PEDAGOGICKÁ FAKULTA JIHOČESKÉ UNIVERZITY
V ČESKÝCH BUDĚJOVICÍCH

Katedra fyziky

Moderní metody výuky
Elektronická učebnice principů počítačů

Bakalářská práce

Vedoucí bakalářské práce

Ing. Michal Šerý

Eva Blažková, České Budějovice 2007

Cílem této práce bylo vytvoření elektronické verze učebnice na téma Principy počítačů a popsání metody tvorby tohoto materiálu prostředky (X)HTML. Vytvořená učebnice je vhodná jak pro žáky tak i pro učitele, látka je určena jak pro uživatele začátečníky tak i pro zkušenější. Jednotlivé části materiálu jsou nabízeny prostřednictvím výsuvné nabídky. První „roletka“ je věnována historii výpočetní techniky a Internetu, v dalších je postupně popsána skříň počítače a periferní zařízení, následuje stručný výčet a rozdělení programového vybavení. Poslední „roletka“ provádí klasifikaci počítačových sítí. Na závěr je zařazen stručný počítačový slovníček, který vychází z dnes frekventovaných pojmů, používaných zejména na Internetu.

Informace pro výukový materiály byly čerpány jednak klasicky z literatury, ale především z Internetu. Vzhledem k tomu, že se jedná o oblast, která se velmi rychle vyvíjí a mění, což se týká též webových adres, mohou se odkazy uváděné v seznamu literatury a www stránek změnit.

V práci se nevěnuji popisu technického vybavení počítačů určených pro spouštění výukového materiálu, neboť počítače, které se používají dnes na školách, jsou již na takové úrovni, že bez problémů vystupují ve výuce v roli didaktické pomůcky. Navíc pro pouhé demonstrační účely takovýchto materiálů úplně stačí jeden počítač umístěný ve třídě s reproduktory a připojený digitální projektor.

Do budoucna lze předpokládat další dynamický rozvoj v oblasti elektronických výukových materiálů nejen ve výuce „informatiky“, neboť programové vybavení poskytuje učitelům dostatek možností pro jejich tvoru a Internet dost inspirace.

Annotation

The aim of this work is to create an electronic textbook on the topic Computer principles and to describe a method of this creation, based on (X)HTML. The resulting textbook is intended for students as well as for teachers, for beginners as well as for advanced. Particular topics are offered by pop-up menus. The first menu concerns the history of computers and Internet. Other menus are consequently devoted to the computer case, peripheral equipment and software specification. The last menu classifies computer networks. A brief computer vocabulary is added at the end of the work. The vocabulary contains contemporary notions and terms frequently used mainly in the Internet.

Děkuji Ing. Michalu Šerému, který vedl mou bakalářskou práci, za podnětné rady a návrhy.

„Prohlašuji, že jsem tuto bakalářskou práci vypracovala samostatně a že jsem veškerou použitou literaturu uvedla v Seznamu použité literatury“

_____

Obsah

1 Úvod

V oblasti materiálních didaktických prostředků dochází v poslední době k expanzivnímu rozvoji a do výuky se dostávají vedle dosavadně užívaných učebních pomůcek a didaktické techniky i počítače a počítačové programy. Běžně se setkáváme s využíváním počítačů při výuce informatiky. Žáci se zde učí vytvářet, získávat, zpracovávat a využívat informace v osobním i pracovním životě. Využití počítače při výuce ovšem přesahuje samotné „ovládání počítačových programů“ a počítače dnes nalézají uplatnění ve vyučování jako didaktická technika, a to nejen v informatice, ale i v „neinformatických“ předmětech. Podobně, jako např. dnes zcela běžný zpětný projektor lze ve výuce libovolného předmětu využít i počítač.

Mezi přednosti počítačové výuky patří zejména její individualizace, kdy si žák vybírá samostatně jednotlivé oblasti, které jsou mu ke studiu předkládány, volí vlastní tempo studia, důležitým rysem je aktivizace studenta , založená na dialogové komunikaci . Výuka je řízená, neboť dosažení cílů je povinné a rozhoduje o dalším postupu studenta. Typická je samostatnost, podporována snahou o dosažení cíle, známá z herních situací. Velmi důležitým prvkem je okamžitá zpětná vazba. Jako negativa počítačové výuky je uváděna značná pracnost přípravy a „vzdálení“ učitele od žáka.

Funkce, které může didaktický software ve výuce zajistit jsou zejména: procvičování, dialogový výklad, dialogová konzultace, simulace a modelování, didaktická hra řešení problémů, učení objevováním, testování znalostí. Je bankou výukových cílů a informací o studijních materiálech a o organizaci výuky.

Hlavním obsahem bakalářské práce bylo sestavení hypertextového dokumentu Principy počítačů a popsání metody jeho tvorby prostředky HTML. Přednosti hypertextového výukového materiálu jsou zřejmé. Je to univerzálnost použití jednak přímo z CD, jednak prostřednictvím Internetu umístěním materiálu na konkrétním webovém serveru, kam mají uživatelé přístup on-line.

Práce obsahuje dvě části. První částí je CD-disk (viz. příloha), jehož obsahem je hypertextový výukový materiál doplněný obrázky a fotografiemi. Druhou částí je návod na vytvoření HTML dokumentu, zaměřený především na využití CSS a zformulování testových otázek k ověřování vědomostí žáků.

2 Stručné seznámení s jazykem (X)HTML

2.1 Úvod

HTML je zkratka z anglického **HyperText Markup Language**, značkovací jazyk pro hypertext. Je jedním z jazyků pro vytváření stránek v systému World Wide Web, který umožňuje publikaci stránek na Internetu. Jazyk je podmnožinou dříve vyvinutého rozsáhlého univerzálního značkovacího jazyka SGML (Standard Generalized Markup Language). Vývoj HTML byl ovlivněn vývojem webových prohlížečů, které zpětně ovlivňovaly definici jazyka.

V roce 1989 spolupracovali Tim Berners-Lee a Robert Caillau na propojeném informačním systému pro CERN, výzkumné centrum fyziky poblíž Ženevy ve Švýcarsku. V té době se pro tvorbu dokumentů obvykle používal TeX, Postscript a také SGML. Berners-Lee si uvědomoval, že potřebují něco jednoduššího a v roce 1990 byl tedy navržen jazyk HTML a protokol pro jeho přenos v síti - HTTP (HyperText Transfer Protocol - přenosový protokol hypertextu).

V roce 1991 CERN zprovoznil svůj web. Současně NCSA (National Center for Supercomputer Applications) vybídlo Marca Andreessena a Erica Binu k vyvinutí prohlížeče Mosaic. Byl vyvinut v roce 1993 pro počítače PC a Macintosh a měl obrovský úspěch. Byl to první prohlížeč s grafickým uživatelským rozhraním. Došlo k velkému rozvoji webu a bylo nutné pro HTML definovat standardy.

2.1.1 Verze jazyka

- **Verze 2.0** Zachycuje stav jazyka v polovině roku 1994. Standard vydala komunita IETF (Internet Engineering Task Force). Je to první verze, která odpovídá syntaxi SGML. Přidává k původní specifikaci interaktivní formuláře.
- **Verze 3.2** Byla vydána v květnu 1996 a zachycuje stav jazyka v tomto roce. Připravovaná verze HTML 3.0 nebyla nikdy přijata jako standard, protože byla příliš složitá a žádná firma nebyla schopna naprogramovat její podporu ve svém prohlížeči. Standard už vydalo konsorcium W3C (World World Wide Web Consortium), stejně jako následující verze. Přidává k jazyku tabulky, zarovnávání textu a stylové elementy pro ovlivňování vzhledu.
- **Verze 4.0** Byla vydána 18. prosince 1997 konsorciem W3C. Do specifikace jazyka přidala nové prvky pro tvorbu tabulek, formulářů a nově byly standardizovány rámy (frames). Tato verze se snaží dosáhnout původního účelu - **prvky by měly vyznačovat význam (sémantiku) jednotlivých částí dokumentu, vzhled má být ovlivňován připojovanými styly**. Některé prezentační elementy byly zavrženy. Je tím vytvořen již vcelku mocný jazyk pro definici struktury dokumentu, vzhled dokumentu nechává téměř výhradně na CSS (Cascading Style Sheets). W3C chtělo ale zároveň nechat výrobcům prohlížečů a autorům stránek čas k přechodu na tuto normu, proto rozdělilo HTML 4.0 na HTML 4.0 Strict, HTML 4.0 Transitional (přechodná norma, zachovává vzhledové atributy HTML 3.2 a zároveň obsahuje všechny výhody a vylepšení HTML 4.0 Strict, především v oblasti přístupnosti a CSS) a HTML 4.0 Frameset (tato norma se téměř shoduje s HTML 4.0 Transitional, je pouze rozšířena o elementy pro definici rámů).

- **Verze 4.01** Tato verze opravuje některé chyby verze předchozí a přidává některé nové tagy. Je to poslední verze HTML, které se již dále nevyvíjí, protože má být nahrazeno novějším XHTML, jehož základem je právě tato poslední verze HTML.

2.2 Popis jazyka

Jazyk HTML je od verze 2.0 aplikací SGML. Je charakterizován množinou značek a jejich atributů pro danou verzi definovaných. Mezi značky se uzavírají části textu dokumentu a tím se určuje význam (sémantika) obsaženého textu. Názvy jednotlivých značek se uzavírají mezi úhlové závorky ("<" a „>“). Část dokumentu uzavřená mezi značkami tvoří tzv. element (prvek) dokumentu. Součástí obsahu elementu mohou být další vnořené elementy. Atributy jsou doplňující informace, které upřesňují vlastnosti elementu.

Značky (také nazývané tagy) jsou obvykle párové. Rozlišujeme počáteční a koncové značky. Koncová značka má před názvem značky znak lomítko. Příklad pro označení odstavce:

```
<p>Text odstavce</p>
```

Některé značky jsou nepárové - neobsahují žádný text. Příklad pro vykreslení vodorovné čáry

```
<hr>
```

Tagy mohou obsahovat atributy popisující jejich vlastnosti nebo nesou jinou informaci. Příkladem může být odkaz (tag a), jehož atribut href říká, kam se uživatel po kliknutí na něj dostane

```
<a href="http://example.com">odkaz</a>
```

Pro každou verzi existuje definice pravidel DTD (Document Type Definition). Od verze 4.01 musí být odkaz na deklaraci DTD v dokumentu uveden pomocí klíčového slova DOCTYPE. DTD definuje pro určitou verzi, které elementy je možné používat a s jakými atributy.

2.3 Struktura dokumentu

Současný trend a doporučovaný postup návrhu webových stránek je ten, že by se měl na stránce **nejprve vytvořit obsah** s pomocí logických HTML značek a až v druhém kroku by se měly těmto značkám **přiřazovat CSS vlastnosti**, pomocí kterých se řekne různým zařízením, jak mají obsah uvnitř daných značek interpretovat. Jednotlivá zařízení si poté podle svých možností vyberou z nabízených vlastností a dokument podle nich naformátují. CSS vlastnosti lze tedy chápat jako něco navíc - základem je správně strukturovaný HTML dokument, který je čitelný pro jakékoli zařízení.

Dokument v jazyku HTML má předepsanou strukturu:

- Deklarace DTD - je povinná až ve verzi 4.01, je uvedena direktivou <!DOCTYPE.
- Kořenový element - element html (značky <html> a </html>) reprezentuje celý dokument. Je nepovinný, ale je doporučeno ho používat.
- Hlavička elementu - jsou to metadata, která se vztahují k celému dokumentu. Definují např. název dokumentu, jazyk, kódování, klíčová slova, popis, použitý styl zobrazení. Hlavička je uzavřena mezi značky <head> a </head>.
- Tělo dokumentu - obsahuje vlastní text dokumentu. Vymezuje se značkami <body> a </body>

2.4 Druhy značek

Značky můžeme z hlediska významu rozdělit na tři základní skupiny:

- Strukturální značky jsou ty, které rozvrhují strukturu dokumentu. Příkladem jsou odstavce (<p>), nadpisy (<h1>, <h2>). Dodávají dokumentu formu.
- Popisné (sémantické) značky popisují povahu obsahu elementu. Příklad nadpis (<title>) nebo adresa (<address>). Současný trend je orientován právě na sémantické značky, které usnadňují automatizované zpracovávání dokumentů a vyhledávání informací v záplavě dokumentů na webu. Vyvrcholením této snahy je v současné době jazyk XML.
- Stylistické značky určují vzhled elementu při zobrazení. Typickým příkladem je značka pro tučné písmo (<b>). Tento druh značek se nedoporučuje používat, trendem je používání kaskádových stylů oddělených od obsahu dokumentu. Problém je především v tom, že tyto značky jsou orientovány na obrazovku monitoru, problémy mohou nastat při používání dokumentu jiným způsobem - alternativní prohlížeče pro postižené (čtečky pro slepce), v mobilních zařízeních a podobně. Kaskádové styly umožňují definovat rozdílné zobrazení pro různá zařízení.

2.5 Parsování („syntaktická analýza“) v prohlížečích

Webové prohlížeče jsou programy, jejichž účelem je prezentovat dokument na zobrazovacím zařízení - převážně monitoru počítače. Dokument je prohlížečem načítán a rozkládán parsován (syntaktická analýza) na jednotlivé elementy. Prohlížeč obsahuje tabulku značek, které podporuje. Tuto tabulku je možné omezit typem dokumentu (DTD), je-li deklarován.

Každému elementu je poté přiřazen styl (způsob zobrazení). Styly mohou být uvedeny ve stylovém předpisu. Vlastnosti stylů, které nejsou předepsány, doplní prohlížeč podle implicitního stylu, který má zabudován. Některé prohlížeče umožňují uživateli implicitní styly definovat.

Novější prohlížeče pracují obecně ve dvou základních režimech:

- Standardní režim - je dodržována specifikace verze HTML, deklarovaná v dokumentu. Elementy v dokumentu musí odpovídat verzi, neznáme elementy jsou považovány za chybu a nezobrazují se. Netolerují se syntaktické chyby.
- Quirk mód - nestandardní režim. Prohlížeč se snaží „napravovat“ chyby v syntaxi dokumentu, domýšlí si chybějící koncové párové značky, neznámé elementy zobrazuje implicitním formátem.

Režim se stanoví obecně podle typu dokumentu, ale rozhodování je dost složité (označuje se výrazem doctype sniffing). Účelem je zachování zpětné kompatibility pro dokumenty, které nedodržují definici HTML. Tato vlastnost ale umožňuje vytvářet nestandardní dokumenty - autory zatím nic nenutí pracovat bez chyb podle specifikace.

Prohlížeče mají různé implicitní styly zobrazování a chovají se různě v quirk módu. Stejný dokument se tedy může v různých prohlížečích zobrazovat různě, rozdíly jsou i mezi verzemi prohlížeče stejného výrobce. Současné chování prohlížečů je výsledek vývoje, kdy si výrobci jednotlivých prohlížečů přizpůsobovali definici HTML podle svých potřeb a prohlížeče podporovaly nestandardní elementy a syntaxi. Řada

těchto „vylepšení“ byla následně přejímána do standardů, a některé zase v dalších verzích vyřazeny.

2.6 Budoucnost HTML

Vývoj jazyka HTML byl ukončen verzí 4.01. Dalším pokračováním vývoje jazyků pro psaní dokumentů na webu je jazyk XHTML.

Důvodů pro tuto změnu je celá řada. Překotný vývoj prohlížečů umožnil vznik obrovského množství dokumentů, které neodpovídají specifikaci HTML. Současné prohlížeče tolerují chybné značkování (neuzavřené, překřížené, či špatně vnořené elementy) a řadu dalších chyb a při parsování se s těmito chybami snaží vyrovnat („myslí“ za uživatele). Tento princip je ale příliš náročný pro nový druh aplikací, které vznikají pro malá méně výkonná zařízení - mobilní telefony, přenosné počítače (palmtopy) atd. Tyto aplikace potřebují přesně definovanou strukturu s přesnými pravidly. To poskytuje jazyk XML, ale HTML není aplikací XML a těmto novým požadavkům nevyhovuje. Pokud by však nové verze prohlížečů začaly vyžadovat přesně napsané HTML dokumenty bez chyb, pravděpodobně většina dokumentů na webu by nešla zobrazit.

V další fázi bylo proto třeba definovat HTML jako podmnožinu jazyka XML (eXtensible Markup Language - rozšiřitelný značkovací jazyk), který vychází ze SGML, avšak je „ořezán“ o některé zbytečné vlastnosti a některé nové vlastnosti naopak přidává. W3C (World Wide Web Consortium) prosazuje XML jako hlavní a jediný značkovací jazyk nejen pro web, proto byla logickým vyústěním této snahy reformulace HTML do XML.

2.7 XHTML

Nově vytvořený jazyk dostal jméno **XHTML** (eXtensible HyperText Markup Language - rozšiřitelný hypertextový značkovací jazyk) a jeho první specifikace se označuje **XHTML 1.0**. Tato specifikace je stejná jako specifikace HTML 4.01, jsou zde pouze integrována pravidla XML.

Organizace W3C, která je tvořena zástupci významných počítačových firem (seznam členů najdeme na stránkách www.w3c.org), chce XHTML prosazovat jako jediný jazyk pro definici webových stránek, bez ohledu na typ zařízení, které je zpracovává. Expanze webu na různá alternativní zařízení ukázala, že tato zařízení kvůli svým omezeným možnostem nemohou podporovat všechny vlastnosti XHTML, že podporují jen nějakou jeho část (podmnožinu). Tyto podmnožiny bylo třeba definovat a standardizovat, proto vznikla specifikace **Modularizace XHTML**. Tato specifikace rozděluje všechny prvky XHTML 1.0 do modulů, ze kterých se následně skládají značkovací jazyky. Skupiny zabývající se webem v alternativních zařízeních mohou definovat nové moduly s prvky specifickými pro dané zařízení, mohou stávající moduly modifikovat, ale hlavně mohou z modulů skládat nové kompletní značkovací jazyky, které vyhovují potřebám a možnostem interpretace webu na těchto zařízeních. Z těchto jazyků zatím W3C uznalo **XHTML 1.1** (až na několik detailů se shoduje s XHTML 1.0 Strict, hlavní rozdíl spočívá v tom, že XHTML 1.1 je definováno pomocí modulů) a **XHTML Basic** (skládá se pouze ze základních modulů, k použití hlavně na mobilních telefonech).

V další části této práce se proto budeme zabývat XHTML. HTML je již uzavřeným standardem, navíc XHTML je s HTML zpětně kompatibilní. Všechny vlastnosti XHTML budeme vysvětlovat na XHTML 1.0 Strict, popř. na XHTML 1.0 Transitional.

2.8 Syntaxe XHTML

2.8.1 Pojmy element, tag, atribut

Element je strukturovací jednotka. Druhy elementů a jejich vlastnosti jsou deklarovány v DTD.

Element je tvořen otvírací a zavírací částí. Otvírací část začíná < pak následuje jméno elementu a končí >. Zavírací část obsahuje < / jméno elementu a >.
<body>Text</body>

Zavírací nebo otvírací část elementu se jinak označuje jako tag. Text případně další obsah se zapisuje mezi tagy, tento text nebo data je pak formátován daným způsobem.
tučný text normální text

Všechny elementy v XHTML musí obsahovat koncový tag. U elementů, které musí mít nulový obsah (*img, hr, br*), se nemusí psát zavírací tag. Namísto zavíracího tagu se před > otvíracího tagu napíše /.
*<hr />,
* nebo s uzavíracím tagem *<hr></hr>,
</br>*

Používá se především 1. způsob. Aby tento zápis přečetli i starší prohlížeče před lomítkem napíšeme mezeru. Pokud se používá způsob s zavíracím tagem, musí mít nuloví obsah.

Do otevírací části elementu se zapisují atributy. Atributy se používají k upřesnění elementu nebo k přiřazení vlastností. Atribut je za názvem elementu a je vždy oddělen mezerou, ať už od názvu elementu nebo dalšího atributu. Hodnotu atributu přiřadíme pomocí = a hodnota musí být v uvozovkách.
**

Element *img* vkládá obrazový soubor. Atribut *src* obsahuje cestu a název souboru. Tento atribut je u elementu *img* povinný. Dalším povinným atributem je alt (alternativa). Atribut by měl obsahovat text charakterizující obrázek pro případ, že prohlížeč obrázky nepodporuje nebo se ho nepodařilo nahrát. Atributy *height* a *width* jsou nepovinné a udávají rozměry obrázku.

2.8.2 Správný zápis

Elementy a atributy se v XHTML musí zapisovat malými písmeny. Plyne to z DTD, kde jsou elementy zapsány malými písmeny.
**

Tento zápis je dobře, ale následující už ne, protože element IMG ani atribut src není zapsán v DTD.
<IMG Src="neco.jpg" alt="" />

Elementy musí být také správně vnořené. Jednotlivé elementy je nutné uzavírat v opačném pořadí než jsou otvírány. Element otevřený jako první je zavírán jako poslední.
* Text tučný, tučný s kurzívou*

Hořejší příklad je napsán správně, naopak v dolním je uzavírací tag kurzívy, až po uzavření tučného textu.

```
<b>Text tučný, <i>tučný s kurzívou</b></i>
```

V XHTML se vyžaduje dodržení následných pravidel vnoření:

- Značka <a> nesmí obsahovat další značku <a>.
- Značka <pre> nesmí obsahovat značky <img>, <object>, <big>, <small>,<sub> a <sup>.
- Do značky <button> nemohou být vloženy značky <input>, <select>, <textarea>, <label>, <button>, <form>, <fieldset>, <iframe>, <isindex>.
- Ve značce <label> nesmí být další značka <label>.
- Značka <form> nesmí obsahovat jinou značku <form>.

Tyto pravidla platí pro každou úroveň vnoření.

2.8.3 Základní kód XHTML souboru

Dokument XHTML musí obsahovat:

- XML deklarace
- Deklarace typu dokumentu
- Hlavička dokumentu
- Tělo dokumentu

XML deklarace- zde se uvádí verze XML a typ znakové sady. Zde je problém s dokumenty psanými ve Windows. Pro češtinu se používá ve Windows znaková sada windows-1250. Dokumenty uložené ve Windows jsou kódovány v sadě windows-1250. Tato sada ale není vždy podporovaná. Doporučuje se používat ISO 8859-2 nebo UTF-8. Proto se musí použít editor, který umí ukládat i v jiných znakových sadách (např. PSPad, Dreamweaver).

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

Za *encoding* můžeme dosadit windows-1250, UTF-16, ISO 8859-2. Záleží v jaké znakové sadě je dokument uložen.

V deklaraci dokumentu se určuje jaké DTD se má použít. Pro XHTML 1.0 existují 3 DTD:

XHTML 1.0 Strict

Striktní XHTML bylo použito jako základ pro novější XHTML 1.1. Ve striktním XHTML je zakázána většina elementů a atributů pro formátování vzhledu. Formát vzhledu se provádí přes CSS, a protože byl atribut style zakázán používají se třídy. Pokud se má pro dokument použít striktní XHTML je nad vlastním html tato deklarace dokumentu:

```
<!DOCTYPE html
    PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

XHTML 1.0 Transitional

Přechodná verze XHTML. Elementy a atributy jsou použity z HTML 4.1. Tato verze je nejblíže HTML.

```
<!DOCTYPE html
   PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

XHTML 1.0 Frameset

Verze XHTML kde jsou povoleny rámy. Jinak odpovídá XHTML Transitional.

```
<!DOCTYPE html
   PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
```

Po definici DTD následuje element html. Tento element je povinný a musí obsahovat atribut xmlns a to vždy s hodnotou http://www.w3.org/1999/xhtml. Atribut deklaruje jmenný prostor XHTML. Dále může obsahovat atributy xml:lang a lang. Hodnota atributů definuje jazyk, který je používán v dokumentu. Pro češtinu cs, angličtinu en.

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="cs" lang="cs" >
```
Počáteční tag elementu html s nastavením českého jazyka.

Do elementu html se pak zapisuje zbytek celého dokumentu.

K vyznačení hlavy dokumentu se používá element *head*. Informace obsažené v hlavičce se většinou vztahují k jazyku dokumentu, také obsahuje informace pro prohlížeč (více kap. 2.8.4). Musí obsahovat element *title* – titulek stránky.

Vlastní tělo je vyznačeno elementem body. Do těla se píše text nebo data které se mají zobrazit v prohlížeči více v kapitole 2.8.5.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html
   PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
        <head >
                <title>XHTML stránka</title>
        </head>
        <body>

                Viditelný text
        </body>

</html>
```

Toto je nejmenší kód validní stránky pro přechodné XHTML.

2.8.4 Záhlaví dokumentu

Záhlaví dokumentu se vymezuje elementem *head*. Tento element se zapisuje před element body (případně *frame*). Oblast *head* může obsahovat informace o dokumentu (jazyk, shrnutí obsahu, klíčová slova pro vyhledávače a jiné). V hlavičce musí být element *title*, nepovinné elementy: *script*, *style*, *meta*, *link*, *object*, *isindex* (není u striktního XHTML) , *base*.

Atributy povolené pro *head* jsou *lang, xml:lang, dir, id, profile*.

Atributy *lang* a *xml*:*lang* slouží k nastavení jazyka, je možné je psát také do elementu *html* (viz kap. 2.8.3). *Dir* označuje směr toku textu v dokumentu- z leva doprava (*ltr*), nebo zprava do leva (*rtl*). Automaticky je nastaveno na *ltr*. V atributu profile se udává adresa na jeden či více meta datových profilů.

2.8.4.1 Title

Element *title* je povinný. Udává titulek stránky, který se zobrazí v záhlaví okna. Titulek by měl označovat obsah stránky.

```
<head>
<title>titulek této stránky</title>
</head>
```

2.8.4.2 Meta informace

Element meta udává informace o použitém jazyce, nastavení kódování jazyka atd. Element *meta* nevyžaduje ukončovací tag a v hlavičce muže být několikrát. Musí mít atribut *content*.

Jedna z nejdůležitějších vlastností *meta* elementu je nastavení jazyka. Přes meta element se uvádí jazyk ve kterém je dokument napsán (pomáhá vyhledávačům) a pak nastavuje kódování jazyka (jako u deklarace XML).

```
<head>
<title>Titulek stránky</title>
<meta http-equiv="Content-Language" content="cs" />
<meta http-equiv="Content-Type" content="text/html; charset=windows-1250" />
</head>
```

Příklad hlavičky pro dokument psaný česky, který je uložen pomocí kódování windows-1250.

Atribut content udává hodnoty pro určitou vlastnost určenou atributem http-equiv nebo name.

Meta element se také používá na uvedení autora, nebo editoru stránek, ve kterém byl dokument napsán.

```
<meta name="author" content="Martin Vondrášek" />
<meta name="generator" content="PSPad editor, www.pspad.com" />
```

Další možnosti *meta* elementu jsou informace pro vyhledávače. V elementu se uvádí popis stránky, klíčová slova a lze vyhledávacím robotům zakázat indexovat stránky (nejde u všech).

```
<meta name="description" content="Seminární práce o XHTML" />
<meta name="keywords" content="XHTML,syntaxe" lang="cs" />
<meta name="robots" content="noindex,follow" />
```

Nastavení češtiny u klíčových slov pomáhá vyhledávacím robotům správně zařadit slova. Hodnota atributu *content noindex,follow* říká robotům, aby stránku neindexovaly, ale sledovaly odkazy na další stránky (např. při použití rámů je zbytečné indexovat část s menu).

2.8.4.3 Link

Tento element určuje vztah k ostatním dokumentům. Nemá ukončovací tag. Podobně jako meta element se může v hlavičce dokumentu vyskytovat několikrát.

Nejdůležitější atributy jsou *rel, href, type, media, hreflang, charset*. Element *link* se nejčastěji používá pro přiřazení externího CSS stylu (v XHML 1.1 je to velmi důležité) nebo na externí skript.

```
<link rel="stylesheet" href="adresa/styl.css“ type="text/css" />
```

Příklad přiložení externího stylu.

Pomocí elementu *link* se může vyhledávačům poskytnout verze v jiném jazyce nebo verze pro tisk.

```
<link title="The manual in Dutch" type="text/html" rel="alternate" hreflang="nl"
href=“http://someplace.com/manual/dutch.html“ />
```

Atribut *hreflang* určuje jazyk dokumentu na který se odkazuje. Je také možno použít ještě atribut *charset*. Ten pak určuje kódování dokumentu, na který se odkazuje.

```
<link media="print" title="The manual in postscript" type="application/postscript"
rel="alternate" href=“http://someplace.com/manual/postscript.ps“ />
```

Atribut *media* s hodnotou *print* udává, že odkaz je na verzi pro tisk.

2.8.4.4 Style, script, isindex, object

Element *style* umožňuje definovat vlastní CSS styl pro dokument. Musí obsahovat atribut *type*. CSS se zapisuje mezi otvírací a ukončovací tag. Zápis se provádí podle pravidel pro CSS

```
<style type=“txt/css“>
        body {  background: white; color: #000000}
</style>
```

Tento CSS styl nastaví bílé pozadí a černou barvu písma.

Script slouží k definici skriptů, které mohou jednotlivé elementy dokumentu vyvolávat. Opět je povinný atribut type, ve kterém se uvádí druh jazyka (např. text/javascript).

Element *isindex* není v XHTML 1.0 strict povolen. Používá se k vytváření indexovacích formulářů pro indexování dokument.

V XHTML jsou určeny druhy zpracování obrázku apod. Element *object* slouží k určení zpracování dalších typů souborů (např. flash aplikace).

2.8.5 Tělo

Data z těla dokumentu se zobrazují v prohlížeči. Element body má ukončovací tag. Atributy ovlivňující základní vzhled dokumentu jsou: *link, alink, vlink, text, bgcolor, background*. Tyto formátovací atributy jsou u striktního XHTML a ve XHTML 1.1 zakázány. Pro formát vzhledu dokumentu se používá CSS.

Atributy *link*, *alink*, *vlink* upravují barvu hypertextových odkazů. *Alink* barvu aktivního odkazu, *vlink* barvu navštíveného a *link* barvu odkazu který ještě nebyl navštíven.

Atribut *bgcolor* udává barvu pozadí dokumenu. Text nastavuje barvu textu v celém dokumentu (kromě odkazů). V atributu background se udává adresa k obrázku, který bude použit na pozadí stránky. Vlastnosti obrázku na pozadí se v XHTML může upravit pouze pomocí CSS (v něm se může nastavit i adresa atd.).

Barvy se zadávají v podobě hexadecimálního čísla. Toto číslo udává jasnost červené zelené a modré barvy (model RGB). Před číslem se píše #.
V XHTML je také možno zapsat 16 barev jejich názvem. Barvi jsou uvedeny v tabulce.

Black (#000000)	černá	Green (#008000)	zelená
Silver (#C0C0C0)	stříbrná	Lime (#00FF00)	světle zelená
Gray (#808080)	šedá	Olive (#808000)	Olivově zelená
White (#FFFFFF)	bílá	Yellow (#FFFF00)	žlutá
Maroon (#800000)	kaštanová	Navy (#000080)	tmavomodrá
Red (#FF0000)	červená	Blue (#0000FF)	modrá
Purple (#800080)	purpurová	Teal (#008080)	zeleno modrá
Fuchsia (#FF00FF)	fuchsiová	Aqua (#00FFFF)	světle zeleno modrá

Další atributy jsou *onload* a *onunload*. Tyto atributy obsahují skript, který se provede při dokončení nahrávání (onload) nebo pří ukončení stránky (onunload).

2.8.5.1 Atributy id, style, class, title, dir, lang

Tyto atributy se dají použít u většiny elementů (*body*, *table*, *p*, *a* atd.).

Atribut *id* slouží jako identifikátor. V dokumentu se nesmí vyskytovat více elementů se stejným *id*. Slouží jako kotva pro odkazy, selektor CSS stylu, jméno deklarovaného elementu *object* .

Style se používá k zápisu CSS. CSS vlastnost bude platná pouze pro daný element.

Atribut *class* slouží k přiřazení třídy (opět se používá pro přiřazení k selektoru CSS) k jedné třídě se může přiřadit více elementů.

Title se používá pro upřesnění. Atribut *title* se v prohlížeči Internet Explorer zobrazuje po najetí myši na odkaz (tool typ).

Atributy *dir* a *lang* byly vysvětleny v kap 2.8.4. Slouží k určení jazyka a směru textu.

```
<body bgcolor="#000000" text="#008000" link="#0000FF">
text stránky
</body>
```

Stránka s černým pozadím a stříbrnou barvou písma a modrými odkazy.

2.8.5.2 Používání rámů

Při užívání rámů se používá XHTML *frameset*. Namísto elementu body je definován element *frameset* a elementy *frame noframe*. Rámy dovolují nahrát více dokumentů do jednoho dokumentu. Tyto dokumenty se zobrazují jako by se otvírali samostatně (pozadí CSS styly atd.) a jejich velikost na stránce je nastavitelná.

Element *frameset* se používá místo elementu body. Atributy jsou *id, style, class, title, dir, lang, onload, onunload* jsou povoleny. K nastavení rámů se používají atributy *rows* (řádky) a *cols* (sloupce). V elementu *frameset* se nastavuje pomocí těchto atributů rozdělení dokumentu na řádky a (nebo) na sloupce, do nichž jsou pak nahrávány pomocí elementu *frame* další dokumenty. Hodnoty mohou být v procentech (25%) nebo v pixelech. Pro 3 rámy je potřeba 3 hodnot. Pro ulehčení je možné použít *, která přidělí rámu zbytek šířky okna.

Je možné vytvořit i vnořené kolekce rámů.

Pro element *frame* jsou platné opět atributy *id, style, class, title, dir, lang, onload, onunload*. Dalšími atributy jsou *longdesc, name, src, frameborder, marginwidth, marginheight, noresize, scrolling* a nevyžaduje uzavírací tag.

Longdesc udává adresu k popisku rámu. Atribut *name* slouží k pojmenování jednotlivých rámů hlavně za účelem správného odkazování. V *src* se uvádí adresa k dokumentu který se má do rámu nahrát. *Frameborder* vypíná/zapíná vykreslení hranice mezi jednotlivými rámy (automaticky nastaven na 1 zapnuto). Atributy *marginwidth* a *marginheight* nastavují šírku a výšku okrajů mezi rámy v pixelech. *Noresize* nastavuje možnost posunu hranice mezi jednotlivými rámy (1 a 0- zapnuto vypnuto). Poslední atribut *scrolling* nastavuje možnost posunu stránky. Je nastaven na auto posuvníky se zobrazí pouze je-li dokument větší než velkost rámu. Další nastavení no – posuvník se nezobrazuje nikdy. A yes posuvník je zobrazen stále.

Rámy se přidělují po řádcích z leva doprava a poté od zhora dolů.

Element *noframe* se používá pro zobrazení obsahu v případě že prohlížeč nepodporuje rámy. Většinou zde bývá odkaz na versi bez rámů.

```
</head>
<frameset cols="50%, 25%, *">
        <frame name="ram1" src="dokument1.html" frame border="0" />
        <frame name="ram2" src="dokument2.html" />
        <frame name="ram3" src="dokument3.html" scrolling="no" />
        <noframe>
        Bohužel váš prohlížeč nepodporuje rámy. Verze bez rámů
        <a href="noframe.htm">zde</a>.
        </noframe>
</frameset>
</html>
```

Na takovéto stránce budou 3 rámy vedle sebe. První rám bude zabírat půlku okna, 2. čtvrtinu a poslední rám také čtvrtinu (zbytek šířky). Třetí rám nebude možno scrollovat.

2.8.6 Práce s textem

Základním formátovacím elementem pro text je odstavec *p*. Atributy jsou *id, style, class, title, dir, lang.* Atribut *align* určuje zarovnání textu (hodnoty *left*- vlevo,

right-vpravo, *center*-střed, *justify*- blok). Automaticky je nastaveno zarovnání doleva. Důležité je nezapomínat na uzavírací tag a ukončovat elementy ve správném pořadí (viz kap. 2.8.2). Odsazuje se před počátečním tagem.

Element *div* slouží k rozdělení obsahu na několik části. Používá se k uspořádání dokumentu a pomocí atributů *id* a *class* k určení vlastností těchto částí. Povolené atributy *id, style, class, title, dir, lang, dále align*. Musí mít ukončovací tag.

Na zalomení řádky se používá element *br*. Ten nemusí mít uzavírací tag. Atributy *id, style, class, title, dir, lang* jsou povoleny. Dalším atributem je *clear*. Tento atribut slouží k zarovnání další řádky v případě, že text obtéká tabulku nebo obrázek. Hodnoty atributu jsou *link, all, right, none*. Ve striktním XHTML není tento atribut povolen.

```
<div id="obsah">
<img src="neco.jpg" align="left" alt="" />
Tex u obrázku, který je zobrazen napravo od něj.
<br clear="left" />Další řádek bude až pod obrázkem přes celou stránku. Nad ním bude napravo nahoře volné místo.
<p align="right" class="popis">
Nový odstavec textu.
</p>
</div>
```

Atribut *clear* upravuje zobrazení další řádky. Odstavec je zarovnán napravo a je přiřazen k třídě popis, která upravuje vzhled odstavce přes CSS. Celý úsek dokumentu je uzavřen v jednom oddílu, pro usnadnění přehlednosti a možnosti zobrazení (opět CSS).

Další možnosti pro formátování textu jsou nadpisy. Celkem je určeno 6 velikostí, přičemž největší je číslo 21 a nejmenší je číslo 6. Element pro nadpis vyžaduje ukončení. Element je zapsán pomocí značek h1, *h2, h3, h4, h5, h6*. Element nadpisů slouží k lepší struktuře a čitelnosti dokumentu. Pro stejné úrovně nadpisů v dokumentu by se měl použít stejný element. Atributy *id, style, class, title, dir, lang a align* jsou povoleny.

```
<h1>1. Co je XHTML </h1>
Text.
<h1>2. Syntaxe</h1>
<h2>2.1 Pojmy element, tag, atribut</h2>
Text.
<h2>2.2 Správný zápis </h2>
```

Takto by mělo vypadat využití nadpisů pro tento dokument.

Elementy pro změnu druhu písma jsou *font, b, i, u, tt, sub, sup, blockquote, cite, s, strike, big, small, pre*- všechny elementy mají koncový tag a jsou u nich povoleny atributy *id, style, class, title, dir, lang.*

Font slouží k nastavení druhu písma, velikosti, barvy. Atribut *size* nastaví velikost písma v bodech. U nastavení fontu (atribut *face*) je nutné aby tento font měl nainstalovaný i návštěvník stránek. Atribut *color* nastavuje barvu písma v hexadecimálním tvaru (viz kap. 2.8.5). V striktním XHTML (tedy i v novějších verzích) se na místo font používá CSS.

Popis elementů:

i – kurzíva, *b* – tučné písmo, *u* – podtržení písma, *tt* – text se zobrazí neproporciálním písmem. *Sub* – text psaný dolním indexem, *sup* – horní index.

Elementy *blockquote* a cite slouží k rozlišení citovaného textu. *Blockquote* odsadí text do vlastního odstavce. Má atribut cite do kterého se může uvést zdroj citátu. Element cite je pro citaci uvnitř textu.

Elementy s a strike přeškrtnou text.
Small a *big* zmenší nebo zvětší text.

Pre slouží k zobrazení textu tak jak byl zapsán- včetně odstavců a počtu mezer. Použití je vhodné například pro zobrazení zdrojových kódů a skriptů.

Span je obdobou elementu *div*. Slouží k určení *class* nebo *id* určitému textu v řádce.

Jako další atributy jsou povoleny *style, title, dir, lang* .

```
<font size="2" color="#006600"><b>Tučný text <u>s podtržením</u></b>
<br />t<sub>t</sub>= m<sup>2</sup>
<blockquote cite=http://citaty.cz/bible.html>
the number of the beast for it is a human number, its number is <span
class="cislo">Six hundred and sixty six</span>.
</blockquote>
Text pokračuje</font>
```

Celý tento příklad bude zobrazen zelenou barvou. Druhá řádka bude vypadat takto $t_t=m^2$. Blokový citát má určenou původní stránku, ze které pochází.

2.8.7 Seznamy

Seznamy slouží k lepšímu uspořádání dat. Seznamy se dělí na 2 základní druhy neuspořádané a uspořádané. Tyto seznamy je možno do sebe vnořovat. Oba dva druhy se dají dobře naformátovat skrze CSS.

2.8.7.1 Neuspořádané seznamy

Element tvořící neuspořádaný seznam je *ul* (unordered). Musí mít koncový tag. Atributy *style, id, class, title, dir, lang* jsou povoleny. Důležitým atributem je type, který určuje druh odrážky. U neuspořádaného seznamu jsou na výběr: *disc*(disk), *square*(čtverec), *circle*(kruh). Automaticky je nastaveno na *disc*. Při použití CSS je možno nastavit jako odrážku obrázek.

Elementem *ul* se začíná a uzavírá oblast seznamu, jednotlivé položky jsou pak vyčleněny elementem li. Koncový tag vyžaduje a atributy *style, id, class, title, dir, lang* a *type* jsou povoleny. Jednotlivé položky nemusí mít stejnou odrážku jako celý seznam – opět pomocí type. Atribut *value* se funkčně uplatní až u uspořádaných seznamů.

```
<ul type=“square“>
        <li type=“disc“>Jiný druh odrážky</li>
        <li>čtverec jako odrážka</li>
</ul>
```

2.8.7.2 Uspořádané seznamy

V uspořádaném seznamu jsou jednotlivé položky očíslovány. Element *ol* vyžaduje zavírací tag. Kromě atributů, které má i ul, je povolen atribut *start*. Tímto atributem se nastavuje začáteční číslo pro číslování (nechceme-li začínat 1). Pro atribut type je u uspořádaného seznamu více možností:

1 – nastaví číslování na arabské číslovky (1, 2, 3 …)

a - čísluje se malými písmeny abecedy (a,b,c…)

A – tentokrát se použijí velká písmena (A, B, C …)

i – čísluje se malými římskými číslovkami (i, ii, iii …)

I – číslování velkými římskými číslovkami (I, II, III …)

Jednotlivé položky jsou v elementu *li* (platí to samé jako u *li* v neuspořádaných seznamech 2.8.7.1). Atribut *value* je zde pro možnost přeskočit dopředu nebo se vrátit v číslování. Hodnota se udává od 0 do čísla, které chcete. Vždy se udává v číslicích.

```
<ol type=“I“ start=“50“>
        <li>římských padesát </li>
        <li type=“A“>Změna na abecedu</li>
        <li value=“0“>Seznam se začne číslovat znovu od nuly</li>
</ol>
```

Číslování začne od padesáti (tedy od L). U druhé položky místo LI bude AY. Třetí položka nastaví počítání od nuly.

2.8.7.3 Vnořování

Jednotlivé druhy seznamů se mohou vnořovat. Po uvedení položky se dá počátek elementu seznamu (*ul* nebo *ol*), ve kterém budou uvedeny další položky. Po ukončení tohoto seznamu se pokračuje dál v původním. Při vnoření neuspořádaných seznamů se automaticky střídají druhy odrážek (*disc, circle, square*).

```
<ol>
        <li>Nabídka 1</li>
                <ul>
                <li>možnost 1</li>
                </ul>
        <li>Nabídka 2</li>
</ol>
```

2.8.7.4 Definice

Element *dl* slouží jako seznam pro definice (slovíčka, pojmy atd.). Vyžaduje ukončovací tag Povolené atributy jsou *style, id, class, title, dir, lang*. Atribut type se neuplatňuje.

Pro pojem slouží element *dt*. Atributy shodné s *dl*. Ukončuje se.

Po elementu *dt* následuje element *dd*, kde je popis/vysvětlení pojmu. Atributy opět shodné s *dl* a vyžaduje ukončení.

```
<dl>
        <dt>XHTML</dt>
        <dd>Extensible HyperText Markup Language</dd>
</dl>
```

2.8.8 Odkazy

Odkazy jsou základem XHTML. Umožňují spojit dokumenty, například s podobnou tématikou, dalšími články atd.

Odkaz (hyperlink) je tvořen elementem *a*. Tento element se používá jak pro odkazování do jiného dokumentu i pro odkaz na určitou část již čteného dokumentu (kotvu). Povolené atributy jsou *samozřejmě style, id, class, title, dir, lang. Dále href, hreflang, target, rel, rev, name* (zaměnitelný s id, nedoporučuje se). Atribut *hreflang* udává jazyk odkazovaného dokumentu (viz kap. 2.8.3).

2.8.8.1 Target

Dalším důležitým atributem je *target*. Tento atribut udává okno do kterého se má stránka otevřít. V případě že se používají rámy se udává jméno rámu do kterého chcete odkaz otevřít. Základní hodnoty *target* jsou *_blank* – odkaz se otevře do nového okna, *_self* – do stejného okna (toto je přednastavené), *_parent* – načte soubor do rodičovského okna nebo množiny rámů.

2.8.8.2 href a cesty k souborům

Atribut *href* udává adresu souboru, na který se odkazuje. Adresa může být zadána relativně. Při relativním adresováním je cesta k souboru napsána vzhledem k pozici čteného dokumentu. Pro označení adresáře o jednu úroveň výš používáme ../. Na oddělení adresářů se používá normální lomítko /. Relativní adresování se používá pro soubory na jednom serveru.

```
<a href="../../books/sci-fi/asimov.html" target="_blank" >Asimov</a>
```

Tento odkaz veda na dokument umístění dvě úrovně nad současným adresářem.

```
<a href="http://server.com/books/sci-fi/asimov.html" target="_blank">Asimov</a>
```

Absolutní adresování se používá pro odkazy na jiné servery. Nezáleží na pozici odkazujícího dokumentu.

Před názvem souboru a cestou se uvádí duh používaného protokolu (ftp, http). Je možné napsat protokol file, takto se neurčuje protokol pro načtení- tato adresa odkazuje většinou na soubor uložený na počítači uživatele.

Přes odkaz se také předávají proměnné. Pro předání hodnot přes *GET* se za název souboru připojí za ? název proměnné a její hodnota. Více proměnných se spojuje v HTML znakem &, v XHTML se musí použít zástupný znak &. Předávání proměnných se používá pro jazyky php a asp, které zpracují požadavek uživatele a odešle připravenou XHTML stránku.

```
<a href="index.php?id=flash/mapa&amp;amp;men=flash">mapa</a>
```

Při zobrazení na stránce se místo *&* zobrazí *&*. V souboru *index.php* jsou proměnné *id* a *men* dále zpracovány.

2.8.8.3 Kotvy

Vytváření záložek v XHTML je velmi jednoduché. Můžeme odkazovat na odkaz s nulovou délkou s atributem *name*, nebo na id, které je v dokumentu použito. Druhý způsob se upřednostňuje. Hodnotu *id* nebo *name* napíšeme do *href* s # před hodnotami. V případě odkazování na kotvu v jiném dokumentu se napíše nejdříve cesta a název souboru pak # a nakonec hodnota *id* nebo *name*.

kapitola 2

…text…

Kapitola 2

nebo s použitím *name*

**

pro odkazování z jiného dokumentu

kapitola 2

Místo elementu *span* je možno použít jakýkoliv element, u kterého je povolen *id* (např. h, div …)

2.8.9 Obrázky

Obrázkový element *img* umožňuje vkládat do textu obrázky. Tento element nevyžaduje uzavírací tag. Atributy *style, id, class, title, dir, lang* jsou povoleny. Je povolen také atribut *name*, ale ten je opět zaměnitelný s id.

Povinné elementy jsou *alt* a *src*. Atribut alt slouží k popisu obrázku. V případě že se obrázek nenačte se použije tento text. Tento popisek využívají i prohlížeče pro nevidomé, a proto by měl charakterizovat obrázek.

Atribut *src* udává cestu k souboru (způsob zadávání viz kap. 2.8.2). Jako obrázky se nejčastěji používají soubory typu gif, jpg nebo i bmp. Gif a jpg jsou komprimované (gif bezstrátově), oproti tomu bmp nekomprimovaný.

Další atributy již nejsou povinné. Atributy *height* a *width*, které nastavují výšku a šířku obrázku. Se doporučuje uvádět z důvodů lepšího nahrávání. Přesněji, když se obrázek nenahraje zůstane prostor, kde má být obrázek, ve stejné velikosti- pokud byl tento obrázek nějak zakomponován do designu stránky tak design bude „držet tvar". Další výhodou je rychlejší rozestavění elementů na stránce. Pokud by nebyla zadána výška a šířka musí se čekat na načtení celého obrázku než se správně zarovná text na stránce. Velikost se udává v pixelech.

Podobný atributu alt je *longdesc*. Ten slouží k delšímu popisu obrázku.

Atribut *align* udává zarovnání obrázku. Tento atribut je v XHTML- strict zakázaný. Slouží k zarovnání obrázku vůči textu a jiným elementům. Hodnoty *left, right, top, middle, bottom.*

Atribut *border* nastavuje velikost okraje obrázku. Při použití obrázku v textu je tento okraj nulový, ale při použití v odkazu se nastaví na 2 pixely. Při použití atributu *border* je tento okraj nastaven na uvedenou hodnotu. Ve striktním XHTML je *border* zakázán.

Hspace vspace jsou atributy nastavující volný prostor kolem obrázku. *Hspace* definuje velikost na levé a pravé straně obrázku. *Vspace* volný prostor nad a pod obrázkem. Tyto atributy nejsou v nových verzích XHTML podporovány. Nastavení se provádí přes CSS.

```
<img src="picture/auto.gif" alt="rychle auto" height="110" widht="400"
align="left" />
```

2.8.10 Objekty

Element *object* slouží ke vkládání externích objektů do stránky (např. obrázky, video, Shockwave aplikace..).

2.8.10.1 Object

Element *object* vyžaduje koncový tag. Atributy *style, id, class, title, dir, lang a type* jsou povoleny. Pokud není vložený objekt podporován zobrazí se obsah elementu (kromě elementu param viz 2.10.2).

Atribut type slouží k nastavení MIME –typu dat, které jsou definovány v atributu data. Například pro video je nastavení video/mpeg, pro Flash aplikace application/x-shockwave-flash.

Data udává adresu datového souboru, který má být nahrán. Hodnotou může být i samotný obsah (např. kód v nějakém jazyce), ale není to doporučováno. Pokud je data uvedeno a prohlížeč nepodporuje daný druh objektu, není objekt stahován.

Classid definuje adresu objektu (nejčastěji třídy jazyku Java), jenž se načte a vloží do dokumentu.

Codetype se doporučuje použít při *classid*. Tento atribut je podobný atributu data, ale identifikuje typ programového kódu.

Codebase nastaví alternativní základní adresu. Pokud jsou například všechny applety v jednom adresáři a chceme si ulehčit psaní adresy, tak nastavíme *codebase*.

```
<object classid="hodiny.class" codebase="http://server/classes/"
codetype="application/java">
</object>
```

Adresa bude ve výsledku server/classes/hodiny.class.

Archive uvádí adresu (i více adres) které mají vztah k objektu. Tyto soubory se nahrají předem, a tím se zkrátí celkový čas na nahrání objektu.

Atributy *height* a *width* nastavují výšku šířku objektu při jeho zobrazení.

Objekty se mohou definovat už v hlavičce dokumentu a to s využitím atributu declare. Pro vyvolání objektu je nutné nastavit id. Toto id pak voláme podobně jako kotvu pomocí #.

```
<objekt declare="declare" id="objekt1" …..></object>
<a href="#objekt1">…</a>
```

2.8.10.2 Param

Element *param* se vkládá do elementu *object*. *Param* nemusí mít koncový tag. Tento element slouží k předávání parametrů objektu.

Atribut *id* označuje jméno dokumentu. Důležitý atribut je *name*. Ten udává název nastavovaného parametru a je povinný. *Value* nastavuje hodnotu parametru.

Atribut *valuetype* určuje typ hodnoty parametru. Nastavení *data, ref, object.*

Data - hodnota je po vyhodnocení předána objektu jako řetězec.

Ref – hodnota je adresa zdroje, kde se nachází parametry objektu.

Object – hodnota je odkaz na jíž definovaný objekt, value musí obsahovat # a id objektu. Automaticky je nastaveno data. Poslední povolený atribut je type. Při nastavení *valuetype* na *ref* definuje MIME-typ dat nacházející se na udané adrese.

```
<object type="application/x-shockwave-flash" data="flash/mapa2.swf" width="440"
height="150">
  <param name="movie" value="flash/mapa2.swf" />
  <param name="bgcolor" value="#ffffff" />
</object>
```

Příklad použití *param* pro Flash aplikaci- nastavení barvy pozadí.

2.8.11 Tabulky

Tabulky slouží k přehlednému uspořádání záznamů. Neměly by sloužit k *layoutu* stránek.

2.8.11.1 Element table

Základní element je *table*. Tento element má koncový tag. Označuje začátek a konec tabulky.

Atributy *style, id, class, title, dir, lang a type* jsou povoleny. Další atributem je *bgcolor* – nastavuje barvu pozadí tabulky. *Border* určuje šířku orámování tabulky a buněk v pixelech. *Width* nastaví šířku tabulky (atribut height u tabulky nepoužíváme).

Cellspacing nastavuje počet pixelů, které mají být volné od sousedních orámováních buněk a okrajů tabulky. *Cellpadding* nastavuje velikost volného prostoru mezi obsahem buňky a jejím orámováním.

Rules umožňuje zobrazení hranic. Jejich zobrazení záleží na prohlížeči. Možné hodnoty *none*(žádné ohraničení), *groups*(hranicese zobrazí podle *rozdělení thead, tbody, tfoot, col*group a *col*), *rows*(hranice se zobrazí po řádcích), *cols*(zobrazení hranic u sloupců), *all*(hranice mezi všemi částmi tabulky).

Frame určuje, která ze stran orámování obklopující tabulku bude vyditelná. Nastavení jsou *void*(žádná část), *above*(horní část orámování), *below*(spodní část), *hsides*(nahoře a dole), *lhs*(levá část), *rhs*(pravá strana), *vsides*(levá a pravá část), *box*(všechny 4 strany), *border*(opět všchny části).

Atribut *summary* slouží k zapsání stručného obsahu tabulky.

Atribut *align* nastavuje zarovnání tabulky na stránce (left, center, right).

2.8.11.2 Elementy pro dělení tabulky

Tyto elementy slouží k lepší přehlednosti a k formátování určených celků.

Element *caption* uvádí nadpis tabulky. Atribut *align* slouží k nastavení zobrazení tohoto popisku-top,bottom, left, right. Tento element má uzavírací tag.

Thead – tento element může být v tabulce pouze jednou a to na jejím začátku. Vymezuje hlavičku tabulky. Pokud je podporována prohlížečem jsou řádky z hlavičky při tisku vkládány na každý list (při dlouhé tabulce). Vyžaduje koncový tag. Atributy align (hodnoty left, center, right, justify, char) a valign (hodnoty top, middle, bottom, baseline) nastavují vodorovné a svislé zarovnání obsahu buněk v hlavičce.

Tbody slouží k rozdělení obsahu tabulky na části. Vyžaduje koncový tag. Atributy jsou shodné s thead.

Tfoot opět slouží k vyznačení zápatí tabulky. Smí se vyskytovat pouze jednou a to na konci tabulky. Atributy jsou shodné s *thead* a *tbody*.

Element *colgroup* slouží jako společná definice několika sloupců nebo jako kontejner pro několik různých sloupců. Koncový tag vyžadován. Atributy jsou shodné s *tbody* a navíc jsou přítomny *span* a *width*. Atribut *span* udává počet ovlivněných sloupců. *Width* celkovou šířku sloupců. Pokud chceme nastavit jednotlivé sloupce použijeme element *col*.

```
<colgroup>
        <col />
        <col />
</colgroup>
```

nebo pro nastavení stejných sloupců

```
<colgroup span="4">
</colgroup>
```

Posledním elementem pro rozdělení na celky je *col*. Tento element se může objevit pouze v elementu *colgroup*. A nastavuje vlastnosti sloupce. Atributy jsou shodné s *colgroup*.

2.8.11.3 Řádky a buňky

Jednotlivé řádky v tabulce se vymezují pomocí elementu tr. Tento element vyžaduje koncový tag. Atributy klasicky *style, id, class, title, dir, lang a type*, dále *align, valign a bgcolor*.

V tomto elementu jsou zapsány buňky. Zapisuje se po řádce z leva doprava.

Element pro buňku je *td*. Koncový tag je vyžadován. Atributy stejné jako u *tr*. Další atributy *width* a *height* nastavující výšku a šířku buňky. *Nowrap* – tento atribut zabrání zalomení obsahu buňky na více řádku. Hodnota se musí nastavit na „*nowrap*“.

Atribut *abbr* je zkrácený obsah buňky pro případ že se ji nelze zobrazit celou.

Důležité atributy jsou *colspan* a *rowspan*. Tyto slouží k nastavení sloučených buněk.

Colspan nastavuje počet spojených sloupců. Nastavením na 1 se nic nemění. 2 – buňka bude zabírat 2 sloupce.

Rowspan nastavuje počet buněk spojených vertikálně.

Při použití se pak vynechává v sloupci nebo řádce počet buněk, které budou spojeny.

Podobně jako *td* slouží element *th*. Má stejné atributy a používá se jako nadpis řádků v tabulce.

```
<table>
	<tr>
		<td rowspan="2" colspan="3">Buňka, která je ve třech sloupcích a na 2 řádkách</td>
		<td>4. buňka</td>
	</tr><tr>
		<td>4. buňka</td>
	</tr><tr>
		<td>1. buňka</td>
		<td>2. buňka</td>
		<td>3. buňka</td>
		<td>4. buňka</td>
	</td>
</table>
```

2.8.12 Formuláře

Formuláře slouží k sběru dat a jejich následnému odeslání pro další zpracování.

Celý formulář je v elementu *form*. Tento element musí mít ukončovací tag. Má povinný atribut *action*, v něm se udává adresa aplikace která má přijímat a zpracovat data formuláře.

Style, id, class, title, dir, lang a type jsou povoleny. Atribut *method* nastavuje způsob odeslání dat – *get* nebo post metodou automaticky je nastaven na get.

Atributy *onsubmit onreset* udávají skripty, které se mají provést při odeslání (*onsubmit*) nebo při stisknutí nulovacího tlačítka (*onreset*).

Atribut target slouží k nasměrování odeslání dat do určitého framu stránky.

2.8.12.1 Element input

Tento element je základním prvkem formulářů. Nevyžaduje ukončovací tag.

U elementu *input* je povinný atribut *name* (kromě 2 vyjímek a to při nastavení type na *submit* a *reset*). Hodnota *name* je název odesílané proměnné. Pomocí atributu type se nastaví požadovaný typ.

Text – vytvoří 1 řádkové textové pole. Pomocí atributu *size* nastavíme šířku v počtu znaků. Pomocí atributu *maxlength* nastavujeme maximální délku obsahu. Pomocí atributu *value* nastavíme výchozí hodnotu pole.

Password- podobné textovému poli, ale místo znaků budou zobrazovány pouze hvězdičky (zadávání hesel).

Checkbox- vytvoří zaškrtávací tlačítko. Pomocí atributu checked s nastavením checked bude před škrtnuto. Najednou lze zaškrtnout více checkboxů s stejným name. Atribut value zde nastavuje hodnotu proměnné.

Radio- vytvoří kulaté tlačítka. Najednou může být pro jednu hodnotu *name* označeno pouze jeden radio button. *Value* opět nastavuje hodnotu proměnné.

Submit – vytvoří tlačítko které odešle hodnoty z formuláře na adresu definovanou v *action* v elementu *form*. Popisek tlačítka lze upravit pomocí *value*. Při použití *name* se odešle jako proměnná *value* tlačítka.

Reset- vytvoří tlačítko podobně jako *submit*. Toto tlačítko nastaví hodnoty formuláře na původní hodnoty.

Hidden- tento element nebude zobrazen. Slouží hlavně ke kontrole, případně pro již zadaná data, které uživatel již nemusí vyplňovat.

Image- dovoluje vytvořit tlačítko z obrázku. Adresa obrázku se zadává přes src.

Další hodnoty jsou *button* a *file*.

2.8.12.2 Element textarea

Vytvoří oblast pro zadávání textu. Vyžaduje koncový tag. Velikost oblasti se upravuje atributy *cols*(šířka) a *rows*(počet řádků). Hodnota je v počtu znaků.

Atribut name nastavuje název proměnné.

Pokud chceme přednastavit hodnotu *texarea* požadovaný text zapíšeme mezi počáteční a koncový tag.

```
<form action="zpracuj.php">
    Jméno: <input type="text" name="jmeno" size="32" />
    <br />Příjmení: <input type="text" name="prijmeni" size="32" />
    <br />Typ školy:
    <br /><input type="radio" name="skola" value="zs" /> Základní škola
    <br /><input type="radio" name="skola" value="ss" /> Střední škola
    <br /><input type="radio" name="skola" value="vs" /> Vysoká škola
    <br />Poznámka:
    <br /><textarea name="pozn" rows="10" cols="50" >
    sem napište co chcete</textarea>
    <br /><input type="submit" value="odeslat" /> <input type="reset" />
</form>
```

Formulář při odeslání zavolá zpracuj.php, zde se provede vyhodnocení a další akce.

3 CSS

3.1 Úvod

CSS je zkratka pro anglický název **Cascading Style Sheets**, česky tabulky kaskádových stylů. Je to jazyk pro popis způsobu zobrazení stránek napsaných v jazycích HTML, XHTML nebo XML.

Jazyk byl navržen standardizační organizací W3C. Byly vydány zatím dvě verze specifikace CSS1(1996) a CSS2(1998), plus CSS 2.1, nyní se pracuje na verzi CSS3 (rok 2006).

Hlavním smyslem je umožnit návrhářům oddělit vzhled dokumentu od jeho struktury a obsahu. Původně to měl umožnit už jazyk HTML, ale v důsledku nedostatečných standardů a konkurenčního boje výrobců prohlížečů se vyvinul jinak. Starší verze HTML obsahují celou řadu elementů, které nepopisují obsah a strukturu dokumentu, ale i způsob jeho zobrazení. Z hlediska zpracování dokumentů a vyhledávání informací není takový vývoj žádoucí.

Používání kaskádových stylů ve srovnání se samotným HTML v praxi přináší výhody:

- širší formátovací možnosti
- snadná tvorba a údržba konzistentního stylu
- oddělení struktury a stylu
- dynamická práce se styly
- formátování XML dokumentů
- větší kompatibilita alternativních webových prohlížečů
- kratší doba načítání stránky

Výhodou CSS oproti starému formátování v HTML je, že kód a obsah webu je uložen v souboru .html a veškerý design a formátování se načítá z jednoho souboru .css, který je většinou společný pro celý web. To znamená, že pokud máte v plánu změnu designu webu, stačí změnit pouze jeden soubor .css a změna se aplikuje na celý web.

Mohou také existovat různé styly pro různá výstupní zařízení. Webdesigner má tak možnost, prostřednictvím CSS stylů, dokumentu určit, jak bude vypadat na papíře, při projekci či na PDA apod. Specifikace CSS nezapomínají dokonce ani na zrakově postižené - je možno napsat styly pro hlasový syntetizátor nebo hmatovou čtečku Braillova písma.

Hlavní nevýhodou CSS je zatím stále špatná podpora v majoritních prohlížečích. Různé prohlížeče interpretují stejný CSS kód jinak a je někdy velmi obtížné jej napsat tak, aby se na všech (resp. na několika vybraných) prohlížečích výsledek zobrazil stejně.

3.2 Způsoby použití

Definice CSS mohou být součástí HTML dokumentu nebo mohou být v samostatném souboru. K HTML dokumentu jsou potom připojeny během jeho načítání do prohlížeče. Možnosti definování stylů jsou:

- Přímo v HTML dokumentu, v části označené párovou značkou <style> v sekci <head> (in head):

```
<head>
  ...
  <style type="text/css">
  seznam pravidel
  </style>
  ...
</head>
```

- V konkrétní značce (tagu) v HTML dokumentu (in line):

```
<XXX ... style="seznam vlastností">
```

- V externím souboru; soubor se načítá pomocí <link> v sekci <head>:

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

- V externím souboru; soubor se načítá CSS direktivou @import v sekci <style> (pouze v CSS2):

```
<style type="text/css">
  @import url("cesta_k_souboru");
  ...
</style>
```

V externím souboru je přitom seznam pravidel, stejně jako v sekci <STYLE>. Syntaxe definic CSS je popsána v další části.

Načtené CSS definice (tabulky stylů, angl. stylesheets) jsou uloženy v poli document.styleSheets[]. Každá položka v poli odpovídá jedné sekci <style> (včetně souborů načtených direktivou @import), nebo jednomu souboru načtenému pomocí příkazu <link>. Pořadí položek v poli pak odpovídá pořadí načítání CSS. Obecně podporovaná je pouze vlastnost disabled, která (de)aktivuje daný stylesheet.

```
document.styleSheets[n].disabled = true | false
```

Příklady

Seznam CSS vlastností lze uvést přímo v HTML značce (tagu). To má smysl jen u příležitostného stylování. Příklad:

```
<p style="color:red; text-align:right">můj podpis</p>
```

V sekci <style> v záhlaví lze definovat současně styly pro celý dokument. Kvůli kompatibilitě se starými prohlížeči je zvykem styly skrýt do komentáře, obdobně jako se to dělá ve značce <script>. Příklad:

```
<style type="text/css>
<!--
  h1 { color:blue }
  p { color:#404040; text-align:justify }
  ...
-->
</style>
```

Nejlepší je uložit definice do samostatného souboru. Lépe se pak spravují a jedna tabulka stylů navíc může být společná pro celý web. Vzhled mnoha stránek se pak snadno změní zásahem do jediného souboru. Příklad:

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

nebo

<style type="text/css>
<!--
  @import url("/lib/moje.css");
-->
</style>
```

Druhá možnost je pak výhodná tehdy, chceme-li, aby námi definované styly načetl pouze prohlížeč, který již podporuje CSS2 - starší prohlížeče tento příkaz neznají a tabulku stylů nenačtou.

Načtené tabulky stylů můžeme i vypínat a zapínat pomocí skriptů. **Příklad:**

```
<link rel="stylesheet" type="text/css" href="modre.css">
<link rel="stylesheet" type="text/css" href="cervene.css">
...
<script type="text/javascript">
function ZmenCSS(x) {
  document.styleSheets[0].disabled = (x!='m');
  document.styleSheets[1].disabled = (x!='c');
}
</script>
...
<button onclick="ZmenCSS('m')">Stranka v modrem stylu</button>
<button onclick="ZmenCSS('c')
">Stranka v cervenem stylu</button>
...
```

3.3 Dědičnost a způsob načítání CSS

V případě větších webů je vhodné využít **dědičnosti** a **způsobu načítání CSS** a vytvořit si styly v několika úrovních - od obecných ke konkrétnějším. Příkladem může být web, obsahující prezentaci firmy a nabídku produktů, přičemž tyto dvě části budeme chtít udělat mírně odlišné. Například v obou budou titulky <h1> bezpatkovým písmem, tučné, o velikosti 24/28 pixelů. Firemní prezentace je ale bude mít červené, zatímco prezentace produktů modré. Můžeme si proto vytvořit nejprve jeden obecný CSS soubor, který bude definovat společné vlastnosti stylů a pro každý oddíl webu pak další samostatný CSS soubor, který tyto definice doplní o vlastnosti specifické pro tuto část. Využíváme zde toho, že **později načtené CSS má vyšší prioritu** a u stejných prvků

(HTML značka či třída) "přebije" shodné atributy novou hodnotou. Atributy, které předefinovány nejsou, se dědí z předchozí definice, naopak nové atributy, které v předchozí definici nejsou uvedeny, jsou do výsledného stylu přidány:

```
// nejprve obecná definice - v souboru "spolecne.css" bude:
h1 { font: 24px/28px 'helvetica ce', 'arial', sans-serif; color: black; font-weight: bold }
```

```
// v souboru "firma.css" bude uvedeno:
h1 { color: red }
```

```
// v souboru "produkty.css" pak:
h1 { color: blue }
```

```
// na stránkách firemní prezentace načteme nejprve obecné, poté firemní css
<style type="text/css">
<!--
@import url("spolecne.css");
@import url("firma.css");
-->
</style>
```

```
// na stránkách prezentace produktů načteme nejprve obecné, poté produktové css
<style type="text/css">
<!--
@import url("spolecne.css");
@import url("produkty.css");
-->
</style>
```

Firemní prezentace potom bude mít červené titulky, stránky produktů modré, všude jinde budou černé; až na barvu budou všechny atributy titulků shodné

3.4 Syntaxe definic CSS

Kaskádové styly (CSS) je možné definovat přímo v HTML značkách (inline styly) nebo v tabulkách stylů (stylesheets) - ty jsou umístěny v samostatných souborech nebo uvnitř značky <style> v **záhlaví stránky**.

Tabulku stylů tvoří seznam položek. Položkou může být buďto pravidlo ve tvaru **selektor {definice}** nebo **@-pravidlo**. Selektor popisuje, na který prvek/prvky se příslušný styl vztahuje. V případě inline stylů (přímo ve značkách HTML) se už selektor neuvádí, selektorem je značka sama: definice je pak jedním z atributů značky, ve tvaru **<značka...style="definice">.** Definice již popisuje konkrétní vlastnosti zvoleného selektoru (resp. html značky) a má tvar vlastnost:hodnota, případně seznam těchto vlastností oddělených středníkem: **vlastnost:hodnota; ... ; vlastnost:hodnota**.

3.4.1 Syntaxe tabulky stylů (stylesheet):

```
[ komentář | pravidlo | @-pravidlo ] *
  komentář : <!-- text -->
 @-pravidlo : @direktiva parametry - např. @import url(...)
  pravidlo : selektor { definice [; definice ] * }
  selektor : viz dále
  definice : vlastnost: hodnota
```

3.4.2 Syntaxe inline stylů:

```
<značka atribut* styl atribut*>

  značka : všechny značky kromě base, basefont, head, html,
        meta, param, script, style a title
    styl : style="definice [; definice]*"
 definice : vlastnost: hodnota
```

[A]* - A se opakuje libovolně krát, nemusí ani jednou

A | B - A nebo B

Příklad tabulky stylů:

```
 <style type="text/css">
  <!-- toto je ukázkový stylesheet -->
  p { color: #3366aa; text-align:center; font-size: 10pt}
  td, th { font-weight: bold }
  .info em { color: red }
  .hlavicka > strong { font-weight: normal }
  p:first-line { font-size: 150% }
</style>
```

Příklad inline stylů:

```
<p style="color:blue"> ...
<table cellpadding="0" style="background:#ffff00" cellspacing="0"> ...
<a href="xxx" style="text-decoration: none; color: red">
```

3.5 Základní selektory

Nejjednodušími selektory jsou *selektory typu* a *univerzální selektor*.

Selektor typu: `X {...}`

`X` je libovolná HTML značka. Tento selektor se páruje s každou instancí daného prvku na stránce. **Příklad:** pravidlo `h1 { color: red }` znamená, že uvedený styl se bude aplikovat na všechny prvky typu `<h1>` na stránce; všechny titulky úrovně 1 budou červené.

Univerzální selektor: `* {...}`

Značí se symbolem `*` a páruje se všemi prvky. To znamená, že styl uvedený se selektorem `*` se bude aplikovat na všechny prvky, u nichž je to možné.

Příklad: pravidlo `*{line-height: 140% }` znamená, že všechny texty na stránce budou mít výšku řádky (proklad) 140 % základní výšky - nezáleží na

tom, zda budou uvnitř značky `<p>`, `<div>`, uvnitř tabulky `<table>` či budou mít definovánu nějakou třídu nebo identifikátor.

3.5.1 Selektory s atributy

U každého selektoru je možné specifikovat i atributy, jež jsou uvedeny u dané HTML značky. Styl se pak aplikuje pouze na ty instance daného prvku, které mají uvedeny atributy tak, jak je určeno v selektoru.

Selektor s kontrolou existence atributu: `X[atr] {...}`

Tento selektor páruje pouze s těmi prvky `X` na stránce, které mají uveden atribut `atr`, bez ohledu na to, jakou má hodnotu. **Příklad:** pravidlo `h1[title] {...}` se bude aplikovat pouze na ty prvky `<h1>`, které mají definován atribut `title` - na prvek `<h1>titulek</h1>` se styl aplikovat nebude, ale na prvek `<h1 title="xxx">titulek</h1>` ano.

Selektor s atributem dané hodnoty: `X[atr=hodn] {...}`

Páruje pouze s prvky, jejichž atribut `atr` je přesně roven hodnotě `hodn`. **Příklad:** pravidlo `table[width="100"] {...}` se bude aplikovat pouze na ty prvky typu `<table>`, které mají uveden atribut právě `width="100"` - na jiné prvky se styl aplikovat nebude. Atribut musí být roven hodnotě přesně, takže ani tabulky s atributem `width="100%"` se s tímto selektorem nespárují.

Selektor s atributem obsahujícím hodnotu: `X[atr~=hodn] {...}`

Páruje pouze s prvky, hodnotou jejichž atributu `atr` je mezerami oddělený seznam "slov" a jedno z nich je přesně rovno `hodn`. **Příklad:** pravidlu `A[rel~=copyleft] {...}` vyhoví takový prvek `<A>`, jež má v atributu `rel` uvedeno mezi jinými hodnotu `copyleft` - například `<A rel="copyright copyleft copyeditor" ...>`.

Selektor pro jazykové varianty: `X[atr|=hodn] {...}`

Páruje pouze s prvky, hodnotou jejichž atributu `atr` je spojovníky - oddělený seznam "slov" začínající na `hodn`. Tato funkce byla vytvořena speciálně k určování jazykových variant v atributu `lang` (podle RFC 1766), které jsou uváděny právě v této podobě. Zatímco `*[lang=en]` páruje pouze s prvky, které mají uveden atribut `lang="en"`, pravidlu `*[lang|=en]` vyhoví všechny prvky s každou variantou anglického jazyka - např. `lang="en"`, `lang="en-US"` nebo `lang="en-cockney"`.

Pozn.: Atributy u selektorů je možné kombinovat - **příklad:** pravidlo `table[width=100][title] {...}` uspěje pouze u těch prvků typu tabulka, které mají uvedeno `width="100"` a mají definován atribut `title`.

3.5.2 Selektory tříd a identifikátory

Selektor třídy: `X.jméno_třídy {...}`

Pravidlu vyhoví ty instance prvků, jejichž třída (`class`) je právě `jméno_třídy`. Tento typ selektoru byl zaveden pouze pro zjednodušení, neboť přesně odpovídá zápisu `x[class=jméno_třídy] {...}`. V případě selektoru `*.třída` není nutné univerzální selektor `*` uvádět, jako selektor postačí `.třída`. **Příklad:** pravidlo `h1.nadpis {...}` páruje se všemi prvky `<h1>`, které mají uveden atribut `class="nadpis"`; pravidlu `.nadpis {...}` - což je totéž jako `*.nadpis {...}` - pak vyhoví všechny prvky na stránce, které mají `class="nadpis"`.

Selektor ID: `X#ident {...}`

Styl se použije pouze u prvku, jehož `id` je `ident`. Pokud není specifikována konkrétní HTML značka, není nutné uvádět `*#ident`, ale jen `#ident`. **Příklad:** pravidlu `h1#nazev {...}` vyhoví pouze ten titulek `<h1>`, jehož `ID="nazev"`. Pravidlu `#nazev {...}` pak vyhoví všechny prvky na stránce, které mají definováno `id="nazev"`.

Pozn.: Selektory id mají vyšší prioritu než selektory s atributem - např. `#nazev` je z hlediska kaskádování specifičtější než `[id=nazev]`.

3.5.3 Kombinování selektorů

Selektory je možné vytvářet vzájemným kombinováním jednodušších selektorů a vytvářet tak specifičtější popis v závislosti na vzájemných vztazích prvků a na struktuře dokumentu. Selektory se vzájemně kombinují pomocí spojek `,` `+>`, z nichž každá má jiný význam.

Seskupování selektorů: x, y {...}

Selektory, které mají stejné vlastnosti, lze definovat jednoduchým způsobem společně. Namísto jednoduchého selektoru lze uvést seznam několika selektorů oddělených čárkou; definovaný styl se poté aplikuje jednotlivě na všechny selektory uvedené v seznamu. **Příklad:** chceme-li nastavit pro titulky všech úrovní stejnou (např. červenou) barvu, můžeme je definovat jednotlivě `h1 {color:red} h2 {color:red}` atd. Pro zjednodušení je ale také můžeme seskupit do jediného selektoru s jednou společnou definici: `h1, h2, h3, h4 {color:red}`. Oba zápisy jsou ekvivalentní, jak je ale vidět, seskupení selektorů je snazší a přehlednější.

Pozn.: Při procházení kaskády stylů (viz dále) se tento zápis nepovažuje za jeden selektor, ale rozdělí se na samostatné selektory se shodnou definicí. Na ostatní kombinace selektorů pomocí spojek `+>` již ale pohlíží jako na jediný selektor.

Selektor následníka: `X Y {...}`

CSS umožňuje návrhářům, aby odlišili HTML prvky podle toho, na jakém místě, v jakém kontextu dokumentu se vyskytují. Oddělíme-li v definici dva selektory `X` a `Y` mezerou (resp. *whitespace*), znamená to, že pravidlo `X Y {...}` se bude aplikovat na všechny prvky typu `Y`, které jsou následníky prvku `X` (prvek `X` je předchůdcem prvku `Y`). Jednoduše řečeno, pravidlo se páruje s těmi prvky `Y`, které jsou obsaženy **kdekoli** uvnitř nějakého prvku `X`. **Příklad:** chceme-li odlišit způsob zvýraznění `<strong>` v titulku `<h1>` od zvýraznění v ostatním textu, můžeme použít kontextový selektor - např. `h1 strong { color:red }`. Všechny prvky `<strong>`, které jsou uvnitř nějakého prvku `<h1>` budou zobrazeny červeně - např. tento: `<h1>Toto je <strong>důležitý</STRONG> titulek</H1>`.

Selektor potomka: `X > Y {...}`

Obdobné jako v předchozím případě - pravidlo se ale aplikuje pouze tehdy, je-li `Y` potomkem *(child)* `X`, neboli `X` je rodič *(parent)* `Y`. Rozdíl mezi potomkem a následníkem je v tom, že `Y` je *potomek* `X` (a tedy `X` je *rodič* `Y`), právě když `Y` je obsaženo přímo uvnitř prvku `X`. Zatímco `Y` je *následník* `X` i tehdy, je-li `Y` ještě obsaženo v nějakém prvku `Z`, který je potomkem `X` (každý potomek je i následníkem, obráceně již to nemusí platit). Např. ve struktuře `<h1>Toto je <a href="xxx"><b>důležitý</b> odkaz</a> v titulku</h1>` je `a` *potomek* `h1` a `b` je *potomek* `a`. `b` již není *potomek* `h1`, ale je stále jeho *následník*. **Příklad:**

pravidlo `body > P {...}` se bude aplikovat na ty odstavce `<P>`, které jsou (přímým) potomkem prvku `<body>`;na prvky `<p>`, které jsou ještě obsaženy v dalším prvku se již vztahovat nebude. Např. v konstrukci `<body><p>...</p><div><p>...</p></div></body>` se toto pravidlo bude aplikovat pouze na první odstavec; druhý odstavec již není *potomkem* `<body>` (je uzavřen v `<div>`), je pouze jeho následníkem (pravidlo `body P {...}` by se ale již vztahovalo i na něj).
Pozn.: přístup k formátování prvního potomka v posloupnosti dokumentu umožňuje pseudo-třída `:first-child` - viz dále.

Selektor sousedního sourozence: `X + Y {...}`

Pravidlo se aplikuje na prvky `Y`, pokud `X` a `Y` mají stejného rodiče (jsou *sourozenci*) a `Y` následuje přímo po `X`. Tento selektor umožňuje specifikovat styl pro sousedící prvky na stejné úrovni ve stromu dokumentu. **Příklad:** pravidlo `h1 + h2 {margin: -10px}` se bude aplikovat na titulky `<h2>`, které následují přímo za titulkem `<h1>`. Tento selektor poskytuje možnost přesněji specifikovat vztahy mezi sousedícími prvky, které jsou jinak upravovány automaticky během formátování a vykreslování dokumentu (např. *margin*).

3.5.4 Pseudo-třídy a pseudo-prvky

Na rozdíl od všech ostatních selektorů, které vycházejí z informací uvedených ve zdrojovém dokumentu, přinášejí pseudo-třídy a pseudo-prvky možnost formátovat informace nad rámec zdrojového dokumentu. **Pseudo-třídy** pro formátování zpřístupňují informace, které nelze zjistit ze struktury dokumentu (s jednou výjimkou), dynamické změny a výsledky interakce dokumentu s uživatelem. **Pseudo-prvky** (pseudo-elementy) umožňují specifikovat některé prvky jemněji, než jsou uvedeny ve struktuře dokumentu (např. první znak či první řádek prvku); a dovolují dokonce přidávat text, který ve zdrojovém dokumentu vůbec není.

Pseudo-třída: `X:first-child`

Toto je ona jediná výjimka - selektor `:first-child` vychází ze struktury dokumentu. Páruje se s *prvním potomkem* daného prvku. **Příklad:** zatímco pravidlo `div > p {...}` se bude aplikovat na všechny `<P>`, které jsou potomky nějakého prvku `<div>`, pravidlo `div > P:first-child {...}` pak pouze na ty `<P>`, které jsou *prvním potomkem* nějakého `<div>`.

Pseudo-třídy odkazů: `a:link` a `a:visited`

Prohlížeče obvykle odlišují odkazy, které již uživatel navštívil a které dosud nikoli. Tyto pseudo-třídy umožňují jejich implicitní formátování. `a:link` vyhovují ty odkazy `<a>`, které uživatel dosud nenavštívil, `a:visited` pak ty, které již navštívil. S těmito psedo-třídami je možné použít i specifičtější selektory (s atributy, třídy atd.) - např. definujeme pravidlo `a.mujllink:visited {color:red}`; jemuž vyhoví např. prvek `<a href="http://www.mujserver.cz" class="mujlink">Můj server</a>`. Definovaný styl se bude aplikovat tehdy, pokud uživatel již s prohlížečem navštívil adresu `"http://www.mujserver.cz"` a odkaz bude zobrazen červeně. `a` není nutné uvádět, `:link {color:blue}` způsobí, že všechny dosud nenavštívené odkazy budou zobrazeny modře.

Dynamické pseudo-třídy: `X:hover`, `X:active` a `X:focus`

Interaktivní prohlížeče mohou odlišovat zobrazení prvků podle akcí uživatele. Klienty, které nepodporují interaktivní media, tyto pseudo-třídy nemusí podporovat. Pseudo-třída `:hover` se aplikuje tehdy, ukáže-li uživatel na daný

prvek, ale neaktivujej jej (např. nad něj najede kurzorem myši); `:active` pak tehdy, je-li prvek aktivován (např. v čase mezi stiskem a puštěním tlačítka myši); a `:focus` tehdy, je-li prvek zvýrazněn (např. formulářové prvky při zadávání textu, prvky akceptující události klávesnice atd.). Tyto pseudo-třídy se vzájemně nevylučují - prvky mohou současně vyhovovat i více z nich. **Příklad:** pravidlo `a: focus:hover {...}` bude splněno tehdy, bude-li mít prvek *focus* a současně na něj uživatel "ukáže". Je potřeba pamatovat na pořadí pravidel s ohledem na kaskádování. Např. máme-li definici `a:link {color:blue} a:visited {color:black} a:hover {color:red} a:active {color:orange}`, musí být `a:hover` uvedeno až za `a:link` a `a:visited`, protože pravidla kaskádování by jinak překryla vlastnost `color` v `a:hover`. Obdobně, protože `a:active` následuje až za `a:hover`, `color:orange` se aplikuje jen tehdy, je-li prvek současně aktivován a je na něj "ukázáno" (*hover*).

Jazyková pseudo-třída: `X:lang()`

Tyto selektory lze použít, pokud dokument specifikuje jazyk, který prvky na stránce používají. V případě HTML4 to může být kombinací informací z hlavičky protokolu (HTTP), z informací uvedených v `<meta>` značkách v záhlaví dokumentu a z atributů `lang="..."`. Pseudo-třídě `:lang(C)` vyhovují prvky v jazyce `C`, kde `C` je kód jazyka podle specifikace HTML4 a RFC 1766. **Příklad:** pravidla `HTML:lang[cs] {quotes: '„' '“'} HTML:lang[en] {quotes: '“' '”'} HTML:lang[fr] {quotes: '« ' ' »'}` specifikují, jak se zobrazí uvozovky v citacích podle toho, v jakém jazyce daný prvek bude - „citace“ v případě češtiny, “citace” v případě angličtiny a « citace » v případě francouzštiny.

Pseudo-prvky: `X:first-line` a `X:first-letter`

Pseudo-prvek `:first-line` aplikuje daný styl na první řádek textu daného bloku, podle toho, jak prohlížeč text zformátuje. Pokud se text zalomí jinak (při změně velikosti okna, při změně velikosti textu atd.) a změní se tedy i místo zalomení prvního řádku, bude text nově zformátován a `:first-line` se bude znovu aplikovat na nově vzniklý první řádek bloku. Obdobně pseudo-prvek `:first-letter` se aplikuje na první znak textu. Tomuto pseudo-prvku je možné i definovat i vlastnost `float`, čímž je možné vytvářet např. zapuštěné iniciály. Klienty by v případě, že text začíná interpunkcí, měly zahrnout do `:first-letter` i toto interpunční znaménko a vykreslit tak odlišně dva znaky - např. u textu `<P>"Text"</P>` se jako `P:first-letter` vykreslí `"T`, tedy písmeno T i s úvozovkou, která mu předchází. **Příklad:** pravidlo `P:first-line { text-transform: uppercase }` zobrazí první řádek každého odstavce verzálkami; pravidlo `p.perex:first-letter { float:left; font-size: 300%; font-weight: bold }` způsobí, že se první znak odstavce, který má `class="perex"`, se vykreslí jako zapuštěná iniciála (obtékaný box), trojnásobné velikosti a tučně.

Pseudo-prvky: `X:before` a `X:after`

Tyto pseudo-prvky se používají ke vložení generovaného obsahu *před* a *za* daný prvek. **Příklad:** pravidlo `p.poznamka:before { content: "Pozn.: " }` způsobí, že na začátek každého odstavce třídy `poznamka` se vloží text `"Pozn.: "`, obdobně `P.poznamka:after { content: " konec poznámky" }` vloží text na konec odstavce. `h1:before { content: counter(chapno, upper-roman) ". " }` bude daným způsobem číslovat titulky `<h1>`.

3.6 Dědičnost a kaskádování v CSS

Při načítání stránky klient postupně vytváří *strom dokumentu* podle použitého objektového modelu. Během tohoto procesu **každému prvku přiřadí hodnotu každé vlastnosti,** která je u daného prvku na daném médiu (obrazovka počítače, tiskárna, zvukový výstup atd.) použitelná. Každá z těchto hodnot prochází třemi fázemi výpočtu: nejprve se načte *definovaná hodnota* - podle pravidel dědění a kaskádování. Poté se tato hodnota, pokud je to nutné, *přepočítá* na absolutní jednotky - pixely pro obrazovku, metrické jednotky pro tiskárnu, barvy se konvertují do jednotného barevného modelu atd. A na závěr je případně *upravena* podle možností výstupního média (zaokrouhlení na celé pixely, přizpůsobení barev atd.)

3.6.1 Definované hodnoty

Klient tedy nejprve přiřazuje hodnotu každé myslitelné vlastnosti každého prvku v dokumentu. Při tomto procesu používá tato pravidla:

- Pokud proces **kaskády** (viz dále) vrátí nějakou hodnotu - tj. vlastnost je specifikována nějakou definicí stylu, uživatelem atd. - použije se tato hodnota.
- Není-li vlastnost explicitně určena kaskádou, hodnota se **dědí,** tj. použije se hodnota, která je definována pro rodičovský prvek (prvek nadřazený ve smyslu stromu dokumentu).
- Není-li možné tuto vlastnost zdědit (rodičovský prvek danou vlastnost nemá definovánu, vlastnost není dědičná atd.), použije se **výchozí hodnota.** Každý prvek má definovány výchozí hodnoty pro každou vlastnost.

3.6.2 Dědičnost v CSS

Některé vlastnosti se mohou v rámci stromu dokumentu dědit. To znamená, že pokud není řečeno jinak, vlastnost definovanou pro jeden prvek přebírají i všechny objekty uvnitř tohoto prvku. Nedědí se ale všechny vlastnosti - každá vlastnost má definováno, zda její hodnota je či není dědičná.

Příklad:

Vlastnost color (barva textu) je dědičná. Pokud v tomto příkladu:

`<h1>toto je <em> důležitý </em> titulek </h1>` nebude mít prvek em nijak specifikovánu vlastnost color, převezme ji od rodičovského prvku h1(z hlediska stromu dokumentu je zde em potomkem h1, neboť prvek em je obsažen uvnitř prvku h1) - celý text titulku bude tedy zobrazen stejnou barvou textu.

Pozn.: ve většině případů se nedědí definované hodnoty, ale až hodnoty přepočítané - tzn. má-li např. rodičovský prvek definovánu vlastnost v procentech, dědí se až hodnota přepočítaná na absolutní jednotky.

3.6.3 Kaskáda

Styly, ovlivňující jeden dokument, mohou pocházet ze tří různých zdrojů: od autora, od uživatele a z klientu (prohlížeče). Autor (tvůrce dokumentu) specifikuje styly přímo v dokumentu (či v externích souborech k němu připojených). Uživatel si v některých prohlížečích může zvolit své vlastní styly (např. připojit svůj vlastní soubor se styly nebo nastavit některé vlastnosti přímo v prohlížeči). A v neposlední řadě každý

klient (prohlížeč) má za povinnost mít definovánu výchozí tabulku stylů, která předchází všem dříve jmenovaným a definuje tak výchozí hodnoty podle možností daného klientu. Ve specifikaci CSS je ukázka typické výchozí tabulky stylů pro HTML4. Všechny tyto styly se navzájem ovlivňují a tvoří výsledné hodnoty podle pravidel kaskádování.

Kaskáda ohodnotí každý styl - přidělí mu tzv. váhu. Pokud je pro danou vlastnost více pravidel, přednost dostane to, které má větší váhu. Standardně mají styly autora větší váhu než styly uživatele; pouze při použití direktivy !important je tomu naopak. Styly autora i uživatele ale mají vždy větší váhu než výchozí styly klientu. Váha také záleží na pořadí a způsobu načtení: styly z dané tabulky stylů mají větší váhu a přepíší styly importované z jiné tabulky.

Aby se mohla určit hodnota dané vlastnosti daného prvku, klient musí všechna pravidla nejprve setřídit podle následujícího třídění kaskády:

1. Nejprve **najde všechna pravidla,** která se vztahují k danému prvku, dané vlastnosti a použitému médiu . Pravidla se použijí, pokud jejich selektor páruje s daným prvkem.
2. Poté **setřídí pravidla podle jejich váhy a zdroje** - styly autora mají přednost před styly uživatele, výchozí styly klientu mají nejmenší váhu atd.
3. Následně **setřídí pravidla podle jejich specifičnosti** - specifičtější pravidla mají přednost před obecnějšími. Se pseudo-třídami a pseudo-prvky se zde nakládá, jako by to byly běžné třídy a prvky.
4. V posledním kroku jsou pravidla **setříděna podle pořadí** v dokumentu. Z pravidel, která mají stejnou váhu, zdroj i specifičnost, se použije to, které je definováno nejpozději. Importované styly se považují za definované dříve než styly popsané přímo v dané tabulce.

3.6.4 Specifičnost (konkrétnost) selektorů

Selektory, popisující, na které prvky se mají dané vlastnosti vztahovat, mohou být velmi obecné i velmi konkrétní. Čím konkrétnější selektor je, tím větší má přednost před jinými, obecnějšími. Specifičnost je v CSS vyjádřena exaktním algoritmem:

1. a = počet atributů ID v selektoru
2. b = počet jiných atributů a pseudo-tříd v selektoru
3. c = počet jmen prvků v selektoru (pseudo-prvky se ignorují)

Výslednou specifičností je pak číslo vzniklé spojením těchto tří hodnot: a-b-c
Příklady:

```
*           {} /* a=0 b=0 c=0 -> specifičnost = 000 */
li          {} /* a=0 b=0 c=1 -> specifičnost = 001 */
ul li       {} /* a=0 b=0 c=2 -> specifičnost = 002 */
ul ol+li    {} /* a=0 b=0 c=3 -> specifičnost = 003 */
h1 + *[rel=up]{} /* a=0 b=1 c=1 -> specifičnost = 011 */
ul ol li.red {} /* a=0 b=1 c=3 -> specifičnost = 013 */
li.red.level {} /* a=0 b=2 c=1 -> specifičnost = 021 */
#x34y       {} /* a=1 b=0 c=0 -> specifičnost = 100 */
```

Pozn.: Styly, definované přímo v HTML značce pomocí atributu style="...", mají za selektor přímo HTML značku, v níž jsou uvedeny, což odpovídá selektoru #ID.

Jelikož jsou ale uvedeny v dokumentu nejpozději, mají podle 4. pravidla kaskády přednost před pravidly definovanými dříve se selektorem #id - např v této ukázce:

```
<style type="text/css">
#hlavni { color: blue }
</style>
...
<h1 id="hlavni" style="color:red">titulek</h1>
```

jsou obě pravidla stejně specifická (abc=100), ale styl uvnitř značky <h1> je uveden později, má tedy přednost podle 4. pravidla kaskádování - titulek proto bude červený.

Pozn. : Formátovací příkazy v HTML, které nejsou součástí CSS (např. značka <font>, atribut align atd.) se většinou překládají do odpovídajících pravidel CSS a v takovém případě se s nimi pracuje, jako by byly umístěny na začátku tabulky stylů autora se specifičností 0.

3.6.5 Pravidlo !important

Styly definované autorem mají větší váhu než styly uživatele . Použije-li se však v pravidle deklarace `!important`, dostává uživatelovo pravidlo přednost, čímž mají uživatelé možnost přizpůsobit si dokumenty svým potřebám a možnostem (větší písma, vhodné barvy atd.). Pokud je `!important` v pravidle uživatele i autora, uživatelův styl má stále přednost (to platí v CSS2, v CSS1 tomu bylo naopak).

Pozn.: Použije-li se direktiva `!important` v nějakém *shorthandu*, je to stejné, jako by se použila ve všech jednotlivých pravidlech, které shorthand zastupuje - např.:

```
{ border: 1px solid black !important }
```

je totéž jako

```
{ border-width: 1px !important; border-style: solid !important;
border-color: black !important }
```

Příklady

V praxi se setkáváme se načítáním externí tabulky stylů pomocí `<link rel="stylesheet"...>`, definováním stylů uvnitř značky `<style>` a uvnitř atributu `style="..."`. Ale jak tabulky stylů v dokumentu uvnitř `<style>`, tak i v externích souborech mohou obsahovat import jiné tabulky stylů pomocí direktivy `@import`. Je pak dobré si důkladně rozmyslet, v jakém pořadí jsou styly do stránky vlastně načteny a jakou tedy budou mít jednotlivá pravidla prioritu.

Příklad:

Budeme-li mít takto definované CSS v naší stránce:

```
...
<LINK REL="stylesheet" href="prvni.css" type="text/css">
...
<STYLE type="text/css">
  @import url("druha.css");
  ... zde nějaká pravidla ...
  @import url("treti.css");
  ...
</STYLE>
...
<STYLE type="text/css">
  @import url("ctvrta.css");
 </STYLE>
...
<BODY>
<H1 style="color:red">Titulek</H1>
... a tak dále...
```

A navíc bude:

```
v souboru "prvni.css" uvedeno:
  @import url("prvni_plus.css");
...
a v souboru "treti.css" uvedeno:
  @import url("treti_plus.css");
...
```

tak máme celkem slušnou sbírku tabulek stylů, které se v rámci kaskády budou překrývat a potřebujeme vědět, které pravidlo z které tabulky stylů bude vlastně použito.

Můžeme - pokud se soustředíme pouze na pořadí stylů - uvažovat stejně, jako bude postupovat klient při kaskádování našich stylů. Nejprve je v dokumentu uveden styl "prvni.css" (načten značkou <link>). Ten načítá (direktivou @import) styl "prvni_plus.css". Protože importované styly mají menší prioritu než styly uvedené přímo v tabulce, budou nejprve pravidla z tohoto importu, teprve za nimi všechna ostatní z tabulky "prvni.css".

Následuje tabulka stylů uvnitř první značky <style> v dokumentu. Do ní jsou importovány styly "druha.css" a "treti.css" - jak bylo řečeno, všechny importy se považují za načtené dříve než ostatní pravidla, nezáleží tedy na tom, že mezi oněmi dvěma importy jsou nějaká pravidla: ta budou tak jako tak považována za pozdější než ta importovaná a budou mít přednost před oběma importy (je proto vhodné všechny importy uvádět nejdřív, aby se to nepletlo).

Další tabulka obsahuje pouze import tabulky "ctvrta.css". Ovšem pozor, importy mají nižší prirotu pouze v rámci jedné tabulky stylů - zde se ale již jedná o novou tabulku (každá značka `<style>` a každý import značkou `<link>` představují jednu tabulku stylů). Pravidla zde uvedená proto zcela normálně následují po všech předešlých a mají tedy vyšší prioritu než předcházející styly.

Na závěr je definován styl přímo ve značce `<h1>` v těle dokumentu. Ten je nejpozdější, má tedy nejvyšší prirotu. Až tedy bude prohlížeč formátovat značku `<h1>` z naší ukázky, bude používat pravidla ze všech stylů, která setřídí takto (od nejnižší po nejvyšší prioritu):

1. pravidla ze souboru "prvni_plus.css"
2. ostatní pravidla ze souboru "prvni.css"
3. pravidla ze souboru "druha.css"
4. pravidla ze souboru "treti_plus.css"
5. ostatní pravidla ze souboru "treti.css"
6. ostatní pravidla z první tabulky `<style>`
7. pravidla ze souboru "ctvrta.css"
8. pravidla ze stylu přímo ve značce ("color:red")

Další příklady

Pravidel kaskádování je vhodné využívat pro přehlednost i pro usnadnění návrhu vzhledu stránek. Často je výhodné vytvořit si pro celý web jeden společný soubor s CSS, který se načte (je jedno jak) do každé stránky - a jednotlivé stránky pak mohou mít samostatné styly, které doplní nebo změní výchozí styly. Aby se řádně uplatnila pravidla kaskádování, musí být tyto "zpřesňující! styly načteny až po stylech globálních.

Příklad:

```
// nejprve obecná definice - v souboru "spolecne.css" bude:
h1 { font: 24px/28px ...; color: black; font-weight: bold }

// v souboru "firma.css" bude uvedeno:
h1 { color: red }

// v souboru "produkty.css" pak:
h1 { color: blue }

// na stránkách firemní prezentace načteme nejprve obecné, poté firemní css
<link rel="stylesheet" href="spolecne.css" type="text/css">
<link rel="stylesheet" href="firma.css" type="text/css">

// na stránkách produktů načteme nejprve obecné, poté produktové css
<link rel="stylesheet" href="spolecne.css" type="text/css">
<link rel="stylesheet" href="produkty.css" type="text/css">
```

Firemní prezentace i stránky produktů převezmou společné styly pro celý web, jen firemní prezentace bude mít navíc červené titulky a stránky produktů modré - všude jinde budou černé a až na barvu budou všechny atributy titulků shodné.

Je také důležité hlídat si specifičnost jednotlivých pravidel. Nejsnazší pomůckou je pamatovat si, že nejspecifičtější jsou pravidla s #ID, poté s třídou a až nakonec samotné značky. Jen při opravdu složitých konstrukcích je nutné si spočítat specifičnost podle algoritmu CSS. Většinou to nutné není:

```
p { align: left }
.pravy { align: right }
#odstavecnastred { align: center }
...
<p>odstavec 1</p>
<p class="pravy">odstavec 2</p>
<p id="odstavecnastred" class="pravy">odstavec 3</p>
```

V tomto příkladu bude první odstavec zarovnaný doleva (pravidlo pro značku <P>). Druhý odstavec doprava (pravidlo pro třídu je specifičtější) a třetí na střed

(pravidlo s #ID je specifičtější než třída i než značka). Pořadí zde nemá vliv - podle specifičnosti se třídí nejdříve a teprve u stejně specifických pravidel se rozhoduje podle pořadí.

Často je praktické použít kaskádování a dědičnost přímo v po sobě následujících definicích, nebo dokonce v definici jediné. Pravidla kaskádování zajistí, že budou eliminována pravidla popisující stejnou vlastnost stejného prvku a použije se jen to nejspecifičtější a nejpozdější.

Příklady:

Chceme-li definovat několik prvků, které mají mnoho podobných vlastností a liší se pouze v detailech, můžeme nejprve napsat společná pravidla pro všechny najednou a u jednotlivých prvků pak pouze předefinujeme nebo doplníme odlišnosti:

```
h1, h2, h3, h4 { font-family:sans-serif; color:blue; border-bottom: 1ex solid blue }
h4 { border-bottom-style: none }
```

Všechny titulky `h1-h4` budou bezpatkovým písmem, modré, s linkou vespod; pouze titulek h4 bude bez linky. Při počítání hodnot vlastností `border-bottom-style` pro značku `<h4>` klient zjistí, že v seznamu pravidel má (přinejmenším) dvě pravidla: nejprve "1ex solid blue" (což je *shorthand*, který je nejprve rozložen na jednotlivé složky - jednou z nich je i `border-bottom-style`) a potom "none". Obě pravidla jsou stejně specifická (selektory oddělené čárkami se považují za několik samostatných selektorů se shodnou definicí), druhé pravidlo je ale uvedeno později. Klient proto to první "zapomene" a použije pravidlo druhé, neboť má vyšší prirotu - značku `<h4>` vykreslí bez spodní linky.

Jinou možností je využití právě shorthandů - místo složitého a zdlouhavého vypisování jednotlivých pod-vlastností opět můžeme využít pravidel kaskády i v jediné definici:

```
#menu { border: 1px solid black; border-left: 4px solid red }
```

Zde opět při vykreslování objektu id="menu" klient najde dvě pravidla pro stejné vlastnosti (`border` zde zastupuje `border-top-width + border-top-style + border-top-color + border-right-width + ... + border-left-color`; obdobně `border-left` popisuje tři vlastnosti najednou). U všech vlastností, které nejsou "předefinovány", použije první pravidlo, ale u vlastností týkajících se `border-left` zjistí, že je "přebilo" pravidlo s vyšší prioritou a použije tedy to druhé. Teprve potom objekt vykreslí. Výsledkem tedy bude rámeček kolem celého objektu, nahoře, vpravo a dole černý, jednopixelový, vlevo pak červený o síle 4px.

3.7 Volba jednotek

Pokud máme udělat "natahovací" stránku, tj. takovou, která se bude přizpůsobovat velikosti okna prohlížeče, používáme procenta. A to nejen pro šířku boxů, ale i pro okraje (margin, padding). Stojí to mnohdy více počítání, ale výsledek bude zaručen.

Chceme-li mít plnou kontrolu nad velikostí písma, použijte pixely [px] namísto bodů [pt]. Velikost (typografického) bodu totiž závisí na rozlišení obrazovky, velikosti monitoru, operačním systému, prohlížeči i jiných faktorech. Při rozlišení 72 dpi bude 12pt písmo výrazně menší než při rozlišení 96 dpi. Pokud však použijete velikost písma 12px, bude všude (přibližně) stejně velké.

Chceme-li proporčně zachovávat rozměry, i pokud si uživatel v prohlížeči změní velikost písma - např. odstup textu od rámečku (padding) nebo rozestup mezi odstavci (top-margin) - používejme jednotky em a ex (1 ex = 1/2 em), které odpovídají stupni velikosti písma. Pokud totiž použijeme fixní jednotky (pt, px atd.), bude při zvětšení velikosti písma text opticky "nalepený" na rámeček, při zmenšení naopak bude příliš daleko.

Při vytváření stylů pro ne-obrazovkový výstup, např. pro tisk používáme běžné centimetry a milimetry. Formátování na papír A4 bude mnohem snazší a přesnější.

Jak již bylo uvedeno v popisu syntaxe pravidel CSS, definice vlastností stylů se uvádějí ve formě `vlastnost: hodnota`. Tím se danému prvku přiřadí vlastnost s uvedenou hodnotou. Každá vlastnost přitom může nabývat jen některých typů hodnot, které jsou pro každou vlastnost přesně vymezeny: např. `font-size` (velikost písma) může nabývat pouze hodnot určujících použitelnou velikost písma - a např. hodnoty typu úhel, čas nebo barva zde nemají smysl. Použitelné hodnoty pro každou vlastnost naleznete vždy v její specifikaci. Jiné hodnoty nejsou povoleny a klient je bude ignorovat.

3.7.1 Číselné hodnoty

Některé vlastnosti mohou nabývat číselných hodnot - a to buď celočíselných (budeme je značit <celé číslo>) nebo reálných (<číslo>). Celé číslo tvoří jedna nebo více číslic 0-9, před nimiž může být uvedeno znaménko + nebo -. Reálným číslem je pak buď celé číslo, nebo číslo desetinné. Desetinná část se vždy odděluje tečkou podle americké zvyklosti, nikoli čárkou jako u nás. Pokud je v desetinném čísle 0 jednotek, nemusí se nula uvádět a je možné začít přímo desetinnou tečkou. V CSS se většinou používají čísla nezáporná.
Příklady: **`<celé číslo>`** je `0`, `1`, `10`, `-123`, `+12`; **`<číslo>`** je např. `1`, `3.1415927`, `-99.9`, `0.5`, `.5` i `-.5`.

3.7.2 Velikosti

Velikostmi se definují vodorovné a svislé vzdálenosti a rozměry. V popisech vlastností je budeme značit **`<velikost>`** a jsou vždy ve formátu: `<číslo><jednotka>` (mezi číslem a jednotkou nesmí být mezera!). Pokud je `<číslo>` nula, není nutné uvádět jednotku - postačí tak pouze `0`. Ačkoli některé vlastnosti teoreticky povolují použití záporných velikostí, může to způsobit problémy při formátování a klient je nemusí podporovat (vezme pak nejbližší použitelnou hodnotu). Je proto lepší se záporným hodnotám velikostí spíše vyhýbat. Velikosti jsou dvojího typu:

3.7.2.1 Relativní velikosti

Velikosti vztažené k jiné hodnotě. Jsou výhodnější než hodnoty absolutní, neboť se snáze přizpůsobují danému výstupnímu zařízení a dobře se konverzují mezi různými médii - např. mezi výstupem na obrazovku a na tiskárnu.

em

aktuální velikost písma. Odpovídá *přepočítané hodnotě* velikosti písma v daném prvku. Pouze v samotné definici velikosti písma (`font-size`) se přebírá velikost písma rodičovského prvku. **Příklad:** `P {line-height:1.5em}` - výška řádku (proklad) odstavců bude 1,5násobkem velikosti písma v tomto odstavci, naproti tomu `P {font-size:1.5em}` - velikost písma v odstavci bude 1,5násobkem

velikosti písma rodičovského prvku (tedy např. `BODY`, pokud `P` není vloženo v ničem jiném). Pozn. pro typografy: `em` je *čtverčík*.

ex

odpovídá přibližně výšce malých znaků "bez nožiček" (x, m, n). Pro jednoduchost lze předpokládat, že `ex` je asi polovina `em` - jakkoli to není zcela přesné, v běžné praxi je to obvykle dostačující.

px

pixely - podle daného zobrazovacího zařízení. Na běžném monitoru odpovídá `px` jednomu bodu obrazovky, při vysokých hustotách zobrazovaných bodů se `px` může přepočítávat podle celkem složitého algoritmu na jinou velikost. Čistě pro zajímavost - exaktně je `px` specifikován jako bod při rozlišení 90 dpi, pozorovaný ze vzdálenosti natažené paže (28 palců). Jeden `px` tak na obrazovce odpovídá přibližně 1/90 palce, tedy asi 0,28 mm, a na tiskárně s rozlišením 600 dpi pak přibližně 5 bodům tiskárny (asi 0,25 mm). Pokud je však vzdálenost pozorovatele jiná, je potřeba tento poměr zachovat - při menší vzdálenosti od očí může být `px` menší, při větší vzdálenosti pak větší: např. pro zobrazení na TV obrazovce sledované z 3,5 metru by měl `px` odpovídat asi 1,4 mm.

Příklady:

```
BODY {margin: 2em}
P {padding: .5ex}
H1 {border-bottom-width: 1px}
```

3.7.2.2 Absolutní velikosti

Jsou vhodné, jen pokud známe fyzické vlastnosti konkrétního výstupního média - např. velikost papíru v tiskárně nebo přesný rozměr a rozlišení jedné konkrétní obrazovky.

cm

centimetry

mm

milimetry

in

palce

pt

"typografické body" - 1pt = 1/72 palce

pc

pica - 1px = 12pt

Příklady:

```
body {margin: 1.5cm}
div.ramecek {border-width: 2mm}
p {padding-left: 2in}
h1 {font-size: 36pt}
h2 {font-size: 2pc}
```

3.7.3 Procenta

Procenta se uvádějí ve tvaru `<číslo>%` (opět bez mezery mezi číslem a jednotkou [%]). Procentní hodnota vlastnosti je vždy relativní vzhledem k nějaké jiné vlastnosti, proto u každé vlastnosti, která povoluje jako hodnotu **`<procento>`**, je vždy uvedeno, k čemu se procentní poměr vztahuje - zda k jiné vlastnosti téhož prvku, k prvku rodičovskému či nadřazenému nebo hodnotě převzaté z formátovacího kontextu

(např. šířka *omezujícího bloku*).
Příklad: `h1 {font-size: 200%}` - pro relativní velikost `font-size` se za základ bere spočítaná velikost písma rodičovského prvku, nadpis `H1` tedy bude zobrazen dvakrát větším písmem než jeho mateřský prvek (např. `BODY`).

3.7.4 URI

Uniform Resource Identificator - platná webová adresa. V popisech vlastností se značí **`<uri>`** a její hodnotou je platné URI (URL/URN podle specifikace [RFC1738] a [RFC1808]), volitelně ohraničené párem jednoduchých nebo dvojitých úvozovek - tedy `uri`, `'uri'` nebo `"uri"`. Použité URI může být absolutní i relativní a v definicích CSS se vždy zapisuje ve formě "funkce": `url(...)` - např.:
`BODY { background: url("/img/pozadi.gif") },`
`@import url(http://www.neco.cz/styly.css);` atd.

3.7.5 Barvy

Značí se **`<barva>`** a hodnotou je definice barvy buď pomocí *klíčového slova* nebo *číselnou specifikací RGB*. Jako *klíčová slova* se používají barvy definované HTML 4.0 (`aqua`, `black`, `blue`, `fuchsia`, `gray`, `green`, `lime`, `maroon`, `navy`, `olive`, `purple`, `red`, `silver`, `teal`, `white` a `yellow`). Číselné hodnoty odpovídají barevnému prostoru sRGB a mohou být zapsány v jedné z následujících forem:

`#rrggbb`
: Za znakem `#` následují tři hexadecimální čísla (00-FF), odpovídající popořadě hodnotám červené (R), zelené (G) a modré (B) složky barvy. `#000000` je minimální hodnota (černá) a `#FFFFFF` je maximální (bílá). **Např:** `#FF0000` (červená, `red`), `#0000FF` (modrá, `blue`), `#FFFF00` (žlutá, `yellow`), `#1AC8FE` atd.

`#rgb`
: Za znakem `#` následují pouze tři hexadecimální číslice. Tento zápis se převádí na předchozí delší zápis ve tvaru #rrggbb - např. hodnota `#fb6` je vlastně pouze zkráceným zápisem hodnoty `#ffbb66`. Tento zápis je výhodný tam, kde chceme použít tzv. *bezpečné barvy* (web-safe colors), což je sada 216 barev omezená na RGB hodnoty 00, 33, 66, 99, CC a FF, které se přednostně zobrazují co nejvěrněji v každém prohlížeči. Stačí tedy používat tento zkrácený zápis pouze s číslicemi 0, 3, 6, 9, C a F - např. `#309` (=`#330099`), `#FF6` (=`#FFFF66`), `#03C` (=`#0033CC`) jsou tedy "bezpečné" barvy.

`rgb(r,g,b)`
: Zápis ve tvaru funkce rgb(), jejímiž parametry jsou celá čísla 0-255, opět popořadě odpovídající hodnotám barevných složek R, G a B. Např. `rgb(0,0,255)` odpovídá `#0000FF` (červená).

`rgb(r%,g%,b%)`
: Parametry funkce jsou (reálná) čísla udávající procentní poměr RGB složek v rozmezí 0.0% až 100.0%. **Např.:** `rgb(100%, 100%, 100%)` = `rgb(255, 255, 255)` = `#FFF` = `#FFFFFF` = bílá.

3.7.6 Úhly

Používají se především ve vlastnostech pro zvukový výstup (aural media) a značí se **`<úhel>`**. Zapisují se ve tvaru `<číslo><jednotka>`, kde číslo může být desetinné i záporné a za ním následuje jedna z jednotek `deg` (stupně), `rad` (radiány)

nebo `grad` (grady). Klient převádí zadanou hodnotu na úhel v rozsahu 0-360 stupňů (`deg`). **Např.:** `90deg` =`100grad` = `1.570796326794897rad` = pravý úhel.

3.7.7 Čas

Používají se opět v aurálních stylech, označení **`<čas>`**. Zapisují se ve formátu `<číslo><jednotka>`, kde číslo je nezáporné a jednotkou jsou `ms` (milisekundy) nebo `s` (sekundy). **Např.:** `150ms`, `30s`.

3.7.8 Frekvence

Opět pro aurální styly (**`<frekvence>`**), ve formátu `<číslo><jednotka>`, kde číslo je nezáporné a jednotkou je `Hz` (Hertz) nebo `kHz` (kiloHertz). **Např.:** `200Hz` (basový tón), `10kHz` (velmi vysoký tón).

3.7.9 Řetězce

Skupina znaků uzavřená do jednoduchých nebo dvojitých úvozovek (značí se **`<řetězec>`**). Pokud se uvnitř řetězce mají vyskytovat úvozovky nebo znak pro nový řádek, musí být uvozeny znakem `\` - místo znaku `"` se uvnitř řetězce píše `\"`, obdobně `\'` namísto `'`, nový řádek se zapisuje jako `\A`. Má-li být v řetězci přímo znak `\`, musí být rovněž uvozen: `\\`. Chceme-li z nějakých důvodů (třeba kvůli přehlednosti) rozdělit řetězec na více řádek, musíme na konci takového řádku uvést `\`.

Např.: *"toto je text", 'zde jsou \"úvozovky\" uvnitř textu',*

"zde potřebuji napsat \\, což obrácené lomítko", "jeden řádek \A a toto už je na novém řádku".

Když potřebuji řetězec opticky rozdělit na více řádek:

*a[title="tady mám nějaký hodně dlouhý *

*text a nechci to psát všechno na jeden *

*řádek. ale je to jen pro mou přehlednost, *

ve skutečnosti v řetězci žádné entery nejsou."]

3.8 Boxy a jejich vlastnosti

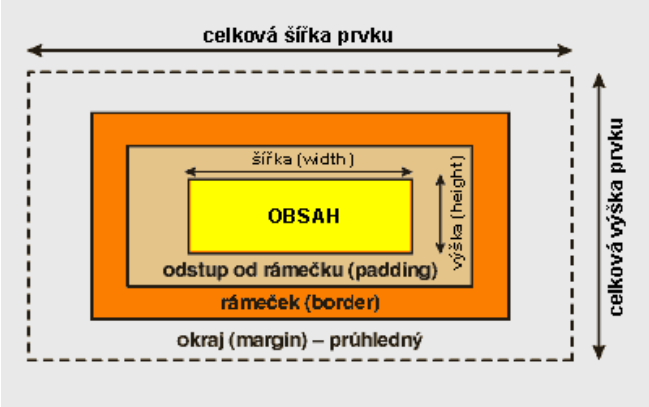
V původní dokumentaci se těmto *přihrádkám, zásuvkám, krabičkám, bednám* říká souhrnně anglicky `boxy` - tento termín použijeme i zde. Specifikace CSS popisuje, jakým způsobem při tvorbě stromu dokumentu vznikají boxy pro každý prvek, a definuje pravidla pro to, jak jsou tyto boxy vykreslovány na výstupu.

Jednotlivé boxy podle svých vlastností a umístění v dokumentu mohou tvořit vlastní **blok** (obdélníková oblast, uvnitř níž je zobrazen obsah boxu), nebo vlastní blok nevytvářejí - pak jsou umístěny v jiném, nadřazeném bloku a jejich obsahem jsou **řádky**. Z toho důvodu je nejvyšší, kořenový box dokumentu vždy typu blok. Všechny boxy, ať již jsou to bloky nebo řádky, mají některé společné vlastnosti, popisující obdélník, v němž se jejich obsah nachází. Pokud se jedná o blok, tyto vlastnosti popisují přímo obdélník bloku; pokud se o blok nejedná, tyto vlastnosti popisují jednotlivé obdélníky tvořící každý řádek boxu. Všechny řádky pak mají stejné vlastnosti, definované pro tento box.

3.8.1 Vlastnosti boxů

Každý box je v prvé řadě tvořen svým **obsahem** (text, obrázek atd.). Kolem něj je několik oblastí, které mohou mít libovolnou (tedy i nulovou) velikost:

- vnitřní oblastí je samotný **obsah** boxu. Jeho rozměry určují vlastnosti `width` a `height`
- kolem obsahu je oblast **padding** (odstup obsahu od rámečku)
- vně oblasti padding je oblast **border** (rámeček)
- kolem rámečku je pak oblast **margin** (okraj, odstup prvku od okolních objektů)



Rozměry všech oblastí mohou být definovány jedinou hodnotou, nebo zvlášť pro každou stranu - vlevo (left), vpravo (right), nahoře (top) a dole (bottom). Ke každé vlastnosti tak existují vždy čtyři pod-vlastnosti, umožňující samostatně definovat jeden ze čtyř rozměrů dané oblasti. V CSS je zvykem postupovat shora ve směru hodinových ručiček - zvykneme si tedy postupovat v tomto pořadí - top, right, bottom, left:

- `padding` = `padding-top`, `padding-right`, `padding-bottom` a `padding-left`
- `border` = `border-top`, `border-right`, `border-bottom`, `border-left`
- `margin` = `margin-top`, `margin-right`, `margin-bottom`, `margin-left`

Celkové rozměry boxu jsou tvořeny součtem rozměrů všech jednotlivých oblastí:

- celková šířka boxu = margin-left + border-left + padding-left + width + padding-right + border-right + margin-right
- celková výška boxu = margin-top + border-top + padding-top + height + padding-bottom + border-bottom + margin-bottom

3.8.2 Vlastnosti popisující okraj boxu:

`margin-top`, `margin-right`, `margin-bottom`, `margin-left` a `margin`

Tyto vlastnosti specifikují rozměry oblasti margin daného boxu, tedy odstup boxu od okolních prvků v dokumentu. Vlastnost `margin` je zkratkou (shorthand), která definuje shodné hodnoty pro více dílčích vlastností současně.

Popis vlastností `margin-top`, `margin-right`, `margin-bottom`, `margin-left`:

- **povolené hodnoty:**
  - `<velikost>` - specifikuje fixní rozměr
  - `<procenta>` - rozměr relativní k velikosti omezujícího bloku
  - `auto` - výpočet rozměrů a okrajů
  - `inherit` - zdědí spočítanou hodnotu z rodičovského prvku
- **výchozí hodnota:** 0
- **lze použít:** na všechny prvky
- **dědí se:** ne
- **procenta:** vztahují se k rozměrům omezujícího bloku
- **media:** vizuální

3.8.3 Vlastnosti popisující odstup obsahu od rámečku:

`padding-top`, `padding-right`, `padding-bottom`, `padding-left` a `padding`

Tyto vlastnosti specifikují rozměry oblasti padding daného boxu, tedy odstup vlastního obsahu boxu od případného rámečku. Vlastnost `padding` je zkratkou (shorthand), která definuje shodné hodnoty pro více dílčích vlastností současně.

Popis vlastností `padding-top`, `padding-right`, `padding-bottom`, `padding-left`:

- **povolené hodnoty:**
  - `<velikost>` - specifikuje fixní rozměr
  - `<procenta>` - rozměr relativní k velikosti omezujícího bloku
  - `inherit` - zdědí spočítanou hodnotu z rodičovského prvku
- **výchozí hodnota:** 0
- **lze použít:** na všechny prvky
- **dědí se:** ne
- **procenta:** vztahují se k rozměrům omezujícího bloku
- **media:** vizuální

Příklady:

- `p {margin-left: 2px}` - nastaví levý okraj na šířku 2px, ostatní okraje budou nulové (výchozí hodnota)
- `h1 {padding-top: 1em; padding-bottom: 2em}` - nastaví horní padding na hodnotu 1em a dolní na 2em
- `p {margin: 2em 1ex}` - horní a dolní okraj bude 2em, pravý a levý pak 1ex
- `h1 {padding: 5% 0 5% 10%}` - odstup obsahu od rámu bude shora i zdola 5 % výšky omezujícího bloku, zprava nulový a zleva o velikosti 10 % šířky omezujícího bloku

Příklad slučování okrajů:

```
<P style="margin-bottom:2em">První odstavec</P>
<P style="margin-top:1em">Druhý odstavec</P>
```

Mezi odstavci bude zobrazena mezera pouze o velikosti většího z obou okrajů, tedy 2em.

3.8.4 Vlastnosti rámečku (border)

Tyto vlastnosti popisují postupně rozměry, barvu a styl rámečku. Prohlížeče mohou vykreslovat rámečky u některých prvků (tlačítka, formulářové prvky atd.) specifickým způsobem, který se liší od ostatních, "běžných" prvků - například zaoblené okraje tlačítek, 3D efekty apod. Nastavíte-li těmto prvkům v CSS specifické vlastnosti rámečku, mohou se vykreslovat velmi neobvykle až podivně a je potřeba si jejich vzhled ověřit na různých platformách.

Vlastnosti popisující velikost rámečku:
`border-top-width`, `border-right-width`, `border-bottom-width`, `border-left-width` a `border-width`

Tyto vlastnosti určují "tloušťku" rámečku.

Popis vlastností `border-top-width`, `border-right-width`, `border-bottom-width`, `border-left-width`:

- **povolené hodnoty:**
  - `thin` | `medium` | `thick` - tenký | střední | tlustý - přesná hodnota závisí na konkrétním prohlížeči
  - `<velikost>` - specifikuje fixní rozměr, hodnota nesmí být záporná
  - `inherit` - zdědí spočítanou hodnotu z rodičovského prvku
- **výchozí hodnota:** `medium`
- **lze použít:** na všechny prvky
- **dědí se:** ne
- **procenta:** nelze použít
- **media:** vizuální

Popis vlastnosti `border-width`:

- **povolené hodnoty:** stejně jako v dílčích vlastnostech
- **výchozí hodnota:** viz dílčí vlastnosti
- **lze použít:** na všechny prvky
- **dědí se:** ne
- **procenta:** nelze použít
- **media:** vizuální

Vlastnosti popisující barvu rámečku:
`border-top-color`, `border-right-color`, `border-bottom-color`, `border-left-color` a `border-color`

Tyto vlastnosti určují barvu rámečku.

Popis vlastností `border-top-color`, `border-right-color`, `border-bottom-color`, `border-left-color`:

- **povolené hodnoty:**
  - `<barva>`
  - `inherit` - zdědí spočítanou hodnotu z rodičovského prvku

- **výchozí hodnota:** hodnota vlastnosti `color`
- **lze použít:** na všechny prvky
- **dědí se:** ne
- **procenta:** nelze použít
- **media:** vizuální

Popis vlastnosti `border-color`:

- **povolené hodnoty:** stejně jako v dílčích vlastnostech | `transparent`
- **výchozí hodnota:** viz dílčí vlastnosti
- **lze použít:** na všechny prvky
- **dědí se:** ne
- **procenta:** nelze použít
- **media:** vizuální

Vlastnosti popisující styl rámečku:
`border-top-style`, `border-right-style`, `border-bottom-style`, `border-left-style` a `border-style`

Tyto vlastnosti určují styl čáry rámečku.

Popis vlastností `border-top-style`, `border-right-style`, `border-bottom-style`, `border-left-style`:

- **povolené hodnoty:**
  - `none` - žádný rámeček. Tato hodnota současně nastaví nulový `border-width`
  - `hidden` - totéž jako `none`, pouze se chová jinak u tabulek (viz dále)
  - `dotted` - tečkovaná čára (.....)
  - `dashed` - čárkovaná čára (`- - - - -`)
  - `solid` - plná čára (_____)
  - `double` - dvojitá plná čára (=====). Součet tloušťky čar a mezery mezi nimi odpovídá šířce `border-width`
  - `groove` - žlábek; čára vypadající, jako by byla vyryta do podkladu
  - `ridge` - hřbet; opak ke `groove` - čára vypadající, jako by vystupovala z podkladu
  - `inset` - rámeček vykreslený tak, že celý box vypadá jako zapuštěný do podkladu
  - `outset` - opak k `inset` - rámeček vykreslený tak, že celý box vystupuje z podkladu
  - `inherit` - zdědí spočítanou hodnotu z rodičovského prvku
- **výchozí hodnota:** `none`
- **lze použít:** na všechny prvky
- **dědí se:** ne
- **procenta:** nelze použít
- **media:** vizuální

Popis vlastnosti `border-style`:

- **povolené hodnoty:** stejně jako v dílčích vlastnostech,

- **výchozí hodnota:** viz dílčí vlastnosti
- **lze použít:** na všechny prvky
- **dědí se:** ne
- **procenta:** nelze použít
- **media:** vizuální

Další zkratky pro vlastnosti rámečku:
`border-top`, `border-right`, `border-bottom`, `border-left` a `border`

Tyto zkratky definují šířku, styl a barvu rámečku současně pro horní, pravou, spodní a levou stranu rámečku, resp. pro všechny strany rámečku najednou. Hodnotou těchto vlastností je trojice

šířka (viz `border-width`), styl (viz `border-style`) a barva (viz `border-color`)
nebo
hodnota `inherit` (zdědí všechny hodnoty z rodičovského prvku)

Jednotlivé hodnoty v seznamu jsou přitom odděleny mezerami (resp. *whitespace*) a není-li některá z nich uvedena, použije se výchozí hodnota pro danou dílčí vlastnost.

Příklady:

`p { border-left-width: thin; border-left-style: solid; border-left-color: black}` - levý rámeček odstavce bude vykreslen tenkou plnou červenou čarou.
`p { border-color: black }` - všechny čtyři strany rámečku budou černé
`p { border-bottom: 1px dotted blue }` - dolní strana rámečku bude vykreslena 1px silnou tečkovanou modrou čarou.
`p { border: thick solid orange }` - kolem celého odstavce bude tučný oranžový rámeček.
`p { border: inherit }` - hodnoty rámečku se převezmou z rodičovského prvku, tedy z prvku, v němž je daný odstavec umístěn.

Společně se zkratkami je vhodné využívat dědičnosti a kaskádování CSS pro snadné definování složitějších konstrukcí, jejichž popis dílčími vlastnostmi by byl příliš složitý:

`h1.nadpis { border: thin solid black; border-left: medium solid red }` - první pravidlo nastaví celý rámeček 2px silný, plnou černou čarou, druhé pravidlo (s vyšší prioritou) pak přepíše vlastnosti pro levou stranu rámečku na silnější červenou čáru. Nadpis tak bude orámován vlevo silněji červeně a na ostatních stranách tenčí černou čarou.

3.8.5 Poznámka k pozadí jednotlivých oblastí:

Styl pozadí jednotlivých oblastí boxu je definován následujícím způsobem:

- **Obsah** a oblast **padding** jsou vykresleny s pozadím odpovídajícím vlastnosti `background` daného boxu.
- Oblast **border** je rovněž vykreslena na stejném pozadí jako obsah boxu; u stylů čar `solid`, `dotted`, `dashed` a `double` se k vykreslení čáry použije barva

specifikovaná v `border-color`; u "plastických" stylů `groove`, `ridge`, `inset` a `outset` se při vykreslování také přihlíží k barvě obsahu (vlastnost `color` daného prvku).
- Oblast **margin** je vždy průhledná.

Příklad:

```
body { background:yellow; color:black }
p.odstavec { padding:1em; background:orange; border: 8px dashed black }
...
<body>
...
<p class="odstavec">text obsahu text obsahu...</p>
...
</body>
```

V korektním prohlížeči byste měli vidět tento výsledek:

3.9 Pozicování

Každý box se v CSS zobrazuje pomocí jednoho ze tří zobrazovacích schémat:

- **normální tok** dokumentu (*normal flow*) - což zahrnuje blokové formátování blokových boxů, řádkové formátování inline boxů, relativní pozicování a formátování boxů typu `compact` a `run-in`,
- **plovoucí boxy** - box je nejprve umístěn do *normálního toku* dokumentu, následně je z něj vyňat a posunut co nejvíce doprava či doleva, přičemž ostatní obsah jej může obtékat po jeho straně.
- **absolutní pozicování** - box je zcela vyjmut z normálního toku (takže nemá žádný vliv na své případné sousední boxy) a je mu přidělena poloha vzhledem k jeho *omezujícímu bloku.*

Zobrazovací schéma (a tím i příslušný zobrazovací algoritmus) si designér volí pomocí vlastností `position` a `flow` a zpřesňuje pomocí dalších níže uvedených vlastností.

3.9.1 Popis vlastnosti `position`

- **povolené hodnoty:**

    - `static` | `relative` | `absolute` | `fixed` | `inherit`
- **výchozí hodnota:** `static`
- **lze použít:** na všechny prvky, ale ne na generovaný obsah
- **dědí se:** ne
- **procenta:** nelze použít
- **media:** vizuální

Popis jednotlivých hodnot:

`static`
: box je normálním boxem, je umístěn v normálním toku dokumentu. Vlastnosti `left` a `top` se nepoužívají.

`relative`
: box je nejprve umístěn v normálním toku a následně je relativně posunut ze své normální pozice o hodnoty specifikované vlastnostmi `left`, `top`, `right` a `bottom`. Následující box se formátuje, jako by tento box posunut nebyl.

`absolute`
: pozice boxu je specifikována vzhledem k jeho *omezujícímu bloku* vlastnostmi `left`, `top`, `right` a `bottom`. Tento box je vyjmut z normálního toku a nemá žádný vliv na vzhled a formátování následujících prvků.

`fixed`
: pozice boxu je spočítána stejně jako v případě `absolute`, s tím rozdílem, že box je zafixován vůči nějakému pevnému bodu - v případě plynulých (continuous) médií bývá zafixován vzhledem k oknu prohlížeče (takže zůstává na místě, i když se dokument roluje), u stránkových (paged) médií to bývá vůči stránce (takže se vytiskne na stejném místě na každé stránce).

`inherit`
: box zdědí vlastnosti z rodičovského prvku

3.9.2 Popis vlastností left, top, right a bottom

- **povolené hodnoty:**
    - `<velikost>` - fixní vzdálenost od hrany příslušného bloku
    - `<procenta>` - procentní poměr k šířce (u `left` a `right`), resp. výšce (`top` a `bottom`) omezujícího bloku. Pokud výška omezujícího bloku není zadána, tato hodnota se interpretuje jako `auto`.
    - `auto`
    - `inherit` - zdědí spočítanou hodnotu z rodičovského prvku
- **výchozí hodnota:** `auto`
- **lze použít:** na pozicované prvky
- **dědí se:** ne
- **procenta:** vzhledem k hranám *omezujícího bloku*
- **media:** vizuální

Hodnoty jednotlivých vlastností specifikují, jak daleko od levé, horní, pravé, resp. dolní hrany příslušného bloku má být umístěna levá, horní, pravá, resp. dolní hrana boxu. U absolutně pozicovaných boxů je oním *příslušným blokem* jejich omezující blok, u relativně pozicovaného boxuse vzdálenost počítá od jeho původní polohy v normálním toku dokumentu.

3.9.3 Normální tok dokumentu

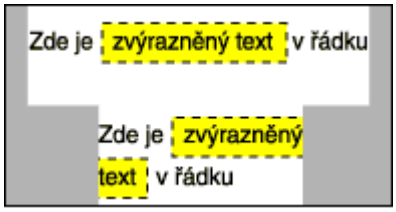
Blokové boxy se umisťují **pod sebe,** jeden za druhým od horní hrany *omezujícího bloku.* Vzdálenosti mezi nimi závisejí na jejich vlastnostech `margin`, přičemž se tyto *okraje* slučují. Levá hrana každého boxu (tj. až vnější hrana oblasti *margin*) se dotýká levé hrany *omezujícího bloku*.

Řádkové (inline) boxy se umisťují za sebe, jeden vedle druhého od horní hrany omezujícího bloku, přičemž se bere ohled i na horizontální rozměry padding, border a margin jednotlivých boxů. Vzájemné výškové uspořádání sousedních boxů ovlivňuje jejich vlastnost vertical-align. Obdélník obsahující inline boxy v jedné linii se nazývá řádek (line box). Výška řádku se řídí vlastním algoritmem (viz další kapitolu), je však vždy dostatečná pro výšku nejvyššího boxu, který obsahuje.

Pokud se některý inline box již nevejde do šířky řádku, jeho obsah je rozmístěn do dvou či více řádků seskupených pod sebou. Řádky navazují přímo na sebe, bez mezer, nikdy se nepřekrývají - vzdálenost *obsahu* jednotlivých řádek je ale ovlivněna spočítanou výškou řádku (svislé rozměry *padding, border* a *margin* výšku řádku neovlivňují). Řádek je vždy tak široký, jak je široký jeho omezující blok (jejich krajní hrany se dotýkají) - pouze v případě, že se v okolí vyskytuje plovoucí objekt, je prostor daný *omezujícím blokem* zmenšen o jeho šířku.

Pokud je celkový součet šířek inline boxů na jednom řádku menší než šířka řádku, je obsah boxů na řádku rozmístěn podle vlastnosti `text-align`. Pokud je tato vlastnost `justify`, mohou být všechny inline boxy roztaženy do plné šířky řádku.

Je-li nějaký inline box rozdělen do dvou řádků, v místě rozdělení se **neprojeví** jeho formátovací vlastnosti (`margin`, `padding`, `border`, `text-decoration` atd.), tak jako na okrajích inline boxu. Box je zobrazen tak, jako by byl zformátován na jednom řádku, následně rozříznut a druhá část byla umístěna na začátek následujícího řádku:



Formátování (margin, padding, border atd.) se projeví pouze **před** slovem "zvýrazněný" a **za** slovem "text" - v místě "řezu" se neprojeví.

Při **relativním pozicování** vše probíhá stejně - pouze má-li některý z inline boxů vlastnost `position:relative`, berou se v potaz i hodnoty jeho vlastností `left`, `top`, `right` a `bottom` a o tyto hodnoty se box posune vůči své původní pozici. Tímto způsobem lze vytvářet různé efekty, mocniny a indexy a dynamickým měněním těchto hodnot (pomocí skriptu) vytvářet animace.

3.9.4 Plovoucí boxy

Každý plovoucí box musí mít specifikovánu šířku (*width*). Stává se automaticky blokem (blokovým boxem) a je posunut ke straně (doleva či doprava), až se dotýká hrany omezujícího bloku nebo jiného plovoucího boxu. Jeho horní hrana je na úrovni horní hrany příslušného řádku (nebo na úrovni spodní hrany předchozího bloku, pokud

zde žádné řádky nejsou). Pokud na řádce již není dostatek místa pro tento box, přesune se na další řádek - tak dlouho, až místo je.

Další obsah následující za plovoucím objektem "obtéká" kolem pravé, resp. levé hrany plovoucího objektu. Tato možnost může být vypnuta pomocí vlastnosti `clear` - další obsah potom pokračuje až za spodní hranou plovoucího boxu.

Popis vlastnosti `float`

- **povolené hodnoty:**
  - `left` | `right` | `none` | `inherit`
- **výchozí hodnota:** `none`
- **lze použít:** na všechny prvky, kromě pozicovaných a generovaného obsahu
- **dědí se:** ne
- **procenta:** nelze použít
- **media:** vizuální

Popis jednotlivých hodnot:

left
: box je plovoucí vlevo, tzn. posouvá se k levé hraně a okolní obsah jej obtéká zprava. Hodnota `display` se ignoruje, pokud není `none`

right
: stejně jako předchozí, ale posouvá se doprava, obsah jej obtéká zleva

none
: box není plovoucí

inherit
: hodnota se zdědí z rodičovského prvku

Popis vlastnosti `clear`

- **povolené hodnoty:**
  - `left` | `right` | `both` | `none` | `inherit`
- **výchozí hodnota:** `none`
- **lze použít:** blokové prvky
- **dědí se:** ne
- **procenta:** nelze použít
- **media:** vizuální

Tato vlastnost určuje, které hrany **nesmí** být mít box společné s nějakým předchozím plovoucím boxem. Hodnoty `left`, `right` a `both` znamenají, že v boxu nesmí být plovoucí box vlevo, vpravo, resp. ani na jedné straně. Hodnota `none` plovoucí boxy povoluje.

3.9.5 Absolutní pozicování

Při absolutním pozicování (`position` je `absolute` nebo `fixed`) je box zcela vyňat z normálního toku dokumentu a vytváří nový *hlavní blokový box* a tím i *nový normální tok* pro své potomky - tzn, že všechny boxy, které jsou obsaženy uvnitř tohoto boxu se pozicují již vzhledem k tomuto bloku. Absolutně pozicované boxy, které nejsou

navzájem vnořené, spolu žádným způsobem neinteragují a mohou se překrývat - stejně jako s obsahem normálního toku dokumentu.

Vrstvy v dokumentu

V případě absolutního pozicování je potřeba mít někdy kontrolu nad tím, jak se boxy navzájem překrývají. Slouží k tomu vlastnost `z-index`, která přidává dokumentu zdánlivý třetí rozměr. Pokud tato vlastnost není specifikována (výchozí hodnota `auto`), vykreslují (a překrývají) se boxy tak, jak za sebou následují ve *stromu dokumentu.* Nastavením jiné hodnoty této vlastnosti můžeme překrývání boxů ovlivnit - čím vyšší hodnota, tím "blíže" k pozorovateli se box nachází a může překrýt boxy s nižšími hodnotami.

Kořen dokumentu na začátku vytvoří zásobník vrstev, v jehož rámci jsou číslovány všechny další boxy, které dokument obsahuje - on sám se do tohoto zásobníku umístí s hodnotou 0. Všichni potomci mají hodnotu v zásobníku stejnou jako jejich rodič, dokud u některého není určena jiná hodnota. Pokud má některý prvek specifikovánu hodnotu `z-index`, zapíše ji jako svou hodnotu do zásobníku svého rodiče a navíc vytvoří nový, vlastní zásobník, v jehož rámci budou číslováni všichni jeho potomci. On sám se sem opět zapíše s hodnotou 0.

Popis vlastnosti `z-index`

- **povolené hodnoty:**
  - `celé číslo>|auto|inherit`
- **výchozí hodnota:** `auto`
- **lze použít:** pozicované prvky
- **dědí se:** ne
- **procenta:** nelze použít
- **media:** vizuální

Popis hodnot:

`<celé číslo>`
: číslo je hodnotou prvku v aktuálním zásobníku. Současně vytvoří nový zásobník, který bude aktuálním pro všechny potomky tohoto prvku; hodnotou prvku v tomto novém zásobníku je 0. Toto číslo může být i záporné.

`auto`
: úroveň v aktuálním zásobníku je stejná jako u rodičovského prvku. Prvek nevytváří nový zásobník

`inherit`
: zdědí hodnotu této vlastnosti z rodičovského prvku

K použití této vlastnosti si stačí pamatovat, že těm prvkům, které se budou překrývat, můžeme nastavit příslušnou hodnotu z-index - čím vyšší číslo, tím bude prvek (a jeho obsah) "výše" než ostatní. Všechny ostatní prvky, jimž hodnotu nenastavíme, se budou chovat, jako by měly nastavenu hodnotu 0. U prvků, které mají stejnou hodnotu, pak rozhoduje jejich pořadí v dokumentu - pozdější prvky překryjí ty dřívější.

3.10 Přehled všech vlastností CSS

Vzhledem k omezenému rozsahu práce, nelze se všem vlastnostem podrobně věnovat. Následující tabulka je shrnutím všech vlastností CSS2. Ke každé vlastnosti

jsou uvedeny její povolené hodnoty; výchozí hodnotu, kterou má vlastnost nastavenu, dokud jí není přiřazena hodnota jiná; množinu prvků, v nichž lze vlastnost použít; určení, zda hodnoty této vlastnosti zdědí potomci daného prvku; v případě, že lze použít procentní hodnotu, je zde uvedeno, k čemu se procentní poměr vztahuje; a v posledním sloupci je uvedena skupina médií, pro něž je daná vlastnost platná.

NÁZEV	MOŽNÉ HODNOTY	VÝCHOZÍ HODNOTA	KDE LZE POUŽÍT (PRÁZDNÉ = VŠECHNY PRVKY)	DĚDÍ SE? (PRÁZDNÉ = NEDĚDÍ SE)	PROCENTA (PRÁZDNÉ = NELZE POUŽÍT)	MÉDIA
azimuth	<úhel> | [[left-side | far-left | left | center-left | center | center-right | right | far-right | right-side] || behind] | leftwards | rightwards | inherit	center		ano		aurální
background	[background-color || background-image || background-repeat || background-attachment || background-position] | inherit				jen v background-position	vizuální
background-attachment	scroll | fixed | inherit	scroll				vizuální
background-color	<barva> | transparent | inherit	transparent				vizuální
background-image	<uri> | none | inherit	none				vizuální
background-position	[[<procenta> | <velikost>]{1,2} | [[top | center | bottom] || [left | center | right]]] | inherit	0% 0%	blokové a nahrazované prvky		relativní k rozměrům boxu	vizuální
background-repeat	repeat | repeat-x | repeat-y | no-repeat | inherit	repeat				vizuální
border	[border-width || border-style ||	viz dílčí vlastnosti				vizuální

	<barva>] | inherit					
border-collapse	collapse | separate | inherit	collapse	table a inline-table prvky	ano		vizuální
border-color	<barva>{1,4} | transparent | inherit	viz dílčí vlastnosti				vizuální
border-spacing	<velikost> <velikost> ? | inherit	0	table a inline-table prvky	ano		vizuální
border-style	<border-style>{1,4} | inherit	viz dílčí vlastnosti				vizuální
border-top border-right border-bottom border-left	[border-top-width || border-style || <barva>] | inherit	viz dílčí vlastnosti				vizuální
border-top-color border-right-color border-bottom-color border-left-color	<barva> | inherit	hodnota vlastnosti color				vizuální
border-top-style border-right-style border-bottom-style border-left-style	<border-style> | inherit	none				vizuální
border-top-width border-right-width border-bottom-width border-left-width	<border-width> | inherit	medium				vizuální
border-width	<border-width>{1,4} | inherit	viz dílčí vlastnosti				vizuální
bottom	<velikost> | <procenta> | auto | inherit	auto	pozicované prvky		vztahuje se k výšce omezujícího bloku	vizuální

Název	Možné hodnoty	Výchozí hodnota	Kde lze použít	Dědí se?	Procenta	Média
caption-side	top | bottom | left | right | inherit	top	table-caption prvky	ano		vizuální
clear	none | left | right | both | inherit	none	blokové prvky			vizuální
clip	<tvar> | auto | inherit	auto	blokové a nahrazované prvky			vizuální
color	<barva> | inherit	závisí na prohlížeči		ano		vizuální
content	[<řetězec> | <uri> | <počítadlo> | attr(X) | open-quote | close-quote | no-open-quote | no-close-quote]+ | inherit	prázdný řetězec	pseudo-prvky :before a :after			všechna
counter-increment	[<identifikátor> <celé-číslo>?]+ | none | inherit	none				všechna
counter-reset	[<identifikátor> <celé-číslo>?]+ | none | inherit	none				všechna
cue	[cue-before || cue-after] | inherit					aurální
cue-after	<uri> | none | inherit	none				aurální
cue-before	<uri> | none | inherit	none				aurální
cursor	[[<uri> ,]* [auto | crosshair | default | pointer | move | e-resize | ne-resize | nw-resize | n-resize | se-resize | sw-resize | s-resize | w-resize| text | wait | help]] | inherit	auto		ano		vizuální, interaktivní

Název	Možné hodnoty	Výchozí hodnota	Kde lze použít	Dědí se?	Procenta	Média
direction	ltr | rtl | inherit	ltr		ano		vizuální
display	inline | block | list-item | run-in | compact | marker | table | inline-table | table-row-group | table-header-group | table-footer-group | table-row | table-column-group | table-column | table-cell | table-caption | none | inherit	inline				všechna

Název	Možné hodnoty	Výchozí hodnota	Kde lze použít	Dědí se?	Procenta	Média
elevation	<úhel> | below | level | above | higher | lower | inherit	level		ano		aurální
empty-cells	show | hide | inherit	show	table-cell prvky	ano		vizuální

Název	Možné hodnoty	Výchozí hodnota	Kde lze použít	Dědí se?	Procenta	Média
float	left | right | none | inherit	none	všechny prvky kromě			vizuální

			pozicovaných a generovaného obsahu			
font	[[font-style || font-variant || font-weight]? font-size [/ line-height]? font-family] | caption | icon | menu | message-box | small-caption | status-bar | inherit	viz dílčí vlastnosti		ano	povoleno ve font-size a line-height	vizuální
font-family	[[<family-name> | <generic-family>],]* [<family-name> | <generic-family>] | inherit	závisí na prohlížeči		ano		vizuální
font-size	<absolutní-velikost> | <relativní-velikost> | <velikost> | <procenta> | inherit	medium		ano, dědí se počítaná hodnota	vzathuje se k velikosti písma rodičovského prvku	vizuální
font-size-adjust	<číslo> | none | inherit	none		ano		vizuální
font-stretch	normal | wider | narrower | ultra-condensed | extra-condensed | condensed | semi-condensed | semi-expanded | expanded | extra-expanded | ultra-expanded | inherit	normal		ano		vizuální
font-style	normal | italic | oblique | inherit	normal		ano		vizuální
font-variant	normal | small-caps | inherit	normal		ano		vizuální
font-weight	normal | bold | bolder | lighter | 100 | 200 | 300 | 400 | 500 | 600 | 700 | 800 | 900 | inherit	normal		ano		vizuální

Název	Možné hodnoty	Výchozí hodnota	Kde lze použít	Dědí se?	Procenta	Média
height	<velikost> | <procenta> | auto | inherit	auto	všechny prvky kromě nenahrazovaných inline prvků a sloupců a skupin sloupců v tabulkách			vizuální

Název	Možné hodnoty	Výchozí hodnota	Kde lze použít	Dědí se?	Procenta	Média

left	<velikost> | <procenta> | auto | inherit	auto	pozicované prvky		vztahuje se k šířce omezujícího bloku	vizuální
letter-spacing	normal | <velikost> | inherit	normal		ano		vizuální
line-height	normal | <číslo> | <velikost> | <procenta> | inherit	normal		ano	vztahuje se k velikosti písma prvku samotného	vizuální
list-style	[list-style-type || list-style-position || list-style-image] | inherit		prvky s display: list-item	ano		vizuální
list-style-image	<uri> | none | inherit	none	prvky s display: list-item	ano		vizuální
list-style-position	inside | outside | inherit	outside	prvky s display: list-item	ano		vizuální
list-style-type	disc | circle | square | decimal | decimal-leading-zero | lower-roman | upper-roman | lower-greek | lower-alpha | lower-latin | upper-alpha | upper-latin | hebrew | armenian | georgian | cjk-ideographic | hiragana | katakana | hiragana-iroha | katakana-iroha | none | inherit	disc	prvky s display: list-item	ano		vizuální

Název	Možné hodnoty	Výchozí hodnota	Kde lze použít	Dědí se?	Procenta	Média
margin	<margin-width>{1,4} | inherit				vztahuje se k rozměrům omezujícího bloku	vizuální
margin-top margin-right margin-bottom margin-left	<margin-width> | inherit	0			vztahuje se k rozměrům omezujícího bloku	vizuální
marker-offset	<velikost> | auto | inherit	auto	prvky s display: marker			vizuální
marks	[crop || cross] | none | inherit	none	kontext stránky			vizuální, stránková
max-height	<velikost> | <procenta> | none | inherit	none	všechny prvky kromě nenahrazovaných inline prvků a prvků tabulek		vztahuje se k výšce omezujícího bloku	vizuální
max-width	<velikost> | <procenta> | none | inherit	none	všechny prvky kromě nenahrazovaných		vztahuje se k šířce omezujícího	vizuální

			inline prvků a prvků tabulek		bloku	
min-height	<velikost> | <procenta> | inherit	0	všechny prvky kromě nenahrazovaných inline prvků a prvků tabulek		vztahuje se k výšce omezujícího bloku	vizuální
min-width	<velikost> | <procenta> | inherit	závisí na prohlížeči	všechny prvky kromě nenahrazovaných inline prvků a prvků tabulek		vztahuje se k šířce omezujícího bloku	vizuální

Název	Možné hodnoty	Výchozí hodnota	Kde lze použít	Dědí se?	Procenta	Média
orphans	<celé-číslo> | inherit	2	blokové prvky	ano		vizuální, stránková
outline	[outline-color || outline-style || outline-width] | inherit	viz dílčí vlastnosti				vizuální, interaktivní
outline-color	<barva> | invert | inherit	invert				vizuální, interaktivní
outline-style	<border-style> | inherit	none				vizuální, interaktivní
outline-width	<border-width> | inherit	medium				vizuální, interaktivní
overflow	visible | hidden | scroll | auto | inherit	visible	blokové a nahrazované prvky			vizuální

Název	Možné hodnoty	Výchozí hodnota	Kde lze použít	Dědí se?	Procenta	Média
padding	<padding-width>{1,4} | inherit	viz dílčí vlastnosti			relativní k šířce omezující ho bloku	vizuální
padding-top padding-right padding-bottom padding-left	<padding-width> | inherit	0			relativní k šířce omezující ho bloku	vizuální
page	<identifikátor> | auto	auto	blokové prvky	ano		vizuální, stránková
page-break-after	auto | always | avoid | left | right | inherit	auto	blokové prvky			vizuální, stránková
page-break-before	auto | always | avoid | left | right | inherit	auto	blokové prvky			vizuální, stránková
page-break-inside	avoid | auto | inherit	auto	blokové prvky	ano		vizuální, stránková
pause	[[<time> | <procenta>]{1,2}] | inherit	závisí na prohlížeči			viz popis pause-before a pause-after ve specifikaci CSS	aurální
pause-after	<time> |	závisí na			viz	aurální

	<procenta> | inherit	prohlížeči			specifikaci CSS	
pause-before	<time> | <procenta> | inherit	závisí na prohlížeči			viz specifikaci CSS	aurální
pitch	<frequency> | x-low | low | medium | high | x-high | inherit	medium		ano		aurální
pitch-range	<číslo> | inherit	50		ano		aurální
play-during	<uri> mix? repeat? | auto | none | inherit	auto				aurální
wt_cssserial_009.html#position	static | relative | absolute | fixed | inherit	static	všechny prvky kromě generovaného obsahu			vizuální

Název	Možné hodnoty	Výchozí hodnota	Kde lze použít	Dědí se?	Procenta	Média
quotes	[<řetězec> <řetězec>]+ | none | inherit	závisí na prohlížeči		ano		vizuální

Název	Možné hodnoty	Výchozí hodnota	Kde lze použít	Dědí se?	Procenta	Média
richness	<číslo> | inherit	50		ano		aurální
right	<velikost> | <procenta> | auto | inherit	auto	pozicované prvky		vztahuje se k šířce omezujícího bloku	vizuální

Název	Možné hodnoty	Výchozí hodnota	Kde lze použít	Dědí se?	Procenta	Média
size	<velikost>{1,2} | auto | portrait | landscape | inherit	auto	kontext stránky			vizuální, stránková
speak	normal | none | spell-out | inherit	normal		ano		aurální
speak-header	once | always | inherit	once	prvky s informací v záhlaví tabulky	ano		aurální
speak-numeral	digits | continuous | inherit	continuous		ano		aurální
speak-punctuation	code | none | inherit	none		ano		aurální
speech-rate	<číslo> | x-slow | slow | medium | fast | x-fast | faster | slower | inherit	medium		ano		aurální
stress	<číslo> | inherit	50		ano		aurální

Název	Možné hodnoty	Výchozí hodnota	Kde lze použít	Dědí se?	Procenta	Média
table-layout	auto | fixed | inherit	auto	table a inline-table prvky			vizuální
text-align	left | right | center | justify | <řetězec> | inherit	závisí na prohlížeči a směru	blokové prvky	ano		vizuální

		textu				
text-decoration	none | [underline || overline || line-through || blink] | inherit	none				vizuální
text-indent	<velikost> | <procenta> | inherit	0	blokové prvky	ano	vztahuje se k šířce omezujícího bloku	vizuální
text-shadow	none | [<barva> || <velikost> <velikost> <velikost>? ,]* [<barva> || <velikost> <velikost> <velikost>?] | inherit	none				vizuální
text-transform	capitalize | uppercase | lowercase | none | inherit	none		ano		vizuální
top	<velikost> | <procenta> | auto | inherit	auto	pozicované prvky		vztahuje se k výšce omezujícího bloku	vizuální

Název	Možné hodnoty	Výchozí hodnota	Kde lze použít	Dědí se?	Procenta	Média
unicode-bidi	normal | embed | bidi-override | inherit	normal	všechny prvky, ale viz specifikaci CSS			vizuální

Název	Možné hodnoty	Výchozí hodnota	Kde lze použít	Dědí se?	Procenta	Média
vertical-align	baseline | sub | super | top | text-top | middle | bottom | text-bottom | <procenta> | <velikost> | inherit	baseline	inline prvky a table-cell prvky		vztahuje se k line-height prvku samotného	vizuální
visibility	visible | hidden | collapse | inherit	inherit				vizuální
voice-family	[[<specific-voice> | <generic-voice>],]* [<specific-voice> | <generic-voice>] | inherit	závisí na prohlížeči		ano		aurální
volume	<číslo> | <procenta> | silent | x-soft | soft | medium | loud | x-loud | inherit	medium		ano	vztahuje se ke zděděné hodnotě	aurální

Název	Možné hodnoty	Výchozí hodnota	Kde lze použít	Dědí se?	Procenta	Média
white-space	normal | pre | nowrap | inherit	normal	blokové prvky	ano		vizuální
widows	<celé-číslo> | inherit	2	blokové prvky	ano		vizuální, stránková
width	<velikost> | <procenta> |	auto	všechny prvky kromě nenahrazovaných		vztahuje se k šířce	vizuální

	auto | inherit		inline prvků a řádky a skupiny řádků v tabulkách		omezujícího bloku	
word-spacing	normal | <velikost> | inherit	normal		ano		vizuální
z-index	auto | <celé-číslo> | inherit	auto	pozicované prvky			vizuální

4 Testové otázky vztahující se k výukovému materiálu „Principy počítačů“

- ***Charakterizujte jednotlivé generace počítačů.***
- ***Nakreslete a popište von Neumannovo blokové schéma počítače.***
- ***Jaké předchůdce počítačů znáte?***
- ***Přibližně od jakého roku se datují počítače dnešního typu?***
- ***Do kterého období a země by jste umístili první děrné štítky? (pomůcka: sčítání obyvatel)?***
- ***Jak fungují děrné pásky?***
- ***Jakým způsobem byly vkládány programy do sálových počítačů (např. ENIAC).***
- ***Proč bylo (a dodnes je) nutné chladit některé prvky počítačů?***
- ***Které konektory je možné nalézt na zadní straně centrální jednotky počítače?***
- ***K připojování kterých zařízení je vhodný 25-pinový sériový konektor umístěný na zadní straně centrální jednotky počítače?***
- ***Které konektory jsou umístěny uvnitř centrální jednotky počítače?***
- ***Lze propojit dva počítače s využitím kabelu zapojeného přes sériové nebo paralelní rozhraní?***
- ***Které základní funkce zajišťuje základní deska?***
- ***Jaké platí pravidlo pro připojování základní desky vodiči ke zdroji napětí?***
- ***K čemu slouží přepínače DIP nebo můstkové propojky (Jumpery)?***
- ***Které důležité prvky je možné nalézt na základní desce?***
- ***Která zařízení (běžně přídavná) je možné mít integrována přímo na základní desce? Má to nějakou výhodu či nevýhodu?***
- ***Podle jakých kritérií je možné dělit paměti?***
- ***Co je to paměť Cache?***
- ***Je možné použít pro paměť Cache dynamický přístup k paměti?***
- ***Vysvětlete pojmy LIFO a FIFO.***
- ***Popište adresovatelné paměti.***
- ***Jaké dnes rozeznáváme paměti podle technického provedení pouzdra?***
- ***Co je to paměť RAM?***
- ***Co je to paměť ROM?***
- ***Jaké jsou základní rozdíly mezi RAM a ROM?***
- ***Čím se vyznačují paměti SDRAM?***
- ***Kolik bitů obsahuje jeden byte?***
- ***Popište základní části operační paměti.***
- ***Jaké typy procesorů znáte?***

- ***Stručně popište architekturu RISC a napište, co tato zkratka znamená.***
- ***Co patří mezi základní části mikroprocesoru?***
- ***Podle kterých kritérií je možné rozlišovat mikroprocesory?***
- ***Jak získáme skutečnou frekvenci mikroprocesoru?***
- ***Jak zpracovává mikroprocesor instrukce?***
- ***Vysvětlete pojem sběrnice.***
- ***Jaké druhy sběrnic obsahuje mikroprocesor?***
- ***Vyjmenujte základní sběrnice na základní desce.***
- ***Charakterizujte sběrnici ISA.***
- ***Pro která zařízení je možné použít rozhraní USB?***
- ***Popište základní výhody sběrnice AGP.***
- ***Co je to BIOS?***
- ***Co je to POST?***
- ***Které čtyři základní funkce plní BIOS?***
- ***Popište kroky, které provádí po zapnutí počítače BIOS.***
- ***Co je to CMOS?***
- ***Co se provede jako první při zapnutí počítače.***
- ***Popište jednotlivé kroky, které provádí počítač, od zapnutí až do natažení operačního systému.***
- ***Co je to „studený“ start?***
- ***Co je to „teplý“ start?***
- ***Co je to FAT?***
- ***K čemu slouží FAT?***
- ***Jaké druhy FAT znáte?***
- ***Co je to pevný disk a k čemu slouží?***
- ***Co je to disketová mechanika a k čemu slouží?***
- ***Jaké je členění oblastí pevného disku?***
- ***Co je to sektor?***
- ***Co je to stopa?***
- ***Co je to cluster (cylindr)?***
- ***Jaký je princip čtení a záznamu dat na pevný disk?***
- ***Vysvětlete proces formátování disku.***
- ***Vysvětlete pojmy: vyhledávací doba, čtecí doba a přístupová doba. Mají spolu něco společného?***
- ***Jaký je způsob záznamu a čtení na magnetická média?***
- ***Vyjmenujte alespoň tři zařízení pracující s magnetickými disky.***
- ***Pro jaké účely se nejčastěji používají magnetické diskové paměti?***

- ***Jaká je zásadní nevýhoda magnetických páskových pamětí?***
- ***Vyjmenuj základní druhy optických pamětí?***
- ***Jaký je princip záznamu na CD-disk a pevný disk?***
- ***Popište metodu zápisu a čtení na magnetooptický disk.***
- ***Jaká je metoda zápisu dat na CD-R?***
- ***Jakým způsobem jsou zaznamenávána data na DVD disk?***
- ***K čemu se používá záložních zdrojů?***
- ***Jaké druhy baterií se používají u přenosných počítačů?***
- ***Jaké rozeznáváme základní druhy portů?***
- ***K čemu slouží sériový port?***
- ***K čemu slouží paralelní port,***
- ***Co je to Game Port?***
- ***Jaké jsou základní druhy paralelních portů?***
- ***Jakým způsobem přenáší data sériový a paralelní port?***
- ***Do jaké kategorie zařízení je možné zařadit grafické karty?***
- ***Jaké dva základní režimy rozeznáváme u grafických karet?***
- ***Které části obsahuje grafická karta?***
- ***K čemu slouží TV-tuner?***
- ***K čemu slouží na grafické kartě DAC převodník?***
- ***Jakou úlohu má na grafické kartě mikroprocesor?***
- ***K čemu slouží paměť na grafické kartě?***
- ***Co je to True Color režim?***
- ***Co je to rychlost grafické karty?***
- ***Vyjmenuj periferní zařízení počítače.***
- ***Na jaké základní části je možné rozdělit klávesnici?***
- ***Popište funkce kláves: Escape, Enter, Pause, PrintScreen***
- ***Popište funkce kláves: NumLock, ScrollLock, CapsLock***
- ***Popište funkce kláves: Ctrl, Alt, Shift***
- ***Popište funkce kláves: PageUp, PageDown, Home, End, Tab***
- ***K jakým účelům slouží klávesy BackSpace a Delete? Popište konkrétní funkce těchto kláves.***
- ***Jakými konektory se nejčastěji připojuje klávesnice k centrální jednotce počítače?***
- ***Co signalizují jednotlivé světelné diody umístěné na klávesnici?***
- ***Má klávesnice svůj mikroprocesor nebo jsou data zpracovávána v mikroprocesoru počítače?***
- ***Jaké principy se využívají při přenosu znaku z hmatníku až do mikroprocesoru?***

- ***Čím je charakterizován konkrétní znak předaný mikroprocesoru po stisku klávesy (kombinace kláves)?***
- ***Jaké rozeznáváme tři základní druhy myší?***
- ***Kde se používají zařízení zvaná TouchPad, TrackBall a TrackPoint a proč?***
- ***Jak probíhá přenos informace o pozici z pohybu mechanické myši až do počítače?***
- ***Které porty jsou pro připojení myši či jiného polohovacího zařízení vhodné?***
- ***Jaké rozlišujeme dva základní principy konstrukce tabletů?***
- ***V jakých oblastech se nejčastěji používají tablety a počítače s perem?***
- ***Jaké rozeznáváme dva základní druhy monitorů?***
- ***Nakreslete schéma zobrazení bodu na masce monitoru s katodovou trubicí.***
- ***Co je to RGB?***
- ***Co je to LCD?***
- ***Rozepište, co je to horizontální a vertikální frekvence monitoru.***
- ***Jak pracuje aditivní míchání barev?***
- ***Co to znamená, pokud je uvedeno, že monitor má 17“?***
- ***Jaké druhy scannerů znáte?***
- ***Jakými třemi způsoby lze sejmout barevný obraz?***
- ***Jakými rozhraními se připojují scannery k počítači?***
- ***Vysvětlete princip snímání scannerem.***
- ***Jaké druhy tiskáren znáte?***
- ***Popište základní principy bodové tiskárny.***
- ***Popište základní principy inkoustové tiskárny.***
- ***Popište základní principy laserové tiskárny.***
- ***K čemu slouží souřadnicové zapisovače?***
- ***Vysvětlete pojem DPI.***
- ***K jakým zařízením patří zvuková karta?***
- ***Jaké konektory můžeme nalézt na zvukové kartě?***
- ***Jaké známe topologie sítí?***
- ***Popište sběrnicovou topologii.***
- ***Popište stromovou topologii.***
- ***Čím se vyznačuje topologie ring?***
- ***Které součásti patří k technickému a programovému vybavení sítí?***

5 Závěr

Ke zrealizování interaktivního výukového materiálu bylo využito jednak klasických zdrojů – knih a časopisů, ale především velkého množství elektronických materiálů umístěných na Internetu. Interaktivní výukový materiál byl použit k výuce třináctiletých až patnáctiletých žáků. Jedná se pestřejší způsob prezentace probírané látky, zdá se i rychlejší. Zejména samostatné úkoly, které žáci prováděli vyhledáním zadaných jevů na internetu a sestavením vlastních prezentací na zadané téma jsou velmi účinné a motivující. V příloze je uvedena ukázka jedné z prací patnáctiletých žáků na téma monitory. Zkoušení vybraných jevů lze opět provést testem prostřednictvím počítače, kdy studenti vyplňují testové dotazníky výběrem z jednotlivých možností a odesílají své odpovědi na školní server.

V práci se nevěnuji popisu technického vybavení počítačů pro spouštění multimediálních prezentací, neboť počítače, které jsou dnes na školách, jsou již na takové úrovni, že bez problémů vystupují ve výuce v roli didaktické pomůcky. Do budoucna lze předpokládat další dynamický rozvoj v této oblasti nejen ve výuce „informatiky“, neboť k pouhým demonstračním účelům takovýchto prezentací úplně stačí jeden počítač umístěný ve třídě a připojený digitální projektor. Samozřejmě lepší realizace je prostřednictvím počítačové učebny, které jsou dnes na školách běžně. Při vytváření školních sítí navíc není problém vyvést do každé třídy (nebo aspoň některých tříd) připojení na Internet.

Je tedy možné říci, že využití počítače a elektronických materiálů ve výuce je výrazným **motivujícím činitelem**. Motivace je tím, co determinuje naše chování od samého začátku. Prostřednictvím těchto výukových postupů dáváme žákům dostatek podnětů, aby na podněty „méně pozitivní“ neměli tolik času. Dalším prvkem je **názornost,** jakožto nedílná součást procesu osvojování vědomostí a dovedností. Povzbuzuje zájem o učivo, podněcuje pozornost, usnadňuje zapamatování vědomostí a dovedností, napomáhá pochopení učiva. Důležitou podstatnou složkou je i způsob, jak se vlastní představy žáků tvoří. Klasický přístup je vytváří postupně a hledá spojovací můstky. Multimediální aplikace (a Internet) je předkládá v mnoha podobách a současně. **Zpětnou vazbu** v počítačem podporované výuce zajišťují výukové materiály jen pokud jsou zcela cílevědomě připraveny. Žák je při práci s výukovým materiálem prostřednictvím počítače nucen k **aktivitě** a **samostatnosti**. Počítač výrazně přispívá k **soustavnosti**, překonává formalismus i jednostranně pamětní typ učení. Učitel nemůže jen "probírat" učivo, ale musí s tímto médiem především pracovat. Základním rysem, který nelze opomenout, je skutečnost, že elektronické výukové materiály poskytují obrovské množství dat, která musí být tříděna a rozdělena do jednotlivých vyučovacích dávek.

Domnívat se však, že počítač z výuky vytlačí „tabuli s křídou resp. fixem“ by bylo nadsazené. Nicméně, se může počítač s vhodnými výukovými programy stát didaktickým prostředkem, který spolu s dalšími přispěje k úspěšnějšímu naplňování výukových cílů.

6 Použitá literatura a webové odkazy:

(1) Musciano, Ch., Kennedy, B., HTML a XHTML Kompletní průvodce. 1 vyd. Computer press. Praha, 2000. 633 s. ISBN 80-7226-407-9.

(2) HTML 4.01 specifikace[online]. http://www.w3.org/TR/1999/REC-html401-19991224

(3) XHTML 1.0 specifikace [online]. http://www.w3.org/TR/2002/REC-xhtml1-20020801

(4) Snížek, Martin. XHTML - element object [online]. http://interval.cz/clanek.asp?article=2164

(5) Snížek, Martin. XHTML - typy atributů [online]. http://interval.cz/clanek.asp?article=1455

(6) Snížek, Martin. XHTML - základní pojmy a prvky [online]. http://interval.cz/clanek.asp?article=1374

(7) Yuhů. Meta tagy [online]. http://www.jakpsatweb.cz/meta_tagy.htm

(8) Staníček Petr, Přehled všech vlastností CSS http://www.webtip.cz/art/wt_tech_html/wt_css_prehled.html

(9) ANTOŠ, M. Připojte se k internetu. 1. vyd. Praha : Computer Press, 1996. ISBN 80-85896-60-5.

(10) Dějiny počítačů. [online]. c2006. poslední revize 23. 8. 2006 [cit. 2006-11-17]. Dostupné z URL

(11) http://cs.wikipedia.org/wiki/D%C4%9Bjiny_po%C4%8D%C3%ADta%C4%8D%C5%AF

(12) TŮMA, J. Náš život s počítači. 1. vyd. Praha : Naše vojsko, 1990. ISBN 80-206-0089-2

(13) VIMMER, M. Internet a jeho služby. 1. vyd. Plzeň : ZČU Plzeň, 1996. ISBN 80-7082-276-7.

(14) ZELENÝ, J., MANNOVÁ, B. Historie výpočetní techniky. 1. vyd. Praha : Scientia, 2006. ISBN 80-86960-04-8.

(15) DOMES, M. Tvorba internetových stránek pomocí HTML, CSS a JavaScriptu. 1. vyd. Kralice na Hané : Computer Media, 2005. ISBN 80-86686-39-6.

(16) JANOVSKÝ, D. Jak psát web – návod na html stránky. [online]. c2006. poslední revize 14. 11. 2006 [cit. 2006-11-17]. Dostupné z URL

(17) VRBÍK, V. Tvorba WWW stránek v HTML. 1. vyd. Plzeň : Pedagogické centrum Plzeň, 2001. ISBN 80-7020-078-2.

(18) Jandová, Lada ing.: Počítačová výuka. Zásady tvorby výukových programů. ZU PF 1995

(19) Nelson, S. L.: Microsoft® FrontPage 98 Na první pokus. Computer Press 1998

(20) Wittmann, M, Bukovanský, St.: Co je to vlastně Internet? Blesk s.r.o. Ostrava 1998

(21) Hlavenka, J. a kol.: Vytváříme WWW stránky a spravujeme moderní Web Site. Computer Press 1997

(22) Minasi, M.: IBM PC – velký průvodce hardware. Grada 1991

(23) Berka, K., Jauris, M.: Logika. SPN Praha 1978

(24) Hrázský, J.: Mikropočítače a počítače – učebnice I. Informatorium 1996

(25) Hrázský, J.: Mikropočítače a počítače – učebnice II. Informatorium 1996

(26) d'Hardancourt, A.: Poznej svůj počítač. Grada 1995

(27) White, R.: Jak pracují počítače. Unis 1994

(28) Kolář, J., Müller, K., Plášil, Fr.: Úvod do počítačů a programování. SNTL Praha 1991

7 Přílohy

Ukázka snímků prezentace tvořená žáky na téma „monitory“