

WWW stránky pro podporu předmětu „Teorie automatického řízení I“

Jiří Giesl

Bakalářská práce
2005



Univerzita Tomáše Bati ve Zlíně
Fakulta technologická
Institut řízení procesů a aplikované informatiky

Vložit oficiální zadání bakalářské práce

Děkuji Ing. Radkovi Matušů za vedení mé bakalářské práce, za jeho věcné připomínky v průběhu řešení práce, poskytnuté materiály a ochotu při řešení problémů.

Souhlasím s tím, že s výsledky mé práce může být naloženo podle uvážení vedoucího bakalářské práce, ředitele ústavu a institutu. V případě publikace budu uveden jako spoluautor.

Prohlašuji, že jsem na celé bakalářské práci pracoval samostatně a použitou literaturu jsem citoval.

Ve Zlíně, 24. 05. 2005

.....

podpis

RESUMÉ

Tato bakalářská práce se zabývá realizací elektronické učebnice v podobě webových stránek pro předmět *Teorie automatického řízení I*. Obsahově vychází ze skript „Teorie automatického řízení - lineární spojité dynamické systémy“ od autorů: Prokop, R., Matušů, R., Prokopová, Z. K vytvoření webových stránek byl použit jazyk HTML ve spojení s profesionálními skriptovacími jazyky a doplňky, jako je PHP, JavaScript a CSS. Součástí stránek jsou také algoritmy napsané v PHP pro návrh základních typů spojitých regulátorů.

SUMMARY

My thesis deals with the realization of electronic textbook in websites form for academic purposes *Theory of automatic process control I*. Its content was taken from publication ‘Teorie automatického řízení - lineární spojité dynamické systémy’, written by: Prokop, R., Matušů, R., Prokopová, Z. For creating websites was used HTML language with the support of professional script languages and complements such as PHP, JavaScript and CSS. Algorithms for realization of basic types of continuous regulators written in PHP are one of the components of websites.

OBSAH

ÚVOD.....	6
1 TEORETICKÁ ČÁST.....	7
1.1 Literární rešerše.....	7
1.1.1 Knižní publikace.....	7
1.1.2 Vysokoškolská skripta.....	9
1.1.3 Elektronické učebnice	11
1.2 HTML.....	14
1.2.1 Tagy.....	14
1.2.2 Vlastnosti tagů.....	15
1.2.3 Struktura HTML kódu.....	16
1.2.4 Úprava textu	16
1.2.5 Obrázky	17
1.2.6 Odkazy.....	18
1.2.7 Rámy.....	20
1.2.8 Tabulky.....	22
1.2.9 HTML editory a postupné psaní tagů.....	24
1.3 CSS Styly	25
1.4 JavaScript.....	27
1.5 PHP	29
2 PRAKTICKÁ ČÁST.....	31
2.1 Tvorba HTML stránek.....	31
2.1.1 Návrh vzhledu internetových stránek	31
2.1.2 Tvorba základních elementů webové stránky.....	34
2.2 Tvorba menu pomocí JavaScriptu	37
2.3 PHP skript pro výpočet spojitého regulátoru	42
2.3.1 PHP Triad	42
2.3.2 Programování skriptu	42
SHRNUTÍ PRÁCE.....	50
ZÁVĚR.....	52
SEZNAM POUŽITÉ LITERATURY	53
SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK.....	54
SEZNAM OBRÁZKŮ.....	55
SEZNAM TABULEK	56
SEZNAM PŘÍLOH.....	57

ÚVOD

V době velkého vzrůstu informačních technologií a jejího čím dál většího využití ve všech oblastech lidského života se stal Internet něčím nepostradatelným. Zvláště pro studenty, kteří Internet potřebují ke svému každodennímu vzdělávání. Internet jim nabízí velkou studnu informací, ze které mohou čerpat. Tyto informace jsou zpravidla prezentovány ve formě webových stránek vytvořených pomocí HTML jazyka. Webové stránky mají oproti tištěné literatury velkou výhodu: jejich úprava a doplňování informací se může dít nezávisle na čtenáři a je čtenáři distribuována velmi jednoduchou cestou. To vše většinou navíc ještě bezplatně, protože publikování pomocí webových stránek s sebou nese několikanásobně nižší náklady než publikování tištěnou formou. Tím se zvyšuje i okruh autorů elektronických publikací a jejich práce mnohdy může přinést nové poznatky a myšlenky k určitému oboru.

I když se každou chvíli na Internetu objevují nové elektronické publikace a webové stránky s technickou či humanitní tematikou určenou pro vzdělávání, přesto bude mít tištěná literatura u čtenářů vždy svou prioritu. Využití elektronických publikací je především v rychlém vyhledání určité informace nebo v získání celistvého přehledu k určité problematice.

1 TEORETICKÁ ČÁST

1.1 Literární rešerše

Tato literární rešerše byla zpracována jako součást bakalářské práce. Tvoří ji celkem 13 publikací. Většina uvedených prací je přístupná v univerzitní knihovně UTB ve Zlíně, na elektronické učebnice je uveden příslušný odkaz jejich webové adresy.

Rešerše je rozdělena do následujících částí:

- 1.1.1 Knižní publikace
- 1.1.2 Vysokoškolská skripta
- 1.1.3 Elektronické učebnice

Pozn.: Publikace, které se nepodařilo sehnat v knihovně či v obchodech s odbornou literaturou a proto nemohly být v této literární rešerši důkladně popsány, jsou uvedeny vždy ke konci podkapitoly.

1.1.1 Knižní publikace

Balátě, J.: Automatické řízení. BEN – technická literatura, Praha, 2003.

- tato kniha je určena především odborné veřejnosti, která v ní nalezne bohaté autorovy znalosti z oboru teorie automatického řízení shrnuté do srozumitelné formy. Kniha je systémově rozdělena do 6 kapitol, které shrnují velké množství informací do logicky rozčleněného celku.

1. Systémový úvod pro teorii automatického řízení – zabývá se definicemi a základními pojmy z oblasti automatického řízení a tyto pojmy klasifikuje do několika tříd.
2. Teorie lineární regulace – diskutuje o analýze a syntéze lineárního regulačního obvodu. Analýza se mimo jiné zabývá Laplaceovou transformací, popisem vlastností systémů, blokovou algebrou, vlastnostmi regulátorů a kritérii stability. Syntéza pak řeší problém volby regulátoru a jakosti regulačního pochodu. Zabývá se také servomechanizmy, rozvětvenými a mnohorozměrnými regulačními obvody.
3. Teorie nelineární regulace – vymezuje problém nelinearity regulačních obvodů, jejich popis, řešení a stabilitu.

4. Diskrétní systémy řízení – řeší základní problémy diskretizace řízení (vzorkování, tvarování, Z-transformaci). Podobně jako ve spojitých regulačních obvodech se zabývá blokovou algebrou, stabilitou diskrétních obvodů a základními algoritmy řízení.
5. Popis systému ve stavovém prostoru – pojednává o převodu diferenciálních rovnic na přenosové matice, které se používají především k popisu mnohorozměrných systémů a jejich vlastností (dosažitelnost, pozorovatelnost, atd.)
6. Doplnky – nabízí přehled základních vlastností Laplaceovy a Z-transformace, jejich slovníky a vysvětluje i základní operace s maticemi.

Švarc, I.: Automatizace. Automatické řízení. Akademické nakladatelství CERM, Brno, 2002.

- jedná se o velmi dobře a přehledně napsaný text, který vysvětluje všechny důležité pojmy automatizace srozumitelným jazykem, který nedělá problémy ani začátečníkovi. Pro studenty, kteří již jsou s problematikou seznámeni pak představuje zajímavé sjednocení dosavadních znalostí a nový pohled na problematiku automatického řízení.

1. Logické řízení – věnuje se kybernetice a jejímu využití v praxi, realizací logických funkcí a využitím programovatelných automatů.
2. Spojité lineární řízení – obsahuje přehledné pojednání o popisu regulačních obvodů pomocí diferenciálních rovnic a charakteristik. Věnuje se také regulátorům, jejich vlastnostem, konstrukčním sestavením a některými metodami seřízení. Nechybí ani zmínka o mnohorozměrných obvodech a jejich autonomnosti.
3. Diskrétní řízení – vymezuje problémy popisu diskrétních obvodů a jejich řízení pomocí číslicových regulátorů.
4. Fuzzy řízení – diskutuje o principech fuzzy logiky, fuzzifikace a realizace fuzzy regulátorů.
5. Nelineární systémy – jedná se převážně o vysvětlení grafického řešení nelineárních rovnic tzv. metodou stavové roviny a nabízí také několik kritérií pro vyšetření stability nelineárních systémů.

1.1.2 Vysokoškolská skripta

Prokop, R., Matušů, R., Prokopová, Z.: Teorie automatického řízení – lineární spojité dynamické systémy. Rukopis, skriptum FT UTB ve Zlíně, 2005.

- jedná se o nejnovější publikaci určenou především pro studenty a posluchače předmětu TAŘ I, kteří v ní najdou znalosti a informace potřebné k základnímu pochopení problematiky lineárních spojitých dynamických systémů. Učební text je rozdělen do 6 kapitol, které probíranou látku prezentují jednoduchou a pochopitelnou formou.

1. Úvod do teorie systémů – vedle základních pojmů a definic rozděluje systémy do skupin podle jejich vlastností. Popisuje také regulační obvod a názvosloví jeho veličin.
2. Lineární spojité dynamické systémy – mimo problému Laplaceovy transformace řeší otázku popisu systému, jeho stability (příslušná kritéria) a chování systému s dopravním zpožděním.
3. Spojité regulátory a metody jejich nastavení – popisuje jednotlivé složky regulátorů a jejich vzájemné kombinace, dále také nabízí přehled nepoužívanějších metod jejich nastavení.
4. Polynomiální metody návrhu regulátorů – pojednává o řešení diofantických rovnic a jejich využití při návrhu regulátoru a jeho parametrů. Představuje principy 1DOF a 2DOF syntézy, využití Matlabu při této syntéze a způsoby aproximace dopravního zpoždění.
5. Mnohorozměrné systémy – představuje pojem mnohorozměrný systém a ukazuje jeho zápis do maticového tvaru. V neposlední řadě se také zabývá návrhem regulátoru pro tyto systémy pomocí elementárních maticových úprav.
6. Popis systémů ve stavovém prostoru – zabývá se vnitřním popisem systémů v časové oblasti, převodem diferenciální rovnice na stavový popis a naopak a dalšími pojmy souvisejícími se stavovým modelem.

Bobál, V., Vašek, V.: Teorie automatického řízení I, skriptum, ES VUT Brno, 1989.

- přestože se jedná již o starší publikaci, obsahově stále, až na pár výjimek, pokrývá rozsah předmětu TAŘ I, tak jak je dnes přednášen na FT UTB ve Zlíně a může tedy stále sloužit jako stěžejní publikace pro dobrý základ znalostí o automatickém řízení.

1. Úvod – základní pojmy – stejně jako předešlé publikace se zde jedná o definice pojmů z oblasti automatického řízení.
2. Základy teorie dynamických systémů – věnuje se rozčlenění a popisu dynamických systémů, jejich struktuře a chování.
3. Lineární spojité systémy – shrnuje v následujících podkapitolách analýzu, syntézu a stabilitu regulačního obvodu, jednotlivé druhy regulátorů. Věnuje se také řízení mnohorozměrových obvodů.
 - a. Analýza vlastností řízených spojitých systémů
 - b. Určení dynamických vlastností systémů
 - c. Regulátory
 - d. Analýza regulačního obvodu
 - e. Stabilita spojitých lineárních systémů
 - f. Syntéza regulačního obvodu
 - g. Mnohorozměrové a rozvětvené regulační obvody

Spal, J., Madarász, L.: Teória automatického riadenia I, ALFA Bratislava, 1985.

- skripta nebyla napsána v zázemí FT, tudíž nelze od textu čekat, že se svým obsahem plně přiblíží přednášené látce. Přesto jsou tato skripta vhodná jako doplňující materiál ke studiu, protože v jednotlivých kapitolách může student najít nové informace, které dopomůžou k celkovému přehledu o problematice lineárního řízení.

1. Dynamické systémy – věnuje se opět základním pojmům a definicím, ale omezuje se pouze na dynamické systémy.
2. Opis spojitých lineárních systémů – pojednání o způsobech popisu systémů, jako jsou např. lineární dif. rovnice, přechodové, impulsní a jiné charakteristiky.
3. Dynamické vlastnosti lineárních přenosů – zabývá se vnitřním a vnějším popisem dynamických vlastností, ukazuje použití některých kritérií pro zjištění stability systému, apod.
4. Akost' dynamických procesů – představuje základní metody nastavení spojitých regulátorů a různé kombinace jejich členů.
5. Syntéza lineárních obvodů – stanovení struktury a parametrů regulačního obvodu na základě určitých požadavků.
6. Mnohoparametrové a zložené systémy – shrnuje poznatky o mnohorozměrových systémech, jejich analýze a řízení.

Prokop, R.: Základy automatizace pro bakalářské studium, skripta FT UTB ve Zlíně, 1998.

Hanuš, B., Balátě, J., Švarc, I., Zikeš, F.: Teorie automatického řízení I, skriptum VŠST Liberec, 1982.

Kubík, S., Kotek, Z., Strejc, V., Štecha, J.: Teorie automatického řízení I. Lineární a nelineární systémy, SNTL, ALFA, Praha, 1982.

Mikeš, J., Fikar, M.: Modely a dynamické charakteristiky spjitých procesov, STU Bratislava, 1999.

1.1.3 Elektronické učebnice

Modrlák, O.: Teorie automatického řízení I, PDF skriptum Technické univerzity v Liberci, 2004.

- skriptum se skládá z několika PDF souborů, které jsou rozvrženy podle okruhů problematiky. Protože jsou určeny pro výuku předmětu na Technické univerzitě v Liberci, jsou obsahově velmi rozdílné od požadavků kladených na studenty FT UTB ve Zlíně, přesto mohou sloužit jako vhodná pomůcka pro doplnění a rozšíření celkových znalostí. Je ovšem nutné dodat, že k plnému pochopení učebního textu je potřeba již mít určité znalosti z teorie automatického řízení, protože text je pro začátečníka napsán příliš technicky a velmi podrobně.

- Analýza dynamických systémů – vymezuje základní pojmy z oblasti dynamických systémů, jejich popisem v časové oblasti (dif. rovnice, obrazový přenos, bloková algebra, apod.)
- Úvod do identifikace – obsahuje základní metody identifikace systémů, včetně podpory softwarových prostředků v prostředí Matlab (pro potřeby předmětu TAŘ I tato kapitola není plně potřebná).
- Syntéza regulačních obvodů – zabývá se strukturou regulačních obvodů, návrhem regulátorů, jakostí regulačního pochodu a různými metodami seřízení regulátoru.
- Syntéza ve frekvenční oblasti, Citlivost a robustnost regulačních obvodů – tyto kapitoly jsou pro studenty FT UTB ve Zlíně vhodné jen jako doplnění vzdělání.

Zabývají se návrhy regulátorů v závislostech na frekvenčních vlastnostech systémů a návrhy regulátorů na základě předem zvoleného kritéria jakosti regulace a dalších omezujících podmínek.

- Systém s více vstupy a výstupy – MIMO systémy – shrnuje poznatky o matematickém popisu MIMO systémů a dále o blokové algebře těchto systémů. Najdeme zde také pojednání o autonomní regulaci.

URL: <http://www.fm.vslib.cz/~krtsub/fm/tr1/materialy.htm>

Švarc, I.: Základy automatizace, PDF skriptum VUT Brno, 2002.

- skripta se zabývají především analýzou regulačního obvodu. Obsahově vycházejí z knihy *Automatizace, Automatické řízení* od stejného autora.

1. Logické řízení – logické funkce, jejich realizace a základní pojednání o programovatelných automatech.
2. Spojité lineární řízení – zabývá se klasickými pojmy z analýzy regulačního obvodu.
3. Diskrétní řízení – stručný a přehledný úvod do problematiky diskrétního regulačního obvodu (Z-transformace, diferenční rovnice, číslicové regulátory, stabilita diskrétního obvodu, apod.)

URL: <http://www.fme.vutbr.cz/iso/opory/pdf/ZakladyAutomatizace.pdf>

Švarc, I.: Teorie automatického řízení, PDF skriptum VUT Brno, 2003.

- skripta do jisté míry navazují na výše uvedená skripta (Švarc, I.: *Základy automatizace*) a rozvíjí nastíněné téma do odbornější roviny. Přesto si uchovávají srozumitelný jazyk.

1. Stavové rovnice – řeší popis jak spojitých, tak diskrétních systémů ve stavovém prostoru a nastiňuje řešení stavových rovnic autonomních i neautonomních systémů.
2. Přesnost regulace – pojednává o statické přesnosti regulace, tj. po odeznění všech přechodových jevů a o vzniku trvalé regulační odchylky.
3. Kvalita regulace – zabývá se základními metodami seřízení regulátorů podle volených kritérií kvality regulace.
4. Rozvětvené regulační obvody – kapitola obsahuje výklad o regulačních obvodech s větším počtem vazeb, tzn. o obvodech s pomocnou akční veličinou, regulovanou veličinou a o modelech s měřením poruchové veličiny.

5. Mnohorozměrné regulační obvody – základní pojmy a definice z oblasti mnohorozměrných systémů a jejich autonomnosti.
6. Teorie nelineárního řízení – obsahuje obecné pojmy nelineárního řízení, vysvětluje metodu stavové roviny a objasňuje stabilitu nelineárních systémů.

URL: http://www.fme.vutbr.cz/iso/opory/pdf/uai/teorie_automat_řízení/TAR03.pdf

Šerý, M.: *Automatizace*, 2002.

- učební text je prezentován ve formě webových stránek, které jsou velmi jednoduché. Jsou rozděleny do 11 kapitol, které obsahují heslovitě rozdělený text. Stránky jsou vhodné především pro rychlé vyhledání určité informace či pojmu.

1. Úvod do matematického řešení
2. Vlastnosti členů regulačního obvodu
3. Základní typové dynamické články
4. Členy s dopravním zpožděním
5. Regulátory
6. Určování celkového přenosu z přenosů dílčích
7. Modelování na analogovém modelu
8. Aproximace přenosových funkcí
9. Konstrukce frekvenční charakteristiky
10. Logaritmické charakteristiky
11. Stabilita regulačního obvodu

URL:

http://www.eamos.cz/amos/kat_fyz/modules/external/index.php?kod_kurzu=kat_fyz_5377

Výše uvedené publikace obsahují vyčerpávající pojednání o lineárních dynamických systémech, jejich analýze, syntéze atd. Nezmiňovali jsme se o starších publikacích, tj. vydání před rokem 1980, protože novější vydání obsahově vycházejí ze starších a čtenáři si starší vydání obtížněji shánějí. Pro studenty je samozřejmě výhodné si publikace zakoupit ve formě vysokoškolských skript (dříve vydaná skripta jsou přístupná v univerzitní knihovně), odborná veřejnost pak dá zřejmě raději přednost knižním vydáním popřípadě elektronickým publikacím.

1.2 HTML

HTML (Hypertext Markup Language) je kód, který slouží k zobrazování dat předem zadaným způsobem. V průběhu let vývoje Internetu se jazyk HTML postupně zdokonaloval. Nejedná se tedy v žádném případě o jednu navržený kód – HTML se dynamicky mění v souladu s technologickým vývojem a požadavky jeho uživatelů.

HTML má svůj zdrojový text, který popisuje grafickou podobu stránky v prohlížeči. Jako každý jazyk i HTML má svoji přesnou syntaxi, kterou je nutno dodržovat, ovšem zároveň je velice přizpůsobivý. I když se designér dopustí nějakých chyb, dokáže většinou prohlížeč zobrazit prakticky celou stránku tak, jak vypadat má. Více než tři čtvrtiny stránek na Internetu nemají zcela správnou syntaxi, a přesto jsou zobrazeny správně. Tato flexibilita prohlížečů je dána rychlým vývojem HTML, prohlížeč tedy přímo předpokládá, že se setká s nesprávnou syntaxí, jako je vynechání značky, špatné formátování, chybějící uvozovky u hodnot parametrů, apod.

1.2.1 Tagy

HTML soubor je obyčejný text obalený značkami, které se nazývají tagy.

- Tagy určují, jak bude text vypadat, tedy jakou bude mít formu.
- Všechny tagy jsou uzavřeny v <ostrých závorkách>.

Co není v ostrých závorkách, je text, který se bude zobrazovat.

Tagy mohou být dvojího druhu:

1. Párové - mají úvodní a koncový tag (koncovému tagu je předřazeno lomítko).
2. Samostatné - u tohoto tagu není třeba koncový (uzavírací) tag.

V úvodním tagu můžou být, pomocí dodatkových atributů, určeny další parametry pro formátování textu a jiných elementů webové stránky. Každý tag má svůj vlastní seznam atributů, které ho rozšiřují.

Tab. 1 Zápis tagu

Popis	Vývoj příkladu
Značka tagu začíná levou ostrou závorkou.	<
Za ní následuje jméno tagu, před kterým nesmí být mezera.	<font
Mohou následovat atributy. Před každým musí být alespoň jedna mezera.	<font color
Za atributem se píše rovnítko a hodnota v uvozovkách. Vše bez mezer.	<font color="blue"
Atributů může být několik.	<font color="blue" size="6"
Tag končí pravou ostrou závorkou.	
Následuje vlastní text, který se zobrazí.	Modrý velký text
Element končí ukončovací tagem s lomítkem a bez atributů.	Modrý velký text

1.2.2 Vlastnosti tagů

HTML má několik málo zásad, které je dobré zmínit:

- Nezáleží na velikosti znaků, <body> je totéž co <BODY> (HTML je jazyk typu „non-case-sensitive“)
- V adresách a jménech souborů záleží na velikosti znaků, nesmějí mezi nimi být mezery a česká diakritika.
- Tagy, které prohlížeč nezná, jako by neexistovaly.
- Na začátku tagu nesmí být mezera, např. < br> je nesprávný zápis.
- Dvě mezery po sobě (nebo více) mají stejný význam jako jedna mezera.
- Konec řádku ve zdrojovém kódu se chápe jako mezera.
- Jména atributů je vhodné dávat vždy do uvozovek, ale není to úplně nutné, pokud uvnitř nejsou mezery.
- Znakové entity se do zdrojového kódu zadávají jako posloupnost znaků &něco; (například znak © je © , pevná mezera je , znak < je < , znak > je > atd.)
- Poznámka se do zdrojového kódu vkládá mezi značky <!-- a -->

1.2.3 Struktura HTML kódu

Tab. 2 Struktura HTML kódu

Tag	Význam	Párový	Výskyt
html	začátek HTML dokumentu	ano	na začátku souboru
head	hlavička stránky	ano	na začátku souboru
body	tělo stránky + definice pozadí	ano	za <head>
<!-- -->	poznámka	ano	kdekoliv

Klasický HTML soubor může vypadat takto:

```
<html>
  <head>
    obsah hlavičky
  </head>
  <body>
    obsah těla
  </body>
</html>
```

1.2.4 Úprava textu

Tab. 3 Fyzické a logické formátování textu

Fyzické formátování		
Tag	Význam	Párový
b	tučné písmo	ano
i	kurzíva	ano
u	podtržení textu	ano
sub	dolní index	ano
sup	horní index	ano
small	zmenšení textu	ano
big	zvětšení písma	ano
s	přeškrtnutý text	ano
strike	přeškrtnutý text	ano
font	písmo	ano
basefont	základní font	ne
blink	blikání písma	ano
nobr	nezalamovat obsah	ano

Logické formátování		
Tag	Význam	Párový
strong	zvýraznění (tučně)	ano
em	zvýraznění (kurzíva)	ano
cite	citace	ano
code	výpis kódu	ano
dfn	nově použitý termín	ano
kbd	vstup z klávesnice	ano
samp	ukázka	ano
tt	teletype	ano
var	formátování proměnné	ano
abbr	ustálený výraz	ano
acronym	zkratka	ano
del	smazaný obsah	ano
ins	přidaný text	ano
q	citace	ano
span	úsek textu	ano

Tab. 4 Bloky a seznamy

Bloky			Seznamy		
Tag	Význam	Párový	Tag	Význam	Párový
p	odstavec	nepovinně	li	položka seznamu	nepovinně
br	řádkový zlom	ne	ol	číslovaný seznam	ano
div	oddíl	ano	ul	odrážkový seznam	ano
center	vycentrování	ano	dir	zvláštní druh seznamu	ano
h1	nadpis 1. úrovně	ano	menu	typ seznamu	ano
h2	nadpis 2 úrovně	ano	dl	seznam definic	ano
h3	nadpis 3. úrovně	ano	dt	definovaný termín	ano
h4	nadpis 4. úrovně	ano	dd	definice termínu	ano
h5	nadpis 5. úrovně	ano			
h6	nadpis 6. úrovně	ano			
blockquote	citace, odsazení	ano			
address	adresa	ano			
pre	předformátovaný text	ano			
hr	vodorovná čára	ne			

1.2.5 Obrázky

Obrázky (angl. images) - v HTML se obrázky vkládají nepárovým tagem IMG, který má jeden povinný a mnoho nepovinných parametrů. Tím povinným je SRC, totiž umístění obrázku. Přípustné jsou soubory typu GIF, JPEG, v novějších prohlížečích také PNG a v Internet Exploreru BMP (jiné typy souborů se musejí vkládat jako objekty). Pokud soubor obrázku nebude nalezen, zobrazí se ve stránce zástupný symbol nenalezeného obrázku (nejčastěji červený křížek) se zástupným textem.

Tab. 5 Vkládání obrázků

Tag	Význam	Párový
img	obrázek	ne
map	klikací mapa	ano
area	oblast v klikací mapě	ne

Tab. 6 Atributy tagu

Atribut	Význam	Hodnoty
src	umístění souboru s obrázkem	URL
alt	alternativní popis	libovolný text
lowsrc	náhradní obrázek pro malé displeje	URL
width	šířka	délka nebo procento
height	výška	délka nebo procento
border	tloušťka rámečku	délka
vspace	vertikální okraj	délka
hspace	horizontální okraj	délka
align	zarovnání obrázku	left, right, top, texttop, middle, absmiddle, baseline, bottom, absbottom
usemap	použití klikací mapy	"#" + jméno mapy, nebo URL
ismap	přidání souřadnic kliknutí	bez hodnoty

Příklad:

```
<IMG SRC="images/rov/rov2.1.gif" ALIGN="center" ALT="Rovnice" BORDER="1">
```

Zobrazí vycentrovaný a orámovaný obrázek s popiskem „Rovnice“ z adresáře „images/rov“ s názvem „rov2.1.gif“

1.2.6 Odkazy

Odkazy (angl. anchors) - slouží k navigaci a k hlubšímu prozkoumávání a surfování po stránkách. Pomocí odkazů můžeme zachytit vztahy mezi částmi textu, jinými dokumenty a externími stránkami. Stránky se tak stávají přehlednými a obsahově strukturovanými. Špatné odkazování může návštěvníka stránek velmi odradit, proto musejí být odkazy vytvářeny jednoduše a logicky.

Tab. 7 Atributy tagu <A>

Atribut	Význam	Hodnoty
href	cíl odkazu	URL
name	jméno záložky	libovolné jméno
target	cílový rám	jméno rámu, virtuální rám (_blank, _top, _parent, _self)

URL adresy:

Všimněme si, že atribut HREF nabývá hodnoty URL (Uniform Resource Locator), což je jednoznačná adresa, která je používána pro identifikaci různých zdrojů na Internetu. Obecný zápis URL je následující:

Schéma//uživatel:heslo@počítač:port/cesta;parametry?dotaz#fragment.

Samozřejmě, že URL nemusí obsahovat všechny části. Většinou postačí pouze schéma a určení doménové adresy počítače a cesta k souborům, které se mají zobrazit („dotaz“ slouží k předávání hodnot získaných z formulářů a „fragment“ se použije, pokud se chceme odkazovat pouze na určitou část stránky - viz. jméno záložky v atributu NAME tagu <A>). Schéma indikuje typ zdroje nejčastěji protokolem, který je nutné použít pro dosažení daného zdroje. Nejčastěji používaným schématem je právě HTTP, které indikuje webovou stránku (či přesněji objekt, který je dosažitelný pomocí protokolu HTTP).

Tab. 8 Nejčastější schémata pro URL

file	soubor na lokálním disku
ftp	File Transfer Protocol (přenos souborů)
gopher	Gopher protocol
http	HyperText Transfer Protocol (World-Wide Web)
mailto	adresa pro elektronickou poštu
news	diskusní skupiny
nntp	diskusní skupina na určitém serveru
telnet	terminálový přístup ke vzdálenému počítači

V praxi rozlišujeme dva typy URL adres:

1. Absolutní - je minimálně dáno identifikací schématu a doménové adresy. Např. <http://www.domena.cz>. Tento způsob adresace se nejčastěji používá pro odkazování na externí dokumenty a stránky či pro změnu serveru.
2. Relativní - používá pouze specifikaci cesty. Schéma a doménovou adresu serveru doplní prohlížeč podle dokumentu, který odkaz obsahuje. Např. se nacházíme na stránce stáhnuté z URL <http://www.domena.cz> a stránka obsahuje následující odkaz:

```
<A href="stranka.html">ODKAZ</A>
```

Prohlížeč tedy automaticky (a interně) doplní celý URL na:

```
<A href="http://www.domena.cz/stranka.html">ODKAZ</A>
```

Pokud se ve zdrojovém kódu nachází odkaz se dvěma tečkami „./“:

```
<A href="../../obrazky/1.jpg">ODKAZ</A>
```

pak jde o odkaz, který znamená výstup z aktuálního adresáře, vstup do adresáře „obrazky“ a odkázání na soubor „1.jpg“. Pomocí dvou teček se totiž můžeme v úrovni adresářů, ve kterém jsou všechny zdroje ke stránce uloženy, dostat na vyšší úroveň.

Relativní adresace je velmi výhodná v případě, že pokud stránky přemístíme na jiný server, nemusíme znovu předadresovat všechny odkazy ve všech stránkách, aby vše fungovalo tak, jak má.

Použití relativní adresaci pro odkaz na externí dokumenty či změnu serveru je nemožné.

Záložky:

Pokud máme příliš dlouhý dokument a chceme se odkazovat na určité části, pak je dobré použít tzv. záložky (resp. kotvy nebo fragmenty). Docílíme toho nejprve definicí takovéto záložky pomocí atributu NAME a následně odkazem na tuto záložku pomocí atributu HREF doplněné o znak # a jméno záložky.

Příklad:

V souboru *obsah.html* mějme uprostřed zdrojového kódu a textu tuto řádku:

```
<A name="Obrazek56"><IMG src="images/obr56.jpg"></A>
```

V souboru *main.html* mějme uprostřed zdrojového kódu a textu tuto řádku:

```
<A href="obsah.html#Obrazek56" target="_blank">Ukaž obrázek</A>
```

Po kliknutí na odkaz „Ukaž obrázek“ na stránce *main.html* nám prohlížeč otevře nové okno (díky parametru „_blank“ atributu „target“), zobrazí do něj stránku *obsah.html* a zároveň také skočí na řádek, na kterém se nachází záložka „Obrazek56“ (tzn. na řádek, na kterém je vykreslen obrázek „obr56.gif“ z adresáře „images“).

1.2.7 Rámy

Rámy (angl. frames) - rozdělují stránku do dvou nebo více obdélníkových oblastí (rámů), do kterých se načítají jiné stránky. Tag <FRAMESET> se vyskytuje za hlavičkou místo tagu <BODY>. Může obsahovat pouze tagy <FRAME> a další <FRAMESET>.

Tab. 9 Rámy

Tag	Význam	Párový
frameset	skupina ráků	ano
frame	rák	ne
noframes	alternativa ráků	ano
iframe	vložený rák	ano

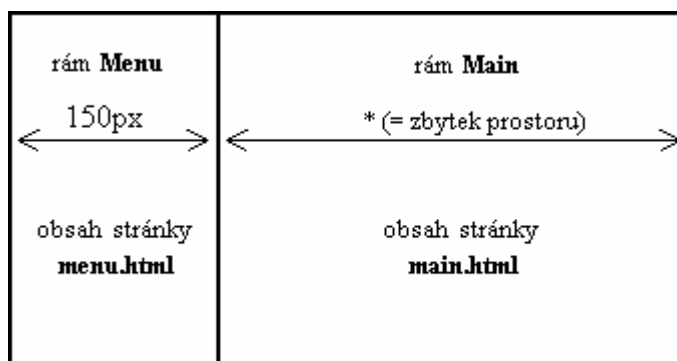
Příklad:

```

<html>
<head>
<title>Titulek stránky</title>
</head>
<frameset border="false" frameborder="0" cols="150,*">
    <frame name="Menu" scrolling="no" target="rtop" src="menu.html">
    <frame name="Hlavni" scrolling="auto" src="main.html">
</frameset>
</html>

```

Tento zdrojový kód zobrazí dvě oblasti s textem z různých zdrojů (webové stránky: *menu.html* a *main.html*) jak to ukazuje Obr. 1. První oblast bude široká 150 pixelů a druhá bude zabírat prostor až na hranici internetového prohlížeče. Zbytek prostoru se v kódu určuje znakem *.



Obr. 1. Stránka vytvořená pomocí ráků

Výhody ráků:

- tvorba trvale viditelných oblastí stránek (např. rám *Menu*)
- jednoduchost a rychlost při návrhu
- snadná údržba stránek.

Nevýhody rámců:

- nekompatibilní se staršími internetovými prohlížeči
- nepřehledné rozmístění souborů při uložení stránky na disk
- problémy s vyhledávači (vyhledávací služba většinou návštěvníka pošle na stránku bez kontextu ostatních rámců a tedy např. bez rámu *Menu*, tzn. zobrazí se pouze rám *Main* a návštěvník tedy nemá k dispozici žádnou navigaci).

1.2.8 Tabulky

Na internetových stránkách se tabulky používají ve dvou případech:

1. Jestliže je potřeba udělat klasickou tabulku.
2. Jako pomůcka pro úpravu stránek, když je něco potřeba někam umístit.

Tab. 10 Tabulky

Tag	Význam	Párový
table	tabulka	ano
tr	řádek tabulky	nepovinně
td	buňka tabulky	nepovinně
th	hlavičková buňka tabulky	nepovinně
caption	hlavička tabulky	ano
col	ovlivnění sloupce tabulky	ne
colgroup	skupina sloupců tabulky	ne
tbody	tělo tabulky	ano
thead	hlavička tabulky	ano
tfoot	patička tabulky	ano

Velmi často je potřeba pracovat s formátem buňky určité tabulky (např. nastavit jí určitou velikost, barvu, zarovnání apod.). Tab.11 ukazuje nejdůležitější atributy tagu <TD> pro definici buňky, jejich význam a hodnoty, kterých můžou tyto atributy nabývat.

Tab. 11 Atributy tagu <TD>

Atribut	Význam	Hodnoty
align	horizontální zarovnání obsahu	left, center, right, justify
valign	vertikální zarovnání obsahu	top, middle, bottom, baseline
width	doporučená šířka buňky	délka nebo procento
height	minimální výška řádku	délka nebo procento
nowrap	obsah buňky se nebude zalamovat	bez hodnoty
background	obrázek na pozadí	URL obrázku
bgcolor	barva pozadí	barva
bordercolor	barva rámečku	barva
rowspan	přesah buňky na n dalších řádků (spojení buněk)	počet přesahujících řádků
colspan	přesah buňky do dalších sloupců	počet přesahujících sloupců

Velmi často používanými atributy jsou „*colspan*“ a „*rowspan*“, které umožňují spojit dvě nebo více sousedních buněk do jedné.

Příklad:

```
<table border="1">
<caption>Hlavička</caption>
  <tr><td>A</td> <td>B</td> <td rowspan="2">C</td> </tr>
  <tr> <td colspan="2">D</td> </tr>
  <tr> <td colspan="3">E</td> </tr>
</table>
```

Tento zdrojový kód vykreslí na webovou stránku tabulku, která je zobrazena na Obr. 2.

Hlavička		
A	B	C
D		
E		

Obr. 2. Tabulka se spojenými buňkami

Atribut *Rowspan="2"* znamená, že buňka C zasahuje přes dva řádky (row = řádek), *colspan="3"* říká, že buňka E zabírá prostor přes tři sloupce (col = sloupec).

1.2.9 HTML editory a postupné psaní tagů

Dnes existuje velká spousta HTML editorů (Macromedia Dreamweaver, Microsoft FrontPage atd.), které dokáží vytvořit uživatelsky velmi příjemné prostředí webových stránek. Jsou ale především používány lidmi, kteří HTML jazyk umí jen zčásti nebo vůbec. Designéři a programátoři podobným HTML editorům ale příliš nevěří a velká většina z nich raději napíše kód sami. Mají k tomu spoustu důvodů, protože „ručně“ napsaný kód je:

- minimální
- přehledný

Příklad:

Stránka vytvořená pomocí postupného psaní tagů, která zobrazuje nadpis, text na stránku formátu A4 a dva odkazy zabírá 3246 bytes diskového prostoru. Naproti tomu stejná stránka vytvořená pomocí běžně dostupného textového editoru Microsoft Word zabírá 25664 bytes! Důvod je jasný - textový editor přidal do zdrojového kódu příliš mnoho zbytečných informací a tagů, které se ovšem při stahování stránky ze serveru berou také v potaz (i když nebývají většinou vůbec důležité). Diference těchto dvou velikostí rozdílných zdrojových kódů je 22418 bytes, tzn. že toto kvantum dat je pro stejný efekt stránky téměř zbytečné, ovšem u stránky z Wordu se bude muset stejně stáhnout, aby byla stránka kompletní. Vidíme tedy, že stránka vytvořená v Microsoft Word se načítá téměř 8-krát déle než stránka napsaná pomocí postupného psaní tagů! Toto je ale samozřejmě extrémní případ a většina HTML editorů se dnes již snaží velikosti stránek co nejvíce minimalizovat. Přesto je ale stále vzhledem k velikosti (a tedy i rychlosti stahování) stránek a přehlednosti lepší napsat HTML kód ručně.

1.3 CSS Styly

CSS (Cascading Style Sheet) vzniklo kolem roku 1997. Jedná se o kolekci metod určenou pro grafickou úpravu webových stránek. Styly umožňují nadefinovat způsoby formátování pro jeden typ elementu nebo pro určitou skupinu různých elementů pouze jedním zápisem - tzn. nemusíme způsob formátování opakovaně zapisovat u každého elementu.

Definování vzhledu jednoho typu elementu je velmi jednoduché. Styl můžeme zapsat obecně takto:

```
element { vlastnosti; }
```

Pokud ale chceme mít pro různé elementy stejný vzhled, musíme definovat tzv. „class“ (třidu, do které element patří). A v HTML kódu následně tuto třídu elementu přidělit. Symbolicky:

```
<tag class="jmeno_tridy">
```

Obecně se třída stylu zapíše takto:

```
.trida { vlastnosti; }
```

Všimněme si tečky před názvem třídy (.trida), která říká, že se jedná o třídu a ne o element.

Nejčastěji se styly nadeklarují pomocí externího stylopisu, tzn. uložením definic elementů a jejich zobrazení do externího souboru a načtením tohoto souboru v hlavičce HTML dokumentu.

Příklad:

Hlavní HTML dokument bude obsahovat něco takového:

```
<html>
<head>
    <title>Pokusná stránka se stylem</title>
    <link href="styl.css" type="text/css" rel="StyleSheet">
</head>
<body>
    <h1>Hlavní nadpis</h1>
    <p class="odstavce">Text v odstavci</p>
</body>
</html>
```

V souboru „styl.css“ budou uloženy např. tyto deklarace:

```
body {background-color:black;}  
h1 {color:blue; font-size:20; text-align:center;}  
.odstavec {color:black;font-size:12;font-family:Arial;text-align:left;;}
```

Po zobrazení stránky v prohlížeči se nám zobrazí stránka s černým pozadím, modrým vycentrovaným nápisem o velikosti 20 a „Text v odstavci“, který bude mít černou barvu, zarovnán doleva, mít velikost 12 a napsán fontem Arial.

Co CSS dovedou:

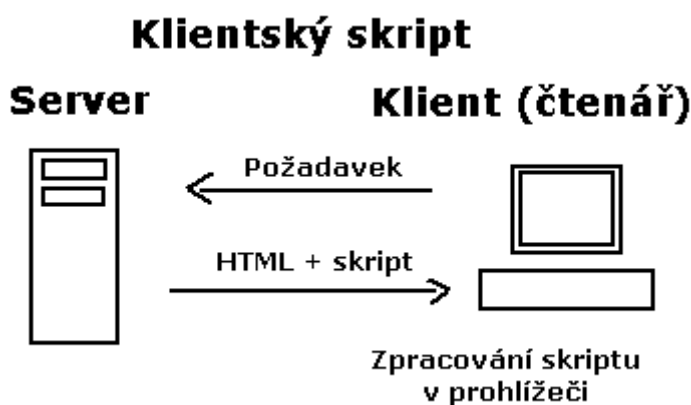
- Nastavit libovolnou a přesnou velikost písma, prokládání, kapitálky
- Udělat odsazení prvního řádku odstavce, zvětšit řádkování
- Zrušit nebo zvětšit prázdný prostor po odstavci
- Automaticky formátovat nadpisy
- Zvýrazňovat odkazy po přejetí myši
- Udělat automaticky grafické odrážky
- Určité části textu zneviditelnit, zprůhlednit nebo nezobrazit
- Předefinovat grafický význam běžných tagů
- Nastavit pozadí čehokoliv, stránky, tabulky ale třeba i odstavce
- Umístit nějaký objekt kamkoliv do stránky, může se to i překrývat s jinými objekty
- Přidat k čemukoli rolovací lišty, oříznout, orámovat, nastavit okraje

Kaskádové styly mají ovšem jednu velkou nevýhodu. Každý prohlížeč zobrazuje CSS styly trochu jinak. Často je tedy potřeba pro každý prohlížeč napsat jiný styl. Pravděpodobně nejlepším prohlížečem, který CSS podporuje přesně podle specifikace je Mozilla. V jiných prohlížečích se některé vlastnosti stylů mohou chovat podivně a je proto dobré vytvořené stránky pomocí CSS otestovat i na těchto prohlížečích, aby se např. nestalo, že některé texty se budou překrývat a budou nečitelné.

Pomineme-li tuto nevýhodu, pak jsou kaskádové styly skutečně mocným nástrojem pro tvorbu graficky příjemných a přehledných stránek.

1.4 JavaScript

JavaScript je programovací jazyk, který se používá v internetových stránkách. Je důležité říci, že se jedná o klientský skript. To znamená, že se program odesílá se stránkou na klienta (do prohlížeče) a teprve tam je vykonáván (protikladem klientských skriptů jsou skripty serverové, které jsou vykonávány na serveru a na klienta se posílají už jen výsledky).



Obr. 3. Klientský skript

Charakteristiky jazyka JavaScript:

- interpretovaný - nemusí se kompilovat
- objektový - využívá objektů prohlížeče a zabudovaných objektů
- závislý na prohlížeči - funguje ale ve většině prohlížečů
- case-sensitivní - záleží na velikosti písem v zápisu syntaxí podobný jazykům C, Java a podobným.

Omezení JavaScriptu:

- JavaScript funguje pouze v prohlížeči.
- Uživatel může JavaScript zakázat
- Existují různé odlišné verze jazyka i prohlížečů, což vede k častým chybám.
- Neumí přistupovat k souborům (kromě cookies) ani k žádným systémovým objektům.
- Neumí žádná data uložit (kromě cookies).

To vše z něj dělá pouze jazyk druhořadý, účelově použitelný pouze v HTML stránkách.

Začlenění skriptu do stránky:

1. Pomocí tagu <SCRIPT> do proudu dokumentu a vložení těla skriptu mezi úvodní a koncový tag.

Příklad:

```
<script>
    alert('Nastala výjimečná situace!');
</script>
```

2. In-line (řádkový) zápis jako atribut tagu bez použití tagu <SCRIPT>

Příklad:

```
<a href=http://www.odkaz.cz
onmouseover="document.write('Najel jste myši na
odkaz');">Odkaz</a>
```

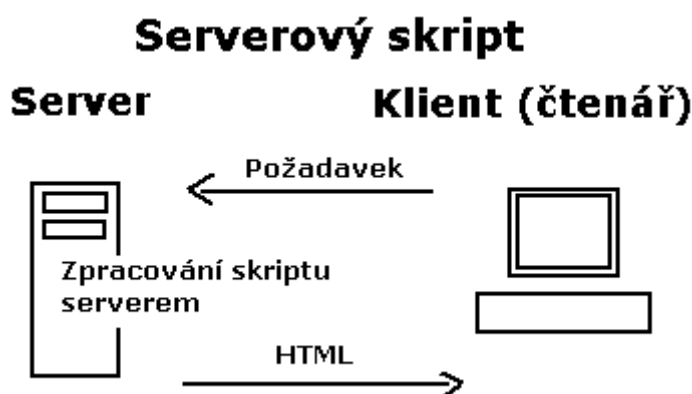
3. Tagem <SCRIPT> s odkazem na externí soubor, ve kterém je JavaScript uložen a poté ho z tohoto souboru do stránky načítat. Tento způsob je nejčastější už jen z důvodu přehlednosti zdrojových kódů stránek a znouvopoužitelností skriptu na jiných stránkách.

Příklad:

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

1.5 PHP

PHP řadíme do skupiny skriptovacích jazyků, které se provádějí na straně serveru. PHP je na serveru závislé, protože na něm běží jeho interpreter, který skripty provádí. PHP se tímto odlišuje např. od JavaScriptu, jehož skripty se stahují přímo s HTML stránkou a jsou vykonány na straně klienta jeho prohlížečem. Má to své výhody i nevýhody. Výhodou PHP v tomto případě je, že se ke zdrojovým kódům skriptů nedostane nikdo jiný než autor, kdežto ke zdrojovému kódu JavaScriptu se dostane každý, kdo si stáhne HTML stránku, v níž je skript obsažen. JavaScript má výhody ve své možnosti dynamicky reagovat na událost způsobenou klientem (např. pohyb kurzoru myši), což PHP nedokáže, protože k provedení každé své nové události musí být vždy prohlížečem znovu odeslán požadavek na server. Proto je nejvhodnější variantou kombinovat PHP s JavaScriptem nebo jiným, dynamicky reagujícím jazykem.



Obr. 4. Serverový skript

Samotné PHP skripty se zapisují přímo do HTML stránky (nejčastěji s příponou *.php). PHP interpreter na serveru pak pracuje tak, že HTML příkazy rovnou ukládá do výsledné HTML stránky, ale narazí-li na PHP skript, nejprve ho provede, a potom je do HTML stránky zapsán jeho výsledek.

Symbolicky zapsáno:

HTML zdroj, `<? PHP vsuvka ?>` , opět HTML zdroj

Do cílového prohlížeče (klienta) po zpracování PHP skriptu serverem dorazí:

HTML zdroj, výsledek vsuvky , opět HTML zdroj

PHP je svou syntaxí velmi blízce spjat s jazykem C, Javou nebo Perlem. Používá velmi podobný syntaktický zápis větvení skriptů, cyklů, tvorby funkcí a práci

s proměnnými.

Co je k PHP potřeba:

Pro vyhodnocování PHP skriptů je potřeba mít na serveru (resp. na stanici, na které vyvíjíme dynamické stránky) nainstalován HTTP server s PHP modulem (např. Apache nebo IIS), který zajistí interpretaci PHP skriptu.

Možnosti PHP:

- získávání dat z formulářů
- široká práce a schopnost operovat s daty
- čtení a ukládání cookies
- dynamická generace stránek
- spolupráce s databázemi (podporuje následující databázové systémy: MySQL, MS SQL server, Oracle, Sybase, Adabas D, PostgreSQL, mSQL, Velocis, Solid, Informix).
- obsahuje knihovny pro práci s některými internetovými protokoly: HTTP, FTP, POP3, SMTP, LDAP, SNMP, NNTP atd.

PHP je silný nástroj pro tvorbu dynamických stránek, ovšem je potřeba již mít potřebné znalosti z oboru programování, algoritmizace a návrhu struktury programů, aby se PHP dalo vhodně využít. Ve spojení s databázovým systémem vytváří skutečně profesionální web-aplikace, které posouvají obyčejné webové stránky o velký kus cesty dopředu.

2 PRAKTICKÁ ČÁST

2.1 Tvorba HTML stránek

2.1.1 Návrh vzhledu internetových stránek

Pro tvorbu webových stránek jsem se rozhodl celý zdrojový kód napsat ručně, tedy bez pomoci nějakého HTML editoru. Zdrojový kód minimální co do velikosti a hlavně také velmi přehledný, takže v případě dalšího zásahu do stránek ve snaze aktualizovat jejich obsah, zvládne jakoukoliv další úpravu kterýkoli člověk se základní znalostí HTML. Orientace ve zdrojovém kódu je pro další designéry mnohem snadnější a přehlednější než v jiných formách návrhu webových stránek.

V průběhu tvorby webových stránek se jejich vzhled mnohokrát měnil v závislosti na nově přidávaných okruzích a tématech. Největším problémem bylo udělat velmi přehledné menu a toto menu také správně umístit tak, aby dlouhé názvy kapitol a témat byly plně napsány a aby kvůli těmto dlouhým názvům nepřekročilo menu rozsah stránky. Stránky jsou totiž optimalizovány pro rozlišení 1024x768px a tudíž, pokud bychom při návrhu přistupovali klasicky, tzn. rozložili stránku podobně jako ukazuje Obr. 5, pak bychom museli minimalizovat oblast, ve které se vypisuje samotný výukový text, což by pro návštěvníka webových stránek znamenalo obtížnější orientaci v textu.

Hlavička stránky	
MENU	text, který se zobrazuje

Obr. 5. Klasické rozložení webové stránky

Zvolil jsem tedy jiný způsob rozložení různých částí stránky, který je dostatečně přehledný a vyhovující omezením, které musíme brát při návrhu v potaz. Výsledné rozložení stránky ukazuje Obr. 6.

Obrázek	MENU
	Nadpis
MENU dodatků	text, který se zobrazuje

Obr. 6. Základní koncept vzhledu stránky

Pro vytvoření tohoto konceptu webové stránky se nabízí 2 možnosti přístupu. Buď použít tzv. rámy (frames), které rozdělují stránku do různých oblastí a každá oblast znamená jednu HTML stránku zdrojového kódu, nebo si stránku představit jako tabulku se 3 řádky a 2 sloupci, jak ukazuje Obr. 6. Protože použití rámců s sebou nese velkou spoustu nevýhod (již výše zmíněných v teoretické části), přistoupilo se raději na druhou a zaběhnutější možnost, kterou používá spousta velkých webů a internetových portálů (např. <http://www.seznam.cz>, <http://www.google.com> apod.), tj. chápání stránky jako jednu velkou tabulku.

Zjednodušeně vypadá tato webová stránka z hlediska zdrojového kódu následovně:

```
<html>
<head>
  <title>Teorie automatického řízení</title>
</head>
<body>
  <table align="left">
    <tr>
      <td rowspan="2">
        Obrázek
      </td>
      <td valign="top">
        <center>
```

```

                Menu
            </center>
        </td>
    </tr>
    <tr>
        <td>
            <center>
                Nadpis
            </center>
        </td>
    </tr>
    <tr>
        <td valign="top">
            <center>
                MENU dodatků
            </center>
        </td>
        <td>
            text, který se zobrazuje
        </td>
    </tr>
</table>
</body>
</html>

```

Při použití tabulek je ovšem jasné, že se tabulka musí opět v každé podstránce znovu vytvořit, resp. znovu do této podstránky zapsat výše zmíněný zdrojový kód. Naskýtá se ale jedna možnost, jak takovéto omezení obejít. Pokud stránky budou vystaveny na serveru, který bude podporovat PHP, můžeme do centrální buňky (určené pro hlavní text), vždy vkládat tyto podstránky, aniž by musely nutně znovu obsahovat strukturu tabulky. O vše se prostě bude starat server. Princip je jednoduchý: odkazům na stránkách vždy přidělíme určitý dotaz, na který se server podívá a vyhodnotí jej.

Dejme tomu, že menu obsahuje následující odkaz:

```
<A href="?m=2.4">Stabilita a její kritéria</A>
```

V hlavním souboru zdrojového kódu (tedy „index.php“) se v buňce pro zobrazování čteného textu vepíše následující kód:

```

<td>
    <?   if ($m="2.4")   include "themes/2.4.php";   ?>
</td>

```

Vše funguje tak, že po kliknutí na odkaz pošle klient serveru proměnnou m naplněnou hodnotou 2.4. Server se podívá, jakou má proměnná hodnotu, a pokud má tuto hodnotu (podmínka *if*, kde znak \$ značí proměnnou), pak se do buňky vloží (příkaz *include*) obsah souboru *2.4.php*, který je uložen v adresáři *themes*.

Je dobré zmínit, že zápis:

```
<A href="?m=2.4">
```

je ekvivalentní zápisu:

```
<A href="index.php?m=2.4">
```

tzn. že klient zašle dotaz ze stránky „index.php“ a server ví, na kterém místě má zaslaný dotaz vyhodnocovat.

V konečné fázi tvorby webových stránek je do hlavní buňky tabulky vložena jedna velká PHP vsuvka obsahující podmínky *if* a *elseif*, které rozhodují, která podstránka se má v této buňce zobrazit.

2.1.2 Tvorba základních elementů webové stránky

Bylo nutné si připravit určitý model, pomocí kterého by se formátovaly určité zvláštní součásti stránek, jako jsou rovnice a obrázky, a poté již tento stávající model jen dále ve stránkách využívat.

2.1.2.1 Rovnice

HTML bohužel neumí zobrazovat speciální formy textu, jako je např. tato rovnice se zlomky a maticovými závorkami:

$$G(s) = (1 \ 0) \begin{pmatrix} s & -1 \\ 3 & s+4 \end{pmatrix}^{-1} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \frac{1}{s^2 + 4s + 3} (1 \ 0) \begin{pmatrix} s+4 & 1 \\ -3 & s \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \frac{1}{\underline{\underline{s^2 + 4s + 3}}} \quad (6.20)$$

Bylo tedy nutné podobné rovnice zobrazovat graficky pomocí GIF formátu, aby byla velikost tohoto „obrázku“ co nejmenší a aby také byla zajištěna transparentnost bílého pozadí rovnice (průhlednost), což nejpoužívanější formát na webových stránkách - JPEG nedokáže.

Celá tato rovnice lze po uložení do grafického souboru na webové stránce zobrazit takto:

```
<A NAME="Rov.6.20"></A>
<TABLE CLASS="sirka"><TR>
<TD>&nbsp;</TD>
<TD CLASS="delka"><IMG SRC="images/rov/rov6.20.gif"></TD>
<TD ALIGN="CENTER">(6.20)</TD>
</TR></TABLE>
```

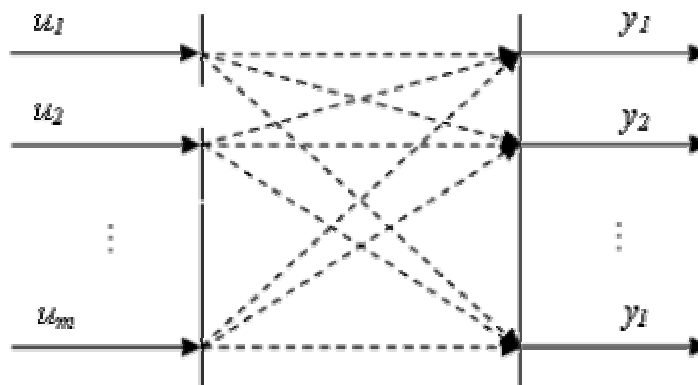
Jde tedy o vytvoření menší tabulky se 3 buňkami - první buňka je prázdná a pouze tabeluje zobrazenou rovnici v druhé buňce (pomocí zobrazení obrázku) o pár pixelů doprava. Třetí buňka obsahuje číslo rovnice. Tabulce je přiřazena třída „sirka“, která je v CSS souboru deklarována jako:

```
.sirka { width:700px; text-align:center; }
```

Znamená to tedy, že tabulka je zarovnána na střed a je široká 700 pixelů. Celé tabulce předchází identifikace kotvy pomocí tagu <A NAME...> pro pozdější lepší orientaci a hledání rovnic na stránkách.

2.1.2.2 Obrázky

Obrázky velmi zpestřují celkový vzhled stránek a dopomáhají k lepší představě o daném tématu. Je tedy nutné, aby měly také nějakou formu a dalo se na ně kdykoliv v průběhu jiných témat odkazovat. Nejlepší formou, jak obrázek s popiskem zobrazovat, je pravděpodobně již zaběhnutý způsob, kdy obrázek je zarovnán na střed a pod obrázkem se nachází jeho popisek. Klasický způsob nám ukazuje následující obrázek.



Obr. 5.1. Schéma mnohorozměrné soustavy

K zobrazení obrázku a jeho popisku ve výše uvedeném formátu je nutné napsat následující krátký HTML kód:

```
<CENTER>
<A NAME="Obr.5.1"></A>
<IMG SRC="images/obr/obr5.1.gif">
<BR>
Obr. 5.1. Schéma mnohorozměrné soustavy
</CENTER>
```

Jak můžeme vidět, v HTML kódu je také zanesena kotva popsána tagem <A NAME...>, pro pozdější návrat k danému obrázku.

2.2 Tvorba menu pomocí JavaScriptu

Menu je na stránkách jedna z nejdůležitějších částí, protože slouží k navigaci podstránek. Musí být tedy dostatečně jednoduché a přehledné pro uživatele. V důsledku dlouhých nadpisů a složitějšímu větvení podstránek by bylo nejlepší, kdyby bylo menu také větvené. Nejlepším způsobem, jak toto udělat, je vytvořit tzn. vysouvací menu a nejlepším nástrojem k jeho vytvoření je JavaScript. Je známo, že JavaScript se klientovi při stahování stránky ze serveru také posílá a proto většinou stačí nalézt stránku se skriptem, který potřebujeme a tento skript pak pouze použít na svých stránkách. Bohužel vzhledem k tomu, že žádné menu jiných stránek nevyhovovalo omezujícím podmínkám vytvářených stránek, bylo nutné se JavaScript naučit a vytvořit si menu podle svých vlastních kritérií.

Z důvodu znalosti velké spousty jiných programovacích a skriptovacích jazyků bylo pochopení základních principů a rysů JavaScriptu poměrně krátkodobou záležitostí. Stačilo nastudovat pouze základní interní funkce JavaScriptu, její události, elementy stránky, se kterými pracuje, a vlastnosti těchto elementů a mohlo se krok po kroku začít tvořit uživatelsky příjemné menu.

Celé menu je tvořeno 9 základními funkcemi:

```
function StartMainMenu(type);
```

- funkce vytváří zázemí pro hlavní menu, tzn. menu, které se bude dále rozvětňovat.
- parametr *type* udává, zda budou položky hlavního menu skládány vertikálně (*type=0*) nebo horizontálně (*type=1*).

```
function MainMenuItem(name, text, url, width, position, style);
```

- funkce tvoří samotnou položku hlavního menu.
- parametry:
 1. *name* = identifikátor pro pozdější poznání položky v kódu a pro vytváření podpoložek (větvení menu).
 2. *text* = zobrazení nápis na položce menu. Pro uživatele to znamená nápověda, co se pod touto položkou skrývá.
 3. *url* = URL adresa, na kterou se po kliknutí na položku půjde.
 4. *width* = udává šířku položky v pixelech.

5. *position* = pozice, na které se bude po najetí myši na položku zobrazovat podmenu (*position* = 0, podmenu se zobrazí vedle položky hlavního menu; *position* = 1, podmenu se zobrazí pod položkou hlavního menu).
6. *style* = vzhled položky hlavního menu (barva pozadí, barva písma, font písma atd.)

```
function EndMainMenu();
```

- funkce ukončuje vytváření menu. Značí, že již nebude následovat žádná další položka hlavního menu.

```
function StartSubMenu(name, style, width);
```

- funkce vytváří zázemí pro podpoložky hlavního menu.
- parametry:
 1. *name* = identifikátor položky z hlavního menu (viz. funkce *MainMenuItem* a její parametr *name*), která přiřazuje vytvořený seznam podpoložek pod určitou hlavní položku.
 2. *style* = obecný vzhled podpoložek.
 3. *width* = šířka podpoložek v pixelech.

```
function SubMenuItem(name, text, url, position, style);
```

- funkce vytváří samotnou podpoložku.
- parametry:
 1. *name* = identifikátor pro další možné větvení (větvit položky a podpoložky lze neomezeně).
 2. *text* = nápis, který se uvede na podpoložku pro uživatele.
 3. *url* = URL adresa, kam bude uživatel po kliknutí přenesen.
 4. *position* = pozice dalších podpoložek pro další možné větvení (stejně jako u funkce *MainMenuItem*).
 5. *style* = vzhled podpoložky.

```
function EndSubMenu(name);
```

- funkce ukončuje vytváření podmenu s identifikátorem *name*. Značí, že již nebude následovat žádná další podpoložka.

```
function OpenMenu(event, id, pos);
```

- funkce která rozvětvená podmenu. Podmenu jsou totiž na stránce neustále, jen je jim přidán atribut *visibility* s hodnotou *hidden*, tzn. jsou skryté (neviditelné). Funkce pouze zjistí, která položka menu je aktivní (na které uživatel najel myši - událost *onmouseover*), a podle identifikátoru pro určité podmenu změní

atribut *visibility* z *hidden* na *visible* (viditelný). Podmenu se tak zobrazí na daných souřadnicích pod nebo vedle hlavního menu.

```
function OverMenu(x,y);
```

- funkce zjistí identifikátor položky menu na souřadnicích *x* a *y*.

```
function TidyMenu();
```

- funkce slouží ke skrytí podmenu. Když uživatel odjede z aktivní položky menu (událost *onmouseout*), zjistí se pomocí funkce *OverMenu* identifikátor podmenu a funkce tomuto podmenu poté nastaví atribut *visibility* na *hidden* (skrytý).

Toto je velmi stručný a základní přehled toho, co tyto hlavní funkce provádějí a jaké mají atributy. Všechny využitelný kód je uložen v souboru *scriptmenu.js*.

Aby se dalo menu vytvořit pro naše stránky, musíme se naučit tyto funkce správně používat. Pro vytváření menu se používá prvních 6 funkcí, ostatní 3 jsou volány pouze interně a designér, který tento skript využívá, je nepotřebuje znát. Z přehledu funkcí je jisté, že pro vytvoření základního menu, potřebuje designér napsat podobný následující kus kódu:

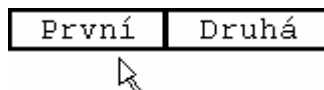
```
StartMainMenu(1);
MainMenuItem("1","První","?m=first",50,1,"mainstyle");
MainMenuItem("2","Druhá ","?m=second",50,1," mainstyle ");
EndMainMenu();

StartSubMenu("1","menustyle",50);
SubMenuItem("", "PrvníA","?m=first#A",0,"substyle");
SubMenuItem("", "PrvníB","?m=first#B",0," substyle ");
SubMenuItem("", "PrvníC","?m=first#C",0," substyle ");
EndSubMenu("1");

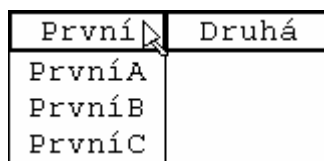
StartSubMenu("2"," menustyle ",50);
SubMenuItem("", "DruháA","?m=second#A",0," substyle ");
SubMenuItem("", "DruháB","?m=second#B",0," substyle ");
EndSubMenu("2");
```

Z kódu vyplývá, že vytváříme jedno hlavní menu se 2 položkami a 2 podmenu. Každé položce v hlavním menu je přiřazeno 1 podmenu s různým počtem položek.

Podpoložky ABC se zobrazí pouze v případě, když se na určitou hlavní položku menu (První nebo Druhá) najede myší jak ukazují Obr. 7 a Obr. 8.



Obr. 7. Náhled základního menu



Obr. 8. Náhled základního menu při nájezdu myši na položku

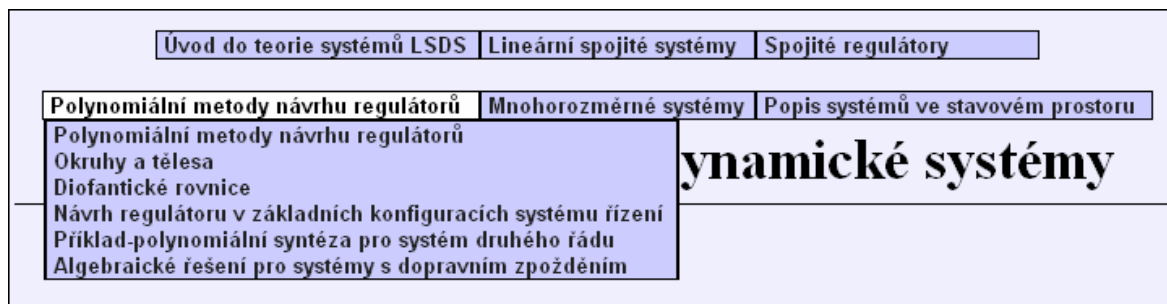
Celý výše uvedený programový kód je nejlépe umístit do externího souboru (v našem případě *menu.js*) a na hlavní stránce *index.php* brát tento externí soubor při načtení stránky v potaz pomocí speciálního tagu `<SCRIPT>` následovně:

```
<SCRIPT src="scriptmenu.js"></SCRIPT>  
<SCRIPT src="menu.js"></SCRIPT>
```

V prvním případě (soubor *scriptmenu.js*) se jedná o vložení naprogramovaných funkcí a v případě druhém (soubor *menu.js*) se již jedná o samotné vytvoření určitého menu pomocí využití naprogramovaných funkcí. Tag `<SCRIPT>` by měl na stránce být vložen tam, kde chceme, aby se vytvořené menu zobrazovalo. Protože v našem případě chápeme celou stránku jako jednu velkou tabulku, pak bude tag `<SCRIPT>` vložen do jedné z buněk této tabulky.

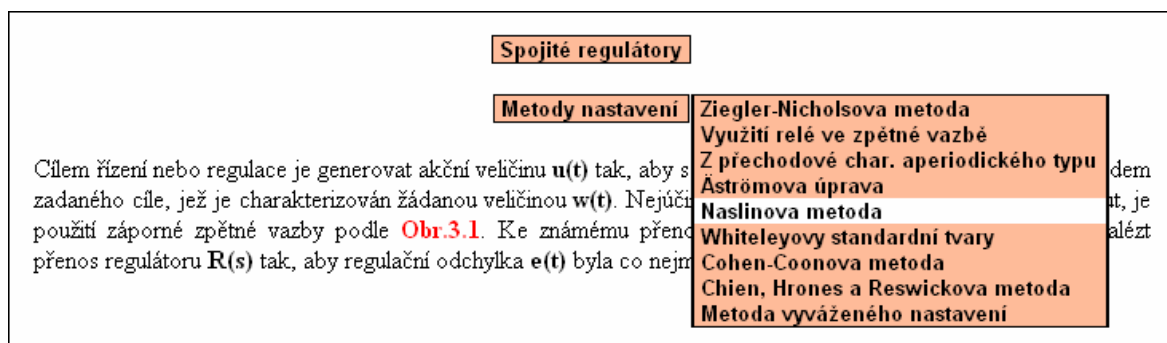
Kompletní zdrojový kód menu je pro ukázkou v příloze P 1.

Hlavní menu, které bylo pomocí skriptu vytvořeno a umístěno na webové stránky lze vidět na Obr.9. Jednoduše a srozumitelně je rozvrstveno na 6 základních stěžejních témat a jejich jednotlivých podkapitol.



Obr. 9. Konečný vzhled menu na webových stránkách

Naproti tomu Obr.10 ukazuje charakter menu, který byl použit hlavně na tvorbu odkazů potřebných k navigaci v dané podkapitole. Na daném obrázku se na webových stránkách nacházíme v kapitole **3.1. Spojité regulátory a metody jejich nastavení** a pomocí tohoto menu se můžeme odkázat na podstránky obsahující vysvětlení námi vybrané metody.



Obr. 10. Menu v podkapitolách

2.3 PHP skript pro výpočet spojitého regulátoru

2.3.1 PHP Triad

K tomu, aby vývojáři dynamických webových stránek mohli psát PHP skripty, je nutné mít na vývojové stanici nainstalován HTTP server Apache nebo IIS s PHP modulem. Nejznámějšími balíčky těchto nutností k vývoji jsou PHP Triad a PHP Home, z českých konkurentů je to Intranetový server. Obsahují právě tři nezbytné součásti pro vývoj:

1. server Apache
2. modul PHP pro Apache
3. databázový server MySQL

V mém případě pracuji s PHP Triad 2.2.1, protože je jednoduchý a velmi zaběhnutý ve světě vývojářů. A také jeho hlavní výhodou je, že je freeware (tzn. zcela zdarma). Další výhodou je již po instalaci vzájemná a bezproblémová spolupráce těchto tří součástí, takže není nutné nic předělávat a komplikovaně konfigurovat (i když samozřejmě možnost konfigurace zde stále je k dispozici).

2.3.2 Programování skriptu

Součástí webových stránek měla být také implementace vybraných algoritmů návrhu spojitých regulátorů. Bylo tedy nutné vytvořit menší uživatelské prostředí, do kterého by uživatel zadal vstupní hodnoty pro výpočet a jaký typ regulátoru chce spočítat. Zadané údaje by se zaslaly na server, kde by se zpracovaly a zpět se poslaly již vypočtené hodnoty a složky regulátoru.

Pro podobné situace se používají tzv. formuláře, které sbalují určitou skupinu dat a zasílají je na server. Uvozují se tagem <FORM> a ve svém těle obsahují všechny textové oblasti, list-boxy a tlačítka, která k danému formuláři patří. V kódu je tedy napsáno např. tento kód:

```
echo ``  
<TABLE ALIGN='center'>  
<FORM ACTION='?m=3.3' METHOD='post'>  
  <TR>  
    <TD ALIGN='center'>Regulátor typu:
```

```

        <SELECT NAME='typeofregulator' CLASS='formselect'>
            <OPTION value='PID'> PID
            <OPTION value='PD'> PD
            <OPTION value='PI'> PI
            <OPTION value='P'> P

        </SELECT>
    </TD>
    <TD ALIGN='center'>metodou:
        <SELECT NAME='typeofmethod' CLASS='formselect'>
            <OPTION value='1'> Ziegler-Nicholsova metoda
            <OPTION value='2'> Využití relé ve zpětné vazbě
            <OPTION value='3'> Nastavení z přechodové char.
aperiodického typu
            <OPTION value='4'> Äströмова úprava
            <OPTION value='7'> Cohen-Coonova metoda
            <OPTION value='8'> Chien, Hrones a Reswickova metoda
            <OPTION value='9'> Vyvážené nastavení
        </SELECT>
    </TD>
</TR>

<TR><TD ALIGN='center' COLSPAN='2'><BR>
    <INPUT TYPE='submit' VALUE='Pokračovat'>
</TD></TR>
</FORM>
</TABLE>";

```

V kódu vytváříme tabulku, která obsahuje formulář. Tabulka je použita pouze pro přehledné rozmístění prvků formuláře na stránce. Samotný důležitý kód začíná tagem

```
<FORM ACTION='?m=3.3' METHOD='post'>
```

který nám tvoří začátek formuláře a říká, že po odeslání dat (*METHOD='post'*) se zobrazí opět stránka, která obsahuje kapitolu 3.3, ovšem již se zpracovanými daty, které byly odeslány.

Další důležitou součástí formuláře je:

```

<SELECT NAME='typeofregulator' CLASS='formselect'>
    <OPTION value='P'> P
    <OPTION value='PI'> PI
    <OPTION value='PD'> PD
    <OPTION value='PID'> PID

```

```
</SELECT>
```

který vytváří tzv. list-box obsahující položky P, PI, PD, PID. Tag `<SELECT>` tento list-box uvozuje a tag `<OPTION>` značí položku (neboli volbu) v list-boxu. Atribut `NAME` tagu `<SELECT>` udává název proměnné, do které se hodnota zvolené položky uloží.

Na též principu funguje i podobný list-box pro výběr typu metody při výpočtu regulátoru.

Poslední a velmi důležitou součástí formuláře je tag:

```
<INPUT TYPE='submit' VALUE='Pokračovat'>
```

Ten zobrazuje na stránce tlačítko, které je popsáno textem uloženým v atributu `VALUE` a slouží k potvrzení dat a vykonání akce formuláře. Tag `<INPUT>` je velmi různorodý element webových stránek. Jeho charakter je totiž udáván atributem `TYPE`.

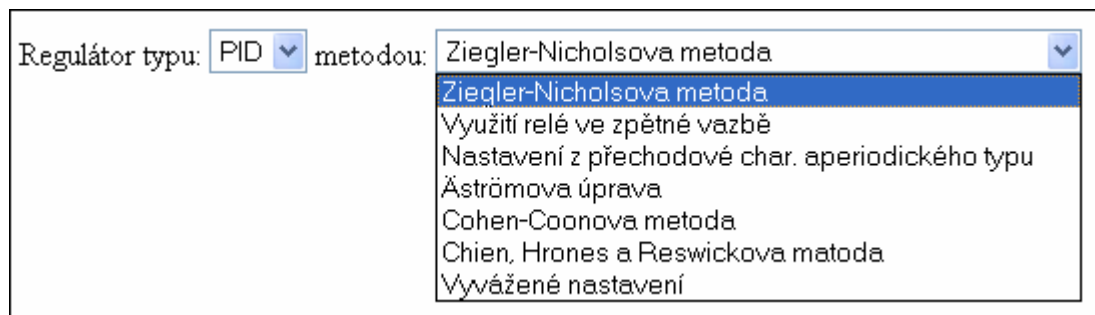
Možnosti atributu TYPE:

1. `submit` - vytváří tlačítko k potvrzení zadaných dat.
2. `text` - vytváří textovou oblast, do které uživatel může zadat hodnoty různého charakteru (jak textového, tak číselného).
3. `hidden` - vytvoří skrytou textovou oblast a používá se zpravidla pro odesílání pomocných dat pro server, které uživatel vůbec nemusí znát. Např. uživatele, který data odesílá, IP adresa počítače apod.
4. `password` - stejný charakter jako 'text', ovšem s tím rozdílem, že při zadávání hodnot, se místo hodnot zobrazuje heslový charakter textu, jako např. `*****`. Slouží zpravidla pro přihlašovací formuláře do databázového systému a zadávání hesel.

Na konci samotného kódu je formulář ukončen jeho párovým tagem `</FORM>`.

Všimněme si také funkce *echo*. Ta serveru říká, kterou část kódu má poslat klientovi k zobrazení. Vše, co se má zobrazit se píše do uvozovek a ukončuje středníkem (např. `echo „Teorie automatického řízení“;`)

Výše zapsaný zdrojový kód vytvoří prostředí pro výběr metody, který je zobrazen na Obr.11.



Obr. 11. Výběr metody pro výpočet spojitého regulátoru

Nyní se podívejme, co se děje na straně serveru, když uživatel odešle výše zmíněné informace (tzn. typ regulátoru a typ metody, která se má pro výpočet použít):

Server si po obdržení informací vytvoří příslušné proměnné *\$typeofmethod* a *\$typeofregulator* a přečte si PHP skript. Jakmile v kódu narazí na tento kus kódu:

```
if(isset($typeofmethod) && isset($typeofregulator))
```

tak server zkontroluje, zda byly tyto dvě proměnné něčím naplněny (resp. vytvořeny) pomocí funkce *isset()*. V tomto případě již něco proměnné obsahují, tudíž se podle zmíněné podmínky vykoná vše, co je obsaženo v těle podmínky. A tím je právě následující kód:

```
switch($typeofmethod)
{
    case 1:
        ZieglerNichols($typeofregulator);
        break;
    case 2:
        ReleVeZpetneVazbe($typeofregulator);
        break;
    .
    .
    .
}
```

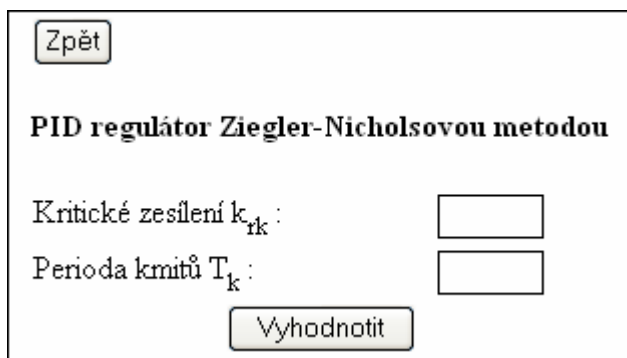
Přepínač *switch()* slouží k rozvrstvení kódu na základě jeho parametru. Zde konkrétně to tedy pracuje takto: V případě, že je v proměnné *\$typeofmethod* obsaženo číslo 1, pak se zavolá funkce *ZieglerNichols* s parametrem *\$typeofregulator*, tedy s proměnnou, která

obsahuje typ regulátoru, který se má počítat. Proměnná *\$typeofregulator* obsahuje buď „P“, „PI“, „PD“, nebo „PID“ textový řetězec. Pro ostatní metody výpočtu regulátoru je postup analogický.

Server tedy zavolá příslušnou funkci, podle zvolené metody výpočtu. Budeme tedy brát, že server volá výše zmíněnou funkci *ZieglerNichols*. Ta ve svém těle obsahuje následující kód:

```
echo "  
<FORM ACTION='?m=3.3' METHOD='post'>  
  <TR>  
    <TD>Kritické zesílení k<SUB>rk</SUB> :</TD>  
    <TD><INPUT TYPE='text' NAME='krk' SIZE='5'></TD>  
  </TR>  
  <TR>  
    <TD>Perioda kmitů T<SUB>k</SUB> :</TD>  
    <TD><INPUT TYPE='text' NAME='Tk' SIZE='5'></TD>  
  </TR>  
  <TR>  
    <TD ALIGN='center' COLSPAN='2'>  
      <INPUT TYPE='hidden' NAME='calculate' VALUE='1'>  
      <INPUT TYPE='hidden' NAME='typeofregulator'  
VALUE='$typeofregulator'>  
      <INPUT TYPE='submit' VALUE='Vyhodnotit'>  
    </TD>  
  </TR>  
</FORM>"
```

Následující kód se díky funkci *echo* již uživateli zobrazí podobně jako na Obr. 12.



Obr. 12. Zadání charakteristických hodnot

Jak je ze zdrojového kódu patrné, jde opět o formulář. Zde ale již o formulář, do kterého uživatel může zadávat hodnoty, které se budou moci při výpočtu regulátoru použít. Obsahuje tedy několik textových oblastí (tzv. input-boxů). Všimněme si, že tag `<INPUT>` obsahuje atribut `NAME`, který stejně jako u atributu `<SELECT>`, o kterém již byla řeč, udává název proměnné, do které se příslušná zadaná hodnota zapíše.

Po kliknutí na tlačítko „Vyhodnotit“ se serveru odešlou informace o údajích nutných k vypočtení regulátoru. Server opět přečte celý PHP skript a jakmile narazí na tuto řádku:

```
if(isset($calculate) && isset($typeofregulator))
```

zjistí se, zda byly proměnné `$calculate` a `$typeofregulator` naplněny a provede se tělo podmínky:

```
switch($calculate)
{
    case 1:
        if(isset($krk) && isset($Tk) && $krk!=0 && $Tk!=0)
            Calculate_ZieglerNichols($krk,$Tk,$typeofregulator);
        else
            echo "<TR><TD>Některý parametr je nulový!</TD></TR>";
        break;

    case 2:
        .
        .
        .
}
```

Proměnná *\$calculate* udává, že se má již v tomto kroku začít počítat, a zároveň obsahuje metodu výpočtu. Pokud je tedy *\$calculate* rovna 1, pak se zkontroluje, zda jsou naplněny všechny potřebné hodnoty (pokud ne, vypíše se chybová hláška) a je-li vše v pořádku, pak se zavolá příslušná výpočetní funkce, v našem případě *Calculate_ZieglerNichols* s parametry potřebných hodnot a typem regulátoru.

Funkce *Calculate_ZieglerNichols* pak obsahuje následující výpočetní kód:

```
switch($typeofregulator)
{
    case "PID":
        // časové konstanty
        $kr = 0.6*$krk;
        $TI = 0.5*$Tk;
        $TD = 0.125*$Tk;

        $r0 = 0.6*$krk;
        $rminus1 = 1.2*$krk/$Tk;
        $r1 = 0.075*$krk*$Tk;
        echo "Zde se vypíší výsledky";
    case "PI":
        // časové konstanty
        $kr = 0.45*$krk;
        $TI = 0.85*$Tk;

        $r0 = 0.45*$krk;
        $rminus1 = 0.54*$krk/$Tk;
        echo "Zde se vypíší výsledky";
        break;
    .
    .
    .
}
```

Funkce tedy rozhodne, kterou část výpočetního kódu má použít v závislosti na zvoleném typu regulátoru a vypočítá složky požadovaného regulátoru na základě tabulek a vypíše výsledky v podobě, kterou reprezentuje Obr.13.

Zpět	
Vypočtené parametry pro PID regulátor Ziegler-Nicholsovou metodou	
<u>Časové konstanty:</u>	<u>PID parametry regulátoru:</u>
$k_r = 3$	$r_0 = 3$
$T_I = 0.5$	$r_{-1} = 6$
$T_D = 0.125$	$r_1 = 0.375$
$R(s) = k_r \left(1 + \frac{1}{T_I s} + T_D s \right)$	$R(s) = r_0 + \frac{r_{-1}}{s} + r_1 s$

Obr. 13. Výpis vypočtených parametrů

Celý postup při odesílání a zpracovávání dat serverem se možná zdá příliš komplikovaný, ovšem při hlubším prozkoumávání částí a každého kroku předávání a manipulaci s proměnnými je proces pro amatérského programátora celkem zřejmý a pochopitelný.

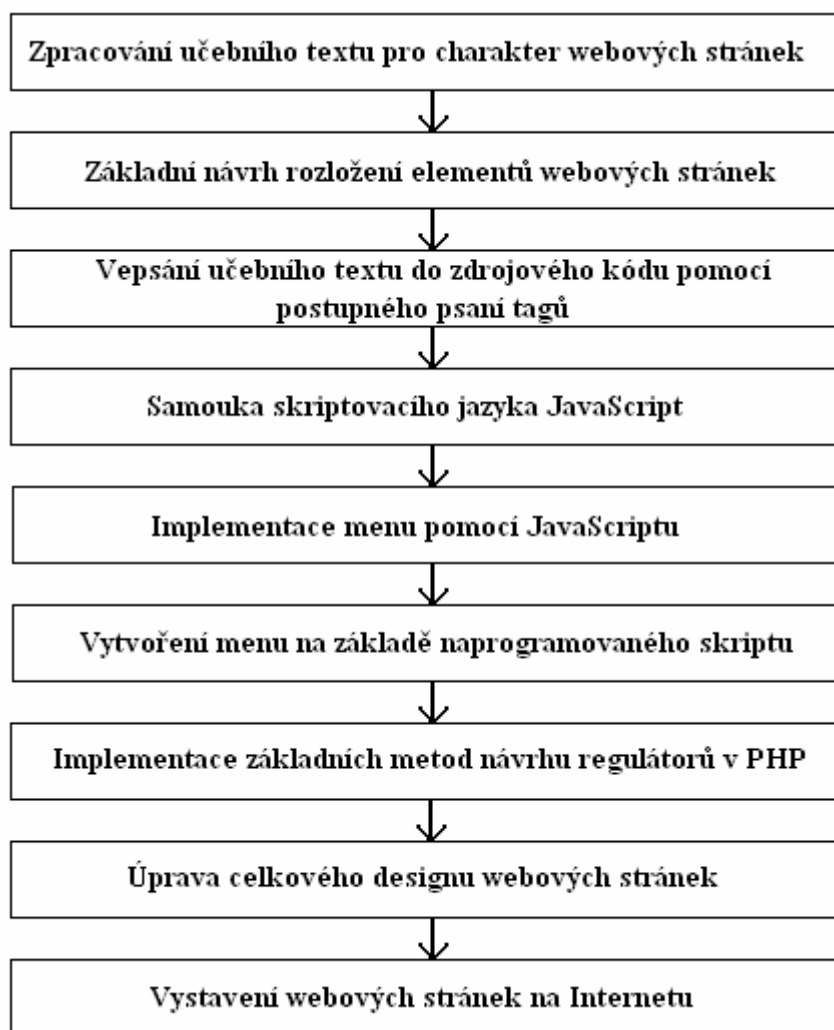
Pro výpočet regulátorů jsou na webových stránkách v kapitole **3.3 Výpočet spojitého regulátoru** programově zpracovány následující metody:

- Ziegler-Nicholsova metoda
- Využití relé ve zpětné vazbě
- Nastavení z přechodové char. aperiodického typu
- Äströмова úprava
- Cohen-Coonova metoda
- Chien, Hrones a Reswickova metoda
- Vyvážené nastavení

Ostatní metody jsou již natolik komplikované, že by jejich implementace v PHP vystačila na samostatnou diplomovou práci.

SHRNUTÍ PRÁCE

Během tvorby webových stránek bylo nutné podniknout několik nezbytných kroků. Základní průběh práce bez řešení problémů, které v téměř každém kroku nastaly, ukazuje diagram na Obr.14.



Obr. 14. Průběh práce

Po zpracování učebního textu tak, aby ho bylo možno použít na webových stránkách, jsem navrhl základní koncept, jak by mohly webové stránky vypadat. To se samozřejmě v průběhu vývoje neustále měnilo v závislosti na dalších přidávaných elementech. Následně byl učební text, doprovodné rovnice a obrázky „ručně“ vepsány do zdrojového kódu. Bylo také potřeba mít pro uživatele nachystáno jednoduché a přehledné menu k navigaci po stránkách. Nejlepším způsobem k vytvoření požadovaného menu bylo naprogramovat ho ve skriptovacím jazyce JavaScript, který se pro tvorbu podobných

elementů hodí nejlépe. Bylo ovšem nutné se JavaScript nejprve od základů až do potřebných znalostí naučit, protože až do této doby jsem ještě z programového hlediska JavaScript nikdy nevyužil. Teprve poté bylo možné skript menu implementovat a přizpůsobit jej webovým stránkám. Součástí stránek mělo také být rozhraní, které má uživateli počítat parametry regulátorů pomocí základních algoritmů návrhu. Pro tento úkol jsem zvolil skriptovací jazyk PHP, který je svou syntaxí velmi podobný jazyku C++. Po skončení těchto základních prací bylo nutné celkově doladit vzhled webových stránek, aby byly pro uživatele příjemné a aby jejich vzhled při studiu učebního textu příliš nerušil. Nakonec byly webové stránky pro studenty a veřejnost zpřístupněny na Internetu.

Konečnou podobu webových stránek ukazuje Obr.15. Hlavní menu můžeme vidět v horní čtvrtině nad hlavním nadpisem. Vpravo pod obrázkem Nyquistovy křivky (které je zde pouze pro zkrášlení celkového vzhledu) je umístěno menu dodatků, které má sloužit k celkovému přehledu zpracovaných témat, nalezení a vysvětlení určitého pojmu či seznamy symbolů, obrázků a tabulek, které jsou na webových stránkách použity. Pod tímto menu doplňků jsou k dispozici odkazy (např. na Matlab WEB Server). Největší prostor webových stránek slouží pro prezentaci a výklad učebního textu určitého vybraného tématu. Zjednodušený koncept tohoto rozvržení stránky je zobrazen na Obr.6.

The screenshot shows a web browser window with the address `http://localhost/tar/?m=3.1`. The website has a blue header with the title 'Teorie automatického řízení - Lineární spojitě dynamické systémy'. Below the header is a navigation menu with links: 'Úvod do teorie systémů LDS', 'Lineární spojitě systémy', 'Spojitě regulátory', 'Polynomiální metody návrhu regulátorů', 'Mnohorozměrné systémy', and 'Popis systémů ve stavovém prostoru'. The main content area is titled 'TAŘ - Lineární spojitě dynamické systémy' and contains a section '3.1. Spojité regulátory a metody jejich nastavení'. This section includes a block diagram of a feedback control system (Obr. 3.1) and a brief text description of the control goal. The diagram shows a reference input w entering a summing junction, which also receives negative feedback from the output y . The output of the summing junction is the error signal e , which is processed by a controller $R(s)$ to produce the control signal u . This signal u is then processed by the plant $G(s)$ to produce the output y .

Obr. 15. Konečná podoba vytvořených webových stránek

ZÁVĚR

Cílem bakalářské práce bylo vytvořit elektronickou učebnici ve formě webových stránek pro posluchače bakalářského studia k předmětu Teorie automatického řízení I. Pro tvorbu webových stránek byl použit HTML jazyk ve spojení s profesionálními skriptovacími jazyky jako je PHP a JavaScript a nástavbou CSS pro lepší formátování textu a vylepšení celkového designu stránek.

Stránky jsou kompletně napsány ručním zapisováním kódu, aby byl kód pro případné úpravy a aktualizace snadno čitelný i pro ostatní designéry a programátory a aby byl celkový kód minimální co do velikosti, což je přímo úměrné rychlosti stahování ze serveru do klienta (internetového prohlížeče). Práce je tedy kvůli postupnému ručnímu zapisování kódu velmi časově náročná.

Stránky obsahují open-source menu, které bylo naprogramováno v JavaScriptu, a jeho kód je možno opětovně použít na tvorbu dalších (i vzhledově odlišných) menu na jiných webových stránkách. Zdrojový kód je obsažen v příloze P 1.

Na webových stránkách je také možno nechat si serverem vypočítat parametry spojitých regulátorů pomocí základních metod návrhu. Vše se děje samozřejmě na základě určitých charakteristických hodnot systému, který chceme regulovat. Algoritmy těchto metod návrhu spojitých regulátorů byly naprogramovány ve skriptovacím jazyce PHP, které je pro podobné početní operace na straně serveru velmi vhodné.

Podařilo se tedy vytvořit jednoduché, ale přesto přehledné webové stránky obsahující studijní text, který má studentům sloužit jako studijní pomůcka a pomoci jim zorientovat se v problematice lineárních spojitých dynamických systémů.

SEZNAM POUŽITÉ LITERATURY

- [1] Prokop, R., Matušů, R., Prokopová, Z.: Teorie automatického řízení - lineární spojité dynamické systémy. Rukopis, skriptum FT UTB ve Zlíně, 2005.
- [2] Balátě, J.: Automatické řízení. BEN - technická literatura, Praha, 2003.
- [3] Švarc, I.: Automatizace, Automatické řízení. Akademické nakladatelství CERM, Brno, 2002.
- [4] Kosek, J.: Vše o WWW [online]. Dostupné na WWW: <http://www.kosek.cz>
- [5] Jak psát web [online]. Dostupné na WWW: <http://www.jakpsatweb.cz>
- [6] Builder [online]. Dostupné na WWW: <http://www.builder.cz>

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

WWW	World Wide Web
HTML	Hypertext Markup Language
PHP	Professional Home Page
CSS	Cascade Style Sheets
URL	Uniform Resource Locator (jednoznačné určení zdroje)
JPG	Joint Photographic Experts Group
GIF	Graphic Interchange Format
TAG	řídící znak
FRAME	rám
px	pixel

SEZNAM OBRÁZKŮ

Obr. 1. Stránka vytvořená pomocí rámu	21
Obr. 2. Tabulka se spojenými buňkami	23
Obr. 3. Klientský skript	27
Obr. 4. Serverový skript	29
Obr. 5. Klasické rozložení webové stránky	31
Obr. 6. Základní koncept vzhledu stránky	32
Obr. 7. Náhled základního menu	40
Obr. 8. Náhled základního menu při nájezdu myši na položku	40
Obr. 9. Konečný vzhled menu na webových stránkách	41
Obr. 10. Menu v podkapitolách	41
Obr. 11. Výběr metody pro výpočet spojitého regulátoru	45
Obr. 12. Zadání charakteristických hodnot	47
Obr. 13. Výpis vypočtených parametrů	49
Obr. 14. Průběh práce	50
Obr. 15. Konečná podoba vytvořených webových stránek	51

SEZNAM TABULEK

Tab. 1 Zápis tagu	15
Tab. 2 Struktura HTML kódu	16
Tab. 3 Fyzické a logické formátování textu	16
Tab. 4 Bloky a seznamy	17
Tab. 5 Vkládání obrázků	17
Tab. 6 Atributy tagu 	18
Tab. 7 Atributy tagu <A>	18
Tab. 8 Nejčastější schémata pro URL	19
Tab. 9 Rámy	21
Tab. 10 Tabulky	22
Tab. 11 Atributy tagu <TD>	23

SEZNAM PŘÍLOH

P 1 : Zdrojový kód souboru scriptmenu.js	58
P 2 : CD-ROM	62

P 1 : Zdrojový kód souboru scriptmenu.js

Zdrojový kód souboru *scriptmenu.js*:

```
var menuType=0;
var menuID=1;
var menuX=0;
var menuY=0;
var menuObject=1;
var menuWidth=0;
var anyMenuIsOpen=0;

function StartMainMenu(type)
{
    menuType=type;
    document.write("<table cellpadding=0px cellspacing=0px>");
}

function EndMainMenu()
{
    document.write("</table>");
}

function MainMenuItem(name,text,url,width,position,style)
{
    if(menuType==0)
        document.write("<tr>");

    document.write("<td align=center>");

    document.write("<a ");
    document.write("href=\""+url+"\" ");
    document.write("onmouseout=\"");
    document.write("TidyMenu();");
    document.write("\" onmouseover=\"");
    document.write("OpenMenu(event, '"+name+"', '"+position+"');");
    document.write("return false;\"");
    document.write(" class=\""+style+"\" ");
    style="width:"+width+"px";
    document.write(">");
}
```

```
document.write("&nbsp;" + text + "&nbsp;");
document.write("</a>");

document.write("</td>");

if(menuType==0)
    document.write("</tr>");
}

function StartSubMenu(name, style, width)
{
    menuWidth=width;
    document.write("<div id=\""+name+"\" class=\""+style+"\"
style=\"width:\"+(menuWidth)+"px\">");
}

function EndSubMenu(name)
{
    document.write("</div>");
    document.getElementById(name).onmouseout=TidyMenu;
}

function SubMenuItem(name, text, url, position, style)
{
    document.write("<a ");

    if(url!="")
        document.write("href=\""+url+"\" ");

    if(name!="")
    {
        document.write("onmouseout=\"");
        document.write("TidyMenu();");
        document.write("\" onmouseover=\"");
        document.write("OpenMenu(event, '"+name+"', '"+position+"');");
        document.write("return false;\"");
    }
    document.write(" class=\""+style+"\"
style=\"width:\"+menuWidth+"px\"");

    document.write(">");
```

```
document.write("&nbsp;" + text + "&nbsp;");
document.write("</a>");
}

function OpenMenu(event, id, pos)
{
    var el, x, y;

    if (anyMenuIsOpen == 1)
    {
        el = document.getElementById(menuID);
        if (menuX != el.offsetLeft || menuY != el.offsetTop)
        {
            el.style.left = menuX;
            el.style.top = menuY;
        }
        TidyMenu();
    }

    x = event.clientX - event.offsetX + document.body.scrollLeft -
document.body.clientLeft;
    y = event.clientY - event.offsetY + document.body.scrollTop -
document.body.clientTop - bd;
    dx = event.srcElement.offsetWidth;
    dy = event.srcElement.offsetHeight;

    el = document.getElementById(id);

    if (el && el.style.visibility != "visible")
    {
        if (pos == 0)
        {
            x += dx;
            el.style.left = x - el.offsetWidth;
            el.style.top = y;
        }
        else if (pos == 1)
        {
            y += dy;
            el.style.left = x;
            el.style.top = y - el.offsetHeight;
        }
    }
}
```

```
        menuX=x;
        menuY=y;

        el.style.left=x;
        el.style.top=y;

        el.style.visibility="visible";
        menuID=id;
        menuObject=event.srcElement.style;
    }
}

function OverMenu(x,y)
{
    var el=document.getElementById(menuID);
    if(el.offsetLeft+el.offsetWidth>x&&el.offsetLeft<=x&&el.offsetTop
+el.offsetHeight>y&&el.offsetTop<=y)
    {
        return menuID;
    }
    return "";
}

function TidyMenu()
{
    t=OverMenu(event.clientX+document.body.scrollLeft-
document.body.clientLeft,event.clientY+document.body.scrollTop-
document.body.clientTop);

    var item_length=menuID.length;
    if(item_length>t.length||t.substring(0,item_length)!=menuID)
    {
        var el=document.getElementById(menuID);
        el.style.visibility="hidden";
        menuObject.backgroundColor="";
        menuObject.color="";
    }
}
```

P 2 : CD-ROM

Příloha obsahuje program PHP Triad 2.2.1, kompletní zdrojové kódy webových stránek a elektronickou podobu bakalářské práce.