

Jihočeská univerzita v Českých Budějovicích

Pedagogická fakulta – Katedra fyziky

**Prostředky pro návrh a tvorbu dynamických HTML stránek
Resources for design and realization dynamical HTML pages**

Bakalářská práce

Vedoucí práce: **Ing. Michal Šerý**

Autor: **Pavel Jurda**

Tato práce pojednává o volbě prostředků pro návrh a tvorbu dynamických HTML stránek. Jejím cílem je teoretický úvod nezbytný pro pochopení principu tvorby HTML stránek, praktický popis a rozbor využití programových prostředků a vybavení, jakými jsou například CSS, ASP, IIS, SQL a další, při návrhu dynamických HTML stránek a ucelený pohled na výše zmíněnou problematiku.

Při tvorbě této práce je hlavní důraz kladen na popis možností využití prostředků při tvorbě samotných dynamických HTML stránek (kapitoly 3 až 6).

Ve druhé části práce (kapitoly 7 až 9) je osvětlena problematika komunikace a sdílení dat mezi dynamickými HTML stránkami a serverovou částí databázového systému.

V závěrečné části práce jsou nastíněny některé obecné skutečnosti vztahující se k problematice tvorby dynamických HTML stránek.

Abstrakt:

This research paper diserts on resources for design and realization dynamical HTML pages. The aim is a theoretical introduction necessary for understanding of principles of creating HTML pages, a practical description and an analysis on an use of software resources and facilities such are CSS, ASP, IIS, SQL etc. for design of dynamical HTML pages and a compact view on a subject mentioned above.

The main emphasis is put on description of possibilities of an use of resources for creating dynamical HTML pages itself (chapters 3 - 6).

The second part of this research paper (chapters 7 - 9) deals with the issues of communication and sharing data between dynamical HTML pages and server fraction of database system.

In the closing part there is some general information linked with the issues of creation of dynamical HTML pages outlined.

Prohlašuji, že předloženou práci jsem vypracoval samostatně, pouze s použitím uvedené literatury.

V Praze dne 26. listopadu 2006

Pavel Jurda

Na tomto místě a touto formou bych rád poděkoval panu Ing. Michalovi Šerému, vedoucímu mé bakalářské práce, za cenné rady a odborné připomínky při zpracování mé práce.

Obsah:

1 Úvod	8
2 Rozbor volby programových prostředků a vybavení	9
3 Jazyk HTML a XHTML	10
3.1 Vložené značky HTML	10
3.2 Kostra dokumentu HTML/XHTML	10
3.3 Komentáře	11
3.4 Text	11
3.4.1 Vnější vzhled textu	11
3.4.2 Textové styly orientované na obsah	12
3.4.3 Fyzické styly	12
3.4.4 Speciální textové znaky	12
3.4.5 Oddíly, odstavce a zalomení řádku	13
3.4.6 Nadpisy	13
3.4.7 Vodorovné oddělovače	13
3.4.8 Předem formátovaný text	13
3.5 Hypertextové odkazy	13
3.6 Adresy URL	14
3.7 Kotvy	14
3.7.1 Kotvy mimo HTML a XHTML	14
3.8 Obrázky	15
3.8.1 Přímo vložené obrázky	15
3.8.2 Obrazové mapy	15
3.9 Seznamy, vyhledávací dokumenty a formuláře	15
3.9.1 Neuspořádané, uspořádané a definiční seznamy	15
3.9.2 Formuláře	16
3.10 Tabulky	16
3.11 Rámečky	17
4 Dynamické HTML stránky	18
4.1 Vznik interaktivních stránek	18
4.2 Definování dokumentu HTML/XHTML	18
4.3 Programování dynamického HTML	19
4.3.1 Objektový model	19
4.3.2 Vývoj hierarchie dynamického HTML	20
4.3.3 Dynamické obtékání	22
4.3.4 Příprava skriptů	22
4.4 Prvek skript	22
4.4.1 Umístění skriptů v dokumentu	23
4.4.2 Odložení provedení skriptu	23
4.4.3 Více jazyků skriptů	23
4.4.4 Přesměrování na stránku klienta	24
4.4.5 Výběr jazyka skriptu: JavaScriptu a VBScript	24
4.5 Události typu intrinsic	24
4.5.1 Atributy události	25
4.5.2 Podpora generické události	26
4.5.3 Časování vazby události	26
4.6 Skripty jednotlivých prvků	26
4.6.1 Prvky Body a Frameset	26
4.6.2 Události okna	27

4. 6. 3	Programování obsahu sady rámu	27
4. 6. 4	Programování prvku A (Anchor) a Base HREF	27
4. 6. 5	Programování prvků IMG a MAP	28
4. 6. 6	Programování prvku Object	29
4. 6. 7	Programování prvku Table	29
4. 6. 8	Skripty prvku Form	30
4. 6. 9	Programování prvku Marquee	30
5	CSS (Cascading Style Sheets)	32
5. 1	Základní koncepce CSS	32
5. 2	Elementy v CSS a jejich vlastnosti	34
5. 2. 1	Základní elementy HTML kódu	34
5. 2. 2	Třídy (CLASS)	35
5. 2. 3	Identifikátory (ID)	36
5. 2. 4	Pseudotřídy a pseudoprvky	36
5. 3	Kaskádové pořadí	37
5. 4	Jednotky	38
5. 4. 1	Délkové jednotky	38
5. 4. 2	Procenta	38
5. 5	Barvy	39
5. 6	Přehled vlastností CSS1	39
5. 6. 1	Vlastnosti barev a pozadí	39
5. 6. 2	Vlastnosti písma	40
5. 6. 3	Vlastnosti textu	40
5. 6. 4	Vlastnosti boxů a bloků	40
5. 6. 5	Klasifikační vlastnosti	42
5. 6. 6	Vlastnosti pro řízení pozice	42
5. 7	Nové vlastnosti CSS2	42
5. 7. 1	Rozlišení typů médií	42
5. 7. 2	Vlastnosti zvukových stylů	43
5. 7. 3	Nastavení kurzoru myši	43
5. 7. 4	Filtry	44
5. 7. 5	Nové vlastnosti písma	44
5. 7. 6	Nové vlastnosti boxů a bloků	46
5. 7. 7	Vlastnosti pro řízení pozice	46
6	ASP (Active Server Pages)	48
6. 1	Kolekce	48
6. 1. 1	Metody kolekcí	48
6. 2	Vnitřní objekty ASP	49
6. 2. 1	Objekt Application	49
6. 2. 2	Objekt ASPError	49
6. 2. 3	Objekt Request	50
6. 2. 4	Objekt Response	51
6. 2. 5	Objekt Server	52
6. 2. 6	Objekt Session	53
6. 3	Skriptovací objekty	54
6. 3. 1	Objekt Dictionary	54
6. 3. 2	Objekt FileSystemObject	54
6. 3. 3	Objekt Drive	55
6. 3. 4	Objekt Folder	56
6. 3. 5	Objekt File	57

6. 3. 6	<i>Objekt TextStream</i>	58
6. 4	ASP versus PHP	59
7	IIS (Internet Information Server)	60
8	ADO (ActiveX Data Objects)	61
8. 1.	Základní objekty ADO	61
8. 1. 1	<i>Connection</i>	61
8. 1. 2	<i>Recordset</i>	62
8. 2.	Specializované objekty ADO	62
8. 2. 1	<i>Error</i>	63
8. 2. 2	<i>Command</i>	63
8. 2. 3	<i>Parameter</i>	63
8. 2. 4	<i>Collection</i>	63
8. 2. 5	<i>Property</i>	64
8. 3	Události	64
8. 3. 1	<i>ConnectionEvents</i>	64
8. 3. 2	<i>RecordsetEvents</i>	64
8. 4	Rozšíření ADO	65
8. 4. 1	<i>ADO Extensions for DDL and Security (ADOX)</i>	65
9	Databáze SQL Serveru (Structured Query Language Server)	66
9. 1	Protokol ODBC	66
9. 2	Jazyk SQL	66
9. 2. 1	<i>Vytvoření tabulky</i>	67
9. 2. 2	<i>Přidání nového záznamu</i>	67
9. 2. 3	<i>Výběr a prohlížení záznamů</i>	67
9. 2. 4	<i>Mazání záznamů v tabulce</i>	67
9. 2. 5	<i>Změna obsahu tabulky</i>	67
9. 3	Začleňování příkazů jazyka SQL do skriptů	67
10	Využití Microsoft Internet Exploreru pro realizaci návrhu	69
10. 1	Historie Microsoft Internet Explorer	69
10. 2	Využití Microsoft Internet Explorer	69
11	Možnosti dalšího rozšíření návrhu	71
12	Zhodnocení, závěr	73
	Seznam literatury	74

1 Úvod

Dynamické HTML stránky - o co se vlastně jedná? Především musíme říci, že dynamické HTML nepřidává do samotného jazyka HTML žádné nové elementy nebo atributy. Dynamické HTML pouze definuje způsob jakým mohou skripty zařazené do stránky měnit její obsah - přidávat a ubírat elementy, měnit jejich obsah a pohrávat si s hodnotami atributů. Veškeré změny jsou prohlížečem okamžitě zjištěny a zobrazení stránky se přizpůsobí aktuálnímu stavu.

Touto technikou lze dosáhnout právě oněch na první pohled dynamicky vypadajících změn stránky. Ještě výraznější dopad na vzhled má měnění stylů. Pomocí kaskádových stylů (CSS) můžeme velice efektivně ovládat atributy zobrazení jednotlivých elementů. V dynamickém HTML můžeme samozřejmě pomocí skriptů měnit i styly a tím dosáhnout mnoha zajímavých efektů.

Pokud tedy chceme vytvářet stránky v dynamickém HTML, musíme umět používat jazyk HTML společně se styly a ovládat některý ze skriptovacích jazyků. Zopakujeme proto nejdůležitější fakta a seznámíme se s rozšířeními, která byla do stylů přidána právě kvůli dynamickému HTML.

Dynamické HTML zatím není nijak standardizováno - např. jako jazyk HTML nebo protokol HTTP. Konsorcium W3C již vytvořilo pracovní návrh tzv. DOM (Document Object Model) - objektového modelu dokumentu. Tento dokument popisuje způsob, jakým jsou přístupné jednotlivé elementy a atributy HTML dokumentu. Implementace DOM v skriptovacím jazyku je tedy nutnou podmínkou pro používání dynamického HTML.

Microsoft však nečekal, až bude DOM schválen a implementoval dynamické HTML již asi před rokem. Naštěstí Microsoft spolupracoval při vytváření DOM a tak dynamické HTML v podání Microsoftu z velké části požadavky definované v DOM splňuje. Z toho také vyplývá, že dynamické HTML zatím podporuje pouze prohlížeč Internet Explorer 4.0 a výše. Netscape měl vlastní představu dynamického HTML, ta však nebyla tak technologicky vyspělá.

2 Rozbor volby programových prostředků a vybavení

Pro návrh a tvorbu dynamických HTML stránek je třeba využívat nejen klasických značek a příkazů jazyka HTML, ale i další programové prostředky a vybavení.

Pro účely této práce budeme používat především skripty a objekty dynamických HTML stránek a dále jazyk CSS (Cascading Style Sheets). Pro tvorbu a provozování dynamických, interaktivních webových aplikací budeme využívat skriptovací prostředí ASP (Active Server Pages), zmíníme též skriptovací jazyk PHP.

Na serverové straně zmíníme produkt Internet Information Server od společnosti Microsoft a programové rozhraní ADO (ActiveX Data Objects). Poslední částí programových prostředků a vybavení je databáze uložená a provozovaná na serveru SQL (Structured Query Language Server).

3 Jazyk HTML a XHTML

Jazyk HTML ve verzi 4.01 již dosáhl určitého věku a nedají se očekávat jeho další kvalitativní změny. Je proto dnes zahrnut jako jakýsi modul v rámci jazyka XML (Extensible Markup Language, rozšiřitelný značkovací jazyk) konsorcia W3C. Jazyk XML nabízí možnost vytváření nových značkovacích jazyků, které vycházejí ze standardů a k jejichž implementaci není potřeba souhlasu W3C. Jazyky založené na standardu XML popisují informace, které se posléze dají analyzovat, zpracovávat, zobrazovat, třdit a sestavovat pomocí řady různých komunikačních technologií. Konsorciem W3C nově vyvinutý jazyk XHTML (Extensible Hypertext Markup Language) tak vyjadřuje jazyk HTML, přeformulovaný podle zásad standardu XML.

Tato kapitola podává stručný přehled poměrně velkého množství různých elementů jazyka HTML a jeho nástupce XHTML.

3.1 Vložené značky HTML

O jazyku HTML a stejně tak i XHTML se dá říci, že jsou to vkládané jazyky. Příkazy neboli značky jazyka vkládáme přímo do stejného dokumentu, který později načítáme do prohlížeče. Ten se pak podle informací v těchto značkách rozhodne, jakým způsobem zobrazit nebo jinak zpracovat následný obsah dokumentu. Značky použité v textu sdělují prohlížeči, jak má text zobrazit. Vžitý termín pro tyto značky je tag.

Příklady:

tag <TITLE> a ukončovací tag </TITLE>	- vymezují název dokumentu
tagy <H1> a </H1>	- vyznačují nadpis
tag <P>	- odděluje odstavce

Většina značek (tagů) definuje a ovlivňuje určitou diskrétní oblast dokumentu. Tato oblast začíná místem, kde se značka se svými atributy poprvé objeví ve zdrojovém dokumentu (zde se takové značce říká počáteční značka), končí odpovídající koncovou značkou. Koncovou značku tvoří jméno počáteční značky před níž se zapisuje lomítko (/). Počáteční značce <i>, tedy odpovídá koncová značka </i>.

Prvním slovem značky je její formální jméno, které zároveň obvykle dobře vystihuje její význam a funkci. Veškeré další výrazy v zápisu značky jsou již speciální atributy, s nimiž někdy bývá spojena hodnota, oddělena od jména atributu znaménkem rovná se (=). Tyto atributy blíže definují nebo modifikují možnosti značky.

Koncové značky nikdy neobsahují atributy. V jazyce HTML má většina značek svoji koncovou značku, nikoli však všechny. Software většiny prohlížečů si navíc často dokáže z okolního a zřejmého kontextu koncovou značku domyslet, takže některé koncové značky do zdrojového dokumentu explicitně doplňovat nemusíme. Jazyk XHTML je oproti tomu přísnější a odpovídající koncovou značku vyžaduje u každé značky.

3.2 Kostra dokumentu HTML/XHTML

Celý dokument HTML začíná a končí značkami <html> a </html>. Tyto značky neříkají prohlížeči nic jiného, než že celý dokument tvoří kód jazyka HTML. Dokumenty jazyka XHTML začínají také značkou <html>, ale od běžných dokumentů HTML se však liší v jiných doplňujících informacích. Standardy HTML a XHTML vyžadují značku <html> v každém dokumentu, většina prohlížečů však dokáže

kódování HTML správně detekovat a zobrazit i v textovém dokumentu, v němž tato značka struktury nejvyšší úrovně chybí.

Všechny dokumenty HTML/XHTML mají dvě základní struktury – záhlaví a tělo, přičemž každou z nich v dokumentu uzavírá odpovídající počáteční a koncová značka (<head> respektive </head> a <body> respektive </body>). Do záhlaví zapisujeme obecné informace o dokumentu včetně značky <title> a ukončovací značky </title>, které vymezují název celého dokumentu. Tělo tvoří vlastní obsah, který se má zobrazit v okně prohlížeče.

Kromě značek <html>, <head>, <body> a <title> obsahují standardy HTML a XHTML již jen několik málo dalších povinných strukturálních elementů. Do obsahu dokumentu pak můžeme zapsat v podstatě cokoliv jiného. Ve skutečnosti existují tři základní typy obsahu HTML/XHTML - značky (o nichž jsme si již zmínili), komentáře a text.

3.3 Komentáře

Komentář je sice součástí dokumentu, v prohlížeči se z něj ale ve výsledku nic nezobrazí, tedy zejména ani text těla komentáře, zapsaného mezi speciální oddělovače – počáteční značku „<!--“ a koncovou značkou „-->“.

3.4 Text

Většina obsahu dokumentů HTML/XHTML - tedy ta část, kterou čtenář uvidí po zobrazení v prohlížeči - je text. Strukturu textu definují speciální značky, jako jsou například záhlaví, seznamy a tabulky. Další značky dávají prohlížeči pokyny, jakým způsobem má obsah dokumentu formátovat a zobrazit. Textové značky jazyků HTML/XHTML tvoří největší podmnožinu obou standardních jazyků. Integrace multimediálního obsahu je již něčím navíc, jakýmsi doplňkem jazyků HTML/XHTML. Data, která tvoří digitální obrázky, videosekvence, zvukové záznamy a další multimediální elementy, jaké mohou být součástí zobrazení v prohlížeči, jsou samostatnými dokumenty a od hlavního dokumentu jsou uloženy odděleně. Do hlavního dokumentu zapíšeme s pomocí speciálních značek pouze odkazy na tyto multimediální elementy. Prohlížeč pak podle těchto odkazů načte dokumenty ostatních typů a integruje je s vlastním textem dokumentu.

3.4.1 Vnější vzhled textu

Jazyk HTML ani XHTML není jazykem pro definici rozvržení stránek. Díky širokému spektru nejrůznějších prohlížečů, které se dají navíc uživatelsky přizpůsobit, a také díky velkému množství různých počítačových platforem pro načítání a zobrazování elektronických dokumentů v podstatě ani všem značkovacím jazykům nezbyvá nic jiného, než že se snaží prohlížeči radit, jak by se dokument dal zobrazit a jak by mohl vypadat, to znamená, že nikomu nic nediktují. Žádný prohlížeč nemůžeme přinutit, aby dokument zobrazil nějakým pevně daným způsobem. V dokumentu nelze například předem předpokládat, v jakém písmu a v jaké absolutní velikosti se zobrazí text danému konkrétnímu uživateli.

Styly mají nicméně svůj nepopíratelný vliv na čitelnost textu, takže je dobré je používat, kdekoli je to jen trochu možné a kde neruší vlastní prezentaci obsahu dokumentu. Běžné atributy stylu můžeme k textu připojit s pomocí značek fyzických stylů, jako je například značka kurzívy <i>. Mnohem důležitější jsou ale v jazyce

HTML a XHTML takzvané kontextově orientované značky, které lépe odpovídají původnímu poslání jazyka, ty totiž přiřazují určitým pasážím textu význam. A charakteristiky zobrazení textu (jako je například řez a velikost písma, barva a podobně) lze změnit pomocí kaskádovitých šablon stylů (Cascading Style Sheets, CSS).

Fyzické i kontextově orientované značky textových stylů dokáží rozpoznat všechny dnešní grafické prohlížeče a samozřejmě umí odpovídajícím způsobem také změnit vzhled příslušného textu, který pak vyjadřuje určitý význam nebo strukturu. Jediné, co nedokážeme předem odhadnout, je konečný vzhled textu po takovéto změně.

Standard jazyka HTML, a zejména pak standard XHTML, přitom zdůrazňuje, že budoucí prohlížeče nemusí být (a ani nebudou) tak spoutané vizuálním zobrazením. Uživatel nakonec příslušným textový obsah neuvidí, ale například jej uslyší. A v těchto případech se kontextové mechanismy hodí rozhodně lépe než fyzické styly.

3. 4. 2 Textové styly orientované na obsah

Značky stylů orientované na obsah prohlížeči říkají, že určitá část textu HTML nebo XHTML má jistý speciální význam nebo využití. Nejběžnějším stylem orientovaným na obsah je v současné době jednoznačně zvýraznění textu, které označuje značka `<em>`. Pokud chceme zvolit ještě silnější zdůraznění textu můžeme to udělat pomocí stylů obsahu `<strong>`. Dalším stylem orientovaným na obsah je například styl `<code>`, který slouží k zápisu částí programového kódu. Dále styl `<kbd>`, který vymezuje text zapisovaný uživatelem z klávesnice, styl `<samp>`, jenž označuje ukázkový (vzorový) text. Dále sem patří ještě styl `<dfn>` pro definice a značka `<var>`, která v ukázkách programového kódu vymezuje jména proměnných. Všechny tyto značky mají samozřejmě své odpovídající koncové značky.

3. 4. 3 Fyzické styly

Jazyky HTML/XHTML nejsou sice nástroji pro zpracování textu v klasickém slova smyslu, přesto ale nabízí značky, které prohlížeči, pokud to umí, nařizují explicitně zobrazit daný znak, slovo či fráze v konkrétním fyzickém stylu. Tak například pomocí značky `<i>` značíme text, který se má vypsát kursivou, značka `<b>` zobrazí podobně daný text v tučném písmu a text definovaný značkou `<tt>` prohlížeč vypíše v neproporcionálním písmu, tedy v písmu s pevnou šířkou znaku, jaké známe například z psacího stroje. Většina tvůrců používá styly orientované na obsah.

3. 4. 4 Speciální textové znaky

Ne všechny znaky, které se dají v prohlížeči zobrazit je možné zapsat přímo z klávesnice. Některé znaky mají navíc speciální význam příkladem mohou být třeba úhlové závorky. Ty pokud je nějak neodlišíme a zapíšeme je přímo do prostého textu jako například znaménko menší než (`<`) v matematickém výrazu, dokáží docela poplést prohlížeč a poničit celý dokument. V jazycích HTML/XHTML můžeme do textu zapisovat libovolné z řady různých znaků, které tvoří znakovou sadu ASCII a to prostřednictvím speciálního kódování příslušné znakové entity. Znaková entita začíná vždy ampersandem, za nímž následuje jméno entity a středník pro ukončení. Druhá možnost je zapsat do sekvence znakové entity místo jejího jména symbol libry (`#`) a číslo pozice znaku v tabulce kódu ASCII. Při zobrazení dokumentu pak prohlížeč vypíše odpovídající znak, ale jen v tom případě, existuje-li v písmu, které jsme zvolili.

Nejběžnějšími používanými entitami jsou, jak si asi každý domyslí, znaky větší než (>), menší než (<) a ampersand (&).

3. 4. 5 Oddíly, odstavce a zalomení řádku

Jazyky HTML/XHTML nabízí možnosti explicitního i implicitního řízení základní struktury dokumentů. Nejjednodušší a zároveň nejběžnější způsoby strukturování dokumentu představují značky pro vytvoření oddílu neboli divize <div>, značka odstavce <p> a zalomení řádku
. Všechny tyto značky přeruší plynulý tok textu, který tím pádem pokračuje na novém řádku. Rozdíl mezi jednotlivými značkami spočívá v tom, že značky <div> a <p> definují v dokumentu nebo textu jistou oblast, přičemž obsah této oblasti můžeme v okně prohlížeče speciálním způsobem zarovnat, využít na něm textové styly a provádět další možné operace. Bez speciálních atributů pro zarovnání přitom značky <div> a
 jednoduše zalomí řádek textu, následující znaky umístí na nový řádek. Oproti tomu značka <p> přidá po zalomení řádku určitou vertikální mezeru.

Ve standardu HTML pro značku
 neexistuje koncová značka. Jazyk XHTML koncovou značku vyžaduje, ale zapisuje se netradičně do jedné a těch samých úhlových závorek.
.

3. 4. 6 Nadpisy

Kromě rozdělení textu na oddíly nebo odstavce můžeme samozřejmě dokument uspořádat i do oddílů s nadpisy. Jazyk HTML zná celkem 6 značek nadpisů, jsou to značky <h1> až <h6> spolu se svými odpovídajícími koncovými značkami. Prohlížeč zobrazuje zpravidla jejich obsah o různé velikosti od největšího po nejmenší, nikdy navíc i v tučném provedení. Značky nadpisů také zpravidla zalomí aktuální tok textů. To tedy znamená, že nadpis tvoří samostatný řádek, oddělený od okolního textu, i když před ním ani za ním nezapišeme explicitně žádnou značku zalomení řádku či odstavce.

3. 4. 7 Vodorovné oddělovače

Jak jazyk HTML tak i XHTML nabízí rozdělení oddílů dokumentů pomocí vodorovného oddělovače, tj. linky. Jakmile prohlížeč v dokumentu narazí v dokumentu na značku <hr>, zalomí tok textu a přes celé zobrazené okno vykreslí vodorovnou linku a dále pokračuje na novém řádku. Podobně jako u značky
 se v jazyce XHTML zapisuje značka vodorovného oddělovače do jedné značky <hr />.

3. 4. 8 Předem formátovaný text

Někdy potřebujeme zobrazit pouze určitý blok textu tak jak je. Připravíme si třeba speciálně odsazené řádky s vertikálně zarovnanými textovými nebo číselnými údaji, které se nesmí změnit ani při změně velikosti okna prohlížeče. Pro tyto případy slouží značka <pre>, která má i svoji koncovou značku. Veškerý text mezi těmito značkami se zobrazí přesně tak, jak jej zapišeme do zdrojového dokumentu.

3. 5 Hypertextové odkazy

Hypertext dává uživateli možnost načíst a zobrazit jiný dokument ze stejného nebo úplně jiného počítače klepnutím klávesnice nebo tlačítka myši na hypertextovým

odkazem. Hypertextový odkaz je speciálně vyznačené slovo nebo fráze v dokumentu. Zapsat do dokumentu HTML hypertextový odkaz na nějaký svůj vlastní dokument nebo na nějaký dokument uložený na serveru v Kanadě je velice jednoduché. Stačí vědět přesnou adresu požadovaného dokumentu a pak již stačí umět umístit do dokumentu takzvanou kotvu.

3. 6 Adresy URL

Adresu URL (Uniform Resource Locator) tvoří název dokumentu, před který se zapisuje hierarchie jmen adresářů, v nichž je uložen soubor, dále internetové doménové jméno serveru, na kterém je soubor uložena pak je zde ještě popsán software a způsob, jakým si prohlížeč vymění dokument s jeho hostitelským serverem, tedy protokolem.

Příklad:

```
protokol://doménové_jméno_serveru/jméno_cesty/  
http://lists.w3.org/Archives/Public/
```

3. 7 Kotvy

Značka kotvy <a> definuje v jazyce HTML a XHTML jak zdroj, tak i cíl hypertextového odkazu. Nejčastěji se značka <a> používá s atributem href, kde značka definuje zdrojový hypertextový odkaz a hodnotou atributu href je pak cílová adresa URL. Obsahem značky mohou být mimo adres také obrázky. Tedy vše co je uzavřeno do závorek tak představuje vyhrazenou část dokumentu, který v okně prohlížeče zobrazí jako aktivní a po jehož klepnutí myši se uživatel dostane na cíl hypertextového odkazu. Ten má obvykle jiný vzhled než zbytek dokumentu (text se vypisuje jinou barvou, nebo je podtržený, obrázky mají barevný okraj, případně se uplatní i jiné zajímavé efekty), a při průchodu nad ním se změní ikona ukazatele myši. Obsahem značky <a> by tak měl být text který intuitivně uživateli říká kam se může pomocí hypertextového odkazu dostat.

3. 7. 1 Kotvy mimo HTML a XHTML

Hypertextové odkazy nemusí ukazovat jen na jiné dokumenty HTML a XHTML. Pomocí kotev se můžeme odkazovat na libovolný typ dokumentu, jaké jsou na Internetu k dispozici, včetně ostatních internetových služeb.

Prohlížeče dokáží zpracovat různé internetové služby, jako je například FTP a GOPHER, takže uživatel může přímo v prohlížeči stahovat i dokumenty jiných formátů než HTML. Multimédia však plně a rozumně ošetřit neumí.

Multimédií existuje celá řada typů a formátů, existuje však pro ně jen málo standardů. Každý počítačový systém připojený k webu umí tyto zvukové a obrazové (filmové) formáty zobrazovat jiným způsobem. Standardní jazyk HTML ale nenabízí žádný prostředek pro zobrazování multimediálních dokumentů (s výjimkou některých grafických obrázků) a tak jediným prostředkem je zde možnost zápisu odkazu na multimediální obsah do kotvy. Prohlížeč, který načte určitý multimediální dokument, musí pro jeho dekódování a zobrazení přímo v rámci hlavního dokumentu vždy aktivovat speciální pomocnou aplikaci, stáhnout a spustit příslušný applet, nebo mít nainstalováno odpovídající plug-in příslušenství.

Jazyk HTML a většina webových prohlížečů tak v současné době uvedené zmatky fakticky neřeší, ale pouze je odsouvá do pozadí. Neznamena to ale, že by jsme neměli

multimédia v dokumentech využívat, právě naopak, ale mějme na paměti příslušná omezení.

3. 8 Obrázky

Obrázky jsou multimediální elementy, na které se v dokumentu můžeme odkazovat pomocí kotev, a prohlížeč je samostatně stáhne ze zadané adresy a zobrazí. Pro obrázky však na rozdíl od ostatních multimédií definují standardy jazyků HTML a XHTML explicitní mechanismus zobrazení jako obrázek inline (přímo vložený) s textem, nebo jako speciální mapy hypertextových odkazů. GIF a JPEG, jsou formáty obrázků, které jsou nejrozšířenější, a také plně podporovány v obou hlavních prohlížečích, ale podporují zobrazení také jiných přímo vložených multimédií. Microsoft Internet Explorer podporuje například značku <bgsound>, která dokáže na pozadí přehrát zvukový záznam. Standardy jazyků HTML a XHTML definují navíc způsob zobrazení ostatních typů přímo vložených multimédií v dokumentu, a to prostřednictvím speciální obecné značky.

3. 8. 1 Přímo vložené obrázky

Pro přímé vložení obrázků slouží v jazyce HTML a XHTML značka . Její povinný atribut src přebírá adresu URL obrázku ve formátu GIF nebo JPEG, který se má vložit do dokumentu. Prohlížeč tedy obrázky načte a umístí je do dokumentu jako by to byly speciální i když někdy dosti velké speciální znaky. Řečeno jinak to znamená, že prohlížeč zarovná spodní hranu obrázku se spodní hranou aktuální toku textu. Jestliže to chceme změnit, využijeme atribut align, který může nabývat hodnot top, middle a bottom, čili umístění k horní hraně, uprostřed a k dolní hraně přilehlého textu. Další možnost je samozřejmě umístit obrázek na samostatný řádek a oddělit na začátku i na konci značkami pro zalomení odstavce.

3. 8. 2 Obrazové mapy

Obrazové mapy jsou obrázky v kotvě se speciálním atributem, mohou totiž obsahovat více než jeden hypertextový odkaz.

Jedna možnost vytvoření obrazové mapy spočívá v doplnění značky , zapsané uvnitř kotvy <a>, o atribut ismap. Hned jak uživatel klepne někde v obrázku myší, odešlou se relativní souřadnice kursoru myši na server, kde se pomocí speciálního programu převedou do určité definované operace, jako například stažení jiného dokumentu.

Druhý postup spočívá ve vytvoření značek <map> a <area> a v doplnění značky o atribut usemap. Takto vložíme veškeré informace, které prohlížeč potřebuje ke zpracování obrazové mapy, přímo do stejného dokumentu jako samotný obrázek. To znamená, že klientská mapa neklade tak vysoké nároky na přenosovou kapacitu sítě a je nezávislá na serveru, což ocení zejména systémový administrátoři.

3. 9 Seznamy, vyhledávací dokumenty a formuláře

3. 9. 1 Neuspořádané, uspořádané a definiční seznamy

Značkovací jazyky znají celkem tři typy seznamů.

Neuspořádaný seznam je takový seznam, v němž pořadí položek není významný. V jazyce HTML a XHTML se zapisuje mezi značky `<ul>` a `</ul>`. Každou jednotlivou položku seznamu tvoří obvykle určité slovo nebo fráze a označuje ji značka položky seznamu `<li>` v jazyce XHTML doplněná navíc o koncovou značku `</li>`. Prohlížeč před každou položkou zpravidla vypíše symbol odrážky.

Uspořádané seznamy vymezují značky `<ol>` a `</ol>`. Jejich formát je úplně stejný jako formát neuspořádaného seznamu, tedy jednotlivé položky se v něm označují pomocí značky `<li>`, v jazyce XHTML doplněná navíc o koncovou značku `</li>`. Zde je pořadí položek významné, a proto prohlížeč zobrazí před každou položkou seznamu pořadové číslo s rostoucí posloupností.

Definiční seznamy, jsou o něco složitější než neuspořádané a uspořádané seznamy. Definiční seznam uzavírá dvojice značek `<dl>` a `</dl>`. Každá položka má ale dvě části, popsané speciální značkou. První část představuje značka `<dt>` a obsahuje zkrácené jméno nebo titulky položky, za ní následuje odpovídající hodnota nebo definice, uvozená značkou `<dd>`, v jazyce XHTML se opět uvádí odpovídající koncové značky `</dt>` a `</dd>`. Při zobrazení se v prohlížeči obvykle vypíše jméno položky na samostatný řádek, a definice, která může obsahovat i několik odstavců, se vypíše pod něj.

Ve standardech značkovacích jazyků můžeme dokonce vytvářet seznam v seznamu, neboli vnořené seznamy, které nám nabízejí obrovské množství zajímavých kombinací.

3. 9. 2 Formuláře

V dokumentu můžeme vytvořit jeden nebo více formulářů. Které vymezují značky `<form>` a `</form>`. Dovnitř formuláře pak umístíme předem definovaná nebo uživatelsky přizpůsobená vstupní textová pole, které umožňují zadávání jednoho nebo více řádků. Kromě toho může formulář obsahovat zaškrtačací políčka, nebo přepínač pro výběr jedné nebo více variant ze seznamu a dále speciální tlačítka, která vyresetují formulář, případně odešlou jeho data na server.

Formuláře znají všechno, co bychom od automatického formuláře očekávali. Mají tedy například popisky vstupních oblastí, integrované mechanismy pro instrukce, implicitní hodnoty vstupních údajů a tak dále. Jedinou výjimkou je automatická verifikace tj. ověření platnosti zadaných údajů, kterou musí provést příslušný program na straně serveru, případně applet na straně klienta.

3. 10 Tabulky

Jazyky HTML/XHTML podporují určitou skupinu značek pro datové tabulky, v nichž můžeme zarovnávat nejen číselné údaje, ale také speciálním způsobem formátovat text.

Tabulky popisuje v jazyce HTML a XHTML celkem pět značek, mezi ně patří samotná značka tabulky `<table>` a značka `<caption>` s popisem tabulky. Vzhled a rozměry tabulky určují různé speciální atributy značek. Tabulky vytváříme většinou řádek po řádku. Každý řádek je uzavřen mezi značku řádku tabulky `<tr>` a její koncovou značku `</tr>`. Pro každou buňku tabulky obsahuje značku záhlaví tabulky `<th>` nebo značku dat tabulky `<td>` a příslušný obsah. V jazyce XHTML se také uvádí odpovídající koncová značka. Hlavička i data tabulky mohou obsahovat téměř jakýkoliv běžný obsah, tedy text, obrázky, formuláře a dokonce i jiné tabulky. Z toho je vidět, že tabulky můžeme využít pro různé pokročilé formátování textu, jako například text o více sloupcích a podobně.

3.11 Rámečky

Rámečky nám umožňují rozdělení okna prohlížeče do několika oblastí pro zobrazení, z nichž každá obsahuje jiný dokument. A nemusí to být nutně jen dokument HTML či XHTML. Rámeček může obsahovat jakýkoliv platný obsah, který je schopen prohlížeč zobrazit, a to včetně multimediálního obsahu. Jestliže je součástí obsahu rámečku určitý hypertextový odkaz, může se obsah nového dokumentu nahradit v tom samém rámečku, nebo celém okně prohlížeče.

Rámečky se definují ve speciálním dokumentu, kde značku `<body>` nahrazujeme jednou nebo více značkami `<frameset>`. Tyto značky v podstatě prohlížeči říkají, jakým způsobem má svoje hlavní okno rozdělit do rámečků. Uvnitř značky `<frameset>` se pak zapisuje speciální značka `<frame>`, která ukazuje na dokumenty zobrazované uvnitř rámečků.

Jednotlivé dokumenty, na které se odkazujeme a které zobrazuje okno dokumentu s rámečky, mohou fungovat do jisté míry nezávisle. Dokumentu jednoho určitého rámečku tak lze nařídit aby načetl nový obsah i do jiného rámečku.

4 Dynamické HTML stránky

World Wide Web spustil počítačovou revoluci tím, že umožnil každému publikovat dokumenty HTML. Donedávna byly informace v těchto dokumentech většinou statické a vyžadovaly odpovědi serveru na vstupy uživatele. Po uvedení dynamického HTML (dále též DHTML) se tvorba těchto dokumentů přesunula od vyžadování interakce serveru do oblasti vytváření interaktivních položek sítě WWW a webových aplikací. Protože DHTML aplikace umožňující interakci uživatele s dokumenty HTML se zcela zaměřily na klienta, můžeme vytvářet bohatší webové aplikace než dříve. DHTML je postaveno na modelu objektu, který rozšiřuje tradiční statické dokumenty HTML.

Atributy DHTML plně fungují v Internet Exploreru 4.0 a výše, Netscape Komunikátor nepodporuje všechny prvky, které jsou uvedeny.

4.1 Vznik interaktivních stránek

Koncepce vytváření interaktivních stránek není nová – Microsoft i Netscape zpočátku definovali jednoduchý model objektu, který byl uváděn jako způsob vytváření interaktivních stránek. Při bližším zkoumání se ukázalo, že tento model objektu byl účinný pouze pro základní formu vyhodnocení. Před nástupem Microsoft Internet Exploreru 4.0 byly dynamické dokumenty jenom mýty, protože jediným okamžikem, kdy bylo možné stránku změnit byla její vlastní příprava. Interaktivní dokumenty byly simulovány úplným novým zavedením stránky nebo objekty vkládanými do stránky. Mnoho z takových appletů Java, ovládacích prvků ActiveX nebo dokonce i animovaných obrázků GIF bylo navrženo k tomu, aby prováděly jakoby manipulaci textu, což umožňovalo obejít přirozenou statickosti HTML. Využití objektů jako náhražky ovládání stylu a obsahu je však ubohým řešením.

Vývojáři Internet Exploreru vyzkoušeli mnoho scénářů pro takové objekty a animované soubory GIF. Poté si uvědomili potřebu výkonnějšího API (*Application Programming Interface* – rozhraní aplikačních programů) pro zpracování dokumentů. S modelem objektu nabízejícím úplný přístup k dokumentu mohou autoři využít výhod možností zpracování vzhledu, vestavěných v HTML, a CSS (*Cascading Style Sheets* – kaskádní nebo postupné styly). Tento model objektu výrazně zvýšil výkon tím, že omezil potřebu zavádění velkých animovaných obrázků nebo appletů. Informace se tak staly trvale přístupné, protože již nebyly skryty v obrázcích nebo objektech.

Dynamické HTML nabízí API, které je nezbytné pro celkovou kontrolu dokumentu HTML. Není již potřebné stránku při jejím zavádění definovat. Po zavedení může být kterákoliv část stránky dynamicky a okamžitě změněna. Například můžeme vytvořit aplikaci, která má rozšiřující se nebo naopak zmenšující se osnovu. Když uživatel osnovu rozšíří nebo zmenší, bude obsah okamžitě zobrazen nebo skryt. Jsou to možnosti přípravy stránek, které se automaticky mění nebo se samostatně upravují podle potřeb uživatele.

4.2 Definování dokumentu HTML/XHTML.

V dokumentech HTML/XHTML jsou tři významné momenty - struktura, styl a obsah. S pomocí dynamického HTML nyní HTML obsahuje čtvrtou komponentu - chování. Termín chování odkazuje na interakci mezi stránkou HTML a uživatelem. Struktura je vystavěna v sadě sbírek prvků, styl je vystavěn v každém prvku a ve sbírce stylů stránek, a obsah je tvořen každým prvkem a objektem TextRange. Aby vytvořily

chování dokumentu, zpracovávají skripty strukturu, styl a obsah jako odpověď na události.

4. 3 Programování dynamického HTML

4. 3. 1 Objektový model

K pochopení vytváření webových stránek plně vyžaduje porozumět architektuře IE 4.0. Tato architektura má podobu objektového modelu.

Většina aplikací jako je např. Microsoft Word, Microsoft Excel a Internet Explorer jsou složeny z objektů. Tyto aplikace jsou sestaveny z vhodně nadefinovaných softwarových modulů, s jejichž daty lze pracovat díky proměnným, známým jako vlastnosti (properties), a jejich služby lze využívat pomocí funkčních volání, známých jako metody (methods).

Vlastnosti a metody realizují užitečnou funkci pro určitou část softwaru. Kolekci vlastností a metod, jež definují určitý objekt, se říká rozhraní (interface). Rozhraní je vyvoláváno uživatelem pomocí nástrojových panelů a nabídek, může je však volat přímo programátor prostřednictvím programového kódu. Vše, co může provádět uživatel v aplikaci, může dělat i programátor, jenž provádí volání k objektům.

Objektové modely Microsoftu jsou založeny na technologii známé jako Component Object Model (COM), což je specifikace, v níž je definováno, jak objekty komunikují za účelem sdílení služeb. Sdílení služeb dodává objektovým modelům výkonnost a flexibilitu.

Základním objektem je window (představuje celý prohlížeč), přes který jsou dostupné všechny další objekty. Nejdůležitějším objektem je document, který v sobě zahrnuje všechny objekty a vlastnosti vztahující se k aktuálnímu dokumentu. Na jednotlivé objekty v hierarchii se odvoláváme pomocí tečkové notace:

Window.screen, window.document, window.document.body. Protože je objekt document nejpoužívanější, můžeme před ním vynechat specifikaci nadřazeného objektu window. Zkráceně tedy můžeme psát window.screen, document, document.body.

Základní hierarchie objektů

```
window
|--location
|--frames
|--history
|--navigator
|--event
|--screen
+--document
    |--links
    |--anchors
    |--images
    |--filters
    |--forms
    |--applets
    |--embeds
    |--plugins
    |--frames
    |--scripts
    |--all
```

|--selection
|--styleSheets
+--body

4. 3. 2 Vývoj hierarchie dynamického HTML

Následující souhrn uvádí vývoj podpory objektu v různých prohlížečích.

Internet Explorer 3.0 podporuje následující objekty:

- anchors – kotvy
- document – dokument
- document.frames – rámce dokumentu
- forms – formuláře
- history – historie
- links – propojení
- location – umístění
- navigator – navigátor
- window – okno
- window.frames -rámce okna

Netscape Navigator 3.0 podporuje stejné objekty jako Internet Explorer 3.0 kromě objektu document.frames, ale navíc ještě následující objekty:

- applets – aplety
- images – obrázky

Internet Explorer 4.0 podporuje stejné objekty, IE 3.0 a NN 3.0 a přidává k nim následující objekty:

- all –všechno
- body – tělo
- clientInformation – informace pro klienta
- event – událost
- screen – obrazovka
- scripts – skripty
- selection – výběr
- styleSheets – styl listů

Window představuje celý prohlížeč a adresujeme-li tento objekt ve skriptu, manipulujeme přímo s IE 4.0.

Location poskytuje informace o aktuální poloze na Internetu. Díky tomu lze zjistit adresu aktuálního místa, a to pomocí vlastnosti Href, nebo získat její části pomocí vlastností Hash, Host, Hostname, Pathname, Port, Protocol, Search.

Frames slouží k práci se všemi rámy v prohlížeči. Tato kolekce představuje nástroj pro programovou manipulaci s jedním rámcem ze skriptu, který běží v jiném rámci. Obvykle se s touto kolekcí pracuje pomocí čísel počínaje jedničkou, přístup k oknům můžeme také provádět pomocí jejich jmen (jsou-li k dispozici). Poté co nadefinujeme rámy uvnitř množiny ráků, můžeme pomocí kolekce Frames pracovat s kterýmukoli z existujících ráků.

Předpokládejme, že jsme nadefinovali tři rámy Toolbar, Banner a Main. S oknem Main můžeme z okna Toolbar manipulovat prostřednictvím:

Window.parent.frames("Main").navigate "adresa odkazu"

Navigator webová stránka s více rámy má pro každý rám zvláštní objekt Window, a proto je možné používat prohlížeč jako celek jen pomocí objektu Navigator. Event tento objekt odpovídá událostem, ke kterým dochází v Internet Exploreru. Zjišťuje značky na

které uživatel klepnul, identifikuje prvek který je do události zapojen, a obstarává šíření události. Je velmi důležitý pro dynamické HTML.

Screen představuje obrazovku klienta a vrací informace o jejích schopnostech. Nejdůležitější vlastnosti jsou Height (výška) a Width (šířka), které vracejí rozlišení obrazovky klienta v pixelech. Pomocí této informace pak lze přesně umístit prvky na webové stránce na základě rozlišení obrazovky.

Document představuje dokument aktuálně načtený v prohlížeči. Je to snad nejdůležitější objekt v hierarchii, protože funguje jako přístupová cesta ke všem součástem dokumentu, počínaje barvou pozadí konče každou jednotlivou značkou obsaženou v dokumentu. DHTML je značně závislé na přístupu ke značkám HTML skrze objekt Document.

Document umožňuje pracovat s mnoha prvky stránky, práce s těmito prvky se zpravidla provádí pomocí kolekce, zvláštního druhu objektu v objektovém modelu Internet Explorer. Kolekce představuje skupinu objektů stejné třídy a umožňuje snadný přístup ke každému členu takové třídy. Např. pomocí kolekce Form se můžeme dostat ke všem formulářům v dokumentu a provádět s nimi jejich funkce. Obecně platí, že s každou vlastností nebo metodou ovládacího prvku na formuláři je možné pracovat pomocí objektů Document a Form následujícím způsobem:

`Document.jménoFormuláře.jménoOvládacíhoPrvku.Vlastnost[metoda]`

Internet Explorer 4.0 umožňuje pracovat s kolekcemi u mnoha jiných objektů včetně Links, Anchors, Images, Filters, Forms, Applets, Embeds, PlugIns, Frames, Script a StyleSheet. Tyto kolekce nám umožňují pracovat s každým aspektem stránky HTML a jádrem Dynamického HTML.

Na dalších nižších úrovních v objektové hierarchii mohou být buď další objekty anebo přímo vlastnosti a metody jednotlivých objektů. Objekt `document.all` obsahuje všechny elementy obsažené v dokumentu. Podstata dynamického provádění změn HTML-obsahu spočívá v zacházení s každou značkou jako se samotným objektem. Z toho důvodu je zapotřebí značky pojmenovat, aby mohly být adresovány z kódu. Název značky je přístupný pomocí svého jména nadefinovaného atributem ID:

```
<H1 ID=Nadpis>nadpis</H1>
```

```
<P ID=Odstavec1>Nějaký odstavec
```

Jakmile je ID definováno, můžeme události, ke kterým může na značce dojít, přiřadit kód který napíšeme v některém ze skriptovacích jazyků. Dynamické HTML rozlišuje rozsáhlou řadu událostí, tyto události se zpracovávají na mnoha úrovních což umožňuje obsaženým prvkům obdržet oznámení o tom, kdy značky v nich spouštějí události. Tento proces kdy se objektům oznamují události, ke kterým v nich došlo, je znám jako šíření událostí (event bubbling).

Díky šíření událostí můžeme určit v jednom místě ošetřovací událostní rutinu pro většinu značek na webové stránce a pro kteroukoli značku specifikovat určité chování. K šíření dochází automaticky a lze je odchytnout každým objektem v hierarchii. Když u značky dojde k nějaké události, objekt Document tuto událost automaticky získá po ukončení jejího ošetření značkou.

Nyní můžeme ve skriptech po pojmenování ID na stránce používat objekty `document.all.Nadpis` a `document.all.Odstavec1`, které zastupují element pro nadpis a odstavec. U těchto objektů máme k dispozici vlastnosti, které odpovídají atributům použitelným u odpovídajících elementů.

U objektů, které odpovídají elementům stránky, máme k dispozici vlastnost i objekt `style`. Pomocí vlastnosti `style` můžeme přistupovat k definici stylu elementu stejně jako pomocí atributu `style`. Následující kód by změnil barvu a velikost nadpisu:

```
document.all.Nadpis.style = "color: red; font-size: 30px";
```

Zajímavé je, že style je zároveň vlastnost i celý objekt. Vlastnosti objektu style odpovídají jednotlivým vlastnostem stylů. To samé tedy můžeme dosáhnout i následujícím kódem:

```
document.all.Nadpis.style.color = "red";  
document.all.Nadpis.style.fontSize = "30px"
```

Location udává adresu dokumentu, který je právě zobrazován. Pokud tedy chceme přímo skriptem změnit stránku, která se právě zobrazuje, můžeme použít příkaz

```
location="http://www.xxx.cz".
```

Pokud máme stránku rozdělenou na rámy, můžeme z jakéhokoliv rámu měnit stránku zobrazenou v ostatních rámech. Stačí novou adresu zapsat do `parent.jméno_rámu.location`. Tímto způsobem můžeme dosáhnout speciálních efektů, jako je změna obsahu více rámu najednou po zvolení jednoho odkazu. Vždy bychom však měli zvážit, zda je chování stránky snadno pochopitelné pro její uživatele.

4. 3. 3 Dynamické obtékání

Vždy, když skript zpracuje a změní atribut prvku nebo stylu listu nebo upraví obsah, dokument inteligentně přepočítá a překreslí stránku s využitím nových informací. Model dokumentu dynamického HTML přináší odraz dokumentu v jazyce skriptu. Skript může například změnit atribut CLASS v každém prvku HTML. V jazyce skriptu je atribut CLASS, podobně jako všechny atributy, vystaven jako vlastnost prvku (v tomto případě jako vlastnost `className`).

4. 3. 4 Příprava skriptů

Skripty nejsou jedinou cestou pro přístup k modelu objektu dynamického HTML. Ten může být proveden třemi způsoby:

Pomocí skriptů ve stránce HTML nebo touto stránkou odkazovaných. Tyto skripty mohou zpřístupnit obsah aktuálního dokumentu nebo dokumentů ze stejné domény v jiných rámcích nebo oknech.

Pomocí vkládaných appletů a ovládacích prvků, které jsou na stránce rezidentní.

Pomocí hostitelů, kteří jsou mimo nebo vedle prohlížeče. Například pomocí dynamického HTML bylo vytvořeno dialogové okno Find v Internet Exploreru 4.0.

4. 4 Prvek script

Uzavírá do sebe skripty a určuje některé jejich vlastnosti nejčastěji jsou používány skripty vytvořeny ve Visual Basicu, JavaScriptu či Tcl.

```
<SCRIPT TYPE="typ" LANGUAGE="jazyk" SRC="soubor_se_skriptem"  
TITLE="titulek_skriptu" EVENT="událost" FOR="objekt"> předávaná data - nejsou  
zobrazována ani formátována, nejsou povinná, každý skript nemusí přebírat vstupní  
data </SCRIPT>
```

TYPE="typ" Nastavuje typ skriptu (TYPE=text/JavaScript).

LANGUAGE="jazyk" Je nedoporučován, jeho místo zaujímá právě TYPE. SRC="url" Soubor obsahující vlastní skript (knihovny skriptů). Toto odkazování na jiný soubor zvyšuje výkon a běžné funkce nemusejí být zapsány na každou stránku. TITLE="titulek_skriptu" Popisující titulek skriptu.

EVENT="událost" Určuje události typu intrinsic. Lze také specifikovat implicitní jazyk pro všechny skripty na stránce, aby nebylo nutné jej pokaždé vypisovat, a to v tagu META, typu HTTP-EQUIV:

<META http-equiv="Content-Script-Type" content= jazyk_skriptu>

Při vkládání skriptů je rovněž vhodné jejich obsah skrýt před staršími prohlížeči pomocí komentářů. Před ukončením HTML komentáře --> použijeme dvě lomítka, aby se interpreter JavaScriptu nepokoušel konec HTML komentáře považovat za část skriptu. To dělá zejména Netscape Navigator.

4. 4. 1 Umístění skriptů v dokumentu

Dokument může obsahovat libovolný počet prvků Script, který může pracovat v hlavičce dokumentu, nebo v těle (pro většinu účelů není umístění ve vztahu k návrhu stránky důležité). Umístění nabývá důležitosti tehdy, když prvek skutečně zapisuje obsah do proudu nebo je odkazován na prvek v dokumentu. Zápis do proudu se provádí pomocí metody write nebo writeln objektu document:

```
<HTML>
<SCRIPT LANGUAGE="JavaScript">
Tímto skriptem je generován celý dokument:
document.write("<HEAD><TITLE>Můj dokument</TITLE></HEAD>");
document.writeln("<BODY><H1>Má stránka</H1></BODY>");
</SCRIPT>
</HTML>
```

Tento program vytváří a obnovuje následující HTML:

```
<HTML>
<HEAD>
<TITLE>Můj dokument</TITLE>
</HEAD>
</BODY>
<H1>Má stránka</H1>
</BODY>
</HTML>
```

4. 4. 2 Odložení provedení skriptu

IE 4.0 může nabídnout vylepšený výkon pro skripty, které neobsahují bezprostředně prováděný program. Pokud prvek Script obsahuje pouze deklaraci funkcí, doplnění atributu DEFER do tagu <SCRIPT> upozorní prohlížeč, že nemusí čekat na celý skript, aby byl analyzován a přeložen.

4. 4. 3 Více jazyků skriptů

Při používání Internet Exploreru může být připravena stránka, obsahující jak program v JavaScriptu, tak program ve VBScriptu. Dále je pro jeden jazyk možné volat funkce definované jiným jazykem. Volání funkce, zapsané ve skriptu v jiném jazyce, je možno proto, že všechny funkce a proměnné jsou přidány objektu window jako metody a vlastnosti.

```
<SCRIPT LANGUAGE="VBScript">
```

Tento skript vysune rámeček hlášení:

```
sub MyAlert(str)
msgBox(str)
end sub
</SCRIPT>
```

```
<SCRIPT LANGUAGE="JavaScript">
Volání MyAlert, definované v předchozím příkladě
MyAlert("Hello World!");
Window.MyAlert("MyAlert je metoda objektu window.") </SCRIPT>
```

4. 4. 4 Přesměrování na straně klienta

Jednou metodou pro zpracování různých softwarových verzí je přesměrování uživatele podle jím používaného prohlížeče na různé stránky. Přesměrování na straně klienta proběhne tehdy, když skript na pozadí stránky podmíněně přepne prohlížeč na jiný dokument. Pomocí podmíněného testování prohlížeče může být zavedena alternativní verze stránky. Při použití této techniky musí být základní stránka tou, na níž se zaměří vaše nejdůležitější návštěva, protože přesměrování bude mít důsledek na zobrazení. Když se objeví přesměrování, vede to k tomu, že budou zavedeny dvě stránky.

```
<SCRIPT LANGUAGE="JavaScript">
var MS = navigator.appVersion.indexOf("MSIE");
Zkontroluje, zda se jedná o IE4:
window.isIE4 = (MS > 0) &&
(parseInt(navigator.appVersion.substring(MS+5, MS+6)) >= 4);
if (!isIE4) // Nejde-li o IE4. dostaneme nedynamickou stránku.
Window.location="downlevelpage.html";
</SKRIPT>
```

4. 4. 5 Výběr jazyka skriptu: JavaScriptu a VBScript

Model objektu dynamického HTML je jazykově neutrální a může být skriptem zpracován v libovolném dostupném programovacím jazyce. V současné době existují dva základní jazyky pro skripty webových stránek: JavaScript a VBScript.

4. 5 Události typu intrinsic

Aby byly naše dokumenty opravdu interaktivní, je potřeba, aby provedení určité části skriptu bylo vyvoláno událostí, kterou způsobil uživatel. Událostí může být přejetí myši přes určitý element, kliknutí myši apod.

onload=script Událost nastane, když prohlížeč načte rám nebo celé okno (tj. veškerá jeho data.). Podle toho, zda se jedná o jeden rám nebo všechny rámy, se tento element umísťuje do oblasti BODY nebo FRAMESET.

onunload=script Událost nastane, když je ukončeno zobrazení celého okna nebo rámu. To nastane, když uživatel zadá načtení jiného okna nebo rámu v okamžiku, kdy se okno se starým obsahem vyprázdní a je započato načítání druhého okna. Opět se používá v oblasti BODY nebo FRAMESET.

onclick=script Událost nastává v okamžiku, když uživatel klepne (klikne, stiskne levé tlačítko myši) nad elementem. Může se jednat i o text a postačí klepnutí nad kterýmkoliv písmenkem textu, není nutné cokoli vybírat (samozřejmě může jít také o obrázek, formulářový element, atd.).

ondblclick=script Totéž jako v předchozím případě, ale zde je nutný dvojitisk (doubleclick, poklepání).

onmousedown=script Podobně jako onclick, ovšem událost nastane, když uživatel drží nad elementem stisknuté levé tlačítko myši; jakmile jej uvolní, událost je ukončena.

onmouseup=script Tato událost nastane, když je tlačítko uvolněno, ovšem kurzor se vyskytuje stále na elementem.

onmouseover=script Událost nastane, ihned v okamžiku, když se kurzor přesune nad element (není nutné stisknout tlačítko).

onmousemove=script Obdobně jako výš uvedená událost: nastává, když se kurzor pohybuje, jakmile se pohyb zastaví, je událost ukončena.

onmouseout=script Doplňková událost k onmouseover - nastává v okamžiku, kdy je kurzor umístěn mimo element.

onfocus=script Událost nastává, když je element tzv. vybrán. Tato událost nemůže nastat u všech HTML elementů a týká se pouze elementů formulářových. Vybrání nastává například v okamžiku, když se pohybujete po dialogovém panelu pomocí klávesy TAB. Okolo vybraného elementu je vykreslen velmi tenký rámeček a element je možné ovládat klávesnicí (mezerník, kurzorové šipky). Lze tedy použít pouze u elementů LABEL, INPUT, SELECT, TEXTAREA a BUTTON.

onblur=script Komplementární událost k onfocus: dá se říci, že pokud není element "onfocus", tak je "onblur". Lze také použít pouze u tagů uvedených v předešlé události.

onkeypress=script Událost nastává, když je nad elementem stisknuta a uvolněna klávesa. Lze použít pouze u elementů, které se ovládají pomocí kláves (tj. opět například u formulářových elementů).

onkeydown=script Obdobně jako u předchozího elementu: událost nastává, když je klávesa stisknuta a přidržena.

onkeyup=script Doplňková událost k předchozí: událost nastává, když je klávesa uvolněna poté, co byla předtím nad elementem stisknuta.

onsubmit=script Událost nastává, když je formulář odeslán ke zpracování; je jasné, že může být proto použita pouze u elementu FORM.

onreset=script Událost nastává, když je formulář resetován, tj. když je stisknuto tlačítko reset <BUTTON type=reset>, opět je samozřejmě platný pouze pro tag FORM.

onselect=script Událost nastává, když uživatel vybere text v políčku, tj. když přes něj "přejede" s přidrženým tlačítkem myši či kurzorovými klávesami s přidrženou klávesou SHIFT. Lze použít u tagů INPUT a TEXTAREA.

onchange=script Jako výše, ovšem v případě, když je text ve vstupním poli změněn, a když uživatel editaci ukončí (nastává tedy v momentě, kdy uživatel překlepne první písmenko v políčku, ale až v okamžiku, kdy přenese fokusaci na jiný element). Lze použít u tagů INPUT, SELECT a TEXTAREA.

4. 5. 1 Atributy události

V dynamickém HTML byly všechny prvky dokumentu rozšířeny, aby podporovaly události klávesnice a myši. Tyto události jsou na každém prvku vytvořeny přímo jako atributy a umožňují přímé spojení mezi prvkem a chováním (podobná spojení jako u prvku a jeho stylem). Můžeme provádět událost tlačítka onclick k funkci, a to pomocí následujícího atributu:

...když uživatel klepne na tlačítko, bude volána funkce foo()...

```
<INPUT TYPE=button VALUE="Klikni zde"  
ONCLICK="foo();" LANGUAGE="JavaScript" >
```

Atribut ONCLICK může buď volat funkci, nebo přímo provést jeden nebo více řádků programu.

...po klepnutí na tlačítko se zobrazí hlášení a potom se zavolá funkce foo()...

```
<INPUT TYPE=button VALUE="Klikni zde"  
ONCLICK="alert('Uživatel kliknul zde.');" foo();" LANGUAGE="JavaScript" >
```

Všechny atributy HTML nejsou citlivé na velikost písmen, citlivost na velikost písmen může být důležitá tehdy, použijeme-li jiný mechanismus vazby události (tj. spojení mezi určitou událostí a skriptem). Vazba události pomocí atributů HTML je vhodná, má ale celou řadu nevýhod. První nevýhodou je že jazyk HTML musí být rozšířen pokaždé, když bude vymyšlena nová událost. Také objekty nebo aplikace, vytvářejí libovolné události, také potřebují jazyk rozšířit nebo vytvořit své vlastní techniky vázání události. Proto je tento přístup využíván jen pro malou sadu vestavěných událostí. Pokud je na stránku vložen libovolný prvek, je jeho událost vytvořena více generickým způsobem

4. 5. 2 Podpora generické události

Pro vazbu funkcí k událostem využívá několik rozšíření prvku SCRIPT – atribut FOR a EVENT.

EVENT – odkazuje na událost a jakýkoli parametr, který do něj může být vložen.

FOR – určuje název nebo ID prvku, pro který je událost napsána.

Například v dokumentu je vytvořena událost *onmousemove*, pro vazbu k této události můžeme použít následující tag SCRIPT:

```
<SCRIPT FOR="dokument" EVENT="onmousemove()"
LANGUAGE="JavaScript">
```

- tento ovladač události je volán pokaždé, když se myš pohybuje po dokumentu

Poznámka: JavaScript je v tagu <SCRIPT> citlivý na velikost písmen jak pro hodnotu EVENT tak pro atribut FOR. Hodnoty musejí být zapsány malými písmeny.

4. 5. 3 Časování vazby události

V JavaScriptu jsou události při zavádění stránky spojovány asynchroně. Každý prvek Script a atribut události je zapojen tak, že je z dokumentu oddělen. VBScript události nesváže, dokud není analyzována celá stránka, nejsou zavedeny všechny externí skripty a nezačaly být zaváděny vložené objekty. V JavaScriptu to znamená, že události mohou začít odpovídat uživateli nebo na jinou akci ještě před tím, než je zavedena celá stránka.

4. 6 Skripty jednotlivých prvků

Objekty prvků mají několik vlastností, které usnadňují jejich identifikaci. Vlastnosti objektu id a className obsahují hodnoty odpovídajících atributů prvku ID a CLASS a jeho vlastnost, tagName obsahuje název tagu prvku. Prvky které mají atribut NAME, mají rovněž vlastnost name, která obsahuje hodnotu atributu.

4. 6. 1 Prvky Body a Frameset

Dokument HTML může obsahovat oba dva typy obsahu: obsah těla nebo definici sady rámců. V obou případech platí pro dokument podobný model objektu. Prvek Body nebo Frameset je také obsažen ve sbírce dokumentu all, proto vlastnost body může být zpřístupněna přímo z dokumentu.

vrací BODY nebo FRAMESET v závislosti na typu dokumentu
document.body.tagName

Abychom určili, zda dokument obsahuje tělo nebo sadu ráků, můžeme použít vlastnost tagName. Následující program zobrazí hlášení které oznámí typ dokumentu (v tomto případě sadu ráků).

```
<HTML>
<HEAD>
<TITLE>sada ráků vyjádřená jako BODY</TITLE>
</HEAD>
<FRAMESET ROWS="100%"
ONLOAD="alert (document.body. tagName);">
<FRAME SRC="neco.html">
</FRAMESET>
</HTML>
```

4. 6. 2 Události okna

Prvky Body a Frameset obsahují atributy odpovídající všem událostem na úrovni okna. Dokument může obsahovat více sad ráků, může mít pro každou událost okna jen jeden ovladač. Pokud několik prvků Frameset v dokumentu definuje pro událost ovladač, bude proveden pouze ten poslední program ovladače.

Událost onresize se spouští tehdy, když dojde ke změně velikosti fyzického okna. Je také definována v prvcích dokumentu, které mají definovanou velikost, v těchto případech se událost spouští, když se změní fyzická velikost prvku.

Událost onscroll se spouští kdykoliv, když je okno rolováno. Tato událost se vyskytuje jen v dokumentech s prvkem BODY.

4. 6. 3 Programování obsahu sady ráků

Dokument sady ráků obsahuje sbírku all, která zajišťuje přímý přístup ke všem prvkům v dokumentu. Pomocí sbírky all mohou být zpřístupněny jednotlivé atributy každého prvku Frameset a prvku Frame a v mnoha případech mohou být dynamicky upraveny.

4. 6. 4 Programování prvku A (Anchor) a Base HREF

Prvek A (Anchor) slouží ke dvěma účelům: aby specifikoval propojení pro navigaci k URL a nebo specifikoval záložky v dokumentu

Mezi hodnotami Href a modelem objektu existuje vztah – částečně HREF je URL, které se explicitně nespouští příkazem (href=home.html). Všechny Href jsou označeny předdefinovaným umístěním. Pokud není zadáno jinak, je předdefinovaným umístěním umístění aktuálního dokumentu. Ke změně předdefinovaného umístění můžeme použít prvek BASE. Pro částečné URL, přiřazených k atributům, bude předdefinované umístění přidáno při analýze dokumentu. Pro částečné URL, přiřazené k vlastnostem, nebude předdefinované umístění přidáno do té doby, pokud nebude URL odkazováno. Atribut Href zadaný skriptem je užitečný tehdy, když rám obsahuje krátký řetězec, protože se pak snižuje počet spojení se serverem. Např. když uživatel klepne na následující kotvu, vytvoří se jednoduchý dokument, který zobrazí řetězec Hello World.

```
<A HREF="JavaScript:'Helo word'">
```

Atributy zadané skriptem běží poté, když se ukončí spuštění události onclick.

Tag A (Anchor) podporuje sadu standardních událostí, které nastanou poté, kdy uživatel klepne, přejede myší přes nebo ji do kotvy zapíše. Události mohou mít počátek

v prvku A podle toho, zda je kotva propojením nebo záložkou. Všechny události mohou být definovány jako atributy v prvku, syntaxí <SCRIPT FOR= EVENT= > nebo pomocí vlastností prvku Anchor. Program přerušuje předdefinovanou akci určitého propojení:

```
<A HREF="foo.ht#100" ONCLICK=event.returnValue=false;">
```

Aby byla zaručena kompatibilita s jinými prohlížeči, můžeme vrátit hodnotu přímo ONCLICK="return false". Kotvy mohou být na úrovni dokumentu přepsány zpracováním události dokumentu onclick.

4. 6. 5 Programování prvků IMG a MAP

Obrázky a obrázkové mapy jsou v Internet Exploreru 4.0 plně programovatelné. Dnes můžeme změnit atribut SRC a změnit velikost obrázku a z obrázkové mapy Area upravovat, přidávat a odstraňovat. Model objektu umožňuje, aby nové obrázky byly do pozadí zaváděny asynchronně, zatímco uživatel pracuje se stránkou.

Animace obrázku:

Jedna nejběžnější technika pro animaci obrázku je změna obrázku při vstupu myši na prvek nebo když myš prvek opustí. Toto vytvoříme pomocí událostí onmouseover a onmouseout:

```
<IMG SRC="start.gif"
ONMOUSEOVER="this.src = 'over.gif';"
ONMOUSEOUT="this.src = 'start.gif';">
```

Sekvence obrázku:

Pro změny obrázku může být namísto uživatelsky generovaných událostí využit časovač událostí. Dynam. HTML zjednodušuje vytvoření sekvenceru obrázků, který obrázky v daném čase mění. Obrázky mohou být pomocí vzácného konstruktéru obrázku předem zavedeny a atribut SRC prvku IMG může být dynamicky změněn. Internet Explorer 4.0 podporuje konstrukci nového obrázku pro zavádění v pozadí a využívá při tom operátor new jako doplněk metody createElement. Tento operátor je podporován proto, aby byla zajištěna kompatibilita s implementacemi JavaScriptu Netscape Navigatoru. Operátor new je technika vytváření nových prvků, která je jazykově nezávislá.

Mapy obrázků:

HTML přináší dva typy map obrázků: mapy na straně serveru a mapy na straně klienta. Mapa na straně serveru je určena pouhým přidáním atributu ISMAP k obrázku a vytvořením souboru mapy obrázku na serveru. Mapy obrázku na straně klienta využívá prvek Map a jejich výhodou oproti mapám na straně serveru je, že nevyžadují opakovaný přístup na server.

Mapy obrázků a události:

Mapu můžeme umístit kamkoli do dokumentu. Protože může více obrázků sdílet jedinou mapu obrázku, umožňuje model objektu dynam. HTML při spuštění události mezi obrázkem a jeho mapou vztah. Když dojde ke spuštění události v mapě obrázku, přijímá událost prvek Area, následován prvkem Map a prvkem IMG, na který uživatel klepnul. Potom, když událost přijme obrázek, pokračuje probublávání události přes rodičovské prvky obrázku. Tudíž mohou být sdíleny jediná mapa obrázku a události, nebo v závislosti na okolnostech, může obrázek sám přepsat nebo přidat své vlastní chování do mapy obrázku. Prvky, které ve zdroji HTML obsahují mapu obrázku, nesmí nikdy přijímat události, které pocházejí z mapy obrázku.

Zpřístupnění mapy:

Vlastnost prvku IMG useMap obsahuje název spojené mapy obrázku, která má předponu znak #. Odstraněním předpony # z vlastnosti useMap můžeme mapu

zpřístupnit. Vlastnost useMap je v režimu čtení/zápis, což umožňuje, aby mapy obrázků s ním byly dynamicky spojeny.

```
Function getMap(ellImage) {  
  //Zajistíme, aby pro daný obrázek byla zadána mapa.  
  if (null !=ellImage.useMap) {  
    //Odstraní předponu # ze záložky.  
    var strMap = ellImage.useMap.substring(1);  
    //Vrací prvek se zadaným názvem  
    return document.all[strMap];  
  }  
  else  
    return null; }  

```

Zpřístupnění prvků Area:

Dynamické HTML obsahuje prvky Area ve sbírkách: Links, All, Areas v prvku Map. Proto, aby bylo možno atributy prvku Area dynamicky upravit, je mohou skripty zpřístupnit kterýmkoliv z těchto přístupů. Prvek Area má atribut Href, který obsahuje Url, a ten obsahuje stejné vlastnosti, obsahující části Url, jaké mají objekty location a anchor. Sbírka areas nabízí novou funkci, která novým prvkům Area umožňuje, aby byly k mapě obrázku přidány nebo z ní odstaněny. Sbírka Areas má metody add a remove. Metoda add používá prvek Area, vytvořený metodou createElement, a přidává jej ke sbírce areas. Metoda remove se používá k odstranění existujícího prvku Area z mapy.

4. 6. 6 Programování prvku Object

Object může mít vlastnosti, metody a události, které mohou být použity ve skriptech stejným způsobem, jako to provádějí jeho vlastní členové. Mezi členy objektu a členy generického prvku Object může vzniknout konflikt vlastností. Např. obsahuje-li objekt vlastnost ID, bude tato vlastnost kolidovat s vlastností ID, která se vyskytuje v prvku Object. Když k takové kolizi dojde, odkazování vlastnosti ID ukazuje na verzi prvku, nikoli objektu. Při odkazování verze objektu vlastnosti ID mají všechny prvky vlastnost object. Tato vlastnost vrací přístup ke vloženým členům objektu.

Document.all.mujObject.id //Vlastnost id prvku HTML.

Document.all.mujObject.object.id //Vlastnost id vloženého prvku.

Objekt může spouštět své vlastní události. Můžeme provázat ovladač tak, aby událost používala syntaxi <SCRIPT FOR= EVENT= > nebo mechanismus nezávislý na jazyku, ale nepoužívala atribut v tagu prvku.

4. 6. 7 Programování prvku Table

Tabulky obsahují řádky, každá řádka obsahuje libovolný počet buněk. Dynamické HTML používá v tabulkách vlastní model objektu, který zajišťuje snadný přístup k podřízeným řádkům tabulky.

Objekt Table zajišťuje přístup ke třem různým částem tabulky: THead, TBody a TFoot. Tabulka může mít jenom jeden prvek THead a Tfoot, ale libovolný počet prvků TBody. Proto model objektu obsahuje jedinou vlastnost tHead a tFoot a sbírku tBodies.

Objekt table přináší metody pro vytváření a odstraňování prvků THead, TFoot a Caption:

CreateTHead(), CreateTFoot(), CreateCaption() Vytváří a vrací určitou část, pokud tato část neexistuje.

DeleteTHead() Pokud v tabulce daná část existuje, vymaže z tabulky tuto určenou část a její řádky.

InsertRows([index]) Vkládá do tabulky řádek před daný index. Není-li zadán index, bude řádka přidána na konec tabulky.

DeleteTFoot(),DeleteCaption(),DeleteRow(index) Odstraňují z tabulky určité části.

Objekt table obsahuje vztahy mezi řádky a buňkami tabulky. Sbíрка rows v objektech tHead, tBody a tFoot obsahuje prvky TR v jejich odpovídajících částech. Každá řádka následně přináší sbírku cells, která odkazuje v řádce prvky TD a TH. Vlastnost Id prvku můžeme využít pro její přímé provázání ve sbírce rows nebo cells.

4. 6. 8 Skripty prvku Form

Formuláře v dokumentu jsou odhalovány pomocí sbírky all a sbírky forms. Následující program demonstruje několik cest, jak zpřístupnit prvky Form pomocí modelu objektu (komentáře popisují hlášení, která budou zobrazovány v dialogových oknech Alert):

```
<HTML>
<HEAD>
<TITLE>Formuláře v modelu objektu</TITLE>
</HEAD>
<BODY>
<FORM NAME="form1">
</FORM>
<FORM NAME="form2">
</FORM>
<SCRIPT LANGUAGE="JavaScript">
alert(document.forms.lenght); // 2
alert(document.forms[0].name); // form1
alert(document.forms.form2.name); // form2
alert(document.form1.name); // form1
alert(document.all.form2.name); // form2
alert(document.forms["form1"].name); // form1
</SCRIPT>
</BODY>
</HTML>
```

Můžeme také použít sbírku elements, která umožňuje přístup ke každému ovládacímu prvku, který ve formuláři existuje (jako sbírka forms).

Document.forms[0].lenght // Počet prvků na prvním formuláři.

Document.forms[0].elements.lenght

Documnet.forms[0].elements[2] //Zpřístupní třetí vnitřní ovládací prvek ve formuláři.

4. 6. 9 Programování prvku Marquee

Prvek Marquee podporuje všechny standardní události myši a klávesnice. Všechny prvky obsažené v Marquee rovněž pokračují ve spoštění individuálních událostí.

onStart Začíná rolovat, v režimu scroll nebo slide spouští tato událost pokaždé novou sekvenci animace.

onMouse - Tato událost se spouští jen tehdy, když je vlastnost behavior nastavena na alternate.

onFinish - Když Marquee dokončilo rolování.

Metody má prvek dvě, pro spuštění a zastavení animace: start a stop. Použitím těchto metod vytvoříme program, který umožní, aby uživatel zastavil marquee stisknutím a uvolněním tlačítka myši, když se myš nachází v prvku.

```
<HTML>
<HEAD>
<TITLE>Metody prvku Marquee</TITLE>
</HEAD>
<BODY>
<MARQUEE TITLE="Pro zastavení podržte stisknuté tlačítko myši."
onMouseDown="this.stop();" onMouseUp="this.start();">
<H1>Test</H1>
<P>Klepnutím na tlačítko myši a jeho podržení jako stisknuté zastaví
rolování.</P>
<INPUT TYPE=button
VALUE="Demo button"onClick="alert('zmáčknuto');">
</MARQUEE>
</BODY>
</HTML>
```

5 CSS (Cascading Style Sheets)

CSS (kaskádní styly) je jazyk obsahující sadu vlastností pro definování vzhledu dokumentu. Specifikace CSS definuje vlastnosti a jazyk deklarací pro spojení těchto vlastností s prvky dokumentu. Jazyk CSS a jeho popisné normy vznikají v pracovních skupinách W3C.

Co tedy vlastně očekávat od CSS? Protože se jedná o popis vzhledu dokumentu je vcelku jasné, že kaskádní styly nejsou náhradou HTML. Jejich použitím pouze HTML kód doplníme a při vhodném použití též zjednodušíme a zefektivníme. Kaskádní styly si lze představit jako oddělený popis jednotlivých struktur od místa jejich použití. Do jaké míry se jedná o oddělenou deklaraci záleží na technologii definice kaskádních stylů. Od těchto technologií se také odvíjí efektivita výsledného kódu. Výraz kaskádní v CSS vyjadřuje schopnost slučovat jednotlivé styly k vytvoření podrobné definice stylu pro prvek či naopak vytvoření obecných pravidel pro prvky celého dokumentu.

5.1 Základní koncepce CSS

Kaskádní styly umožňují nastavit každému elementu HTML dokumentu svůj specifický styl. Pokud si tedy pod pojmem představíme grafický vzhled jednotlivých elementů objeví se nám široká škála nejrůznějších vlastností (pro písmo např. druh, velikost, barva, atd.). A právě k popisu těchto vlastností slouží CSS. Jednotlivé vlastnosti deklarujeme pomocí jednoduchého zápisu tohoto tvaru při čemž syntaxe kaskádních stylů není Case senzitivní (na velikosti písmen nezáleží):

Příklad:

```
element {deklarace}  
p {font-color:#000000; background:#ff0000;}
```

Tento popis je proveden více či méně odděleně od použití popisovaného elementu a to podle zvolené technologie vložení stylu. Tyto technologie existují tři. In – line styly, globální styly a propojené styly.

In – line styly by měly být používány pouze pro jeden výskyt daného elementu a jsou definovány přímo v tagu. In – line styl je definován atributem STYLE a informace za tímto atributem jsou specifikována pomocí jazyka CSS (toto je jediná výjimka kdy nedochází k zápisu pomocí předchozí deklarace).

Příklad:

```
<p style="font-color:#000000; background:#ff0000;">  
Odstavec s černým textem na červeném pozadí.  
</p>
```

Tato technologie vkládání CSS nedodrží ducha strukturovaného dokumentu. Její použití se doporučuje pouze v případě základního osvojení vlastností CSS či při potřebě rychlé změny jednoho elementu.

Globální styly se již doslovně řídí koncepcí CSS a proto se jejich umístění provádí odděleně od přímého použití konkrétních elementů. Jejich definice se obvykle udává v hlavičce HTML (může být i jinde) za tagem <STYLE>.

Příklad:

```
<html>
```

```

<head>
<style type="text/css">
<!--
p {font-color:#000000; background:#ff0000;}
-->
</style>
</head>
<body>
<p>Všechny odstavce v dokumentu budou napsány černým textem
s červeným pozadím.
</p>
</body>
</html>

```

Atribut TYPE v tagu <STYLE> definuje jazyk stylu listu jako typ MIME. Standard typu MIME umožňuje přenášet prostřednictvím služeb sítě Internet zprávy, v jejichž těle jsou i jiné než čisté ASCII texty. Prohlížeče v současné době podporují jako typ pouze styl listů CSS proto je třeba do atributu TYPE uvést text/css jinak by byl styl ignorován. Atribut TYPE může být taktéž vynechán protože implicitní nastavení v prohlížečích u tagu <STYLE> je právě CSS. Za tagem <STYLE> v předchozím příkladě následuje tag pro komentář. Ten je používán pro skrytí před staršími verzemi prohlížečů, které kaskádní styly nepodporují.

Propojené styly jsou nejefektivnější technologií vkládání kaskádních stylů a to obzvláště v kontextu rozsáhlejší webové prezentace. Jde o takový styl, který je uložen v externím souboru a pomocí tagu <LINK> vkládán do HTML dokumentu. Obrovskou výhodou je možnost sdílení tohoto souboru více dokumenty. To zefektivňuje prohlížení stránek, protože soubor se stylem je nahráván do paměti pouze jednou. Zároveň toto řešení značně usnadňuje modifikaci. Při změně v souboru stylu dojde ke změně na všech stránkách (ocení obzvláště webový programátoři při ladění velkých projektů).

Příklad:

```

<html>
<head>
<link rel="stylesheet" type="text/css" href="styl.css">
</head>
<body>
<p>Tento odstavec bude naformátován dle stylu pro element <p>
definovaném v souboru styl.css.</p>
</body>
</html>

```

V tagu <LINK> je potřeba nastavit atribut REL na „stylesheet“ což definuje že propojený soubor je styl listu. Atribut TYPE je shodný s globálními styly a atribut HREF definuje cestu a název propojeného souboru.

Externí soubor se stylem musí obsahovat pouze kontextuální pravidla a definice stylu a nesmí obsahovat žádný HTML kód. Jeho začátek je uvozen stejně jako u globálních stylů tagem <STYLE> a doporučuje se taktéž použít komentáře pro skrytí před prohlížeči, které styly nepodporují pomocí komentářů. Při potřebě vložení vlastního komentáře je pak potřeba použít párových značek /* a */.

```

<style type="text/css">
<!--
/* Změní barvu textu H1 na červenou. */
h1 {font-color:#ff0000}
/* Odstavce P budou mít černou barvu a červené pozadí. */
p {font-color:#000000; background:#ff0000;}
-->
</style>

```

5. 2 Elementy v CSS a jejich vlastnosti

Jak je z předchozího zápisu patrné elementy se v podstatě rozumí jednotlivé tagy kódu HTML. Je však také patrné že pokud bychom byli odkázáni pouze na tagy kódu, byli bychom značně omezeni a kaskádní styly by postrádaly jeden ze svých hlavních účelů, tedy zjednodušení strukturální podoby kódu. Proto jsou v CSS vytvořeny nové elementy, které rozšiřují schopnosti předdefinování vlastností prvků kódu HTML. Těmito elementy jsou třídy (CLASS) identifikátory (ID) a pseudotřídy (resp. pseudoprvky).

5. 2. 1 Základní elementy HTML kódu

Jak je již z předchozích příkladů patrné základní a nejjednodušší použití CSS je v předdefinování vlastností pro již existující HTML tagy. Tyto tagy se vyznačují vzájemnou dědičností. To znamená, že například tag <BODY> je nadřazený všem tagům používaným v jeho těle a tedy všechny tyto tagy dědí nepředdefinované vlastnosti tagu <BODY>. V předchozím příkladě to tedy například znamená, že tagy <P>, , <TD> dědí po tagu <BODY> barvu písma a jeho velikost, při čemž navíc nastavují typ písma jeho zarovnání. Tag <A> však dědí pouze velikost písma, protože barvu předdefinovává. Navíc nastavuje typ písma.

Tagy lze rovněž seskupovat. To znamená že pro několik tagů využívá stejná pravidla. Příklad seskupování je rovněž uveden v předchozím příkladě a to v řádku v němž zároveň definujeme vlastnosti pro tagy <P>, a <TD>.

V CSS lze navíc nastavovat vlastnosti dle kontextuálního použití elementů. To znamená, že můžeme například všem odkazům (tag <A>) použitým v seznamech (tag) nastavit jiné vlastnosti než ostatním. Toho docílíme pomocí tzv. kontejnerového zápisu. Tento zápis je definován jako seznam, který je rozdělen mezerami. Kontejnerový zápis lze použít i u ostatních typů elementů a dokonce i v jejich kombinacích. Mohou tak vznikat velice složité zápisy, pro velmi specifické případy použití kombinací elementů v HTML kódu. Tyto zápisy však bývají mnohdy velmi nepřehledné a v případě modifikace spíše stěžují práci než zlehčují.

Základní elementy HTML kódu dělíme do několika skupin. Tyto skupiny poté určují pro jaké skupiny je použitelná jaká vlastnost CSS. Těmito skupinami jsou:

Blokové elementy – jsou takové elementy, za kterými je zalomena řádka, nebo takové které vytvářejí určité bloky. Př. <P>, <H1>, <TD>, atd.

Inline elementy – jsou takové elementy, které jsou běžnou součástí textu na řádce, není před nimi ani za nimi žádné zalomení řádky. Př. , <I>, , atd.

Nahrazované elementy – jsou takové, které jsou nahrazeny nějakým obsahem a pro jejichž zařazení do okolního textu stránky jsou důležité pouze jejich vlastní rozměry.

Příklad:
, <OBJECT>, atd.

5. 2. 2 Třídy (CLASS)

Třídy slouží k dodefinování či předefinování vlastností HTML tagů, ve kterých jsou použity. Třidu definujeme obdobně jako tag až na to, že před její název předřadíme tečku (.třída {specifikace}). Její volání v HTML kódu je prováděno pomocí parametru CLASS (<tag class="třída">).

Příklad:

```
<html>
<head>
<style type="text/css">
<!--
/* Změní barvu textu H1 na červenou. */
h1 {font-color:#ff0000}
/* Odstavce P budou mít černou barvu a červené pozadí. */
p {font-color:#000000; background:#ff0000;}
/* změna barvy a nastavení písma na helveticu */
.special {font-color:#030609; font-family: helvetica;}
/* přidání černého pozadí */
h1.bgr {background:#000000;}
-->
</style>
</head>
<body>
<h1>Předdefinovaný nadpis H1</h1>
<h1 class="bgr">Nadpis H1 předdefinovaný třídou „bgr“</h1>
<h1 class="special">Nadpis H1 předdefinovaný třídou „special“</h1>
    <p>Tento odstavec bude naformátován dle předdefinovaného
nastavení tagu <P>.</p>
<p class="special">Tento odstavec bude naformátován dle změn ze třídy
„special“.</p>
<p class="bgr">Tento odstavec bude naformátován dle
předdefinovaného nastavení tagu <P>. Nebude totiž nalezena žádná
třída „bgr“, která by se vztahovala k tagu <P>.</p>
</body>
</html>
```

Jak vidíme z předchozího příkladu, třídy je možno použít několika způsoby. Mohou být určeny ke globálnímu použití, tedy použití s libovolným tagem HTML kódu, nebo být určeny pouze k použití s jedním konkrétním tagem. V prvním případě musí být název třídy pro celý kaskádní styl jedinečný, v druhém případě se může opakovat avšak z důvodu přehlednosti se toto řešení nedoporučuje. V druhém případě je také nutné uvést před název deklarované třídy tag, ke kterému se tato třída bude vztahovat (tag.třída {specifikace}). Je samozřejmostí, že pro jeden tag může být vytvořen libovolný počet tříd a, že libovolný tag HTML dokumentu se může odvolávat na libovolnou globální třídu CSS stylu (pokud to má samozřejmě smysl).

5. 2. 3 Identifikátory (ID)

Identifikátory mají podobné vlastnosti jako třídy. Tedy taktéž doplňují nebo mění vlastnosti definovaných tagů. Jejich hlavní užití je ovšem v jejich spojení s dynamickými prvky HTML. Jeho podstatou je tedy jednoznačně definovat element tzv. jej identifikovat. Proto se identifikátory používají obvykle pouze pro jeden určitý element v HTML dokumentu (nemusí být podmínkou). Syntaxe identifikátorů je taktéž obdobná jako u tříd, avšak na místo tečky před jménem je použita zahrádka (#identifikátor {specifikace}). Volání v HTML dokumentu probíhá pomocí atributu ID (<tag id="identifikátor">).

Konkrétní užití identifikátorů je např. při dynamickém vypínání a zapínání viditelnosti některých částí dokumentů (např. některých textů). Názvy identifikátorů se mohou shodovat s názvy tříd, ale opět se to ve většině případů nedoporučuje z důvodu nepřehlednosti. Jsou však i opačné případy např. při použití třídy i identifikátoru v jednom tagu, v takových případech bývá shodné pojmenování naopak přehlednější.

5. 2. 4 Pseudotřídy a pseudoprvky

Pseudotřídy (resp. pseudoprvky) obsahují prvky jednoho typu, které odpovídají určitým kontextuálním kritériím. Asi nejznámější příklad bude tag <A> tedy odkaz. U toho jsou ustaveny dle CSS2 čtyři kontextuální kritéria a tedy i čtyři pseudotřídy. Navštívené odkazy (pseudotřída visited), aktivní a nenavštívené odkazy (pseudotřída active), propojené odkazy (pseudotřída link) a odkazy, na které umístíme kurzor myši (pseudotřída hover – je součástí až normy CSS2 přesto, že je IE již dlouho podporována a je asi nejčastěji používanou pseudotřídou). Pseudotřída je ve stylu listů definována pomocí dvojtečky jako oddělovače (:pseudotřída {specifikace}). Následné volání potom probíhá zcela automaticky a pokud prvek odpovídá některému kontextu automaticky se provede formátování dle daného kontextu.

Příklad:

```
a <color:#ff0000; font-family: helvetica;>
a:visited <color:#982306; font-family: helvetica;>
<a href="second.html" >Druhá stránka</a>
```

V předchozím případě máme nadefinován styl pro tag <A> bez kontextu a pro již navštívené stránky (pseudotřída visited). Pokud tedy dojde k načítání tagu <A>, prohlížeč nejdříve vyhodnotí zda byl zobrazovaný odkaz již navštíven a pokud ano použije automaticky formátování dle definice pseudotřídy. Definice pseudotříd se většinou zapisují s předřazením tagu, ke kterému se vztahují. Je to vhodné obzvláště pro zvýšení přehlednosti stylu listu. V některých případech však můžeme tag zcela vynechat, protože daná pseudotřída se může vztahovat pouze k jednomu jedinému tagu (př. :link <color:#982306;>). Ovšem pokud použijeme pseudotřídu bez předřazeného tagu musíme si zároveň uvědomit, že tato pseudotřída se použije ve všech elementech, které tuto pseudotřídu podporují a které odpovídají požadovanému kontextu. Díky tomu se nám mnohdy může stát, že se elementy budou přeformátovávat aniž bychom to chtěli. Proto používání definice pseudotříd bez konkretizace vztažného elementu nelze doporučit.

Je samozřejmostí že i pseudotřídy lze kombinovat s ostatními elementy CSS stylů. V takovém případě platí pro zápis, že pseudotřída je uvedena až jako poslední. Pokud by byla pseudotřída uvedena před jiným elementem (např. třídou) mohl by si tento zápis

prohlížeč vysvětlit jako nastavení kontextu pro obecnější definici bez třídy a zachovat se jinak než jste zamýšlely (V tomto ohledu se mohou různé prohlížeče a různé kontexty chovat velmi různorodě – dle normy by však měla nastat popsaná reakce).

Pseudoprvky jsou naprostou obdobou pseudotříd, a jsou to vlastně specifické pseudotřídy, které se vztahují ke specifickým kontextům. V CSS1 jsou definovány pouze dva pseudoprvky a to pro zvýraznění prvního písmene či prvního řádku odstavce. Těmito pseudoprvky jsou first-letter (písmeno) a first-line (řádek). Je však nutno dodat, že tyto pseudoprvky jsou velmi špatně podporovány a například prohlížeč IE 5.0, který by měl dokonce splňovat normu CSS2 tyto prvky nepodporuje. Jejich zápis je naprosto shodný se zápisem pseudotříd.

5.3 Kaskádové pořadí

Tento odstavec se částečně vztahuje u k předchozí kapitole, ve které jsou popisovány technologie vkládání stylů listů. Jak si asi každý dokáže představit může díky mnoha typům elementů dojít při psaní kódu k přetěžování některých vlastností či dokonce celých elementů. I tyto situace se snaží norma CSS řešit. V podstatě platí pravidlo, že naposledy definované vlastnosti mají nejvyšší prioritu. Neznamena to, že by snad záleželo v jakém pořadí zapíšete elementy do stylu listů jako spíš v tom jak je voláte a jak je přetěžujete. Pokud jde o technologii vkládání stylů pak platí že nejnižší prioritu mají propojené styly (tedy styly listu definované v externím souboru). Pak následují globální styly listu (styly definované v hlavičce HTML dokumentu, při přetěžování je vhodné zapsat definici stylu až za voláním externího souboru, zabráníte tím nesmyslnému chování v některých prohlížečích). Nejvyšší prioritu mají pak in-line styly (styly definované v konkrétním tagu). Pokud jde o prioritu elementů dalo by se říci, že asi největší prioritu má identifikátor (ID). Za ním následuje třída (CLASS) dále pak pseudotřída (resp. pseudoprvek) a nejnižší prioritu mají definice základních elementů HTML. Avšak určení priority u elementů CSS není vůbec snadná záležitost a při pohledu z jiného úhlu můžeme definovat priority odlišně. Zde je několik příkladů, aby bylo snáze pochopitelné jak se jednotlivé prvky v různých kombinacích chovají.

Příklady:

```
p {font-family: helvetica; text-align:justify}
p.times {font-family: times;}
p#red {color:#ff0000; font-family: helvetica;}
#blue {color:#0000ff;}
p.first-letter {font-size:120%;}
```

<p>Text psaný písmem Helvetica, zarovnaný do bloku s implicitní barvou a velikostí písma. První písmeno by mělo být zvětšené o 20% (Většinou není podporováno prohlížeči).</p>

<p class="times">Text psaný písmem Times New Roman, zarovnaný do bloku s implicitní barvou a velikostí písma.</p>

<p class="times" id="red">Text psaný písmem Helvetica, zarovnaný do bloku s červenou barvou a implicitní velikostí písma.</p>

<p class="times" id="blue">Text psaný písmem Times New Roman, zarovnaný do bloku s modrou barvou a implicitní velikostí písma. Dochází k ignoraci atributu CLASS.</p>

<p id="blue" >Text psaný písmem Helvetica, zarovnaný do bloku s modrou barvou a implicitní velikostí písma. První písmeno by mělo být zvětšené o 20%!!? Toto

neodpovídá prioritě, ale přesto by to tak mělo fungovat. vlastnost `p:first letter` by se totiž dle zásad CSS měla vztahovat pouze k samostatně volanému `<P>` bez dodefinovaných tříd a identifikátorů.

5. 4 Jednotky

5. 4. 1 Délkové jednotky

Mnoho parametrů nastavovaných v kaskádových stylech je hodnoty udávající míru, či vzdálenost. Tyto hodnoty se zapisují jako celá či desetinná čísla (desetinná místa se oddělují tečkou), jež mohou být buďto kladné či záporné hodnoty. Ihned za vlastní číslicí potom následují jednotky ve kterých je hodnota zadána (pozor mezera mezi těmito údaji je chybou). Tyto jednotky jsou vyjádřeny dvou písmennými zkratkami. Pokud je číslice uvedena bez jednotek je považována za hodnotu udávanou v pixelech (proto dochází k chybám při mezeře mezi číslicí a jednotkou). Používané jednotky se dělí na absolutní a relativní.

Absolutní jednotky jsou takové jejichž hodnota je přesně definována a je odvoditelná od základních jednotek tabulky SI, přesněji řečeno od jednotky délky metru. Do této skupiny bývá někdy zařazován i pixel. Pixel je však bezrozměrný a jeho aktuální fyzická velikost je závislá na nastavení parametrů zobrazení monitoru popř. i tisku na tiskárně.

Relativní jednotky jsou takové jejichž hodnota není přesně definována a závisí na nastavení jiných parametrů.

5. 4. 2 Procenta

Tak jako v klasickém HTML mohou být i v kaskádních stylech délkové hodnoty (a dokonce i barvy) specifikovány pomocí procent. Jejich velkou výhodou a zároveň nevýhodou je, že pevně nenastavuje konečné hodnoty, ale pouze upravuje implicitní či dříve nastavené hodnoty. Nastavíme – li tedy velikost písma např. na 120% bude se klientovi náš text zobrazovat o 20% větším písmem než má nastaveno v prohlížeči. To může mít velké výhody pro klienta (např. pokud je klient zrakově postižený a využívá speciálních přednastavení prohlížeče, zobrazí se mu text v pro něj čitelné formě). Nerozumné používání procent však může naopak přinést velké problémy programátorům (Pokud totiž použije klient nestandardní nastavení prohlížeče, může se stát že např. právě velikostí písma zcela naruší zobrazení ostatních obzvláště grafických prvků a celý design stránky může být narušen). Při použití procent bychom proto měly být opatrní a pokud může specifické nastavení prohlížeče narušit koncepci našich stránek měli by jsme se jej úplně vyvarovat či jej použít pouze pro nastavení hodnot, které jsme již před tím nastavili běžným způsobem. Velmi vhodné se naopak jeví použití procent v případě kdy potřebujeme hodnoty (např. velikosti tabulek) vztáhnout k nám neznámým hodnotám (např. rozlišení monitoru).

Ve své podstatě jsou procenta pouze další relativní jednotkou. To tedy v případě procent znamená, že dle vztažné hodnoty (implicitní nastavení či předchozí hodnota) dojde k odvození absolutní hodnoty. Také u procent můžeme před číselnou hodnotu předsadit znaménko `[+ / – ]`. Implicitní je plus. Za číselnou hodnotu potom dosadíme znak „%“. Pozor ani mezi znamínkem a číslicí a ani mezi číslicí a znakem procent nejsou mezery. V případě mezery před znakem procent by totiž prohlížeč přiřadil číslici implicitní jednotky a procenta by ignoroval.

Příklad:

```
h1 {font-size: +120%; font-weight: bold;}  
p.small {font-size: -20%; font-weight: lighter;}
```

5. 5 Barvy

Při zápisu barev máme dvě hlavní možnosti zápisu.

První možností je zápis pomocí anglického jména v řetězcové notaci. Jména barev jsou specifikována v normě HTML a proto se jimi nebudeme zabývat.

Druhou možností jak zapsat barvu je zadat přímo její barevné složení pomocí RGB (RED – GREEN – BLUE) notace. Pro tento zápis existují v kaskádových stylech čtyři syntaktické způsoby. V každém z těchto způsobů odděleně definujeme tři složky barvy, červenou, zelenou a modrou. A to buď pomocí hexadecimálních čísel, dekadických čísel a nebo procentuálního vyjádření.

Příklad:

```
p.blue {color: #00f; font-family: helvetica; text-align: justify}  
p.yellow {color: #ffff00; font-family: helvetica; text-align: justify}  
p {color: rgb(0, 0, 0); font-family: helvetica; text-align: justify}  
p.red {color: rgb(100%, 0%, 0%); font-family: helvetica}
```

5. 6 Přehled vlastností CSS1

Následující kapitola obsahuje souhrn vlastností, které lze definovat pomocí kaskádních stylů normy 1. Protože se jednotlivé vlastnosti vztahují ke konkrétním elementům resp. typům elementů jsou také vlastnosti rozděleny do skupin. K většině elementů se vztahuje více skupin, ale pro větší přehlednost jsou skupiny rozděleny také podle vlastností se kterými pracují. První skupina pracuje s vlastnostmi nastavujícími barvy a pozadí, druhá s vlastnostmi písma, třetí textu, čtvrtá boxů a bloků, pátá pracuje s klasifikačními vlastnostmi a šestá (která má povahu doporučení a je standardizována až v CSS normy 2) pracuje s vlastnostmi řízení pozice.

5. 6. 1 Vlastnosti barev a pozadí

Vlastnost „color“

Tato vlastnost nastavuje barvu.

Vlastnost „background-color“

Tato vlastnost nastavuje barvu pozadí.

Vlastnost „background-image“

Tato vlastnost vkládá do pozadí elementu grafický soubor.

Vlastnost „background-repeat“

Tato vlastnost určuje způsob opakovaného vkládání obrázku do pozadí.

Vlastnost „background-attachment“

Vlastnost určuje zda dochází k pohybu pozadí s elementem nebo zda je pozadí fixováno a pohybuje se pouze element po pozadí.

Vlastnost „background-position“

Vlastnost určuje výchozí pozici pro vkládání obrázku do pozadí (udává se X a Y pozice)

Vlastnost „background“

Parametr pro komplexní nastavení vlastností pozadí. Vlastnosti nastavujeme v pořadí background-color, background-image, background-repeat, background-attachment, background-position. (pořadí není povinné a kterákoli položka může být vypuštěna)

5. 6. 2 Vlastnosti písma

Vlastnost „font-family“

Tato vlastnost nastavuje rodinu písem, tedy seznam písem která se mají použít pro zobrazení textu.

Vlastnost „font-style“

Vlastnost nastavuje styl písma (normálně, kurzívu či sklonění).

Vlastnost „font-variant“

Vlastnost nastavuje variantu písma (normálně či kapitálky).

Vlastnost „font-size“

Vlastnost nastavuje velikost písma.

Vlastnost „font-weight“

Vlastnost nastavuje duktus (tloušťku) písma.

Vlastnost „font“

Parametr pro komplexní nastavení vlastností pozadí. Vlastnosti nastavujeme v pořadí font-style, font-variant, font-weight, font-size, line-height a font-family .

5. 6. 3 Vlastnosti textu

Vlastnost „word-spacing“

Vlastnost nastavuje velikost mezislovní mezery v textu.

Vlastnost „letter-spacing“

Vlastnost nastavuje velikost mezery mezi jednotlivými písmeny ve slově.

Vlastnost „text-decoration“

Vlastnost nastavuje dekoraci (speciální efekty) písma

Vlastnost „vertical-align“

Vlastnost nastavuje souvislé vertikální zarovnání textu (resp. vloženého inline elementu)

Vlastnost „text-transform“

Vlastnost převede text na daný formát (kapitálky a velká nebo malá písmena).

Vlastnost „text-align“

Vlastnost nastavuje odstavcové zarovnání textu (k levému či pravému okraji, na střed nebo do bloku).

Vlastnost „text-indent“

Vlastnost nastavuje velikost odstavcové odrážky (o kolik bude odsazen první řádek v odstavci).

Vlastnost „line-height“

Vlastnost nastavující výšku řádku textu (tzv. meziřádkový proklad).

5. 6. 4 Vlastnosti boxů a bloků

Vlastnost „margin-top“

Vlastnost nastavuje velikost horního okraje (př. velikost místa nad blokem textu).

Vlastnost „margin-right“

Vlastnost nastavuje velikost pravého okraje (př. velikost místa za blokem textu).

Vlastnost „margin-bottom“

Vlastnost nastavuje velikost spodního okraje (př. velikost místa pod blokem textu).

Vlastnost „margin-left“

Vlastnost nastavuje velikost levého okraje (př. velikost místa před blokem textu).

Vlastnost „margin“

Vlastnost komplexně nastavuje okraje kolem bloku.

Vlastnost „padding-top“

Vlastnost nastavuje velikost vnitřního horního okraje (příp. velikost místa nad blokem textu v buňce tabulky).

Vlastnost „padding-right“

Vlastnost nastavuje velikost pravého vnitřního okraje (př. velikost místa za blokem textu v buňce tabulky).

Vlastnost „padding-bottom“

Vlastnost nastavuje velikost spodního vnitřního okraje (př. velikost místa pod blokem textu v buňce tabulky).

Vlastnost „padding-left“

Vlastnost nastavuje velikost levého vnitřního okraje (př. velikost místa před blokem textu v buňce tabulky).

Vlastnost „padding“

Vlastnost komplexně nastavuje vnitřní okraje bloku.

Vlastnost „border-top-width“

Vlastnost nastavuje šířku rámečku nad elementem. (př. velikost horního okraje tabulky).

Vlastnost „border-right-width“

Vlastnost nastavuje šířku rámečku za elementem. (př. velikost pravého okraje tabulky).

Vlastnost „border-bottom-width“

Vlastnost nastavuje šířku rámečku pod elementem. (př. velikost spodního okraje tabulky).

Vlastnost „border-left-width“

Vlastnost nastavuje šířku rámečku před elementem. (př. velikost levého okraje tabulky).

Vlastnost „border-width“

Vlastnost komplexně nastavuje šířku rámečku kolem elementu.

Vlastnost „border-color“

Vlastnost nastavuje barvu rámečku.

Vlastnost „border style“

Vlastnost nastavuje způsob vykreslení rámečku (tvar nebo styl).

Vlastnost „border-top“

Vlastnost komplexně nastavuje horní část rámečku. Je kombinací vlastností „border-top-width“, „border-style“ a „border-color“.

Vlastnost „border-right“

Vlastnost komplexně nastavuje pravou část rámečku. Je kombinací vlastností „border-right-width“, „border-style“ a „border-color“.

Vlastnost „border-bottom“

Vlastnost komplexně nastavuje spodní část rámečku. Je kombinací vlastností „border-bottom-width“, „border-style“ a „border-color“.

Vlastnost „border-left“

Vlastnost komplexně nastavuje horní část rámečku. Je kombinací vlastností „border-top-left“, „border-style“ a „border-color“.

Vlastnost „border“

Vlastnost komplexně nastavuje rámeček elementu. Je kombinací vlastností „border-width“, „border-style“ a „border-color“.

Vlastnost „width“

Vlastnost nastavující šířku elementu

Vlastnost „height“

Vlastnost nastavující výšku elementu

Vlastnost „float“

Vlastnost nastavuje způsob, kterým mohou ostatní elementy obtékat tento element.

Vlastnost „clear“

Vlastnost povoluje či zakazuje elementu obtékat okolo jiného elementu.

5. 6. 5 Klasifikační vlastnosti

Vlastnost „display“

Vlastnost nastavuje či předefinovává druh elementu (způsob zobrazení). Tato vlastnost (zde spíše parametr) přistupuje k tvorbě CSS stylů trochu odlišným způsobem. Na místo přiřazení konkrétní vlastnosti, totiž třídí elementy do předem specifikovaných kategorií.

Vlastnost „white-space“

Vlastnost určuje způsob práce s mezerami tabelátory konci řádků, atd., uvnitř elementu.

Vlastnost „list-style-type“

Vlastnost nastavuje vzhled odrážky (uvozující značky seznamu).

Vlastnost „list-style-image“

Vlastnost nastavuje místo uvozujícího znaku seznamu obrázek.

Vlastnost „list-style-position“

Vlastnost určuje umístění odrážky seznamu, buďto před textem nebo v textu.

Vlastnost „list-style“

Vlastnost komplexně nastavuje vzhled seznamu. Vlastnosti nastavujeme v pořadí „list-style-type“, „list-style-position“, „list-style-image“, při čem definování první a zároveň poslední vlastnosti není nutné a postrádá smysl.

5. 6. 6 Vlastnosti pro řízení pozice

Protože tyto vlastnosti kaskádních stylů jsou implementovány do normy CCS1 pouze jako doporučení a s několika rozšířeními jsou standardizovány až v normě CSS2. Proto budou všechny tyto vlastnosti („position“, „left“, „top“, „width“, „height“, „clip“, „overflow“, „z-index“, „visibility“) probrány až v následujícím odstavci.

5. 7. Nové vlastnosti CSS2

5. 7. 1 Rozlišení typů médií

Parametr „@media“

Parametr přiřazuje nedefinovaný styl jednotlivým zobrazovacím (a zvukovým) médiím.

Parametr „@page“

Parametr „@page“ slouží k oddělení definici vlastností pro stránkovaná média.

Vlastnost „size“

Vlastnost nastavuje velikost stránky a orientaci písma na stránce.

Vlastnost se používá ve stránkovém kontextu.

Vlastnost „marks“

Vlastnost specifikuje vkládání znaků pro řez stránky a jiné specifické formátovací symboly.

Vlastnost se používá ve stránkovém kontextu.

Vlastnost „margin“

Vlastnost shodná s vlastností „margin“ u nastavení boxů a bloků. Taktéž lze použít „margin-top“, „margin-right“, „margin-bottom“, „margin-left“.

Pseudoprvky „:first“, „:left“, „:right“

Tyto pseudoprvky slouží ke specifickému nastavování vlastností stránek. Pseudoprvek „:first“ nastavuje styl první stránky dokumentu. Pseudoprvek „:left“ (resp. „:right“) pro každou levou (resp. pravou) stránku dokument. Tyto pseudoprvky jsou vhodné zejména pro nastavování okrajů.

5. 7. 2 Vlastnosti zvukových stylů**Vlastnost „volume“**

Vlastnost nastavující hlasitost přehrávání zvukového stylu. Při použití zvukových stylů by mělo být dbáno na to aby alespoň tuto vlastnost mohl uživatel ovlivnit (sám si ji optimálně nastavit), protože u této hodnoty nelze nastavit optimální hranici každý audio systém je trochu jiný a každý uživatel má trochu jiné požadavky.

Vlastnost „speak“

Vlastnost nastavuje způsob hlasové interpretace textů.

Vlastnost „pause-before“

Vlastnost nastavuje velikost zvukové pomlky před hlasovým překladem elementu.

Vlastnost „pause“

Vlastnost nastavuje velikost zvukové pomlky před a za hlasovým překladem elementu. Je kombinací „pause-before“ a „pause-after“.

Vlastnost „cue-after“

Vlastnost nastavuje zvukovou ikonu, která bude spuštěna za hlasovým překladem vztažného elementu.

Vlastnost „cue“

Vlastnost nastavuje zvukové ikony, která budou spuštěny před a za hlasovým překladem vztažného elementu.

Vlastnost „play-during“

Vlastnost nastavuje zvukové pozadí hlasového překladu.

Vlastnost „azimuth“

Vlastnost nastavuje 3D pozici hlasu.

Vlastnost „elevation“

Vlastnost nastavuje horizontální polohu hlasu.

Vlastnost „speech-rate“

Vlastnost nastavuje rychlost hlasového překladu.

Vlastnost „voice-family“

Vlastnost nastavuje rodinu hlasů (resp. použitý hlas či skupinu možných použitých hlasů). Tato vlastnost je obdobou výběru správného písma a pokud tuto vlastnost nepodporuje prohlížeč nebo není v systému žádné generické zvukové schéma, nelze používat zvukový překlad.

5. 7. 3 Nastavení kurzoru myši**Vlastnost „cursor“**

Vlastnost nastavuje tvar kurzoru (resp. seznam možných tvarů), který bude použit při umístění kurzoru nad daný element.

5. 7. 4 *Filtry*

Filtr „alpha“

Filtr nastavuje průhlednost elementu a umožňuje vytvářet tzv. transparentní přechody (nazývány též gradienty).

Filtr „blur“

Filtr nastavuje rozmazání elementu. Simuluje zachycení pohybu daného elementu.

Filtr „chroma“

Filtr vypustí nadefinovanou barvu a element se v těchto místech stává průhledný.

Filtr „dropshadow“

Filtr vytvoří stín elementu a zdánlivý efekt že je element umístěn nad povrchem stránky samotné.

Filtr „fliph“

Tento filtr převrací element okolo horizontální osy.

Filtr „flipv“

Tento filtr převrací element okolo vertikální osy.

Filtr „glow“

Filtr vytváří efekt podkreslení objektu definovanou barvou.

Filtr „grey“

Filtr vytváří efekt černobílého zobrazení elementu. Zobrazení provedeno v odstínech šedi.

Filtr „invert“

Filtr vytváří efekt negativu. Všechny barvy elementu jsou invertovány na opačné barvy spektra.

Filtr „mask“

Filtr vytváří pro daný element transparentní masku.

Filtr „shadow“

Zjednodušený filtr vytvářející efekt stínu.

Filtr „wave“

Filtr vytváří horizontální efekt vlnění ve tvaru sinusoidy.

Filtr „xray“

Filtr vytváří efekt rentgenu. To znamená, že vytváří černobílý obrázek (barevná hloubka nastavena na 1), který zachycuje obrysy původního elementu.

5. 7. 5 *Nové vlastnosti písma*

Norma CSS2 přináší i nové možnosti do práce s písmy. Jejich největším nedostatkem byla práce s konkrétními fonty. Pokud totiž uživatel nevlastnil požadovaný font písma neexistoval způsob jak tento font nahradit či dokonce tento font uživateli zprostředkovat. Norma CSS2 proto nabízí čtyři různé způsoby jak se dopracovat k požadovanému fontu či vykreslení písma. Prvním z těchto možností je, že uživatel požadované písmo vlastní (původní metoda CSS1), pak je vše v pořádku a požadované písmo se použije. Další tři nové možnosti však řeší případ kdy uživatel požadované písmo nevlastní. V takovém případě je možno použít metodu „inteligentního hledání“. To znamená, že je hledáno písmo, které je co nejvíce podobné požadovanému fontu skutečným vzhledem a způsobem vykreslování. Třetí metoda, „metoda syntézy“ je schopna navíc toto nejpodobnější písmo ještě upravit, aby výsledné zobrazení co nejvíce odpovídalo vlastnostem původně požadovaného fontu. Poslední čtvrtá metoda umožňuje uživateli načíst požadované písmo přímo z Internetu. CSS2 navíc umožňuje tyto tři metody kombinovat (tzv. progresivní vykreslování). To znamená že v případě

nenalezení požadovaného fontu je automaticky hledán nejpodobnější font a zároveň je požadovaný font načítán z webu. Jakmile je podobný font nalezen a popřípadě i upraven je požadovaný text přechodně zobrazen náhradním fontem. V okamžiku kdy je pak načten požadovaný font dojde k automatickému překreslení pravým fontem.

Abychom mohly těchto nových vlastností využít musíme pro každé písmo nadefinovat nový specifický parametr „@font-face“. Pomocí tohoto parametru definujeme vlastnosti fontu dle kterých je při „inteligentním hledání“ vybírán podobný font, při „metodě syntézy“ i upravován náhradní font. V tomto parametru taktéž definujeme umístění tohoto fontu na Internetu pro načtení písma. Parametr má syntaxi:

@font-face {font-family: hodnota; <font-description>}

Jeho definice je trochu odlišná od obou nových specifických parametrů „@media“ a „@page“. Namísto definování jména před samotnými vlastnostmi parametru je jeho specifikace provedena až v samotném těle pomocí známé vlastnosti „font-family“. V definici „@font-face“ musí být vždy jedno písmo což přímo vyplývá z kontextu.

Prvních pět vlastností je shodných s vlastnostmi písma v normě CSS1. Je však třeba si uvědomit v jakém kontextu jsou tyto vlastnosti definovány zde. Zde tyto vlastnosti nenastavují existující font, ale určují vlastnosti hledaného fontu.

„font-family“ – vlastnost shodná s normou CSS1. Uvádíme však vždy jen jméno popisovaného písma.

„font-style“ – tato vlastnost popisuje řez požadovaného písma. Vlastnost shodná s normou CSS1 avšak přibývá hodnota „all“, která dovoluje shodu se všemi řezy písma.

„font-variant“ – popisuje variantu písma zda písmo klasická či kapitálová. Vlastnost shodná normou CSS1.

„font-weight“ – popisuje šířku (tučnost) požadovaného fontu. Vlastnost shodná s normou CSS1, avšak nedoporučuje používat hodnoty „bolder“, „lighter“ (v kontextu „@font-face“ tyto hodnoty ani nelze použít) a přibývá hodnota „all“, která dovoluje shodu se všemi šířkami.

„font-size“ – popisuje dovolené velikosti pro shodu fontů. Vlastnost shodná s normou CSS1, avšak kontextu „@font-face“ nejsou povoleny jiné než číselné hodnoty a nová hodnota „all“. Ten je doporučován pro všechny True-Type písma.

Vlastnost „font-stretch“

Vlastnost popisuje šířku písma.

Vlastnost „unicode-range“

Vlastnost popisuje hodnoty variant Unicode, které má obsahovat písmo hledané pro shodu s původním fontem.

Vlastnost „units-per-em“

Vlastnost popisuje délkovou jednotku písma pro použití v dalších parametrech. Konkrétně sděluje, kolik jednotek se vejde do čtverčíku „em“.

Vlastnost „src“

Odkaz (či seznam odkazů) na soubor obsahující definici požadovaného písma.

Vlastnost „panose-1“

Vlastnost popisuje požadovaný typ písma podle standardu Panose, dnes zřejmě nejpoužívanějšího standardu pro písma typu latinka.

Vlastnosti „stemv“ a „stemh“

Vlastnosti popisují vertikální (stemv) a horizontální (stemh) šířku písma.

Vlastnost „slope“

Vlastnost popisuje základní úhel sklonu písma od svislé osy.

Vlastnost „cap-height“

Vlastnost popisuje výšku velkých písmen u původního písma.

Vlastnost „x-height“

Vlastnost popisuje výšku malých písmen u původního písma.

Vlastnost „ascent“

Vlastnost popisuje maximální výšku znaku bez diakritiky.

Vlastnost „descent“

Vlastnost popisuje maximální výšku písma pod základnou bez diakritiky. Některé jazyky totiž používají diakritiku i pod písmenem - např. arabština.

Vlastnost „text-shadow“

Vlastnost nastavuje textu elementu efekt stínu. Stín je součástí textu nezmění velikost boxu okolo textu, i když tento box přesáhne. „z-index“ (viz. vlastnosti pro řízení pozice) stínu je stejný jako samotného textu avšak stín nikdy nepřekryje text. Efektů stínů může být definován v jedné definici elementu libovolný počet.

5. 7. 6 Nové vlastnosti boxů a bloků

Vlastnost „border-top-color“

Vlastnost nastavuje barvu horního rámečku. Díky této vlastnosti taktéž vlastnost „border-top-style“ nastavuje barvu pouze pro horní rámeček.

Vlastnost „border-right-color“

Vlastnost nastavuje barvu pravého rámečku. Díky této vlastnosti taktéž vlastnost „border-right-style“ nastavuje barvu pouze pro pravý rámeček.

Vlastnost „border-bottom-color“

Vlastnost nastavuje barvu dolního rámečku. Díky této vlastnosti taktéž vlastnost „border-bottom-style“ nastavuje barvu pouze pro dolní rámeček.

Vlastnost „border-left-color“

Vlastnost nastavuje barvu levého rámečku. Díky této vlastnosti taktéž vlastnost „border-left-style“ nastavuje barvu pouze pro levý rámeček.

5. 7. 7 Vlastnosti pro řízení pozice

Celá tato skupina vlastností se sice již v normě CSS1 vyskytuje avšak pouze ve formě doporučení (jež není plnoprávnou součástí normy). V normě CSS2 jsou tyto vlastnosti již vedeny jako předepsané vlastnosti. Oproti normě CSS1 došlo v těchto vlastnostech k několika změnám a bylo přidáno několik dalších vlastností, které se v doporučení CSS1 nevyskytují. Mezi vlastnosti pro řízení pozice je v normě CSS2 zařazena i vlastnost „display“, která je v normě CSS1 řazena mezi vlastnosti klasifikační. Je to proto že tato vlastnost získala nové hodnoty, které mají více společné s pozicováním (či způsobem pozicování) elementu než z klasickou klasifikací.

Vlastnost „display“

Vlastnost rozděluje elementy do kategorií, které mají specifické vlastnosti chování (především specifický způsob zobrazení).

Vlastnost „position“

Vlastnost určuje způsob pozicování (umístění elementu na stránce dokumentu) elementu. Doporučení této vlastnosti v normě CSS1 neobsahuje hodnotu „fixed“.

Vlastnost „left“

Vlastnost nastavuje posunutí od levého okraje (resp. posunutí vpravo oproti původní pozici v případě relativního formátování).

Vlastnost „top“

Vlastnost nastavuje posunutí od horního okraje (resp. posunutí dolů oproti původní pozici v případě relativního formátování).

Vlastnost „right“

Vlastnost nastavuje posunutí od pravého okraje (resp. posunutí vlevo oproti původní pozici v případě relativního formátování). Tuto vlastnost doporučení normy CSS1 neobsahuje. Jde tedy o novinku CSS2.

Vlastnost „bottom“

Vlastnost nastavuje posunutí od spodního okraje (resp. posunutí nahoru oproti původní pozici v případě relativního formátování). Tuto vlastnost doporučení normy CSS1 neobsahuje. Jde tedy o novinku CSS2.

Vlastnost „width“

Vlastnost nastavující šířku elementu.

Vlastnost „height“

Vlastnost nastavující výšku elementu.

Vlastnosti „min-width“ a „max-width“

Tyto dvě vlastnosti dovolují programátorovi vytvořit určité rozmezí při nastavování šířky boxu elementu. Vlastnost „min-width“ určuje minimální a vlastnost „max-width“ maximální šířku boxu. Tyto vlastnosti jsou novinkou CSS2.

Vlastnosti „min-height“ a „max-height“

Tyto dvě vlastnosti dovolují programátorovi vytvořit určité rozmezí při nastavování výšky boxu elementu. Vlastnost „min-height“ určuje minimální a vlastnost „max-height“ maximální výšku boxu. Tyto vlastnosti jsou novinkou CSS2.

Vlastnost „z-index“

Vlastnost nastavuje pozici elementu na pseudoose Z. Tedy zda bude element překrývat ostatní nebo zad bude naopak překrýván jinými.

Vlastnost „clip“

Vlastnost nastavuje obdélníkovou viditelnou část elementu. Běžně je viditelný celý element, ale v případě absolutního polohování a použití „width“ a „height“ můžeme zobrazit menší část než je celý element.

Vlastnost „overflow“

Vlastnost nastavuje způsob práce s elementy, které se nevejdou do nadefinovaných prostorů. Zda element tento prostor přeteče, bude oříznut či jej bude možno skrolovat a tím prohlížet celý. Tato vlastnost má oproti doporučení CSS1 zcela odlišné hodnoty.

Vlastnost „visibility“

Vlastnost nastavuje viditelnost elementu. Norma CSS2 obsahuje novou hodnotu pro toto vlastnost a to „collapse“.

6 ASP (Active Server Pages)

ASP je skriptovací prostředí pro servery, které můžete využít pro tvorbu a provozování dynamických, interaktivních webových aplikací. S využitím ASP můžeme kombinovat HTML stránky, skripty a COM komponenty a vytvářet tak interaktivní webové stránky nebo výkonné webové aplikace, které se snadno vyvíjejí i upravují.

Pokud si prohlédneme libovolnou ASP stránku, zjistíme, že je složena ze tří typů zápisu - částečně ASP, částečně HTML příkazy a částečně prostý text.

text:

- informace, která se zobrazí uživateli
- text je zobrazen prohlížečem
- prostý ASCII text

HTML značky:

- slouží pro formátování textu, vkládání obrázků do stránky apod.
- prohlížeč podle těchto značek provede formátování textu
- každá značka je ohraničena špičatými závorkami < >

ASP příkazy:

- kód, podle něhož webový server, na kterém běží překladač ASP, generuje části stránek odesílaných klientovi
- ASP příkazy zpracuje dynamická knihovna asp.dll na webovém serveru
- každý úsek ASP je uzavřen mezi znaky <% a %>, ASP umožňuje použití proměnných, větvení, cyklů atd.

Další možnost připomíná vkládání JavaScriptu do HTML stránek.

```
<SCRIPT LANGUAGE="jazyk" RUNAT="SERVER"SRC="jméno_souboru">  
</SCRIPT>
```

6.1 Kolekce

Kolekce by se daly popsat jako objekty, které v sobě uchovávají seznam pojmenovaných hodnot. Vnitřní objekty ASP mají hned několik kolekcí. Každý kdo programuje stránky v ASP se dříve či později setká alespoň s kolekcí Request.Form nebo Request.QueryString, které zpřístupňují data zadaná uživatelem do formuláře.

6.1.1 Metody kolekcí

Kolekce standardně nabízejí metody Remove a RemoveAll, atributy Count, Item a Key. Obě metody slouží pro odebrání hodnot z kolekce.

Metoda Remove - odstraní z kolekce hodnotu se zadaným jménem.

Chceme-li například odstranit z kolekce Request.Form hodnotu s názvem Jmeno, použijeme následující příkaz:

```
Request.Form.Remove("Jmeno")
```

Metoda RemoveAll - odstraní z kolekce všechny hodnoty.

Atribut Count - obsahuje informaci o počtu hodnot uložených v kolekci.

Atribut Item - slouží pro zjištění hodnoty prvku kolekce.

Příklad zjištění hodnoty textového pole heslo formuláře odeslaného metodou POST:

```
hodnota = Request.Form.Item("heslo")
```

Jako index atributu Item můžeme použít název hodnoty (klíč - viz atribut Key) nebo pořadové číslo atributu z rozsahu 1 až Count.

Při přístupu k prvkům kolekce je možné v zápisu vynechat atribut Item.

Atribut Key - slouží ke zjištění názvu (klíče) prvku kolekce na základě jeho pořadového čísla.

klic = Request.Form.Key(1)

6. 2 Vnitřní objekty ASP

6. 2. 1 Objekt Application

Tento objekt umožňuje předávat hodnoty proměnných mezi jednotlivými stránkami dané ASP aplikace, ale také mezi jednotlivými uživateli pracujícími s naší aplikací. Typicky se proměnné viditelné na úrovni celé aplikace využívají například pro realizaci počítačové aplikace.

Objekt Application je vytvořen v okamžiku, kdy první uživatel přistoupí k některé stránce naší webové aplikace. Ihned po vytvoření objektu Application, je vytvořen také objekt Session, do něhož se ukládají proměnné viditelné pouze pro tohoto uživatele v rámci jeho session.

Objekt Application poskytuje dvě metody, které umožňují řídit přístup ke globálním proměnným na úrovni aplikace, aby bylo možné ochránit obsah těchto proměnných proti jeho porušení při souběžném přístupu více uživatelů.

Metoda Lock uzamkne objekt Application tak, že k němu může přistupovat pouze daná ASP stránka, tím se zamezí porušení dat při souběžném čtení a aktualizaci více stránkami.

Metoda Unlock uvolní zámek na objektu Application.

Tato metoda by měla být zavolána co nejdříve po uzamčení metodou Lock, neboť při pokusu o přístup k zamčenému objektu Application musí ostatní klienti čekat na uvolnění zámku a zbytečné uzamykání tak prodlužuje dobu načítání stránek ostatním uživatelům.

Atribut Value zprostředkovává přístup k hodnotám uloženým pro danou aplikaci. Obvykle se vynechává.

Událost OnStart nastává, když první klient zavolá některou ze stránek dané ASP aplikace.

Událost nastane před vytvořením session a před zpracováním stránky, kterou klient požaduje. V obslužné funkci této události se provádí inicializace proměnných a objektů, které mají být globální pro danou aplikaci (tzn. pro všechny stránky aplikace a pro všechny uživatele).

Událost OnEnd je vyvolána v případě ukončení aplikace. K tomu může dojít například při vypínání webového serveru.

Kolekce Contents obsahuje všechny proměnné a hodnoty uložené pro danou aplikaci, kromě objektů vytvořených s využitím elementu <OBJECT>.

Přestože je možný přístup k proměnným dané aplikace prostřednictvím kolekce Contents, obvykle se tato kolekce používá jen pro zjištění seznamu proměnných, které jsou pro tuto aplikaci dostupné. K jednotlivým proměnným se obvykle přistupuje přímo přes objekt Application.

6. 2. 2 Objekt ASPError

V ASP 3.0 přibyl nový objekt ASPError, který umožňuje jednodušší ladění aplikací. Tento objekt se používá pro zjištění detailnějšího popisu poslední chyby, která nastala při zpracovávání ASP stránky. Instanci objektu ASPError získáme voláním metody Server.GetLastError.

Atribut ASPCode obsahuje kód chyby.

Hodnotu atributu ASPCode generuje IIS a pro většinu chyb ve skriptu není nastaven.

V případě, že se jedná o chybu související s ASP, obsahuje atribut ASPDescription podrobný popis této chyby. Hodnota tohoto atributu je generována preprocesorem ASP a opět je pro většinu ASP chyb prázdná.

Atribut Category obsahuje informaci o původu chyby, tzn. zda ji vygenerovalo ASP, překladač skriptovacího jazyka nebo externí objekt.

Hodnota tohoto atributu je v případě výskytu chyby vždy nastavena.

Atribut Column obsahuje pozici znaku v řádce zdrojového souboru, na níž se vyskytla chyba.

Atribut Description obsahuje stručný popis chyby. V případě výskytu chyby je vždy nastaven.

Atribut File obsahuje název souboru, který byl zpracováván, když došlo k chybě.

Atribut Line obsahuje číslo řádky souboru, kde byla detekována chyba.

Atribut Number obsahuje standardní COM chybový kód. V případě, že na stránce nedošlo k chybě, obsahuje tento atribut hodnotu 0.

Atribut Source - v případě, že se jedná o syntaktickou chybu, obsahuje atribut Source zdrojový text řádky, na níž došlo k chybě.

6. 2. 3 Objekt Request

Objekt Request obsahuje metody pro práci s daty obsaženými v HTTP požadavku. Používá se např. pro přístup k datům formulářů. Tento objekt poskytuje pouze jednu metodu.

Metoda BinaryRead umožňuje přímé čtení dat odeslaných z formuláře, což je vhodné především pro zpracování uploadovaného souboru. Před použitím metody BinaryRead nesmí být použita kolekce Form a před použitím kolekce Form nesmí být použita metoda BinaryRead.

Atribut TotalBytes obsahuje informaci o počtu bytů v těle HTTP požadavku.

Kolekce ClientCertificate – v případě zabezpečeného přístupu (např. technologií SSL) obsahuje tato kolekce informace o digitálním certifikátu klienta.

Kolekce Cookies umožňuje webovému serveru ukládat informace (cookies) do souboru na počítači klienta.

Kolekce Request.Cookies slouží ke čtení těchto informací uložených u klienta, pro jejich zápis se používá kolekce Response.Cookies.

Každá cookie může být jednoduchá nebo vícehodnotová. Každý prvek kolekce Request.Cookies je opět kolekcí, která má kromě standardních vlastností všech kolekcí navíc ještě atribut HasKeys.

Atribut HasKeys má hodnotu false, pokud jde o jednoduchou cookie. Jde-li o vícehodnotovou cookie, má atribut HasKeys hodnotu true.

Kolekce Form - každý prvek HTML formuláře může být pojmenován (atribut NAME), po odeslání formuláře jsou zadané hodnoty v ASP přístupné prostřednictvím tohoto jména. Podle metody odesílání hodnot formuláře (atribut METHOD) jsou hodnoty přístupné přes kolekci Request.Form nebo Request.QueryString. Některé prvky formuláře umožňují výběr více hodnot. V tom případě je možné z atributu Count zjistit počet vybraných hodnot.

Kolekce QueryString se používá pro práci s daty odeslanými metodou GET, s touto kolekcí se pracuje úplně stejně jako s kolekcí Form. Rozdíl je jen ve způsobu odesílání dat od klienta na server. Hodnoty z formuláře odesílaného metodou GET se předávají v koncové části URL adresy.

Kolekce Request.ServerVariables obsahuje všechny HTTP hlavičky odeslané klientem v HTTP požadavku a navíc ještě některé systémové proměnné, týkající se daného webového serveru.

Všechny proměnné jsou pouze pro čtení (tzn. jejich hodnotu nelze měnit). Hodnotu proměnné zjistíme příkazem:

hodnota = Request.ServerVariables(jméno_proměnné_HTTP)

6. 2. 4 Objekt Response

Objekt Response obsahuje metody pro vytváření odpovědi odesílané klientovi.

Metoda AddHeader slouží k vytváření nebo změně hodnoty HTTP hlaviček. Všechny hlavičky musí být odeslány dříve, než je odeslán jakýkoli jiný výstup. Hlavičky přidané metodou AddHeader již nelze dodatečně odstranit. Názvy hlaviček nesmějí obsahovat podtržítka, protože by bylo nesprávně interpretováno jako pomlčka.

Metoda AppendToLog přidá zadaný text na konec logového souboru webserveru.

Metoda BinaryWrite zapíše obsah zadaného parametru do výstupního HTTP proudu, aniž by byla prováděna libovolná konverze znaků. Tato metoda je vhodná například pro odesílání binárních dat při generování souboru s obrázkem a podobně.

Metoda Clear vymaže obsah vyrovnávací paměti používané pro odesílání generované stránky. Tato metoda se často používá společně s metodou Response.End, chceme-li zabránit odeslání obsahu bufferu klientovi. Pokud není atribut Response.Buffer nastaven na hodnotu True, způsobí použití metody Clear vyvolání runtime chyby. Metoda Clear také nezruší odeslané HTTP hlavičky. Metoda Clear dokáže zrušit pouze obsah vygenerovaný po posledním použití metody Response.Flush.

Metoda End způsobí, že ASP přestane zpracovávat skript stránky a odešle dosud vygenerovaný obsah. Další příkazy skriptu již nejsou zpracovány.

Metoda Flush způsobí odeslání obsahu bufferu klientovi. Pokud není atribut Buffer nastaven na hodnotu True, dojde při volání metody k runtime chybě.

Zkratka PICS znamená Platform for Internet Content Selection. Jedná se o technologii umožňující označování stránek informací o typu obsahu. Označení PICS v sobě obsahuje například informace o množství násilí, nahoty, sexu a o stupni vulgárnosti jazyka použitého ve stránce..

Metoda Redirect odešle klientovi HTTP hlavičku "302 Object Moved", čímž způsobí přesměrování klienta na zadanou URL adresu.

Metoda Write zapíše zadaný text do výstupního HTTP proudu, čímž se text stane součástí obsahu odesílaného klientovi.

Nastavování hodnot atributů objektu Response (kromě atributu IsClientConnected) ovlivňuje HTTP hlavičky odesílané klientovi. Proto je nutné tyto atributy nastavit před odesláním prvního znaku klientovi.

Atribut Buffer - určuje, zda má ASP používat vyrovnávací paměť nebo zda má být veškeré informace klientovi odesílány přímo. Hodnota tohoto atributu musí být nastavena ještě před odesláním HTTP hlaviček.

Atribut CacheControl - určuje, zda může proxy server ukládat vygenerovanou stránku do své paměti cache.

Atribut Charset - slouží pro nastavení znakové sady dokumentu a je přidávána do HTTP hlavičky Content-Type. Tím je zajištěno správné zobrazení stránky v prohlížeči.

Atribut ContentType - slouží pro nastavení HTTP hlavičky Content-Type, která obsahuje informaci o typu dokumentu (tzv. MIME typ). Implicitní hodnota je "text/html".

Atribut **Expires** – v případě, že jsme povolili nastavením atributu **CacheControl** ukládání stránky do cache, obsahuje tento atribut délku platnosti stránky v minutách. Do vypršení platnosti je klientovi zobrazována stránka uložená v cache. Po vypršení platnosti je stránka znovu vygenerována.

Atribut ExpiresAbsolute má stejnou funkci jako atribut **Expires**. V tomto atributu se však nenastavuje doba platnosti stránky, ale přímo datum a čas, kdy skončí platnost stránky.

Atribut **IsClientConnected** - hodnota **True** znamená, že je klient stále připojen a načítá stránku, hodnota **False** znamená, že klient již přestal stránku načítat.

Atribut Status obsahuje stavovou hodnotu a zprávu, která se odešle klientovi jako informace, zda byla stránka úspěšně zpracována nebo zda došlo k nějaké chybě.

Kolekce Response.Cookies slouží pro nastavení hodnot cookies odesílaných klientovi. Pro zjištění hodnot cookies uložených u klienta se používá kolekce **Request.Cookies**.

Obdrží-li klient odpověď na HTTP požadavek, uloží cookies podle obsahu kolekce **Response.Cookies**. Každá položka kolekce **Response.Cookie** je opět kolekcí, která má kromě standardních vlastností všech kolekcí navíc ještě atributy **Domain**, **Expires**, **HasKeys**, **Path** a **Secure**.

Atribut Domain slouží pro nastavení domény, z níž bude daná cookie viditelná. Hodnotu tohoto atributu nelze číst, ale pouze nastavovat. Implicitní hodnota odpovídá doméně dané stránky.

Atribut Expires umožňuje nastavit datum ukončení platnosti dané cookie. Hodnotu tohoto atributu nelze číst, ale pouze nastavovat. Hodnota se zadává jako absolutní časový údaj.

Atribut HasKeys - hodnota **False** znamená, že se jedná o jednoduchou cookie, hodnota **True** znamená, že jde o vícedílnou cookie.

Atribut Path - hodnotou tohoto atributu můžeme ovlivnit viditelnost dané cookie. Pokud nastavíme hodnotu atributu **Path**, bude daná cookie viditelná pouze ze stránek ležících v dané cestě.

Atribut **Secure** - nastavíme-li hodnotu atributu na hodnotu **True**, bude daná cookie předávána pouze při použití zabezpečeného protokolu **HTTPS**. Na straně klienta je však obsah cookie ukládán v otevřené (nezašifrované) podobě.

6. 2. 5 Objekt Server

Objekt **Server** obsahuje metody pro převod speciálních znaků v řetězcích na jejich HTML ekvivalenty nebo na ekvivalenty vhodné pro URL adresy. Dále poskytuje metody umožňující spuštění externího skriptu. Nabízí také metodu pro vytvoření instance externích objektů.

Metoda CreateObject - vytváří instanci objektu (tj. komponenty, aplikace nebo scriptovacího objektu) a vrací referenci na vytvořený objekt. Jako parametr se zadává jedinečný identifikátor třídy (**Class ID**, např. "{clsid:2933BF90-7B36-11d2-B20E-00C04F983E60}") nebo programový identifikátor (**Prog ID**, např. "ADODB.Connection").

Metoda Execute zastaví vykonávání aktuální stránky a předá řízení stránce, jejíž URL adresa je zadána jako parametr. Po zpracování této stránky se pokračuje ve zpracovávání původní stránky. V podstatě se jedná o volání podprogramu uloženého v samostatném souboru.

Metoda GetLastError vrací referenci na objekt **ASPError**, který obsahuje informace o poslední chybě, která se na stránce vyskytla před voláním této metody.

Metoda HTML Encode vrací řetězec, v němž jsou všechny speciální znaky zadaného řetězce převedeny na své HTML entity.

Metoda MapPath vrátí pro zadanou URL adresu odpovídající fyzickou cestu v souborovém systému serveru.

Metoda Transfer se chová podobně jako metoda Execute. Jediný rozdíl je v tom, že po dokončení zpracování volané stránky se již nepokračuje ve zpracovávání původní stránky.

Metoda URLEncode vrátí řetězec, v němž jsou všechny nepovolené znaky zadaného řetězce nahrazeny svými HTTP/URL ekvivalenty.

Atribut ScriptTimeout obsahuje počet sekund, které má stránka k dispozici, než je její zpracovávání ukončeno a než je vypsané chybové hlášení. Standardní hodnota je 90 sekund, pokud nastavíme nižší hodnotu, použije se standardní hodnota.

6. 2. 6 Objekt Session

Jestliže přijde od klienta požadavek na odeslání nějaké stránky, vytvoří se tzv. session (sezení). Stránka může používat proměnné nebo objekty viditelné na úrovni session. Tak je možné předávat data mezi různými stránkami jedné aplikace. Pokud přijde od téhož klienta požadavek na jinou stránku aplikace, zpřístupní se této stránce data, která byla na serveru uložena v rámci dané session. Session zaniká při ukončení prohlížeče (všech oken), při uplynutí časového limitu nebo při zavolání metody Abandon.

Mezi hlavní výhody mechanismu session patří:

Data předávaná mezi stránkami se ukládají na serveru a mezi klientem a serverem se přenáší pouze identifikátor dané session (tzv. SessionID). Tím se oproti jiným způsobům snižuje množství přenášených dat.

Protože se data ukládají na serveru, není možná jejich modifikace ze strany klienta.

Protože se data ukládají na serveru, máme plnou kontrolu nad jejich rušením. Pokud by se data posílala jako cookie, nemůžeme mít jistotu, že se v požadovaný čas opravdu zruší. Při použití session se však zruší data na serveru a k případnému nezrušenému SessionID se na serveru nenajde odpovídající záznam.

Metoda Abandon zruší aktuální session (zruší se všechny proměnné a objekty viditelné na úrovni session). Při příštím požadavku na některou ze stránek dané aplikace se vytvoří nová session a vytvoří se nové proměnné a objekty podle obsahu souboru global.asa.

Atribut CodePage obsahuje číselný kód znakové stránky používané pro zobrazování stránky v prohlížeči. Pro zobrazování českých znaků se používá nejčastěji kódování windows-1250 (číselné označení 1250) nebo iso-8859-2 (číselné označení 28592).

Atribut LCID (locale identifier) definuje národní prostředí stránky odesílané prohlížeči. Například hexadecimální hodnota 0409 (1033 desítkově) označuje USA se symbolem měny '\$' a formátem pro datum 'mm/dd/yy'.

Atribut SessionID obsahuje identifikátor aktuální session.

Atribut Timeout - pokud od posledního požadavku v dané session uplyne doba určená tímto atributem, je tato session ukončena a objekt Session je zrušen.

Atribut Value zprostředkovává přístup k hodnotám uloženým pro danou session. Obvykle se vynechává.

Událost OnStart nastává, když klient zavolá některou ze stránek dané ASP aplikace a nemá ještě aktivní session. Událost nastane před zpracováním stránky, kterou klient požaduje. V obslužné funkci této události se provádí inicializace proměnných a objektů,

které mají být uchovávány pro danou session (tzn. i mezi voláními jednotlivých stránek, ale pouze pro daného klienta).

Událost OnEnd je vyvolána v případě ukončení session. K tomu může dojít z důvodu uplynutí doby nastavené v atributu Timeout od posledního požadavku na stránku ASP aplikace. Další příčinou vzniku události OnEnd může být explicitní ukončení session voláním metody Abandon.

Kolekce Contents obsahuje všechny proměnné a hodnoty uložené v aktuální session, kromě objektů vytvořených s využitím elementu <OBJECT>.

Kolekce StaticObjects obsahuje objekty vytvořené pro aktuální session prostřednictvím elementu <OBJECT> v souboru global.asa.

6.3 Skriptovací objekty

6.3.1 Objekt Dictionary

Tento objekt je určen pro ukládání pojmenovaných hodnot. Tyto hodnoty jsou přístupné přes svůj název (tzv. klíč). Objekt Dictionary se liší od kolekcí zejména v tom, že poskytuje také metodu pro přidávání hodnot. Protože se nejedná o vnitřní objekt ASP, je nutné před jeho použitím vytvořit jeho instanci. ProgID, které potřebujeme znát pro vytvoření instance je "Scripting.Dictionary".

Metoda Add přidá do objektu Dictionary zadanou hodnotu se zadaným klíčem.

Metoda Exists vrací true pokud objekt obsahuje hodnotu se zadaným klíčem, v opačném případě vrací hodnotu false.

Metoda Items vrací pole obsahující všechny hodnoty z objektu Dictionary.

Metoda Keys vrací pole všech klíčů z objektu Dictionary.

Metoda RemoveAll odstraní z objektu Dictionary všechny hodnoty.

Atribut CompareMode určuje způsob porovnávání klíčů. Hodnota 0 (BinaryCompare) znamená, že klíče budou porovnávány binárně (tzn. bude brán ohled na velikost písmen). Hodnota 1 (TextCompare) znamená, že při porovnávání klíčů nebude brán ohled na velikost písmen. Aby bylo možné použít místo hodnot 0 a 1 konstanty, musí stránka (nebo příslušný soubor global.asa) obsahovat následující direktivu (pro WinNT).

```
<!-- METADATA TYPE="TypeLib" FILE="C:\WinNT\System32\scrrun.dll" -->
```

Atribut Count obsahuje počet hodnot uložených v objektu Dictionary.

slouží pro přístup k hodnotám uloženým v objektu Dictionary. K hodnotám se přistupuje prostřednictvím klíče.

Atribut Key je přístupný pouze pro zápis a slouží pro přejmenování klíče v objektu Dictionary.

6.3.2 Objekt FileSystemObject

Objekt FileSystemObject zprostředkovává přístup k souborovému systému serveru. Obsahuje kolekci Drives všech disků na serveru. Přes tuto kolekci získáme přístup ke všem složkám a souborům na serveru. Protože objekt FileSystemObject není vnitřním objektem ASP, musíme nejdříve získat jeho instanci. Instanci získáme voláním metody Server.CreateObject:

```
objFSO = Server.CreateObject("Scripting.FileSystemObject")
```

Metoda BuildPath přidá zadaný soubor nebo složku do existujícího řetězce s cestou. V případě potřeby vloží oddělovač '\'.

Metoda CopyFile zkopíruje soubor do zadané složky.

Metoda CopyFolder zkopíruje soubor do zadané složky.

Metoda CreateTextFile vytvoří a otevře soubor a vrátí objekt TextStream.

Metoda DeleteFile umožňuje smazat zadaný soubor.

Metoda DeleteFolder umožňuje smazat zadanou složku.

Metoda DriveExists vrací hodnotu true, pokud se zadaný disk vyskytuje v souborovém systému. V opačném případě vrací hodnotu false.

Metoda FileExists vrací hodnotu true, pokud zadaný soubor existuje. V opačném případě vrací hodnotu false.

Metoda FolderExists vrací hodnotu true, pokud zadaná složka existuje. V opačném případě vrací hodnotu false.

Metoda GetAbsolutePathName vrací absolutní cestu odvozenou od aktuální složky a od zadané relativní cesty k souboru nebo složce.

Metoda GetBaseName vrací pro zadanou cestu k souboru jméno souboru bez cesty a bez přípony.

Metoda GetDrive vrací objekt Drive představující zadaný disk.

Metoda GetDriveName vrací označení disku.

Metoda GetExtensionName vrací příponu souboru určenou ze zadané cesty.

Metoda GetFile vrací objekt File představující zadaný soubor. Zadává se cesta k souboru a není tedy nutné procházet postupně několik kolekcí Folders.

Metoda GetFileName vrací k zadané cestě jméno souboru nebo jméno poslední složky v cestě.

Metoda GetFolder vrací objekt Folder představující zadanou složku.

Metoda GetParentFolderName vrací jméno složky nadřazené zadanému souboru nebo složce.

Metoda GetSpecialFolder vrací objekt Folder představující zadanou speciální složku.

Metoda GetTempName vrací náhodně vygenerované jméno, které může být použito pro vytvoření dočasné složky nebo souboru.

Metoda MoveFile slouží k přesunu nebo přejmenování souboru.

Metoda MoveFolder přesune nebo přejmenuje zadanou složku.

Metoda OpenTextFile otevře soubor a vrátí objekt TextStream, který slouží pro čtení ze souboru, zápis do souboru nebo přidávání na konec souboru.

Atribut Drives vrací kolekci Drives obsahující objekty Drive představující všechny disky v souborovém systému.

6. 3. 3 Objekt Drive

Tento objekt je určen pro zjišťování informací o disku na serveru. Protože se nejedná a vnitřní objekt ASP, je nutné před jeho použitím získat jeho instanci. Instanci objektu Drive zastupující disk C získáme posloupností příkazů:

```
Set objFSO = Server.CreateObject("Scripting.FileSystemObject")
    'Objekt souborového systému
Set kolDrives = objFSO.Drives
    'Kolekce Drives všech disků v systému
Set objDrive = kolDrives.Item("C")
    'Objekt Drive zastupující disk C
```

Atribut AvailableSpace obsahuje údaj o množství místa (v bytech) dostupného aktuálnímu uživateli. Hodnota atributu zohledňuje kvótu daného uživatele a případná další omezení.

Atribut DriveLetter obsahuje písmeno označující disk, pro nějž byla vytvořena tato instance objektu Drive, například "C".

Atribut DriveType se používá pro zjištění typu disku.

Atribut FileSystem – hodnota odpovídá souborovému systému disku. Může nabývat například hodnot "FAT", "NTFS" nebo "CDFS".

Atribut FreeSpace obsahuje informaci o velikosti volného diskového prostoru dostupného aktuálnímu uživateli.

Atribut IsReady obsahuje logickou hodnotu indikující, zda je zařízení dostupné.

Atribut Path obsahuje cestu k zařízení ve tvaru řetězce, který obsahuje písmeno zařízení následované dvojtečkou. Například "C:".

Atribut RootFolder vrací objekt Folder reprezentující kořenovou složku disku.

Atribut SerialNumber obsahuje sériové číslo, které jednoznačně identifikuje daný disk (i logický).

Atribut ShareName obsahuje název disku, přes který je dostupný v síti (ve formátu UNC).

Atribut TotalSize obsahuje informaci o celkové kapacitě disku.

Atribut VolumeName se používá pro zjištění názvu disku. Nastavením atributu na novou hodnotu je možné disk přejmenovat.

Kolekce Drives – na rozdíl od většiny běžných kolekcí má kolekce Drives pouze dva atributy a žádné metody nebo události.

Atribut Count - udává počet disků dostupných na serveru.

Atribut Item - vrací objekt Drive zadaného disku.

6.3.4 Objekt Folder

Tento objekt představuje složku (adresář) v souborovém systému serveru. Zprostředkovává přístup k informacím o složce a poskytuje metody pro manipulaci se složkami. Protože se nejedná o vnitřní objekt ASP, je nutné před jeho použitím získat jeho instanci. Instanci objektu Folder můžeme získat čtyřmi způsoby:

- přes atribut RootFolder objektu Drive,
- přes atribut ParentFolder objektu File nebo jiného objektu Folder,
- jako prvek kolekce SubFolders objektu Folder,
- voláním metody GetFolder objektu FileSystemObject.

Metoda Copy zkopíruje složku a vše, co obsahuje, do jiné zadané složky.

Metoda Delete umožňuje smazat prázdnou složku. Tato metoda má nepovinný parametr, který určuje, zda má být složka smazána i v případě, že je určena pouze pro čtení.

Metoda Move slouží k přesunu nebo přejmenování složky.

Metoda CreateTextFile vytvoří a otevře soubor a vrátí objekt TextStream, který slouží pro čtení ze souboru, zápis do souboru nebo přidávání na konec souboru.

Atribut Attributes slouží pro zjištění a nastavování systémových atributů složky.

Atribut DateCreated vrací datum a čas vytvoření složky.

Atribut DateLastAccessed vrací datum a čas posledního přístupu ke složce.

Atribut DateLastModified obsahuje datum a čas poslední modifikace složky.

Atribut Drive vrací objekt Drive reprezentující disk, na němž je složka umístěna.

Atribut Files vrací kolekci Files obsahující všechny objekty File pro všechny soubory ve složce (kromě souborů v dalších podsložkách).

Atribut IsRootFolder má hodnotu true, pokud objekt představuje kořenovou složku. V opačném případě má hodnotu false.

Atribut Name vrací jméno složky. Nastavením hodnoty atributu způsobíme přejmenování složky.

Atribut ParentFolder vrací objekt Folder, reprezentující nadřazenou složku.

Atribut Path vrací celou cestu ke složce. V cestě se mohou vyskytovat tzv. dlouhé názvy.

Atribut ShortName vrací krátké jméno složky ve stylu operačního systému MS-DOS.

Atribut ShortPath obsahuje absolutní cestu k složce. Tato cesta obsahuje pouze tzv. krátké názvy složek ve formátu 8.3.

Atribut Size udává velikost složky v bytech. Do velikosti složky jsou započítány velikosti všech souborů a podsložek dané složky.

Atribut SubFolders vrací kolekci Folders obsahující objekty Folder pro každou podsložku nacházející se v dané složce.

Atribut Type vrací textový popis typu složky, pokud je znám. Atribut může mít například hodnotu "File Folder", "System Folder" nebo "Recycle Bin".

Kolekce Folders – na rozdíl od většiny běžných kolekcí má kolekce Folders pouze dva atributy a žádné metody nebo události.

Atribut Count udává počet podsložek ve složce.

Atribut Item vrací objekt Folder reprezentující složku se zadaným jménem.

6.3.5 Objekt File

Tento objekt představuje soubor v souborovém systému serveru. Zprostředkovává přístup k informacím o souboru a poskytuje metody pro manipulaci se soubory. Protože se nejedná o vnitřní objekt ASP, je nutné před jeho použitím získat jeho instanci. Instanci objektu File můžeme získat dvěma způsoby. Buď jako prvek kolekce Files v objektu Folder nebo přímo z objektu FileSystemObject metodou GetFile.

Metoda Copy zkopíruje soubor do zadané složky.

Metoda Delete umožňuje smazat soubor. Tato metoda má nepovinný parametr, který určuje, zda má být soubor smazán i v případě, že je určen pouze pro čtení. Pokud parametr neuvedeme, použije se implicitní hodnota false, která znemožní smazání souborů určených pouze pro čtení.

Metoda Move slouží k přesunu nebo přejmenování souboru.

Metoda OpenAsTextStream otevře soubor a vrátí objekt TextStream, který slouží pro čtení ze souboru, zápis do souboru nebo přidávání na konec souboru.

Atribut Attributes slouží pro zjištění a nastavování systémových atributů souboru.

Atribut DateCreated vrací datum a čas vytvoření souboru.

Atribut DateLastAccessed vrací datum a čas posledního přístupu k souboru.

Atribut DateLastModified obsahuje datum a čas poslední modifikace souboru.

Atribut Drive vrací objekt Drive reprezentující disk, na němž je soubor umístěn.

Atribut Name vrací jméno souboru. Nastavením hodnoty atributu způsobíme přejmenování souboru.

Atribut ParentFolder vrací objekt Folder, reprezentující složku, v níž je soubor umístěn.

Atribut Path vrací celou cestu k souboru. V cestě se mohou vyskytovat tzv. dlouhé názvy.

Atribut ShortName vrací krátké jméno souboru ve stylu operačního systému MS-DOS.

Atribut ShortPath obsahuje absolutní cestu k souboru, která obsahuje pouze tzv. krátké názvy složek ve formátu 8.3 popsaném u předchozího atributu.

Atribut Size udává velikost souboru v bytech.

Kolekce Files – na rozdíl od většiny běžných kolekcí má kolekce Files pouze dva atributy a žádné metody nebo události. Kolekci Files můžeme získat přes atribut Files objektu Folder.

Atribut Count udává počet souborů ve složce.

Atribut Item vrací objekt File reprezentující soubor se zadaným jménem.

6.3.6 Objekt TextStream

Objekt TextStream zprostředkovává přístup k obsahu textových souborů uložených na disku. Používá se pro čtení z textových souborů nebo pro zápis do textových souborů. Instanci objektu TextStream můžeme získat několika způsoby:

- voláním metody CreateTextFile objektu Folder,
- voláním metody CreateTextFile objektu FileSystemObject,
- voláním metody OpenTextFile objektu FileSystemObject,
- voláním metody OpenAsTextFile objektu File.

Metoda Close slouží k uzavření otevřeného souboru. Při volání této metody je objekt zrušen a již nelze dále přistupovat k jeho atributům a metodám.

Metoda Read přečte ze souboru zadaný počet znaků a vrátí je jako řetězec. Soubor před tím musí být samozřejmě otevřen v režimu čtení. Pokud bude soubor otevřen v jiném režimu, dojde k chybě.

Metoda ReadAll přečte celý obsah souboru a vrátí ho jako řetězec. Soubor před tím musí být samozřejmě otevřen v režimu čtení. Pokud bude soubor otevřen v jiném režimu, dojde k chybě.

Metoda ReadLine přečte ze souboru jednu řádku a vrátí ji jako řetězec. Soubor před tím musí být samozřejmě otevřen v režimu čtení. Pokud bude soubor otevřen v jiném režimu, dojde k chybě.

Metoda Skip přeskočí zadaný počet znaků v souboru. Zadaný počet znaků může být kladný nebo záporný, v tom případě se ukazatel pozice v souboru vrátí o patřičný počet znaků zpět. Metodu Skip lze použít pouze pro soubory otevřené v režimu čtení.

Metoda SkipLine přesune ukazatel pozice v souboru na začátek další řádky. Tato metoda může být použita pouze u souborů otevřených v režimu čtení. Pro zapsání prázdných řádek do souboru složí metoda WriteBlankLines.

Metoda Write zapíše zadaný řetězec do souboru. Pokud byl soubor otevřen v režimu ForWriting, zapisují se řetězce postupně od začátku souboru. Byl-li soubor otevřen v režimu ForAppending, přidávají se zapisované řetězce na konec původního souboru.

Metoda WriteLine zapíše do souboru zadaný řetězec (pokud je nějaký zadán) a poté do souboru vloží znak konce řádky.

Metoda WriteBlankLines zapíše do souboru zadaný počet znaků konce řádky.

Atribut AtEndOfLine vrací hodnotu true, pokud je ukazatel pozice v souboru nastaven hned před znakem konce řádky. Tento atribut lze použít pouze u souborů otevřených v režimu čtení, při použití v jiných režimech nastane chyba.

Atribut AtEndOfStream vrací hodnotu true, pokud ukazatel pozice v souboru ukazuje na konec souboru. Tento atribut lze použít pouze pro soubory otevřené v režimu čtení. Při použití v ostatních režimech dojde k chybě.

Atribut Column – hodnota odpovídá číslu znaku v řádce, který bude první přečten při následujícím čtení. Znaky v řádce jsou číslovány od jedničky.

Atribut Line - hodnota odpovídá číslu řádky, z níž se bude číst první znak při následujícím čtení. Řádky jsou číslovány od jedničky.

6. 4 ASP versus PHP

V ASP je možné programovat v libovolném skriptovacím jazyce, pro nějž máme na serveru k dispozici patřičný překladač. Standardní součástí serveru IIS (nebo odlehčené verze zvané Personal Web Server) jsou překladače pro jazyky JavaScript a VBScript.

Jazyk PHP neboli „Hypertextový Preprocessor“ je skriptovací jazyk běžící na straně serveru. Slouží především k tvorbě dynamických aplikací, které poskytují interaktivní přístup k informacím všeho druhu. Umožňuje vývoj nejrůznějších programů, počínaje jednoduchým počítačem přístupů a konče virtuálním Internetovým obchodem.

Na počátku zrodu PHP stál v roce 1994 Rasmus Lerdorf. Původně měl tento systém sloužit pro evidování přístupů k jeho stránkám, postupem času se ale stal oblíbeným stále většinu počtu uživatelů, kteří ho začali používat. Autor proto systém rozšířil, doplnil o dokumentaci a uvolnil jej pod názvem „Page Construction Kit“. Celosvětovou proslulost si získal systém PHP/FI 2.0. Tento systém vznikl spojením předchozího programu a nástroje, který umožňoval začleňování SQL-dotazů do stránek. Další verzí bylo PHP 3.0. Tato novější verze byla rozšířena o mnoho dalších funkcí a zároveň přepracována tak, aby celý systém byl rychlejší. Nejnovější verzí je pak PHP 4.0, jehož nové jádro nese název Zend.

PHP skripty se začleňují do HTML stránek. Aby server poznal, že se nejedná o obyčejné HTML soubory, ukládáme je s příponou *.php, *.php3, *.phtml. Vždy, když server obdrží požadavek na soubor končící s touto koncovkou, předá jej systému PHP. Ten provede všechny příkazy uložené v souboru a výsledek předá zpět serveru, který výslednou stránku zobrazí klientovi. Výsledkem běhu PHP skriptu je obyčejný HTML dokument, který umí zobrazit každý prohlížeč. Tímto odpadají problémy s kompatibilitou, které přináší Java a JavaScript.

Výhodou PHP je možnost integrace s mnoha databázovými systémy a podpora protokolů jakými jsou IMAP, SNMP, NNTP, POP3, HTTP a mnoha dalších. Jazyk PHP je nezávislý na jednotlivých platformách. Dnes jsou k dispozici verze PHP pro Windows, Unix a Macintosh. Dalším plusem jazyka, je jeho volná šířitelnost, která ho favorizuje před konkurenčním programovacím jazykem ASP. Jazyk ASP je komerčním produktem firmy Microsoft.

Přestože jde o dvě konkurenční technologie od různých výrobců, mají podle mého názoru v podstatě stejné možnosti. Na první pohled je vidět, že ASP je spíše objektově orientované a PHP je realizováno velkým množstvím samostatných funkcí.

7 IIS (Internet Information Server)

Produkt Internet Information Server od společnosti Microsoft je dnes jeden z nejrozšířenějších nástrojů pro provozování internetových aplikací a služeb, do nichž jsou zařazeny především služby WWW, FTP a SMTP server. Nejdůležitější z nichž je samozřejmě služba WWW, umožňující provozovat na serveru či i na osobním počítači plnohodnotný web server.

Odlehčená verze IIS 5.0 je součástí Windows 2000 Professional, proto se tento produkt dostane i běžným uživatelům a bez problémů lze provozovat Intranet na vnitřní podnikové síti či v domácnosti.

Ve Windows 98 je možno provozovat web server prostřednictvím produktu Personal Web Server, což je velmi odlehčená verze IIS a je poskytována zcela zdarma. K jeho správě se využívá nástroje Správa osobního serveru WWW. Ten ovšem nenabízí příliš mnoho možností, pouze přidávání virtuálních adresářů do struktury web serveru a to je celkem vše. Nelze nastavovat možnosti jednotlivých složek či souborů na web serveru. Nástroj Správce osobního WWW serveru je k dispozici také v IIS 5.0 ve Windows 2000, ovšem je zde pouze pro snadnější přechod na konzoli IIS.

IIS se ovládá pomocí programu Microsoft Management Console, což je administrační prostředek pro společnou správu celého systému Windows a ostatních aplikací. Vše je tedy možno spravovat z jednoho místa a podobným způsobem. Pomocí konzole IIS můžeme spravovat více počítačů, na začátku zde ovšem najdeme pouze správu místního počítače. K ostatním počítačům se můžeme dále připojit. Rozbalením stromové struktury u místního počítače se zobrazí jednotlivé služby v rámci IIS. Jsou to FTP server, WWW server a SMTP server. Všechny jsou pojmenovány Výchozí WWW server, Výchozí FTP server a Výchozí SMTP server. Jejich názvy si můžeme pozměnit, což nemá vliv na jejich funkčnost, pod tímto názvem dále vystupují v síti. Kliknutím pravým tlačítkem myši na název místního počítače se rozbalí menu pro správu celé služby IIS, běžící na místním počítači.

Vybráním volby „Připojit“ je možno připojit se ke vzdálenému počítači, na němž běží služba IIS, a takto službu na dálku spravovat. Do dialogového okna lze zadat buď název počítače nebo jeho IP adresu. Takto lze počítač spravovat nejen v rámci vnitřní sítě, ale i přes Internet. Připojený počítač se zobrazí v konzoli a lze pracovat s jeho službami. Volba „Odpojit“ odpojí počítač od konzole a ukončí se tím tedy správa služeb IIS tohoto počítače.

Vzhledem k tomu že se další konkrétní služby a aplikace podstatně liší v závislosti na použité verzi IIS a vzhledem k dodržení stanoveného rozsahu této práce nebude detailnější popis těchto služeb a aplikací uveden.

8 ADO (ActiveX Data Objects)

Zkratka ADO znamená ActiveX Data Objects. Jedná se o programové rozhraní (API) pro přístup k databázovým serverům, využívající možností ActiveX a tím i technologie COM. ADO je postaveno nad OLE DB a tím i nad ODBC, takže je použitelné pro všechny DBMS podporující ODBC. Díky OLE DB ale podporuje i přístup k jiným než SQL databázím, od textových souborů až po různé heterogenní databáze.

ADO je objektově orientované, tj. nejedná se o sadu funkcí, ale o sadu objektů a jejich metod. Tato sada objektů je navržena tak, aby její použití bylo jednodušší než u značně rozsáhlého OLE DB. Kromě toho se v případě ADO jedná o Automation, a ne pouze o COM, takže jeho použití je univerzálnější.

8.1. Základní objekty ADO

8.1.1 Connection

První objektem, který je v ADO nutné použít, je Connection. Tento objekt obhospodařuje hlavně připojení k databázi (určení OLE DB datového zdroje a providera). Dále obsahuje sadu funkcí pro správu transakcí (begin, commit, rollback) a kolekci Errors pro evidování případných chybových stavů ADO. Nejdůležitější metodou objektu Connection je Open, která jej připojí k zadanému datovému zdroji. Datový zdroj je nutno uvést v prvním parametru metody, tzv. ConnectionString. Další parametry této metody jsou nepovinné a je v nich možné specifikovat uživatelské jméno a heslo pro připojení a to, zda připojení bude synchronní či asynchronní. Veškeré další informace nutné pro specifikaci připojení obsahuje ConnectionString. Skládá se ze seznamu dvojic typu

parametr=hodnota.

Každá tato dvojice je oddělená středníkem. Příkladem ConnectionStringu může být hodnota

"Provider=Microsoft.Jet.OLEDB.3.51;Data Source=c:\demo\nwind.mdb;"

kteřá zajistí připojení k databázi s názvem Nwind.mdb pod DBMS Microsoft Access (Pro MS Access je součástí identifikace databáze i určení adresáře, ve kterém se nachází její soubor). Nejdůležitějším parametrem je Provider, určující OLE DB providera. Konkrétní hodnoty pro konkrétní DBMS je možné najít v dokumentaci ADO, případně u méně rozšířených DBMS v dokumentaci poskytnuté příslušným providere.

ADO použije z ConnectionStringu pouze parametry Provider, File Name, Remote Provider a Remote Server. Všechny ostatní parametry předá OLE DB providerovi a jsou tedy specifické pro konkrétního providera. Parametr File Name určuje soubor s dalšími konfiguračními informacemi pro připojení, specifickými pro daného providera. Parametry Remote Provider a Remote Server se používají při připojování k aplikačnímu serveru ve třívrstvé architektuře.

Ještě před otevřením Connection je vhodné nastavit další parametry připojení v jeho kolekci Properties. Za pozornost stojí hlavně property IsolationLevel, která určuje způsob zamykání při přístupu. Defaultní hodnota je adXactReadCommitted, za pozornost stojí ještě adXactReadUncommitted, adXactRepeatableRead a adXactSerializable. IsolationLevel je možné měnit za provozu, ale nikoliv během transakce.

Connection má též metody pro správu transakcí - BeginTrans, CommitTrans, a RollbackTrans. Jejich použití je zřejmé již z názvů. Pokud to podporuje provider, je

možné používat i vnořené transakce. Connection má samozřejmě i metodu pro ukončení spojení – Close.

8. 1. 2 Recordset

Objekt Recordset slouží k práci daty. Tato data jsou typicky výsledkem nějakého dotazu do databáze, například SQL příkazu SELECT. Data se do recordsetu načtou pomocí jeho metody Open, která jako parametr dostává jednak onen databázový příkaz (ve tvaru řetězce), jednak objekt typu Connection, pomocí kterého provede vlastní komunikaci s databázovým serverem. RecordSet může obsahovat různý počet řádků, k pohybu mezi nimi slouží metoda Move, která Recordset nastaví na požadovaný záznam. Pro jednodušší použití jsou k dispozici ještě metody MoveFirst, MoveLast, MoveNext a MovePrevious, které jsou bez parametrů a odpovídají konkrétnímu použití metody Move. Aby nebylo nutné hlídat začátek a konec recordsetu pomocí integer proměnných, jsou k dispozici property BOF a EOF (begin of file, end of file). Mají-li hodnotu True, znamená to, že poslední volání některé z funkcí Move by se dostalo mimo řádky Recordsetu, a tudíž sekvenční procházení recordsetu skončilo.

Přístup k datům v Recordsetu se realizuje pomocí jeho kolekce Fields, jejíž jednotlivé prvky jsou přístupné metodou Item. Parametrem této metody je buď název pole, nebo jeho pořadí. Tak například hodnotu pole „customer_id“ v recordsetu my_rst získáme (v jazyce Visual Basic Script) výrazem

```
my_rst.Fields.Item("customer_id")
```

případně, protože Fields je implicitní property Recordsetu a Item implicitní metodou Fields, můžeme použít tento zkrácený zápis:

```
my_rst!customer_id.
```

Stejná notace se používá i při přiřazení hodnoty do pole:

```
My_rst.Fields.Item(„customer_id“) = 123
```

respektive

```
my_rst!customer_id = 123.
```

Recordset má i prostředky pro uložení změn do databáze – metodu Update. Je bez parametrů, protože RecordSet si pamatuje Connection, se kterým byl otevřen a přebírá z něho kontext, tj. způsob zamykání a aktuální stav transakce. Pro přidávání a mazání řádků se používají metody Recordsetu AddNew a Delete. Pro použití RecordSetu k aktualizaci dat je ale ještě nutné nastavit mu ještě před jeho načtením property LockType na jinou hodnotu než je implicitní adLockReadOnly. S touto hodnotou totiž není možné RecordSet aktualizovat. Pro zápis jsou použitelné hodnoty adLockPessimistic, adLockOptimistic, které se liší způsobem zamykání záznamů při update. Konkrétní implementace způsobu zamykání ale záleží na providerovi.

Recordset má ještě další konfigurovatelné vlastnosti, mezi které patří různé způsoby uložení kurzoru (na klientovi či na serveru) a výběr mezi mu okamžitou či dávkovou aktualizací dat (batch update). Pomocí property Filter je také možné z již načteného recordsetu dočasně vyřazovat řádky.

8. 2. Specializované objekty ADO

Při základním použití ADO vystačíme s již probranými dvěma objekty, Connection a Recordset. Při složitějších aplikacích je vhodné použít i další, specializované objekty.

8. 2. 1 Error

Objekt Error zachycuje informace o jedné chybě v rámci ADO operace. Protože při jedné operaci může dojít k více chybám, může objektů třídy Error existovat několik. Všechny jsou umístěny v kolekci Errors, která patří objektu Connection. Před každou další operací ADO se kolekce Errors vyčistí, takže v ní jsou informace o chybách pouze při poslední operaci. Záznam v Error je ale věcí konkrétního providera – standardní chyby ADO se řeší standardním mechanismem výjimek aktuálně použitého programovacího jazyka a v kolekci Errors jsou tedy pouze doplňkové informace.

8. 2. 2 Command

Příkazy obecně slouží k načítání, vkládání, změně a mazání dat. Objekt Command je předpřipravený příkaz, který je možné předat metodě Recordset.Open místo textového tvaru příkazu (metoda je přetížená a umožňuje více typů parametrů). Hlavní použití objektu Command je pro optimalizaci vykonání připravených příkazů.

Při použití objektu Command nejdříve vytvoříme jeho instanci, její property ActiveConnenction přiřadíme spojení, které se má pro zpracování příkazu použít, a do property CommandText nastavíme textový tvar příkazu. Pokud nyní nastavíme property Prepared na True, provede se kompilace příkazu podle specifikace konkrétního providera a výsledek se uloží opět do CommandText. Takto připravený příkaz je možné použít opakovaně (i s různými parametry, viz níže), přičemž jeho provedení by mělo být rychlejší.

Objekt typu Command se dá použít dvěma způsoby, jednak jako parametr metody Recordset.Open, jednak pomocí jeho vlastní metody Execute, která potom vrátí nově naplněný Recordset. Výhoda objektu Command vynikne při použití s dále uvedeným objektem Parameter, protože to umožní optimalizovat provedení příkazu pro obecný parametr.

8. 2. 3 Parameter

Objekt typu Command obsahuje kolekci Parameters, v níž mohou být specifikovány parametry daného příkazu. Nový objekt třídy Parametr se vytvoří voláním metody Command.CreateParameter s parametry název, typ parametru. Tato metoda vrátí ukazatel na nový objekt typu Parameter, ale nepřihadí ho do kolekce - je to proto, aby na něm bylo případně možné provést další změny. Jedná se zejména o nastavení způsobu předávání parametru v property Direction. Možné hodnoty Direction jsou mj. adParamInput (implicitně) tj. jedná se o vstupní parametr (tj. předávaný hodnotou), adParamOutput - výstupní parametr a adParamInputOutput - vstupní i výstupní, tj. jakoby předávaný odkazem. Pro provedení těchto změn se objekt parametr přidá do kolekce metodou Command.Parameters.Append. Před vykonáním parametrizovaného příkazu je pak nutné hodnotu parametru nastavit v property Value toho kterého parametru.

8. 2. 4 Collection

Collection je objekt, který obsahuje další objekty. Vlastní kolekce v ADO programátor tvořit nemůže, ale ADO jich několik používá, takže jejich popis je vhodné uvést na jednom místě. Model ADO používá celkem čtyři typy kolekcí: Errors, Parameters, Fields a Properties. Tyto kolekce jsou vždy vlastněny některým objektem

(Errors patří Connection, Parameters Commandu, Fields Recordsetu a svou kolekci Properties mají objekty Connection, Command, Recordset a Field).

Všechny kolekce mají společné to, že je v nich možné vyhledávat metodou Item, a to podle názvu a nebo čísla položky. Další společnou vlastností kolekcí je property Count, obsahující počet položek v kolekci. Jednotlivé kolekce mají ještě další specifické metody.

8. 2. 5 Property

Objekt typu Property implementuje vlastnosti ADO objektu. V ADO se rozlišují properties built-in a dynamické. Built-in property je nativní součástí ADO, dynamické properties přidává k objektu provider (resp. do jejich kolekce Properties) a tutíž existují pouze když se používá OLE DB konkrétního providera. Property při svém čtení či zapisování mohou automaticky provádět další akce, nejsou to tedy pouhé proměnné. Objektový model ADO obsahuje několik desítek různých property realizujících nastavení a konfiguraci objektů. Property ale mají rozumně nastavené defaultní hodnoty, takže až na výjimky se jimi nemusíme zabývat. Příkladem může být již uvedená property Prepared objektu Command - chová se tak, že pokud do ní přiřadíme hodnotu True, provede se okamžitě kompilace příkazu.

8. 3 Události

Většina ADO objektů má definované různé události a event handlers, neboli metody zpracovávající tyto události. Event handlers jsou dvou typů - volané před operací a po operaci. Handler volaný před operací se dá použít například ke změně kontextu, k logování komunikace se serverem apod. Handler volaný po operaci je využitelný jako upozornění na dokončení asynchronně volané operace (např. RecordSet je možné otevřít asynchronně, tj. bez čekání na načtení dat.)

Pokud to konkrétní programovací jazyk podporuje, může programátor napsat vlastní procedury které použije jako tyto event handlers.

ADO rozlišuje dvě skupiny událostí:

8. 3. 1 ConnectionEvents

Jsou události vznikající při práci s objektem Connection. Týkají se vlastního připojování, zahajování, commitování, rollbackování transakce, a také provádění příkazů. Příkladem je událost WillExecute, která vzniká těsně před provedením příkazu, nebo RollbackTransComplete, která vzniká po provedení rollbacku transakce.

8. 3. 2 RecordsetEvents

Jsou události vznikající při práci s objektem RecordSet. Týkají se navigace mezi řádky, nastavování hodnot polí apod. Příkladem můžou být události WillChangeField, WillChangeRecord či RecordSetChangeComplete. Za pozornost stojí událost FetchProgress, která při delším zpracování příkazu vzniká periodicky a informuje klienta o počtu zpracovaných řádků a o počtu celkem předpokládaných řádků. Je tedy možné ji použít k informování uživatele o průběhu zpracování složitějších příkazů.

8. 4 Rozšíření ADO

8. 4. 1 ADO Extensions for DDL and Security (ADOX)

Jedná se rozšíření určené pro Data Definition Language a aplikace zahrnující bezpečnostní hlediska (kontrola a nastavování práv a pod). ADOX obsahuje objektový model zahrnující mj. objekty Table, Column, Index, User, View - většinou jsou to objekty, které se v SQL vytváření/mění pomocí příkazů CREATE/ALTER. Dále ADOX obsahuje procedury pro zavedení a zrušení uživatelů, ověření a nastavení přístupových práv konkrétnímu uživateli a pro nastavení hesla.

Příklad použití ADOX:

Vytvoření tabulky provedeme tak, že vytvoříme ADOX objekt typu Catalog, pomocí jeho metody Create jej propojíme s konkrétní databází (parametrem metody je již známý ConnectionString), připravíme si objekt typu Table (včetně sloupců) a ten přidáme do Catalog.Tables její metodou Catalog.Tables.Append. Metoda Append zajistí vytvoření tabulky v databázi, tedy vygenerování příslušných (SQL) DDL příkazů a jejich zadání DBMS.

9 Databáze SQL Serveru (Structured Query Language Server)

Pro zápis požadavků na databázový server se nejčastěji používá jazyk SQL (Structured Query Language). Tento jazyk prošel dlouhým vývojem a v různé míře jej dnes podporují téměř všechny běžně používané databázové servery. Někdy se proto databázovým serverům říká zjednodušeně SQL servery. Jazyk SQL nabízí vše potřebné pro vytváření, modifikování a rušení tabulek a pro práci s údaji v tabulce - vyhledávání, přidávání, modifikování a mazání údajů.

V roli klienta pro SQL server může samozřejmě vystupovat i skript zapsaný v ASP nebo PHP. To znamená, že tyto skripty mohou obsahovat příkazy zapsané v jazyce SQL a zpracovávat jejich výsledky po provedení na SQL serveru.

9.1 Protokol ODBC

Každý SQL server má svůj vlastní protokol, kterým s ním může klient komunikovat. Pokud má klient umět komunikovat s více různými servery, musí podporovat více protokolů. To není zrovna nejšťastnější řešení a proto na platformě Windows vzniklo rozhraní ODBC. To slouží jako prostředník mezi klientskou aplikací a databázovým serverem. Rozhraní ODBC se volá jednotně a ODBC ovladač pak požadavek předá databázovému serveru pomocí správného protokolu.

V ASP i PHP může k připojení k SQL serveru používat rozhraní ODBC. PHP navíc nabízí pro některé servery přímou podporu - přístup k datům je pak obvykle rychlejší. Mezi přímo podporované databáze mimo jiné patří Oracle, Sybase, Solid, MySQL a PostgreSQL. ASP neobsahuje nativní databázové ovladače a přístup ke všem databázím musíme provádět přes rozhraní ODBC. Nejprve musíme nainstalovat ODBC ovladače pro náš databázový server a pak pro požadovanou databázi vytvořit datový zdroj ODBC.

Velkou výhodou ODBC je, že stejným způsobem můžeme přistupovat k libovolné databázi. Pokud se tedy z nějakého důvodu změní SQL server, na kterém běží naše aplikace, nemusíme měnit v skriptech žádný kód. Stačí upravit mapování datových zdrojů na databáze v konfiguračním programu ODBC.

Počáteční nevýhodou, která mluvila proti použití ODBC, byl nižší výkon oproti nativním ovladačům. Staré ODBC ovladače sloužily pouze jako mezistupeň mezi aplikací a nativním protokolem databáze. Novější ODBC ovladače jsou však optimalizovány a k databázovému serveru přistupují přímo - jejich výkon je srovnatelný s použitím nativních ovladačů.

Poslední otázka, která se nabízí, se týká využitelnosti ODBC na jiných platformách než jsou Windows. ODBC bylo původně vyvinuto pro platformu Windows, ale dnes jeho implementace existuje i pro všechny významnější verze operačního systému Unix.

9.2 Jazyk SQL

Jak již bylo řečeno, jazyk SQL nabízí vše potřebné pro vytváření, modifikování a rušení tabulek a pro práci s údaji v tabulce - vyhledávání, přidávání, modifikování a mazání údajů. Na tomto místě uvedeme základní příkazy jazyka SQL. Kompletní přehled příkazů jazyka SQL vzhledem k jeho rozsáhlosti a vzhledem k dodržení stanoveného rozsahu této práce uveden nebude.

9. 2. 1 Vytvoření tabulky

K vytvoření nové tabulky v databázi slouží SQL příkaz `CREATE TABLE`. Jeho syntaxe je následující:

```
CREATE TABLE jméno_tabulky (jméno_položky typ_položky, ... )
```

Názvy všech tabulek a položek je vhodné zadávat bez diakritických znamének, protože ne všechny servery si zde s češtinou poradí.

9. 2. 2 Přidání nového záznamu

K přidání nového záznamu do tabulky s N atributy slouží v SQL příkaz `INSERT INTO`:

```
INSERT INTO jméno_tabulky  
VALUES (hodnota1, ..., hodnotaN)
```

Textové řetězce musíme uzavírat do apostrofů nebo do uvozovek. Obdobným způsobem můžeme přidat i další záznamy.

9. 2. 3 Výběr a prohlížení záznamů

K výběru a vypsání záznamů z tabulky slouží příkaz `SELECT`. Pokud chceme vypsát obsah celé tabulky, zadáme příkaz:

```
SELECT * FROM jméno_tabulky
```

Informace lze vybírat i selektivně. Podmínka, která musí pro vybrané záznamy platit, se uvádí za klíčové slovo `WHERE`.

```
SELECT * FROM jméno_tabulky  
WHERE podmínka
```

9. 2. 4 Mazání záznamů v tabulce

K mazání záznamů v tabulce slouží příkaz `DELETE FROM`. Záznamy, které chceme smazat, musíme určit podmínkou v klauzuli `WHERE`:

```
DELETE FROM jméno_tabulky  
WHERE podmínka
```

9. 2. 5 Změna obsahu tabulky

Ke změně obsahu záznamu slouží příkaz `UPDATE`:

```
UPDATE jméno_tabulky SET položka=hodnota, ...  
WHERE podmínka
```

9. 3 Začleňování příkazů jazyka SQL do skriptů

ASP i PHP nabízejí pohodlný pro způsob s databázemi právě pomocí jazyka SQL. Práce s databází ve skriptu se skládá z několika kroků. Nejprve je nutné se k databázi připojit. Poté můžeme na server odeslat SQL dotaz. Zpět se nám vrátí výsledek, se kterým můžeme dále pracovat - vybírat z něj jednotlivé záznamy a položky. Dotazy a zpracování výsledků můžeme provádět opakovaně a nakonec bychom se měli od databáze odpojit.

Nyní se dostáváme k samotnému principu práce s databázemi v ASP.

Nejprve se musíme k připojit k datovému zdroji. Vytvoříme si proto objekt **spojeni**, který slouží pro připojování ke zdrojům dat:

```
Set spojeni = Server.CreateObject("ADODB.Connection")
```

K samotnému připojení slouží metoda **Open**, jejímž parametrem je identifikace datového zdroje.

Příklad připojení k datovému zdroji **test**:

```
spojeni.Open "DSN=test"
```

Pro zadávání dotazů a zpracování jejich výsledků musíme vytvořit objekt typu **Recordset**.

Příklad vytvoření objektu **vysledek**:

```
Set vysledek = Server.CreateObject("ADODB.Recordset")
```

K jednotlivým položkám aktuálního záznamu se dostaneme pomocí volání

```
vysledek.fields("jméno_položky").
```

SQL dotaz se spouští pomocí metody **Open**, která má jako druhý parametr již vytvořené spojení k požadovanému datovému zdroji. Výsledek dotazu je nyní přístupný pomocí objektu **vysledek**. Při práci s databázemi přes ODBC nemůžeme přistupovat náhodně k libovolnému záznamu výsledku, ale musíme je procházet postupně. K pohybu na další záznam výsledku slouží metoda **MoveNext**. Pomocí metody **EOF** můžeme zjistit, zda výsledek obsahuje ještě nějaké nezpracované záznamy.

10 Využití Microsoft Internet Exploreru pro realizaci návrhu

10.1 Historie Microsoft Internet Exploreru

Microsoft Internet Explorer 1, 2 a 3

Tyto tři první verze prohlížeče nepodporují DHTML.

Microsoft Internet Explorer 4

Čtvrtá generace prohlížeče od Microsoftu přišla v době, kdy začínal opravdový boom webu, v roce 1997, a byl součástí Windows 98. Nová verze Exploreru se snažila přinést řadu novinek a také mnoho doplňkových programů k samotnému prohlížeči. Poprvé byla například představena technologie Active Desktop, došlo též ke sloučení programu Mail a News do jediné aplikace a vznikl Outlook Express, prohlížeč je dále vybaven řadou multimediálních vylepšení a podobně. Co je však nejdůležitější, tato verze Exploreru byla první, která podporovala DHTML.

Microsoft Internet Explorer 5

V březnu 1999 se objevila další verze Internet Exploreru a byla součástí Windows 98 Second Edition. Při vývoji měl být kladen důraz hlavně na stabilitu a výkon. Většina novinek byla skryta v podporovaných technologiích. Byla rozšířena a vylepšena podpora kaskádových stylů a objevuje se i XML a XSL.

Microsoft Internet Explorer 5.5.

Na první pohled k nerozeznání od svého předchůdce, změny byly opět pod povrchem. Týkaly se zejména opětovného vylepšení podpory kaskádových stylů a renderingu stránek. Zároveň to byla první verze, která byla standardně vybavena 128 bitovým šifrováním.

Microsoft Internet Explorer 6

Poslední veřejně dostupná finální verze je šestou generací tohoto prohlížeče. S podporou standardů je na tom šestá verze prakticky stejně jako verze předchozí. Drobné změny se objevují s příchodem druhého Service Packu pro Windows XP. Objevuje se jednoduchý blokovač vyskakovacích oken, je zlepšena bezpečnost a správa doplňků (lepší možnost povolování a zakazování jednotlivých ActiveX komponent).

Microsoft Internet Explorer 7

Novou verzí prohlížeče dohání Microsoft to, co za posledních pár let zaspal. Zatím je sice k dispozici jen betaverze, ale několik chystaných vlastností si již můžeme prozradit teď. Konečně by se nemělo jednat o neustálé vylepšování páté generace prohlížeče, ale o radikálnější změny nejen v jádře, ale i v uživatelském prostředí. Uživatelské prostředí bylo předěláno do modernější podoby, konečně už budou mít uživatelé k dispozici prohlížení v záložkách (tabbed browsing), podporu RSS, ochranu proti phishingu a samozřejmě další zlepšení podpory webových standardů.

10.2 Využití Microsoft Internet Exploreru

Lze říci, že DHTML pracuje jako spojení JScriptu a CSS v prohlížečích Internet Explorer verze 4, 5 a 6. Na tomto místě je vhodné připomenout, že JScript je ve své podstatě přizpůsobením JavaScriptu firmou Microsoft pro prohlížeče Internet Explorer. Jiné prohlížeče DHTML nezavádějí jako pojem, většinou v nich ale fungují (jde v

zásadě o aplikaci JavaScriptu). Vzhledem k tomu, že Microsoft Internet Explorer je v dnešní době nejvyužívanějším www prohlížečem, jeví se jeho použití při realizaci návrhu dynamických HTML stránek jako optimální, a to i přes neustálou nekázeň Microsoftu při dodržování standardů konsorcia W3C. Jak je však uvedeno v následující kapitole, neznamena to, že použití JScriptu (případně VBScriptu) a CSS v prohlížečích Internet Explorer je nejlepší a už zdaleka ne jedinou cestou.

11 Možnosti dalšího rozšíření návrhu

Dynamické HTML umožňuje vytvářet interaktivní stránky. To je možná důvod, proč je porovnáváno s Javou - pomocí Java appletů vložených do stránky můžeme dosáhnout podobných efektů. Nesmíme však zapomenout, že Java je mnohem více. V Javě lze psát zcela samostatné plnohodnotné aplikace, které ke svému běhu nepotřebují prohlížeč. S uživatelem komunikují pomocí dialogových oken, která si otevírají přímo v grafickém rozhraní hostitelského systému. Aplikace v Javě jsou samozřejmě zcela přenositelné mezi různými platformami. Stejná aplikace může bez úprav běžet ve Windows, MacOS nebo v Unixu.

Porovnejme však nejdůležitější rysy Javy a dynamického HTML. Programy v Javě musí být před svým používáním zkompileovány do binární podoby (tzv. byte code), který je přenositelný mezi platformami. Tento byte code lze velice snadno převést přímo do strojového kódu libovolného procesoru - o tento proces stará virtuální stroj Javy. Oproti tomu se v dynamickém HTML používají skripty, které jsou interpretované - prohlížeč je načte spolu se stránkou, louská jeden příkaz za druhým a provádí je. Interpretovaný kód je samozřejmě mnohem pomalejší než kód předem zkompileovaný. Z tohoto důvodu nejsou skripty vhodné pro použití při náročnějších výpočtech.

Tím, že je Java kompilovaná, přináší vývojářům další výhodu. Na Internetu stačí vystavit zkompileovanou verzi programu - zdrojové texty programu zůstanou k dispozici pouze autorovi, který se nemusí bát zcizení své práce. Skripty používané v dynamickém HTML dorazí do prohlížeče vždy ve zdrojové podobě, takže není technický problém stáhnout si je a libovolně jejich části používat na svých stránkách.

V dalším porovnávání budeme předpokládat, že dynamické HTML používá jako skriptovací jazyk JavaScript (při použití VBScriptu by to nedopadlo o mnoho lépe).

JavaScript je objektově založený jazyk - můžeme tedy používat předem vytvořené objekty. Nemůžeme si však vytvářet vlastní třídy a používat přitom dědičnost. Všechny tyto vlastnosti samozřejmě Java jako plnohodnotný objektově orientovaný jazyk nabízí.

Cílem tvůrců JavaScriptu bylo vytvořit jednoduchý a snadno použitelný jazyk. Z tohoto důvodu nemusíme u proměnných deklarovat jejich typ. To nám ušetří trochu práce, ale ve větších programech je slabá typová kontrola zdrojem častých chyb. Java oproti tomu při kompilaci kontroluje, zda jsou typy všech operandů navzájem kompatibilní.

Programy v Javě mají obecně mnohem větší možnosti ošetření chyb. V Javě lze používat velice flexibilní model výjimek, který umožňuje, aby aplikace sama napravila následky chyb vzniklých při jejím běhu. Možnosti detekce chyb jsou v JavaScriptu velmi omezené.

Java umožňuje pracovat se soubory libovolného typu. Implementace JavaScriptu v IE 4.0 (JScript) nabízí pouze poněkud nestandardní způsob pro práci s textovými soubory. Mnohem větší nedostatek však je, že v běžných skriptovacích jazycích nelze vytvářet komunikační spojení s ostatními počítači a aplikacemi pomocí protokolu TCP/IP. V Javě lze přitom psát jak klienty tak servery, které používají libovolný aplikační protokol.

Možnosti síťové komunikace jsou v Javě využity i jinde. Standardní databázové rozhraní JDBC umožňuje přístup k SQL databázím právě přes Internet.

Jak vidíme, dynamické HTML nedopadlo právě nejlépe. To však neznamená, že se jedná o špatnou technologii. Dynamické HTML i Java mají své vlastní oblasti použití, které se jen nepatrně překrývají. Budoucnost tedy nabízí dostatek prostoru pro obě dvě technologie.

Zajímavé je, že konsorcium W3C v definici objektového modelu dokumentu (DOM) myslelo i na Javu, a tak brzy půjde k dynamické změně obsahu stránky používat místo skriptovacích jazyků přímo Java applety.

12 Zhodnocení, závěr

Dynamické HTML stránky jsou protikladem statických HTML stránek. Při jejich návrhu a realizaci ovšem zdaleka nepostačíme s vývojovým prostředím pro tvorbu klasických HTML stránek, ať už se jedná o různé HTML editory či prostý Notepad. Naopak je nutné pochopit princip a použití dalších programových prostředků a vybavení, jakými jsou například CSS, ASP, PHP, JavaScript, IIS, ADO, problematika SQL a některé další.

Cílem této práce je přispět k celkovému zvládnutí problematiky návrhu a tvorby dynamických HTML stránek za použití výše zmiňovaných programových prostředků a vybavení a podat základní informace potřebné k úspěšné realizaci návrhu dynamických HTML stránek. Vzhledem k rozsáhlosti tématu a vzhledem k nutnosti dodržení stanoveného rozsahu této práce se však nebylo možno zabývat detailnějším rozpracováním jednotlivých kapitol. Pro další studium a k získání podrobnějších informací k problematice je určen seznam literatury, uvedený v závěru práce.

Seznam literatury

LEE, James, WARE, Brent: OpenSource - vývoj webových aplikací, Computer Press; 2003

AVERY, James: Microsoft ASP.NET konfigurace a nastavení, Computer Press; 2004

GILMORE, W. Jason: Velká kniha PHP 5 a MySQL, Zoner Press; 2005

STANÍČEK, Petr: CSS Kaskádové styly, Computer Press; 2003

Internet:

<http://www.w3c.org>

<http://www.microsoft.com>

<http://www.kosek.cz>

<http://www.interval.cz>

<http://home.pf.jcu.cz/~pepe>

<http://www.zive.cz>

<http://www.developer.sk>