OBSAH

1. ÚVOD

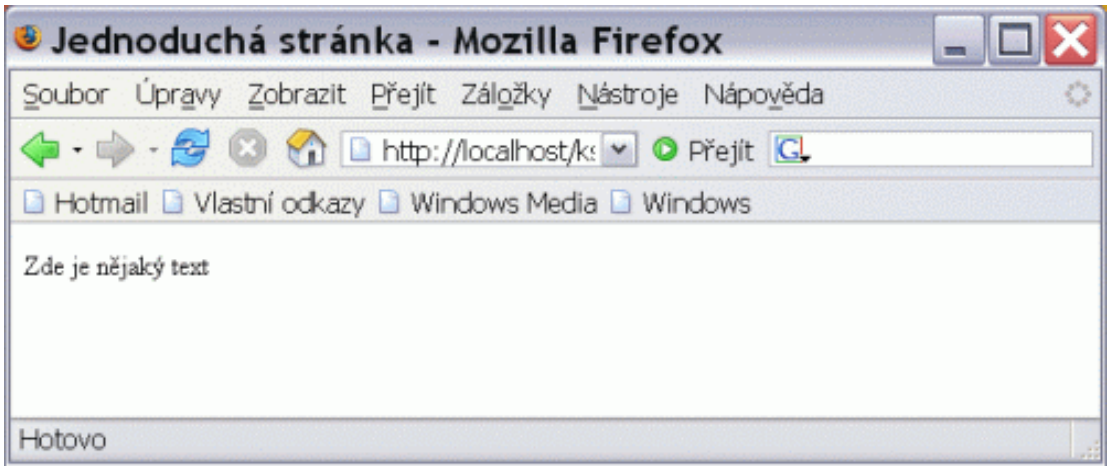
1.1 PRINCIP WEBOVÝCH STRÁNEK

WWW (z angl. world wide web) je služba, která umožňuje především způsob multimediální prezentace. Celý princip spočívá v tom, že někde na vzdáleném serveru je umístěná webová stránka, kterou si uživatel stáhne do svého počítače a webový prohlížeč mu jí zobrazí.

1.1.1 WWW STRÁNKA

Webová stránka, kterou si uživatel stáhne ze serveru ovšem nevypadá tak, jak se zobrazí v prohlížeči. Je to totiž jen pouhý text, neboli zdrojový kód stránky a přiložené soubory (obrázky, hudba...) a teprve prohlížeč ji interpretuje v podobě, kterou uživatel vidí.

Například kód této stránky:



vypadá následovně:

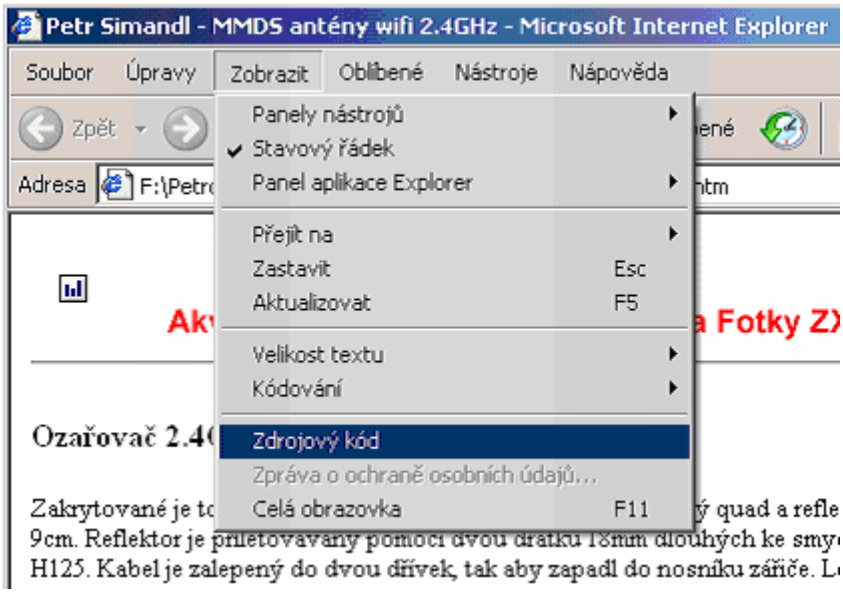
```
<?xml version="1.0" encoding="UTF-8"?>

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

<html xmlns="http://www.w3.org/1999/xhtml"
xml:lang="cs" lang="cs">

<head>
<meta http-equiv="content-type" content="text/html;
charset=UTF-8" />
<meta http-equiv="content-language" content="cs" />
<title>Jednoduchá stránka </title>
</head>

<body>
<p>Zde je nějaký text</p>
</body>

</html>
```

Poté, co takovýto kód dorazí do prohlížeče, si ho prohlížeč přeloží a pak teprve zobrazí kompletní webovou stránku.

Pokud je stránka uložena na pevném disku, funguje pochopitelně i offline.

Když chcete zjistit kód nějaké konkrétní stránky, máte dvě možnosti:

- buď můžete stránku otevřít v nějakém textovém editoru (např. ve Windows standardně Poznámkový blok)
- nebo (pro Internet Explorer) můžete otevřít stránku ve webovém prohlížeči a najít si nahoře na liště ***zobrazit > zdrojový kód*** - otevře se textový dokument s kódem stránky. Ve Firefoxu je to ještě jednodušší - stačí použít klávesovou zkratku *Ctrl+U*



Pokud chci nějakou webovou stránku vytvořit, stačí do textového editoru (Poznámkový blok, Writer...) napsat kód stránky a uložit soubor s příponou .html nebo .htm (používají se obě varianty). Webový prohlížeč pak bude tento text automaticky chápat jako webovou stránku. Další možností je využití nějakého XHTML editoru - programu určeného přímo pro tvorbu stránek.

Nově vytvořená stránka se samozřejmě může jmenovat jakkoli. Pokud ale stránky umístíte na server, tak se uživateli po zadání adresy zobrazí automaticky stránka s názvem index.html (některé servery hlásí chybu, pokud neexistuje). To platí v případě, že zadaná adresa nebude obsahovat i konkrétní stránku - jako například: www.něco.cz/odkazy.htm.

1.2 HISTORIE XHTML

Značkovací jazyk HTML (HyperText Markup Language) vznikl původně pouze pro potřeby vědců, jako jednoduchý vývěskový systém. Postupem času se ale ukázalo, že jeho možnosti jsou mnohem širší.

První veřejně vydanou verzí jazyka HTML bylo HTML 2.0 vydané IETF (internetovou standardizační organizací v roce 1995. Ovšem protože HTML vznikalo původně pro vědce, je tato specifikace dosti strohá, co se týče grafických prvků a designu.

Postupem času ale vznikla potřeba publikované texty nějak graficky zvýraznit. Proto byla uvolněna verze HTML 3.2. Tu již vydalo W3C (světové webové konsorcium).

Předposlední verzí je HTML 4.0. Zde se počítá s kompletním oddělením obsahu od formy, takže veškerý vzhled by se měl definovat pomocí CSS. Aby W3C usnadnilo přechod na normu HTML 4.0, rozdělilo ji na:

- HTML 4.0 Strict - striktně dodržuje všechna pravidla HTML 4.0.
- HTML 4.0 Transitional - přechodová norma. Využívá všech novinek HTML 4.0 ale navíc ještě podporuje zavrhnuté prvky z HTML 3.2.
- HTML 4.0 Frameset - je totožná s HTML 4.0 Transitional, jen navíc obsahuje ještě definici rámů.

Specifikace HTML 4.0 byla později revidována, byly opraveny některé chyby a stala se z ní specifikace HTML 4.01.

Další specifikace již nenese název HTML, ale XHTML (eXtensible HyperText Markup Language - rozšiřitelný hypertextový značkovací jazyk). První verze XHTML 1.0 je totožná s HTML 4.01, jen je rozšířena o obecná pravidla jazyka XHTML. Stejně jako HTML 4.01 se dělí na:

- XHTML 1.0 Strict
- XHTML 1.0 Transitional
- XHTML 1.0 Frameset

Myšlenkou XHTML je tvorba dokumentů pro různá výstupní zařízení. Tato rozličná zařízení ovšem nemohou podporovat všechny vlastnosti jazyka. Proto byly definovány různé podmnožiny. Ty bylo třeba definovat a standardizovat, proto vznikla specifikace Modularization of XHTML. Tato specifikace rozděluje všechny prvky XHTML 1.0 do modulů, ze kterých se následně skládají značkovací jazyky. Z těchto jazyků zatím W3C uznalo XHTML 1.1 a XHTML Basic (k použití hlavně na mobilních telefonech, PDA a podobně).

V této práci se budu zabývat konkrétně specifikací XHTML 1.0 Strict.

1.3 OBECNĚ O XHTML

Teď by asi bylo dobré, vyjasnit některé pojmy co se týče kódu.

Vezměme si nějaký příklad:

```
<p> První pokus </p>
```

Celý řádek uvedený v příkladu se nazývá element. Ten se skládá z otevírací části (otevíracího tagu), obsahu elementu a uzavírací části (uzavíracího tagu).

 je otevírací tag. Tím začíná nějaká konkrétní oblast (zde konkrétně odstavec). Tagy slouží pouze jako informace pro prohlížeč - na stránce se tedy nezobrazí.

"první pokus" je obsah elementu. Ten se na stránce zobrazí. Jeho formátování určují právě tagy, ve kterých je uzavřen.

 neboli uzavírací tag ukončuje element. Je vlastně totožný s tagem otevíracím, jen obsahuje navíc zpětné lomítko.

Pokud do sebe vnořujeme více elementů, není je možné křížit mezi sebou. Přitom nezáleží na tom, který element bude vnější a který vnitřní - viz. Příklad:

```
<div> <p> Obsah elementu </div> </p> - toto je špatně
<div> <p> Obsah elementu </p> </div> - toto je dobře
```

Zkrátka pokud se nějaký element nachází uvnitř jiného elementu, musí se tam nacházet celý, tedy jeho otevírací část, obsah i uzavírací část.

Zatímco v HTML na tom nezáleželo, v XHTML je třeba psát vše kromě obsahu elementu malými písmeny.

V XHTML existují také elementy, které nemají povolen žádný obsah. Například element pro vytvoření oddělovací čáry, přechod na novou řádku a podobně. Tyto elementy mohou obsahovat jen otevírací a uzavírací část. Tedy:

```
<br></br>
```

Tento zápis je sice správný, ale zbytečně dlouhý. Proto se užívá zkrácená verze:

```
<br/>
```

Ta je podle specifikace také správně, ale některé starší prohlížeče to neberou. Problém se vyřeší přidáním mezery před zpětné lomítko:

```
<br />
```

Tyto entity, nazývané nepárové tagy, v HTML neměly uzavírací část. Ta je ale XHTML povinná, proto se používá takovýto zápis.

Atributy (též parametry) určují další vlastnosti elementu. Nachází se v otevírací části elementu, oddělené od jména elementu a od dalších atributů mezerou. Atribut obsahuje jméno atributu, znak "=" a hodnotu atributu v uvozovkách nebo apostrofech.

Příklad:

```
<p class="zluta"> Obsah elementu </p>
<p class='zluta'> Obsah elementu </p>
<p class="zluta" id="první"> Obsah elementu </p>
```

Jeden element může mít více atributů. U nich nezáleží na pořadí.

1.4 XHTML EDITORY

XHTML editory jsou programy, pomocí nichž můžete vytvářet webové stránky. Obecně se dělí na dva druhy:

- wysiwyg editory
- strukturní editory

1.4.1 WYSIWYG EDITORY

Název je zkratkou anglického "What you see is what you get", tedy česky "co vidíš, to dostaneš".

Jsou určeny především pro uživatele, kteří se tvorbou webových prezentací nechtějí zabývat moc do hloubky. Autor tam pouze vloží text, který si přeje na stránky umístit a pomocí uživatelského rozhraní si nastaví jednoduchý vzhled stránky. Editor si veškerý XHTML kód vygeneruje sám a autor se nemusí o nic starat. Ovšem kvalita takto udělaných stránek nebývá nijak valná (to se týká optimalizace, validity a podobných aspektů - nikoliv obsahu).

Pokud se tedy chcete zabývat tvorbou stránek trochu víc, je používání *pouze* nějakého wysiwyg editoru nemyslitelné. Nemluvě o tom, že implementace nástrojů jako CSS, JavaScript nebo PHP bývá v těchto editorech většinou dost náročná. Ovšem na druhou stranu se tyto editory dají dobře využít například při obarvování částí textu.

1.4.2 STRUKTURNÍ EDITORY

Tak tady už je to trochu o něčem jiném. Především pokud chcete dělat stránky ve strukturním editoru, potřebujete umět alespoň XHTML. Ovšem pokud to zvládáte, lze pak využít úplně všechny možnosti, jaké jazyk XHTML nabízí a nemusíte se omezovat jen na to, co vám wysiwyg editor dovolí. Navíc spousta strukturních editorů si umí "domýšlet" tagy které píšete a samy vám je doplní. Takže stačí napsat jen začátek a on si doplní celý tag. V případě že je párový, dopíše někdy i jeho zakončující protějšek. Další šikovnou vlastností je obarvování kódu pro lepší přehlednost (to umí třeba i Firefox, když si necháte zobrazit Zdrojový kód).

2. XHTML

2.1 STRUKTURA XHTML DOKUMENTU

Nejjednodušší validní (a bez problémů fungující) XHTML dokument může vypadat třeba takto:

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

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

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="cs" lang="cs">

<head>
<meta http-equiv="content-type" content="text/html; charset=UTF-8" />
<meta http-equiv="content-language" content="cs" />
<title>MojeWebovky|příklad 1: XHTML stránka</title>
</head>

<body>
<p>Plnohodnotný XHTML dokument</p>
</body>

</html>
```

Jen pro úplnost, jak se stránka zobrazí v prohlížeči.

2.1.1 VYSVĚTLENÍ ČÁSTÍ KÓDU

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

XML deklarace. Řádek obsahuje informaci o verzi XHTML a kódování dokumentu. Pokud je dokument v kódování UTF-8 nebo UTF-16, může se tento řádek vynechat, ale vždy je lepší ho tam mít.

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

Doctype, neboli Definice Typu Dokumentu (Document Type Definition - DTD). Udává verzi použitého XHTML. O jednotlivých verzích jsem se zmiňoval v kapitole o historii XHTML. Zápisy doctype pro jednotlivé verze jsou:

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

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

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

DTD udává verzi použitého XHTML, aby prohlížeč věděl, jak má k dokumentu přistupovat.

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="cs" lang="cs"> </html>
```

Párový tag, musí povonně obsahovat entity `<head>` a `<body>`. V obou atributech se udává jazyk používaný v dokumentu. Správně je podle specifikace xml:lang . Atribut lang je zde pouze kvůli zpětné kompatibilitě se staršími prohlížeči.

```
<head> </head>
```

Je také párový a musí povinně obsahovat element `<title>`. Jinak tento element obsahuje metadata – tedy informace o dokumentu.

```
<meta http-equiv="content-type" content="text/html; charset=UTF-8" />
<meta http-equiv="content-language" content="cs" />
```

Jsou elementy obsahující metadata. O nich je zde celá kapitola.

```
<title> </title>
```

Tento element obsahuje titulek stránky a je povinný. Například:

```
<title> Název mojí stránky </title>
```

Zobrazí titulek stránky:


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

Značí samotné tělo dokumentu (tedy veškerý obsah, který se uživateli zobrazí – texty, obrázky…).

```
<p> </p>
```

Jen pro úplnost - element obsahující odstavec.

2.2 FORMÁTOVÁNÍ TEXTU

2.2.1 ZÁKLADNÍ FORMÁTOVÁNÍ

`<h1> </h1>`

znamená nadpis první úrovně. Obsah této entity je automaticky tučně a písmem o velikosti 24 pixelů. Nadpisy (angl. heading) usnadňují formátování dokumentu a orientaci v něm. Je jich celkem šest stupňů (tedy `<h1>` až `<h6>`). `<h1>` (nadpis první úrovně) je nejvýše a měl by se v textu vyskytovat pouze jednou. Příklad toho, jak by mohla vypadat struktura nadpisů v dokumentu:

```
<h1>automobily</h1>
        <h2>Škoda</h2>
                <h3>Superb</h3>
                <h3>Roomster</h3>
        <h2>Peguot</h2>
                <h3>206</h3>
                <h3>207</h3>
```

(tato ukázka slouží pouze k bližšímu pochopení principu nadpisů, v praxi by se na podobný výčet použily seznamy)

Samozřejmě, že to co umí nadpisy (po vizuální stránce) lze udělat i bez nich, ale je to zbytečně složitější. Nelze pak využít snadného formátování pomocí CSS a je to podle specifikace nesprávné.

Výchozí formátování nadpisů:

tag	stupeň písma	výchozí velikost	využití
<h1>	6	24px	Hlavní nadpis stránky
<h2>	5	18 -19px	Podnadpisy, názvy kapitol
<h3>	4	16px	Mezinadpisy
<h4>	3	13px	Důležitější odstavce
<h5>	2	10px	Žádné rozumné použití
<h6>	1	9px	Nejde ani pořádně přečíst

Uvedené hodnoty jsou pouze výchozí a dají se samozřejmě upravit pomocí CSS.

`<p> </p>`

je odstavec. Ten ovšem nevypadá jako klasický český odstavec, protože chybí odsazení prvního řádku a po ukončení odstavce následuje vertikální mezera. Text je tak vlastně rozdělen do bloků, což usnadňuje orientaci v dokumentu. Navíc dlouhý souvislý blok textu zpravidla spoustu potenciálních čtenářů odradí.

`<br />`

Tato entita bez povoleného obsahu značí zalomení řádku neboli enter (`<br />` z angl. brake). Když totiž zalomíte řádek v samotném kódu, tak se to na stránce nijak neprojeví (ale na druhou stranu to není špatné pro vaší orientaci v kódu). Na rozdíl od odstavce `<p>`, entita

 jen zalomí řádek a pokračuje hned na dalším. Neudělá vertikální mezeru jako odstavec.

```
<p>
Možnosti:
Jedna
Dvě
Tři
</p>
```

Se na stránce vůbec neprojeví.

Prohlížeč ignoruje všechny entery, a taky více mezer za sebou chápe pořád jen jako jednu. Pro enter se tedy použije tag <br> a pro mezeru (pokud zrovna potřebujete víc mezer za sebou) existuje speciální znak (znaková entita)   (pozor na ten středník, ten tam musí být).

2.2.2 BEZ FORMÁTOVÁNÍ

<pre> </pre>

Element, který zachovává původní formátování zdroje. Neplatí zde tedy pravidlo o ignorování konců řádků a nahrazování více mezer jedinou. Tato vlastnost lze nastavit i v kaskádových stylech.

2.2.3 ELEMENTY URČUJÍCÍ KONKRÉTNÍ VÝZNAM TEXTU

Následuje souhrn elementů, které definují nějakou specifickou část textu. Rozdělil jsem je do skupin, podle toho jak se většinou zobrazují (pokud to není upraveno pomocí CSS).

2.2.3.1 Kurzíva

Vyznačuje zvýraznění (emphasis).

<dfn> </dfn>

Obsahem tohoto elementu je pojem nebo definice (definition).

<var> </var>

Označuje proměnnou.

<cite> </cite>

Tento element označuje citovaný zdroj, odkaz na další zdroje nebo citaci. Poměrně se vžil zvyk uzavírat do něj jména osob, organizací apod.

2.2.3.2 Tučně

Má být ještě důraznější zvýraznění než . Obsah je tučně.

2.2.3.3 Písmo s pevnou šířkou znaků

(pokud nevíte co si pod tím představit, podívejte se na ukázky zdrojového kódu kdekoliv na těchto stránkách)

<code> </code>

Používá se k označení počítačového nebo programového kódu.

<samp> </samp>

Vyznačuje vzorový výstup programů, skriptů apod.

<kbd> </kbd>

Indikuje text, který má být zadán uživatelem.

2.2.3.4 Nijak nezvýrazněný text

<abbr> </abbr>

Označuje zkratku (abbreviation), jejíž plné znění by se alespoň při prvním výskytu v dokumentu mělo nacházet v jeho atributu title. Například: <abbr title="Střední průmyslová škola elektrotechnická">SPŠE</abbr>

<acronym> </acronym>

Tento element se používá k uzavírání zkratkových slov. Ty se narozdíl od zkratek vyslovují většinou jako jedno slovo, a ne po jednotlivých písmenech. Pravidla pro použití atributu <title> jsou zde stejná jako u elementu <abbr>. Příklad: <acronym title="Česká dopravní kancelář">Čedok</acronym>

2.2.4 CITACE

<q> </q>

Značí citaci (quote). Je to ale pouze řádkový element, takže nemůže obsahovat blokové prvky. Je tedy výhodné ho používat pro kratší citace uvnitř textu. Obsah elementu je automaticky v uvozovkách.

<blockquote> </blockquote>

Obsahuje citaci stejně jako <q>. Na rozdíl od něj je ale blokový, nepřidává citovanému textu automaticky uvozovky a má předdefinované odsazení zleva i zprava.

2.2.5 ELEMENTY PRO ZÁPIS VZORCŮ

Nejčastější využití <sub> a <sup> je při zápisu matematických a chemických vzorců. <sub> totiž vytvoří dolní index ($_{\text{subskript}}$), <sup> ($^{\text{superskript}}$) dělá horní index.

2.2.6 ZMĚNY V DOKUMENTU

Nakonec uvedu ještě dva elementy sloužící ke sledování změn v dokumentu. Výhody jejich užití mi ale přijdou dost rozporuplné.

`<ins> </ins>`

Element značí text, který byl do dokumentu později přidán (insert) nebo aktualizován. Běžně se zobrazí podtržený.

`<del> </del>`

Obsahuje staré a již neplatné informace, které v dokumentu vlastně už nemají co dělat a měly by být správně odstraněny (delete). Text je přeškrtnutý. Tento element se využívá hlavně ve spojení s `<ins>`.

2.3 HYPERTEXTOVÉ ODKAZY

2.3.1 RELATIVNÍ A ABSOLUTNÍ

Neodmyslitelnou součástí webových stránek jsou pochopitelně hypertextové odkazy. Ty uživatele buď odkazují z jedné vaší stránky na druhou (v tom případě se jedná o odkaz s relativní adresou) nebo je mohou poslat na nějaké úplně jiné stránky (pak se jedná o odkaz s absolutní adresou).

Příklad odkazu s absolutní adresou:

```
<a href= "http://www.cokoliv.cz"> Odkaz na cokoliv </a>
```

Odkaz značí element `<a>` (anchor = kotva). Obsahuje atribut `href`, jehož hodnota udává cíl odkazu. Obsah elementu se zobrazí takto: Odkaz na cokoliv

Chceme-li, aby odkaz směřoval na jinou naší stránku (která je ovšem součástí stejného webového dokumentu), vypadá zápis takto:

```
<a href="moje-stranka.html"> Moje stránka </a>
```

Kde cíl, tedy moje-stranka.html musí být ve stejném adresáři, jako stránka, která na ní odkazuje.

2.3.2 CESTA

Pokud tomu tak není, je třeba zadat cestu k požadované stránce:

```
<a href="cesta/nazev_stranky.html"> text odkazu </a>
```

Takže to pak může vypadat třeba takto:

```
<a href="./priklady/priklad1.html"> Příklad číslo 1 </a>
```

`./` značí stejný adresář, v jakém je stránka s odkazem. `../` znamená adresář nadřazený (rodičovský). Je dobré uvádět cestu i v případě, že je požadovaná stránka ve stejném adresáři:

```
<a href="./priklad1.html"> Příklad číslo 1 </a>
```

2.3.3 ZÁLOŽKY

Hypertextové odkazy mohou také odkazovat na místa v aktuální stránce, neboli záložky. Záložka může být jakýkoliv element s nějakým obsahem, který má nastaven atribut `id`. Dříve se ale namísto id používal atribut `name` a proto je kvůli podpoře starších prohlížečů dobré, uvádět ještě oba:

```
<h2 id="rukavice" name="rukavice">Rukavice na lyže</h2>
```

Odkaz na nějakou záložku pak vypadá takto:

```
<a href="#rukavice"> Odkaz na rukavice </a>
```

Je-li pak požadovaná záložka na jiné stránce, než odkaz na ní, lze se to zapsat dohromady:

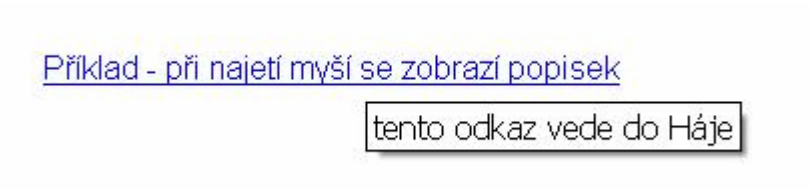
```
<a href= "../lyže/vybaveni.html#rukavice"> Odkaz na rukavice </a>
```

2.3.4 POPISKY

Na vytvoření bubliny s texetm, která se zobrazí při najetí myši na prvek, slouží atribut `title`. Ten lze použít i u jiných elementů:

```
<a href="./stranka2.html" title="tento odkaz vede do Háje"> Jiná stránka
</a>
```

Vypadá to takto:



2.4 OBRÁZKY

V dnešní době ale už webové stránky nejsou jen o textu a odkazech. Podstatnou součástí stránek se stávají různé obrázky, ať již fotografie nebo nějaká úvodní animace. Jsou prakticky nepostradatelné. Aby ale vaše stránky nezabírali 20 MB a uživatelé na načtení jednoho úvodníku nečekali půl hodiny, je třeba umisťovat obrázky na stránky s rozvahou – je to totiž to největší co se tam vyskytuje a prakticky to jediné, co zásadně zpomaluje načítání. Aby byly obrázky co nejmenší, používají se ve formátu *.jpg pro fotografie a ve *.gif pro ostatní obrázky a animace. V poslední době je nejpopulárnější třetí formát: *.png. Narozíl od *.gif v něm ale nejdou ukládat animace. Lze používat i jiné, ale tyto tři formáty jsou jistým standardem. Rozhodně by nebylo dobré třeba přes celé pozadí dát obrázek ve formátu *.bmp. Snad každý uživatel bez extra rychlého připojení by po chvilce načítání usoudil, že vaše stránky za tu dobu, jakou se budou načítat, určitě nestojí.

2.4.1 POVINNÉ ATRIBUTY

Obrázek dostanu na stránku pomocí nepárového elementu `<img>`:

```
<img src= "../obrazky/fotka.jpg" alt="Fotka z mé dovolené u moře" />
```

`src` (z angl. source) udává zdroj – tedy místo, kde se obrázek nachází. Zápis hodnoty je totožný jako u atributu `href` u odkazů.

Povinný atribut `alt` znamená alternativní text. Jeho obsah se uživateli vypíše, pokud interpret XHTML nemůže zobrazit požadovaný obrázek. To může být způsobeno chybou serveru, prohlížeče (např. vypnuté obrázky), ale i autora webu.

Aby jste se ve struktuře svého webu lépe vyznali, je dobré si udělat složku obrázky, kde je budete mít všechny pohromadě. Nebudou se vám tak plést mezi jednotlivými stránkami.

2.4.2 OBRÁZEK JAKO ODKAZ

Samozřejmě, že z obrázku lze udělat odkaz a to velice jednoduše:

```
<a href="./dalsi-kapitola.html">
<img src="obrazky/odkaz-next.gif" alt="Odkaz na další kapitolu" />
</a>
```

Ovšem v tomto případě bude obrázek modře ohraničen (aby bylo jasné, že se jedná o odkaz).

Toho se dalo v HTML zbavit pomocí atributu border. Ten však striktní XHTML nepodporuje, a proto je nutné použít kaskádové styly.

2.4.3 POPISKY

I u elementu `<img />` lze využít atribut `title`. Funguje stejně jako při použití v odkazech:

```
<img src="../obrazky/na_horach.jpg" title="To jsem já v Himalájích" />
```

Pokud tedy na takovýto obrázek najedete kurzorem, zobrazí se popisek To jsem já v Himalájích

2.4.4 ROZMĚRY

U každého obrázku jdou také předdefinovat rozměry. Dělá se to pomocí atributů `width` (šířka) a `height` (výška). Pokud nezadáte jiné jednotky, jsou obě hodnoty v pixelech. Další možností je zadat procenta.

```
<img src="../obrazky/na_horach.jpg" title="To jsem já v Himalájích"
width="150" height="50%" />
```

V případě, že udáte jinou výšku a šířku než obrázek ve skutečnosti má, automaticky se přizpůsobí vámi zadaným hodnotám. To se ale v praxi nepoužívá, protože nejlepší je mít obrázek uložen již s potřebnými rozměry. Pokud by jste přesto tuto vlastnost chtěli využít, obsahují ji i CSS.

Zadávání rozměrů obrázku pomocí atributů `width` a `height` má přesto svůj význam. Pokud totiž uvedete skutečné rozměry obrázku, interpret XHTML si pro ně automaticky vyhradí místo a pokračuje ve stahování stránky. Když se totiž stránka načítá a jsou známé rozměry obrázků, text potom neskáče. Pokud však obrázek nemá zadané rozměry, stává se, že prohlížeč nejprve načte text a až později když načte obrázek. Text pak poskočí dolů a udělá obrázku místo. To je zejména pro uživatele s pomalejším připojením dost nepraktické – začte se do ještě ne kompletní stránky bez obrázků a najednou mu text doslova uteče před očima.

2.5 TABULKY

Důležitým prvkem webových stránek jsou pochopitelně tabulky. Jsou výborné pro zpřehlednění dat a dříve se používaly i na vytvoření layoutu stránky. Toto řešení se ale silně nedoporučuje, protože to odporuje významu tabulek ve specifikaci a navíc má takovýto layout spoustu nedostatků. Mnohem lépe se dá vytvořit pomocí elementu `<div>` a CSS.

2.5.1 JEDNODUCHÁ TABULKA

Tedy příklad jednoduché tabulky:

Levá horní	Pravá horní
Levá spodní	Pravá spodní

Zápis vypadá takto:

```
<table border="1">
<tr> <td> Levá horní </td> <td> Pravá horní </td> </tr>
<tr> <td> Levá spodní </td> <td> Pravá spodní </td> </tr>
</table>
```

vysvětlení kódu:

`<table> </table>`

Párový element, který v sobě uzavírá celou tabulku.

`<tr> </tr>`

Element značící řádek tabulky (z angl. table row).

`<td> </td>`

Vymezuje buňku tabulky. Sem už se píší rovnou data tabulky (angl. table data).

Atribut `border` určuje šířku ohraničení tabulky (v pixelech).

2.5.2 SLOŽITĚJŠÍ TABULKY

To byla úplně nejjednodušší tabulka. Mnohem častěji se ale požívá členění, kde je vyznačena hlavička, obsah a patička tabulky:

```
<table border="1">
<caption>Ceník</caption>

<thead>
 <tr>
  <th>Název</th>
  <th>Cena</th>
 </tr>
</thead>

<tbody>
 <tr>
  <td>Lednice</td>
  <td>10000,-</td>
 </tr>
 <tr>
  <td>PC</td>
  <td>20000,-</td>
 </tr>
</tbody>

</table>
```

Vypadá takto:

Ceník	
Název	**Cena**
Lednice	10000.-
PC	20000

`<caption> </caption>`

Hlavička tabulky – obsah tohoto párového elementu se vypíše nad střed tabulky. Pomocí CSS je možné umístit `<caption>` i pod tabulku.

`<thead> </thead>`

Hlavičková část tabulky. Musí být uvedena první, ještě před `<tfoot>` a `<tbody>`

`<th> </th>`

Hlavičkové pole (angl. table head) - používá se přímo místo `<td>` a text který obsahuje by měl být tučně a zarovnaný na střed. Hlavní využití ale spočívá až v kombinaci s CSS.

`<tfoot> </tfoot>`

Je patička tabulky, v příkladu jí ale uvedenou nemám. V kódu musí být uvedena ještě před `<tbody>` - tedy obvykle po elementu `<thead>` (pokud je přítomen).

`<tbody> </tbody>`

Tělo tabulky. Obsahuje konkrétní data. Elementů `<tbody>` může být v jedné tabulce více, pokud logicky oddělují dva různé druhy dat.

2.5.3 SLUČOVÁNÍ BUNĚK

Jednotlivé buňky se dají slučovat pomocí atributů `colspan` a `rowspan`

`colspan` slučuje sousední buňky pod sebou

`rowspan` slučuje sousední buňky vedle sebe

Zápis řádku se sloučenými buňkami:

Buňka	Buňka
Dvě buňky v jednom	
Buňka	Buňka
Buňka	Buňka

```
<tr> <td colspan="2"> Dvě buňky v jednom </td> </tr>
```

Hodnota vždy určuje kolik buněk bude sloučeno.

2.6 SEZNAMY A VÝČTY

2.6.1 ROZDÍL

Nejprve bych upřesnil, co je (podle XHTML) seznam a co výčet.

Seznam:

- Jana
- Petr
- František

Výčet (nebo také seznam definic):

Jana
: jedna moje kamarádka

Petr
: s ním jsem byl včera večer na hokeji

František
: můj dlouholetý kamarád
: bezvadně hraje na kytaru

2.6.2 SEZNAMY

Seznamy dál dělíme na uspořádané a neuspořádané. Uspořádané seznamy mají každou svoji položku nějak označenou (číslem, písmenem...). Celý seznam je v elementu `<ol>` a jednotlivé jeho položky v `<li>`. Neuspořádané seznamy mají před každou položkou nějaký, stále stejný, symbol (puntík, odrážku...). Jsou ohraničené elementem `<ul>` a každá položka je opět v `<li>`.

Uspořádaný seznam

1. první položka
2. druhá položka
3. třetí položka

použitý kód:

```
<ol>
    <li>první položka</li>
    <li>druhá položka</li>
    <li>třetí položka</li>
</ol>
```

a neuspořádaný seznam

- první položka
- druhá položka
- třetí položka

použitý kód:

```
<ul>
    <li>první položka</li>
    <li>druhá položka</li>
    <li>třetí položka</li>
</ul>
```

Použitý druh odrážek a číslování lze nadefinovat v kaskádových stylech.

2.6.3 VÝČTY

Výčet je soubor nějakých definic a jejich popisů. Celý výčet je uzavřen do elementu `<dl>`. V něm jsou pak jednotlivé položky v elementu `<dt>`, který je in-line a neměl by proro být zbytečně dlouhý. Komentář definice je v elementu `<dd>`. Ten už je blokový aby mohlo být vysvětlení pojmu dostatečně obsáhlé. Jedna definice může mít i několik různých vysvětlení.

Seznam definic (výčet):

první pojem
: vysvětlení prvního pojmu

druhý pojem
: vysvětlení druhého pojmu

třetí pojem
: vysvětlení třetího pojmu
: další vysvětlení třetího pojmu

použitý kód:

```
<dl>
   <dt>první pojem</dt>
     <dd>vysvětlení prvního pojmu</dd>
   <dt>druhý pojem</dt>
     <dd>vysvětlení druhého pojmu</dd>
   <dt>třetí pojem</dt>
     <dd>vysvětlení třetího pojmu</dd>
     <dd>další vysvětlení třetího pojmu</dd>
</dl>
```

2.6.4 MOŽNOSTI KOMBINACE

Seznamy a výčty lze navíc vzájemně kombinovat:

```
<ol>
 <li>Úvod</li>
 <li>Stať</li>
  <dt>Historie XHTML </dt>
   <dd>vývoj až do XHTML</dd>
  <dt>Obecně o XHTML </dt>
   <dd>popis fungování jazyka XHTML </dd>
 <li>Závěr</li>
</ol>
```

1. Úvod
2. Stať

   Historie XHTML
   vývoj až do XHTML
   Obecně o XHTML
   popis fungování jazyka XHTML

3. Závěr

Stejným způsobem je možné libovolně vnořit uspořádaný seznam do neuspořádaného a podobně.

2.7 FORMULÁŘE

2.7.1 FUNKCE FORMULÁŘŮ

Jako formulář můžeme označit vše, pomocí čeho je možné odesílat na server informace. Skládá se tedy z různých vstupních polí, zaškrtávacích políček, vysouvacích nabídek a tlačítek. Těm se souhrnně říká ovládací prvky. Pomocí formulářů je možné získat od uživatele data a s nimi dál pracovat (anketa, registrace na fóru, změna hesla...). Data získaná z formulářů pak zpracovává nějaký klientský nebo serverový skript.

Celý formulář (všechny ovládací prvky) je ohraničen elementem `<form>`. Ten musí povinně obsahovat atribut `action`, jehož hodnotou je nějaká webová stránka. Na ní se zadaná data odešlou a ona je zpracuje. V praxi se dost často formulář odkazuje na tu samou stránku ve které je obsažen.

Dalším často používaným atributem je `method`, který udává, jak budou data odeslána. V případě `method="get"` jsou data odesílána spolu s adresou stránky. Ta pak může vypadat třeba následovně: http://www.adresa.cz/index.htm?jmeno=Petr za otazníkem jsou informace které uživatel vyplnil a odeslal. Pokud se informací odesílá více, oddělí se ampersandem ("&").

Druhou variantou je `method="post"`. V tom případě se data odesílají v požadavku HTTP hlavičky a nejsou tedy vidět v adrese požadované stránky. Toho se využívá zejména při odesílání hesla nebo delších textů.

Element `<form>` může obsahovat jakékoliv prvky kromě dalšího `<form>`. Nesmí se tedy vnořovat jeden do druhého. Obsahem také nemůže být rovnou text - musí se vnořit například do odstavce (`<p>`).

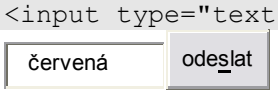
2.7.2 FORMULÁŘE PRO ZADÁVÁNÍ TEXTU

Nejčastěji používaný je element `<input />`. Jeho atribut `type` pak určuje, o jaký konkrétní ovládací prvek se bude jednat. Dále musí jako každý jiný ovládací prvek obsahovat atribut `name`, který je jedinečný v celém elementu `<form>`. Na stránce se však může vyskytovat jiný formulář obsahující elementy se stejným atributem `name`.

`type="text"`

Jedná se o výchozí hodnotu (pokud není nastaven atribut `type`). Zobrazí vstupní pole určené pro zadávání textu. Pokud obsahuje atribut `value`, bude jeho obsah zobrazen jako výchozí hodnota.

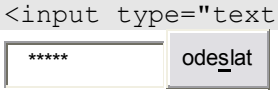
```
<input type="text" name="barva" value="červená" />
```



`type="password"`

To samé jako `type="text"`, ale zadávaný text se nezobrazí. Místo něj je každý znak nahrazen kolečkem nebo hvězdičkou (záleží na prohlížeči).

```
<input type="text" name="heslo" />
```



`type="checkbox"`

Vytvoří zaškrtávací políčko. Těch může být označeno hned několik nebo nemusí být vybráno žádné (narozdíl od `type="radio"`). Již předem zatržené políčko je možné vytvořit pomocí atributu `checked="checked"`. Nezaškrtnutá políčka se vůbec neodesílají.

```
<p><input type="checkbox" name="zluta" value="ano" />Žlutá</p>
<p><input type="checkbox" name="zelena" value="ano" checked="checked"
/>Zelená</p>
```

`type="radio"`

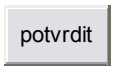
Je něco podobného jako `type="checkbox"`. V tomto případě je ale možné vybrat jen jednu možnost. Všechny elementy `<input />` zde proto mají stejný atribut `name`, jelikož se odešle pouze hodnota u zaškrtnuté možnosti.

```
<p><input type="radio" name="barva" value="zelena" /> Zelená</p>
<p><input type="radio" name="barva" value="cerna" checked="checked" />
Černá</p>
```

`type="submit"`

Vytvoří odesílací tlačítko, které odešle celý formulář. Hodnota atributu `value` se zobrazí jako nápis na tlačítku.

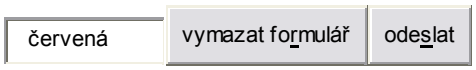
```
<input type="submit" value"potvrdit" />
```



`type="reset"`

Tlačítko, které nastaví všechny hodnoty ve formuláři na výchozí.

```
<input type="reset" value="vymazat formulář" />
```

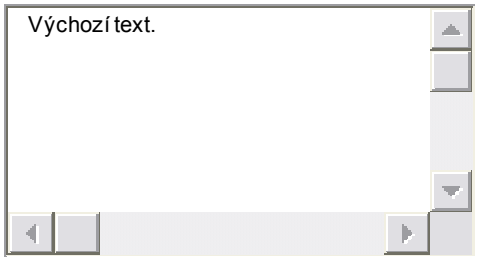


`type="hidden"`

Skrytý ovládací prvek, o kterém uživatel vůbec neví. Používá se většinou pro odesílání některých informací nutných pro správný chod skriptu, který bude formulář zpracovávat. Jako atributy připadají v úvahu pouze `name` a `value`.

Druhým používaným formulářovým elementem pro zadání textu je `<textarea>`. Ten slouží pro vkládání větších bloků textu. Je párový a jeho obsah se zobrazí jako výchozí hodnota. Povinně se u něho musí uvádět atribut `rows`, udávající počet viditelných řádek (pokud je jich víc, lze skrolovat) a atribut `cols`, který udává počet sloupců. Jeden sloupec je zde definován jako šířka průměrného znaku.

```
<textarea rows="7" cols="40" name="text">
Výchozí text.
</textarea>
```



2.7.3 VYSUNOVACÍ NABÍDKY VE FORMULÁŘÍCH

Pokud chci dát uživateli na výběr z předdefinovaných možností, použiji nejčastěji nějakou vysunovací nabídku.

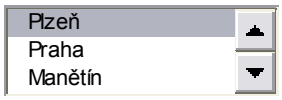
Zdrojový kód:

```
<select name="mesto">
 <option value="plz" selected="selected">Plzeň</option>
 <option value="pra">Praha</option>
 <option value="man">Manětín</option>
 <option value="spo">Spálené Poříčí</option>
</select>
```

Jak je z ukázky patrné, celý vysunovací seznam reprezentuje entita `<select>`, která má udaný atribut `name`. V ní jsou pak jednotlivé položky ohraničené elementem `<option>`. Jejich hodnotu udává atribut `value`.

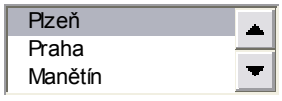
Položka, která obsahuje atribut `selected`, je automaticky vybrána při načtení stránky (stejně jako atribut `value` u elementu `<input />`). `selected` může nabývat pouze jediné hodnoty: `selected="selected"`.

Je-li potřeba aby bylo najednou vidět více položek, použije se atribut `size`. Jeho hodnota udává, kolik jich bude vidět.




```
<select name="město" size="3">
 <option value="plz" selected="selected">Plzeň</option>
 <option value="pra">Praha</option>
 <option value="man">Manětín</option>
 <option value="spo">Spálené Poříčí</option>
 <option value="mos">Most</option>
 <option value="par">Pardubice</option>
 <option value="ckr">Český Krumlov</option>
</select>
```

Pro možnost označení a následného odeslání více položek v adresáři slouží atribut `multiple`. Může nabývat jedinou hodnotu, a to `multiple="multiple"`.




```
<select name="město" size="3" multiple="multiple">
```

Pro označení více položek je třeba podržet klávesu Ctrl (To platí v Internet Exploreru, Firefoxu a Opeře - u ostatních prohlížečů to odzkoušené nemám).

Máte-li ve vysunovací nabídce více podobných možností, je dobré je rozdělit do tématických skupin. To lze udělat pomocí elementu `<optgroup>`. Ten musí povinně obsahovat atribut `label`, jehož hodnota je nadpisem skupiny prvků. Straší prohlížeče, které `<optgroup>` nepodporují, zobrazí nabídku jako obyčejnou - nečleněnou. Nadpisů skupin (atributu `label`) si nevšímají.


```
<select name="auto">
 <optgroup label="Škoda">
  <option value="oct" selected="selected">Octavia</option>
  <option value="fab">Fabia</option>
  <option value="sup">Superb</option>
 </optgroup>

 <optgroup label="Peguot">
  <option value="206">206</option>
  <option value="306">306</option>
 </optgroup>
</select>
```

2.7.4 LOGICKÉ ČLENĚNÍ FORMULÁŘŮ

Aby byly formuláře přehlednější jak pro interprety XHTML, tak pro uživatele, existují další elementy určené pro jejich členění.

`<label> </label>`

Přiřazuje popisky k jednotlivým ovládacím prvkům.


```
<p><label>Vaše jméno: <input name="jmeno" type="text" /></label></p>
```

`<fieldset> </fieldset>`

Orámuje vybranou skupinu ovládacích prvků.

`<legend> </legend>`

Obsahuje nadpis (název) skupiny ovládacích prvků. Uvádí se v elementu `<label>` a to hned na prvním místě. Před ním mohou být v `<label>` pouze bílá místa.

Souhrnný příklad:

```
<fieldset>
<legend>Osobní údaje</legend>

<p><label>Vaše jméno: <input name="jmeno" type="text" /></label></p>
<p><label>Bydliště: <input name="bydliste" type="text" /></label></p>
<p><input type="submit" value="odeslat" /></p>
</fieldset>
```

Osobní údaje

Vaše jméno:

Bydliště:

odeslat

2.8 ELEMENT <META />

Je element, který obsahuje informace o dokumentu. Vyskytují se dva jeho druhy, jež definují použité atributy. První, který slouží k nastavení HTTP hlavičky obsahuje atribut `http-equiv`. Jeho hodnota udává, co konkrétně bude nastavovat. Druhý typ obsahuje namísto toho atribut `name` a jeho hodnotou je vlastnost, které se element týká. Dále element `<meta />` obsahuje atribut `content`, určující hodnotu vlastnosti kterou ovlivňuje.

2.8.1 NEJDŮLEŽITĚJŠÍ A NEJPOUŽÍVANĚJŠÍ <META /> ELEMENTY

```
<meta http-equiv="content-type" content="text/html; charset=kódování" />
```

Určuje kódování dokumentu. Pro češtinu je možné použít "UTF-8" (univerzální kódování), "ISO-8859-2" (čeština) nebo "windows-1250" (čeština ve Windows). V dnešní době si většina interpretů XHTML poradí se všemi uvedenými možnostmi kódování. Tento element je povinnou součástí každého XHTML dokumentu.

```
<meta http-equiv="content-language" content="cs" />
```

Udává jazyk, ve kterém je daný dokument. Podle tohoto meta elementu se řídí například vyhledávače. Je povinnou součástí každého XHTML dokumentu.

```
<meta name="description" content="popis" />
```

Definuje popis dokumentu. Ten se někdy zobrazí ve výsledcích vyhledávání, pod titulkem stránky. Měl by obsahovat co nejvíc podstatných informací o obsahu dokumentu.

```
<meta name="keywords" content="klíčová slova" />
```

Obsahuje klíčová slova, týkající se dokumentu, pro usnadnění vyhledávání. Například stránky cestovní kanceláře mohou obsahovat takovouto entitu:

```
<meta name="keywords" content="cestovní, kancelář, zahraničí, Itálie,
Řecko, moře, dovolená, zájezd, levně" />
```

```
<meta name="author" content="autor dokumentu" />
```

Jméno autora. Za ním se ještě většinou uvádí e-mailová adresa na autora. Je-li autorů více, použije se více těchto elementů a u každého se uvede, čeho je autor.

```
<meta name="author" content="code: Petr Kuda; pkuda@mail.cz" />
<meta name="author" content="style: Josef Pepa; pepajosef@mail.cz" />
```

```
<meta name="copyright" content="copyright dokumentu" />
```

Copyright dokumentu. Udává se jméno a kontakt na osobu, která má autorská páva a dále rok vytvoření.

```
<meta name="copyright" content="Petr Kuda (pkuda@mail.cz) 2006" />
```

2.9 OSTATNÍ

2.9.1 ZNAKOVÉ ENTITY

Jelikož někdy je nutné v textu uvádět i znaky, které jsou jinak vyhrazeny pro zápis XHTML (< >), dají se zapsat speciálními znakovými entitami. Ty pak prohlížeč interpretuje jako požadované znaky.

zápis	znak	překlad
&amp;	&	ampersand - a
&lt;	<	less than - menší než
&gt;	>	greather than - větší než
&quot;	"	quote - uvozovky
		něco jako: nonbrake space - pevná mezera

Všechny znaky se také dají zapsat pomocí kódu v ASCII tabulce. Snazší je ale pamatovat si tyto zkratky.

` ` se také využívá, pokud potřebujete více mezer za sebou. Většinou se bez toho ale díky CSS obejdete.

2.9.2 KONTEJNERY

`<div>` a `<span>`

jsou kontejnery. Tyto entity mohou obsahovat libovolné další. Nemají žádný konkrétní význam a nijak neformátují svůj obsah. Používají se na formátování textu (pomocí CSS), který nevyhovuje ani jedné z předepsaných definic nebo na tvorbu layoutu stránek. Kontejner `<div>` je blokový - může tedy obsahovat téměř všechny ostatní entity, jen ne rovnou text. `<span>` je řádkový (inline) a nesmí tedy obsahovat blokové prvky.

2.9.3 KOMENTÁŘE

`<!-- text komentáře -->`

Do této speciální entity se dají psát poznámky (komentáře) k vlastnímu kódu. Entita nemá povolený žádný obsah, ani nemá ukončovací část. Musí bezpodmínečně začínat `<!--` a to bez mezer, následuje libovolný text a nakonec je opět série znaků `-->` bez mezer. Cokoliv co je v komentáři napsáno, se na stránce nezobrazí a je to vidět pouze v kódu. U obsáhlejších stránek je používání komentářů prakticky nezbytné a mnohdy se vyplatí je používat i u menších projektů. Zpřehledňuje orientaci v kódu dokumnetu.

Příklad využití:

```
<div> <!-- tady začíná menu -->
…
</div> <!-- konec menu -->
<div> <!-- začátek obsahu -->
```

3.CSS

3.1 ÚVOD DO CSS

3.1.1 VÝVOJ

V době, kdy se začal rozvíjel web a vznikaly první verze značkovacího jazyka HTML, se nějak nepočítalo s tím, že by webové stránky někdy mohly vypadat pěkně. V představách původních tvůrců HTML se totiž web jevil jako velké množství stroze vyhlížejících stránek plných informací, přičemž vzhled takových stránek nebyl vůbec důležitý. Postupem času se však internet rozšiřoval i mimo akademickou komunitu. Ukázalo se, že je potřeba možnosti webových stránek rozšířit a přidat do nich prostředky pro tvorbu vzhledu.

CSS, neboli kaskádové styly (angl. Cascading Style Sheets), je doplněk k XHTML, který umožňuje formátování obsahu XHTML dokumentu. Navíc umožňuje předdefinovat automatický vzhled jakéhokoliv elementu, umístit cokoliv kamkoliv na stránku nebo třeba mít styl všech stránek někde v externím souboru. V případě změny designu stačí přenastavit jen tento soubor a nedělat to zbytečně na každé stránce zvlášť.

3.1.4 ZÁPIS CSS

Možnosti implementace CSS do stránek jsou celkem tři:

- přímý zápis stylu u formátovaného prvku
- zápis stylu do hlavičky dokumentu
- externí soubor *.css

3.1.4.1 Přímý zápis

```
<h1 style="color: blue;"> Tento nadpis bude modrý </h1>
```

Jak jste si jistě všimli, je zde drobný rozdíl v zápisu oproti XHTML. Za CSS vlastností se nedělá *rovnítko* ("=") ale *dvojtečka* (":"). Mezi jednotlivými atributy se nenechává pouze mezera, ale píše se tam i *středník* (";").

Přímý zápis stylu platí jen pro jeden konkrétní formátovaný prvek.

3.1.4.2 Zápis do hlavičky dokumentu

```
<head>
<title> Název stránky </title>
...

<style type="text/css">
h1 {
color: blue;
}
</style>
</head>
```

Styl se zapíše přímo do hlavičky dokumentu a bude podle něho vypadat celá stránka. V uvedeném případě budou veškeré nadpisy první úrovně modré.

3.1.4.3 Využití externího souboru

Ovšem v případě, že se rozhodnete změnit celkový vzhled stránek, museli by jste v každé stránce změnit styl. Pokud máte jen nějaké osobní stránky tak to ještě není takový problém, ale v případě rozsáhlejší webové prezentace by to byla práce na dlouhé zimní večery. Tomu se dá předejít využitím externího CSS souboru. Princip spočívá v tom, že provedete zápis stylu stejně jako při zápisu do hlavičky dokumentu, ale celý soubor uložíte pod názvem *.css a v každé stránce se na něj odkážete. Změna souboru se pak automaticky projeví u všech stránek využívajících tento styl.

Externí soubor např. styl.css pak může vypadat takto:

```
h1 {
color: blue;
}
```

Pak už stačí jen uvést do hlavičky dokumentu řádek, který na externí stylopis odkazuje:

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

Při tomto zápisu budou modré nadpisy první úrovně ve všech dokumentech, které využívají soubor styl.css.

3.1.5 NADŘAZENOST

U CSS platí také určitá hierarchie. Když mám například v hlavičce dokumentu nadefinováno že všechny odstavce budou červené:

```
<style type="text/css">
p {
color: red;
}
</style>
```

a přesto chci mít jeden odstavec modrý, udělám to pomocí přímého zápisu stylu:

```
<p style="color: blue"> Modrý odstavec </p>
```

Platí zde pravidlo, že přímý zápis má v případě konfliktu nastavených hodnot přednost před zápisem do hlavičky stránky a ten je zas nad externím stylopisem. Přímý zápis má tedy největší prioritu.

3.1.6 HROMADNÁ DEKLARACE

CSS může také využívat hromadnou deklaraci:

```
h1, h2, h3 {
color: green;
}
```

To znamená, že nadpisy první, druhé a třetí úrovně budou zelené.

Pochopitelně lze také k jednomu elementu přiřadit více hodnot

```
h1 {
color: green;
font-weight: bold;
}
```

Nadpisy první úrovně budou zelené a tučně.

3.1.7 TŘÍDY A IDENTIFIKÁTORY

Další možností CSS je nadefinovat si vlastní třídy:

```
.zelena {
color: green;
}
```

".zelena" je třída, kterou jsem si sám nadefinoval v hlavičce stránky nebo v externím stylopisu. Název třídy musí vždy začínat tečkou.

```
<p class= "zelena"> Tento odstavec je zelený </p>
```

Nepovinný atribut class (angl. třída) určuje podle jaké třídy bude element formátován. Šikovné je, že lze třídy kombinovat:

```
<p class= "zelena vpravo"> Tento odstavec je zelený a vpravo </p>
```

V případě, že by obě třídy obsahovaly stejný element ale s různými hodnotami, bude platit ta která je dříve nadeklarovaná.

Něco podobného jako třídy jsou identifikátory. Fungují prakticky stejně jako třídy, jen s tím rozdílem, že v jednom dokumentu se může každý identifikátor objevit jen jednou (využívají se zejména ve skriptech).

Deklarace identifikátorů se odlišuje tím, že před název se místo tečky píše *křížek* (#):

```
#hlavni {
color: green;
}
```

K elementu se třída přiřadí pomocí atributu `id`:

```
<h1 id="hlavni"> Nadpis <h1>
```

3.1.8 PSEUDOTŘÍDY

Ještě zde existují pseudotřídy (jen u elementu <a>). Pokud ve stylopisu zapíši vlastnost pro element <a>, bude ho to ovlivňovat kdykoliv beze změny. Pseudotřídy umožňují rozlišit různé "stavy" odkazu a nastavit mu v různých situacích různé vlastnosti. Jednotlivé pseudotřídy elementu <a> jsou:

Pseudotřída	**Vysvětlení**
a:active	právě vybraný odkaz
a:hover	odkaz na který nejedete myší
a:link	dosud nenavštívený odkaz
a:visited	navštívený odkaz

```
a:hover {
color: red
}
```

Při najetí myší bude odkaz červený.

3.1.9 HROMADNÁ DEKLARACE

Další výhodou CSS je dědičnost. To znamená, že pokud do kontejneru , který má nastavenou barvu textu na červenou, vložím odstavec <p>, bude i jeho písmo červené. Odstavec tuto vlastnost zdědí od svého rodičovského prvku.

3.2 VLASTNOSTI PÍSMA

3.2.1 FONT-FAMILY

Určuje druh písma. Hodnota je písmo, kterým chceme aby se text zobrazil. To se zapisuje jménem (Arial, Times New Roman). Problém nastane pokud uživatel námi definované písmo nemá nainstalované u sebe v počítači. Proto je dobré uvést písem více – pokud počítač písmo zná, použije ho, pokud ne, zkusí použít další v pořadí. Jako výchozí je nastaveno Times New Roman.

```
body {
font-family: "Times New Roman", Garamond, serif ;
}
```

V tomto případě vypíše počítač text písmem Times New Roman , jestliže ho nemá nainstalované tak písmem Garamond a pokud nemá ani to, použije jakékoli jiné patkové (serif) písmo.

Do uvozovek se dávají jen hodnoty, které obsahují mezeru.

Přehled kategorií písem:

druh písma	**popis**	**příklad písma**
sans-serif	bezpatkové písmo	Arial, Helvetica, Verdana...
serif	patkové písmo	Times New Roman, Garamond...
monospace	psací stroj	Courier, Monaco...
fantasy	ozdobné písmo	Stencil, Western...
cursive	ručně psané písmo	Mistral, Zapf Chancery...

3.2.2 FONT-STYLE

```
p {
font-style: italic;
}
```

Určuje styl písma

hodnota	**popis**	**ukázka**
normal	normální písmo	ukázka
italic	kurzíva (totéž jako `<em>`)	*ukázka*
oblique	uměle skloněné písmo (taky kurzíva)	*ukázka*

3.2.3 FONT-VARIANT

```
p {
font-variant: small-caps;
}
```

Napíše text kapitálkami. Má pouze 2 hodnoty: `normal` – normální písmo a `small-caps` – kapitálky: UKÁZKA KAPITÁLEK

3.2.4 FONT-WEIGHT

```
p {
font-weight: bold;
}
```

Definuje tučnost písma. Možností je spousta ale v praxi stejně stačí jen hodnoty `normal` a `bold`.

Přehled	
hodnoty	**tučnost písma**
normal	písmo nebude tučné
bold	písmo bude tučné
100	co nejslabší, nejméně tučné písmo
400	normální písmo
700	písmo bude tučné jako je bold
900	písmo bude nejtučnější, co to jde
násobky 100 od 100 do 900	různé stupně tučnosti
bolder	o něco tučnější, než by bylo, kdyby se to nezadalo
lighter	totéž co bolder , ale naopak

3.2.5 FONT-SIZE

```
p {
font-size: xx-small;
}
```

Nastaví velikost písma. Zde jsou tři typy hodnot.

Asi nejjednodušší jsou hodnoty `smaller` a `larger`. Ty značí, že dané písmo bude o stupeň větší (`larger`) nebo menší (`smaller`) než písmo ostatní (běžně 16 pixelů).

Další možností je použít klíčové slovo:

klíčové slovo	odpovídá stupni písma	velikost v pixelech
xx-small	1	9 px
x-small	2	10 px
small	3	13 px
medium	4	16 px
large	5	18 px
x-large	6	24 px
xx-large	7	30 px
bez zadání	4	16 px

Poslední variantou je zadat velikost písma přesně v nějakých jednotkách.

3.2.6 LINE-HEIGHT

```
p {
line-height: 200%;
}
```

Nastaví řádkování – vzdálenost mezi jednotlivými řádky textu (odborně proklad):

hodnoty	**výška řádku**
normal	výška řádku bude normální - tak vysoká, jak je nejvyšší prvek na řádku
délka	výška řádku nastavená napevno, nejčastěji v pixelech
procento	procento normální výšky, např. line-height: 150%
číslo	násobek normální výšky, např. line-height: 1.5

3.2.7 FONT

Tato vlastnost je shrnutím všech předchozích. Obsahuje tedy `font-style`, `font-variant`, `font-weight`, `font-size`, `line-height` a `font family`. Jednotlivé údaje se oddělují mezerou (jen před `line-height` je nutné napsat zpětné lomítko "/")a musí se zadávat přesně v tomto pořadí. Pokud nějakou hodnotu změnit nechci, může se vynechat. Pouze `line-height` nefunguje pokud není nastaveno `font-size`.

Příklad zápisu:

```
p {
font: italic small-caps bold 12pt/150% "Arial ce", Heveltica, sans-serif ;
}
```

V tomto případě bude text psán kurzívou, kapitálkami, tučně, bude mít velikost 12 bodů, řádkování 150% a prohlížeč použije písmo Arial, popřípadě Heveltica a pokud ani jedno z nich nemá uživatel nainstalováno, tak nějaké jiné bezpatkové.

3.3 BARVY A POZADÍ

3.3.1 COLOR

```
p {
color: #669933;
}
```

Pomocí této CSS vlastnosti lze obarvit téměř cokoliv. Možnosti zápisu barev mají samostatnou kapitolu, takže to pouze shrnu:

způsob zápisu	**příklad (obarvení načerveno)**
název barvy	color: red;
#rrggbb	color: ff0000;
#rgb	color: f00;
rgb(r,g,b)	color: rgb(255,0,);
rgb(r%,g%,b%)	color: rgb(100%,0%,0%);

3.3.2 BACKGROUND-COLOR

```
p {
background-color: green;
}
```

Nastaví barvu pozadí libovolného prvku. Je možné použít buď hodnotu `transparent` pro průhledné pozadí nebo jakýkoliv z výše uvedených zápisů barvy.

3.3.3 BACKGROUND-IMAGE

Umožňuje umístit na pozadí prvku obrázek, který se bude neustále opakovat. Je vhodné zároveň s ním uvádět i barvu pozadí, pro případ, že by se obrázek nenačetl.

Možné hodnoty:

hodnota	**popis**
none	výchozí nastavení – pozadí bez obrázku
url("cesta/obrazek.gif")	pozadí s obrázkem "obrazek.gif"

Příklad zápisu:

```
body {
background-color: black;
background-image: url(./obrazky/logo.png);
}
```

3.3.4 BACKGROUND-REPEAT

```
p {
background-repeat: repeat-x;
}
```

Nastaví možnosti opakování obrázku na pozadí:

hodnoty	**způsob opakování pozadí**
repeat	pozadí se opakuje, až vyplní celý prostor prvku (výchozí hodnota)
no-repeat	pozadí se neopakuje, vykreslí se pouze jednou
repeat-x	pozadí se opakuje v ose x, tj. horizontálně; na výšku se vykreslí jednou
repeat-y	pozadí se opakuje v ose y, tj. vertikálně; na šířku se vykreslí jen jednou

3.3.5 BACKGROUND-POSITION

```
p {
background-position: 50px 50px;
}
```

Umístí obrázkové pozadí kamkoliv v daném prvku. Je dobré zároveň zakázat opakování pozadí, protože jinak docílíte pouze posunutí neustále se opakujícího obrázku.

Možnosti zápisu:

- "procento" nebo "procento procento"

  Pokud se uvede pouze jedna hodnota, bude použita pro horizontální směr.
  Ve vertikálním se automaticky nastaví 50%. V případě uvedení obou hodnot určuje první horizontální směr a druhá směr vertikální.

- "vzdálenost" nebo "vzdálenost vzdálenost"

  Funguje to stejně jako procenta, jen se zadává vzdálenost v pixelech. Pokud je zadána pouze jedna hodnota, vertikální směr se automaticky nastaví na 50%. Vzdálenosti v pixelech a procentech jdou kombinovat.

- poslední možností je použití klíčových slov:

slovo	**překlad**	**druh zarovnání**
top	nahoru	vertikální
center	na střed	vertikální
bottom	dolů	vertikální
left	doleva	horizontální
center	na střed	horizontální
right	doprava	horizontální

Opět lze vertikální a horizontální zarovnání kombinovat (`top left`, `center bottom`...). Nezáleží na tom, zda uvedete jako první zarovnání vertikální nebo horizontální.

3.3.6 BACKGROUND-ATTACHMENT

```
p {
background-attachment: fixed;
}
```

Umožní ukotvit pozadí. To znamená, že při scrollování se pozadí neposouvá, ale na rozdíl od obsahu prvku zůstane na místě. Tato vlastnost může nabývat pouze dvou hodnot:

`scroll` – výchozí hodnota, pozadí se posouvá
`fixed` – pozadí se napevno ukotví a neposouvá se

3.3.7 BACKGROUND

Umožňuje zkrácený zápis všech vlastností týkajících se pozadí, a to v tomto pořadí: `background-color`, `background-image`, `background-repeat`, `background-position` a `background-attachment`. Stejně jako u vlastnosti font se jednotlivé hodnoty oddělují mezerou a libovolnou hodnotu je možno vynechat.

Příklad:

```
p {
background: red url("obrazky/hory.gif") no-repeat top center fixed;
}
```

V příkladu bude pozadí červené a přes něj obrázek hory.gif, který se nebude opakovat a bude napevno uchycen nahoře, uprostřed odstavce.

3.4 DALŠÍ ÚPRAVA TEXTU

3.4.1 WORD-SPACING

```
p {
word-spacing: 10px;
}
```

Tato vlastnost nastaví velikost mezer mezi slovy. Zadat se může buďto `normal` (výchozí hodnota) nebo vzdálenost v jakýchkoliv jednotkách podporovaných CSS. Tato vzdálenost může být i záporná.

3.4.2 LETTER-SPACING

```
p {
letter-spacing: 10px;
}
```

Jedná se vlastně o to samé jako u předchozí vlastnosti `word-spacing`. Jen místo mezer mezi slovy nastavuje mezery mezi písmeny. Možné hodnoty jsou totožné jako u `word-spacing` - tedy `normal` a nebo vzdálenost .

3.4.3 TEXT-DECORATION

```
p {
text-decoration: underline;
}
```

Jak se dá podle názvu snadno odvodit, nastavuje tato vlastnost dekoraci textu. Možné hodnoty jsou tyto:

hodnoty	**přikrášlení textu**	**ukázka**
none	žádné	formátovaný text
underline	podtržení	formátovaný text
overline	nadtržení	formátovaný text
line-through	přeškrtnutí	~~formátovaný text~~
blink	blikání	nemá smysl v tištěném dokumentu

Hodnota `none` je zde vhodná pro odstranění podtržení u odkazů.

3.4.4 VERTICAL-ALIGN

```
p {
vertical-align: sub;
}
```

Určuje vertikální (svislé) zarovnání:

hodnoty	popis
baseline	prvek leží na účaří řádku (výchozí nastavení)
sub	dolní index (nezmenšený)
super	horní index (nezmenšený)
top	co nejvýše – podle nejvyššího elementu na řádce
text-top	horní okraj prvku lícuje s horním okrajem řádku
middle	střed prvku na střed řádku
bottom	co nejníže podle nejnižšího elementu na řádce
text-bottom	spodek prvku lícuje se spodkem řádku
procento	procento výšky prvku lícuje s procentem výšky řádku

Vypadá to dost složitě, ale hodnoty `sub`, `super` a případně ještě procenta v praxi bohatě postačí.

Ukázkový text text text text _{sub} text text ^{super} text text text ^{100%} text text text.

3.3.5 TEXT-TRANSFORM

```
p {
text-transform: capitalize;
}
```

Jedná se o převod textu na velká nebo malá písmena.

hodnoty	převedení velikosti písmen
none	nic se nepřevádí (výchozí hodnota)
capitalize	první písmena všech slov budou velká
uppercase	všechna písmena se převedou na velká
lowercase	všechna písmena se převedou na malá

3.3.6 TEXT-ALIGN

Vlastnost určená k zarovnání textu.

```
p {
text-align: center;
}
```

hodnoty	zarovnání bloku textu
left	zarovnání doleva (výchozí hodnota)
center	zarovnání na střed
right	zarovnání napravo
justify	zarovnání do bloku

`justify` (zarovnání do bloku) je vhodné používat jen v širších sloupcích textu, protože prohlížeče nepodporují dělení slov a proto pak mezi nimi vznikají nehezké velké mezery.

3.3.7 TEXT-IDENT

```
p {
text-ident: 1em;
}
```

Definuje odsazení prvního řádku textu o určitou vzdálenost. Dají se zde použít dva typy hodnot: velikost (v jednotkách podporovaných CSS pro absolutní odsazení prvního řádku) nebo procenta (pro relativní odsazení prvního řádku).

Pomocí této vlastnosti a vlastnosti margin se dá vytvořit "český" odstavec.

3.5 VLASTNOSTI ZOBRAZENÍ

3.5.1 BORDER-COLOR

```
p {
border-bottom-color: #CC0033;
}
```

Nastaví barvu rámečku. Je zde možno použít tři hodnoty:

hodnoty	**barva rámečku**
barva	nastavená barva
transparent	průhledný rámeček
inherit	zděděná barva rámečku

`border-color` nastaví barvu celého rámečku. Pokud je potřeba obarvit pouze nějakou jeho část, použijí se další varianty této vlastnosti:

varianta	**popis**
border-left-color	obarvení levého okraje rámečku
border-right-color	obarvení pravého okraje rámečku
border-top-color	obarvení horního okraje rámečku
border-bottom-color	obarvení dolního okraje rámečku

3.5.2 BORDER-WIDTH

```
p {
border-top-width: 3px;
}
```

Určuje šířku rámečku:

hodnoty	**šířka rámečku**
pixely nebo jiný rozměr kromě procent	šířka rámečku v pixelech (nebo jiných jednotkách)
0px	rámeček tam nebude
thin	slabý rámeček
medium	středně silný rámeček (výchozí nastavení)
thick	tlustý rámeček

Tato vlastnost ovlivní celý rámeček. Pokud je potřeba změnit jen jeho část, použijí se opět další varianty vlastnosti:

varianta	**popis**
border-left-width	ohraničení levého okraje rámečku
border-right- width	ohraničení pravého okraje rámečku
border-top- width	ohraničení horního okraje rámečku
border-bottom- width	ohraničení dolního okraje rámečku

3.5.3 BORDER-STYLE

```
.popis {
border-style: dotted;
}
```

Vlastnost definuje styl rámečku:

hodnoty	druh rámečku	příklad
solid	plný	border-style: solid
dotted	tečkovaný	border-style: dotted
dashed	čárkovaný	border-style: dashed
double	dvojitý	border-style: double
groove	příkop	border-style: groove
ridge	val	border-style: ridge
inset	dolík	border-style: inset
outset	návrší	border-style: outset
none	žádný	border-style: none

I zde pro formátování části rámečku existují další varianty této vlastnosti:

varianta	popis
border-left-style	styl ohraničení levého okraje rámečku
border-right-style	styl ohraničení pravého okraje rámečku
border-top-style	styl ohraničení horního okraje rámečku
border-bottom-style	styl ohraničení dolního okraje rámečku

3.5.4 MARGIN

```
.popis {
margin-left: 10px;
}
```

Tato vlastnost definuje vnější okraj prvku.

hodnoty	vnější okraj
délka	vzdálenost mezi případným rámečkem a okolním dokumentem
procento	tatáž vzdálenost vypočítaná z rozměrů omezujícího nadřazeného prvku
auto	automatické nastavení okrajů tak, aby protilehlé byly stejné (nefunguje v IE v quirk módu)

Pokud je potřeba nastavit `margin` jen u části prvku, použijí se opět modifikace:

varianta	popis
margin-left	velikost levého okraje rámečku
margin-right	velikost pravého okraje rámečku
margin-top	velikost horního okraje rámečku
margin-bottom	velikost dolního okraje rámečku

3.5.5 PADDING

```
.popis {
padding: 1em;
}
```

Je to prakticky totéž jako margin, jen namísto vnějšího určuje okraj vnitřní. Jeho hodnota je tedy velikost mezery mezi rámečkem (`border`) a obsahem prvku. Možnosti hodnot a zápisu jsou totožné jako u vlastnosti `margin`.

3.5.6 HEIGHT

```
.popis {
height: 400px;
}
```

Určuje výšku prvku.

hodnoty	výška prvku
délka	zadaná výška prvku
procento	procento z výšky nadřazeného prvku
auto	výška je přirozená

3.5.7 WIDTH

Nastaví šířku prvku. Možnosti zápisu jsou stejné jako u vlastnosti `height`.

3.6 POZICOVÁNÍ

3.6.1 FLOAT

```
.popis {
float: right;
}
```

Tato vlastnost nastaví obtékání prvku na stránce.

hodnoty	způsob obtékání prvku
none	prvek nebude obtékán (výchozí hodnota)
right	prvek bude umístěn k pravému okraji, následující text ho bude obtékat vlevo
left	totéž, ale k levému okraji

3.6.2 CLEAR

```
.popis {
clear: both;
}
```

Zakazuje obtékání u některých prvků:

hodnoty	neobtékané prvky
none	plovoucí prvky jsou obtékané
both	prvek se vykreslí až pod všemi plovoucími prvky
left	prvek čeká na obtékané elementy přiražené doleva
right	prvek čeká na obtékané elementy přiražené doprava

3.6.3 POSITION

```
.popis {
position: relative;
}
```

Určuje, zda bude prvek na stránce umístěn absolutně nebo relativně. Absolutní umístění znamená vypočítání pozice objektu vzhledem k okrajům dokumentu, relativní umístění je umístění vzhledem k pozici, kde by byl objekt normálně vykreslen.

hodnoty	způsob výpočtu umístění
absolute	prvek bude vyjmut z toku dokumentu a umístěn na určitých souřadnicích
relative	prvek bude posunut ze svého normálního umístění
static	prvek je umístěn normálně v dokumentu

3.6.4 UMÍSTĚNÍ

```
.popis {
left: 20px;
}
```

Pokud je u prvku nastavena vlastnost `position: absolute;` nebo `position: relative;`, určí se pomocí vlastností `left` a `top`, kde bude umístěn. V obou případech se zadává hodnota v jednotkách podporovaných CSS nebo v procentech. `left` značí vzdálenost prvku od levého okraje, `top` od horního okraje.

3.6.5 VISIBILITY

```
.schovano {
visibility: hidden;
}
```

Určuje zda bude prvek na stránce viditelný. Pokud viditelný není, zůstane po něm na stránce prázdné místo.

hodnoty	**viditelnost prvku**
visible	normálně viditelný
hidden	neviditelný, ale zabere místo

3.6.6 OVERFLOW

```
.text {
overflow: scroll;
}
```

Pokud se do boxu (např. `<div>` s pevně nastavenými rozměry) nevejde všechen jeho obsah, určí tato vlastnost, co se s ním stane.

hodnoty	**zacházení s přetečeným obsahem**
visible	nechá se přetékat a je vidět
auto	nenechá se přetékat, je-li potřeba, zobrazí se rolovací lišta
scroll	nenechá se přetékat, rolovací lišta se zobrazí vždy, i když není potřeba
hidden	nenechá se přetékat, co se nevejde se nezobrazí, žádná rolovací lišta

3.7 JEDNOTKY A BARVY CSS

3.7.1 JEDNOTKY

V CSS lze používat hned několik druhů jednotek. Zde je jejich přehled:

Absolutní jednotky			
jednotka	**význam**	**příklad**	**velikost písma příkladu**
px	pixel, obrazovkový bod	12px	dvanáct pixelů (obrazovkových bodů)
pt	typografický bod (jako ve Wordu)	9pt	devět typografických bodů (na Windows 12px)
mm	milimetr	5mm	pět milimetrů
in	palec	.5in	půl palce

Relativní jednotky			
jednotka	**význam**	**příklad**	**velikost písma příkladu**
em	šířka písmena "M" (tzv. čtverčík)	1.5em	jeden a půl normální velkosti
%	procento	80%	osmdesát procent normální velikosti
ex	výška písmena "X"	1.5ex	jeden a půl normální velkosti

V praxi se ale používají z absolutních jednotek většinou jen pixely a z relativních procenta a čtverčíky (em).

3.7.2 BARVY

Barvy se v CSS dají zadávat několika různými způsoby:

způsob zápisu	**zříklad: červené písmo**	**poznámka**
Jménem v angličtině	color: red;	Existuje mnoho pojmenovaných barev, ale v praxi se moc nepoužívají.
Procentuálním RGB zápisem	color: rgb 100%,0%,0%);	rgb znamená Red, Green, Blue (červená, zelená, modrá)
Desetinným RGB zápisem	color: rgb(255,0,0);	
Šestnáctkovým RGB zápisem	color: #ff0000;	Tento způsob je nejjistější, nejpoužívanější a nejlepší.
Zkráceným šestnáctkovým RGB zápisem	color: #f00;	Jen v případě, že se všechny dvojice cifer shodují

Pojmenovaných barev jsou desítky ale doporučuje se používat jen 16 základních z Windows, protože ne všechny prohlížeče by právě vámi vybranou barvu musely znát.

Jsou praktické, protože název barvy se pamatuje mnohem lépe než její RGB zápis ale stejně se to moc nepoužívá.

RGB model využívá toho, že všechny barvy jdou získat pouze složením tří základních barevných světel – červené (Red), zelené (Green) a modré (Blue) – ostatně toho se využívá i u monitorů a obrazovek. Zápis RGB tedy značí, jak moc je tam která barva zastoupena a z toho se získá barva výsledná.

Například modrá barva se zapíše takto:

```
rgb (0%,0%,100%)
```

Zatímco žlutá, která se vytvoří smíchání červeného a zeleného světla má zápis:

```
rgb (100%,100%,0%)
```

RGB zápis barvy tedy udává zastoupení jednotlivých barev.

Může být zapsán procentuelně a nebo pomocí poměru jednotlivých barev. Ten se zapisuje v desítkové (0-255) nebo v šestnáctkové (00-ff) soustavě. V šestnáctkové soustavě lze ještě využít tzv. zkráceného zápisu – ovšem jen v případě, že jsou všechny dvojice cifer stejné: např. `#ff5511` lze zkrátit na `#f51`.

4. PHP

4.1 ÚVOD DO PHP

4.1.1 HISTORIE PHP

Skriptovací jazyk PHP vzniknul v roce 1994. Tehdy jistý pan Rasmus Lerdorf chtěl vytvořit program pro počítání přístupů na jeho web. Po několika úpravách bylo vydáno *Personal Home Page Tools* - zkráceně PHP.

Dále se tento projekt roku 1995 spojil s jiným projektem téhož autora a vzniklo PHP/FI 2.0.

Roku 1998 bylo vydáno PHP 3.0. Tato verze již fungovala i pod Windows a byla hojně rozšířena. Jednalo se o plnohodnotný nástroj pro tvorbu webu.

Následovala verze 4, která se na některých serverech stále ještě používá. Pracuje na jádru Zend.

V současnosti se využívá PHP verze 5. Ne všude, protože spousta autorů je pochopitelně líná svůj čtyřkový kód přepsat do pětkového. PHP 5 pracuje na jádru Zend II a má již kvalitní podporu objektově orientovaného programování.

4.1.2 PRINCIP PHP

Webové skripty slouží k tvorbě dynamických webových prezentací. Dělíme je na klientské skripty a serverové skripty.

Klientské skripty běží na samotné webové stránce. Prohlížeč, který stránku stáhne, tento skript pak spustí při splnění nějaké události (kliknutí na tlačítko, načtení dokumentu...). Nevýhodou je, že prohlížeč tyto skripty musí podporovat. Nejčastěji používaným zástupcem je JavaScript.

Serverové skripty běží na serveru a prohlížeč s nimi vůbec nepřijde do styku (není zde tedy nutná podpora prohlížečů). Po zaslání dotazu na stránku tento dotaz skript vyhodnotí, stránku na serveru sestaví podle požadavků a zpět odešle již jen čistý XHTML kód.

4.1.3 CO JE POTŘEBA?

Chceme-li si vyzkoušet PHP doma v počítači, potřebujeme nějaký HTTP server a instalaci požadované verze PHP. Jako server se nejčastěji využívá server Apache (je to sesterský projekt PHP a je freeware). Použitá verze PHP záleží na vás, ale doporučil bych PHP 5, protože je nejnovější, ale zároveň již zaběhlá a bez chyb, které se novému produktu nikdy nevyhnou.

Nejjednodušší je pořízení nějakého instalačního balíčku, který obsahuje jak Apache a PHP, tak databázi, se kterými PHP hojně spolupracuje.

Já osobně používám VertrigoServ, který je volně ke stažení na adrese: http://www.sourceforge.net/...

Potěší, že narozdíl od konkurenčního ASP (Active Server Pages) od Microsoftu, je vše, co se točí kolem PHP, feeware.

Když ale hotovou webovou stránku s PHP otevřete normálně jako soubor, PHP nefunguje a skript se zobrazí se jako obsah stránky. Pokud chcete aby PHP normálně fungovalo, je třeba aby běžel Apache server a stránka se nesmí otevírat jako soubor. Musí být uložena v localhost, jehož umístění se různí podle instalace Apache. Pokud využijete avizovaného VertrigoServ, přesuňte všechny svoje projekty s PHP do c:\Program Files\VertrigoServ\www*.*. Přes prohlížeč se na ně pak dostanete z http://localhost/*.*.

4.2 ZÁKLADY SYNTAXE

4.2.1 ZÁPIS DO DOKUMENTU

Kód PHP jde zapsat přímo do stránky XHTML nebo může být celá stránka čistě PHP, na které se pak nějaká jiná stránka odkazuje. Takový dokument ale musí mít příponu *.php, a to i v případě, že je to z větší části spíše XHTML dokument.

PHP kód se ve stránce odděluje znaky <?php a ?>. Často se také užívá zkrácená podoba <? a ?>. To sice není úplně správně, ale funguje to a nejsou s tím problémy.

```
<p>
Tato stránka je částečně generována pomocí
<?php
echo "jazyka PHP.";
?>
</p>
```

Příkaz echo zobrazí (vytiskne) svůj obsah. Když se podávíte do zdrojového kódu, je tam pouze text "jazyka PHP." - veškeré operace se tedy provedou na serveru a do prohlížeče dorazí již hotová stránka. K PHP kódu se uživatel nedostane.

Jednotlivé příkazy se oddělují středníkem:

```
<?php
echo "první řádka";
echo "druhá řádka";
?>
```

4.2.2 KOMENTÁŘE

Jako v každém jazyce, tak i v PHP jsou nepostradatelné komentáře. Ty mohou být buď jednořádkové, kde konec řádku v kódu ukončí komentář, nebo víceřádkové.

Jednořádkový komentář

```
// text komentáře
```

Víceřádkový komentář

```
/*
První řádka komentáře
Druhá řádka komentáře
*/
```

Opět se doporučuje komentáře hojně používat, aby jste se v obsáhlejších projektech neztratili.

4.2.3 PROMĚNNÉ

Práce s proměnnými je zde velmi jednoduchá. Žádné proměnné se totiž nemusí deklarovat, ani se nemusí udávat jejich typ. To všechno si program vyřeší sám, když do nějaké proměnné přiřadíme určitý obsah.

Přesto, že se o to nemusíme starat, i zde fungují základní typy proměnných. Jsou to integer, real (float), string a boolean.

Každá proměnná je jedinečná a musí začínat znakem $ (dolar). Následovat může libovolné písmeno a-z nebo podtržítko. Pozor, PHP je citlivé na malá a velká písmena.

Přiřazení hodnoty do proměnné se provede pomocí rovnítka (=).

Příklad:

```
<?php
$oblibene_cislo = 7; // integer
$vek = 19.5; // real, misto desetinné čárky se používá tečka
$jmeno = "Petr je borec"; // string, řetězec musí být v uvozovkách
$muz = true; // boolean
?>
```

4.3 PŘÍKAZ ECHO

Nejzákladnější příkaz, který vás bude provázet celým PHP je echo. Ten vypíše nějaký text do dokumentu. Text, který chcete vypsat, se musí zapsat do uvozovek.

```
<?php
echo "tenhle text je vypsaný pomocí echo";
?>
```

4.3.1 ZNAKOVÉ ENTITY

Stejně jako v XHTML jsou i zde speciální znaky (středník, uvozovky...), které nejdou zobrazit normálně, protože mají speciální význam. Ty se zapisují pomocí tzv. escape sekvencí.

Sekvence	Význam
\n	Nový řádek
\"	Uvozovky
\r	Návrat vozíku
\t	Tabulátor
\\	Zpětné lomítko
\$	Dolar

Veškerý text, který takto pomocí PHP zapíšete, se pak zobrazí ve zdrojovém kódu dokumentu.

Například

```
<?php
echo "první řádka \n druhá řádka";
?>
```

Se sice v kódu zobrazí takto:

```
první řádka
druhá řádka
```

Ale v prohlížeči je text pořád na jedné řádce, protože chybí element `<br />`. Dostat se tam dá takto:

```
<?php
echo "první řádka <br /> \n druhá řádka";
?>
```

Obsah proměnné vypíšete jednoduše:

```
<?php
echo $promenna;
?>
```

4.3.2 SPOJOVÁNÍ ŘETĚZCŮ

Spojování řetězců se v PHP neprovádí pomocí znaménka +, ale pomocí tečky.

```
<?php
$vek = 36;
echo "Je mi ".$vek." let";
?>
```

4.4 POLE

I PHP umí pracovat s poli. Hodnoty se dají do pole zapsat několika způsoby.

```
<?php
$lide[0]="Jana";
$lide[1]="Petr";
$lide[2]="Láďa";
$lide[3]="Honza";
?>
```

nebo kratším zápisem

```
<?php
$lide = array ("Jana" , "Petr" , "Láďa" , "Honza");
?>
```

Pozor, pole se číslují od nuly.

Pole mohou být i dvojrozměrná a jako indexy není nutné uvádět čísla. Našel jsem na to jeden příklad s šachovnicí.

```
<?php
$figura ["a"][1]="bílá věž";
$figura ["b"][1]="bílý jezdec";
//...atd...
$figura ["h"][2]="bílý pěšec";
//...atd...
$figura ["g"][8]="černý jezdec";
$figura ["h"][8]="černá věž";
echo "Na poli b1 je při zahájení šachové partie ".$figura["b"][1];
?>
```

4.5 OPERÁTORY

Každý programovací jazyk má zápis logických operátorů trochu jiný. Proto zde uvádím operátory použitelné v PHP.

4.5.1 LOGICKÉ OPERÁTORY

Operátor	Význam	Pravda když
`!`	Negace	True když bylo False a naopak
`and`	Logický součin	Jsou obě hodnoty pravdivé
`&&`	Logický součin	Jsou obě hodnoty pravdivé
`or`	Logický součet	Je alespoň jedna hodnota pravdivá
`||`	Logický součet	Je alespoň jedna hodnota pravdivá
`xor`	Exkluzivní OR	Je právě jedna hodnota pravdivá

4.5.2 POROVNÁVACÍ OPERÁTORY

Operátor	Význam
<	Je menší
>	Je větší
<=	Je menší nebo rovno
>=	Je větší nebo rovno
==	Rovná se
<>	Nerovná se
!=	Nerovná se

4.5.3 INKREMENTACE

PHP umí ještě jednu šikovnou věc, o které bych se v této kapitole chtěl zmínit. Nazývá se inkrementace.

Pokud mám proměnnou a chci její hodnotu zvýšit o jeden, mám dvě možnosti, jak to udělat.

```
<?php
$cislo=2;

$cislo=$cislo+1; // je to správně, ale zbytečně dlouhé

$cislo++; // inkrementace - podstatně kratší a přehlednější
?>
```

4.6 PODMÍNKY

4.6.1 IF

Jako snad v každém programovacím jazyce, tak i v PHP existují podmínky. Ty slouží většinou k větvení kódu. Zápis je jednoduchý:

```
<?php
if ($promenna == 2)
{
echo "podmínka je splněna";
}
?>
```

Podmínka, která musí být splněna se uvádí do závorky za klíčové slovo if. Pokud je proměnná typu boolean, lze zápis zkrátit.

```
<?php
if ($promenna)
{
echo "podmínka je splněna";
}
?>
```

Složené závorky obsahují veškerý kód, který se provede při splnění podmínky. Jejich využití není jen u příkazu if, ale všude, kde je potřeba označit část kódu.

Při nesplnění podmínky, se buďto nic nestane, nebo je na řadě případný kód za příkazem else.

```
<?php
if ($promenna)
{
echo "podmínka je splněna";
}
else  // co se stane, pokud podmínka neplatí
{
echo "podmínka není splněna";
}
?>
```

Poslední možností je příkaz elseif. Jak z názvu vyplývá, nastane v situaci, kdy není splněná předchozí podmínka if a zároveň je splněná další podmínka.

```
<?php
if ($jmeno== "Petr")
{
echo "Ahoj Petře";
}
elseif ($jmeno== "Lucie")  // neplatí předchozí a zároveň platí stávající
podmínka
{
echo "Ahoj Lucko";
}
else
{
echo "Tebe neznám";
}
?>
```

4.6.2 SWITCH

Pokud je podmínek více, dají se nahradit příkazem switch. Takto vypadá předchozí (a trochu pozměněný) příklad přepsaný pomocí něho.

```
<?php
switch ($jmeno)
{
case "Petr":
echo "Ahoj Petře";
break;
case "Lucie":
echo "Ahoj Lucko";
break;
case "Ladislav":
echo "Čau Láďo";
break;
default:
echo "Tebe neznám"
}
?>
```

default je výchozí hodnota, která nastane, pokud neplatí ani jedna z podmínek.

4.7 CYKLY

4.7.1 CYKLUS FOR

Nejjednodušším cyklem, který se vyskytuje snad ve všech programovacích jazycích, je cyklus for. Jedná se o tzv. cyklus s parametrem, u kterého je předem určeno, kolikrát se bude opakovat.

```
<?php
for ($i=1; $i<=5; $i++) // celý cyklus se zopakuje pětkrát
{
echo "tohle je ".$i.". řádek <br />";
}
?>
```

Funguje to tak, že se nejprve před během celé smyčky přiřadí nějaká hodnota proměnné $i ($i=1). Následně se vyhodnotí podmínka ($i<=5), která když platí, tak proběhne obsah smyčky. Jako poslední máme zvýšení hodnoty $i o jedna ($i++), které proběhne až po dokončení obsahu smyčky. Pokud i nadále platí podmínka ($i<=5), smyčka pokračuje; pokud ne tak se ukončí.

4.7.2 WHILE

Další možností je smyčka s podmínkou na začátku, která se zapíše jako while (česky dokud). Takto by vypadal předchozí příklad řešený pomocí ní:

```
<?php
$i=1;
while ($i<5)  // dokud platí podmínka
{
echo "tohle je ".$i.". řádek <br />";
$i++;
}
?>
```

Obsah smyčky se provádí to té doby, dokud platí udaná podmínka.

Cyklus while se dá přepsat tak, že bude mít podmínku na konci. Moc se to ale nevyužívá.

```
<?php
$i=1;
do
{
echo "tohle je ".$i.". řádek <br />";
$i++;
}
while ($i<5)  // podmínka je na konci
?>
```

Změna je v tom, že obsah smyčky se nejprve provede a až potom kontroluje podmínku. Jednou tedy proběhne i v případě, že podmínka nikdy nemůže platit.

4.8 FUNKCE A PROCEDURY

Pokud máte na stránce část kódu, která se tam vyskytuje vícekrát, byla by otročina jí tam neustále kopírovat. Proto existují funkce a procedury, které stačí jednou nadeklarovat a pak je můžete kdykoliv zavolat podle potřeby.

4.8.1 PROCEDURY

Například:

```
<?php
function hvezdicky ()
{
echo"* * * * *";
}
?>
```

Pokud pak chci kdekoliv na stránce vypsat 5 hvězdiček, stačí jednoduše zavolat proceduru hvezdicky:

```
<?php
hvezdicky;
?>
```

pozn.: Závorky za názvem funkce slouží k předávání parametrů.

Nezáleží na pořadí, takže můžeme funkci klidně zavolat ještě dřív, než je definována v celku. Kvůli přehlednosti to ale stejně moc lidí nedělá.

Rozdíl mezi procedurou a funkcí je ten, že procedura pouze provede kód, který obsahuje, zatímco funkce navíc ještě vrací nějakou hodnotu. hvezdicky z příkladu jsou tedy procedura.

Obojí, funkce i procedury, se ale v PHP zapisují klíčovým slovem function.

4.8.2 FUNKCE

```
<?php
function secti ()
{
$cislo1=5;
$cislo2=3;

return ($cislo1 + $cislo2);
}
?>
```

Klíčové slovo return udává hodnotu, kterou funkce vrátí. Každá funkce může vracet pouze jednu hodnotu.

Funkce se pak klidně dá začlenit do ostatního textu.

```
<?php
function secti ()
{
$cislo1=5;
$cislo2=3;

return ($cislo1 + $cislo2);
}


echo 1+secti (); // vypíše hodnotu 9 (5+3+1)
?>
```

4.8.3 FUNKCE S PARAMETREM

Pokud chceme (a že většinou chceme) aby například funkce secti sčítala libovolná čísla, která jí program předá, je třeba přiřadit jí parametry. Pro ně je určena ta závorka za funkcí.

```
<?php
function secti ($cislo1, $cislo2)
{
return ($cislo1 + $cislo2);
}

echo secti (5, 9);
?>
```

Pozor, funkce pracují jen s proměnnými které jim předáme v parametru. Nelze proto odkázat na proměnnou uvedenou jinde v kódu.

To jde obejít jedině když použijeme u požadované proměnné klíčové slovo global.

Tohle tedy bude fungovat:

```
<?php
$cislo1=10;

function pricti ($cislo2)
{
global $cislo1;
return ($cislo1 + $cislo2);
}

echo pricti(5); // vrátí 15
?>
```

4.9 ZPRACOVÁNÍ FORMULÁŘŮ

Protože webové stránky jsou interaktivní záležitost, jednoznačně k PHP patří i formuláře. Díky nim může uživatel zadávat nějaká vstupní data, se kterými pak PHP skript pracuje.

Asi nejjednodužší je udělat si dvě stránky. První bude čisté XHTML s formulářem a druhá bude obsahovat PHP skript, který odeslané informace zpracuje.

První stránka (formular.htm) obsahuje takovýto formulář:

```
<form action="./zpracuj.php" method="post">
<p>
Co jste měli dnes k obědu:
<input type="text" name="jidlo" />
<input type="submit" value="potvrdit" />
</p>
</form>
```

Jako hodnota atributu `action` je nastavena stránka, na kterou se zadané informace odešlou (zde stránka zpracuj.php).

Druhá stránka (zpracuj.php):

```
<?php
echo "Dnes jste obědvali ".$_REQUEST['jidlo'];
?>
```

Proměnná $_REQUEST je asociativní pole, které obsahuje všechna odeslaná data. Namísto toho lze ještě použít pole $_POST a $_GET, která rozlišují data podle metody odeslání.

Tento příklad lze ještě přepracovat tak, aby se využívala jen jediná stránka, která bude odkazovat sama na sebe. Ta musí být pochopitelně uložena jako PHP. Kód by pak vypadal následovně:

```
<?php
if (empty ($_REQUEST))
{
?>

<form action="./zpracuj.php" method="post">
<p>
Co jste měli dnes k obědu:
<input type="text" name="jidlo" />
<input type="submit" value="potvrdit" />
</p>
</form>

<?php
}
else
{
echo "Dnes jste obědvali ".$_REQUEST['jidlo'];
}
?>
```

Vysvětlení je jednoduché. Funkce empty zjistí, zda je prázdná proměnná $_REQUEST. Pokud ano, nebyl ještě formulář odeslán, a proto ho zobrazíme. Když je proměnná naplněna, nemá již smysl vypisovat formulář, ale rovnou zobrazíme výstup PHP, které zpracovalo odeslané informace.

Z uživatelského hlediska bude vše vypadat stejně, jako předchozí varianta.

5. ZÁVĚR

Dle mého názoru je webová prezentace jeden z nejefektivnějších a zároveň nejlevnějších způsobů reklamy. Přitom udělat jednoduchý web není takový problém, jak to může na první pohled vypadat. Myslím, že toto stále se vyvíjející téma má jistou budoucnost a určitě se vyplatí, vědět o něm něco víc. Proto se jsem se snažil, aby mohla být moje práce použita i pro studijní účely.

6. POUŽITÁ LITERATURA A JINÉ ZDROJE

WEBY

www.jakpsatweb.cz
Perfektní web o tvorbě stránek. Zabývá se HTML, CSS, JavaScriptem a spoustou dalších věcí. Vše je psáno hezky polopaticky a proto je to nejlepší způsob jak s webem začít. Vřele doporučuji.

www.linuxsoft.cz
Jak název napovídá, zabývají se tyto stránky především Linuxem. Mimo to se tam dá ale najít třeba skvělý seriál o PHP od Petra Zajíce nebo jiný seriál o MySQL.

www.jaknaweb.com
Spousta odkazů na články zabývající se webovou tvorbou. Odkazuje i na jiné servery.

www.interval.cz
Další server o webu. Našel jsem zde bezva seriál o XHTML od Martina Snížka.

KNIHY

Kučera Miroslav. CSS – úvod do kaskádových stylů.
Mobil media a.s. 2002

Gröpl Tomáš. HTML, CSS a JavaScript – referenční příručka.
BEN – technická literatura, Praha 2002.

7. OSTATNÍ

7.1 SLOVNÍČEK POJMŮ

Ampersand
Anglický výraz pro znak `"&"` (and – česky a)

Atribut
Součást elementu. Většinou bývá nepovinná, ale není to pravidlo. Udává doplňující informace (konkrétní vlastnosti) o elementu.
Například: `<img src="./cosi.jpg" alt="obrazek" />`
Atribut src udává cestu k obrázku a atribut alt obsahuje alternativní text.

Bílá místa
(v kódu) – mezery a tabulátory. Je jedno, kolik jich tam je, protože tabulátory prohlížeč ignoruje a více mezer za sebou nahradí mezerou jedinou. Slouží tedy jen pro lepší orientaci při tvorbě nebo editaci stránky.

Blokový prvek
Element, chápaný jako blokový může obsahovat další blokové i inline prvky. Následující prvky jsou automaticky na další řádce.

Browser
(prohlížeč) viz. Interpret XHTML

CSS
Neboli Cascading Style Sheets (česky kaskádové styly), je soubor metod pro vizuální úpravu webových prezentací.

Element
Část XHTML dokumentu. Začíná otevíracím tagem, následuje obsah elementu (nějaký text) a pak je uzavírací tag. Z jednotlivých elementů se skládá celá stránka.
Příklad elementu: `<p>První odstavec</p>`

HTTP hlavička
Informace o dokumentu, významné pouze pro prohlížeč. V kódu stránky se nezobrazují.

Inline prvek
Element, který může obsahovat pouze jiné inline prvky. Následující elementy začínají na stejné řádce.

Interpret XHTML
Program, který načte zdrojový kód webové stránky, přeloží ho do grafické podoby (vyhodnotí ho) a zobrazí (interpretuje). Často se používá výraz prohlížeč.

PHP
Neboli "Personal Home Page" je programovací jazyk určený pro tvorbu dynamických webových prezentací.

Skript
Malý program, který je součástí jiného většího celku (stránek). Většinou běží někde na pozadí a uživatel o něm ani neví.
Serverový skript běží na serveru a k uživateli dorazí jen čisté XHTML. Například PHP, ASP.
Klientský skript se stáhne spolu se stránkou a probíhá v prohlížeči. Pomocí něho se dají udělat například vysouvací menu a podobně. Nejrozšířenější je JavaScript.

Tag
Část elementu. Může se jednat buď o otevírací část elementu (`<p>`) nebo o uzavírací část elementu (`</p>`)

XHTML
eXtensible HyperText Markup Language (rozšiřitelný hypertextový značkovací jazyk) je jazyk určený pro tvorbu webových dokumentů.

Znaková Entita
Protože v jazyce XHTML jsou znaky, které pro něj mají specifický význam, nelze je uvést rovnou jako obsah dokumentu. Proto existují znakové entity, které tyto prvky v dokumentu nahrazují, a za které si prohlížeč následně dosadí správné znaky. Jejich přehled je v kapitole o znakových entitách.

7.2 MOJE WEBOVKY

Tato práce je ochuzena o spoustu interaktivních příkladů. Pokud máte zájem o opravdu plnohodnotný přehled tvorby webu, podívejte se na http://mw.unas.cz. Tento textový dokument je pouze zkopírovaný a protříděný obsah zmíněné adresy.