Pedagogická fakulta,
Jeronýmova 10, 371 15 České Budějovice

**Tvorba internetových aplikací v XHTML 2.0**

# BAKALÁŘSKÁ PRÁCE

Vedoucí absolventské práce: PaedDr. Petr Pexa

Autor: Vojtěch Soukup

České Budějovice, 2004

## PODĚKOVÁNÍ

Děkuji PaedDr. Petru Pexovi za hodnotné rady a odborné vedení během mé práce.

## PROHLÁŠENÍ

„Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně, a že jsem uvedl veškerou použitou literaturu“.

## Vojtěch Soukup

# Obsah

# 1 Úvod

V bakalářské práci se zabývám značkovacími jazyky sloužícími pro prezentaci informací systémem WWW (World Wide Web). Hlavním tématem mé práce je značkovací jazyk XHTML 2.0.

Historie značkovacích jazyků začíná jazykem SGML a pokračuje jazyky HTML, XML a XHTML. V srpnu 2002 se objevil jazyk XHTML 2.0. Tento jazyk je nástupcem jazyka XHTML 1.1. Z tohoto důvodu uvedu obecná pravidla platící pro tvorbu dokumentu v jazyce XHTML 2.0 a v jednotlivých kapitolách podrobně porovnám se staršími jazyky. Cílem mé práce je obeznámit čtenáře se značkovacím jazykem XHTML 2.0 (a samozřejmě i HTML a XML z nichž vychází). Snažím se srozumitelným způsobem ozřejmit základy tohoto jazyka, poukázat na jeho silné i slabé stránky při tvorbě internetových/intranetových informačních systémů. Předpokládám základní znalosti jazyka HTML, a proto zde uvedu pouze základní syntaxi HTML, XHTML 1.0 a dále se budu zabývat jazykem XHTML 2.0.

# 2 Historie WWW a vývoj značkovacích jazyků

Web je v dnešní době velice oblíbený a využívaný nástroj. Neslouží jen k pobavení uživatele, nýbrž může posloužit i ke vzdělávání a studiu, můžeme si přečíst denní tisk, nahlédnout do encyklopedie, učebnice nebo se nechat informovat o jiných knihách. Nalezneme tam i nejrůznější hudební skladby a nahrávky, zajímavé fotografie, informace o cestování, o počasí, o počítačích. I cestovní kanceláře a různé jiné firmy a podnikatelé nabízejí svoje služby a produkty prostřednictvím internetu.

Ve svých počátcích byl web statický a jednosměrný. Uživateli dovoloval pouze shlédnout obsah. V dnešní době je tomu jinak, web komunikuje s uživatelem pomocí různých formulářů, kde si uživatel může vybrat zboží, či napsat nějaký svůj vzkaz.

Internet pro nás není pouhé informační médium. Je pravdou, že jako každé jiné médium (knihy, časopisy, rozhlas, televize) má i web svůj obsah a formu – co nám chce sdělit a jakým způsobem to chce sdělit. Velkou

atraktivitu webu přinesla právě jeho mnohotvárná forma. Vytvořit si svou vlastní webovou prezentaci není nikterak obtížné.

Každý jazyk se skládá z určitých symbolů. Když je smysluplně sestavíme, můžeme vyjadřovat a předávat různé informace. Existují jistá pravidla, která nám říkají, jak dané symboly poskládat do slov, vět či odstavců tak, aby informace byla srozumitelná i ostatním. To jistě známe z gramatiky.

Podobně je tomu i s počítačovými jazyky. Nadefinujeme jeden speciální jazyk, **metajazyk**, který vymezuje pravidla a symboly jiných jazyků. Pomocí metajazyka jsou přesně určena obecná pravidla pro definici jazyků a podle nich pak jeden, či více konkrétních jazyků nadefinován.

Metajazyky pomáhají ve vytváření automatizovaných agentů, kteří zobrazují nebo zpracovávají obsah dokumentu. Mezi metajazyky patří níže zmíněný SGML i XML.

## 2.1 Jazyk SGML

Jedná se o standardní jazyk určený k formálnímu popisu struktury dokumentů. Počátkem 70. let vyvinula firma IBM značkovací jazyk, který měl usnadnit přenositelnost dat. Základem byl jazyk **GML** (*Generalized Markup Language*) vytvořený v roce 1969. Teprve roku 1986 byl tento jazyk standardizován normou ISO-8879 (organizace ANSI – *American National Standardization Institute*). Byl označen jako **SGML** (*Standard Generalized Markup Language* – standardizovaný obecný značkovací jazyk).

Vznikl v rámci projektu **ODA** (*Open Document Architecture*). Cílem ODA je poskytnout standardní architekturu pro vytváření, předávání, uchovávání a zpracování různorodých dokumentů v elektronické podobě. Zahrnuje proto různé standardy pro formáty dat, architekturu předávání zpráv, zabezpečení informací atd.

Pro potřeby projektu ODA bylo zapotřebí formátu umožňujícího uložení textů v elektronické podobě a nezávislého na softwarové a hardwarové platformě. A přitom poskytoval dostatečnou flexibilitu. Jediným možným řešením se stal značkovací jazyk (*Markup Language*).

Jazyk SGML je značně univerzální, což sebou přináší vysokou složitost. Právě složitost tohoto jazyka zabránila jeho většímu rozšíření. SGML je velmi vhodný pro popis vzájemného vztahu dat a hlavně pro následné strojové zpracování. Díky výše uvedené složitosti se používá jen v systémech pracujících s velkými objemy dat, kde se možnosti tohoto jazyka plně uplatní.

Programy, které s ním pracují, jsou značně složité. To by brzdilo rozvoj internetu a zpomalilo vývoj aplikací a prohlížečů webových stránek. Proto se hledal jiný jazyk, který by byl jednodušší než SGML, byl přístupný pro široké spektrum vývojářů a neodrazoval od použití. Tímto jazykem se stal HTML.

### 2.1.1 Využití SGML

SGML není omezenou definicí značek. Nejedná se o formátovací jazyk, ani o konkrétní značkovací jazyk. Je to specifikace, která umožňuje vytvářet vlastní značkovací jazyky. Tento metajazyk umožňuje definovat vlastní sady elementů a jejich vzájemné vztahy pomocí tzv. definic typu dokumentu (DTD).

Původně to měl být snad jediný značkovací metajazyk, jelikož se v něm může nadefinovat všechno, od egyptských hieroglyfů až po jazyk HTML. Je však příliš obecný a má tak velkou šíři záběru, že s ním normální člověk nemůže ani pracovat. Z těchto důvodů byly postupně vytvořeny i další značkovací jazyky, s nimiž se pracuje lépe, protože mají omezenou šíři záběru.

Je zcela otevřeným standardem nezávislým na hardware nebo aplikacích. Soubory SGML jsou ukládány jako prostý text ve formátu **ASCII** (*American Standard Code for Information Interchange* – Americký standardní kód pro výměnu informací, osmibitová abeceda tvořící základ pro většinu dnes používaných abeced pro kódování písmen a číslic).

Hlavní význam SGML se projeví v okamžiku, kdy se začnou používat příslušné značkovací příkazy. Jimi je definována struktura a vnitřní vztahy v dokumentu. Narozdíl od nestrukturované informace otevírá zcela nové možnosti zpracování, publikování a opakovaného používání této informace. Je vhodný pro projekty, kde se vyžaduje velké množství podobně

strukturovaných údajů (katalogy, manuály, seznamy, přepisy, statistické přehledy apod.).

Jak jsem již naznačil, největší výhodou systémů využívajících SGML je jejich otevřenost. Používá se standardu, který je neměnný, a proto nehrozí další utrácení finančních prostředků v důsledku nutnosti přechodu na nové verze.

Při vytváření SGML dokumentů se zajímáme pouze o obsah a nikoliv o jejich formu. Tu určíme až podle potřeby v závislosti na požadovaném výstupu. Znamená to, že stačí změnit pouze pravidla formátování a dokument je připraven pro výstup (například na jehličkovou tiskárnu, laserovou tiskárnu, webový server apod).

## 2.2 Jazyk HTML

Tento jazyk také vychází z SGML, je však výrazně jednodušší a do značné míry redukuje možnosti SGML. Díky této redukci je dosaženo mnohem jednoduššího kódu.

**HTML** (*Hypertext Markup Language*) byl vytvořen v roce 1991 Timem Bernersem-Lee, který použil jako základ výše zmiňovaný jazyk SGML. Tento jazyk se velmi rychle rozšířil a stal se nejpoužívanějším formátem na platformě WWW. Na jeho rozšíření se výraznou měnnou podílely i samotné prohlížeče. Pokud prohlížeč příslušnému tagu rozumí, provede ho, v opačném případě ho ignoruje. Toto platí i o atributech. Pokud někde chybí koncový tag, prohlížeč se jej snaží odhadnout. Tyto skutečnosti ulehčovaly amatérským tvůrcům webových stránek jejich práci a přispěly k masovému rozšíření jazyka HTML.

Na druhou stranu rychlý nástup jazyka HTML měl za následek jeho nekonzistentnost. Mnoho programátorů vytvářelo své stránky a jazyk HTML je nijak neomezoval. Takto zapsaná data jsou proto nevhodná pro strojové zpracování a není ani žádný jednotný algoritmus pomocí kterého by bylo možno takováto data analyzovat (v mnoha případech nejsou použitelná vůbec).

### 2.2.1 Využití jazyka HTML

Jazyk HTML původně sloužil jako nástroj pro psaní velmi strohých dokumentů a praxi tudíž příliš nevyhovoval. Umožňoval použití pouze několika druhů zvýraznění textu, vkládání odkazů a obrázků. Byl využíván převážně pro vědecké dokumenty. Chybělo mu totiž jakékoli zkrášlení, které dnes známe.

Mnoho tagů tohoto jazyka bylo v rámci konkurenčního boje navrženo výrobci jednotlivých prohlížečů a pouze následně byly některé z nich zahrnuty do doporučení organizace W3C o struktuře HTML. Důsledkem je nekompatibilita některých tagů v prohlížečích různých výrobců. Ještě dnes nalezneme tagy, které jsou správně zobrazeny jen v určitých prohlížečích a v jiných jsou ignorovány. Nejvýrazněji do tohoto konkurenčního boje zasáhly firmy Netscape a později i Microsoft.

## *2.3 Jazyk XML*

Poměrně nový jazyk a daleko mocnější než HTML je **XML** (*eXtensible Markup Language* – rozšiřitelný značkovací jazyk). Je také odvozen od SGML, je jednodušší než SGML, ale využívá z něj více možností než HTML. Zachovává si možnost definovat vlastní DTD a tudíž i vlastní elementy. Na rozdíl do SGML má předem určeny některé parametry – maximální délku názvů elementů, speciální znaky nebo použité oddělovače.

Je to metajazyk pomocí kterého je definován další značkovací jazyk – XHTML (o něm se zmíním níže).

XML je jazyk nedávno vytvořený, a tak bude určitou dobu trvat, než se začne masově využívat pro tvorbu webových stránek. Je však snadno rozšiřitelný a jeho velkou výhodou je i možnost vytváření vlastních tagů.

HTML je velice rozšířený, ale můžeme říci, že i do jisté míry omezující, protože v každé jeho verzi je pevně dána sada tagů, které můžeme používat. Jazyk XML se může zase zdát pro začátečníky složitý a nepřehledný, ale zkušený webdesigner jistě jeho kvality brzy ocení, i když bude nutno používat v dokumentech přísnější syntaxi.

Podstatnou novinkou jazyka XML je **logické uspořádání dat**. Využívá se hlavně tam, kde je třeba elektronicky zpracovávat data či hledat nějaké informace. Důvodů, proč logicky uspořádávat data na internetu je mnoho.

Jedním z nich je například možnost snadnějšího a přesnějšího vyhledávání dat, neboť u textových údajů bude definována i jejich logická struktura. Nebudeme se tudíž muset spoléhat na fulltextové vyhledávače, jejichž pomocí nalezneme všechny stránky, které obsahují hledaný text, ať už má s námi hledaným významem něco společného nebo ne.

Jiným důvodem je využívání dat z internetu. Protože data nemají definovanou strukturu, je jejich převod z formátu HTML obtížný a nelze použít univerzální převodní programy do jiných formátů (např. databází). Pokud by data byla ve formátu XML, tento převod by byl bezproblémový. XML lze tedy považovat za použitelný kompromis mezi složitým jazykem SGML (vhodným pro strojové zpracování) a uživatelsky přívětivým jazykem HTML.

Snahou jazyka XML je využít kladné vlastnosti jazyka SGML a vytvořit nástroj, který nebude komplikovaný a přesto bude výkonný. XML je typem formátu, který nám říká, jak můžeme data zapsat i spolu s jejich významem.

Formální počátek tohoto jazyka můžeme datovat rokem 1998 kdy vzniklo oficiální doporučení konsorcia W3C nesoucí název W3C XML 1.0 Recommendation.

Pro svoji filosofii přenositelnosti a platformové nezávislosti aspiruje na pozici univerzálně použitelného formátu pro přenos dat nejen mezi počítači, ale také například mezi specializovanými zařízeními, mobilními telefony, apod.

## 2.4 Jazyk XHTML

Jazyk XML nám nabízí velice lákavou možnost vytvářet si svoje vlastní sady tagů. Avšak u většiny dokumentů zobrazovaných na WWW je to zbytečné. Proč dělat vše od začátku a nevyužít tagy již exitující v HTML a prověřené časem? Nabízí se nám tedy jiná možnost využití jazyka XML. Dále využívat tagy z jazyka HTML a pouze je doplnit o pár dalších tagů, které budou označovat místa důležitá pro vyhledávací roboty.

Tuto možnost nám nabízí značkovací jazyk – **XHTML** (*eXtensible HyperText Markup Language* – rozšiřitelný hypertextový značkovací jazyk). Je syntézou jazyků HTML 4.0 a XML 1.0 a upravených typových definic dokumentu (DTD), které určují strukturu, elementy a atributy daného dokumentu. V podstatě se jedná o „přidání inteligence“ do jazyka HTML a zpřísnění syntaktických pravidel.

Organizace W3C počátkem roku 2000 vydala doporučení XHTML 1.0 Recommendation definující tento jazyk.

Vývojářům se tedy naskytuje možnost využívat v rámci jednoho dokumentu možnosti HTML i XML, což začíná být v dnešní době čím dál více oblíbenější. Tak mohou tvůrci webových stránek snadněji přecházet se svými projekty z formátu HTML do XHTML.

Jazyk XHTML je zpětně kompatibilní s HTML.

## 2.5 Jazyk XHTML 2.0

XHTML 2.0 je nástupce jazyka XHTML 1.1. Byl vytvořen, aby ujednotil syntaxi jazyka. Jazyk je složen z tagů (značek), které používáme v HTML 4 a XHTML 1.1. V podstatě se jedná o „přidání inteligence“ do jazyka HTML a zpřísnění syntaktických pravidel.

Organizace W3C počátkem roku 2002 vydala doporučení XHTML 2.0 Recommendation definující tento jazyk.

Jazyk XHTML 2.0 je v současné době kompatibilní s předchozími jazyka (HTML a XHTML 1.1). Předpokládá se, že kompatibilita s předchozími jazyky bude zrušena.

# 3 Principy jazyka HTML

## 3.1 Obecný popis

HTML, neboli HyperText Markup Language je značkovací jazyk, který je definován v rámci jazyka SGML a slouží k vytváření dokumentů.

HTML je nejrozšířenější a nejoblíbenější jazyk při publikování na webu. HTML se zpočátku využíval pro tvorbu webových stránek. Postupně pronikl do dalších oblastí:

- nápovědy,
- dokumentace,
- prezentace,

Při používání tohoto jazyka musíme dodržovat jistá pravidla. Pro jejich pochopení si nejprve musíme objasnit některé základní pojmy jako jsou tagy, elementy, atributy nebo odkazy. Pochopení základních pojmů je stěžejní pro snadné pochopení následujících kapitol.

## 3.1.1 Tagy

Tagy jsou uzavřeny do špičatých závorek <>. Vše v těchto závorkách nazýváme tagem. Co je uvedeno mimo tagy, je vlastní obsah webové stránky. Tagy dělíme na **párové** a **nepárové**. V dalších kapitolách budeme používat pouze párové tagy. O tom až později.

### 3.1.1.1 Párové tagy

Párové tagy se skládají ze dvou částí: počátečního a koncového tagu. Mezi ně je vložen vlastní text.

**Příklad:**

```
<U>Použili jsme podtržené písmo</U>
```

**Vysvětlení:**

<U> je „otevírací“ tag. Po něm následuje vlastní obsah. Na závěr je uvede „uzavírací“ tag </U>. V prohlížeči by se zobrazil následující nápis: **Použili jsme podtržené písmo**

Tagy lze do sebe vnořovat.

**Příklad:**

```
<U><B>Použili jsme tučný a podtržený řez
písma</B></U>
```

**Vysvětlení:**

<U><B> jsou „otevírací“ tagy. Pak následuje vlastní obsah. Na závěr jsou uvedeny „uzavírací“ tagy </B></U>. V prohlížeči by se zobrazil následující nápis: **Použili jsme tučný a podtržený řez písma**

#### 3.1.1.2 Nepárové tagy

Nepárové tagy obsahují pouze počáteční značku. Popis daného elementu je pak vložen přímo mezi hranaté závorky <>. Vše, co je vloženo mezi tyto závorky a není chráněným slovem jazyka HTML nebo číselnou hodnotou parametru tohoto chráněného slova, musí být uvedeno v uvozovkách.

**Příklad:**

```
<IMG SRC="vojta.gif">
```

**Vysvětlení:**

Zde není „uzavírací“ tag. V prohlížení se zobrazí obrázek vojta.gif.

### 3.1.2 Atributy

V jazyce HTML se atributy velmi často používají. Slouží k předávání doplňkových informací o tagu. Definují nebo mění činnost tagu s kterým jsou spojeny. Zapisují se pouze do počátečního tagu.

**Příklad:**

```
<IMG SRC="vojta.gif" WIDTH="80" HEIGHT="120">
```

**Vysvětlení:**

Na začátku je uveden „otevírací“ nepárový tag, po něm následuje název obrázku, který se má zobrazit (vojta.gif). Na závěr jsou zobrazeny atributy WIDTH a HEIGHT. Atributy nastavují šířku (WIDTH) a výšku (HEIGHT) obrázku vojta.gif. Typů atributů je daleko více. Pro demonstraci jak atributy fungují nám to stačí.

#### 3.1.2.1 Popis atributu

Názvem tagu je většinou slovo, které dobře vystihuje význam a funkci tohoto tagu. Veškeré další výrazy v zápisu tagu jsou nepovinné atributy, s nimiž bývá spojena jejich hodnota, oddělená od názvu atributu znaménkem rovná se (=).

V jednom tagu se může vyskytovat i více atributů, které jsou navzájem odděleny jedním či více znaky tabulátoru, mezery, či nového řádku. Každý atribut má svoji hodnotu uzavřenou mezi uvozovkami, kromě číselných hodnot těchto atributů (přesněji řečeno, každý atribut by měl mít svoji hodnotu v uvozovkách, ale záleží na autorovi stránek, co do uvozovek zapíše, a co nikoli. Prohlížeč to zobrazí v pořádku). V jazyce HTML je však možné uvést jen název atributu bez jeho hodnoty v případě, že název i hodnota jsou identické. Tomuto se říká **minimalizace.**

### 3.1.3 Odkazy

Odkaz (hypertext) odkazuje na jiná data, která nemusí být a většinou ani nejsou v aktuální webové stránce. Druhy odkazů:

- odkaz na jiný web,
- odkaz na další webovou stránku,
- odkaz multimediální soubor,
- odkaz na část dokumentu a mnoho dalšího.

**Příklad:**

```
<A HREF="#B">Odkaz na pojmenované místo v
dokumentu</A>
<A HREF="http://www.pf.jcu.cz">Odkaz na WWW server
školy</A>
<A HREF="obsah.htm">Odkaz na jiný  dokument</A>
```

**Vysvětlení:**

Odkaz je párový tag. Myslím si, že tyto příklady jsou natolik jednoduché, že je nemusím popisovat.

## 3.2 Kódování

Myslím si, že je ještě nutné vysvětlit základní principy kódování. Pro české znaky bychom měli používat kódování **ISO 8859-2** (bývá používáno v unixových systémech) nebo kódování **windows-1250**, které se používá pro české texty ve Windows. Mezi nejznámější znakové sady patří 16bitová sada Unicode a 32bitová sada ISO 10646. Tyto sady obsahují znaky všech běžně používaných jazyků. Název kódovacího schématu musí být uzavřen do dvojitých uvozovek a popsán znakovou sadou Latin.

V následující tabulce jsou uvedeny v současnosti používané typy kódování:

| Schéma kódování | Počet bitů | Význam |
|---|---|---|
| EUC-JP | 8 | Japonština (používá vícebajtové kódování) |
| ISO-10646 | 32 | 32 bitová rozšířená sada, obsahuje Unicode jako podsadu |
| ISO-2022-JP | 7 | Japonština (používá vícebajtové kódování; pro poštu a konference) |
| ISO-8859-1 | 8 | Latinská abeceda č. 1 (západní Evropa, Latinská Amerika) |

| Schéma kódování | Počet bitů | Význam |
|---|---|---|
| ISO-8859-2 | 8 | Latinská abeceda č. 2 (střední a východní Evropa) |
| ISO-8859-3 | 8 | Latinská abeceda č. 3 (jihovýchodní Evropa a okolí) |
| ISO-8859-4 | 8 | Latinská abeceda č. 4 (Skandinávie, Británie) |
| ISO-8859-5 | 8 | Latina, cyrilice |
| ISO-8859-6 | 8 | Latina, arabština |
| ISO-8859-7 | 8 | 8 bit, Latina, řečtina |
| ISO-8859-8 | 8 | Latina, hebrejština |
| ISO-8859-9 | 8 | Latina, turečtina |
| ISO-8859-10 | 8 | Laponština, laponština, severština, eskymáčtina |
| Shift_JIS | 8 | Japonština (používá vícebajtové kódování) |
| UCS-2 | 16 | Canonical Unicode |
| UCS-4 | 32 | Canonical Unicode |
| UTF-16 | 16, 32 | Unicode Transformation,<br>vypouští 32 bitové znaky |
| UTF-7 | 7 | Unicode Transformation – 8 bitová |
| UTF-8 | 8 | Unicode Transformation – 8bitová |
| Windows-1250 | 8 | Nejčastěji používané kódování – kódová stránka Windows (cp-1250) – střední a východní Evropa |

Zdroj: Tvorba WWW a Wap (Petr Pexa)

## 3.3 Základní části dokumentu

Po vysvětlení základních pojmů přejdeme k základní „kostře“ webové stránky bez kterých se žádná správně napsaná stránka neobejde.

**Příklad:**

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//CZ">
<HTML>
<HEAD>
<META HTTP-EQUIV="Content-Type" CONTENT="text/html;
charset=windows-1250">
<TITLE>titulek dokumentu</TITLE>
</HEAD>
<BODY>
</BODY>
</HTML>
```

**Vysvětlení:**

**<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//CZ">** - určuje typ DTD a místo, kde jej nalezneme a verzi HTML

**<HTML>** - začátek HTML dokumentu

**<HEAD>** - začátek hlavičky, vymezuje záhlaví dokumentu, slouží pro potřeby prohlížeče

**<META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=windows-1250"> -** zde nastavíme kódování češtiny

**<TITLE>titulek dokumentu</TITLE>** - titulek dokumentu

**</HEAD> -** konec hlavičky

**<BODY>** - začátek těla dokumentu

**</BODY>** - konec těla dokumentu

**</HTML> -** konec dokumentu HTML

### 3.4 Jednoduchý příklad HTML stránky

V minulém příkladě jsme si ukázali nosnou „kostru“ každého dokumentu napsaného v jazyce HTML. V této si ukážeme jednoduchý HTML dokument ve kterém budou shrnuty prvky, které jsme doposud použili. V následujících kapitolách Vás provedu základy jazyka XHTML 2.0.

**Příklad:**

```
<! DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//CZ">
<HTML>
<HEAD>
  <TITLE>První HTML dokument</TITLE>
  <META HTTP-EQUIV="Content-Type"
CONTENT="text/html; charset=windows-1250">
</HEAD>
<BODY>
  <P> Toto je první odstavec. Tady začíná tělo
dokumentu</P>
  <P> Toto je druhý odstavec</P>
</BODY>
</HTML>
```

# 4 Principy jazyka XHTML

### 4.1 Obecný popis

Jazyk XHTML si vzal něco od HTML i XML. Naskytla se nám možnost napsat HTML dokument tak, aby vyhovoval přísnějším syntaktickým pravidlům platným pro XML dokumenty. Tak vznikl standard jazyka XHTML. Pokud v něm budeme psát naše dokumenty, budeme mít jistotu, že budou vyhovovat jak standardu XML, tak i HTML. Uvedu zde pouze základní „kostru“ v jazyku XHTML 1.0., abychom si uvědomili, provázání jazyků.

## 4.2 Struktura dokumentu

XHTML dokument je strukturován podobně jak jsme zvyklý z XML či HTML. Rozdíl je však v tom, že u HTML nemusí být na začátku dokumentu uveden tag <!DOCTYPE>, kdežto v XHTML je to nezbytné. Podívejme se na strukturu XHTML dokumentu.

**Příklad:**

```
<!DOCTYPE . . .>
<html ...>
<head> ...</head>
<body> ...</body>
</html>
```

Vytváření dokumentu v jazyce XHTML je obdobné jako vytváření běžného HTML dokumentu. Můžeme pracovat v textovém editoru, avšak nesmíme zapomenout doplňovat k jednotlivým částem obsahu dokumentu elementy ve správném pořadí určeném v DTD. Výsledek můžeme zobrazit pomocí klasického oblíbeného prohlížeče. Na začátku dokumentu musíme uvést deklaraci šablony, která uvádí definici DTD, pomocí níž je dokument vytvořen a definuje prostor jmen (namespace) dokumentu.

Při vytváření XHTML dokumentu bychom neměli opomenout následující kritéria:

- Dokument musí být platný na základě jedné ze tří přičleněných DTD.
- Kořenovým elementem dokumentu musí být <html>
- V kořenovém elementu dokumentu musí být určen prostor jmen (namespace), který používají xmlns atributy. Pro XHTML je definován takto: http://www.w3.org/1999/xhtml
- Deklarace DOCTYPE musí být uvedena ještě před kořenovým elementem a musí obsahovat jednu ze tří externích DTD (určena pomocí veřejných identifikátorů)
- V hlavičce dokumentu musí být na prvním místě uveden element <title>.

## 4.3 Jednoduchý XHTML dokument

Na tomto místě uvedu příklad jednoduchého XHTML dokumentu. Bude mít všechny potřebné části jako je prolog, deklarace DTD a vlastní obsah dokumentu. Naše ukázka začíná XML deklarací za níž následuje deklarace DTD. Pokud se podíváme pozorně, zbytek dokumentu je stejný jako HTML dokument.

Připomeňme si ještě jednou, co znamenají jednotlivé části tohoto dokumentu.

**Příklad:**

```
<?xml version="1.0"?>
<!DOCTYPE html
     PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
     "DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml"
lang="cs">
  <head>
    <title>První dokument v XHTML</title>
  </head>
  <body>
  </body>
</html>
```

**Vysvětlení:**

Jak vidíme, v prologu je uvedena pouze verze použitého XML, zbylé dva nepovinné atributy (encoding a standalone) uvedeny nejsou. V dokumentu bude tedy použito implicitní kódování UTF-8 nebo UTF-16.

Další důležitou částí, která nesmí u XHTML dokumentu chybět, je deklarace DTD. Z předchozích kapitol již víme, jak taková deklarace vypadá. V našem případě používáme externí DTD, která nemá lokální použití. Veřejný identifikátor, který není zaregistrován, nám dále říká, že vlastníkem takto označeného souboru je konsorcium W3C. Dále uvádí, že soubor je typu DTD, je označen jako XHTML 1.0 Strict a je napsán v anglickém jazyce. Za tímto veřejným identifikátorem následuje v

uvozovkách ještě napsaná URL, tj. adresa, na které nalezneme daný typ DTD (DTD Strict) v případě nerozpoznání tohoto veřejného identifikátoru.

Kořenovým elementem našeho dokumentu je element <html>. V tomto elementu je uveden atribut xmlns, který určuje prostor jmen dokumentu. Pak již následují části, které dobře známe z HTML.

## *4.4 DTD – možnosti pro XHTML*

Tuto kapitolu jsem zařadil zcela záměrně. DTD je v XHTML 1.0 novinkou, a proto je třeba se s ním seznámit. Každý dokument psaný v XHTML by měl mít na první řádce uveden typ DTD. Tato deklarace začíná slovem !DOCTYPE a obsahuje informace o dokumentu a o tom, kde nalezneme DTD. S tím jsme se již setkali u HTML dokumentů, kde máme na výběr ze tří DTD definic. Podobnou deklaraci jsme již také viděli u XML. Jediný rozdíl byl v tom, že u XML máme na výběr z mnoha různých DTD nebo ho můžeme nadefinovat sami.

U XHTML je tomu trošku jinak. Zde máme možnost vybrat si jeden ze tří typů DTD. O tom, který kdy použít se zmíním o něco níže. Také nezapomenu uvést jejich přesnou deklaraci, ale samozřejmě ji naleznete i na adrese W3C konsorcia: http://www.w3c.org.

Když budeme vytvářet vlastní XHTML dokument, musíme si nejprve zvolit jednu ze tří níže zmiňovaných definic DTD, a pak vytvořit konkrétní dokument podle elementů a pravidel této zvolené definice. Tak se pojďme podívat jak vypadají zmiňované typy DTD.

### 4.4.1 DTD Strict

Strict (striktní) definice je určena pro stránky, jejichž autoři se rozhodli zbavit je všech pozůstatků dob minulých. Jak víme, je možno různé věci zapsat různým způsobem a autoři stránek s tímto typem DTD se rozhodli ze všech možných způsobů použít jen ten nejnovější. Takovéto stránky neobsahují žádné nedoporučované elementy jazyka HTML 4.01 (to znamená tagy ani atributy). Chybí elementy a atributy, které řídí prezentaci a vzhled dokumentu jako například tag  a u některých tagů nejsou

podporovány atributy font a align. Vzhled stránky je upraven pouze pomocí kaskádových stylů (CSS – Cascading Style Sheets).

Potíže mohou nastat se zobrazováním dokumentů, protože dnešní prohlížeče plně podporují všechny nedoporučované elementy a zároveň nejsou ještě schopny plně implementovat nové standardní prvky. Zdá se, že zatím jedinou opravdovou výhodou striktní definice DTD jazyka XHTML je plná podpora takovýchto dokumentů v budoucích verzích XHTML.

**Příklad:**

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

### 4.4.2 DTD Transitional

Druhým typem je tzv. přechodová definice DTD. Ta se nejvíce přibližuje současnému standardu jazyka HTML.

Stránky s tímto typem DTD jsou velmi běžné, neboť používají vše na co jsme z HTML zvyklý. Znamená to, že můžeme používat i starší možnosti zápisu, které vyhovují standardu. Můžeme tedy bez problémů využívat všechny elementy a atributy z HTML 4.0. Nepoužívají se však rámy (frames).

**Příklad:**

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

### 4.4.3 DTD Frameset

Třetí definice DTD slouží pro rámy (frames). Odtud patrně název Frameset. Je totožná s přechodovou definicí DTD. Jediným rozdílem je, že v této definici navíc nalezneme rámy.

**Příklad:**

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

# 5 Principy jazyka XHTML 2.0

## *5.1 Obecný popis*

Podle definice zveřejněné konsorciem X3C je XHTML 2 značkovací jazyk určený k publikování dokumentů na internetu. Vzhledem k tomu se nesnaží být úplně univerzální, neklade si za cíl pracovat se všemi možnými značkami, které mnohdy nemají se standardem nic společného. Naproti tomu definuje řadu užitečných elementů, které mohou být rozšiřovány pomocí tagů (značek) ,  nebo v kombinaci s kaskádovými styly. XHTML 2 obsahuje jen základní formátovací moduly. XHTML 2 se vrací k původní myšlence a to oprostit text od grafických elementů. Ty mají být realizovány pomocí CSS (*Cascading Style Sheets* – kaskádové styly). XHTML 2.0 převzalo značky používané v XHTML 1.1 a HTML 4. Dále k ním přibylo několik nových značek, které rozšiřují možnosti jazyka XHTML 2.

### 5.1.1 Upozornění a zpětná kompatibilita

Jak z mé práce vyplývá, značkovací jazyky se stále vyvíjejí. Než se jedna verze stihne zabydlet, hned je na světě další (dokonalejší) verze jazyka. Upozorňuji na to, že co budete číst v následujících řádcích může být za pár týdnů opět novelizováno. XHTML 2 je stále ještě v „plenkách“, a proto současné prohlížeče neumí zobrazit nové elementy přidané do XHTML 2. V současné době se čeká na novou verzi Internet Explorer 7 od firmy Microsoft.

Starší verze HTML byly navrženy tak, aby byly zpětně kompatibilní s verzí předcházející. Mnoho nových dokumentů bylo prohlíženo ve starých prohlížečích. V jazyku XHTML 2 bylo nejprve zamýšleno zrušit zpětnou

kompatibilitu. Po velké diskusi se konsorcium W3C nakonec rozhodlo zpětnou kompatibilitu prozatím ponechat.

## 5.2 Požadavky na XHTML 2

Z toho co jsem zde uvedl zformulujeme základní požadavky, které musí XHTML 2 splňovat:

- **méně vzhledu, více struktury** – dokument se tak stává přehlednějším, srozumitelnějším,
- **větší použitelnost** – snaha o vytvoření jazyka, který bude snadný pro každého,
- **větší srozumitelnost** – návrh jazyka by měl být co nejjednodušší a nejdostupnější.
- **nezávislost na výstupním zařízení** – snaha o to, abychom vytvořili jednu stránku a mohli si ji prohlédnout mobilem, televizí, PDA a samozřejmě počítačem,
- **méně skriptování** - zjednodušit dosavadní skripty tím, že do základních značek zahrneme nejčastěji používané funkce.

## 5.3 Základní struktura dokumentu v XHTML 2

Nastal pravý čas ukázat si, jak má vypadat dokument napsaný v jazyku XHTML 2. Většinu programátorů nepřekvapí, že struktura základního dokumentu v XHTML 2 je podobná struktuře v jazyku XHTML 1.1.

**Příklad:**

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 2.0//EN"
"TBD">
<html xmlns="http://www.w3.org/2002/06/xhtml2"
xml:lang="en">
<head>
<title> </title>
```

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

**Vysvětlení:**

**<?xml version="1.0" encoding="UTF-8"?>** - Zde nic měnit nemusíme oproti XHTML 1.1.

**<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 2.0//EN" "TBD">** - Změníme pouze typ dokumentu, aby bylo jasné, že se jedná o dokument vytvořený v XHTML 2.

**<html xmlns="http://www.w3.org/2002/06/xhtml2" xml:lang="en">** - Prostor jmen musíme uvést za značkou <html> u atributu xmlns.

Význam zbylých tagů je stejný jako v HTML a XHTML 1.1.

## 5.4 Moduly

XHTML 2 člen rodiny značkovacích jazyků. Skládá se, stejně jako jeho předchůdce z celé řady modulů, které společně popisují kompletní sadu elementů a atributů daného jazyka.

XHTML 2 aktualizuje celou řadu modulů definovaných v XHTML 1.1. Zatímco předchozí verze XHTML 1.1 používaly sémantiku převzatou z HTML 4. XHTML 2 ji má definovanou přímo ve své specifikaci.

Následující tabulka ukazuje přehled všech modulů definovaných v XHTML 2:

| Modul | Dostupné značky a atributy |
|---|---|
| Struktura | body, head, html, title |
| Text | abbr, address, blockquote, cite, code, dfn, div, em, h, hr, kbd, I, p, pre, quote, samp, section, span, strong, sub, sub, var |

| Modul | Dostupné značky a atributy |
|---|---|
| Hypertext | a |
| Seznamy | dl, dt, dd, label, nl, ol, ul, li |
| Odkazy | link |
| Metainformace | meta |
| Objekt | object, param |
| Skripty | noscript, skript |
| Styly | style |
| Tabulky | caption, col, colgroup, table, tbody, td, tfoot, th, these, tr |
| XForm | action, alert, bind, case, choices, copy, delete, dispatch, extension, filename, group, help, hint, input, insert, instance, item, temset, label, load, mediatype, message, model, output, range, rebuild, recalculate, refresh, repeat, reset, revalidate, secret, select, select1, send, setfocus, setindex, setvalue, submission, submit, switch, textarea, toggle, trigger, upload, value |

Zdroj: http://www.w3.org/TR/2003/WD-xhtml2-20030506

V následujících kapitolách si rozebereme jednotlivé moduly.

### 5.4.1 Modul struktura

#### 5.4.1.1 Tag body

Základní tag každého dokumentu, který byl vytvořen značkovacími jazyky. Jeho význam byl již vysvětlen, proto se jím tu nebudu již zabývat.

#### 5.4.1.2 Tag head

Tag head patří mezi základní stavební kámen dokumentu. Obsahuje informace o zpracovávaném dokumentu (nastavení kódování, informace o autorovi atd.), jako jsou tagy <title>,<meta>. V následujícím příkladě ukáži jaké všechny tagy může obsahovat tag head.

**Příklad:**

```
<head>
<title>Výuka jazyka XHTML v předmětu WWW1</title>
</head>
```

**Vysvětlení:**

**<head></head>** - Párový tag.

**<title>Výuka jazyka XHTML v předmětu WWW1</title>** - V záhlaví prohlížeče bude napsáno: „Výuka jazyka XHTML v předmětu WWW1“.

#### 5.4.1.3 Tag title

Tag title zajistí, že v záhlaví prohlížeče bude titulek dokumentu. Myslím si, že příklad uvedený v minulé kapitole je dostačující, a proto se jím již nebudu zabývat.

### 5.4.2 Modul Text

#### 5.4.2.1 Tag abbr

Tento tag signalizuje, že text v tomto tagu je zkratka. Význam ukáži na příkladu.

**Příklad:**

```
<abbr title="Limited">Ltd.</abbr>
<abbr title="Abbreviation">abbr.</abbr>
```

**Vysvětlení:**

Po přejetí myší nad zkratkou Ltd. se nám zobrazí vysvětlení zkratky (Limited). V současné době zatím není tento tag prohlížeči podporován.

#### 5.4.2.2 Tag address

Text uvedený v tagu <address> může obsahovat v textu další elementy, s výjimkou druhé značky <address>. Vzhledovou část tag můžeme doupravit použitím CSS stylů.

Tag <address > může obsahovat atribut dir, který informuje prohlížeč o směru v jakém se má segment textu zapsaného ve značce <address> zobrazit. Další atribut je identifikátor id. Uvedl jsem pouze základní atributy. Ve skutečnosti existují další atributy a to především ve spojitosti s jazykem JavaScript.

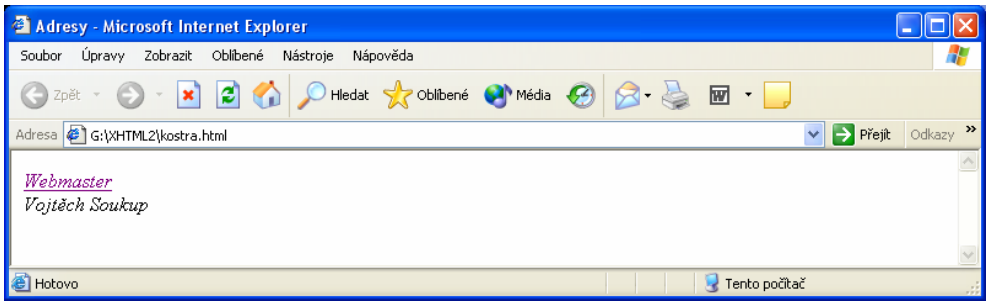
Většina dokumentů by měla mít uvedenu adresu autora, ve které by měl uživatel najít nejméně e-mailovou adresu autora.

**Příklad:**

```
<address>
<a
href="mailto:soukup.vojta@tiscali.cz">Webmaster</a>
<br />
Vojtěch Soukup <br />
</address>
```

**Vysvětlení:**

Tímto způsobem se ve zdrojových dokumentech často uvádí odkaz na správce webu (obrázek 1).




**Obrázek 1:** Správné uvedení odkazu na správce webu.

### 5.4.2.3 Tag blockquote

Pomocí značky <blockquote> se často z hlavního textu vyčleňují delší citace z jiných zdrojů.

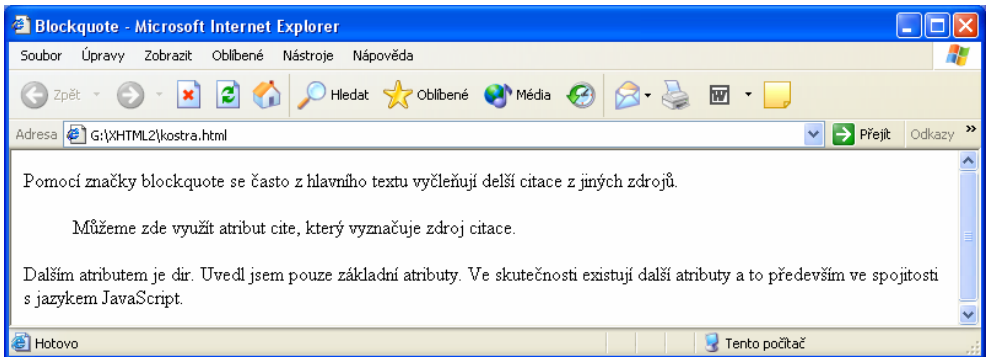
Můžeme zde využít atribut cite, který vyznačuje zdroj citace. Dalším atributem je dir. Uvedl jsem pouze základní atributy. Ve skutečnosti existují další atributy a to především ve spojitosti s jazykem JavaScript.

**Příklad:**

```
Pomocí značky blockquote se často z hlavního textu
vyčleňují delší citace z jiných zdrojů.
<blockquote cite="http://vojtechsoukup.wz.cz">
Můžeme zde využít atribut cite, který vyznačuje
zdroj citace.
</blockquote>
```

**Vysvětlení:**

Vše je vidět na následujícím obrázku.




**Obrázek 2:** Použití tagu <blockquote>

### 5.4.2.4 Tag cite

Tag <cite> označuje text, který je citací (název knihy, časopisu, jméno autora, ovšem ne nic jiného). Mohlo by se zdát, že tag <cite> je zbytečný, ale není tomu tak. Tag <cite> umožňuje autorovi i ostatním uživatelům vytvářet automaticky z dokumentu seznam použité literatury.

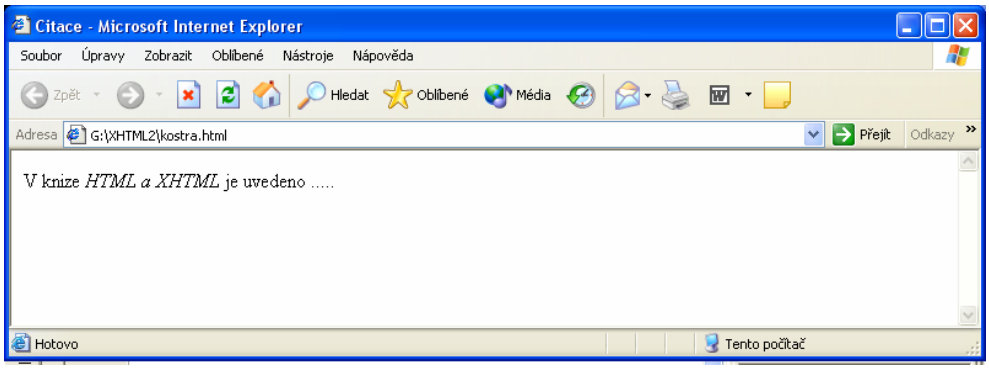
Tag <cite> má též atribut cite. Jeho parametrem je adresa, odkud byla citace čerpána. V současné době žádný z prohlížečů tento atribut nepodporuje. V budoucnosti po přejetí myší nad tagem <cite> se objeví adresa, odkud byla citace čerpána. Uvedu dva příklady. První je prohlížeči podporován (obrázek 3), ovšem u druhého příkladu není podpora atributu citace.

**Příklad 1:**

```
V knize <cite>HTML a XHTML</cite> je uvedeno .....
```

**Příklad 2:**

```
V knize <cite cite="http://vojtechsoukup.wz.cz">
HTML a XHTML</cite> je uvedeno .....
```



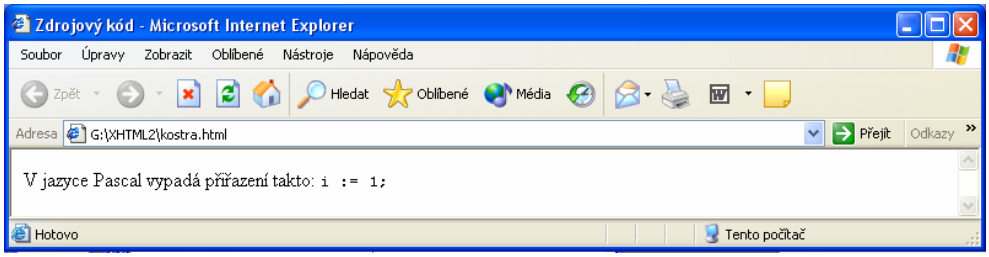

**Obrázek 3:** Správné použití citace.

### 5.4.2.5 Tag code

Tag code nám slouží k oddělení zdrojových kódů od ostatního textu.

**Příklad:**

```
V jazyce Pascal vypadá přiřazení takto:
<code>i := 1;</code>
```



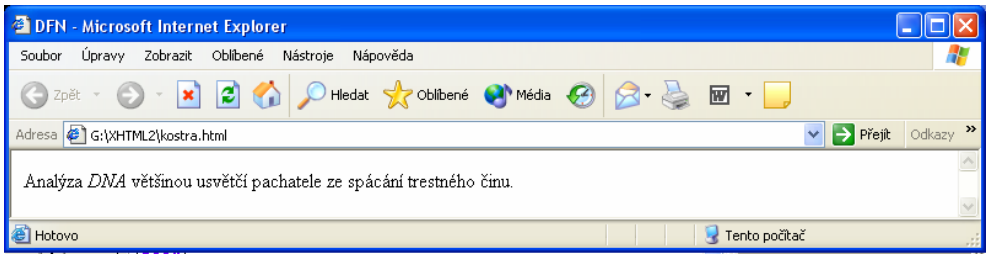

**Obrázek 4:** Tag code použijeme pouze na zdrojový kód.

### 5.4.2.6 Tag dfn

Tag <dfn> označuje první výskyt speciálního pojmu. Jak se ukáže později (obrázek 5), je to obrovská výhoda při sestavování rejstříku dokumentu.

**Příklad:**

```
 Analýza <dfn>DNA</dfn> většinou usvětčí pachatele
ze spácání trestného činu.
```



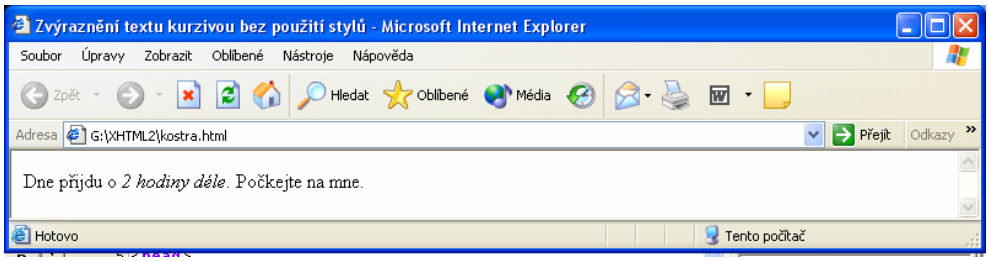

**Obrázek 5:** S použitím tagu <dfn> je sestavení rejstříku snadnou záležitostí.

### 5.4.2.7 Tag em

Text uvedený v tagu  je většinou zobrazen kurzivou. Tag  je výhodné používat nejen k zvýraznění textu, ale i k definici nových pojmů.

**Příklad:**

```
Dne přijdu o <em>2 hodiny déle</em>. Počkejte na
mne.
```




**Obrázek 6:** Použití tagu .

### 5.4.2.8 Tag div

Tag  vytvoří oddíl v dokumentu. Použijeme atribut class, který slouží k přilinkování CSS stylů.

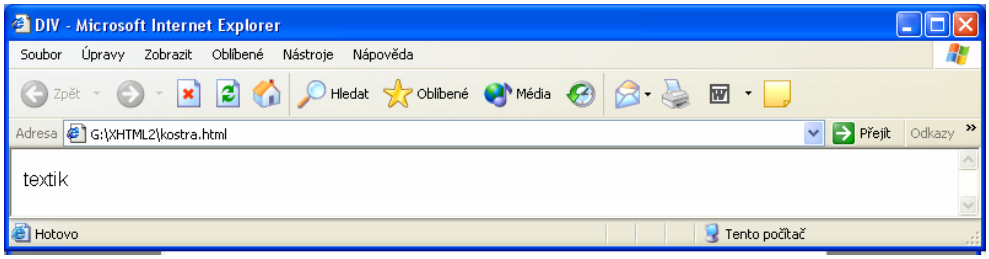
**Příklad:**

```
<head>
<style type="text/css">
.hel {font-family:helvetica}
</style>
</head>
```

```
<body>
<div class="hel">textik</div>
</body>
```

**Vysvětlení:**

Slovo textík (obrázek 7) bude napsáno pomocí písma Heltvica díky atributu class, který zajišťuje komunikaci s CSS styly. Rozdíl v písmu je vidět na následujícím obrázku.




**Obrázek 7:** Správné použití tagu .

### 5.4.2.9 Tag h

Čtení textu zobrazeného na obrazovce není nikdy jednoduché. Dlouhý text, který není rozdělen žádným titulkem, je velmi nepřehledný. K lepší orientaci v textu přispívá tag <h>.

Grafickou část nadpisů nastavíme pomocí CSS stylů. Jejich implementaci v XHTML 2 zajistíme pomocí atributů class nebo id.

V XHTML 2 přišlo konsorcium W3C s revolučním řešení tohoto tagu. Dříve se dělil na h1, h2, h3, h4, h5, h6. Nyní je to jiné. Záležet bude pouze na hloubce zanoření tagu <h> v kombinaci s tagem <section>. Tag <section> rozdělí text na části. V každé této části budou nadpisy stejné úrovně. Vypadá to velmi složitě, ovšem v principu je používání tohoto tagu velmi jednoduché a elegantní. Uvedu dva příklady. V prvním příkladu ukáži tvorbu nadpisů starým způsobem. Ve druhém příkladě uvedu ten samý příklad ovšem dle syntaxe XHTML 2.0.

**Příklad 1:**

```
<h1>Nadpis h1</h1>
```

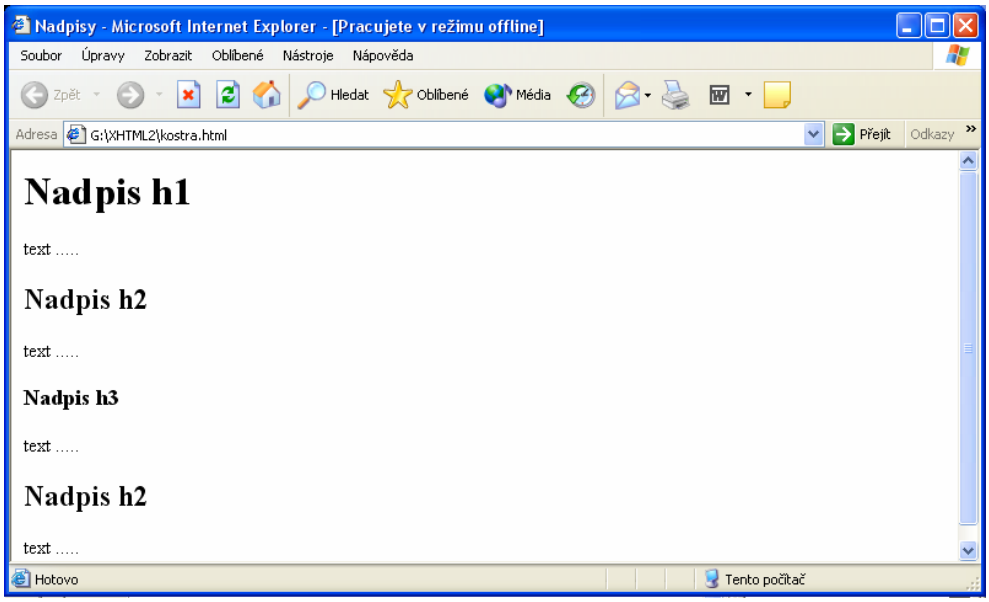
```
   <p> text .....</p>
<h2> Nadpis h2</h2>
   <p> text .....</p>
<h3> Nadpis h3</h3>
   <p> text ..... </p>
<h2> Nadpis h2</h2>
   <p> text .....</p>
```

**Příklad 2:**

```
<section>
      <h>Nadpis h1</h>
      <p> text ..... </p>
      <section>
         <h> Nadpis h2</h>
         <p> text .....</p>
         <section>
            <h> Nadpis h3</h>
            <p> text .....</p>
         </section>
      </section>
      <section>
         <h> Nadpis odpovídající h2 </h>
         <p>text .....</p>
      </section>
</section>
```

**Vysvětlení:**

Uvedl jsem záměrně jeden delší příklad, aby bylo názorně vidět rozdíl v syntaxi jazyka. Na následujícím obrázku bude zobrazen příklad 1. Příklad 2 vypadá úplně stejně, ale v současné době jej nedokáží prohlížeče zobrazit.




**Obrázek 8:** Zobrazení nadpisů.

### 5.4.2.10 Tag hr

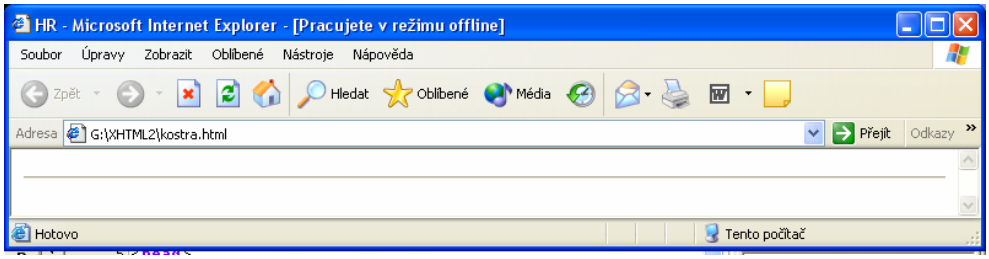
Tag <hr> vkládá do dokumentu přes celé zobrazované okno vododorovnou čáru. Tag <hr> si vynutí prosté zalomení řádku. Navíc způsobí změnu zarovnání odstavce na implicitní (zarovnání k levému okraji). Tag <hr> je původně nepárovým tagem, proto jej musíme ukončit mezerou a lomítkem (/).

Konsorcium W3C zatím nerozhodlo zda bude tag <hr> vypuštěn nebo přejmenován na tag <separator>. Zatím můžeme používat tag <hr>.

Tag <hr> má celou řadu atributů. V XHTML 2 je nejlépe používat atribut class, díky němuž můžeme nastavit grafickou část v CSS stylech. V příkladu uvedu pouze samotný tag <hr> bez využití CSS stylů.

**Příklad:**

```
<hr />
```



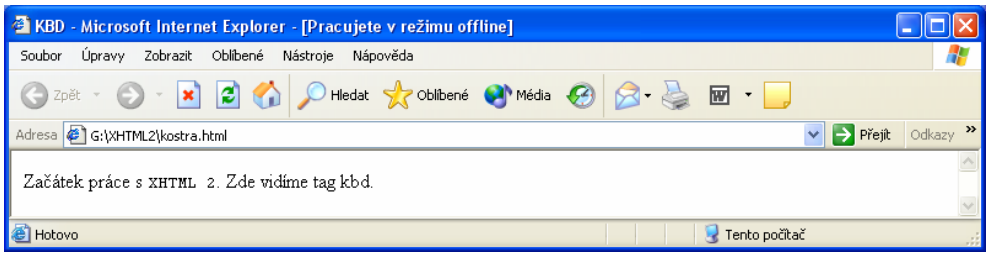
**Obrázek 9:** Použití oddělovače textu.

### 5.4.2.11 Tag kbd

Tag <kbd> slouží pro zvýraznění technických pojmů. Tag se nejčastěji používá v počítačové dokumentaci.

**Příklad:**

```
Začátek práce s <kbd>XHTML 2</kbd>. Zde vidíme tag kbd.
```




**Obrázek 10:** Tag <kbd> použijeme pouze pro odborné názvy.

### 5.4.2.12 Tag l

Tag <l> je novým tagem XHTML 2. Text v tomto tagu bude na samostatném řádku. V budoucnu nahradí tag <br> (zalomení řádky). Grafickou stránku tagu <l> upravíme pomocí CSS stylů, za použití atributu class. Bohužel v současných prohlížečích není tento tag podporován.

**Příklad:**

```
<p>
<l>program p (input, output);</l>
<l>begin</l>
<l> writeln("Hello world");</l>
<l>end.</l>
</p>
```

**Vysvětlení:**

V následujících řádcích Vám popíši jak by příklad měl vypadat ve skutečnosti:

```
program p (input, output);
begin
writeln(("Hello world");
end.
```

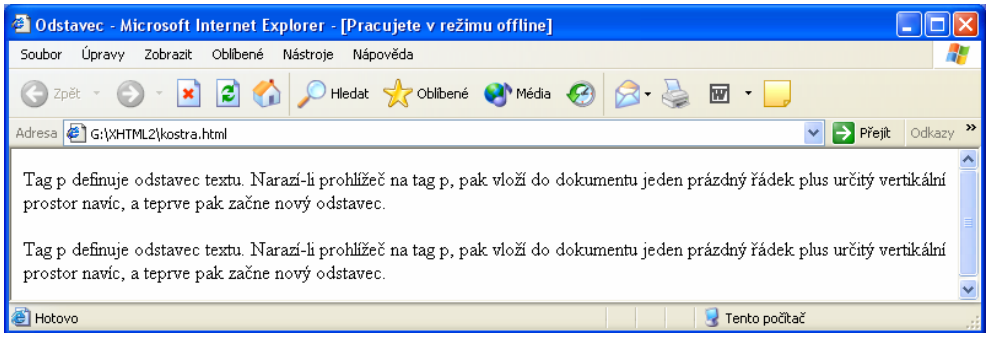
### 5.4.2.13 Tag p

Tag <p> definuje odstavec textu. Narazí-li prohlížeč na tag <p>, pak vloží do dokumentu jeden prázdný řádek plus určitý vertikální prostor navíc, a teprve pak začne nový odstavec. Poté prohlížeč vloží požadovaný text, obrázky atd. Prohlížeč se postará o délku řádků, lámání slov a zalamování řádků.

Pochopitelně i zde existují atributy. Konsorcium W3C nás tlačí k tomu, abychom používali pouze atribut class, který zajišťuje spojení s CSS styly. Dříve se používal atribut align (zarovnání). Ten je dnes zbytečný plně ho nahradily CSS styly.

**Příklad:**

```
<p> Tag p definuje odstavec textu. Narazí-li
prohlížeč na tag p, pak vloží do dokumentu jeden
prázdný řádek plus určitý vertikální prostor navíc,
a teprve pak začne nový odstavec. </p>
<p> Tag p definuje odstavec textu. Narazí-li
prohlížeč na tag p, pak vloží do dokumentu jeden
prázdný řádek plus určitý vertikální prostor navíc,
a teprve pak začne nový odstavec. </p>
```



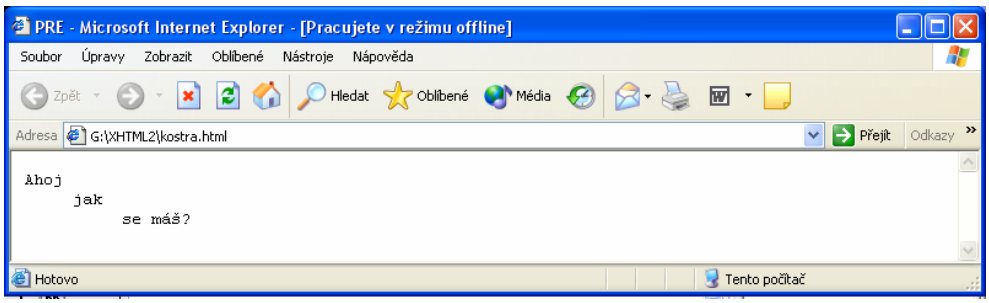

**Obrázek 11:** Odstavce.

### 5.4.2.14 Tag pre

Tag <pre> prohlížeč zobrazí přesně s tím prokládáním znaků a řádků, jaké je uvedeno ve zdrojovém dokumentu. Normální zalamování slov a zaplňování odstavců je tak v této značce zablokováno a prohlížeč dodržuje přesně i veškeré vedoucí a koncové mezery. Prohlížeč zobrazuje navíc veškerý text v neproporcionálním písmu. Tag <pre> je používán v situacích, kdy je třeba přesně zachovat strukturu textu. Vše si ukážeme na jednoduchém příkladu. Jako důkaz nám bude sloužit přiložený obrázek.

**Příklad:**

```
<pre>
Ahoj
     jak
               se máš?
</pre>
```




**Obrázek 12:** Tag <pre>.

### 5.4.2.15 Tag quote

Tag <quote> je nástupcem tagu <q> a je skoro totožný s tagem <blockquote>. Rozdíl mezi tagy spočívá v jejich prezentaci prohlížeči. Tag <quote> použijeme pro krátké citace, které jsou přímo vloženy do stejného řádku jako ostatní text.

Můžeme zde používat atribut cite, který vyznačuje zdroj citace. Hodnotu tohoto atributu tvoří adresa URL, uzavřená do uvozovek, která ukazuje na dokument. Můžeme použít i další atributy: dir, lang, class, title. Atribut dir informuje prohlížeč o směru, v jakém se má text zobrazit. Atribut lang určí jazyk použitý v rámci tagu. Atribut class aplikuje předem definovaný styl. Hodnotu atribut title zapíšeme do uvozovek. Bohužel tag <quote> je nový a v současné době ho prohlížeče nepodporují.

**Příklad:**

```
Pepa řekl: <quote
cite="http://vojtechsoukup.wz.cz">Zítra sem
nechoď.</quote>
```

**Vysvětlení:**

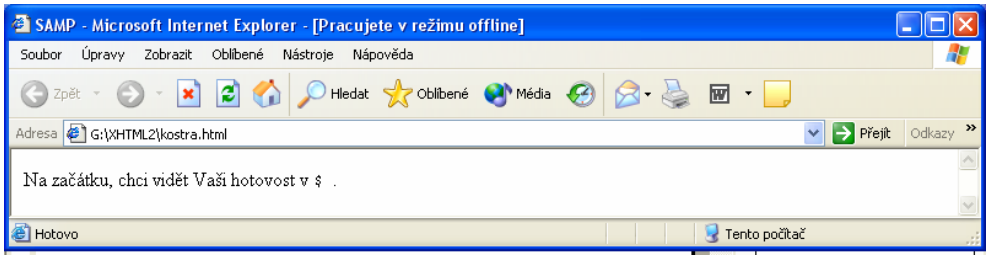
Text Pepa řekl: bude zobrazen standardním způsobem. Zbytek textu bude zobrazen kurzivou nebo neproporcionálně (záleží na typu prohlížeče).

### 5.4.2.16 Tag samp

Tag <samp> označuje text , který uživatel nemá interpretovat žádným jiným způsobem. Tag se nejčastěji používá při vyjmutí určité množiny znaků z normálního kontextu.

**Příklad:**

```
Na začátku, chci vidět Vaši hotovost v <samp>$
</samp>.
```




**Obrázek 13:** Tag <samp>.

### 5.4.2.17 Tag section

Tag <section> je novým tagem jazyka XHTML 2, který rozčleňuje text na části a připravuje tak text pro tvorbu nadpisů (použití tagu <h>). Nebudu zde uvádět žádný příklad, pokud si někdo nepamatuje implementaci tagu <section>, ať se podívá na implementaci tagu <h>.

### 5.4.2.18 Tag span

Tag  slouží ke změně vzhledu určité části obsahu značky (u textu). Prohlížeče pracují s tagem  podobně jako s běžnou značkou fyzického stylu. Jediný rozdíl spočívá v tom, že implicitní význam tagu  je ponechat text beze změny.

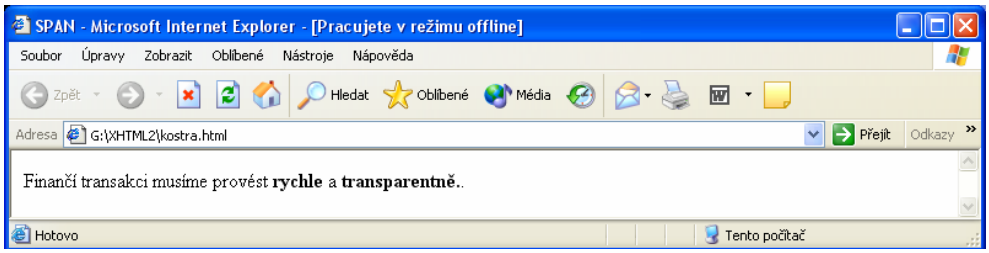
**Příklad:**

```
<head>
<title>SPAN</title>
<style type="text/css">
.xref {
      font-weight:bold;
      }
</style>
</head>
<body>
<p>Finančí transakci musíme provést <span
class="xref">rychle</span>
a <span class="xref">transparentně.</span>.</p>
```

```
</body>
```

**Vysvětlení:**

Do tagu <p> (odstavec) vložíme tag , který nám umožní, aby slova rychle, transparentně byla zobrazena tučně. Viz následující obrázek.



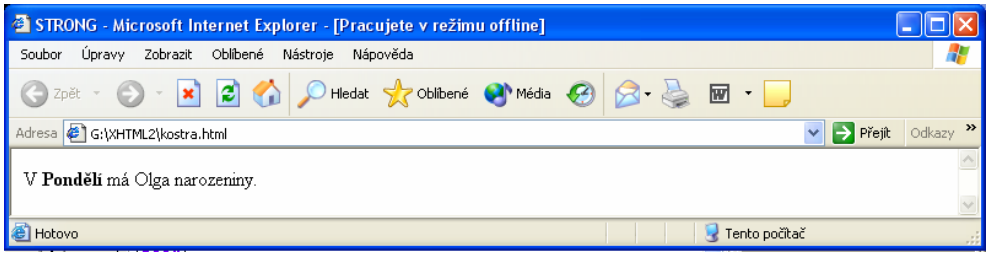

**Obrázek 14:** Vkládání CSS stylů.

### 5.4.2.19 Tag strong

Tag  zvýrazňuje text. Text je zobrazen tučně. Tento tag je třeba používat velmi obezřetně, protože působí velmi „vyzývavě“. V současné době nedošlo konsorcium W3C k dohodě, jestli tento tag nechat, nebo ho vypustit. Zaznívají názory, že tag  je plně nahraditelný CSS styly. Až čas ukáže, jak to s tímto tagem dopadne.

**Příklad:**

```
V <strong>Pondělí</strong> má Olga narozeniny.
```



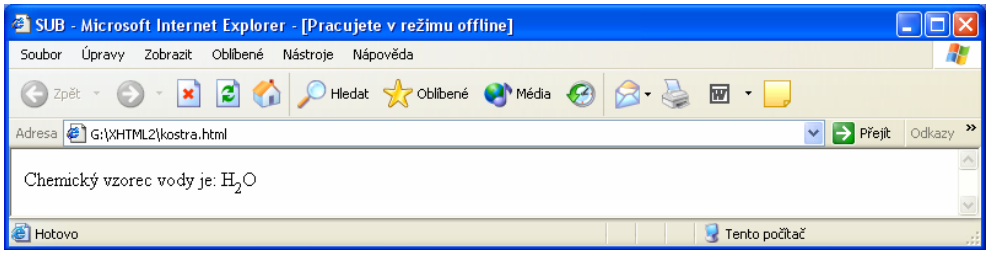

**Obrázek 15:** Tag .

### 5.4.2.20 Tag sub

Tag <sub> zobrazí příslušný text dolním indexem. Tag <sub> se využívá v matematickém zápisu a pro zobrazení chemických vzorců.

**Příklad:**

```
Chemický vzorec vody je: H<sub>2</sub>O
```



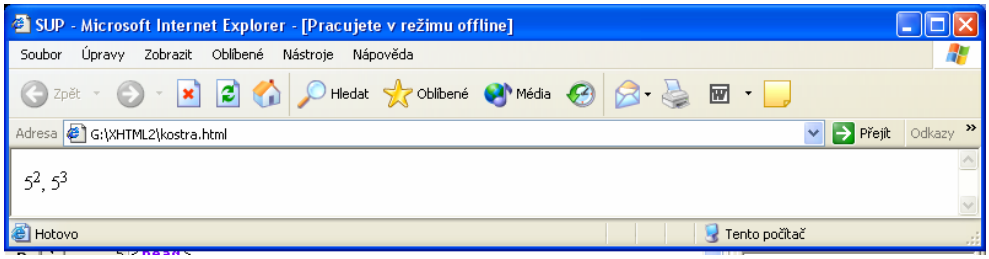

**Obrázek 16:** Dolní index.

### 5.4.2.21 Tag sup

Tag <sup> zobrazí text jako horní index. Tag je užitečný například pro zápis poznámek pod čarou dokumentů, a také pro vyjádření hodnoty exponentu v matematických vztazích.

**Příklad:**

```
5<sup>2</sup>, 5<sup>3</sup>
```



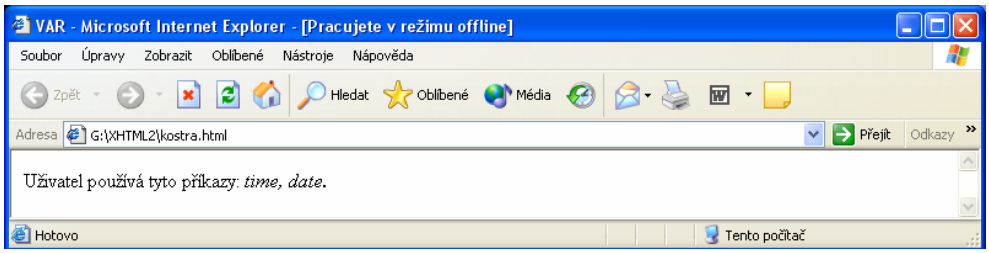

**Obrázek 16:** Horní index.

### 5.4.2.22 Tag var

Tag <var> je určený pro počítačovou dokumentaci. Text označený pomocí tagu <var> se zobrazuje zpravidla kurzívou.

**Příklad:**

```
Uživatel používá tyto příkazy: <var>time,
date.</var>
```




**Obrázek 17:** Tag <var>.

## 5.4.3 Modul Hypertext

Objevení hypertextu znamenal velký průlom v práci s dokumenty. Až do této doby jsme mohli pracovat pouze v jednom dokumentu. Hypertext spojuje minimálně dva, ale v dnešní době již celé řadě dokumentů mezi sebou. Hypertext je důležitým a nepostradatelným nástrojem, který „roztáčí“ celý web. Díky tomu se dostala písmenka „HT“ do názvu „HTML“ a jejich následovníků. Písmenka „HT“ znamenají již zmiňovaný hypertext. Základním tagem pro vytváření hypertextových odkazů je tag <a>.

### 5.4.3.1 Tag a

Tag <a> se používá ve spojitosti s atributy. Nejčastěji se používá atribut href. Ten vytvoří hypertextový odkaz na jiný dokument nebo na jiné místo v témže dokumentu. Atributu href přiřadíme URL adresu, která musí být v uvozovkách.

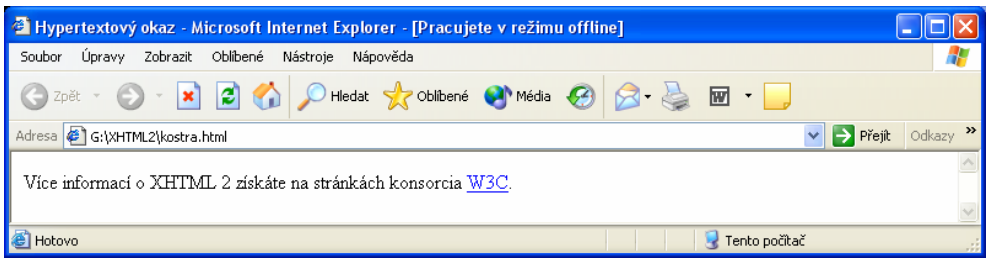
Druhým nejčastěji používaným atributem je id, kterým můžeme označit libovolnou část dokumentu. Atribut id nahradil atribut name.

#### *5.4.3.1.1 Standardní odkaz*

V následujícím příkladu si ukážeme tvorbu klasického odkazu na stránku: http://vojtechsoukup.wz.cz. Výsledek si pak můžete prohlédnout v následujícím obrázku (obrázek 18).

**Příklad:**

```
Více informací o XHTML 2 získáte na stránkách
konsorcia <a href="http://www.w3.org/">W3C</a>.
```




**Obrázek 18:** Hypertextový odkaz.

**Poznámka:**

Hypertextový odkaz poznáme podle toho, že text je implicitně modrý a podtržený. Po „přejetí“ myší nad touto částí textu (obrázku) se změní tvar z čárky na ruku. Barvu textu hypertextového odkazu můžeme pochopitelně měnit pomocí CSS stylů.

#### *5.4.3.1.2 Odkaz na jiný dokument*

S hypertextovým odkazem na web bychom nevystačili. Velmi důležité je umět vytvořit odkaz na jiný dokument.

**Příklad:**

```
Odkaz na "kostru" dokumentu <a
href="http://vojtechsoukup.wz.cz/kostra.html">
XHTML 2</a>.
```

#### *5.4.3.1.3 Odkaz pomocí atributu id*

Pomocí atributu id tagu <a> vytvoříme v dokumentu identifikátor fragmentu. Takto vytvořený identifikátor fragmentu se stává potenciálním cílem odkazu. Identifikátor fragmentu si vysvětlíme jako určitou analogii k příkazu goto. Hodnotu atributu id může být libovolný textový řetězec zapsaný do uvozovek. Řetězec musí tvořit jednoznačnou identifikaci a nesmí se znovu použít v žádném jiném atributu id ve stejném dokumentu (v jiných dokumentech se vyskytnout smí).

Než si uvedeme příklad, musím Vám ukázat typy (druhy) odkazů.

- **absolutní odkaz:** http://www.mycompany.com/one.html#anchor-one
- **relativní odkaz:** máme dvě možnosti: ./one.html#anchor-one nebo one.html#anchor-one

Tento odkaz by sám o sobě nebyl funkční. Je nutné v dokumentu one.html nadefinovat návěští #anchor-one. V následujícím příkladu si ukážeme jak sladit dosud nabyté poznatky o „návěštích“ hypertextových odkazů tak, aby fungovala.

**Příklad:**

```
<a id="začátek"></a>
Tento odkaz by sám o sobě nebyl funkční. Je nutné
v dokumentu one.html nadefinovat návěští #anchor-
one. V následujícím příkladu si ukážeme jak sladit
dosud nabyté poznatky o „návěštích“ hypertextových
odkazů tak, aby fungovala.
<a href="#začátek">Odkaz na návěští.(na konkrétní
místo téže stránky)</a>
```

**Vysvětlení:**

Odkaz na návěští si musíme vysvětlit zde obrázky nepomůžou. Po kliknutí na nápis **Odkaz na návěští** se přesuneme na začátek dokumentu a to díky tomu, že zde máme nadefinovanou „kotvu“ dokumentu **<a id="začátek"></a>**.

### *5.4.3.1.4 Návěští na jiný dokument pomocí id*

„Přesouvat“ se pouze v rámci jednoho dokumentu je málo, proto si ukážeme jak vytvořit odkaz na návěští v jiném dokumentu.

**Příklad:**

soubor **dva.html**

```
<a
href="http://vojtechsoukup.wz.cz/one.html#začátek">
Odkaz na návěští v jiném dokumentu. </a>
```

soubor **one.html**

```
<a id="začátek"></a>
Tento odkaz by sám o sobě nebyl funkční. Je nutné
v dokumentu one.html nadefinovat návěští #anchor-
one. V následujícím příkladu si ukážeme jak sladit
dosud nabyté poznatky o „návěštích" hypertextových
odkazů tak, aby fungovala.
```

**Vysvětlení:**

Založíme si dva soubory: **dva.html** a **one.html**. Do souboru **one.html** umístíme „kotvu" návěští **<a id="začátek"></a>**. Soubor **dva.html** musí obsahovat URL adresu za kterou umístíme # a název „kotvy" (ze souboru one.html).

## 5.4.4 Modul seznamy

XHTML 2 (i jeho předchůdci HTML, XHTML) nám díky seznamům umožňuje uspořádat informace v dokumentu přehledně. Seznamy nejsou nic neobvyklého či zásadního. Patří mezi nejelegantnější prvky pro rozčlenění textu.

### 5.4.4.1 Neuspořádané seznamy

Nejčastějším neuspořádaným seznamem na webu je kolekce hypertextových odkazů na jiné weby. Neurčujeme pořadí jednotlivých webů.

#### *5.4.4.1.1 Tag ul a li*

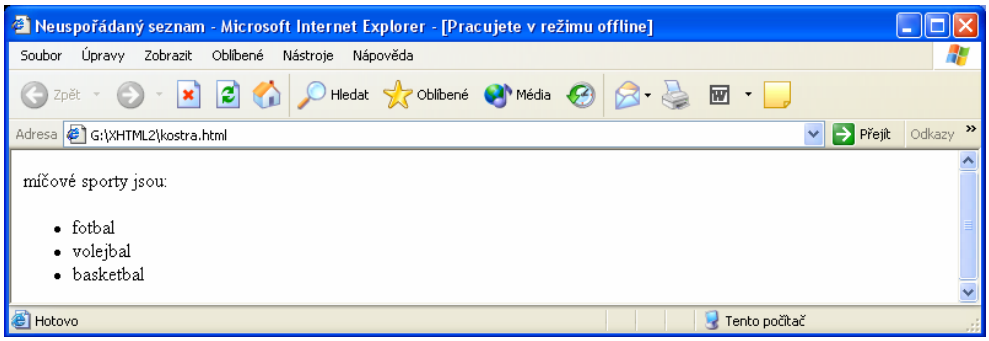
Tag <ul> prohlížeč zobrazí jako neuspořádaný seznam položek. Pro vytvoření jednotlivých položek seznamu potřebujeme tag <li>. Do seznamu můžeme vložit jakýkoliv text, seznam, odkaz atd.

Oba tagy mají několik atributů. Uvedeme si ty nejdůležitější a nejpoužívanější. Nejpoužívanějším atributem je class. Tento atribut nám je velmi dobře známý a slouží k implementaci nadefinovaných částí v CSS stylech. Díky tomu můžeme před každou položku seznamu vložit i svoji vlastní ikonu.

Zcela samostatnou kapitolou jsou atributy událostí. Tyto události můžeme převzít z JavaScriptu a DHTML. Dost bylo teorie. Ukážeme si příklad, kde bude vše jasně vidět.

**Příklad:**

```
míčové sporty jsou:
<ul>
<li>fotbal</li>
<li>volejbal</li>
<li>basketbal</li>
</ul>
```




**Obrázek 19:** Seznamy.

### 5.4.4.2 Uspořádané seznamy

Uspořádaný seznam použijeme v situaci, kdy potřebujeme jednotlivé položky seznamu očíslovat.

#### *5.4.4.2.1 Tag ol*

Tag <ol> formátuje obsah seznamu stejně jako tag <ul>. Jednotlivé položky neoznačuje odrážkami, ale pořadovými čísly. Číslování začíná od jedné a každou následující položku seznamu zvyšujeme o jedničku.

Nebudu zde jmenovat všechny atributy, uvedu ty nejdůležitější: class, href. Nejpoužívanějším atributem je class. Tento atribut nám je velmi dobře známý a slouží k implementaci nadefinovaných částí v CSS stylech.

Atribut href je novinkou mezi atributy a umožňuje vytvořit hypertextový odkaz bez použití tagu <a>.

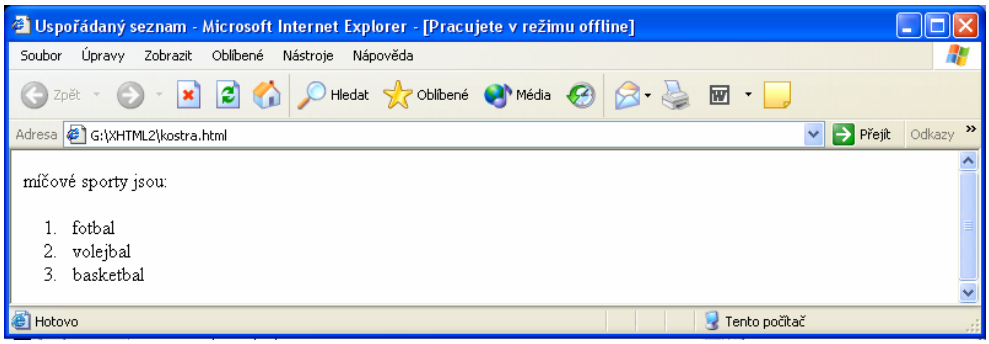
Dříve jsme byli zvyklý používat v seznamech atributy start a type. V XHTML 2 byly plně nahrazeny CSS styly.

**Příklad:**

```
míčové sporty jsou:
<ol>
<li>fotbal</li>
<li>volejbal</li>
<li>basketbal</li>
</ol>
```

**Vysvětlení:**

Jednotlivé položky seznamu jsou očíslovány. Viz. obrázek.




**Obrázek 20:** Číslované seznamy.

### 5.4.4.3 Navigační menu pomocí seznamu

V značkovacích jazycích byl problém vytvořit jednoduchým způsobem elegantní navigační menu. Konsorcium W3C vytvořilo nové elementy, které problém vyřešily. Nyní prohlížeče neumí navigační menu zobrazit, proto uvedu pouze příklady bez výsledné interpretace.

#### *5.4.4.3.1 Tag nl a label*

V XHTML 2 se objevuje novinka. Můžeme vytvořit menu. Na první pohled budou zobrazeny texty v tagu <label>. Po "najetí" kurzorem myši na tento text se nám zobrazí zbylé části menu. Pokud z textu "odjedeme" kurzorem myši zůstane nám na obrazovce pouze "základní menu" (podpoložky menu se schovají).

Při tvorbě menu musíme začít tagem <nl>. Poté uvedeme tag <label>, který slouží jako hlavní položka, pod kterou jsou uvedeny jednotlivé položky menu. Pochopitelně můžeme vkládat tagy do sebe, tím vytvoříme podmenu. Nesmíme zapomenout příslušné tagy včas uzavřít, aby nedošlo ke křížení tagů.

V současné době můžeme použít nový atribut href a standardní atribut class. Díky atributu href můžeme tvořit hypertextové odkazy (ve starších verzích jazyka nebylo možno). Atribut class jsme si již vysvětlili dříve.

**Příklad:**

```
<nl>
<label>Zdroje </label>
<li href="http://www.nasa.gov">Nasa</li>
<li>
<nl>
<label>Měsíce</label>
<li href="#leden">Leden</li>
<li href="#únor">Únor</li>
</nl>
</li>
</nl>
```

**Vysvětlení:**

Tag **<nl>** uvozuje menu. Pak musí následovat tag **<label><label/>**, ve kterém uvedeme hlavní položku menu. Dále následují jednotlivé položky menu uvozené do tagu **<li></li>**. V příkladě je ukázáno, jak vložíme do menu další menu.

### 5.4.4.4 Definiční seznamy

XHTML 2 podporuje ještě jeden typ seznamů, který je odlišný od standardních seznamů. Jsou to seznamy s definicí. Seznam se podobá encyklopedii, většinou obsahuje text, obrázky nebo multimediální elementy. Slouží k prezentaci slovníkových pojmů, hesel.

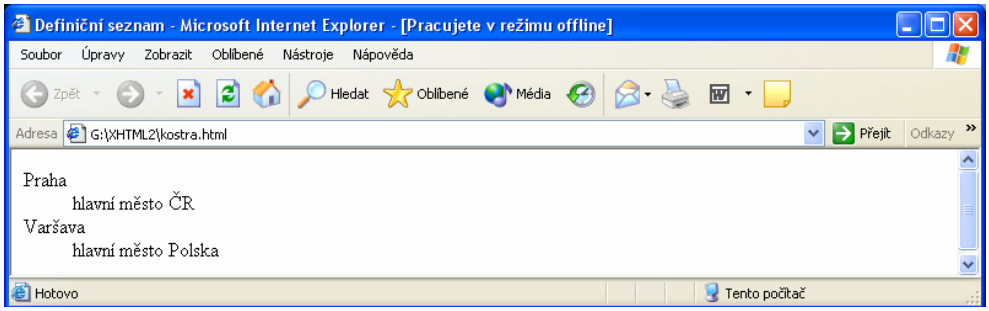
#### *5.4.4.4.1 Tagy dl, dt, dd*

Pomocí tag <dl> definujeme celý seznam. Položky v seznamu označujeme pomocí tagu <dt>. Dále následuje tag <dd> vysvětlující příslušný pojem.

O grafickou část se nám starají CSS styly. Komunikaci předdefinovaného stylu s konkrétním tagem zajistí atribut class. Můžeme zde využít služeb atributu href, který je popsán v minulých kapitolách.

**Příklad:**

```
<dl>
    <dt>Praha</dt>
        <dd>hlavní město ČR</dd>
    <dt>Varšava</dt>
        <dd>hlavní město Polska</dd>
</dl>
```




**Obrázek 21:** Definice pojmů.

## 5.4.5 Modul link

Modul link má jediný tag <link>. Definuje propojení mezi tímto dokumentem a jiným dokumentem v záhlaví dokumentu <head>. Představme si rozsáhlý. Orientace v něm bývá velmi špatná. Nyní nastane následující situace. Do vyhledávacího serveru zadáme požadavek na nalezení slova. Díky tomu, že poskládáme jak spolu stránky sousedí, bude vy výsledku lepší přehled o našem webu. Dost bylo teorie přejdeme k tvorbě linků. Tag <link> na obrazovce vůbec neuvidíme. Pokud ho ve svém dokumentu neuvedeme nic se nestane.

Tag <link> má celou řadu nezbytných atributů. Prvním z nich je atribut rel, který popisuje vztah mezi aktuálním elementem a adresou, na který se element odkazuje. Dalším atributem je href, který udává adresu příslušného dokumentu. Atribut rel popisuje vztah mezi aktuálním elementem a adresou, na niž element odkazuje. Atribut rev udává zpětný odkaz ze záložky, na kterou element odkazuje.

**Příklad:**

Máme dva dokumenty. Dokument A a dokument B. Nastavme pomocí tagu <link> vztah tak, aby bylo jesné, že dokument A je „indexový“ (nadřazený) dokumentu B.

Zdrojový kód dokumentu A:

```
<head>
<link href="docB" rel="index"/>
</head>
```

Zdrojový kód dokumentu B:

```
<head>
<link href="docA" rev="index"/>
</head>
```

### 5.4.6 Modul metainformace

Modul metainformace má jediný tag <meta>. Tag má formu informativní. Prohlížeč ho ignoruje, protože tag je uveden v hlavičce <head> dokumentu. Tento tag můžeme vynechat(na funkčnost dokumentu nemá vliv). Tag <meta> nám umožní, aby naše stránky byly rychleji nalezeny a vynikly mezi milióny webových stránek. V následujícím příkladě uvede úplný seznam atributů, které tag <meta> obsahuje. V syntaxi jazyka XHTML 2 nepoužíváme atribut content a tag <meta> zde zapisujeme jeko párový tag (tag musíme zakončit pomocí </meta>).

**Příklad:**

```
<head>
<!-- atribut name  -->
<meta name="author">Vojtěch Soukup</meta>
<meta name="copyright">&copy; 2004 Acme
Corp.</meta>
<meta
name="keywords">corporate,guidelines,cataloging</me
ta>
<meta name="date">1994-11-06T08:49:37+00:00</meta>
<meta name="description"> Výuka XHTMl </meta>
```

```
<meta name="generator">Golden html </meta>
<meta name="robots">index,follow,all </meta>
<meta name="country">cz </meta>
<meta name="language">czech </meta>
<meta name="resource-type">document</meta>
<meta name="googlebot">nosnippet</meta>

<!-- atribut http-equiv -->
<meta http-equiv="Content-Type" charset=windows-
1250">text/html</meta>
<meta http-equiv="pragma">no-cache</meta>
<meta http-equiv="expires"> Sat 21 Jun 2004
20:00:00</meta>
<meta http-equiv="refresh";
URL=http://www.nekam.cz/cokoliv">6</meta>
<meta http-equiv="content-language">cz </meta>
<meta http-equiv="cache-control">no-cache</meta>
<meta http-equiv="PICS-Label">hodnocení PICS</meta>
<meta http-equiv="content-script-type">text-
javascript </meta>
<meta http-equiv="imagetoolbar">no</meta>

<!-- přechodový efekt stránky  -->
<meta http-equiv="Page-Enter">
revealTrans(Duration=3.0,Transition=12)</meta>

<title>Výuka jazyka XHTML v předmětu WWW1</title>
</head>
```

**Vysvětlení:**

**<head></head>** - Párový tag.

**<meta name="author">**Vojtěch Soukup**</meta>** - Do tagu <meta> uvedeme jméno autora dokumentu. Pokud tento atribut vyplníme, mohou vyhledavače vyhledávat podle jména autora.

**<meta name="copyright**">© 2004 Acme Corp.**</meta>** - Vyplníme-li tag <meta> budou všichni vědět v jakém roce byl dokument vyroben a označen značkou autorství.

**<meta name="keywords">**corporate,guidelines,cataloging**</meta>** - Do tagu <meta> napíšeme klíčová slova podle kterých vyhledávají vyhledavače (Gogole, aj.).

**<meta name="date">**1994-11-06T08:49:37+00:00**</meta>** - Tag <meta> obsahuje příslušné datum tvorby dokumentu.

**<meta name="description">** Výuka XHTMl **</meta>** - Tag <meta> slouží k popsání obsahu stránky. Dobré je používat jen krátké heslo (název firmy, atd).

**<meta name="generator">**Golden html **</meta>** - Do tagu <meta> vyplníme název programu, ve kterém byl dokument vytvořen.

**<meta name="robots">**index,follow,all **</meta>** - Tag <meta> umožní vyhledávačům, aby mohli jít na stránky mých kamarádů.

**<meta name="country">**cz **</meta>** - Vyplněný tag <meta> zajišťuje rychlejší vyhledávání stránek.

**<meta name="language">**czech **</meta>** - Vyplněný tag <meta> zajistí, že vyhledávač bude vědět v jakém jazyce jsou stránky napsány.

**<meta name="resource-type">**document**</meta>** - Zápis udává, že se jedná o dokument. Tag je důležitý pro některé vyhledávače.

**<meta name="googlebot">**nosnippet**</meta>** - Tag může obsahovat stejné hodnoty jako robots a navíc je obohacen o hodontu **nosnippet** ( ve výpisu vyhledaných stránek nebude u naší stránky popisek), **noarchive** (vyhledavač google znepřístupní uživateli verzi stránky, kterou má uloženou u sebe).

Vysvětlili jsme u tagu <meta> možnosti atributu name. U tagu <meta> můžeme použít atribut http-equiv.

**<meta http-equiv="Content-Type" charset=windows-1250">** text/html**</meta>** - Vyplnění tagu <meta> s atributem http-equiv="Content-Type" a atributu charset zajistí nastavení češtiny. Atribut charset může obsahovat pro nastavení češtiny následující hodnotu: **iso-8859-2**.

**<meta http-equiv="pragma">**no-cache**</meta>** - Nastavení tohoto tagu nám zamezí ukládání stránky na počítači klienta. Dobře si rozmyslete zda uživatelům zamezíte ukládání Vaší stránky. Může se Vám stát, že tyto stránky nebude nikdo navštěvovat.

**<meta http-equiv="expires">** Sat 21 Jun 2003 20:04:00**</meta>** Prohlížeč si takto označenou stránku zapamatuje. Kdyby se na ní chtěl uživatel z téhož počítače podívat po 20 hodině 21. června 2004, prohlížeč si sám data zaktualizuje ze serveru, protože ví, že předchozí data jsou neplatná.

**<meta http-equiv="refresh"; URL=http://www.nekam.cz/cokoliv">** 6**</meta>** - Vyplnění tagu <meta> nám zajistí, že po 6 vteřinách od načtení, zajistí přesměrování na webovou adresu v atributu URL.

**<meta http-equiv="content-language">**cz **</meta>** - Uvedení této informace je velmi důležité pro správné zařazení stránky do databáze vyhledávačů, navíc některé vyhledávače přikládají stránce v takovém případě vyšší prioritu.

**<meta http-equiv="cache-control">**no-cache**</meta>** - Pomocí této hodnoty ovládáme cache přímo. Jako hodnotu můžeme použít: **no-chache** (stránka se nesmí ukládat, **public** (stránka se smí ukládat), **max-age=sekundy** (specifikuje dobu od požadavku, po jakou je stránka aktuální – po uplynutí této doby musí prohlížeč zažádat o novou verzi stránky).

**<meta http-equiv="PICS-Label">**hodnocení PICS**</meta>** - PICS znamená *Platform for Internet Content Selection*. Jedná se o systém hodnocení obsahu webových stránek, původně určený k tomu, aby se děti nedostali k „závadnému obsahu“. Postupem času se z něj ale stal systém pro charakteristiku jakýchkoli dat.

**<meta http-equiv="content-script-type">**text-javascript **</meta>** - Díky tomuto nastavení můžeme v dokumentu používat javascript.

**<meta http-equiv="imagetoolbar">**no**</meta>** - Tag zruší v Internet Exploreru tlačítko, které je aktivováno „přejetím“ myší přes obrázek

Pomocí tagu <meta> můžeme nastavit přechodové efekty mezi stránkami.

**<meta http-equiv="Page-Enter" >** revealTrans(Duration=3.0, Transition=12) **</meta> -** Pomocí tohoto tagu aktivujeme přechody mezi jednotlivými stránkami. Parametr http-equiv může mít následující hodnoty:

- **Site-Enter** - efekt se zobrazí při vstupu na server
- **Site-Exit** - efekt se zobrazí při opuštění serveru

- **Page-Enter** - efekt se zobrazí při vstupu na server
- **Page-Exit** - efekt se zobrazí při opuštění serveru

Atribut Duration udává délku efektu v sekundách. Transition nastavuje druh přechodu a výsledný efektu zobrazení:

- **0** - Obdélník dovnitř
- **1** - Obdélník ven
- **2** - Kruh dovnitř
- **3** - Kruh ven
- **4** - Rolování nahoru
- **5** - Rolování dolů
- **6** - Rolování doprava
- **7** - Rolování doleva
- **8** - Svislá roleta
- **9** - Vodorovná roleta
- **10** - Šachovnice napříč
- **11** - Šachovnice dolů
- **12** - Nahodile prolnout
- **13** - Rozdělit svisle ke středu
- **14** - Rozdělit svisle k okraji
- **15** - Rozdělit vodorovně ke středu
- **16** - Rozdělit vodorovně k okraji
- **17** - Odříznout doleva dolů
- **18** - Odříznout doleva nahoru
- **19** - Odříznout doprava dolů
- **20** - Odříznout doprava nahoru
- **21** - Náhodné proužky vodorovně
- **22** - Náhodné proužky svisle
- **23** - Náhodné

**<title>Výuka jazyka XHTML v předmětu WWW1</title>** - V záhlaví prohlížeče bude napsáno: „Výuka jazyka XHTML v předmětu WWW1“.

Tag meta není povinný. Pokud ho ovšem vyplníme zajistíme si lepší výchozí pozici, aby naše stránky někdo našel. Tag <meta> v nové úpravě pro XHTML 2 není prohlížeči podporován. Doufejme, že Internet Explorer 7 začne podporovat XHTML 2.

## 5.4.7 Modul objekt

Modul objekt je velmi mladý a byl doplněn až v poslední době. Začíná získávat na síle o čemž se v XHTML 2 přesvědčíme.

### 5.4.7.1 Tag object

Tag <object> implementoval původně Microsoft pro podporu appletů ActiveX. Později byla doplněna podpora jazyka Java. S verzí XHTML 2 byl tag <object> doplněn o podporu obrázků, hudby, filmů atd.
Tag object má následující atributy: archive, content-lenght, data, declare, type, xml:base, id.

- **archive** – Atribut archive urychluje načítání objektů. Hodnotu atributu archive tvoří do uvozovek zapsaný seznam URL adres. Každá adresa ukazuje na archiv, který má prohlížeč před vlastním zobrazením objektu načíst.
- **content-lenght** – Autor nastaví jak velká část dat příslušného objektu má být načtena do zásobníku.
- **data** – V atributu data nastavujeme datový soubor. Hodnotu atributu tvoří URL adresa požadovaného souboru.
- **declare** – Díky atributu declare můžeme objekt pouze nadeklarovat. Prohlížeč jej nebude načítat.
- **type** – Atribut type definuje typ MIME dat, obsažených v souboru deklarovaném pomocí atributu data. Nadeklarujeme

jaký druh dat chceme v prohlížeči zobrazit, či přehrát (obrázek, hudební soubor, film).

- **xml:base** – Atribut definuje odkud jsou data čerpána.
- **id** – Pomocí atributu id vkládáme předdefinované prvky CSS stylů.

#### *5.4.7.1.1 Vložení obrázku, textového a html souboru*

Ve verzích HTML a XHTML 1.1 se vkládaly obrázky pomocí tagu <img>. Již ve verzi XHTML 1.1 bylo poprvé možno vkládat obrázky pomocí tagu <object>. Ve verzi XHTML 2 vkládáme obrázky pouze pomocí tagu <object>. **Tag <img> byl definitivně zrušen.**

**Příklad:**

**Načtení obrázku pomocí XHTML 2:**

```
<object data="http://www.example.com/foo.jpg"
type="image/jpeg">
<em>alternate text</em>
</object>
```

**Načtení obrázku pomocí XHTML 1.1:**

```
<object data="http://www.example.com/foo.jpg"
type="image/jpeg" width="300" height="530" >
</object>
```

**Vysvětlení:**

Již v XHTML 1.1 bylo možno používat tag <object> k načítání obrázků. V XHTML 2 nastali některé úpravy příslušných atributů. XHTML 2 nevyužívá atributů height a width. Tyto atributy nahrazují CSS styly.

Pokud budeme chtít načíst obrázek gif, musíme změnit atribut type takto: **type="image/gif"**. V případě použití png bude atribut type vypadat takto: **type="image/png"**. Kromě obrázků můžeme načítat html stránky a textové soubory: **text/html, text/plain**. Tag  obsahuje alternativní text (v jazyku HTML a XHTML 1.1 známý jako atribut alt), zformátovaný podle tagu . Místo tagu  můžeme dát jiné formátovací tagy textu.

Pokud načteme obrázek pomocí XHTML 1.1 bude zobrazen standardním způsobem. Bohužel, pokud budeme chtít zobrazit obrázek pomocí XHTML 2, načte se nám prázdná obrazovka. Důvod je prostý. V současné době není prohlížeči podporováno načtení obrázku pomocí XHTML 2.

Uvedu zde příklady načtení html souboru a textového souboru pomocí XHTML 2.

**Příklad:**

**Načtení textového souboru:**

```
<object data="textfile.txt" type="text/plain" >
</object>
```

**Načtení html souboru:**

```
<object data="htmlfile.html" type="text/html" >
</object>
```

#### *5.4.7.1.2 Vložení hudby, videa*

V předchozích verzích značkovacích jazyků byly hudba a film vkládána pomocí tagu <img>. Od jazyka XHTML 2 budeme hudbu a film vkládat pomocí tagu <object>.

**Příklad:**

```
<object data="http://www.example.com/foo.mp3"
type="audio/mpeg">
<em>alternate text</em>
</object>
```

**Vysvětlení:**

Rozdíl mezi vkládáním obrázků a hudby je pouze jeden. Atribut type nastavíme takto: **type="audio/mpeg"**. Vkládání filmů je velmi jednoduché. Stačí změnit atribut type tímto způsobem: **type="video/mpeg"**. V současné době není vkládání hudby a filmů přes tag <object> prohlížeči podporováno.

#### *5.4.7.1.3 Vložení appletů*

Dříve se vkládaly applety pomocí tagu <applet>. V jazyku XHTML 2 jsme již pokročili. Vkládání appletů provedeme pomocí tagu <object>.

**Příklad:**

```
<object
data="http://www.example.com/TheEarth.class"
type="application/x-java-applet">
</object>
```

**Vysvětlení:**

Applety vkládáme stejným způsobem jako obrázky, hudbu, filmy. Musíme změnit atribut type a to takto: **type="application/x-java-applet"**. Vkládání appletů je velmi jednoduché. V současné době prohlížeče nepodporují vložení appletů přes tag object.

#### *5.4.7.1.4 Obrázkové mapy*

V této kapitole nebudu klikací mapy vysvětlovat. Mapy vysvětlím v kapitole 8.12, kde ukáži jak postupoval vývoj „klikacích map“ od jazyka HTML přes jazyk XHTML 1.1 až XHTML 2.

### 5.4.7.2 Tag param

Tag <param> popisuje paramenty tagu <object>. Např: velikost okraje objektu, barva pozadí atd. Sám o sobě tag <param> nemá význam.

Tag <param> má následující atributy: name, value. Atribut name (name) obsahuje v uvozovkách jméno. Atributu value (hodnota) obsahuje v uvozovkách hodnotu k příslušnému atributu name.

**Příklad:**

```
<object data="Clock.class"
type="application/x-java-applet">
<param name="link" value="http://www.example.com"/>
```

```
<param name="border" value="5"/>
<param name="bgcolor" value="ddddff"/>
<em>alternate text</em>
</object>
```

**Vysvětlení:**

Tagu <object> se věnovat nebudu. Vysvětlíme si pouze hodnoty tagu <param>.

**<param name**="link" **value**="http://www.example.com"**/> -** Hodnota atributu name zajistí nastavení cesty, odkud bude applet načten. Atribut value obsahuje URL adresu.

<**param name**="border" **value**="5"**/>** - Atribut name obsahuje hodnotu border (nastavení okrajů objektu). Atribut value obsahuje šířku okraje.

**<param name="bgcolor" value="ddddff"/>** - Atribut name s hodnotou bgcolor (pozadí) nastaví barvu pozadí. Barvu mu předá atribut value.

Pochopitelně i v tomto případě je problém s podporou prohlížečů. Doufejme, že v nových verzích webového prohlížeče Internet Explorer bude podpora lepší.

## 5.4.8 Modul skript

V modulu skript využijeme dva tagy (tag <script>, tag<noscript>). Modul slouží k vložení skriptovacích jazyků (JavaScript, VBScript, aj.). JavaScript využívá vestavěných funkcí prohlížeče. Příkazy jazyka JavaScript můžeme zapsat na libovolné místo dokumentu.

### 5.4.8.1 Tag script

JavaSript můžeme použít v XHTML 2 díky tagu <script>. Vše, co se nachází mezi tagem <script> a její koncovou značkou </script>, zpracovává prohlížeč jako příkazy jazyka JavaScript. Do tohoto tagu **nesmíme** zapsat běžný HTML, XHTML 1.1 a XHTML 2.

Tag <script> obsahuje základní atributy: charset, type, src, declare.

- **charset** – Používá se ve spojení s atributem src. Označuje znakovou sadu (standardní znaková sada ISO), v níž je program jazyka JavaScript zakódován.
- **src** – V jazyce JavaScript můžeme vytvořit rozsáhlé programy, které uložíme do zvláštního souboru. V tomto případě využijeme atribut src, který načte tento soubor. Hodnotu atributu src tvoří adresa URL, kterou musíme uzávorkovat.
- **type** - Atribut type nahradil atribut language. Jeho hodnota bude vypadat následovně: **type="text/javascript"**. Prohlížeč bude vědět, že má zobrazit JavaScript.
- **declare** - Atributu declare objekt pouze nadeklaruje. Prohlížeč jej nebude načítat.

**Příklad:**

**Deklarace Javascriptu:**

```
<script type="text/javascript">
document.write("<h2>Table of Factorials</h2>");
for(i = 1, fact = 1; i < 10; i++, fact *= i) {
     document.write(i + "! = " + fact);
     document.write("<br>");
}
</script>
```

**Deklarace VBScriptu:**

```
<script type="text/x-vbscript"
src="http://example.org/progs/vbcalc">
<!—- zde vložíme příslušný kód -->
</script>
```

**Upozornění:**

Ve verzích HTML a XHTML se obsah tagu <script> zapisoval do komentáře, kvůli tomu, že starší prohlížeče tag <script> nepodporují.

Konsorcium W3C zatím nerozhodlo, jestli budeme muset obsah tagu zapisovat do komentáře.

#### 5.4.8.2 Tag noscript

Tag <noscript> slouží prohlížečům, které nepodporují tag <script>. V dnešní době je takových prohlížečů velmi málo, proto mnoho uživatelů tento tag nepoužíval ani v HTML a XHTML.

Tag <noscript> obsahuje atributy jako tag <script> a to jsou: : charset, type, src, declare. Nebudu tu uvádět význam jednotlivých značek, protože jsem je vysvětlil u tagu <script>.

**Příklad:**

```
<script type=" text/javascript "
src="http://example.org/script">
<noscript>
<p>Access the <a
href="http://example.org/data">data.</a></p>
</noscript>
</script>
```

**Vysvětlení:**

Nepodporuje-li prohlížeč tag <script>, zobrazí obsah tagu <noscript>. Tohle není jediný způsob jak zajistit podporu JavaScriptu. Další postupy zajištění podpory JavaScriptu najdete v knize: JavaScript: Kompletní průvodce (O'Reilly & Associates, překlad Computer Press).

### 5.4.9 Modul styly

Hlavní myšlenkou jazyka XHTML2 je snaha oddělit obsah dokumentu, který nese informace, od definice grafického vzhledu. Konkrétní podobu dokumentu vytváříme tak, že definujeme vzhled jednotlivých elementů. K tomu používáme jiné soubory, které jsou k našemu dokumentu připojeny prostřednictvím odkazů. Pro jeden dokument můžeme mít několik stylů,

nebo naopak pro více dokumentů podobného charakteru styl jediný. To je výhodné při tvorbě profesionálního webu, kde se klade důraz na jednotnost nadpisů, písma, barev, pozadí, atd. Pouhým změněním jednoho atributu změníme pozadí na celém webu. K tomu abychom mohli v dokumentu používat CSS styly, potřebujeme tag <style>.

#### 5.4.9.1 Tag style

Abychom mohli k našemu XHTML dokumentu připojit soubor s formátovacím stylem, použijeme tag <style>. Tag <style> musíme uvést v záhlaví dokumentu (<head>). Vše co bude obsaženo mezi tagy <style> a </style>, považujeme za součást stylů.

Pro tag <style> využijeme celou řadu atributů: type.

- **type** – Definuje typy stylů, které do tagu zapisujeme. Kaskádové styly mají typ **text/css**.

**Příklad:**

```
<head>
<style type="text/css">
h {text-align: center}
</style>
</head>
<body>
<h>Kapitola jedna</h>
</body>
```

**Vysvětlení:**

V hlavičce jsme nadeklarovali CSS styly (**<style type="text/css"> </style>**). Tento tag obsahuje přeformátovaný tag <h>. Text „Kapitola jedna“ bude vycentrován na střed obrazovky.

##### *5.4.9.1.1 Style jako atribut*

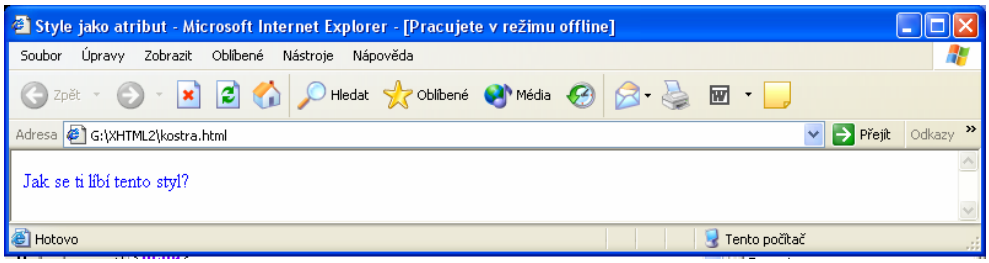
Kaskádové styly můžeme vložit do každého tagu samostatně pomocí atributu style.

**Příklad:**

```
<p style="font-size: 12pt; color: blue">Jak se ti
líbí tento styl?</p>
```

**Vysvětlení:**

Díky atributu style bude text odstavce (tag **<p>**) zobrazen s těmito parametry: Písmo bude mít velikost 12 bodů a barva bude modrá.




**Obrázek 22:** Využití CSS stylů.

### *5.4.9.1.2 Externí šablony stylů*

Představme si, že máme více dokumentů. Potřebujeme, aby všechny dokumenty měly stejné písmo, pozadí, nadpisy, atd. Jak toho docílíme? Velmi jednoduchým způsobem. Využijeme znalosti tagu <link>.

Tag <link> má celou řadu nezbytných atributů: rel, type.

- **rel** - Atribut rel popisuje vztah mezi aktuálním elementem a adresou, na který se element odkazuje. Pro CSS styly bude vypadat takto: **rel=stylesheet**.
- **href** - Atribut href udává URL adresu příslušného dokumentu.

**Příklad:**

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

**Vysvětlení:**

Nejprve si založíme nový soubor s názvem **vojta.css**. V něm vytvoříme pravidla stylů. Poté musíme importovat tento soubor do našeho dokumentu. Kód importu musí být vložen do hlavičky dokumentu a měl by vypadat takto: **<link href="vojta.css" rel="stylesheet" type="text/css"/>**. V těle dokumentu se pravidla pro použití CSS stylů nemění.

## 5.4.10 Modul tabulky

Základní model tabulky je velmi jednoduchý. Skládá se z buněk, které tvoří řádky a sloupce. Většina buněk obsahuje datové hodnoty. Ostatní pak obsahují záhlaví řádků a sloupců s bližším popisem dat.

### 5.4.10.1 Tag <table>

Tag <table> definuje tabulku. Zde také zapisujeme veškeré tagy a hodnoty příslušných buněk. Tabulku definujeme směrem shora dolů a zleva doprava. Obsah povolený uvnitř tabulky tvoří tag <tr>, který definuje jednotlivé řádky tabulky. Dále použijeme tyto značky: <thead>, <tbody>, <tfoot>, <col>, atd..

V tag <table> můžeme použít tyto základní atributy: class, dir. V jazycích HTML a XHTML 1.1 jsme používali další atributy (border, width, atd.). V XHTML 2 je doporučeno tyto tagy nepoužívat. Místo nich je vhodné použít CSS styly.

- **class** – Umožňuje použití nadefinovaných CSS stylů v dokumentu.
- **dir** – Atribut udává směr textu. Může nabývat následujících hodnot: ltr – zleva doprava, rtl – zprava doleva, lro – zleva doprava překryté, rlo – zprava doleva překryté.

**Příklad:**

```
<table dir="rtl">
<!--  Zde uvedeme obsah tabulky-->
```

```
</table>
```

### 5.4.10.2 Tag <caption>

Tag <caption> (titulek) slouží jako titulek tabulky. Tag zapisujeme hned za tag <table>. Titulek obsahuje libovolný text. Prohlížeče zatím tento tag nepodporují. V budoucnu se po najetí kurzorem myši na tabulku objeví hodnota tagu <caption>. V příkladu si ukážeme místo, kam zapisujeme tag <caption>.

**Příklad:**

```
<table>
<caption>Prospěch studentů</caption>
</table>
```

### 5.4.10.3 Tag <tr>

Tag <tr> vytvoří nový řádek tabulky. Do tagu <tr> zapisujeme jednu nebo více buněk se záhlavím tabulky.

Využití atributů je stejné jako u tagu <table>. Konsorcium W3C nás nutí používat CSS styly, proto jsou starší atributy typu bordercolor, aj. vypuštěny.

**Příklad:**

```
<table>
<tr></tr>
<tr></tr>
<tr></tr>
</table>
```

**Vysvětlení:**

Tabulka obsahuje tři prázdné řádky.

### 5.4.10.4 Tagy <th> a <td>

Tagy <th>a <td> zapisujeme do značky <tr>. Pomocí nich vytváříme jednotlivé buňky řádku. Oba tagy fungují podobným způsobem. Rozdíl mezi nimi je jediný: Tag <th> použijeme při tvorbě záhlaví. Záhlaví tabulky tak bude zvýrazněno a odlišeno od ostatních buněk tabulky. Data v tagu <th> jsou implicitně zarovnávána na střed. Tag <td> použijeme pro ostatní hodnoty tabulky. Data jsou implicitně zarovnávána doleva.

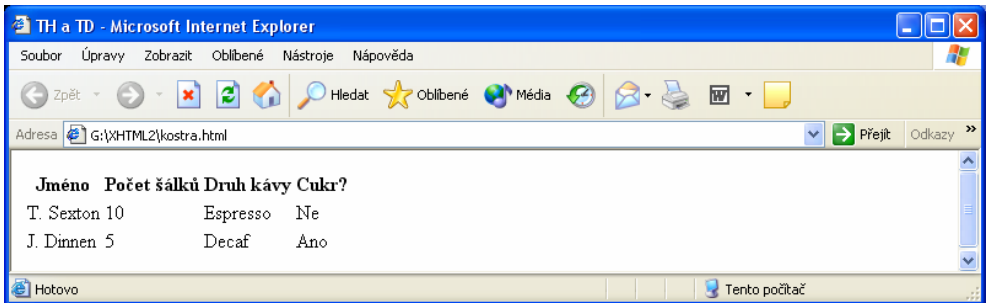
U tagů <th> a <td> můžeme použít následující atributy: abbr, axis, colspan, headres, rowspan, scope.

- **abbr** – Atribut obsahuje zkrácený popis obsahu buňky uvedený v uvozovkách. Prohlížeč tento atribut zobrazí při nedostatku místa.
- **axis** – Hodnotu atributu axis tvoří do uvozovek zapsaný seznam názvů, na které můžeme vytvořit dotaz. Tento atribu využijeme pokud budeme chtít vytvořit součty cen za potraviny. Nadefinujeme si atribut takto: axis=potraviny.
- **colspan** – Atribut obsahuje celočíselnou hodnotu v uvozovkách, která vyjadřuje počet sloupců, přes něž má buňka záhlaví či dat zasahovat.
- **headres** – Atribut slouží pro nevizuální prohlížeče (data jsou čtena). Atribut oddělí záhlaví od jednotlivých buněk.
- **rowspan** – Atribut rowspan rozšíří buňku tabulky přes několik sloupců (směrem dolů).

**Příklad:**

```
<table>
<tr>
<th>Jméno</th>
<th>Počet šálků</th>
<th>Druh kávy</th>
<th>Cukr?</th>
</tr>
<tr>
<td>T. Sexton</td>
<td>10</td>
```

```
<td>Espresso</td>
<td>Ne</td>
</tr>
<tr>
<td>J. Dinnen</td>
<td>5</td>
<td>Decaf</td>
<td>Ano</td>
</table>
```




**Obrázek 23:** Tabulka.

### 5.4.10.5 Tag <thead>

Tag <thead> definuje množinu řádků záhlaví tabulky. Tag <thead> smí být v každém tagu <table> obsažen pouze jednou. Tento tag se může objevit v tabulce, kde následné řádky mohou být jen tělem nebo zápatím tabulky. Tag <thead> používá stejné atributy jako tag <tr>.

**Příklad:**

```
<table>
<thead>
<tr> Hlavička dokumentu</tr>
</thead>
<tr></tr>
<tr></tr>
</table>
```

**Vysvětlení:**

Tabulka obsahuje tři řádky. První řádek tvoří záhlaví tabulky, kde budou uvedeny nadpisy sloupců. Zbylé dva řádky slouží k zadání hodnot tabulky.

### 5.4.10.6 Tag <tfoot>

Tag <tfoot> definuje zápatí tabulky. Tag <tfoot> se smí objevit v tabulce pouze jednou, a to na konci dokumentu. Tag může obsahovat jeden nebo několik tagů <tr>.

**Příklad:**

```
<table>
<tr> </tr>
<tr> </tr>
<tfoot>
<tr> zápatí tabulky</tr>
</tfoot>
</table>
```

### 5.4.10.7 Tag <tbody>

Tag <tbody> rozdělí tabulku do několika oddílů. Tag obsahuje jeden, nebo více řádků tabulky, z nichž je vytvořena ucelená sekce. Tag <tbody> je nepovinná a v tabulce nemusí být uvedena. Atributů je celá řada, ovšem prohlížeče je nepodporují, proto je nebudu uvádět.

**Příklad:**

```
<table>
<tbody>
<tr>
<th>Jméno</th>
<th>Počet šálků</th>
<th>Druh kávy</th>
<th>Cukr?</th>
</tr>
<tr>
```

```
<td>T. Sexton</td>
<td>10</td>
<td>Espresso</td>
<td>Ne</td>
</tr>
<tr>
<td>J. Dinnen</td>
<td>5</td>
<td>Decaf</td>
<td>Ano</td>
</tbody>
</table>
```

**Vysvětlení:**

Tabulka byla publikována v kapitole 5.4.10.4. Zde jsme ji obohatili pouze o tag <tbody>. Prohlížeč tabulku zobrazí naprosto stejně jako v minulém příkladě.

### 5.4.10.8 Tag <colgroup>

Tag <colgoup> definuje skupinu sloupců. Tag můžeme použít dvěmi způsoby: Tag definujeme společně pro několik identických sloupců nebo jako kontejner pro několika různých sloupců.

Nejpoužívanějším atributem je span. Obsahuje celočíselnou hodnotu, která vyjadřuje počet sloupců spadajících pod tag <colgroup>. Využijeme ještě atribut id. Ten slouží k propojení nadefinovaných tagů v CSS stylech.

### 5.4.10.9 Tag <col>

Tag <col> slouží k definování vzhledu sloupců v skupině. V tabulce se smí objevit až za tagem <colgroup>. Sama o sobě nemá žádný význam.

Tag <col> obsahuje atribut span. Má stejný význam jako v tagu <colgroup>.

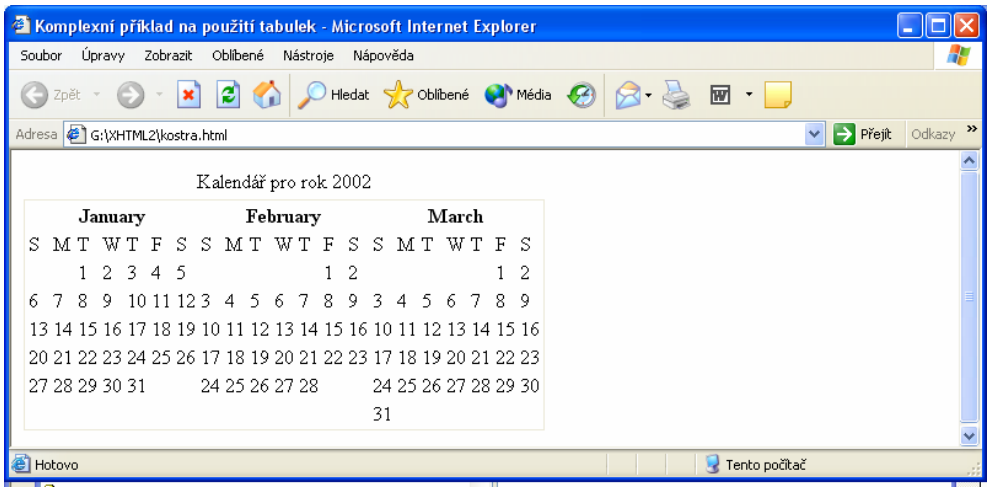
Na závěr modulu tabulky si ukážeme jeden komplexní příklad na využití tabulek v praxi.

**Příklad:**

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 2.0//EN"
"TBD">
<html xmlns="http://www.w3.org/2002/06/xhtml2"
xml:lang="en">
<head>
<title>Komplexní pøíklad na použití tabulek</title>
<style type="text/css">
.tabulka
{
     border: 1px;
     border-style: solid;
}
</style>
</head>
<body>
<table class="tabulka">
<caption>Kalendáø pro rok 2002</caption>
<colgroup span="7"/>
<colgroup span="7"/>
<colgroup span="7"/>
<thead>
<tr>
<th colspan="7" >January</th>
<th colspan="7">February</th>
<th colspan="7">March</th>
</tr>
</thead>
<tbody>
<tr >
<td>S</td><td>M</td><td>T</td><td>W</td><td>T</td><
td>F</td><td>S</td>
<td>S</td><td>M</td><td>T</td><td>W</td><td>T</td><
td>F</td><td>S</td>
```

```
<td>S</td><td>M</td><td>T</td><td>W</td><td>T</td><
td>F</td><td>S</td>
</tr>
</tbody>
<tbody id="tabulka">
<tr >
<td></td><td></td><td>1</td><td>2</td><td>3</td><td
>4</td><td>5</td>
<td></td><td></td><td></td><td></td><td></td><td>1<
/td><td>2</td>
<td></td><td></td><td></td><td></td><td></td><td>1<
/td><td>2</td>
</tr>
<tr>
<td>6</td><td>7</td><td>8</td><td>9</td><td>10</td>
<td>11</td><td>12</td>
<td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><
td>8</td><td>9</td>
<td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><
td>8</td><td>9</td>
</tr>
<tr>
<td>13</td><td>14</td><td>15</td><td>16</td><td>17<
/td><td>18</td><td>19</td>
<td>10</td><td>11</td><td>12</td><td>13</td><td>14<
/td><td>15</td><td>16</td>
<td>10</td><td>11</td><td>12</td><td>13</td><td>14<
/td><td>15</td><td>16</td>
</tr>
<tr>
<td>20</td><td>21</td><td>22</td><td>23</td><td>24<
/td><td>25</td><td>26</td>
<td>17</td><td>18</td><td>19</td><td>20</td><td>21<
/td><td>22</td><td>23</td>
<td>17</td><td>18</td><td>19</td><td>20</td><td>21<
/td><td>22</td><td>23</td>
</tr>
<tr>
```

```
<td>27</td><td>28</td><td>29</td><td>30</td><td>31<
/td><td></td><td></td>
<td>24</td><td>25</td><td>26</td><td>27</td><td>28<
/td><td></td><td></td>
<td>24</td><td>25</td><td>26</td><td>27</td><td>28<
/td><td>29</td><td>30</td>
</tr>
<tr>
<td></td><td></td><td></td><td></td><td></td><td></td><td></
td><td></td>
<td></td><td></td><td></td><td></td><td></td><td></td><td></
td><td></td>
<td>31</td><td></td><td></td><td></td><td></td><td></td><td>
</td><td></td>
</tr>
</tbody>
</table>
</body>
</html>
```




**Obrázek 24:** Tvorba kalendáře pomocí tabulky.

Se základním představením jazyka XHTML 2 jsme na konci. V jazyku XHTML 2 jsou ještě další velké změny (náhrada tagu <br />, vypuštění tagů). Všechny je uvedu v porovnání jazyků HTML, XHTML 1.0 a XHTML 2.0, aby byly patrné posuny v jednotlivých jazycích.

## 5.4.11 Formuláře

Ve formulářích nastává zásadní zlom. U nich se nevychází z XHTML 1.1 ale z XForms 1.0. XForms 1.0 je založen na jazyku XML. Musíme si uvědomit, že zde využijeme minimální množství tagů pro tvorbu formulářů z jazyka XHTML 1.1. Modul XForms 1.0 nebudu popisovat celý. Modul je tak rozsáhlý, že by mohl tvořit samostatnou práci.

### 5.4.11.1 Tag <xforms>

Tag <xforms> nahrazuje tag <form>. Uvedeme ho v záhlaví dokumentu (<head>). Spojení s ostatními prvky formuláře uvedené v části <body> zajišťuje atribut ref. Do formuláře zapisujeme instrukce, speciální popisky vstupních polí atd. V příkladu si ukážeme spojení mezi tagem <xforms> a ostatním tagy formuláře.

**Příklad:**

**Tag <form> v jazyku XHTML 2:**

```
<head>
<xforms:bind nodeset="@as"
relevant="../@as='credit'" required="true()" />
</head>
<body>
<select1 ref="@as">
<label >Auta</label>
<choices>
<item>
<label>Volvo</label>
<value>v</value>
</item>
<item>
<label>Škoda</label>
<value>s</value>
```

```
</item>
</choices>
</select1>
</body>
```

**Vysvětlení:**

Díky atributu ref můžeme využít hodnot, které jsme si nadefinovali v záhlaví dokumentu (typ spojení, grafickou prezentaci).

### 5.4.11.2 Tag <input>

Tag <input> definuje vstupní pole. Tag <input> má atributy ref a class. Atribut class slouží k aplikaci CSS stylů. Atribut ref určuje jaký typ vstupního pole má být vytvořeno. Atribut ref může mít následující hodnoty:

- **/login/password** – Hodnota zapsaná do vstupního pole bude zobrazena pomocí hvězdiček. Vhodné pro psaní hesel.
- **/order/shipDate** – Vstupní pole bude přeměněno na kalendář.

V následujícím příkladě si ukážeme tvorbu vstupního pole ulice pomocí XHTML 2

**Příklad:**

```
<input ref="order/shipTo/street"
class="streetAddress">
  <label>Ulice</label>

</input>
```

### 5.4.11.3 Tag <textarea>

Tag <textarea> ve formuláři vytvoří textovou vstupní oblast o více řádcích. Atribu ref a class mají stejný význam jako u tagu <input>.

**Příklad:**

```
<textarea ref="message/body" class="messageBody">
  <label>Zpráva</label>
  <hint>Zde můžeš nechat svůj vzkaz </hint>
</textarea>
```

**Vysvětlení:**

Tag <label> vytvořil jmenovku textového pole umístěnou vlevo nad textovým polem.

### 5.4.11.4 Tagy <select>, <label>, <choices>, <item>, <value>

Pomocí tagů <select>, <label>, <choices>, <item>, <value> vytvoříme rozbalovací menu. Tag <select> vytvoří základní menu. Tag <label> vloží „jmenovku“ menu. Tag <choices> oddělí „jmenovku“ menu od jednotlivých položek menu. Tag <item> vytvoří jednotlivé položky menu. Tag <value> předá jednotlivým položkám hodnotu.

**Příklad:**

```
<select ref="my:flavors">
<label>Příchutě</label>
<choices>
<item>
<label>Vanilka</label>
<value>v</value>
</item>
<item>
<label>Jahoda</label>
<value>s</value>
</item>
<item>
<label>Čokoláda</label>
<value>c</value>
</item>
</choices>
</select>
```

**Vysvětlení:**

Nemohu ukázat grafickou prezentaci menu, protože tagy <choices> a <item> jsou nové tagy jazyka XHTML 2 a nejsou prohlížeči podporovány. Po rozkliknutí menu si můžeme vybrat jednu ze tří položek: Vanilka, Jahoda, Čokoláda.

#### 5.4.11.5 Odesílací tlačítko (submit)

V každém formuláři potřebujeme odesílací či potvrzovací tlačítko. V XHTML 2 vytvoříme odesílací či potvrzovací tlačítko pomocí tagu <submit>. Jeho atributem je submission.

**Příklad:**

```
<submit submission="timecard">
  <label>Pošli</label>
</submit>
```

**Vysvětlení:**

Tag <label> vytvoří nápis na tlačítku. Tag <submit> slouží k vytvoření odesílacího či potvrzovacího tlačítka.

# 6 Kaskádové styly – CSS

S kaskádovými styly (*Cascading Style Sheets*) jsme se již setkali v jazyce HTML. Můžeme jimi modifikovat výchozí vlastnosti stylů, které v HTML již existují (H1, CODE, atd.) a nebo vytvářet nové styly. V současné době existují dvě verze označované jako CSS1 a CSS2. Styl se skládá z jednotlivých pravidel, která mohou vypadat takto:

**Příklad:**

```
h { font-style: italic; color: red }
```

První část pravidla, v našem případě h, se nazývá **selektor** a vymezuje část dokumentu na něž se bude pravidlo vztahovat. Pak následuje **deklarace**, která určuje, jak bude příslušná část dokumentu vypadat (jaký bude styl a barva písma, zarovnání, atd.). V našem případě je v pravidle uvedeno, že na nadpis úrovně h1 bude aplikován řez písma italic a jeho barva bude červená. V jazyce XML si můžeme definovat elementy sami, tak proč bychom si nemohli nadefinovat i element, který známe z HTML?

Pokud ovšem budeme chtít, aby v našem dokumentu nebyly červené všechny nadpisy úrovně h, ale například jen nadpisy v elementu titulek, pak tuto skutečnost vyjádříme takto:

**Příklad:**

```
titulek h { font-style: italic; color: red }
```

# 7 WAP

V dnešní době se stále více diskutuje možnost, že v blízké budoucnosti budou různá jednoduchá zařízení využívat Web. Mluví se stále více o protokolu **WAP** (*Wireless Application Protocol*), který je obdobou služby WWW pro jednoduchá bezdrátová zařízení. Pod pojmem jednoduchá zařízení mám na mysli mobilní telefony, telefony s jednoduchým terminálem, osobní komunikátory atd.

Pro tvorbu WAPových stránek se používá jazyk **WML** (*Wireless Markup Language*) založený na XML. Jednotlivé servery již postupně začínají nabízet svůj obsah v tomto jazyce. Ovšem mohou se vyskytnout problémy s konverzí, pokud někdo nabízí informace v rozličných formátech. Musí je složitě převádět z jednoho formátu do dalšího, což je dosti pracné a zdlouhavé.

Řešením tohoto problému je zcela jistě jazyk XML, který je vynikající pro výměnu dat mezi serverem a prohlížečem. Vytvoříme jeden XML dokument a k němu několik souborů s různými formátovacími styly. Pak zbývá jen vybrat jeden z těchto formátovacích souborů pro výstup určitého zařízení.

Jednoduchá zařízení nemají výpočetní potenciál počítačů. Pro ně by byl kód dnešních prohlížečů velmi složitý. Jsou vybaveny jen jednoduchým analyzátorem, který je odkázán na správnost kódu. Prohlížeče jsou schopny do jisté míry tolerovat syntaktické chyby autorů webových stránek. Pomocí jazyka XHTML 2, který má přísnější syntaxi, je možno autory WWW stránek přimět k tvorbě platných (valid) dokumentů. Z tohoto důvodu se uvažuje o jeho využití pro jednoduchá zařízení.

# 8 Rozdíly mezi HTML, XHTML 1.0 a XHTML 2

Již jsme se seznámili s tím jak a z čeho vznikl jazyk XHTML. Také jsme se dozvěděli, že dokumenty v něm napsané jsou zpětně kompatibilní s jazykem HTML a navíc splňují i přísnější syntaktická pravidla, která platí

pro XML dokumenty. Při tvorbě dokumentů v jazyce XHTML musíme mít více disciplíny a všímat si maličkostí, které jsou v jazyce XHTML velmi důležité, a které z jazyka HTML neznáme. Měli bychom se zaměřit na dodržování správné struktury a naše dokumenty musí být well-formed. Proto si nyní uvedeme, v čem je syntaxe přísnější a na co si máme dávat při psaní XHTML dokumentů velký pozor.

Podrobněji si tyto rozdíly v psaní HTML a XHTML dokumentů popíšeme v následujících kapitolách.

## 8.1 Elementy a atributy malým písmenem

HTML je jazyk v němž můžeme používat jak malá, tak velká písmena. Bývá pravidlem, že názvy elementů a atributů píšeme velkými písmeny a parametry malými. Lépe je tak odlišíme od vlastního obsahu, který bývá psán malými písmeny.

Jazyk XML je však na velikost písmen citlivý (*case sensitive*) a <li>, <LI> jsou brány jako dva odlišné tagy a alt, ALT, Alt nebo aLT jsou považovány za různé atributy. To se promítá i do jazyka XHTML, kde musíme psát vše kromě <!DOCTYPE> malými písmeny. V definici DTD jazyka XHTML jsou všechny původní elementy a atributy jazyka HTML psány malými písmeny. Jména elementů a atributů, která jsou napsána velkými písmeny jsou tudíž v XHTML neplatná.

Abychom nemuseli při změně HTML dokumentu na XHTML dokument složitě přepisovat velká písmena na malá, můžeme použít různé editační nástroje (např. Word či různé validátory umožňující změnu velikosti písma u názvů elementů a atributů).

**Příklad:**

**Špatně napsané:**

```
<BODY BGCOLOR=#FF00FF text=blue>
```

**Správně napsané:**

```
<body bgcolor="#ff00ff" text="blue">
```

## 8.2 Nutnost ukončovacích tagů

Víme, že všechny elementy v HTML, které mají nějaký obsah, mají i své ukončovací tagy. V některých případech lze koncový tag v kódu vynechat a zpracovávající agent si ho domyslí. Můžeme například vynechat koncový tag </tr>, </td> a </li> v tabulkách a koncový tag  mezi dvěma sousedními odstavci.

V XHTML musí mít všechny elementy své ukončovací tagy nebo musí být napsány speciálním způsobem. Není možné vynechat tento koncový tag. I prázdné elementy jako <hr> - vodorovná čára a <br> - zalomení řádku, musí být ukončeny.

Ukončit je můžeme dvěma způsoby. První způsob je velice prostý. Elementu přidáme ukončovací tag.

**Příklad:**

```
<p>Tohle je jeden odstavec</p>
<p>Tohle je další odstavec</p>
```

Jestliže zkombinujeme počáteční a koncový tag dohromady, dostaneme druhý způsob, jak tagy ukončit. Do počátečního tagu vložíme před pravou úhlovou závorku znak lomítka. Vznikne tak následující značka **'/>'**. Tento způsob se používá převážně u prázdných elementů, tj. elementů bez vlastního obsahu.

**Příklad:**

```
<br/>                    <meta/>
<hr/>                    <input/>
```

Některé prohlížeče mají s takto ukončenými tagy problémy, a proto bychom měli nepárové tagy ukončit raději takto: **' />'**. Před lomítko vložíme mezeru. To není požadavek XHTML, ale starší prohlížeče pak lomítko ignorují a zmiňovaný tag bez problémů zobrazí. Prázdný odstavec můžeme v XHTML zapsat pomocí elementu  a prázdnou buňku v tabulce například <td />.

## 8.3 Správně vnořené elementy

V dobře formulovaném dokumentu musí platit, že všechny elementy jsou do sebe správně vnořeny. To známe i z jazyka HTML. Víme však, že většina dnešních prohlížečů zobrazí i špatně napsaný HTML kód, v němž se jednotlivé tagy kříží. V XHTML se však tagy křížit nesmí, neboť by to mělo za následek nezobrazení celého dokumentu.

Co si tedy představíme pod pojmem správně vnořené elementy? Znamená to, že vnořené elementy musíme uzavírat v opačném pořadí než jsme je otevírali. Je-li například v elementu `<b>` vnořen element `<i>`, pak se musí koncový tag vnitřního elementu, tj. `</i>` objevit ještě před koncovým tagem vnějšího elementu `</b>`.

**Příklad:**

**Špatně napsané:**

```
<b>To je tučné písmo <i>a tohle je kurzíva</b></i>
```

**Správně napsané:**

```
<b>To je tučné písmo <i>a tohle je kurzíva</i></b>
```

**Upozornění:**

Tagy <b> a <i> můžeme používat v HTML a XHTML 1.0. V jazyku XHTML 2.0 jsou vypuštěny.

## 8.4 Náhrada tagu <br />

V jazycích HTML a XHTML 1.0 (1.1) jsme běžně používali tag <br /> (zalamování řádků). Změna nastává od XHTML 2, kde je tento tag označen jako **deprecated** (překonaný). Tag <br /> můžeme zatím ještě používat. Konsorcium W3C to však nedoporučuje. Místo něj byl zaveden nový tag <l> Tento krok zatím ve mně vyvolává rozporuplné pocity. Na jednu stranu je pravda, že tag `<br>` je jedním z nejvíce zneužívaných prvků jazyka, který často supluje CSS, ale dokážu si představit případy, kde bych se bez tohoto tagu obešel jen ztěží.

Tag `<l>` představuje pododstavec. Byl vytvořen jako strukturovaná náhrada tagu `<br>`. Obsahuje text, který má při vizuálním zobrazení začít na novém řádku a na konci odřádkovat. Zdali se text rozdělí na více řádků nebo nikoliv, závisí na dané stylové hodnotě.

Co to znamená v praxi? Element `l` se bude používat například pro kratší řádky, u nichž si můžeme být jisti, že nepřesáhnou rozměr řádku:

**Příklad:**

**Zalomení řádku v jazyce HTML a XHTML 1.0 (1.1):**

```
<p>
první řádek <br />
druhý řádek <br />
třetí řádek
</p>
```

**Zalomení řádku v jazyce XHTML 2:**

```
<p>
<l>první řádek</l>
<l>druhý řádek</l>
<l>třetí řádek</l>
</p>
```

## 8.5 Změna prezentace textu

Konsorcium W3C se rozhodlo v jazyku XHTML 2 vypustit mnoho „prezentačních“ tagů. Byly vypuštěny tyto tagy: `<b>`, `<big>`, `<i>`, `<small>`, `<tt>`. Zůstaly pouze tagy `<sub>`, `<sup>` a `<hr>`. Za tímto výrazným zeštíhlením prezentačního modulu je patrná snaha o prosazení oddělení obsahu od vzhledu. Vzhled dokumentu budou zajišťovat CSS styly.

## 8.6 Tvorba nadpisů

V XHTML 2 přišlo konsorcium W3C s revolučním řešení tohoto tagu. V této části se uvedenou kapitolou nebudu již zabývat. Pokud někdo

zapomněl na rozdíly v tvorbě nadpisů ať zalistuje v mé publikaci do kapitoly 5.4.2.9.

## 8.7 Hodnoty atributů v uvozovkách nebo apostrofech

V HTML jsme mohli do uvozovek psát jakékoli hodnoty atributů. Záleželo pouze na autorovi HTML dokumentu, které hodnoty atributů do uvozovek napsal. Povinně se však psaly jen v případě, že hodnota atributu obsahovala prázdné (mezerové) znaky nebo jiné speciální znaky.

V jazyce XHTML musíme mezi uvozovky uzavírat všechny hodnoty atributů, tedy i číselné, které mohly být v HTML uvedeny bez použití uvozovek.

**Příklad:**

**Špatně napsané:**

```
<table rows=3>
<font face="Times New Roman CE" size=4>
```

**Správně napsané:**

```
<table rows="3">
<font face="Times New Roman CE" size="4">
```

## 8.8 Explicitní hodnoty atributů

Jen málo atributů nemá v jazyce HTML svoji hodnotu. Takovéto atributy zpravidla v elementu reprezentují nějaké přepínače typu zapnuto/vypnuto. Patří mezi ně například atribut compact u různých elementů seznamu nebo atribut ismap v elementu <img>. Uvádím několik atributů, které v XHTML musí mít uvedenu explicitní hodnotu:

**Příklad:**

```
compact="compact"          checked="checked"
nowrap="nowrap"            noresize="noresize"
ismap="ismap"              selected="selected"
```

```
multiple="multiple"                disabled="disabled"
```

XML nepodporuje příznak minimalizace. Hodnota atributového páru (atribut="hodnota") musí být napsána v plném znění. I v jazyce XHTML musí mít každý atribut svoji hodnotu.Do atributu bez hodnoty se zapisuje jeho vlastní jméno. Například místo atributu compact v HTML bychom v jazyce XHTML uvedli **compact="compact"**.

## 8.9 Bílé znaky (whitespace) v hodnotách atributů

Termínem whitespace (bílé znaky) se rozumí znaky mezera, tabelátor a přechod na nový řádek.

V hodnotách atributů uživatelský agent (zpracovávající aplikace) odstraní počáteční a koncové bílé znaky z hodnot atributů a nahradí sekvenci jednoho nebo více bílého znaku jednou mezerou (znak mezera ze sady ASCII – ordinální hodnota 32).

## 8.10 Elementy skript a styl

V jazyce HTML zapisujeme skripty a šablony stylů do komentářů. V XHTML jsou elementy skript a styl brány jako elementy, které mají obsah typu #PCDATA. To znamená, že znaky jako < budou chápány jako začátek elementu a znak & jako začátek entity.

V XML i XHTML je nutné obsah skriptů a šablon umístit do sekce CDATA (kapitola **Chyba! Nenalezen zdroj odkazů.**). Hlavním účelem sekce CDATA je ignorovat znaky, které by byly jinak chápány jako elementy. Jediný oddělovač, který je rozpoznán v sekci CDATA je "]]>" sloužící jako ukončení sekce CDATA.

Toto se uplatní pouze v případě, že skripty a šablony umisťujeme přímo do dokumentu. Ale i zde mohou nastat komplikace. HTML prohlížeče neznají sekci CDATA, a proto ji ignorují. Z tohoto důvodu by měla být v níže uvedeném příkladu vypuštěna sekce CDATA. Ovšem XML prohlížeče se chovají jinak a z obsahu komentáře by mohly náš skript vymazat. Proto je uvedení sekce CDATA nutností.

Při použití externích souborů to bude stejné jako v HTML: <script language="JavaScript" src="/sourcecode.js"></script>.

**JavaScript v jazyce HTML:**

```
<script language="JavaScript">
<!--
document.write("<h2>Table of Factorials</h2>");
for(i = 1, fact = 1; i < 10; i++, fact *= i) {
     document.write(i + "! = " + fact);
     document.write("<br>");
}
//-->
</script>
```

**JavaScript v jazyce XHTML:**

```
<script language="JavaScript">
<!--
<![CDATA[
document.write("<h2>Table of Factorials</h2>");
for(i = 1, fact = 1; i < 10; i++, fact *= i) {
     document.write(i + "! = " + fact);
     document.write("<br />");
}
]]>
//-->
</script>
```

**JavaScript v jazyce XHTML 2:**

```
 <script type="text/javascript">
document.write("<h2>Table of Factorials</h2>");
for(i = 1, fact = 1; i < 10; i++, fact *= i) {
     document.write(i + "! = " + fact);
     document.write("<br>");
}
</script>
```

Nejjednodušším a nejspolehlivějším je uvádět skripty a styly do externích souborů.

## 8.11 Náhrada atributu "name"

Atribut `name` nalezneme v HTML například u elementů <frame>, <form> nebo <img>. I na části dokumentu se odkazujeme pomocí atributu `name` v elementu <a> například takto:

```
<p><a name="zpět na začátek stránky"></a></p>

<a href="#zpět na začátek stránky">
```

V XML i XHTML je již atribut `name` nepřístupný, a proto musíme používat atribut `id`. Většina prohlížečů však atribut `id` nezná. Proto je lepší zkombinovat oba způsoby a uvést jak atribut `id`, tak i atribut `name`.

```
<p><a name="zpět na začátek stránky" id="zpět
      na začátek stránky"></a></p>

  ...

<a href="#zpět na začátek stránky">
```

## 8.12 Obrázkové mapy

Obrázkové mapy jsem zcela úmyslně vsunul až na závěr mé práce. Ukáži zde na několika případech jak se obrázkové mapy vyvíjely v jazyce HTML, XHTML 1 a XHTML 2.

### 8.12.1 Obrázkové mapy v HTML

Klikací mapy jsou součástí značkovacích jazyků od nepaměti. Jejich použití je velmi omezené. Správný tvůrce webu se bez nich neobejde. V jazyce HTML se obrázkové mapy uvozovaly pomocí tagu <map> a tagu <area>. Vše si vysvětlíme na následujícím příkladě.

**Příklad:**

```
<IMG SRC="tabor.jpg" usemap="#tabor" alt="Klikmapa
- Tučapy, Červená Lhota, Soběslav" >
<map name="tabor">
     <area shape="rect" coords="230,135,270,155"
href="mapa.gif">
     <area shape="rect" coords="300,25,360,45"
href="http://home.pf.jcu.cz/~pepe/chousnik.htm"
alt="Choustník">
     <area shape="rect" coords="350,270,425,300"
href="http://home.pf.jcu.cz/~pepe/clhota.htm"
alt="Červená Lhota">
     <area shape="rect" coords="100,169,161,202"
href="stan.jpg" alt="Stan">
```

**Vysvětlení:**

Nejprve musíme načíst obrázek **<IMG SRC="tabor.jpg" usemap=#tabor alt="Klikmapa - Tučapy, Červená Lhota, Soběslav" >**. Atribut src nám zajistí načtení příslušného obrázku. Neméně důležitým je atribut usemap, který nám zajistí propojení s tagem <map>. Jeho hodnota musí být uvozena mřížkou (#).

V druhé fázi musíme dokončit spojení mezi obrázkem a tagem <map>. Tag <map> musí obsahovat atribut name, jehož hodnota musí být shodná s hodnotou atributu usemap.

V poslední fázi musíme nadefinovat souřadnice a odkazy pro oblast obrázkové mapy pomocí tagu <area>. Ostatní atributy vysvětlím v kapitole 8.12.3.

### 8.12.2 Obrázkové mapy v XHTML

V jazyce XHTML můžeme také nadefinovat obrázkové mapy a to pomocí tagu <map> nebo s použitím tagu <object>. V příkladě 1 si ukážeme využití tagu map pomocí XHTML 1.1. V druhém příkladě si ukážeme nadeklarování obrázkových map pomocí tagu <object>.

**Příklad 1:**

```
<img src="tabor.jpg" usemap="#tabor" alt="Klikmapa
- Tučapy, Červená Lhota, Soběslav" />
<map name="tabor">
<area shape="rect" coords="230,135,270,155"
href="mapa.gif" />
<area shape="rect" coords="300,25,360,45"
href="http://home.pf.jcu.cz/~pepe/chousnik.htm"
alt="Choustník" />
<area shape="rect" coords="350,270,425,300"
href="http://home.pf.jcu.cz/~pepe/clhota.htm"
alt="Červená Lhota" />
<area shape="rect" coords="100,169,161,202"
href="stan.jpg" alt="Stan" />
</map>
```

**Příklad 2:**

```
<object data="tabor.jpg" shapes="shapes">
<a shape="rect" coords="230,135,270,155"
href="mapa.gif"></a>
<a shape="rect" coords="300,25,360,45"
href="http://home.pf.jcu.cz/~pepe/chousnik.htm"
></a>
<a shape="rect" coords="350,270,425,300"
href="http://home.pf.jcu.cz/~pepe/clhota.htm" ></a>
<a shape="rect" coords="100,169,161,202"
href="stan.jpg"></a>
</object>
```

**Upozornění:**

Internetovské prohlížeče umí zobrazit pouze příklad 1. Příklad 2 má podle oficiální verze XHTML 1.1 fungovat, ovšem prohlížeče ho zatím nepodporují.

### 8.12.3 Obrázkové mapy v XHTML 2

Jazyk XHTML 2 využívá znalostí jazyka XHTML 1.1. V této verzi jazyka můžeme vytvořit obrázkové mapy dvěma způsoby. Starší způsob

využívá tag <object>. V XHTML 2 se doporučuje používat nový způsob, který ukáži v příkladě.

Obrázkové mapy v XHTML 2 mohou mít následující atributy:

- **usemap** – Propojí element s obrázkovou mapou. Hodnota tohoto atributu musí odpovídat hodnotě atributu id.
- **isemap** – Pokud je použit tento atribut, elementem se zachází jako s obrázkovou mapou pracující na straně serveru. To znamená, že se na server pošlou souřadnice myší.
- **shape** – Udává tvar aktivní oblasti. Může mít následující hodnoty: default, rect (definuje čtverec), circle (definuje kruh), poly (definuje n-úhelník).
- **coords** – Atribut udává souřadnice aktivní oblasti.

**Příklad:**

```
<p src="tabor.jpg" usemap="#tabor" />
<nl id="tabor">
<li shape="rect" coords="230,135,270,155"
href="mapa.gif"></li>
<li shape="rect" coords="300,25,360,45"
href="http://home.pf.jcu.cz/~pepe/chousnik.htm">
</li>
<li shape="rect" coords="350,270,425,300"
href="http://home.pf.jcu.cz/~pepe/clhota.htm" >
</li>
<li shape="rect" coords="100,169,161,202"
href="stan.jpg"> </li>
</nl>
</p>
```

**Vysvětlení:**

Obrázkové mapy získávají v XHTML 2 nový náboj. Konsorcium W3C je uvedlo v kombinaci s tvorbou menu. Tento způsob si získá celou řadu příznivců. Tag <img> je nahrazen tagem <p>. Atribut id nahradil atribut name. Bohužel tento způsob zatím nepodporují internetové prohlížeče.

## 8.13 „Zrychlené“ odkazy

V jazycích HTML a XHTML se tvořili odkazy následujícím způsobem.

**Příklad 1:**

```
<a href="http://vojtechsoukup.wz.cz"><strong>
Státnice </strong></a>
```

V jazyku XHTML 2 byl atribut href zařazen do Common Atribute Collection (kolekce běžných atributů). Což znamená jediné, každý element může být zároveň odkazem. V příkladě 2 si ukážeme jak může tvorba odkazů vypadat. Ovšem ani tag <a> není zrušen. Nakonec zbude na uživateli jakou možnost tvorby odkazů použije.

**Příklad 2:**

```
<strong href="http://vojtechsoukup.wz.cz"> Státnice
</strong>
```

**Upozornění:**

Prohlížeče zatím nepodporují tvorbu odkazů novým způsobem.

## 8.14 Vypuštěné tagy a atributy z XHTML 2

### 8.14.1 Zrušené tagy

V této kapitole uvedu seznam tagů, které jsou z jazyka XHTML 2 vypuštěny: <acronym>…..</acronym>, <applet>…..</applet>, <base>, <basefont>, <bdo>…..<bdo>, <bgsound>, <big>…..</big>, <blink>…..</blink>, ….. , <comment>…..</comment>, <del>…..</del>, <embed>, <font>…..</font>, <frame>…..</frame>, <frameset>…..</frameset>, <h1>…..</h1>, <h2>…..</h2>, <h3>…..</h3>, <h4>…..</h4>, <h5>…..</h5>, <h6>…..<h6>, ….., <iframe>…..</iframe>, <ilayer>…..</ilayer>, <img>, <ins>…..</ins>, <isindex>, <layer>…..</layer>, <legend>…..</legend>, <marquee>…..</marquee>, <menu>…..</menu>, <multicol>…..</multicol>, <nobr>…..</nobr>, <noframes>….</noframes>, <q>…..</q>, <s>…..</s>,

<server>….</server>, <small>…..</small>, <strike>…..</strike>, <u>….</u>, <xmp>…..</xmp>.

Většinu těchto tagů jsou úmyslně vypuštěny. Konsorcium W3C chce všechny donutit, aby používali CSS styly.

### 8.14.2 Zrušené atributy

Odnětím zakázaných značek, resp. jejich nahrazením značkami povolenými a CSS vaše práce zdaleka nekončí. Zbývající povolené značky totiž mají mnoho zakázaných atributů. Většinou jsou to tzv. prezentační atributy (jejichž jediným účelem je ovlivnit vzhled dokumentu). Podle této vlastnosti si je můžete snadněji zapamatovat. Atribut name byl plně nahrazen atributem id. Seznam zrušených atributů: target, name, bgcolor, color, link, vlink, novrap, background, alink, leftmargin, topmargin, valign, width, height, align, border, hspace, vspace, size, bordercolor, framespacing, rows.

# 9 Závěr

V mojí bakalářské práci jsem měl za úkol seznámit čtenáře se značkovacími jazyky a zaměřit se na nově vzniklý jazyk XHTML 2.

XHTML 2 vychází ze značkovacího jazyka XHTML 1.1. Předpokládal jsem základní znalosti jazyka HTML, a proto jsem se více zaměřil na XHTML 2. Snažil jsem nové tagy ozřejmit. Jazyk XHTML 2 je však sám o sobě velice rozsáhlý a nebylo mým cílem jej celý obsáhnout. Vybral jsem z něj kapitoly, které napomohou pochopení problematiky XHTML 2.

XHTML 2 obsahuje elementy rozdělené do různých nezávislých modulů. V jednom modulu budou jen elementy pro formátování textu, v druhém pro tvorbu tabulek, ve třetím pro odkazy, další pro formuláře, obrázky atd.

Moduly nejsou jedinou novinkou. Protože každé výstupní zařízení má jiné schopnosti, bude zapotřebí je nějakým způsobem sjednotit do tříd. V jednotlivých třídách budou zařízení se stejnými schopnostmi a bude pro ně vytvořen jeden profil, který bude obsahovat informace o možnostech těchto zařízení. V profilu by měl být uveden seznam modulů, které jsou podporovány, jejich DTD, seznam grafických a dalších formátů důležitých pro vložení různých objektů.

Každý dokument by měl mít svůj vlastní profil. Prohlížeč by pak měl spolu s požadavkem na nějakou stránku poslat i svůj profil (informace, co dané zařízení zvládne zobrazit). Když bude stránka dostupná ve více variantách, vybere server jen tu s odpovídajícím profilem, případně může stránku automaticky konvertovat pro profil klienta.Tento princip umožní vyvíjet stránky pro mnoho zařízení s rozdílnými schopnostmi (PC, mobilní telefony aj.).

Myslím, že jsem čtenářům nastínil výhody i nevýhody jazyku XHTML 2. Uvedl jsme ukázky zdrojového kódu dokumentů v různých formátech, navzájem jsem je porovnával. Snažil jsem se poukázat na hlavní rozdíly mezi nimi. Posouzení, zda se mi to povedlo, nechám na čtenáři.

# 10 Seznam zkratek a pojmů

| | |
|---|---|
| **ASCII** | *American Standard Code for Information Interchange* – Americký standardní kód pro výměnu informací, osmibitová abeceda tvořící základ pro většinu dnes používaných abeced pro kódování písmen a číslic |
| **CDATA** | *Character DATA* - Datový typ, data, která se nemají analyzovat. Jsou to prosté znakové řetězce. Typ CDATA je implicitně předpokládán pouze u atributů. |
| **CSS** | *Cascading Style Sheet* - Kaskádní tabulky (sešity) stylů, které vznikly v HTML 4.0. V současné době existují specifikace CSS1 a CSS2. |
| **DTD** | *Document Type Definition* – Definice typu dokumentu. Datová struktura sloužící k popisu elementů, atributů a entit .Je to formální definice pravidel pro značkování. Tato definice je určena tagem `<!DOCTYPE...>` |
| **HTML** | *HyperText Markup Language* – jazyk sloužící k popisu webových stránek. Má dvě důležité složky: hypertext – odkazy na externí elementy a markup – využívání značek (elementů) pro formátování dokumentu. |
| **Metajazyk** | Definuje pravidla a symboly jiných jazyků. |
| **MSIE - Internet Explorer** | WWW prohlížeč firmy Microsoft. Podporuje XML a XSL analýzu i technologii DOM, s jejíž pomocí lze vytvořit program na ověření platnosti dokumentu. Podporuje i pseudoelementy CSS. |
| **ISO** | *International Organization for Standardization* – Mezinárodní normalizační organizace |
| **#PCDATA** | *Parsed Character DATA* – Analyzovaná znaková data. Elementy, které obsahují tento typ dat, mohou obsahovat prostý text i další povolené tagy podle určené definice DTD. |
| **SGML** | *Standard Generalized Markup Language* – Standardizovaný obecný značkovací jazyk určený k formálnímu popisu struktury dokumentů. |
| **Soubor HTML** | Textový soubor obsahující kód jazyka HTML. |
| **WAP** | *Wireless Application Protocol* - Protokol používaný pro bezdrátovou výměnu informací mezi sítí internet a mobilním telefonem (klientem). pro přenos se používá jazyk WML |
| **Webová stránka, WWW stránka** | Dokument, který je uložen na webovém serveru ve svých základních elementech. Je načítán a zobrazován prohlížečem. |

| | |
|---|---|
| **WFC** | *Well-formedness constraint* – Omezení kladená na dobře vytvořený dokument. |
| **WML** | *Wireless Markup Language* - Značkovací jazyk, který je pomocí speciálního DTD odvozen z jazyka XML. Používá se ke značkování dat protokolu WAP (bezdrátový přenos dat). |
| **WWW** | *World Wide Web* – V češtině tento výraz označuje pavučinu, tedy celosvětovou pavučinu. Využívá technické struktury Internetu (počítače v celosvětové síti) a zprostředkovává informace uložené na těchto počítačích v atraktivní grafické podobě. |
| **W3C** | *WWWC - World Wide Web Consortium* - Konsorcium zabývající se standardizací internetových technologií pro WWW. |
| **XHTML** | *eXtensible HyperText Markup Language* – Rozšiřitelný hypertextový značkovací jazyk. Kříženec HTML a XML jazyků. |
| **XHTML 2** | *eXtensible HyperText Markup Language – Značkovací jazyk vycházející z jazyka XHTML 1.1.* |
| **XML** | *eXtensible Markup Language* – Rozšiřitelný značkovací jazyk odvozený z SGML. |

# 11 Literatura a ostatní použité zdroje

**[1]** HLAVENKA, J. a kol.: Vytváříme WWW stránky a spravujeme moderní web site, 1. vyd. Praha, Computer Press 1997, počet str. 393, ISBN 80-7226-039-1

**[2]** Kučera, M.: HTML – tipy a triky od profisionálů, UNIS Publishing, s. r. o. 2001, počet stran 80, ISBN 80-86097-64-1

**[3]** Musciano, Ch. a Kennedy, B.: HTML a XHTML Kompletní průvodce, 1. vyd., Computer Press 2000, počet str. 633, ISBN 80-7226-407-9

**[4]** Pexa, P.: Tvorba WWW a WAP, nakladatelství Kopp 2001, počet stran 215, ISBN 80-7232-161-7

**[5]** Písek, S.: HTML a XHTML (začínáme programovat), Grada Publishing 2003, počet stran 256, ISBN 80-247-0571-0

## Internetové zdroje

- http://www.w3c.org
- http://www.kosek.cz
- http://www.interval.cz
- http://www.kosek.cz/clanky/
- http://www.site-builder.co.uk/xhtml.htm
- http://www.xhtml.org/
- http://wdvl.com/Authoring/Languages/XML/XHTML/
- http://www.w3.org/TR/xhtml1/
- http://www.w3schools.com/xhtml/default.asp
- http://www.simonstl.com/xhtml/index.html

- http://xhtml.waptechinfo.com/extxhtml.html
- http://indy.cs.concordia.ca/kamthan/publ/html-xml/index.html
- http://www.xhtmlguru.com/articles.htm
- http://www.encyclozine.com/WD/XHTML
- http://noviny.trafika.cz/chip/2000/000401/chip120.html
- http://encyclozine.com/WD/XHTML
- http://www.jakpsatweb.cz/meta_tagy.htm
- http://www.w3.org/TR/xhtml2/
- http://www.w3.org/TR/xhtml1/
- http://www.pcsvet.cz/art/article.php?id=4077
- http://www.pf.jcu.cz/~pexa/
- http://www.japsatweb.cz/meta tagy.htm