rozsah této práce. Proto zůstaneme u tohoto jednoduchého příkladu, který slouží jako ukázka základních možností využití tohoto jazyka a především jako znázornění souvislosti mezi jednotlivými úseky zdrojového kódu a výslednou podobou daných elementů ve webovém prohlížeči. [1]

1.2.2 XML

V kapitole zaměřené na vývoj jazyka HTML byl zmíněn jazyk XHTML, který je v současné době považován za nástupce HTML. Tento jazyk vychází z konečné verze HTML 4.01 a řídí se pravidly standardu XML. Jazyk HTML byl již v textu zmíněn, zbývá tedy objasnit pojem XML.

Za zkratkou XML se skrývá pojem „eXtensible Markup Language“, česky „rozšiřitelný značkovací jazyk.“ XML je obecný značkovací jazyk, který byl vyvinut a standardizován konsorciem W3C. Tento jazyk umožňuje snadné vytváření konkrétních

značkovacích jazyků pro různé účely a široké spektrum různých typů dat, který tvoří obecný standard pro strukturaci dat.

Tím vytváří jednoduchý otevřený formát, který není úzce svázán s nějakou platformou nebo aplikací, je založen na jednoduchém textu a v případě potřeby je zpracovatelný libovolným textovým editorem. Specifikace XML konsorcia W3C je zdarma přístupná všem. Každý může bez problémů ve svých aplikacích implementovat podporu XML. To je velký rozdíl oproti komerčním formátům různých firem, k nimž není k dispozici žádná dokumentace a navíc se jedná v porovnání s XML o velice složité, často binární, formáty.

Jazyk XML, stejně jako původně HTML, vychází ze standardu SGML. Nepodléhá však všem jeho pravidlům; některé vlastnosti SGML, které nebyly v souladu s účelem, za kterým bylo XML vyvinuto, byly z tohoto nového jazyka odstraněny a jiné, chybějící, naopak přidány.

1.2.2.1 Vlastnosti XML

Jazyk XML je určen především pro výměnu dat mezi aplikacemi a pro publikování dokumentů. Jazyk umožňuje popsat strukturu dokumentu z hlediska věcného obsahu jednotlivých částí, nezabývá se sám o sobě vzhledem dokumentu nebo jeho částí. Další možností tohoto jazyka je pomocí různých stylů provést transformaci do jiného typu dokumentu, nebo do jiné struktury XML. Jazyk XML nemá žádné předdefinované značky (tagy, názvy jednotlivých elementů) a také jeho syntaxe je podstatně přísnější, než např. u HTML, což vede na jedné straně k velké flexibilitě využití jazyka při zachovávání daného strukturálního standardu na straně druhé.

Výraznou výhodou jazyka XML je také mezinárodní podpora, která spočívá zejména v používání znakové sady ISO 10646 (Unicode). V XML proto lze vytvářet dokumenty, které obsahují texty v mnoha jazycích najednou – lze přepínat mezi různými jazyky v jednom dokumentu. Současně je přípustné i jiné libovolné kódování (např. windows-1250, iso-8859-2), musí však být v každém dokumentu přesně určeno. Odpadají tak problémy s konverzí z jednoho kódování do druhého.

Jak vyplývá ze strukturální povahy jazyka, tagy v XML slouží k vyznačení významu jednotlivých částí dokumentu. Proto jsou XML dokumenty jsou informačně bohatší -

obsahují více informací, než kdyby se používalo značkovaní zaměřené na vzhled (definice písma, odsazení apod.). Tuto vlastnost lze samozřejmě s výhodou využít v mnoha oblastech, z nichž nejvýraznější přínosem bude pro vyhledávače a jiné automaty, kterým strukturální zápis velmi usnadňuje určení skutečného významu daného textu.

Protože samotný jazyk XML nenabízí žádné prostředky pro definici vzhledu dokumentu, využíváme pro tento účel stylových jazyků, které umožňují definovat, jak se mají jednotlivé elementy zobrazit. Stylem rozumíme soubor pravidel nebo příkazů, které definují, jak se dokument převede do jiného formátu. Jeden vytvořený styl můžeme aplikovat na mnoho dokumentů stejného typu, stejně tak můžeme na jeden XML dokument aplikovat různé styly, čímž vznikají různé formáty dokumentu téhož obsahu (např. PostScriptový soubor, HTML kód apod.).

Protože jazyk XML je obecným standardem definujícím strukturu a pravidla zápisu programovacích jazyků určených pro konkrétní využití, neobsahuje předdefinované značky (tagy). Proto je třeba pro konkrétní jazyk definovat vlastní značky, které se budou používat. Tyto značky je možné definovat v souboru DTD (Document Type Definition), což má tu výhodu, že je možné automaticky kontrolovat, zda vytvářený XML dokument odpovídá této definici. Program, který tyto kontroly provádí (tzv. parsuje), se nazývá parser. Při vývoji aplikací lze parser použít, a ten detekuje většinu chyb v datech. Nevýhodou tohoto způsobu kontroly je, že dosud neumožňuje kontrolovat typy dat. Proto v současné době pracuje konsorcium W3C na vytvoření jednotného standardu, který tyto kontroly umožní.

Pro různé aplikace standardu XML byla postupně vytvořena schémata, která definují značky (názvy elementů) pro konkrétní typy dokumentů. Příkladem může být DocBook, který definuje struktury pro vytváření knih, článků, vědeckých publikací apod. Výhodou takových aplikací je, že současně s definičními soubory DTD je dodávána sada stylů (XSL souborů) pro následné zpracování a přípravu pro tisk, nebo pro převod do jiných standardních tvarů (PostScript, HTML atd.).

K výhodám XML patří možnost používat v jednom dokumentu najednou nezávisle na sobě několik druhů značkování pomocí jmenných prostorů (namespaces). To umožňuje kombinovat v jednom dokumentu několik různých definic ve formě DTD nebo schémat bez konfliktů v pojmenování elementů.

Součástí standardu XML jsou také zdokonalené nástroje pro vytváření odkazů v rámci jednoho dokumentu i mezi různými dokumenty s možností vytvářet i vícesměrné odkazy, které spojují více dokumentů dohromady. Tvorba odkazů je popsána ve třech standardech – XLink, XPointer a XPath. XPath (XML Path Language) je jazyk, který umožňuje adresovat jednotlivé části dokumentu. XPointer (XML Pointer Language). Je rozšířením XPath. Používá k určování jednotlivých částí dokumentu, např. prvního odstavce třetí kapitoly. Není nutné ty části dokumentu, na které chceme odkazovat, explicitně označovat pomocí návěstí jako v HTML. XLink (XML Linking Language) je samotný jazyk pro tvorbu odkazů. Jednotlivé dokumenty se určují pomocí jejich URL adresy, za kterou lze uvést ještě XPointer pro přesnější určení části dokumentu. [6]

1.2.2.2 Syntaxe XML

XML dokument je text ve formátu Unicode, v České republice obvykle kódovaný jako UTF-8. Narozdíl od např. HTML, efektivita XML je silně závislá na struktuře, obsahu a integritě. Aby byl dokument považován za správně strukturovaný („well-formed“), musí splňovat následující pravidla:

- Musí mít právě jeden kořenový (root) element.
- Neprázdné elementy musí být ohraničeny startovací a ukončovací značkou. Prázdné elementy mohou být označeny tagem „prázdný element“.
- Všechny hodnoty atributů musí být uzavřeny v uvozovkách – jednoduchých (') nebo dvojitých ("), ale jednoduchá uvozovka musí být uzavřena jednoduchou a dvojitá dvojitou. Opačný pár uvozovek může být použit uvnitř hodnot.
- Elementy mohou být vnořeny, ale nemohou se křížit; to znamená, že každý element (kromě kořenového) musí být celý obsažen v jiném elementu.
- Jazyk XML je case-senzitivní, to znamená, že rozlišuje malá a velká písmena: např. `<Příklad>` a `</Příklad>` je pár, který vyhovuje správně strukturovanému dokumentu, pár `<Příklad>` a `</příklad>` je chybný.

Příkladem správně strukturovaného XML kódu může být následující kód:

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<bylina              český_název="Podběl              lékařský"
latinský_název="Tussilago farfara“ čeleď=“ Asteraceae“>
  <titulek>Podběl lékařský</titulek>
  <výskyt>
     <místo>Pole<místo>
     <místo>štěrkoviště<místo>
     <místo>náspy<místo>
     <místo>příkopy<místo>
  </výskyt>
  <sběr>
     <sbíraná_část období_sběru=“duben“>Květ</sbíraná_část>
     <sbíraná_část období_sběru=“květen“>List</sbíraná_část>
  </sběr>
  <využití>
     <nemoc forma=“čaj“ část_rostliny=“květ“>
       Nachlazení
     </nemoc>
     <nemoc forma=“odvar“ část_rostliny=“list“>
       Špatně se hojící rány
     </nemoc>
     <nemoc forma=“ suchý zábal“ část_rostliny=“list“>
       Bolesti kloubů
     </nemoc>
     <nemoc forma=“obklad“ část_rostliny=“list“>
       Pásový opar
     </nemoc>
  </využití>
</bylina>
```

V tomto příkladu je kořenovým elementem značka `<bylina>`, všechny ostatní elementy se nacházejí uvnitř. Všechny názvy značek jsou psány malými písmeny a atributy jsou ve dvojitých uvozovkách, elementy jsou ohraničeny značkami a navzájem se nekříží – kód tedy splňuje základní pravidla pro správné strukturování XML dokumentu.

1.2.2.3 Aplikace XML

Mezi nejznámější aplikace, které vycházejí z jazyka XML, patří následující:

- XHTML – Nástupce jazyka HTML.
- RDF – Resource Description Framework, specifikace, která umožňuje popsat metadata, např. obsah a anotace HTML stránky.
- RSS – rodina XML formátů, sloužící pro čtení novinek na webových stránkách.
- SMIL – Synchronized Multimedia Integration Language, popisuje multimedia pomocí XML.
- MathML – Mathematical Markup Language, značkovací jazyk pro popis matematických vzorců a symbolů pro použití na webu.
- SVG – Scalable Vector Graphics, jazyk pro popis dvourozměrné vektorové grafiky, statické i dynamické (animace).
- DocBook – sada definic dokumentů a stylů pro publikační činnost.
- Jabber – protokol pro Instant Messaging.
- SOAP – protokol pro komunikaci mezi webovými službami.
- OpenDocument – souborový formát určený pro ukládání a výměnu dokumentů vytvořených kancelářskými aplikacemi.

1.2.2.4 Verze XML

V průběhu vývoje jazyka XML vzniklo několik jeho verzí. Aktuální verzí je XML 1.1, jejíž vznik se datuje 4. února 2004. První verze XML 1.0 vznikla v roce 1998 a v současnosti existuje ve třetí revizi. Obě verze se liší v požadavcích na použité znaky v názvech elementů, atributů apod. Verze 1.0 dovoluje v názvech elementů a atributů pouze užívání znaků platných ve verzi Unicode 2.0, která obsahuje většinu světových písem, ale neobsahuje později přidané sady jako je Mongolština apod. Verze XML 1.1 zakazuje pouze řídicí znaky, což znamená, že mohou být použity jakékoli jiné znaky. Dalším rozdílem mezi XML 1.0 a 1.1 je to, že ve verzi 1.1 se řídicí znaky mohou vkládat pouze v

podobě escape sekvencí, což jsou zástupné posloupnosti znaků, a se speciálními znaky „form-feed“ se zachází jako s bílými znaky (mezery, tabulátory – tzv. „whitespaces“).

Ve verzi 1.1 jsou platné také všechny dokumenty verze 1.0 s výjimkou některých dokumentů deklarovaných s kódováním ISO-8859-1 (Windows CP1252), které používají blok řídicích znaků pro speciální zobrazení, jako jsou znaky €, Œ, nebo ™.

1.2.3 XHTML

Snahou konsorcia W3C je vést web k tomu, aby byl přístupný z nejrůznějších zařízení a nebyl omezen vzhledovými prvky, určenými např. jen pro zobrazení na počítači. Proto jsou upřednostňovány strukturální prvky, které rozlišují jednotlivé elementy dokumentů podle jejich významu, přičemž vzhled dokumentu je druhotnou záležitostí, ovlivňovanou formátovacími styly nezávisle na obsahu dokumentu a přizpůsobovanou prezentaci na různých zařízeních. Ideální nástroj pro významové strukturování dokumentů představuje jazyk XML, který však není ani tak jazykem sám o sobě, jako spíše standardem pro vytváření jazyků pro konkrétní účel. Když vezmeme poslední verzi jazyka HTML 4.01 a upravíme ji tak, aby vyhovovala standardu XML, dostaneme jazyk, který je postavený na letech vývoje jazyka HTML, ale není zatížen jeho omezeností, spočívající jednak ve výhradně vzhledové orientaci některých prvků, způsobujících různou, mnohdy nevyhovující, prezentací výsledných dokumentů na různých zařízeních, a také v mnoha různých způsobech zápisu stejných vlastností daného elementu, což je komplikací pro prohlížeče a automaty, pracující s dokumenty.

Jazyk XHTML (eXtensible HyperText Markup Language, česky rozšiřitelný hypertextový značkovací jazyk) je aplikací standardu XML. To znamená, že se řídí pravidly, danými tímto standardem. Na rozdíl od standardu XML však jazyk XHTML definuje vlastní značky, podobné značkám HTML, které podléhají pravidlům XML. Tyto značky slouží k vytváření strukturovaného obsahu webových stránek a usnadňují tak jejich interpretaci již zmiňovaným automatům a webovým prohlížečům. [6]

1.2.3.1 XHTML ve srovnání s HTML

Když srovnáme jazyk XHTML s poslední verzí jazyka HTML, rozdíly nejsou na první pohled velké. Jejich význam, především při zjednodušení interpretace stránek, je přesto velký. Hlavními rozdíly jsou z pohledu vytváření zdrojového kódu tyto:

- v XHTML musí být všechny tagy ukončené a to včetně nepárových jako jsou
, `<hr>` nebo `<img>`
- všechny tagy a jejich atributy musí být napsány malými písmeny
- všechny hodnoty atributů musí být uzavřeny do uvozovek
- u tagu `<img>` je povinný atribut `alt`, který slouží k popisu obrázku jednak pro textové prohlížeče a také pro případ, že software uživatele daný formát obrázku nepodporuje apod.
- v XHTML 1.0 Strict byl z tagu `<href>` odstraněn atribut `target`, který umožňoval otevřít odkaz na nové stránce

V XHTML je vše jasně dáno, jazyk má poměrně jednoduchá pravidla, která je však třeba striktně dodržovat, a snahou je odstranit všechny zbytečné a nahraditelné prvky tak, aby XHTML poskytovalo základní možnosti pro tvorbu strukturovaných dokumentů, ale funkce doplňujících nástrojů zůstala na stylových, skriptovacích a dalších jazycích.

1.2.3.2 Verze a modularizace XHTML

Stejně jako HTML a další značkovací jazyky prošlo i XHTML určitým vývojem, během něhož vzniklo několik verzí, lišících se jednak svou definicí a také určením pro různá zařízení.

- XHTML 1.0 - první specifikace z roku 2000, jejímž cílem bylo upravení jazyka HTML verze 4.01 tak, aby vyhovoval podmínkám tvorby XML dokumentů a přitom byla zachována zpětná kompatibilita. Tato specifikace se od HTML 4.01 liší v bodech zmíněných výše. Protože konsorcium W3C chtělo ponechat výrobcům prohlížečů a autorům stránek dostatek času k přechodu na novou normu, rozdělilo XHTML 1.0 na tři „podverze“: XHTML 1.0 Strict, XHTML 1.0 Transitional a XHTML 1.0 Frameset. Verze Strict se striktně řídí zásadami nové specifikace. Transitional představuje přechodnou verzi, která zachovává vzhledové atributy HTML 4.01 a zároveň obsahuje výhody XHTML 1.0. Verze Frameset se téměř shoduje s verzí Transitional, je pouze rozšířena o elementy pro definici rámů.
- Modularizace XHTML (Modularization of XHTML) vznikla následkem rozšíření webu na různá alternativní zařízení (např. mobilní zařízení, tiskárny, čtečky apod.), která kvůli svým omezeným možnostem nemohou podporovat všechny vlastnosti XHTML, a proto podporují pouze určitou jejich podmnožinu. Tyto podmnožiny bylo třeba

standardizovat, proto byla vytvořena právě modularizace XHTML, která rozděluje všechny prvky do modulů, z nichž se následně skládají značkovací jazyky. To umožňuje společnostem zabývajícím se webem v alternativních zařízeních definovat nové moduly s prvky specifickými pro dané zařízení, modifikovat stávající moduly, ale především skládat z modulů nové kompletní značkovací jazyky, které vyhovují potřebám a možnostem interpretace webu na těchto zařízeních. Konsorciem W3C byly zatím uznány dva takto vytvořené jazyky, XHTML 1.1 a XHTML Basic.

- XHTML 1.1 – modulově založené XHTML, tvořené rozsáhlou sadou modulů pro komplexnější tvorbu XHTML dokumentů. Je velice podobné XHTML 1.0 Strict, ale na rozdíl od něj může díky své modularizaci sloužit jako základ budoucím rozšířeným dokumentům z rodiny XHTML. Tato specifikace postrádá již prakticky všechny vzhledové prvky. Především z důvodu správné interpretace v prohlížeči je výslovně zakázáno odesílat dokument ve formátu XHTML 1.1 s MIME typem text/html jako HTML dokument – tyto dokumenty je nutno odesílat s MIME typem application/xhtml+xml, což lze realizovat jednak pomocí deklarace MIME typu v hlavičce dokumentu, jednak příponou „*.xhtml“.
- XHTML Basic je tvořeno pouze základní sadou modulů potřebných k vytvoření XHTML dokumentu. Tato verze je určená pro mobilní aplikace (mobilní telefony, PDA apod.).
- XHTML Mobile Profile, nebo také XHTML MP, je postaveno na základě XHTML Basic a je určeno pro použití v mobilních telefonech. Někdy je také označováno jako WAP 2.0. XHTML MP však narozdíl od WAP 2.0 podporuje barvu a barevné obrázky ve formátech GIF, JPEG a PNG.
- XHTML-Print představuje verzi zaměřenou na tiskový výstup. V současné době je ve vývojovém stádiu Candidate Recommendation.
- XHTML 2.0 je v současnosti ve vývoji (stádium Working Draft). Tato verze není zamýšlena tak, aby byla zpětně kompatibilní se svými předchůdci. [6]

1.3 Kaskádové styly

Vedle prezentovaných informací je nejdůležitější vlastností webové stránky její vzhled. Při prvním dojmu bývá dokonce vlastností rozhodující. Dobrá úroveň grafiky webu vypovídá hodně o tvůrci stránky a tím i o kvalitě a spolehlivosti informací, které na stránce

můžeme čekat. Obrázky, zvýrazněná slova a optické členění stránky navíc velmi usnadňují orientaci, protože bývají lidským mozkem zpracovávány rychleji než samotný smyslu textu. Grafické prvky také slouží k snadnému zapamatování stránky, takže si ji při příští návštěvě rychleji zařadíme. Vzhledová stránka webu plní celou řadu nenahraditelných funkcí.

XHTML a novější verze HTML nenabízejí kvůli svému strukturálnímu zaměření a snaze o jednoduchost téměř žádné nebo vůbec žádné prvky umožňující ovlivňování vzhledu webových stránek. Proto používáme v kombinaci s nimi styly.

Styl je soubor příkazů a pravidel, která určují, jak se bude stránka a její jednotlivé elementy zobrazovat v prohlížeči, případně v jiném zařízení. Stylových jazyků existuje dnes několik. K nejznámějším z nich patří kaskádové styly (CSS), které slouží k upravení vzhledu HTML nebo XHTML dokumentu v prohlížeči. Další možností je rodina jazyků XSL (eXtensible Stylesheet Language), která umožňuje dokument různě upravovat a transformovat – vybírat části dokumentu nebo generovat obsahy a rejstříky. XSL se používá především v kombinaci s dokumenty vytvořenými v aplikacích jazyka XML.

CSS – Cascading Style Sheets, česky kaskádové styly, jsou v současnosti pravděpodobně nejrozšířenějším nástrojem sloužícím k úpravě vzhledu webových stránek. Nabízí široké možnosti nastavení vzhledu stránky i jednotlivých elementů a velmi zjednodušují jeho změny, což je velkou výhodou především při častých úpravách, zejména u rozsáhlejších stránek.

1.3.1.1 Výhody a nevýhody CSS

K výhodám používání kaskádových stylů patří následující:

- CSS nabízí rozsáhlejší formátovací možnosti než samotné HTML. Např. pro formátování bloku textu – tj. určení vzdálenosti textu od okraje nadřazeného elementu či okraje stránky – nejsou v HTML k dispozici žádné prostředky. V CSS můžeme tuto vlastnost nastavit pomocí hodnot vlastností `padding` a `margin`. V HTML by bylo pro dosažení stejného efektu potřeba vytvořit složitou konstrukci vnořených tabulek.
- Konzistentní styl, dynamická práce se styly – všechny elementy stejného druhu (nadpisy, seznamy apod.) se v jednom dokumentu nebo více dokumentech používajících stejný stylový soubor zobrazují stejně, což jednak zaručuje jednotný

vzhled stránek a především umožňuje měnit vzhled všech elementů stejného druhu pouze změnou definice stylu daného elementu ve stylovém souboru. V HTML by bylo třeba pro dosažení stejného efektu upravovat vzhled každého konkrétního objektu zvlášť, což je zejména u rozsáhlejších stránek kvůli vysoké časové náročnosti prakticky neproveditelné.

- Oddělení struktury a informačního obsahu dokumentu od vzhledových vlastností jednotlivých prvků – CSS na sebe zcela přebírá zodpovědnost za vzhled dokumentu, což umožňuje HTML nebo XHTML kódu zaměřit se zcela na jeho strukturu a obsah.
- Kratší doba načítání stránky - výhodou CSS oproti formátování v HTML je, že kód a obsah webu je uložen v jednom souboru (**.html*, **.xhtml*) a veškerý design a formátování se načítá z jiného souboru **.css*, který je většinou společný pro celý web. Soubor CSS se tak uloží do paměti prohlížeče a pokud není změněn, načítá se pouze jednou, čímž se zobrazení stránek velmi urychlí.
- Větší kompatibilita alternativních zařízení – CSS umožňuje vytvořit pro jednu stránku různé styly pro různá výstupní zařízení. Dokument tak může vypadat jinak na papíře, při projekci či na PDA apod. Specifikace CSS nezapomínají dokonce ani na zrakově postižené - je možno napsat styly pro hlasový syntezátor nebo hmatovou čtečku Braillova písma. Možné je samozřejmě také upravit pomocí několika souborů stylu formátování podle prohlížeče, v němž si uživatel danou stránku zobrazuje. Tím se dá eliminovat problém různé interpretace kódu jednotlivými prohlížeči.

Co se týče nevýhod CSS, hlavní a jedinou nevýhodou zůstává fakt, že určité množství prohlížečů kaskádové styly buď nepodporuje, nebo se jejich interpretace jednotlivých vlastností CSS liší. Tento problém však s postupujícím vývojem prohlížečů mizí. [6]

1.3.1.2 Verze CSS

V současné době existuje několik specifikací kaskádových stylů.

- CSS1 – vznik této verze se datuje do roku 1996, kdy se objevila také první podpora CSS v prohlížečích
- CSS2 – tato verze existuje od roku 1998
- CSS2.1 – revize verze CSS2, jejíž pracovní návrh vznikl v roce 2003
- Verze CSS3 je v současnosti ve stádiu vývoje.

1.3.1.3 Syntaxe CSS a příklad zdrojového kódu

Stylový předpis se skládá z posloupnosti pravidel. Každé pravidlo určuje vzhled některého elementu dokumentu nebo skupiny elementů, případně se týká celé stránky. Pravidlo začíná tzv. selektorem, který specifikuje („adresuje“) skupinu elementů. Selektor je následován seznamem deklarací, které určují vzhled vybrané skupiny elementů. Celý seznam je uzavřen ve složených závorkách a jednotlivé deklarace jsou odděleny středníkem (za poslední deklarací středník už být nemusí). Komentáře se v CSS píšou mezi znaky `/*“` a `*/`.

Příklad stylu:

```
h1 {                    /* vzhled nadpisu první úrovně */
    margin: 5px;        /* okraj šířky 5 pixelů        */
    font-size: 12pt     /* velikost fontu 12 bodů      */
   }
p  {                      /* styl odstavce       */
    text-align: center;   /* zarovnání na střed  */
    line-height: 10pt;    /* výška řádku 10 bodů */
   }
```

Uvedený příklad by mohl být součástí souboru definujícího styl. Styl však lze zapsat třemi různými způsoby: do samostatného souboru (s příponou **.css*) pro ovlivnění vzhledu jednoho nebo více dokumentů (tzv. propojený styl), do hlavičky HTML dokumentu pro ovlivnění pouze tohoto dokumentu (tzv. globální styl) nebo přímo do tagu ohraničujícího konkrétní prvek pro ovlivnění pouze tohoto konkrétního elementu (tzv. in-line styl). První z uvedených způsobů je přitom z hlediska snadnosti provádění úprav a redukce kódu nejvýhodnější, neboť umožňuje upravovat současně vzhled všech dokumentů, které se odkazují na daný soubor stylu. Druhý způsob stále zjednodušuje práci, ale můžeme jím ovlivňovat vzhled elementů pouze v rámci jednoho souboru. Poslední způsob slouží prakticky pouze k definici nějaké ojedinělé vlastnosti u jednoho konkrétního prvku, ačkoliv tento prvek lze stejně jako ostatní nadefinovat i v hlavičce dokumentu nebo v externím souboru stylu. Tento třetí způsob vede ve výsledku k většímu objemu zdrojového kódu a zápis je dokonce ještě složitější, než kdybychom zapsali tutéž vlastnost pomocí HTML.

Takto vypadají jednotlivé způsoby zápisu, pokud chci například obarvit písmo v jednom odstavci na červeno pomocí CSS:

1) Propojený styl – část kódu externího souboru stylu:

```
p {color: red}
```

Odkaz na soubor v hlavičce HTML souboru:

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

Přiřazení stylu odstavci v HTML souboru:

```
<p>Tento odstavec bude červený. </p>
```

2) Globální styl – část hlavičky HTML dokumentu:

```
<style>

p {color: red}

</style>
```

Přiřazení stylu odstavci v HTML souboru:

```
<p>Tento odstavec bude červený. </p>
```

3) In-line styl – tag daného odstavce:

```
<p style="color: red">Tento odstavec bude
červený.</p>
```

K uvedeným příkladům je nutno dodat, že poslední způsob slouží k obarvení pouze jednoho konkrétního odstavce. Pokud se nadefinuje barva odstavce prvními dvěma způsoby, bude tato vlastnost platit pro všechny elementy ohraničené tagy `<p>` a `</p>`. Pokud by bylo třeba upravovat vlastnosti pouze některých odstavců v dokumentu, musela by se např. pro červený odstavec nadefinovat třída nebo identifikátor. Zápis v externím souboru stylu pomocí třídy by pak vypadal takto:

```
.cerveny { text-align: justify; font-weight: bold;
color: red}
```

Použití třídy `cerveny` v souboru HTML by vypadalo následovně:

```
<p class="cerveny">Tento odstavec bude červený. </p>
```

Stejným způsobem lze nadefinovat třídu `cerveny` také ve stylopisu v hlavičce HTML dokumentu. Zápis tříd se od selektorů, kterými jsou přímo HTML tagy, liší pouze tečkou na začátku, zápis identifikátorů má na začátku místo tečky znak # a jejich využití je téměř stejné jako využití tříd. [1], [6]

1.4 Skriptovací jazyky

Pomocí jazyka HTML (příp. XHTML) a kaskádových stylů lze vytvořit webové stránky, které mají z pohledu uživatele neměnný obsah a vzhled. Takové stránky umožňují prezentaci informací ve zvolené formě, ale nic víc. Pokud je třeba vytvořit stránky dynamicky měnící svůj vzhled a obsah v závislosti na požadavcích uživatele, tzn. obohatit je například o vizuální efekty, animace, hlášky, možnost změny vzhledu ze strany uživatele nebo třeba diskusní fórum a podobné funkce, je potřeba nástroj, který poskytne odpovídající možnosti. Takový nástroj představují skriptovací jazyky.

Skriptovací jazyky jsou narozdíl od HTML a CSS skutečnými programovacími jazyky. Jedná se o jazyky interpretované. To znamená, že k jejich spuštění je potřebný tzv. interpreter, což je aplikace, která skript vykonává přímo v jeho původní podobě zdrojového kódu, případně ho překládá do mezikódu. Interpreter prochází zdrojový kód a vykonává postupně příkaz za příkazem bez potřeby kompilace. Protikladem k interpretovaným jazykům jsou jazyky kompilované, které jsou po svém naprogramování přeloženy, tj. převedeny do podoby strojového kódu a teprve v této podobě jsou spouštěny. Výhodou skriptovacích jazyků je jejich snadná údržba a vývoj. Skript je možné kdykoliv spustit bez nutnosti vždy znovu kompilovat. Programátorovi v tomto případě stačí obyčejný textový editor, nepotřebuje tedy žádný speciální kompilátor nebo vývojový nástroj.

Nevýhodou skriptovacích jazyků je, že jsou mnohem pomalejší ve srovnání s kompilovanými programovacími jazyky, které se spouští binárními spustitelnými soubory. Zdrojový kód skriptu je totiž třeba před každým spuštěním znovu zkontrolovat, zda

neobsahuje syntaktické chyby, a interpreter musí následně analyzovat každý příkaz ve zdrojovém kódu, aby ho mohl vykonat.

Z hlediska programování pomocí skriptovacích jazyků jsou jednotlivé prvky HTML dokumentu chápány jako objekty. Objekty a jejich skupiny (kontejnery) jsou součástí hierarchické struktury – objektového modelu dokumentu (DOM – Document Object Model). Nad těmito objekty pak provádí skriptovací jazyky jednotlivé operace. Pokud je třeba provést operaci s konkrétním objektem, měnit jeho vlastnosti nebo obsah, je nezbytné ho identifikovat, přiřadit mu nějaký konkrétní název. Přes tento název pak lze přistupovat k samotnému objektu i k jeho jednotlivým vlastnostem.

Skriptovací jazyky se dělí na dvě základní skupiny - klientské a serverové, podle toho, kde a čím je skript vykonáván. Tato kapitola se bude věnovat dvěma nejrozšířenějším zástupcům obou těchto skupin. Z klientských skriptovacích jazyků je to JavaScript, ze serverových jazyk PHP.

1.4.1 JavaScript

JavaScript patří mezi skriptovací jazyky určené pro tvorbu klientských skriptů. Tento jazyk má své kořeny v jazyku Java. Proto má podobnou syntaxi, ale představuje samostatný programovací jazyk. JavaScript je, stejně jako ostatní skriptovací jazyky, jazykem interpretovaným a objektově orientovaným. Je také case-senzitivní – tzn. rozlišuje malá a velká písmena např. v názvech jednotlivých objektů.

Zdrojové kódy JavaScriptu lze vytvářet v textovém editoru nebo pomocí specializovaných vývojových nástrojů. Soubory skriptů se ukládají s příponou „**.js*“.

1.4.1.1 Výhody a nevýhody JavaScriptu

Jazyk JavaScript má následující výhody:

- Jako klientský skriptovací jazyk má JavaScript vyšší rychlost zpracování požadavků než skripty serverové, což je následkem toho, že není nutné odesílat požadavek ke zpracování na server a čekat na jeho odpověď
- Zdrojové kódy JavaScriptu se zapisují přímo do HTML dokumentů, což je výhodné pro jednoduchost tohoto způsobu zápisu

- Možnosti jazyka JavaScript: provádění změn vzhledu stránek (v kombinaci s CSS), provádění výpočtů, změny obsahu dokumentu, komunikace s uživatelem pomocí různých hlášek, animace atd.

K nevýhodám JavaScriptu patří tyto:

- JavaScript je závislý na jeho interpretaci prohlížečem, která se v různých prohlížečích může lišit. Odlišnosti mohou způsobovat také různé verze jazyka (např. JScript)
- Uživatel může JavaScript zakázat
- JavaScript neumožňuje přístup k souborům na straně klienta (kromě souborů cookies) ani k žádným systémovým objektům, proto neumožňuje ukládání a čtení dat kromě zmíněných cookies. Cookies jsou soubory malé velikosti (řádově bytů), které slouží k ukládání dat sloužících pro potřeby skriptu v prohlížeči uživatele.

1.4.1.2 Způsoby zápisu skriptu a syntaxe JavaScriptu

Skript v jazyce JavaScript je podobně jako jiné klientské skripty možné začlenit do webové stránky třemi způsoby, které jsou téměř shodné se způsoby zápisu CSS stylů. Patří sem in-line zápis přímo u konkrétního objektu, se kterým skript pracuje, zápis skriptu kdekoliv v těle dokumentu a zápis skriptu s odkazem na externí soubor. U posledních dvou uvedených způsobů se používá tag `<script>`.

1) In-line zápis:

```
<a href="http://www.seznam.cz"
onmouseover="alert('Toto je odkaz na
Seznam.cz.')">Seznam</a>
```

Tento kód se narozdíl od dalších dvou neohraničuje tagem `<script>`, ale píše se do tagu konkrétního elementu, v tomto případě odkazu, jako jeho atribut. Tento způsob slouží k zápisu jednoduchých kódů. Kód uvedený v tomto příkladu vygeneruje při přejetí odkazu myší hlášku, „vyskakovací“ okno, které uživatel zavře kliknutím na tlačítko „*OK*“. Slovo `onmouseover` označuje událost, při které dojde k provedení skriptu (v tomto případě je to přejetí daného objektu myší, ale existuje i řada dalších událostí). To, co je v uvozovkách za `onmouseover`, je vlastní skript, v tomto případě hláška `alert`. Text hlášky je uzavřený v jednoduchých uvozovkách a v kulatých závorkách.

In-line zápis lze také realizovat formou odkazu na skript namísto reakce na událost. Takový zápis vypadá následovně:

```
<a href="javascript: alert('Toto je hláška JavaScriptu.')">Skript</a>
```

Uvnitř tagu `<a>` přitom může být i obrázek nebo jiný element. Skript se spustí po kliknutí na tento element.

2) Zápis v těle dokumentu:

```
<script>
  alert('Toto je hláška JavaScriptu.');
</script>
```

3) Externí soubor:

V externím souboru skriptu (s názvem "*externi_skript.js*") bude následující kód:

```
document.write('Toto je hláška externího JavaScriptu.');
```

V HTML dokumentu bude tento kód, odkazující na externí soubor:

```
<script src="externi_skript.js"></script>
```

Každý z těchto dvou zápisů vygeneruje hlášku stejně jako první příklad. Skript zde není uzavřený v uvozovkách, jak je tomu u atributového způsobu zápisu, ale je ohraničený tagem `<script>`. Takto lze zapisovat i posloupnost více příkazů, pro kterou není in-line zápis vhodný. Pro složitější kódy je však zpravidla vhodnější zápis v externím souboru, jednak pro větší přehlednost a snadnější úpravy, ale také z hlediska redukce kódu samotné HTML stránky a v neposlední řadě kvůli možnosti využití téhož kódu pro obsluhu událostí na více stránkách.

Poslední dva způsoby zápisu je možné také kombinovat. Záleží však na správném zápisu v HTML dokumentu. Aby oba skripty fungovaly, je potřeba je zapsat takto:

```
<script src="externi_skript.js"></script>
<script>
  alert("Toto napsal interní skript.")
</script>
```

Zdrojový kód souboru skriptu bude stejný, jak je uvedeno v předchozím příkladu. Výsledkem budou dvě hlášky, jedna vygenerovaná externím souborem a druhá „interním“ skriptem.

Všechny uvedené způsoby zápisu jsou rovnocenné, výhodnost jejich použití však záleží na konkrétním účelu. Externí skript se zpravidla používá k definování různých funkcí, zápis do HTML dokumentu pomocí tagu `<script>` slouží k inicializaci proměnných a in-line skripty volají funkce podle událostí v závislosti na reakcích uživatele. [1], [6]

1.4.2 PHP

Jazyk PHP (PHP: Hypertext Preprocesor nebo Personal HomePage) se řadí k serverovým skriptovacím jazykům. V současné době panuje na poli webových stránek tvrdá konkurence mezi tímto jazykem a jazykem ASP (Active Server Pages). Kromě těchto dvou jazyků jsou velmi rozšířené také CGI skripty, které jsou v kombinaci se skripty v jazycích Perl a Python hojně využívány zejména pro speciální řešení na velkých serverech.

Jako serverový skriptovací jazyk nabízí PHP možnost vytváření plnohodnotných aplikací, spolupráce s formuláři, databázemi a s poštou. Je jazykem interpretovaným, na rozdíl od klientských skriptů však software, který provádí jeho interpretaci, pracuje přímo na webovém serveru. Serverem je v tomto případě myšlen software, který umožňuje běh interpreteru PHP a dalších programů, určených například pro práci s databázemi. K rozšířeným serverovým softwarům patří např. Apache, Xitami apod.

Syntaxe jazyka PHP vychází ze syntaxe několika z nejpoužívanějších programovacích jazyků: jazyka C, Perlu, Pascalu a Javy. Zdrojové kódy PHP se začleňují přímo do kódů HTML dokumentů, což je výhodou při tvorbě webových aplikací. Výsledné soubory mají pak obvykle příponu **.php*. Kromě toho je však PHP využíváno také pro tvorbu jiných aplikací. Výhodou jazyka PHP je přitom jeho nezávislost na platformě, což znamená, že programy napsané v PHP fungují na mnoha různých operačních systémech bez nutnosti úprav. PHP disponuje rozsáhlými knihovnami funkcí pro zpracování textu, grafiky, práci se soubory, přístup k většině databázových serverů (např. MySQL, ODBC, Oracle, PostgreSQL, MSSQL), podporu celé řady internetových protokolů (HTTP, SMTP, SNMP, FTP, IMAP, POP3, LDAP aj.). PHP se stalo velmi oblíbeným především díky

jednoduchosti použití a kombinaci vlastností několika programovacích jazyků, což poskytuje částečnou svobodu v syntaxi. V kombinaci s databázovým serverem (nejčastěji s MySQL nebo PostgreSQL) a webovým serverem Apache je často využíván k tvorbě webových aplikací. S verzí PHP 5 se výrazně zlepšil přístup k objektově orientovanému programování podobný Javě. [6]

1.4.2.1 Vznik a vývoj PHP

PHP vzniklo v roce 1995. Jeho tvůrce Rasmus Lerdorf jej vytvořil pro svou osobní potřebu přepsáním z Perlu do jazyka C. Ještě v témž roce byla vydána sada skriptů pod názvem Personal Home Page Tools, zkráceně PHP. V polovině roku se systém PHP spojil s programem Form Interpreter stejného autora. Tak vzniklo v roce 1996 PHP/FI 2.0. Zeev Suraski a Andi Gutmans v roce 1997 přepsali parser a zformovali tak základ PHP3. Současně byl název změněn na dnešní podobu, PHP: Hypertext Procesor. Verze PHP3 byla vydána v roce 1998. Tato verze byla rychlejší, obsahovala více funkcí a fungovala i na platformě Windows. Verze PHP 4 vyšla v roce 2000, následovaly ji verze 4.1 (2001), 4.2 (duben 2002), 4.3 (prosinec 2002) a 4.4 (2003). V roce 2004 bylo vydáno PHP 5.0 s vylepšeným objektovým přístupem, podobným jazyku Java. V roce 2005 následovala verze 5.1 a v roce 2006 PHP 5.2. Zatím poslední verzí je PHP 5.2.1 z roku 2007. [6]

1.4.2.2 Výhody a nevýhody PHP

Nejvýznamnějšími výhodami jazyka PHP jsou tyto:

- Zdrojový kód skriptů se neodesílá do prohlížeče klienta, ale zůstává na serveru, což zaručuje ochranu kódu proti kopírování
- PHP narozdíl od klientských skriptů umí pracovat se soubory, tj. číst a ukládat data, proto jsou možnosti jejich využití téměř neomezené
- PHP spolupracuje s databázemi, např. MySQL, PostgreSQL, což mu umožňuje snadné zpracování velkého množství informací
- Kromě toho také pracuje s databázemi pomocí rozhraní ODBC (Open DataBase Connectivity), které umožňuje práci s daty programů, jako je např. Microsoft Access nebo Microsoft Excel
- Velkou výhodou PHP je i jeho nezávislost na platformě

PHP má také následující nevýhody:

- Jako serverový skript má delší dobu odezvy na požadavek uživatele, která je závislá na rychlosti internetového připojení a na době potřebné ke zpracování požadavku na serveru, proto nedovede dynamicky reagovat na události jako např. JavaScript
- K ladění PHP skriptů je nezbytný serverový software, např. Apache nebo Xitami

Vzhledem k tomu, že PHP není vhodné pro implementaci funkcí, které vyžadují rychlé zpracování, je vhodné jej používat v kombinaci s některým z klientských skriptovacích jazyků, např. s JavaScriptem.

1.4.2.3 Syntaxe PHP a příklady zdrojového kódu

V jazyce PHP se od sebe jednotlivé příkazy oddělují středníkem nebo HTML tagy. Při psaní příkazů nezáleží na velikosti písmen, ale při psaní proměnných už ano.

Proměnné se v jazyce PHP uvozují znakem `$`. Není je nutné deklarovat předem ani určovat jejich datové typy, protože se určí automaticky v okamžiku přiřazení hodnoty. Proto má PHP dva typy porovnání:

- rovnost == platí stejně jako v jazyce C
- rovnost === platí pouze tehdy, pokud jsou obě porovnávané proměnné stejného typu

  Pole jsou v PHP heterogenní, mohou tedy obsahovat různé datové typy.

Řetězce lze uzavírat jak do uvozovek (obsah takových řetězců je parsován), tak do apostrofů (obsah není parsován).

Komentáře v PHP jsou obdobné jako v jazyce C. Text mezi znaky `//` a koncem aktuálního řádku je komentář, text mezi znaky `/*` a `*/` také. Komentáře ohraničené znaky `/*` a `*/` můžou být i víceřádkové.

Bloky příkazů se uzavírají do složených závorek, podmínky do závorek kulatých stejně jako v jazyce C.

Takto vypadá v PHP základní výpis textového řetězce:

```
<? echo "Hello World!" ?>
```

PHP soubor je tvořen úseky HTML a PHP kódů, přičemž PHP kódy se uzavírají mezi znaky `<?` a `?>`. Vše, co je mezi těmito znaky, je interpreterem chápáno jako skript a bude

interpretováno jako PHP. Interpretace PHP probíhá na serveru a do prohlížeče klienta se odesílá HTML kód společně s výsledky PHP skriptů. Výsledkem kódu uvedeného v příkladu by byl text „Hello World!“. Pomocí PHP lze vypisovat také HTML tagy, které budou odeslány do prohlížeče a v prohlížeči budou interpretovány jako obyčejné HTML, které je pevně zapsáno ve zdrojovém kódu.

Takto může vypadat začlenění PHP skriptu do HTML kódu:

```
<p>
Nějaký text.
<? echo "Hello World! " ?>
Další text.
</p>
```

Takto bude vypadat text, který se ve výsledku zobrazí v prohlížeči:

```
Nějaký text. Hello World! Další text.
```

Využití PHP tak, jako v tomto příkladu, nemá velký smysl, protože text „Hello World!“ lze stejně tak zapsat do HTML. Mnohem větší význam mají například podmíněné příkazy, které v závislosti na vyhodnocení určité podmínky vygenerují nějaký HTML kód.

Následující příklad demonstruje použití podmíněného výpisu textu:

```
<? if (date("A")=="AM"): ?>
Dobré ráno!
<? else: ?>
Dobré odpoledne!
<? endif ?>
```

Pokud je podmínka `date("A")=="AM"` splněna, vypíše se v HTML souboru text „Dobré ráno!“, pokud splněna není, vypíše se „Dobré odpoledne!“.

Jazyk PHP má velmi široké možnosti využití, především v kombinaci s databázemi a ve spolupráci s dalšími programovacími jazyky. Účelem této kapitoly je pouze stručný nástin syntaxe jazyka a jeho základního použití, což demonstrují uvedené příklady. [6]

II. PRAKTICKÁ ČÁST

2 SEZNÁMENÍ S PROGRAMEM EULER

Program Euler je volně dostupnou alternativou komerčního softwaru Matlab. Přestože je tomuto softwaru velmi podobný, není jeho klonem. Euler je výkonná numerická laboratoř s vlastním programovacím jazykem. Program pracuje s reálnými a komplexními čísly, intervaly, vektory a maticemi a je schopen vykreslit dvojrozměrné i trojrozměrné grafy. Euler spolupracuje s algebraickým počítačovým systémem Yacas, což umožňuje kombinovat symbolické a rychlé numerické programování.

2.1 Instalace

Euler existuje v několika verzích pro různé operační systémy. Dvěma aktuálními verzemi jsou verze 2.4 pro Microsoft Windows, vytvořená Dr. Rene Grothmannem, a verze 1.61 pro Linux/Unix, kterou vytvořil Eric Boucharé v GTK. Další verze jsou určeny pro systémy OS/2 a DOS, avšak používání těchto verzí se nedoporučuje, protože už nejsou aktualizovány.

Protože operační systém Microsoft Windows je v současné době nejrozšířenějším operačním systémem, tato práce se zabývá verzí 2.4 pro Microsoft Windows. Práce v ostatních verzích by se však neměla od práce v této verzi příliš lišit, protože rozdíly jsou pouze v uživatelském rozhraní a princip fungování je u všech verzí téměř stejný.

Euler je volně šiřitelný software, tzv. freeware, a všechny jeho verze jsou šířeny v souladu s GNU General Public License. To znamená, že každá modifikovaná verze zůstává freeware a open source (tzn. její zdrojové kódy jsou volně přístupné komukoliv) a musí být šířena s jasným odkazem na původní verzi. Další licence jsou povoleny.

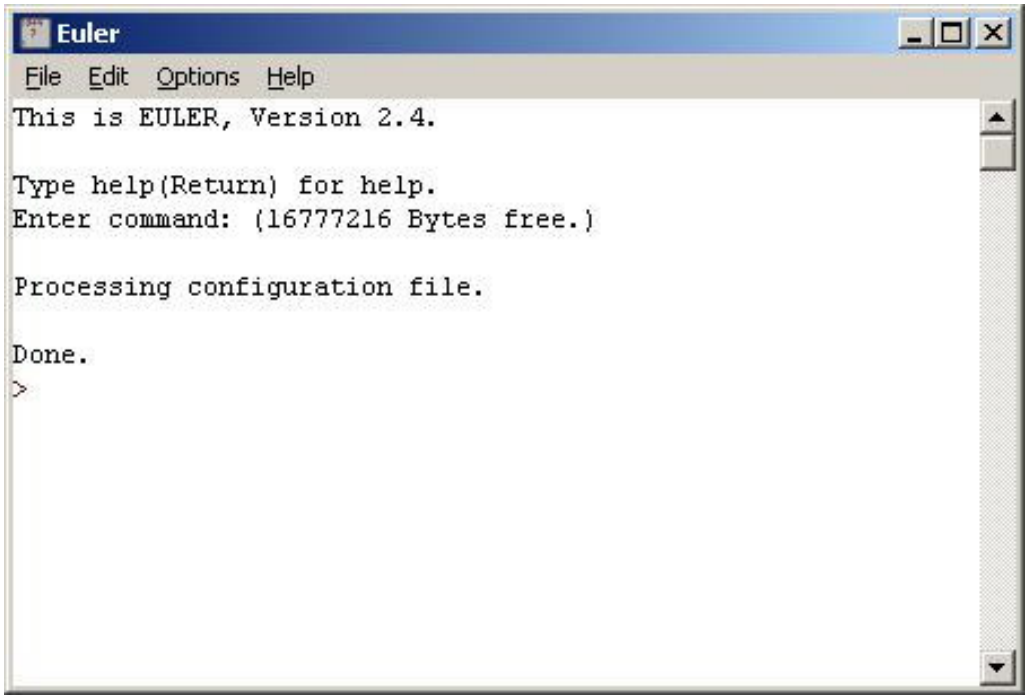
Instalační soubor verze 2.4 pro Windows je ke stažení na stránkách Dr. Rene Grothmanna (http://mathsrv.ku-eichstaett.de/MGF/homes/grothmann/euler/) pod názvem *euler95i.exe*.

euler95i.exe je samorozbalovací spustitelný instalační soubor. Po spuštění nainstaluje program Euler a vytvoří potřebné spouštěcí ikony v nabídce *Start*. Vytvořený adresář Euler obsahuje mimo jiné konfigurační soubor *euler.cfg* a podadresáře *progs* a *docs*, které obsahují rozšiřující soubory Euleru a dokumentaci k programu.

2.2 Spuštění a ukončení programu

Program Euler se spouští buď ikonou na pracovní ploše, nebo položkou *Euler* v nabídce *Start*, která je automaticky vytvořena instalačním souborem. Po spuštění programu se objeví pracovní okno a probíhá provádění konfiguračního souboru a načítání rozšiřujících souborů Euleru. To trvá několik málo vteřin, což je ve srovnání s podobnými programy velmi krátká doba.

Po vypsání několika úvodních řádků se na následujícím řádku objeví znak „>“, který signalizuje, že je program připraven a očekává zadání příkazu. Nyní je možné v programu začít pracovat.



Obrázek 2: Euler po spuštění čeká na zadání prvního příkazu

Po skončení práce můžeme ukončit program dvojím způsobem. Buďto zadáním příkazu `quit`, který se provede po stisku klávesy Enter. Pracovní okno Euleru se zavře a program je bez dalších dotazů ukončen. Veškerý obsah pracovního okna, který nebyl uložen do souboru, je smazán. Druhým způsobem je zavření pracovního okna Euleru křížkem v pravém horním rohu. Při tomto způsobu ukončení se program zeptá, zda chceme skutečně program ukončit. Kliknutím na tlačítko „*Ano*“ je program ukončen stejně jako v prvním případě, kliknutím na tlačítko „*Ne*“ se vrátíme zpět do programu.

2.3 Práce s programem

Program pracuje podobně jako Matlab v textovém okně, v němž jsou zadávány příkazy. Pokud je výsledkem zadaného příkazu grafický výstup zobrazí se tento výstup v automaticky otevíraném grafickém okně. Pokud je výsledkem textový výstup, zobrazí se v textovém okně na následujícím řádku za zadaným příkazem.

Příkazy se v Euleru zobrazují červenou barvou. Výstupní hodnoty se zobrazují černě. Toto rozlišení zpřehledňuje a usnadňuje práci. Všechny příkazy v textovém okně je možné editovat a znovu spouštět stiskem klávesy Enter, když se kurzor nachází na příslušném řádku.

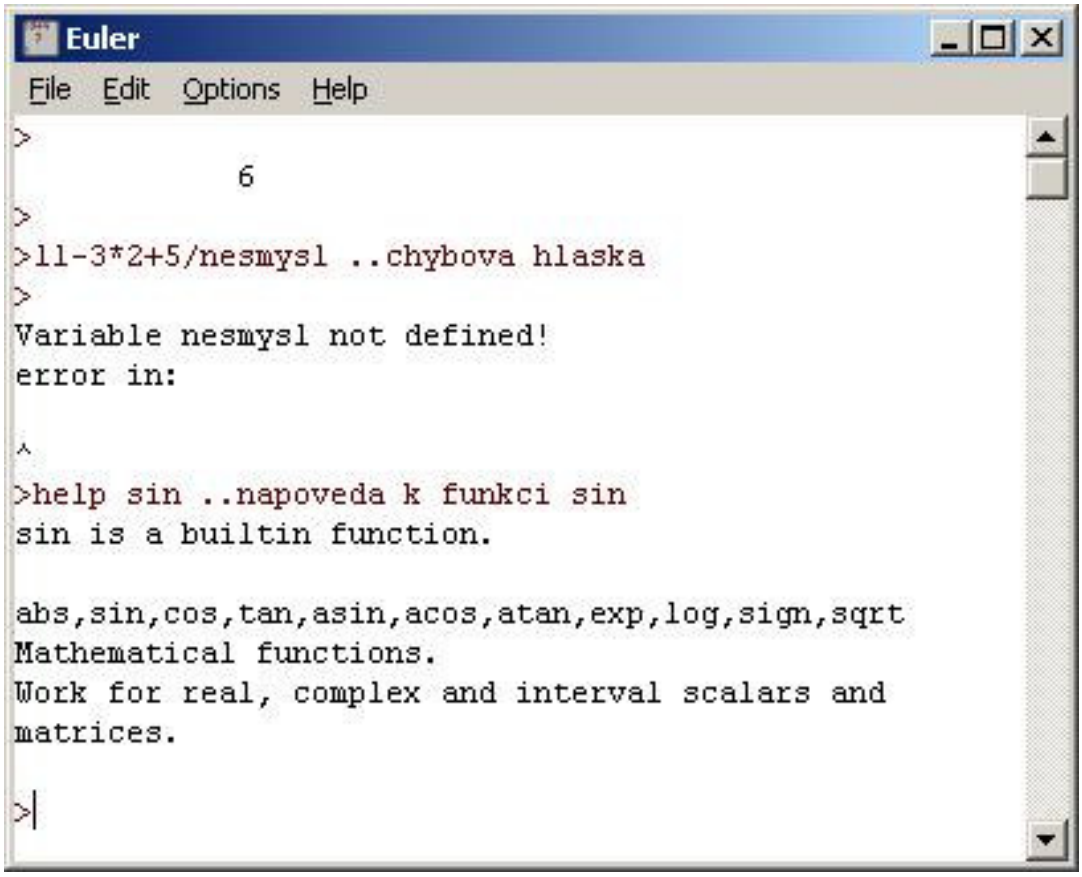
Ke zpřehlednění práce slouží také komentáře. Ty jsou v textovém okně a v notebookových souborech zobrazeny červenou barvou. Komentáře v rozšiřujících souborech se zobrazují zeleně.

Euler je schopný vytvářet dva druhy souborů. Jsou jimi notebooky s příponou „**.en*“ a rozšiřující soubory s příponou „**.e*“ Notebooky slouží k ukládání příkazů, které jsou zadávány v textovém okně. V těchto souborech lze programovat také funkce, ale ty je možné spouštět pouze tehdy, když je soubor otevřen v textovém okně. Rozšiřující soubory jsou přímo určeny k programování nových funkcí. Když je takový soubor Eulerem načten, mohou být funkce, které jsou v něm zapsány, spouštěny stejně jako standardní funkce a příkazy Euleru. Těchto souborů je možné mít načteno i více současně. Po dalším spuštění Euleru je však nutné načíst tyto soubory znovu. Pohodlnějším řešením je přidání názvů těchto souborů do konfiguračního souboru *euler.cfg*, který při každém spuštění systému načítá všechny standardní rozšiřující soubory Euleru. Tak je možné přizpůsobit si Euler podle individuálních potřeb a účelu využití.

Pokud je příkaz v textovém okně Euleru zadán chybně, například je použita neznámá proměnná, Euler vypíše chybovou hlášku. Spolu s touto hláškou vypíše daný příkaz a na následujícím řádku zobrazí znak „^“, kterým označí místo, kde došlo k chybě. Tak je možné chybu přesně identifikovat a opravit, což velmi usnadňuje práci zejména u zápisu složitých vzorců apod.

Nápověda, jak ji známe z jiných programů, v programu Euler zatím neexistuje. Při kliknutí na položku menu *Help* se otevře pouze okno odkazující na domovskou stránku autora programu, jejíž součástí je i dokumentace k programu. Existuje však funkce `help`, která

vyvolá stručnou nápovědu k jednotlivým funkcím. Například příkaz `help sin` vyvolá nápovědu k funkci sinus. V orientaci může pomoci také funkce `list`, která slouží k vypsání všech funkcí, které má Euler aktuálně k dispozici, včetně funkcí zapsaných v aktuálně načtených rozšiřujících souborech.



Obrázek 3: Ukázka základní práce v Euleru, chybové hlášky a příkazu help

Za zmínku stojí také středník „`;`“, který v Euleru slouží k potlačení výstupu daného příkazu. To znamená, že pokud zapíšeme příkaz ve tvaru „`prikaz;`“, jeho výstup se na následujícím řádku nevypíše. To je praktické zejména při zadávání posloupnosti více příkazů za sebou, například při vykreslování grafů funkcí, kdy nejsou podstatné mezivýsledky, pouze konečný výstup.

3 UKÁZKOVÉ PŘÍKLADY PRÁCE V PROGRAMU EULER

3.1 Přiřazení hodnoty proměnné a jednoduché výpočty

Proměnné v Euleru jsou jednoznačně určeny názvem. Název proměnné musí začínat písmenem, dalšími znaky mohou být číslice nebo písmena. Euler rozlišuje v názvech proměnných mezi malým a velkým písmenem, `X` a `x` jsou tedy chápány jako dvě různé proměnné. České znaky v názvech proměnných Euler nepodporuje. Pokud zadáme proměnnou obsahující písmeno s čárkou nebo háčkem, následuje chybová hláška. Maximální délka názvu proměnné je 11 znaků. Datový typ proměnné je určen automaticky v okamžiku, kdy je proměnné přiřazena konkrétní hodnota. Euler rozeznává následující datové typy:

0 - reálná čísla

1 - komplexní čísla

2 - vektory a matice s reálnými prvky

3 - vektory a matice s komplexními prvky

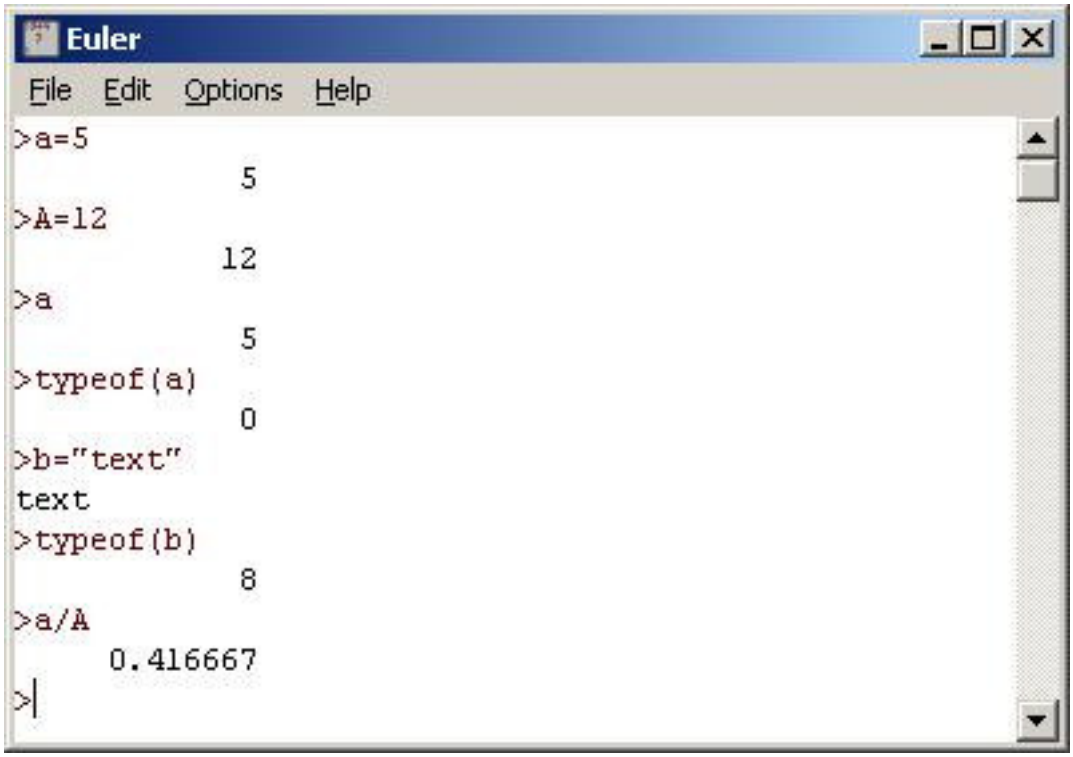
8 - řetězce

10 - intervaly

Čísla přiřazená jednotlivým datovým typům jsou návratovými hodnotami funkce `typeof()`, která slouží ke zjištění datového typu proměnné.

Datové typy jsou důležité zejména u funkcí. Datový typ parametrů předávaných funkcí se musí shodovat s datovým typem její návratové hodnoty. Proto musí být například při odmocňování záporného reálného čísla být toto číslo převedeno na komplexní pomocí funkce `complex(a)`.

K přiřazení hodnoty proměnné slouží operátor „=“. K porovnání hodnot dvou proměnných slouží dvě rovnítka („==“).



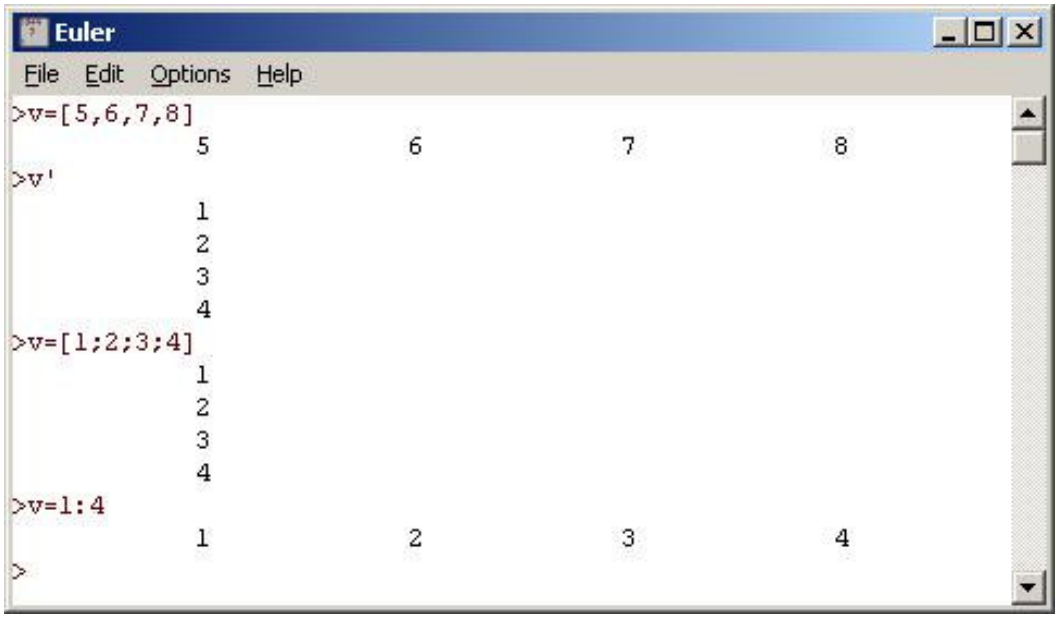
Obrázek 4: Přiřazování hodnot proměnným a funkce typeof()

3.2 Operace s vektory

Vektory v Euleru mohou obsahovat reálné nebo komplexní prvky. Jednotlivé prvky je možné zadat buď přímo z klávesnice nebo nechat vygenerovat v daných mezích. Určit lze také to, o kolik se od sebe jednotlivé prvky budou lišit, případně vygenerovat vektor, jehož prvky budou stejná čísla.

Vektory mohou být řádkové nebo sloupcové. Při ručním zadávání od sebe jednotlivé prvky řádkového vektoru oddělujeme čárkou. Prvky sloupcového vektoru od sebe oddělujeme středníkem.

Automaticky generované vektory jsou pouze řádkové. Pokud je třeba sloupcový vektor, pak lze řádkový vektor transponovat pomocí následujícího znaku „‘“ (apostrofu). Transponovat lze také řádkový vektor na sloupcový.



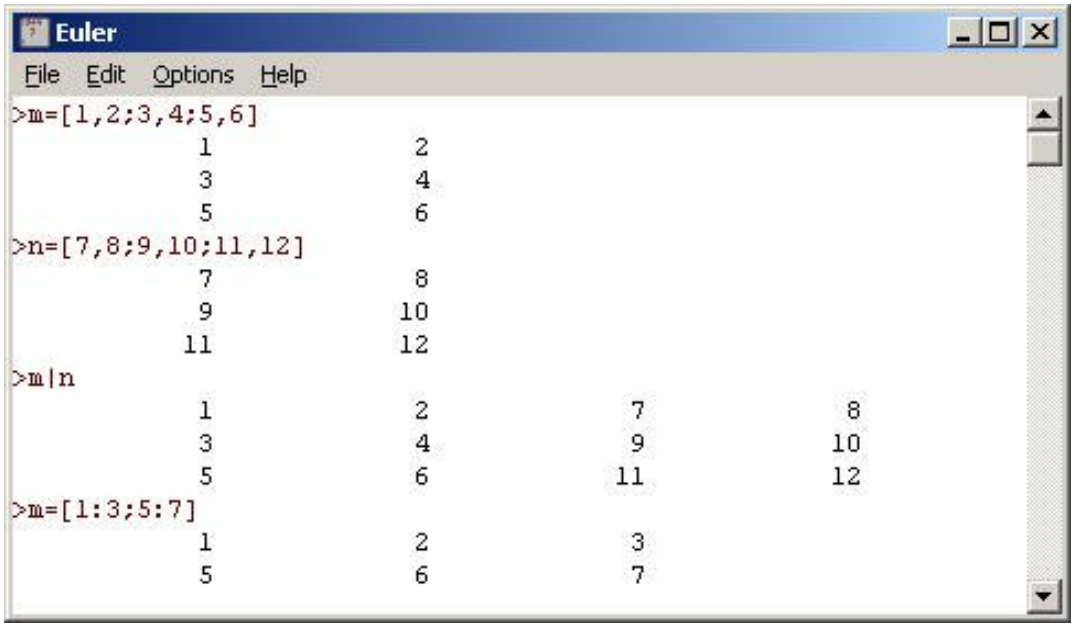
Obrázek 5: Vytváření a transpozice vektorů

3.3 Operace s maticemi

Matice v Euleru mohou stejně jako vektory obsahovat reálná nebo komplexní čísla. Stejně tak lze zadávat jejich jednotlivé prvky přímo z klávesnice nebo je nechat vygenerovat v daných mezích.

Při ručním zadávání se od sebe oddělují jednotlivé prvky na řádcích čárkou. Jednotlivé řádky se od sebe oddělují středníkem. Automaticky lze nechat vygenerovat pouze jednotlivé řádky.

Matice je možné stejně jako vektory transponovat pomocí apostrofu. Pomocí speciálních funkcí lze testovat, zda je daná matice čtvercová nebo symetrická. Euler umožňuje také vybrat z matice jeden prvek nebo submatici. Matice lze spojovat pomocí operátoru „|“.



Obrázek 6: Práce s maticemi

3.4 Vykreslování grafů

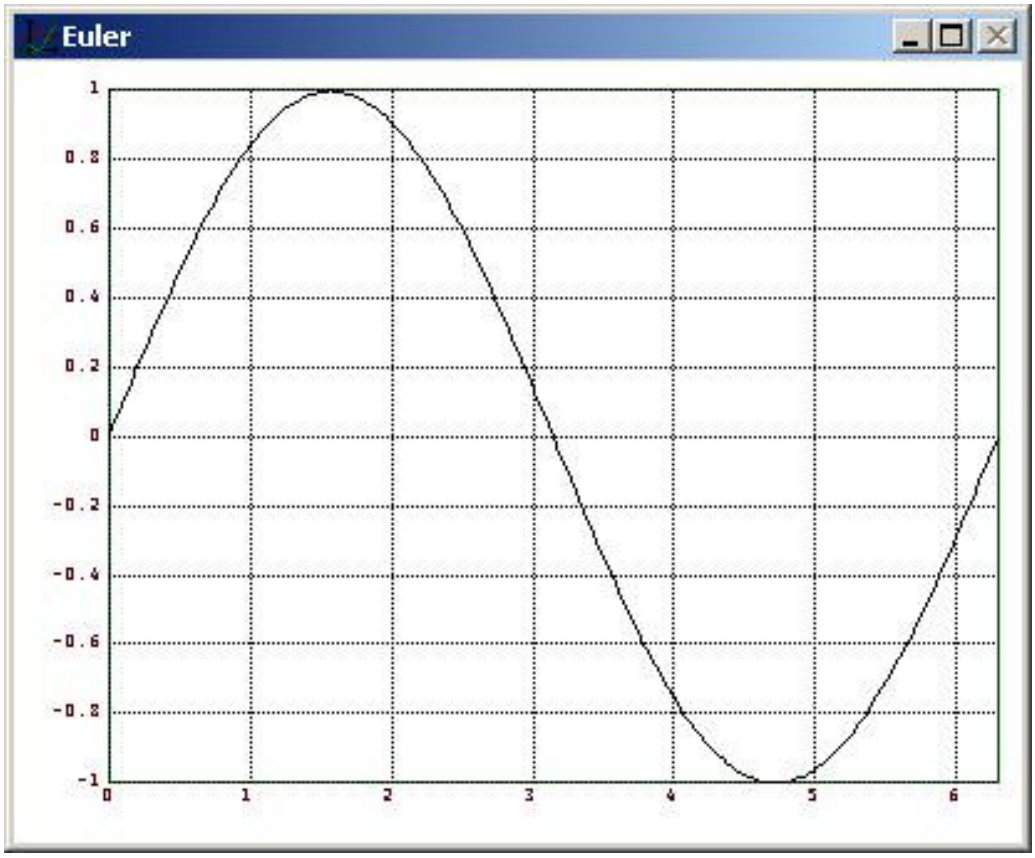
Euler nabízí poměrně bohaté možnosti vykreslování dvojrozměrných a především trojrozměrných grafů. Volit můžeme z mnoha druhů a vzhledů grafů, k dispozici jsou také grafy statistických a dalších funkcí.

3.4.1 2D grafy

Vykreslování grafů je v Euleru velmi podobné jako v Matlabu. Veškerý grafický výstup Euleru se zobrazuje v samostatném grafickém okně, které se otevírá automaticky, pokud má prováděný příkaz grafický výstup. K vykreslování základních 2D grafů slouží příkaz `plot(x,y)`. Proměnná x představuje vektor, jehož prvky jsou hodnoty, které se zobrazí na ose x, pokud zobrazování hodnot zapneme. Proměnná `y` je funkcí proměnné `x`, je také vektorem, jehož jednotlivými prvky jsou funkční hodnoty pro dané hodnoty prvků vektoru `x`. Funkce `plot` však vykresluje pouze jednoduché grafy bez mřížky a hodnot u jednotlivých os, proto byl v této práci vybrán příklad používající funkci `fplot`, jejíž výstup je více propracovaný.

Příklad: Vykreslení grafu funkce sin(x) pomocí funkce `fplot`

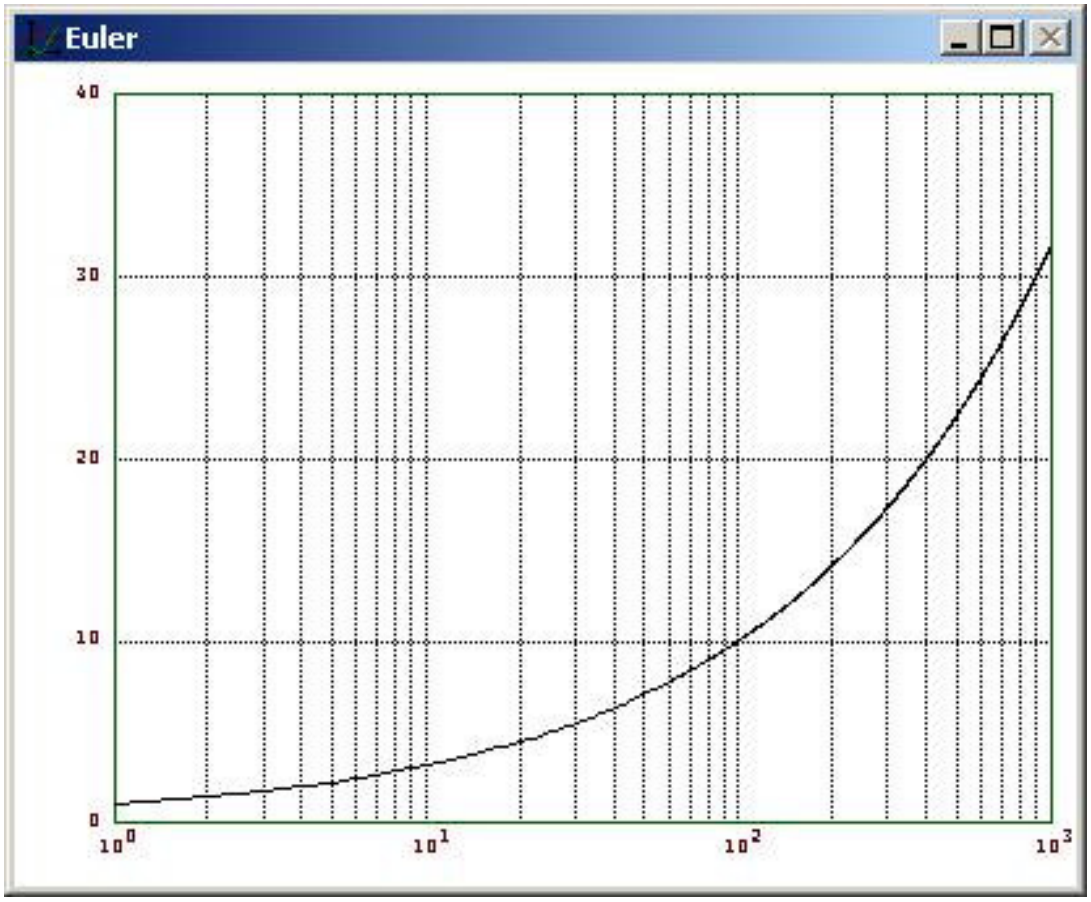
```
fplot("sin",0°,360°);
```



Obrázek 7: Vykreslení grafu funkce sin(x)

Příkladem grafu v logaritmických souřadnicích je graf využívající funkce `xlogplot`:

```
t=1:1000; ss=1:1000; s=sqrt(ss);
xlogplot(t,s);
```



Obrázek 8: Graf funkce xlogplot v logaritmických souřadnicích

3.4.2 3D grafy

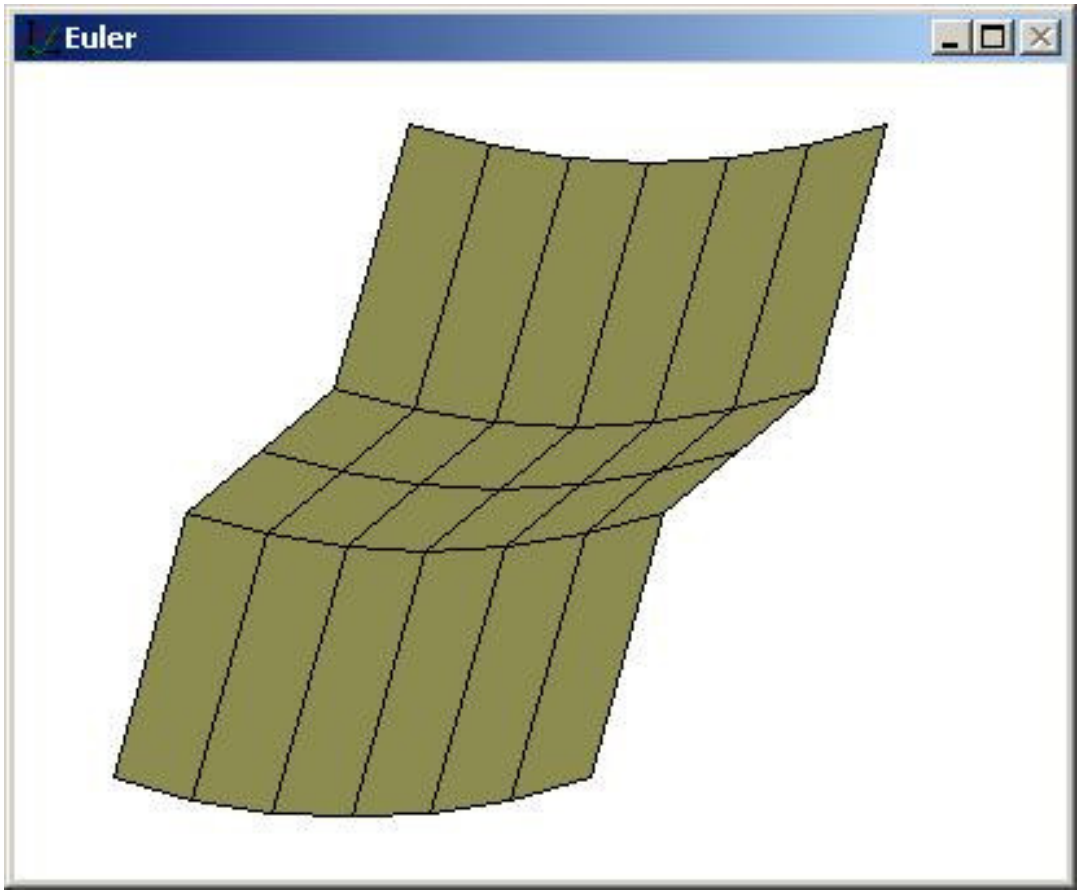
Trojrozměrné grafy v Euleru nabízí velké množství možností provedení. K jejich vykreslování slouží v Euleru funkce `mesh(m)`. Parametrem této funkce je matice `m`. Trojrozměrné grafy jsou funkcemi dvou proměnných. Proto je nejprve třeba vytvořit řádkový vektor pro hodnoty proměnné `x` a sloupcový vektor pro hodnoty proměnné `y`.

Následně může být vytvořena matice `m`, jejíž prvky jsou funkčními hodnotami funkce `z=f(x,y)`. Nad touto maticí je pak vytvořen plošný nebo jiný trojrozměrný graf.

Příkladem jednoduchého trojrozměrného grafu může být graf funkce $z = x^2 + y^3$:

```
>xmin=-1; xmax=1; nx=20;
>x=linspace(xmin,xmax,nx);
>ymin=-1; ymax=1; ny=20;
>y=(linspace(ymin,ymax,ny))';
```

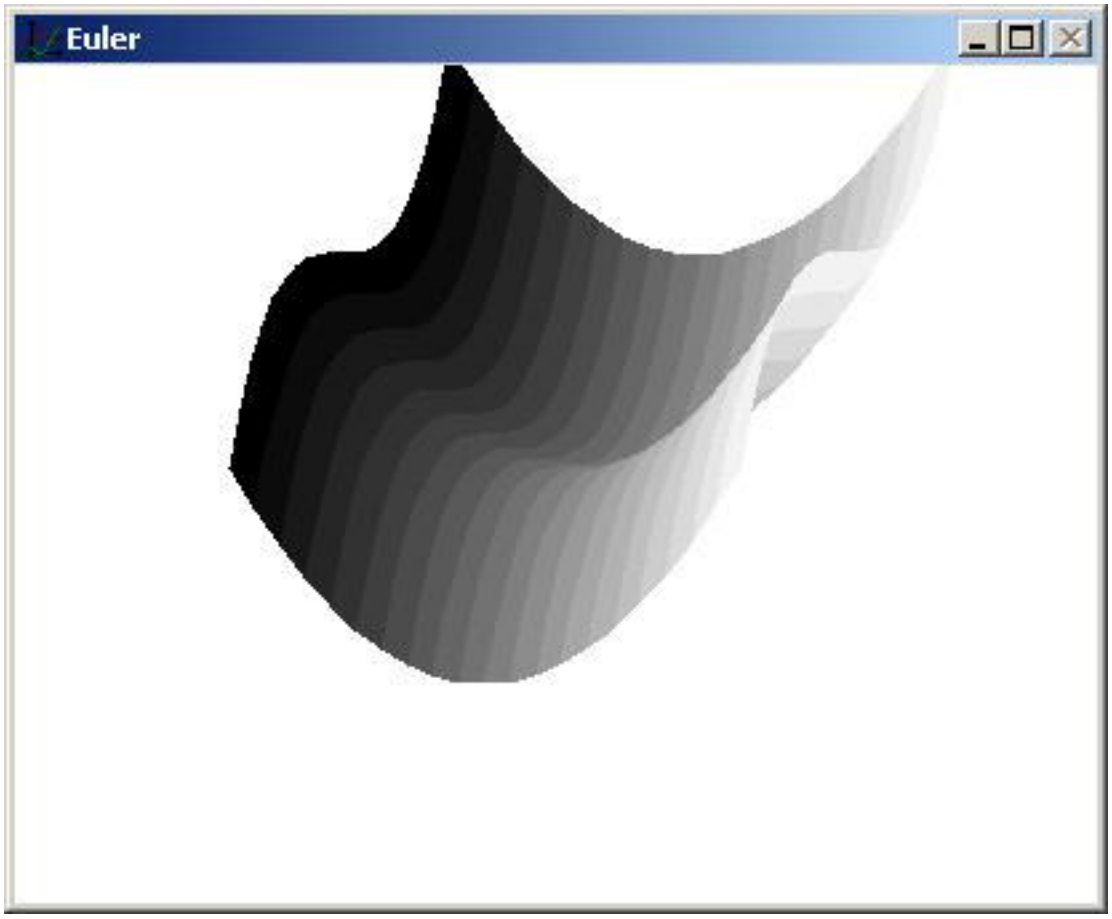
```
>m=x^2+y^3;
>view(100,58,23°, 0°); solidhue(x,y,m,x,1);
>
```



Obrázek 9: Příklad jednoduchého 3D grafu

Tento příklad nepůsobí příliš propracovaně. Euler je však schopný vytvářet také složitější grafy, jak ukazuje následující příklad:

```
xmin=-12; xmax=12; nx=6;
x=linspace(xmin,xmax,nx);
ymin=-10; ymax=10; ny=4;
y=(linspace(ymin,ymax,ny))';
m=x^2 + y^3;
fillcolor(12,14); twosides(1);
mesh(m);
```



Obrázek 10: Příklad složitějšího 3D grafu

3.5 Standardní příkazy a funkce Euleru

Euler nabízí několik standardních příkazů a funkcí, které lze využít jak při programování vlastních funkcí, tak při běžné práci v textovém okně.

3.5.1 Standardní příkazy

K standardním systémovým příkazům Euleru patří následující:

`clear` - vymaže všechny proměnné

`clg` - vymaže obsah grafického okna

`cls` - vymaže obsah textového okna a nastaví kurzor na začátek prvního řádku

`dump("filename")` - výstup všech příkazů v textovém okně se bude vypisovat jak na obrazovku, tak i do souboru s názvem "`filename`"

dump; - ruší předchozí příkaz, výstup bude opět pouze na obrazovku

free - vrací volné místo v zásobníku

forget("function") - odstraní z Euleru načtenou funkci "`function`"

help sin - zobrazí nápovědu k funkci `sin()`

hexdump(a) - vrátí výpis paměti proměnné a v hexadecimálním tvaru

list - vypíše všechny standardní funkce, příkazy a načtené funkce

listvar - vypíše všechny proměnné

memorydump - zobrazí všechny elementy na zásobníku

notebook "jmeno_souboru" otevře notebook s názvem "`jmeno_souboru`". Pokud se notebook nachází jinde než v aktuálním adresáři, musí být zadána jeho úplná cesta. K nastavení aktuálního adresáře slouží příkaz cd.

quit - ukončí Euler

remove("function") - smaže funkci "`function`"

shg - zobrazí grafické okno. Po stisku klávesy se vrátí opět do textového okna.

3.5.2 Standardní funkce

Euler má tyto standardní systémové funkce:

time - vrací čas v sekundách. Tato funkce je využívána také při zkoumání časových průběhů.

wait(n) - čeká *n* sekund nebo dokud není stisknuta klávesa. Vrací aktuální čas v sekundách.

wait - pokud je funkce `wait` zavolána bez argumentu, čeká tak dlouho, dokud není stisknuta klávesa. [5]

key - čeká na stisk klávesy a po stisknutí klávesy vrací její ASCII kód. Která klávesa byla stisknuta je možné testovat např. takto: `key()==ascii("a")`.

playwave - přehrává zvukové soubory "**.wav*".

4 TVORBA PRŮVODCE K PROGRAMU EULER

4.1 Příprava práce

Před započetím samotné práce na tvorbě webových stránek bylo nutno zvážit, jakým způsobem mají být stránky vytvořeny. Bylo třeba vybrat jednak technologii tvorby stránek, jazyky, v nichž se budou stránky vytvářet, a také programové prostředky, konkrétní programy, v nichž se budou stránky vytvářet. Při výběru těchto prostředků bylo třeba vzít v úvahu požadovanou funkci stránek, což je dostatečná interaktivita a příjemný vzhled stránek v nejběžněji používaných prohlížečích.

4.1.1 Výběr technologií k tvorbě stránek

Vzhledem k tomu, že k tvorbě stránek, jejichž hlavním cílem je zprostředkování informací v podobě obrázků a textu, není třeba používání žádných speciálních funkcí, k jejichž realizaci by bylo nutné využití skriptovací jazyků, stránky jsou vytvořeny v jazyce HTML v kombinaci s kaskádovými styly CSS.

Jazyk HTML byl vybrán zejména pro jeho jednoduchost a také proto, že mám s používáním tohoto jazyka dobré zkušenosti z dřívější doby. V jazyce HTML je vytvořena základní struktura všech stránek a také jejich propojení.

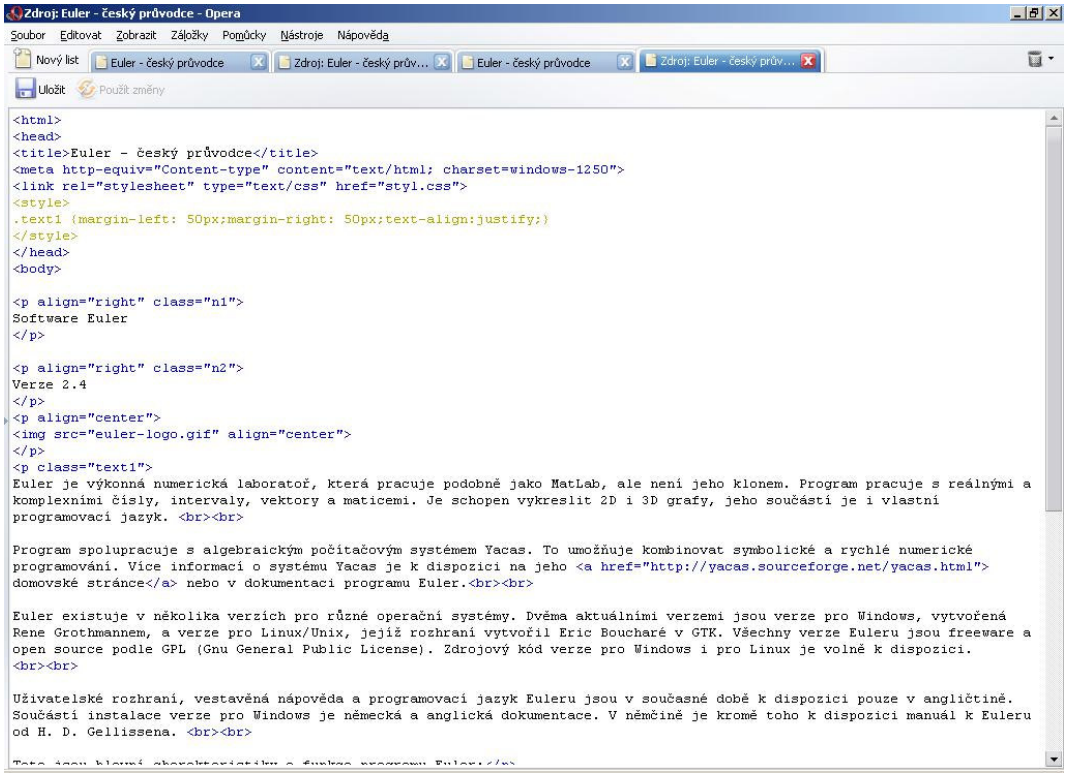
Aby byl vzhled všech stránek jednotný a aby bylo možné ho také snadno upravovat, byl zvolen pro definici vzhledu stránek kaskádové styly. Výhodou kaskádových stylů je kromě snadných změn celkového vzhledu také jednotná interpretace většiny vlastností jednotlivých prvků v různých prohlížečích, což by bylo pouze s použitím HTML obtížně realizovatelné. CSS také nabízejí více vzhledových vlastností a umožňují tak přesněji nastavit vzhled jednotlivých prvků.

4.1.2 Výběr softwarových prostředků

V současné době existuje na internetu velké množství volně dostupných editorů pro tvorbu webových stránek. Práce v těchto editorech je rychlá a pohodlná. Nevýhodou těchto editorů je však to, že zdrojový kód je generován automaticky na základě nastavení požadovaných vlastností jednotlivých prvků, což sice usnadňuje práci, ale na druhé straně podoba

zdrojového kódu v editoru nemusí přesně odpovídat výsledné podobě zdrojového kódu ve webové stránce.

Z tohoto důvodu byl zvolen časově náročnější způsob vytváření stránek v jednoduchém textovém editoru. V textovém editoru byla vytvořena jakási univerzální kostra stránek, skládající se z hlavičky s titulkem stránky a odkazem na externí stylopis a z ukončení stránky. Do této předem připravené univerzální formy jsem pak byl vkládán konkrétní obsah. Úpravy stránek byly prováděny v okně webového prohlížeče Opera, který zobrazuje zdrojový kód stránek se zvýrazněním syntaxe a umožňuje editaci a okamžitou aktualizaci provedených změn přímo v prohlížeči. Ačkoliv je tento způsob méně praktický a u stránek většího rozsahu by se zcela jistě nevyplatil, pro realizaci stránek menšího rozsahu se jeví jako dostačující.



Obrázek 11: Zobrazení zdrojového kódu v prohlížeči Opera

4.2 Práce na vlastní tvorbě HTML průvodce k programu Euler

Tvorbě webových stránek předcházela příprava podkladů, která spočívala v překladu větší části originální německé dokumentace do češtiny. Při této práci byl použit interaktivní německo-český slovník, dostupného na adrese *http://slovnik.seznam.cz* [23. 5. 2007], který

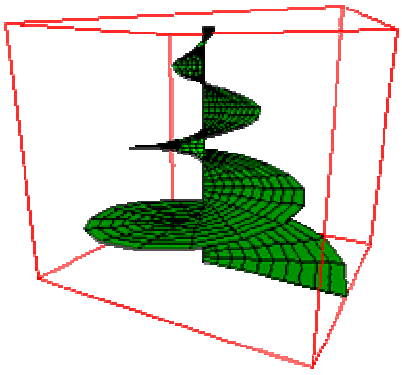
významnou měrou přispěl ke srozumitelnosti konečné podoby českého průvodce a také k co nejpřesnějšímu zachování původního informačního obsahu textů.

Snahou bylo nejen vytvořit uživatelsky přívětivého a informačně uspokojivého českého průvodce k programu Euler, ale také vytvořit českou alternativu k německým a anglickým stránkám týkajícím se tohoto programu. Konkrétně jde o stránky vytvořené autorem programu Euler Dr. Rene Grothmannem a zaměřené na verzi Euleru pro Windows.

4.2.1 Grafický návrh stránek

Protože jedním z cílů práce bylo vytvořit české stránky věnované programu Euler, které by se příliš nelišily od své německé a anglické předlohy, podřídil se této myšlence částečně také design stránek. Ačkoliv stránky nejsou vzhledově shodné s originálem, snahou bylo alespoň zčásti zachovat původní design, což se projevilo především ve výběru použitých barev.

Pro pozadí hlavní stránky byla zvolena světle šedá varianta loga Euleru, stejná jako je pozadí originálních stránek. Barvy pozadí záhlaví a menu stránky vycházejí z těchto barev. V záhlaví stránky nechybí ani logo Euleru, které je zmenšenou podobou loga umístěného na originálních stránkách. Podle původních stránek byla vybrána také červenohnědá barva odkazů, jejichž barva vychází z barvy loga.



.

Obrázek 12: Logo Euleru

4.2.2 Struktura stránek

Stejně jako u grafického vzhledu se také u struktury stránek vycházelo z uspořádání originálních anglických stránek Rene Grothmanna. Snahou nebylo přesné okopírování

struktury stránek, ale především zachování původního informačního obsahu. Proto například stránka *O Euleru* odpovídá po stránce obsahu originálním stránkám *About Euler* a *Features*. Stejně jako u anglických stránek byl vlastní průvodce k programu Euler realizován jako samostatná stránka, která se otevírá v novém okně. Z této stránky je možné přejít na hlavní stránky věnované Euleru kliknutím na logo Euleru v levém horním rohu.

Hlavní stránky obsahují následující menu:

- *O Euleru* – tato stránka slouží k základnímu představení programu Euler a jeho možností
- *Podporované OS* – zde se nachází přehled verzí Euleru pro různé operační systémy
- *Průvodce* – samotný průvodce programem Euler, který se otevírá v samostatném okně
- *Download* – odkazy umožňující stažení instalačních souborů jednotlivých verzí Euleru
- *Odkazy* – zde jsou k dispozici odkazy na stránky autorů jednotlivých verzí programu, originální německé a anglické stránky, stránky systému Yacas nebo stránky H. D. Gellissena, kde je možné stáhnout německý manuál k programu Euler

Samotný průvodce obsahuje tyto kapitoly:

- *Úvod*
- *Seznámení s programem Euler*
- *Práce v programu Euler*
- *Systémové příkazy a funkce*
- *Datové typy, konstanty a proměnné*
- *Operátory*
- *Funkce*
- *Vektory*
- *Matice*
- *Grafika*
- *Vstup a výstup dat*
- *Programování v Euleru*

4.2.3 Koncepce stránek

Rozdělení stránky na jednotlivé části je realizováno pomocí rámů. Rámy jsou celkem tři: záhlaví stránky, menu s odkazy a hlavní část s textem, kde se zobrazují po kliknutí na odkaz v menu příslušné stránky. Výška pruhu záhlaví a šířka menu je stanovena pevně, ostatní rozměry se přizpůsobují velikosti okna prohlížeče. Každý z rámů obsahuje samostatný soubor, přičemž záhlaví je stejné jak u hlavních stránek, tak u průvodce, nabídka menu se však liší.

K jednotnému vzhledu přispívá rozvržení pomocí rámů, kde záhlaví zůstává stále stejné a menu průvodce i menu hlavních stránek má stejné pozadí. Významný podíl na jednotném vzhledu mají také CSS styly, které jsou u všech souborů kromě záhlaví a obou menu realizovány pomocí externího stylopisu. To umožňuje upravovat vzhled všech prvků stejného významu na všech stránkách současně, pouze změnou definice stylu v souboru css. Veškeré úpravy jsou tak velmi jednoduché a rychlé.

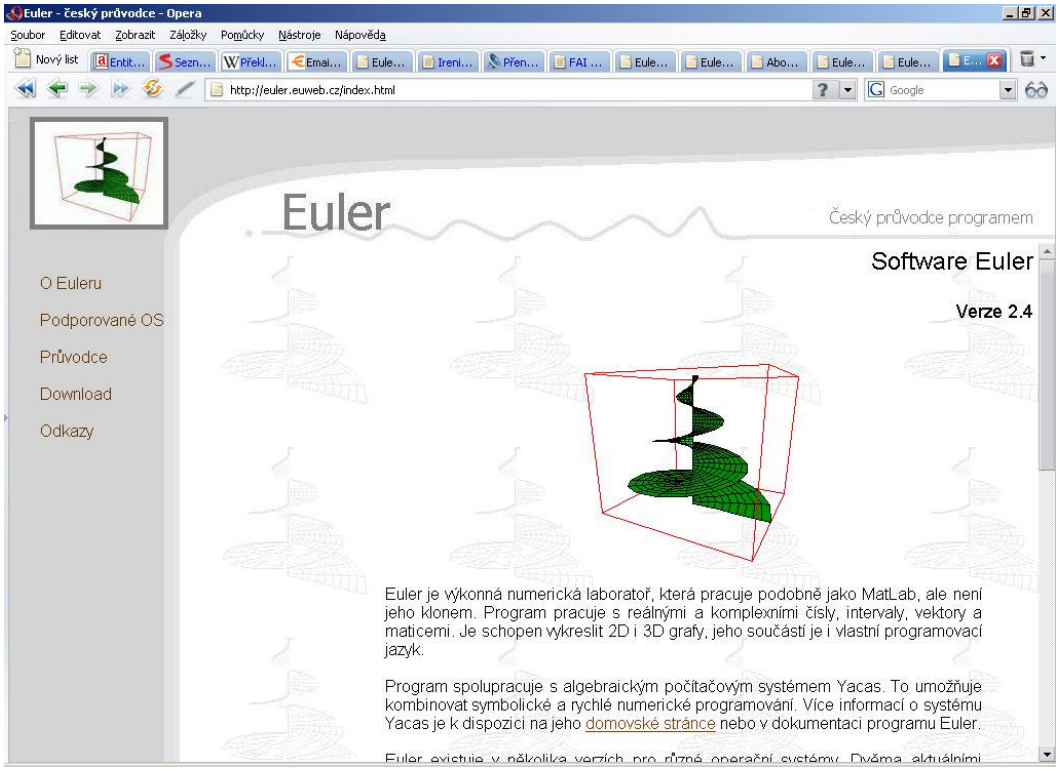
Obrázek 13: Rozvržení stránky pomocí rámů

Rozvržení stránky pomocí jednotlivých rámů na obrázku znázorňují červené čáry. Jak je vidět, rámy jsou umístěny ve dvou řádcích a druhý řádek je rozdělený ještě na dva sloupce.

Výhodou použití rámů je rychlejší zobrazování jejich obsahu v prohlížeči ve srovnání s tabulkami a menší celková délka zdrojového kódu. Nevýhodou je, že některé prohlížeče nemusí rámy podporovat. Při zkušebním zobrazení v Internet Exploreru, Firefoxu a Opeře se však stránky zobrazily správně a žádné výrazné problémy nebyly pozorovány.

4.2.4 Ukázka zdrojového kódu stránky

Výsledná stránka bez zvýraznění rozvržení rámů vypadá v prohlížeči takto:



Obrázek 14: Ukázka výsledného vzhledu stránky

A takto vypadá její zdrojový kód:

```
<html><head>
<title>Euler - český průvodce</title>
<meta http-equiv="Content-type" content="text/html;
charset=windows-1250">
</head>
<frameset rows="132, *" border="0" >
```

```
<frame src="pruh.html" noresize >
<frameset cols="165,*">
<frame src="menu2.html" >
<frame src="guide0.html" name=hlavni_ram >
</frameset>
<body>
<noframes>
Nacházíte se na stránce věnované programu Euler.
Omlouváme se, ale tato stránka zatím bohužel není
optimalizována pro prohlížeče, které nepodporují rámy.
</noframes>
<body></html>
```

Toto je kód hlavní stránky – obsahu, který je zobrazený v hlavním rámu:

```
<html>
<head>
<title>Euler - český průvodce</title>
<meta http-equiv="Content-type" content="text/html;
charset=windows-1250">
<link rel="stylesheet" type="text/css" href="styl.css">
<style>
.text1 {margin-left: 50px;margin-right: 50px;text-
align:justify;}
</style>
</head>
<body>
<p align="right" class="n1">Software Euler</p>
<p align="right" class="n2">Verze 2.4</p>
```

```
<p align="center">
<img src="euler-logo.gif" align="center">
</p>
<p class="text1">
Euler je výkonná numerická laboratoř, která pracuje podobně
jako MatLab, ale není jeho klonem. Program pracuje s reálnými
a komplexními čísly, intervaly, vektory a maticemi. Je
schopen vykreslit 2D i 3D grafy, jeho součástí je i vlastní
programovací jazyk. <br><br>

Program spolupracuje s algebraickým počítačovým systémem
Yacas. To umožňuje kombinovat symbolické a rychlé numerické
programování. Více informací o systému Yacas je k dispozici
na jeho <a href="http://yacas.sourceforge.net/yacas.html">
domovské stránce</a> nebo v dokumentaci programu
Euler.<br><br>
Euler existuje v několika verzích pro různé operační systémy.
Dvěma aktuálními verzemi jsou verze pro Windows, vytvořená
Rene Grothmannem, a verze pro Linux/Unix, jejíž rozhraní
vytvořil Eric Boucharé v GTK. Všechny verze Euleru jsou
freeware a open source podle GPL (Gnu General Public
License). Zdrojový kód verze pro Windows i pro Linux je volně
k dispozici. <br><br>
Uživatelské rozhraní, vestavěná nápověda a programovací jazyk
Euleru jsou v současné době k dispozici pouze v angličtině.
Součástí instalace verze pro Windows je německá a anglická
dokumentace. V němčině je kromě toho k dispozici manuál k
Euleru od H. D. Gellissena. <br><br>
Toto jsou hlavní charakteristiky a funkce programu Euler:</p>
<ul class="text1">
<li>Práce s reálnými a komplexními čísly, intervaly a
maticemi
```

```
<li>Vlastní programovací jazyk s lokálními proměnnými,
defaultními hodnotami parametrů, proměnlivým počtem
parametrů, parametry funkcí

<li>Vykreslování dvojrozměrných a trojrozměrných grafů,
bodových, obrysových a plošných grafů, animace a
stereografické grafy

<li>Numerická integrace (Romberg, Simpson, Gauss) a derivace,
diferenciální rovnice

<li>Minimalizace funkcí (Brent, Nelder-Mean), hledání kořenů
polynomů

<li>Statistické funkce a testy

<li>Simplexní algoritmus

<li>Interpolace a aproximace

<li>Volání funkcí v externích DLL souborech

<li>Přehrávání zvuku

<li>Rychlá Fourierova transformace (FFT)

<li>Možnost výpočtu přesných skalárních hodnot díky velkému
zásobníku

<li>Příkazové okno s možností úprav a komentářů, uživatelsky
přívětivé grafické rozhraní, interní a externí editor,

<li>Grafický výstup v různých formátech (WMF přes schránku),
postscriptový grafický výstup (díky Ericovi Boucharé)

<li>Symbolické výpočty s využitím systému Yacas.

</ul>

<br><br>

</body>

</html>
```

4.2.5 Umístění stránek na serveru

Aby byly stránky volně dostupné komukoliv, kdo se zajímá o práci s programem Euler, bylo třeba umístit je na webový server. Server byl vybrán podle dvou základních kritérií: hosting zdarma a možnost odkazování ze stránek na jiné servery, což některé freehostingové servery neumožňují. Proto byl zvolen server webzdarma.cz. Stránky byly umístěny na nově vytvořenou doménovou adresu http://euler.euweb.cz .

ZÁVĚR

Hlavním cílem této práce byla realizace českého průvodce k programu Euler, který je volně dostupnou alternativou komerčního softwaru Matlab od firmy MathWorks. Přestože Euler pracuje podobně jako Matlab, není klonem tohoto softwaru a v některých směrech se od něj liší. Proto může být v některých oblastech využití vhodnější než Matlab. Euler může dobře posloužit jako výukový program pro vzdělávací instituce, ale stejně tak může být i účinným pracovním nástrojem pro výpočty a simulace, jejichž výsledky budou využity v praxi.

Otázkou zůstává, zdá dát přednost komerčnímu softwaru, který představuje spolehlivý a ověřený pracovní nástroj, rozšířený do mnoha oblastí využití, a nabízí výhodu technické podporu týmu placených expertů, nebo využít výhody bezplatného softwaru, který ve své základní podobě nedosahuje takové úrovně komfortu a spolehlivosti, jako komerční software, ale jeho výhodou je velká flexibilita a možnost upravit si program na míru pro konkrétní způsob a oblast využití.

Snahou bylo vytvořit k programu Euler podklady v češtině, které budou dostatečně obsáhlé a srozumitelné, aby poskytly případným zájemcům dostačující základ pro efektivní práci s tímto softwarem. Průvodce ve formě internetových stránek má nejen nabídnout rychlý a pohodlný přístup ke konkrétní požadované informaci, ale především zpřístupnit tyto informace opravdu každému a přispět tak k dalšímu šíření tohoto softwaru a umožnit tím jeho další vývoj.

CONCLUSION

Creation of Czech guide for Euler program was the main aim of the thesis. Euler is freely available alternative software for commercial MATLAB software developed by The MathWorks. Although Euler works in similar way to Matlab, it is not a clone of this software and it diverges from Matlab in some ways. Therefore it may be for some applications more appropriate as Matlab. Euler may serve as a good learning program for educational institutions and it can be an effective tool for computing and simulation. The results can be used in praxis, too.

The question is, whether to choose a commercial software, which represents a safe and proved tool, expanded to many fields of application and offering an advantage of technical support of a team of experts, or to take an advantage of a freeware, which in its basic form does not manage such level of comfort and reliability as a commercial software, but its advantage is a great flexibility and a possibility to modify the program for specific ways and fields of application.

My effort was to create Czech materials for Euler program, which are comprehensive and understandable enough to afford appropriate persons interested in work with this program a sufficient base for effective work with it. The guide in the form of web pages not only offers a quick and comfortable access to required information, but using all above makes this information available really for everybody and helps to further propagation of this software together with the support of its further development.

SEZNAM POUŽITÉ LITERATURY

[1] Broža P. (2000): Programování www stránek pro úplné začátečníky. Computer Press, Praha, ISBN 80-7226-278-5

[2] Kristián, P. (2001): Návrh webu a Frontpage 2000. UNIS Publishing, ISBN 80-86097-57-9

[3] Kučera, M. (2001): HTML - tipy a triky od profesionálů. UNIS Publishing, ISBN 80-86097-64-1

[4] Niederst, J. (1999): Web Design in a Nutshell. O'Reilly, USA, ISBN 1-56592-515-7.

[5] http://mathsrv.ku-eichstaett.de/MGF/homes/grothmann/euler/german/index.html (Euler Dokumentation) [cit. 17.5.2007]

[6] http://cs.wikipedia.org/wiki/HTML [cit. 17.5.2007]

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

ASP	Active Server Pages
CERN	Centre Européan pour Recherche Nucléaire
CGI	Common Gateway Interface
CSS	Cascading Style Sheets
DOM	Document Object Model
DTD	Document Type Definition
FI	Form Interpreter
FTP	File Transfer Protocol
GIF	Graphic Interchange Format
GNU	GNU is Not Unix
GPL	General Public License
GTK	GIMP ToolKit
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
IMAP	Interactive Mail Access Protocol
INRIA	l'Institut National de Recherche en Informatique et en Automatique
ISO	International Standards Organization
JPEG	Joint Photographic Expert Group
LDAP	Lightweight Directory Access Protocol
MathML	Mathematical Markup Language
MIME	Multipurpose Internet Mail Extension
MP	Mobile profile
MSSQL	MicroSoft Structured Query Language
NCSA	National Centre for Supercomputer Applications

NLS	oN-Line System
ODBC	Open DataBase Connectivity
PDA	Personal Digital Assistant
PHP	Personal Home Page
PNG	Portable Network Graphics
POP3	Post Office Protocol 3
RDF	Resource Description Framework
RSS	Really Simple Syndication
SGML	Standard Generalized Markup Language
SMIL	Synchronized Multimedia Integration Language
SMTP	Simple Mail Transfer Protocol
SNMP	Simple Network Management Protocol
SOAP	Simple Object Access Protocol
SQL	Structured Query Language
SVG	Scalable Vector Graphics
TSW	Tanggaard SoftWare
URL	Uniform Resource Locator
UTF-8	Uniform Transformation Format
VBScript	Visual Basic Script
W3C	World Wide Web Consortium
WAP	Wireless Application Protocol
WWW	World Wide Web
XHTML	eXtensible HyterText Markup Language
XML	eXtensible Markup Language
XSL	eXtensible Stylesheet Language

SEZNAM OBRÁZKŮ

Obrázek 1: Zobrazení stránky v prohlížeči *22*
Obrázek 2: Euler po spuštění čeká na zadání prvního příkazu *46*
Obrázek 3: Ukázka základní práce v Euleru, chybové hlášky a příkazu help *48*
Obrázek 4: Přiřazování hodnot proměnným a funkce typeof() *50*
Obrázek 5: Vytváření a transpozice vektorů *51*
Obrázek 6: Práce s maticemi *52*
Obrázek 7: Vykreslení grafu funkce sin(x) *53*
Obrázek 8: Graf funkce xlogplot v logaritmických souřadnicích *54*
Obrázek 9: Příklad jednoduchého 3D grafu *55*
Obrázek 10: Příklad složitějšího 3D grafu *56*
Obrázek 11: Zobrazení zdrojového kódu v prohlížeči Opera *59*
Obrázek 12: Logo Euleru *60*
Obrázek 13: Rozvržení stránky pomocí rámů *62*
Obrázek 14: Ukázka výsledného vzhledu stránky *63*

SEZNAM PŘÍLOH

PŘÍLOHA P I: CD-ROM (obsahuje samotnou bakalářskou práci, průvodce programem Euler ve formě www stránek a prezentaci v MS PowerPoint)