Tvorba internetových aplikací v XHTML 2.0, XForms a XHTML Print

Bakalářská práce
Adam Zluky
Vedoucí bakalářské práce: PeadDr. Petr Pexa
Jihočeská univerzita v Českých Budějovicích
Pedagogická fakulta
Katedra informatiky
Rok 2008

Prohlášení

Prohlašuji, že svoji bakalářskou práci jsem vypracoval/-a samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury.

Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své bakalářské/diplomové práce, a to v nezkrácené podobě pedagogickou fakultou elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách.

V Českých Budějovicích dne: 11. dubna 2008

Adam Zluky

Anotace:

Cílem této bakalářské práce je vytvoření uživatelské příručky v jazycích XHMTL 2.0, XForms a XHTML Print. Jelikož jazyku XHTML 2.0 v minulosti předcházelo několik jiných značkovacích jazyků, v úvodu práce se budu věnovat jim, jejich využití, nedostatkům a výhodám. V další části, bych se věnoval samotnému stavu pracovního návrhu jazyka XHTML 2.0, jeho syntaxi a podpoře v internetových prohlížečích Internet Explorer 7, FireFox 2.0 a Opera 9.0. Jazyku XForms, jakožto náhradě HTML formulářových prvků, a XHTML Print, zaměřenému na tiskový výstup budou věnovány další samostatné kapitoly.

Abstract:

The aim of this bachelor work is to create a user guide in languages XHMTL 2.0, Xforms and XHTML Print. Due to the fact that the XHTML 2.0 language follows several previous marking languages, their usage, advantages and disadvantages are discussed in the introduction of the work. In the next part I'm going to deal with the state of the concept of the XHTML 2.0 language, with its syntax and support in internet browsers Internet Explorer 7, Fire Fox 2.0 and Opera 9.0. The languages XFORMS, as a substitution of HTML form components, and XHTML Print, focusing on print output, will be discussed in the subsequent chapters.

Poděkování:

Rád bych poděkoval PeadDr. Petru Pexovi za odborné a organizační vedení při zpracování této práce.

Obsah

1. Úvod

Tato bakalářská práce se zabývá značkovacími jazyky pro tvorbu internetových stránek a aplikací.

Jak je v práci popsáno, značkovacích jazyků je celá řada. Cílem této publikace je zmapovat některé nejmodernější značkovací jazyky. Práce však není soustředěná pouze na dnešek. Svojí koncepcí s pohledem do minulosti i budoucnosti se snaží o kritický pohled na značkovací jazyky v době, kdy nejpoužívanější specifikace XHTML 1.0 Strict je stará již téměř deset let a nová verze jazyka je prozatím nepoužitelná.

Tato bakalářská práce má několik částí. Kapitoly zabývající se tradičními značkovacími jazyky a jazykem XHTML 2.0, jsou více teoretické. Oproti tomu kapitoly věnující se technologii XForms a XHTML-Print jsou zaměřeny prakticky s množstvím ukázek zdrojových kódu přímo v textu a s ukázkami řešení konkrétních příkladů, na které bychom mohli narazit při psaní webových stránek a aplikací.

Součástí bakalářské práce je také příloha, obsahující všechny zmíněné ukázky ve formě konkrétních webových dokumentů.

Tato práce je určena čtenářům, kteří mají zkušenosti se značkovacími jazyky (X)HTML a CSS a chtějí si chtějí udělat obrázek o současném stavu návrhů některých jazyků z dílny konsorcia W3C a zorientovat se ve stávajících, minulých a možná i budoucích značkovacích jazycích.

2. Tradiční značkovací jazyky

V úvodní kapitole nazvané tradiční značkovací jazyky popíši čtenáři jednotlivé jazyky, které byly v průběhu posledních tří desetiletí používány pro tvorbu elektronických dokumentů. Několik těchto jazyků se dnes již používá sporadicky, část z nich je v současné době moderním standardem a některé jsou hudbou budoucnosti.

V každé kapitole je zmíněna stručná historie jazyka a jeho návaznost na starší technologie. Dále pak odkaz na organizaci spravující standardy toho konkrétního jazyka, zpravidla je to konsorcium W3C. A konečně pojednání o syntaxi a ukázka zdrojového kódu.

2.1. SGML

Prvním jazykem pro tvorbu elektronických dokumentů byl jazyk SGML (*Standard Generalized Markup Language*). Jeho vznik je spojen s projektem ODA (*Open Document Architecture*). Na přelomu 70. a 80. let 20. století vznikla potřeba standardizovaného formátu elektronických dokumentů pro přenášení, zálohování, zpracování dat, který by byl nezávislý na softwarové i hardwarové platformě.

Jazyk SGML vznikl spojením dvou jazyků a to GenCode a GML. Z jazyku GenCode převzal sémantická pravidla (přisuzující význam přípustným kombinacím symbolů) a jazykem GML se inspiroval po stránce syntaktické (pravidla pro vytváření přípustných kombinací symbolů). V roce 1986 byl přijat mezinárodní organizací pro standardy pod názvem ISO 8879.

V jazyce SGML popisujeme pouze obsah a strukturu dat, nikoliv vzhled. To z něj dělá nástroj vhodný pro prezentaci na různých výstupních zařízeních jako je tiskárna, webový server a jiné.

SGML byl navržen velmi univerzálně. Umožňuje definici vlastních značkovacích jazyků. V samostatném textovém souboru jsou definovány přípustné elementy a vztahy mezi nimi označované jako DTD (*Document Type Definition*). Samotný SGML dokument pak využívá těchto elementů. SGML soubor je uložen v ASCII (*American Standard Code for Information Interchange*), díky tomu je použitelný téměř na jakékoliv platformě. Tento obecný přístup způsobil, že SGML dokumenty byly značně složité v implementaci a měli malou podporu softwarových produktů. Toto zabránilo jeho masovějšímu rozšíření.

V současné době stále existují aplikace a projekty využívající jazyka SGML. Jedná se o rozsáhlé struktury ve vládních a akademických kruzích. Jejich neúplný seznam je k nahlédnutí na http://xml.coverpages.org/acadapps.html a http://xml.coverpages.org/gov-apps.html.

Zdrojový kód jazyka SGML si ukážeme na příkladu sbírky básní. Demonstrujeme na něm vlastní strukturu i význam jednotlivých elementů.

```
<sbirka>
      <basen>
            <nazev>Svíčka</nazev>
            <sloka>
                  <vers>Hořela svíčka na poli,</vers>
                  <vers>sfouknout ji mohl kdokoli.</vers>
            </sloka>
            <sloka>

                  <vers>I všiml si jí vítr, krutý
                  samovládce,</vers>
                  <vers>hořela svíčka krátce.</vers>
            </sloka>
      </basen>
      <basen>
      ...
      </basen>
</sbirka>
```

Soubor s DTD bude mít následující strukturu.

```
<!ELEMENT sbirka        - - (basen+)>
<!ELEMENT basen         - - (nazev?, sloka+)>
<!ELEMENT nazev         - O (#PCDATA)>
<!ELEMENT sloka         - O (vers+)>
<!ELEMENT vers          O O (#PCDATA)>
```

Nový element deklarujeme klíčovým slovem !ELEMENT a názvem elementu. Je na jednom řádku a ohraničen ostrými závorkami. Za názvem elementu je výčet jeho vlastností. První dva symboly značí začáteční a koncový tag, přičemž symbol - znamená, že tag musí být uveden a symbol **O**, že tag je volitelný. V závorkách je pak uveden seznam vnořených elementů, které v definovaném mohou být obsaženy. Za názvem vnořeného elementu se ještě vyskytuje symbol +, který znamená jeden a více výskytů, nebo symbol **?**, značící žádný nebo jeden výskyt, nebo konečně symbol --, který určuje žádný, jeden, nebo více výskytů. Pokud je v závorkách uvedeno klíčové slovo #PCDATA, znamená to, že element obsahuje text, tj. obsah dokumentu.

2.2. HTML

Právě potřeba zjednodušení kódu vedla ke vzniku jazyka HTML (*HyperText Markup Language*). Jeho autorem je Tim Berners-Lee a jeho první verzi vytvořil v roce 1991. Tim Berners-Lee je v současné době [1] ředitelem konsorcia W3C.

Základem pro jazyk HTML se stal SGML. Bylo navrženo DTD (http://www.w3.org/MarkUp/html-spec/html.dtd), které umožňovalo použití základního formátování textu, jako velikost a tučnost písma, odrážky, nadpisy a odřádkování. Dále pak ještě možnost vkládání obrázků a formulářových prvků a samozřejmě značky pro hypertextové odkazy.

Tehdejší webové prohlížeče měli za povinnost ignorovat elementy a atributy, které neznají, resp. které nejsou definovány. Další vývoj a verze jazyka HTML pak zčásti ovlivňovali právě tvůrci internetových prohlížečů, kteří dávali návrhy na možné elementy, ty byly posléze přijaty, či zamítnuty.

Nejvýrazněji zasáhly firmy Netscape s prohlížečem Navigator a Microsoft s Explorerem.

Určitá benevolence ze strany prohlížečů, jako odhadování chybějícího koncového tagu, či nesprávné vnořování elementů, jež bylo u SGML zakázáno, však vedla k nejednotnosti HTML dokumentů. Na druhou stranu právě tato vlastnost přispěla k obrovskému rozšíření jazyka HTML mezi amatérskými tvůrci stránek.

Nejznámější verzí jazyka HTML je specifikace HTML 4.01 (http://www.w3.org/TR/html4/) z roku 1999.

Ukázkový příklad určuje strukturu zdrojového kódu. Hlavní stavební kameny webové stránky jako `HTML HEAD TITLE` a `BODY` zůstávají i dnes po téměř deseti letech od vydání specifikace.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
   "http://www.w3.org/TR/html4/strict.dtd">
<HTML>
   <HEAD>
      <TITLE>Dokument v HTML 4.01</TITLE>
   </HEAD>
   <BODY>
      <P>Odstavec
   </BODY>
</HTML>
```

2.3. WML

Předtím než začneme s popisem jazyka WML (*Wireless Markup Language*), musíme se seznámit s pojmem WAP (*Wireless Application Protocol*). Zjednodušeně řečeno, WAP je služba podobná WWW, ale komunikace probíhá mezí serverem a mobilním telefonem. O WAPové standardy se stará organizace WAP FORUM (http://www.wapforum.org/). První specifikace protokolu WAP byla zveřejněna v roce 1998.

Mobilních telefonů umožňujících připojení na wapové stránky a jejich zobrazení je v dnešní době snad ještě více, než těch, které tuto službu

nepodporují. Problém, proč wap nikdy nedosáhl v české republice masového rozšíření, je podle mě v jeho zpoplatnění a malé reklamě. Přitom potenciál bezpochyby měl. Informace z internetu dostupné všude tam, kde je dostupný signál mobilního telefonu, jsou jistě silnou devizou. V době psaní této práce je již mobilní webový prohlížeč, který je schopný zobrazit plnohodnotné www stránky, běžnou součástí nejednoho mobilního telefonu a tak je otázkou, jaká je budoucnost protokolu WAP.

Samotný jazyk WML je velice podobný jazyku (X)HTML. Některé elementy, jako například `<p>` pro odstavec, `<b>` pro tučné písmo nebo `<a>` pro odkaz, jsou dokonce stejné. Soubor s příponou wml začíná XML deklarací a použitým DTD. Celý WML dokument je uvozen tagem `<wml>`. Jednotlivé stránky WAPové prezentace jsou značeny jako karty `<card>`. Karty jsou identifikovány atributem `id`. Mohou být v jednom i ve více souborech.

Ukázka WAPové stránky v jazykce WML.

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">
<wml>
<card id="index" title="Index">
<p align="left">
<small>
<b>Restaurace U Karásků</b><br/><br/>
<a href="cenik.wml"><i>Naše ceny</i></a><br/>
<a href="doba.wml"><i>Kdy máme otevřeno.</i></a><br/>
<a href="ubytovani.wml"><i>Možnost ubytování</i></a><br/>
</small>
</p>
</card>
</wml>
```

2.4. XHTML 1.0

XHTML (*eXtensible HyperText Markup Language*) je dalším členem rodiny značkovacích jazyků. Specifikace XHTML 1.0 byla vydána 26.1.2000 a

její znění najdeme na adrese http://www.w3.org/TR/xhtml1/. Upravuje jazyk HTML 4.01 a to tím způsobem, že nastavuje přísnější pravidla pro deklarace jednotlivých elementů a je patrný určitý posun ke XML. Dále se dělí na tři verze. Striktní (*strict*), přechodovou (*transitional*) a s použitím rámců (*frameset*).

Nejprve se podíváme podrobněji na přísnější pravidla pro psaní webových stránek. Uvedeme si ukázku v HTML 4.01 a v XHTML 1.0

Nesprávné vnořování elementů HTML umožňuje. V XHTML je zakázané.

HTML

```
<b><i>tučná kurzíva</b></i>
```

XHTML

```
<b><i>tučná kurzíva</i></b>
```

Všechny názvy elementů a atributů musí být malými písmeny. Jazyk XHTML je citlivý na velikost písmen (*case sensitive*)jako samotné XML.

HTML

```
<TITLE>Webová prezentace</TITLE>
```

XHTML

```
<title>Webová prezentace</title>
```

Nutnost uzavíracích elementů. Prohlížeče v HTML tolerovali nepřítomnost ukončovacích tagů a často jej také správně interpretovali, když našli následující začáteční element.

HTML

```
<p>První odstavec
<p>Druhý odstavec
```

XHTML

```
<p>První odstavec</p>
<p>Druhý odstavec</p>
```

Uzavírání parametrů atributů do uvozovek.

HTML

```
<hr align=center>
```

XHTML

```
<hr align="center">
```

Zákaz bezhodnotových atributů.

HTML

```
<input type="checkbox" checked>
```

XHTML

```
< input type="checkbox" checked="checked"
```

Nepárové tagy musí být ukončeny. Stejně jako v XML, pokud chceme zapsat prázdný element můžeme to učinit dvěma způsoby. `<jmeno></jmeno>`, nebo `<jmeno/>`. XHTML si zvolila druhý způsob, přičemž se stalo dobrým zvykem před ukončovací lomítko vložit mezeru.

HTML

```
<br>
```

HTML

```
<br />
```

XML deklarace. Na začátku XHTML dokumentu musí být XML deklarace o použitém kódování.

XHTML

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

Další podmínkou validního XHTML dokumentu je nutnost deklarace `DOCTYPE` před samotným elementem `html`. Varianta XHTML Transitional nutí webmastera používat výše vypsaná pravidla a její deklarace vypadá takto.

```
<?xml version="1.0" encoding="windows-1250"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="cs"
lang="cs">
   <head>
     <meta http-equiv="content-type" content="text/html;
charset=windows-1250" />
     <title>Dokument v XHTML Transitional</title>
   </head>
<body>
<p>
      Odstavec.
</p>
</body>
</html>
```

Verze XHTML Strict nejen, že se musí řídit výše vypsanými pravidly, ale ještě veškeré formátování musí být provedeno použitím kaskádových stylů. Což prakticky znamená, že elementy nesmí obsahovat atributy.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="cs"
lang="cs">
   <head>
     <meta http-equiv="content-type" content="text/html;
charset=utf-8" />
     <title>Dokument v XHTML Strict</title>
     <link rel="stylesheet" href="styl.css" type="text/css"
media="all" />
   </head>
<body>
<p>
      Odstavec.
</p>
</body>
</html>
```

Poslední možností DOCTYPE v XHTML 1.0 je Frameset. To uvažuje webovou stránku s použitím rámců.

```
<?xml version="1.0" encoding="windows-1250"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
```

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="cs"
lang="cs">
   <head>
     <meta http-equiv="content-type" content="text/html;
charset=windows-1250" />
     <title>Dokument v XHTML Frameset</title>
   </head>
<frameset>
</frameset>
</html>
```

Závěrem této podkapitoly bych rád zmínil, že výše popsaná pravidla pro tvorbu hypertextového dokumentu v XHTML 1.0 jsou závazná pro splnění validity stránky, přesto však prohlížeče spoustu prohřešků tolerují a zpravidla je zobrazí tak, „jak jsme zamýšleli.“

Otázkou zůstává, zda je to ze strany tvůrců prohlížečů krok správným směrem. Mám na mysli problematiku toho, že www stránka bude zobrazena, i když není napsaná striktně a není validní. Podle mého názoru je určitá liberálnost na místě. Jak jsem již zmínil v podkapitole HTML, internet, potažmo tvorba www stránek se stala tím čím je právě díky relativní jednoduchosti tvorby a publikování a možná i určité toleranci k začátečnickým syntaktickým chybám.

Navíc možnost umístit s čistým svědomím na svůj web ikonu XHTML a CSS valid, opatřenou odkazem na nějaký online validátor, je podle mě vhodný prostředek pro prezentaci kvalitního kódu.

Nicméně tato práce se zabývá webovými standardy, proto otázka validity je pro následující texty samozřejmostí.

2.5. XHTML 1.1

Poslední verze byla zveřejněna v únoru 2007 a to v pracovním návrhu Working Draft.

Specifikace XHTML 1.1 je založena na XHTML modulech (http://www.w3.org/TR/2006/WD-xhtml-modularization-20060705/).

Modularizace dovoluje používat XML DTD a XML schémata, nedefinuje sémantiku elementů a atributů, ale pouze jejich rozdělen do modulů. Tyto moduly jsou pak využívány jednotlivými značkovacími jazyky.

Dokument napsaný ve značkovacím jazyku XHTML 1.1 musí mít stejně jako v předchozí verzi 1.0 kořenový element `html`. Před ním musí být informace o použitém XML kódování a deklarace DOCTYPE. Element `html` může obsahovat deklaraci několika jmenných prostorů ve tvaru prefix:název a adresa použitého jmenného prostoru.

Kostra dokumentu v jazyce XHTML 1.1

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
    "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.w3.org/MarkUp/SCHEMA/xhtml
11.xsd"
     xml:lang="en" >
<head>
<meta http-equiv="content-type" content="application/xhtml+xml;
charset=utf-8" />
<title>Dokument v XHTML 1.1</title>
  </head>
  <body>
    <p>Ahoj světe.</p>
  </body>
</html>
```

Na první pohled se tato kostra příliš neliší od kostry XHTML 1.0 Strict. Z velké části z ní také vychází, především ve smyslu oddělení formy od obsahu a maximálního využití kaskádových stylů. Syntaktické rozdíly oproti XHTML 1.0 Strict jsou pouze tři.

atribut `lang` je nahrazen atributem `xml:lang`

v elementech `a` a `map` byl atribut `name` nahrazen atributem `id`

byla přidána kolekce elementů `ruby`

Jak již bylo řečeno dříve elementy jsou sdružovány do modulů podle určitých funkčních vlastností.

Modul struktury: `body, head, html, title`

Textový modul: `abbr, acronym, address, blockquote, br, cite, code, dfn, div, em, h1, h2, h3, h4, h5, h6, kbd, p, pre, q, samp, span, strong, var`

Hypertextový modul: `a`

Modul seznamů: `dl, dt, dd, ol, ul, li`

Objektový modul: `object, param`

Prezentační modul: `b, big, hr, i, small, sub, sup, tt`

Editační modul: `del, ins`

Obousměrný textový modul: `bdo`

Formulářový modul: `button, fieldset, form, input, label, legend, select, optgroup, option, textarea`

Tabulkový modul: `caption, col, colgroup, table, tbody, td, tfoot, th, thead, tr`

Obrázkový modul: `img`

Modul klientských obrazových map: `area, map`

Modul serverových obrazových map: `atribut ismap v elementu img`

Modul vnitřních událostí: `atributy událostí`

Modul metainformací: `meta`

Modul skriptů: `script, noscript`

Modul stylů: `element style`

Modul atributu style: `atribut style (překonaný)`

Modul link: `link`

Modul base: `base`

Modul vysvětlivek Ruby: `ruby, rbc, rtc, rb, rt, rp`

Poslední důležitá poznámka k jazyku XHTML 1.1 se bude týkat MIME (*Multipurpose Internet Mail Extention*) typů. MIME typ je definovaný dvěma částmi oddělenými lomítkem. Část před lomítkem určuje hlavní typ souboru, část za lomítkem podtyp. Určuje prohlížeči v jakém formátu jsou data odesílána.

Do specifikace XHTML 1.0 byla stránka odesílána jako text/html. V jazyce XHTML 1.1 a XHTML 2.0 však už musí být stránka odesílána jako application/xhtml+xml. Je to z toho důvodu, že tyto dva jazyky jsou striktně založeny na xml.

Použití MIME typu `application/xhtml+xml` přináší několik zajímavých vlastností. Především nedovolí zobrazit stránku pokud není napsaná podle syntaktických pravidel XML. V případě chyby XML parser pouze zobrazí informaci o tom ,kolikátý znak na kterém řádku způsobuje problém parsování. Tento způsob v podstatě zamezuje publikování nevalidní stránky.

MIME typ `application/xhtml+xml` se nastaví ve dvou krocích. Zaprvé je to vložení meta řádku s příslušnou deklarací a zadruhé uložení dokumentu ve formátu *.xhtml.

Takto napsaný dokument nezobrazí prohlížeč Internet Explorer ani ve verzi 7. A ačkoliv internetový prohlížeč Mozilla FireFox 2.0 a Opera 9 stránku zobrazí správně, právě omezení se strany Microsoftu dělá značkovací jazyk XHTML 1.1 v současné době nepoužitelný.

2.6. CSS

CSS (*cascading style sheet*) v překladu znamená kaskádové stylové předlohy. Je to jazyk , o jehož standardy se stará konsorcium W3C (http://www.w3.org/Style/CSS/). Jazyk CSS je v současné době ve své třetí verzi CSS3.

Tato technologie by mohla být popsána minimálně třemi cíly. Oddělením formy od obsahu, snadným zobrazením téže webové stránky na různých výstupních zařízeních a v neposlední řadě použitím nových možností formátování stránky s příklonem k DTP (*DeskTop Publishing*)

Kaskádový styl můžeme k dokumentu připojit třemi způsoby.

1. In-line.

Styl je deklarován přímo v elementu. To umožňuje použití pokročilého formátování, avšak nijak neodděluje formu od obsahu. Navíc pro každý element musí být deklarace znovu napsána.

```
<p style="background-color:#f2f">
Odstavec s růžovým pozadím.
</p>
```

2. Globální styl.

CSS vlastnosti elementů se deklarují pro celou stránku najednou. Tento způsob sice také neodděluje formu od obsahu, ale nemusíme již pro každý element deklarovat styl zvlášť.

```
<style type="text/css">
      p {background-color:#f2f
      }
</style>
</head>
<body>
      <p>
      První odstavec.
      </p>
      <p>
      Druhý odstavec.
      </p>
</body>
</html>
```

3. Externí soubor.

Deklarace kaskádových stylů v externím souboru a jeho následné připojení pro každou stránku webu umožňuje naprosté oddělení formy od obsahu. Navíc je možno pro každé výstupní zařízení (monitor, tiskárna, mobilní telefon) nadefinovat jiný styl.

HTML

```
<!-- připojení externího souboru  -->
<link rel="stylesheet" href="css_externi.css" type="text/css"
media="all" />
```

CSS

```
/* vlastnosti jednotlivých elementů v externím souboru *.css */
p{background-color:#f2f
      }
```

Jazyk XHTML 1.0 Strict a XHTML 1.1 vyžaduje aby veškeré formátování dokumentu bylo prováděno výhradně pomocí externího souboru CSS.

2.7. DHTML

DHTML (*Dynamic HyperText Markup Language*) je způsob rozšíření webové stránky o některé události. HTML stránka, kterou můžeme nazvat statickou, se po jejím načtení již nezmění. DHTML je způsob jak oživit HTML dokument o různé druhy animací, událostí a interakce s uživatelem použitím (X)HTML, CSS a jazyka JavaScript. JavaScript je objektově orientovaný interpretovaný jazyk.

Pro DHTML neexistuje žádný W3C standard. Jazykem DHTML je nazýváno pouze určité použití výše vypsaných prostředků.

Podobně jako u jazyka CSS existují tři způsoby, jakými lze do webové stránky vložit skript.

1. Zápis přímo do elementu.

```
<input type="button" value="OK" onClick="window.alert('Ahoj
světe.')">
```

2. Deklarace v hlavičce dokumentu elementem `script`.

Takový program bude spuštěn, když na něj prohlížeč narazí. V našem případě je deklarován v hlavičce dokumentu, proto bude vykonán ještě před načtením stránky.

```
<head>
<script>
  alert('Dialogové okno \n zobrazené před \n načtení
stránky.');
</script>
</head>
```

3. Vložení odkazu na externí soubor *.js

```
<head>
      <script type="text/javascript"
src="dhtml_externi_script.js">
</script>
</head>
```

Skriptovací jazyk JavaScript není využíván pouze pro zkrášlování stránky, ale i při práci s webovými formuláři a pokročilejším programováním webových stránek. Jedním z moderních využití jazyka se jeví jeho použití v technologii AJAX.

2.8. XML

XML (*eXtensible Markup Language*) je značkovací jazyk spravovaný konsorciem W3C (http://www.w3.org/XML/). Je určený primárně k uchování dat a jejich výměně s důrazem kladeným na strukturu. Neuchovává v sobě žádné informace o vzhledu dokumentu a veškeré formátování je prováděno pomocí jazyků CSS nebo XSL (*eXtensible Stylesheet Language*).

Nejnovější verze jazyka je ze srpna 2006 a nese označení XML 1.1. Z velké části využívá principy jazyka SGML, jak byly popsány v předchozích kapitolách.

Síla XML je ve strukturování dat. V následujícím příkladu uvažujeme obchod s potravinami. U každé potraviny nás bude zajímat název zboží, cena, výrobce a několik dalších parametrů. XML soubor, který uchovávající data, by mohl vypadat takto.

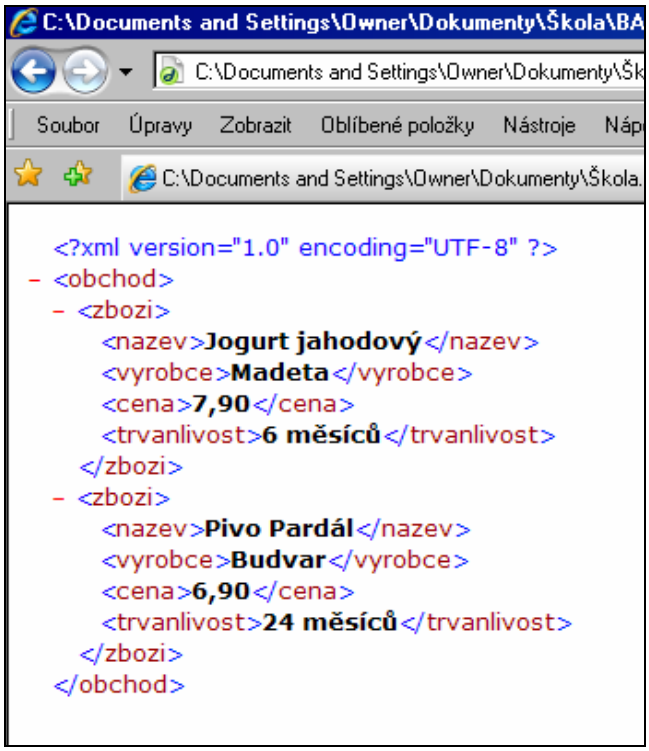
```
<?xml version="1.1" encoding="UTF-8" standalone=“no“?>
<obchod>
    <zbozi>
     <nazev>Jogurt jahodový</nazev>
     <vyrobce>Madeta</vyrobce>
     <cena>7,90</cena>
     <trvanlivost>6 měsíců</trvanlivost>
    </zbozi>
    <zbozi>
     <nazev>Pivo Pardál</nazev>
     <vyrobce> Budvar</vyrobce>
     <cena>6,90</cena>
     <trvanlivost>24 měsíců</trvanlivost>
    </zbozi>
</obchod>
```

Na prvním řádku je informace o použité verzi XML a o kódování. Element `xml` dále obsahuje nepovinný atribut `standalone=“yes|no“`, který dává informaci o tom, zda ke XML souboru jsou připojeny ještě nějaké další soubory, například CSS nebo XSL. Veškerá data jsou uzavřena kořenovým elementem `obchod`.

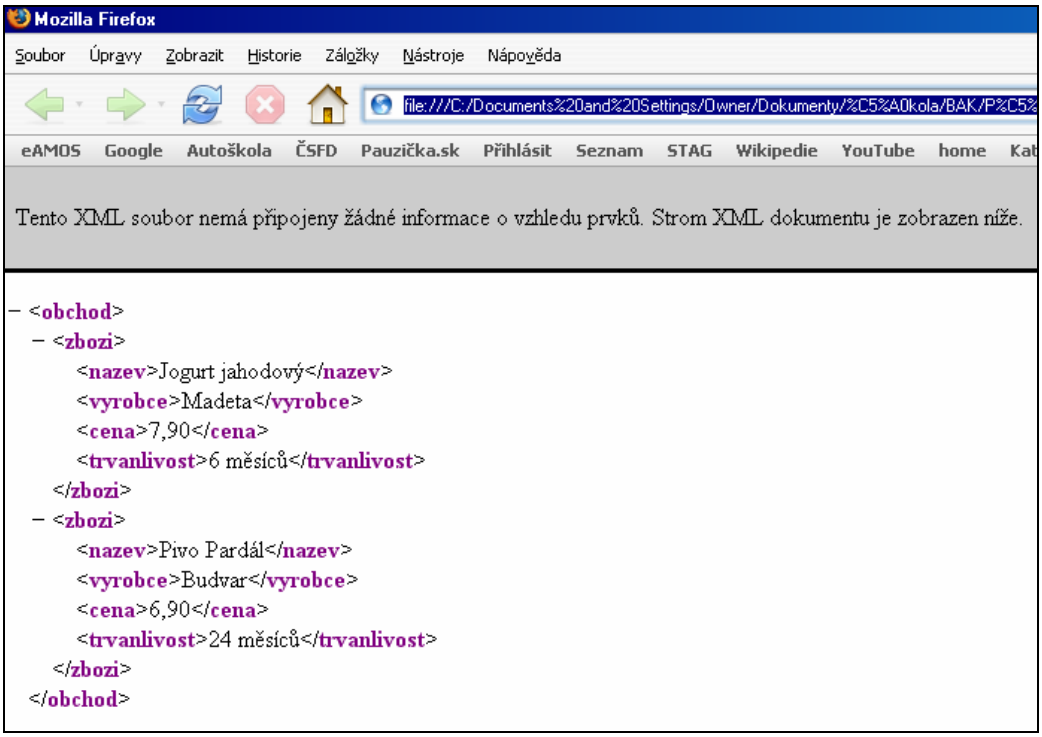
Použitím vhodného souboru XSL pak můžeme data uložená ve formátu XML transportovat do HTML souboru, kde budou uložena například ve formě tabulky.

Název	**Výrobce**	**Cena**	**Trvanlivost**
Jogurt jahodový	Madeta	7,90	6 měsíců
Pivo Pardál	Budvar	6,90	24 měsíců

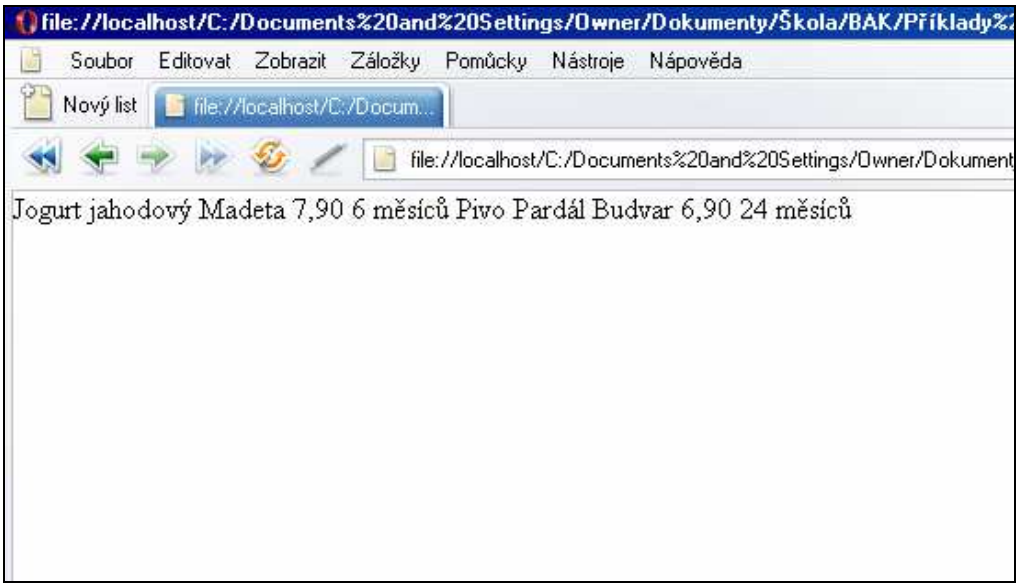
A jak je to se zobrazením XML souboru prohlížeči? Každý ze tří testovaných prohlížečů zobrazil XML, bez připojených informací o vzhledu, jinak.



Internet Explorer 7 zobrazil celou strukturu XML i s názvy elementů a řádkem o použitém kódování



Firefox 2 zobrazil také strukturu XML i s názvy elementů ovšem bez první řádky.



Opera 9 zobrazila pouze obsah dokumentu bez jakéhokoliv formátování a bez názvů elementů

Jazyk XML je hojně využíván na internetu, i v desktopových aplikacích. Moderní praktické využití jazyka XML v tvorbě webových stránek je na příklad v technologii RSS (*Rich Site Summary*). Takzvané RSS čtečky se používají na upozornění čtenáře o novinkách na webu. Uživatel, který chce znát například nové články na některém portálu si umístí RSS kanál buď do svého prohlížeče, instant messengeru nebo jiného softwaru. Autor webu po přidání nějaké novinky, edituje také RSS soubor. Ten je právě ve formátu XML, jehož strukturu tvoří například název článku, popis a odkaz na něj.

3. XHTML 2.0

Dalším stupněm vývoje značkovacích jazyků založených SGML je jazyk XHTML ve verzi 2.0.

První zmínky o jazyku byly zveřejněny na stránkách W3C (http://www.w3.org/) již v roce 2002 a v době psaní této práce je specifikace v pracovním návrhu (*working draft*).

Tato kapitola se nebude zabývat podrobným popisem specifikace, neboť na toto téma již zpracoval práci Vojtěch Soukup v roce 2004 (http://home.pf.jcu.cz/~pepe/Diplomky/soukup.pdf). Můj přínos k tomuto tématu bude v současném stavu návrhu a především v jeho podpoře v aktuálních verzích prohlížečů.

3.1. S čím novým přichází XHTML 2.0

Základní směr udaný konsorciem W3 je patrný v podstatě již od specifikace XHTML 1.0 Strict. Jazyky XHTML 1.1 a XHTML 2.0 pak prohlubují a rozšiřují určené cíle.

- Méně prezentace, více strukturování.

Veškeré formátování provádět pomocí kaskádových stylů a více strukturovat web správným používáním k tomu určených elementů.

- XML.

Využívat stávajících XML technologii, psát web pro prohlížeče založené na XML.

- Méně skriptování.

Typické příklady použití skriptů zahrnout do značkovacího jazyka.

- Nezávislost výstupních zařízení.

Dosáhnout toho, aby byly stránky stejně zobrazitelné jak na monitoru počítače, PDA, mobilním telefonu či televizní obrazovce.

- Integrace se sémantickým webem.

Sémantický web se snaží nebýt kolekcí dokumentů, ale kolekcí dat. Přichází se zcela novými možnostmi vyhledávání informací a souvisí a spolupracuje s pojmy jako jsou umělá inteligence, popisovací logika, mikroformáty či web 2.0. Sémantický web je taktéž aktivita skupiny W3C (http://www.w3.org/2001/sw/).

3.2. Zpětná kompatibilita.

Konsorcium W3C od počátku udávalo, že XHTML 2.0 nebude zpětně kompatibilní. Ve vývoji značkovacích jazyků je to první krok tímto směrem. Zatím každá verze byla s tou předchozí kompatibilní. Teprve čas ukáže, zda to bylo ze strany W3C správné rozhodnutí.

3.3. Konkrétní změny v jazyce.

Specifikace XHTML 2.0 přináší několik konkrétních změn v zápisu některých prvků webové stránky. Změny jsou spočívají ve vypuštění některých elementů, zavedení některých nových, případně přejmenování stávajících.

Největších změn se dočkaly elementy v textovém a prezentačním modulu.

Zavedení elementu `h` a `section` namísto `h1-h6`.

```
<body>
<h>Nadpis první úrovně</h>
<section>
    <h>Nadpis druhé úrovně</h>
    <section>
      <p>Text</p>
    </section>
</section>
</body>
```

Strukturování textu do nadpisů a sekcí různých úrovní je zdůvodněno nesprávným užíváním starších elementů `h1-h6`.

Element `separator` místo `hr`.

```
<p>První odstavec.</p>
<separator />
<p>Druhy odstavec vizuálně oddělený separatorem.</p>
```

Použití elementu `separator` namísto stávajícího `hr` je na první pohled pouhé přejmenování. W3C však uvádí, že pojmenování oddělovače `hr` (*horizontal rule*) bylo zavádějící, neboť některé texty potřebují být odděleny i vertikálně. Navíc typografickou značkou oddělení mohou být i například tři hvězdičky a element separator nám dovoluje definovat jej kaskádovými styly podle libovůle.

Element `br` nahrazen `l`.

```
<l>Já jsem hrozně rád na světě,</l>
<l>když můžu krmit labutě.</l>
```

Další cestou k lepší strukturalizaci webové stránky je nahrazení odřádkování jedním řádkem.

Rozšířené možnosti obsahu odstavce.

```
<p>Obsahem odstavce je seznam
<ul>
<li>první položka</li>
<li>druhá položka</li>
</ul>
</p>
```

Dřívější verze (X)HTML dovolovali, aby byl obsahem odstavce pouze prostý text. XHTML 2.0 umožňuje do elementu `p` vložit tabulky, seznamy, předformátovaný text, či citace.

Implementace menu elementem `nl`.

```
<nl>
```

```
   <label>Menu</label>
   <li href="#kdo_jsme">Kdo jsme</li>
   <li>
      <nl>
          <label>O nás</label>
          <li href="#nabidka">Nabídka</li>
          <li href="#cenik">Ceník</li>
          <li href="#reference">Reference</li>
      </nl>
   </li>
   <li href="#kontakt">Kontakt</li>
</nl>
```

Možnost napsat navigační menu stránky svislé, vodorovné či rolující přímo v XHTML je dostání závazku W3C, který sliboval časté použití skriptů implementovat přímo do značkovacího jazyka.

Univerzální použití elementu `object`.

```
<object src="album/song1.mp3">
    <em>První píseň z alba.</em>
</object>
<object src="fotky/123.jpg" srctype="image/jpeg">
   <em>Fotografie náměstí</em>
</object>
```

Specifikace ruší element `img` a veškerá grafika je vkládána elementem `object`. Použití `object` zůstává i pří vkládání flashových animací nebo java appletů. Atribut `srctype` již nemá pouze informativní charakter, nýbrž dává prohlížeči přímo informaci, jakým způsobem object zpracovat.

Odkazem může být cokoliv.

```
<abbr href="http://www.w3.org/TR/xhtml2/"
title="eXtensible HyperText Markup Language">XHTML 2.0</abbr>
```

Přidáním atributu `href` můžeme udělat odkaz z jakéhokoliv elementu. Nicméně stávající element `a` zůstává pro obecné použití.

Další části jazyka se staly samostatným značkovacím jazykem. Jsou to XML Events pro obsluhu událostí, XForms pro práci z formuláři a XFrames pro podporu rámců.

3.4. Kostra XHTML 2.0

Základní kostra jazyka je tvořena podle známého klíče. Úvodní řádek tvoří xml deklarace, za ní následuje definice příslušného DTD (*Document Type Definition*). Struktura `head`, `title` a `body` pak zůstává stejná jako u starších specifikací.

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/css"
                 href="kostra.css"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 2.0//EN"
    "http://www.w3.org/MarkUp/DTD/xhtml2.dtd">

<html xmlns="http://www.w3.org/2002/06/xhtml2/" xml:lang="en">
  <head>
    <meta http-equiv="content-type"
content="application/xhtml+xml; charset=utf-8" />
    <title>Kostra XHTML 2.0</title>
  </head>
  <body>
    <p>Odstavec s odkazem na
      <a href="http://www.google.com">Google</a>.
    </p>
    </body>
</html>
```

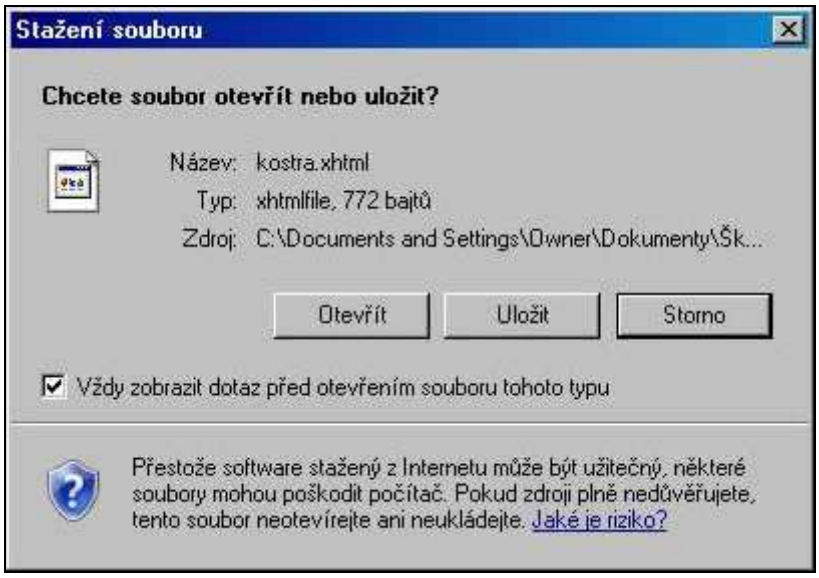
3.5. Podpora prohlížečů.

Kámen úrazu je opět v MIME (*Multipurpose Internet Mail Extension*) typu. Stejně jako tomu bylo u specifikace XHTML 1.1.

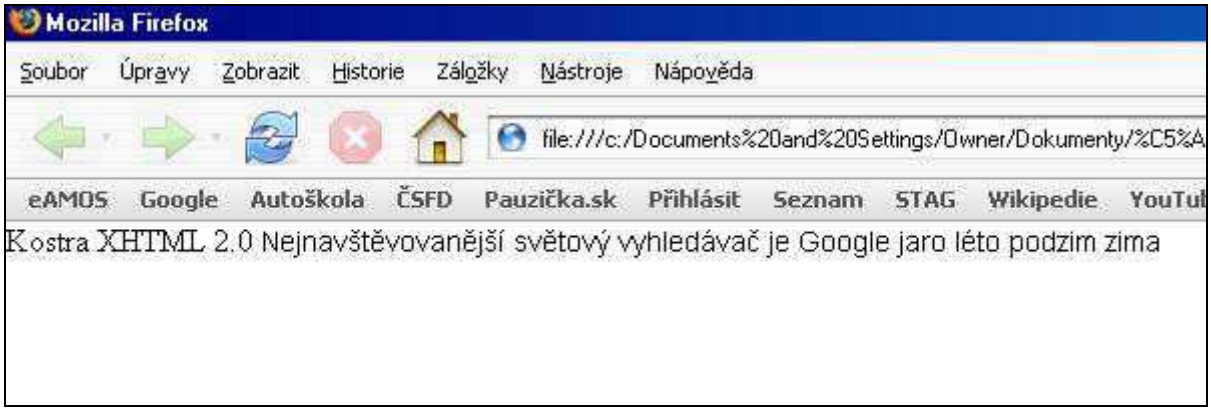
Jak už bylo zmíněno v předchozí kapitole, Internet Explorer není schopen zobrazit stránku ve formátu *.xhtml resp. s MIME typem `application/xhtml+xml`. Avšak ani ostatní dva testované prohlížeče, FireFox 2 a Opera 9, dokument napsaný v XHTML 2.0 nezobrazí.

Testovaný soubor, který je k dispozici v příloze bakalářské práce obsahuje elementy odstavce, odkazu na vyhledávač Google, seznamu čtyř ročních období a obrázku dopravní značky s alternativním textem značka.

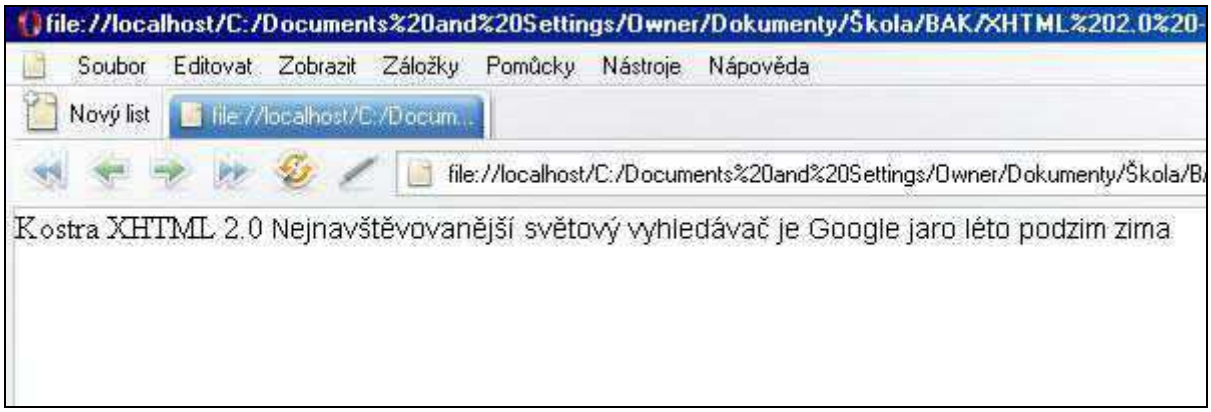
V prohlížeči je zobrazen pouze text mezi jednotlivými elementy. Dokonce neexistuje ani podpora titulku stránky, či alternativního textu obrázku.



Internet Explorer 7 nabídne stažení souboru.



Firefox 2 zobrazí pouze text uvnitř elementů



Opera 9 zobrazí také pouze text.

3.6. Budoucnost

Jak ukázalo testování jazyka XHTML 2.0 podpora v nejpoužívanějších současných webových prohlížečích je nulová. Ačkoliv specifikace přináší mnoho zajímavých novinek zejména ve strukturování dokumentu a příklonu k sémantickému webu, domnívám se, že její budoucnost jejího praktického využití je velmi nejistá.

Může za to několik faktorů. Předně je to zpětná nekompatibilita. Dále pak dlouhá doba vývoje. S tím úzce souvisí i podpora jazyka ze strany výrobců prohlížečů. Tito nebudou do svých produktů implementovat podporu XHTML 2.0, pokud není ve stádiu doporučení. A jako v začarovaném kruhu, internetové vyhledávače nebudou podporovat technologii, kterou nepodporují prohlížeče.

Nedovolí mi to nezmínit se ještě o jednom velice významném výrobci prohlížečů a to sice společnosti Apple se svým produktem Safari. Pan Maciej Stachowiak z Apple oznámil, že se odmítají účastnit pracovní skupiny XHTML 2.0, protože si myslí, že to není vhodná technologie pro web [2].

V současné době se zdá, že jazyk XHTML 2.0 ustupuje do pozadí a nikdy nedosáhne rozšíření a podpory. Budoucnost značkovacích jazyků je přikládána spíše specifikaci HTML 5. Tu vyvíjí skupina WHATWG (*Web Hypertext Application Technology Working Group*). HTML 5 je zpětně kompatibilní. Na rozdíl od XHTML 2.0 má podporu všech hlavních výrobců prohlížečů kromě

Microsoftu, a ačkoliv specifikace ještě není dokončena některé části jazyka již jsou v prohlížečích realizovány.

4. XForms

S příchodem XHTML 2.0 se mění i přístup k webovým formulářům. Bez formulářových prvků se neobejde žádná webová stránka, která chce nabízet něco víc, než pouhé statické prohlížení textu a obrázků. Uživatelská přívětivost s jakou k nám přistupuje množství portálů, vyhledávacích služeb, internetových obchodů a dalších elektronických produktů je přímo závislá na složitosti implementace. S komplexním řešením, jakým XHTML 2.0 jistě chce být, se nemohlo vyhnout revoluci v oblasti webových formulářů. Jak bylo zmíněno v kapitole XHTML 2.0, pro webové formuláře byl vyvinut samostatný značkovací jazyk a to sice Xforms. Ačkoliv tato kapitola pojednává o webových formulářích, jazyk XForms bude v budoucnu možno použít i v desktopových aplikacích, či wapových prezentacích.

Po úvodních informacích o XForms bude následovat popis jednotlivých elementů a možnosti jejich použití. Příkazy budou zobrazeny ve formě zdrojového kódu. U složitějších struktur, bude popsán modelový příklad použití, zdrojový kód, případně obrázek výstupu programu. Všechny zde popisované příklady jsou součástí přílohy bakalářské práce.

4.1. Co to XForms je?

XForms je nový značkovací jazyk. Vychází z (a také plně využívá možností) jazyka XML, proto také ten název. O standardy jazyka se stará konsorcium W3 (http://www.w3.org/MarkUp/Forms/). Jak již bylo zmíněno, XForms vznikl z potřeby specifikace XHTML 2.0 , avšak jeho integrace je možná i do verze XHTML 1.1. Právě v ní budou zobrazeny všechny ukázky a příklady v příloze bakalářské práce.

4.2. Vlastnosti XForms

XForms je, a plně podporuje XML. Odesílá data získaná z formulářů v XML, může načítat výchozí hodnoty z XML souboru, využívá stávajících XML technologií, jako jsou XPath pro adresování a vypočítání hodnoty nebo XML Schema pro definování datových typů. Tento přístup nám dává příslib, že kdo ovládá XML, nebude mít s přechodem na tento nový jazyk problémy.

XForms sází na interaktivitu. Data zadávaná uživatelem jsou ihned ošetřována dle našich požadavků. Je možné uživatele upozornit, nebo mu přímo zakázat pokračování vyplňování formuláře při zadávání nelogických nebo nevhodných dat (např. druhé datum je starší než první). Tímto odpadají okružní cesty k serverům.

XForms nabízejí možnost přepsat stávající formulářové prvky, jak je známe novým způsobem, plus existenci prvků zcela nových. Například řešení se snadným vkládáním datum dialogu nebo hodnoty pomocí pozice posuvníku přímo ve značkovacím jazyku se jeví jako velice zajímavé.

Vývojový tým XForms nám doslova slibuje vytvoření formuláře nákupního koše a průvodce bez nutnosti skriptování [3].

4.3. Jak je to s podporou

V době, kdy píši tuto práci žádná nativní podpora v prohlížečích Internet Explorer 7, FireFox 2 a Opera 9 není. Při testování technologie XForms jsem se uchýlil k řešení, kdy jsem do webového prohlížeče FireFox, nainstaloval plugin pro podporu XForms (https://addons.mozilla.org/cs/firefox/addon/824). Plugin však funguje pouze ve verzi FireFox 2.0.0.13. Při práci mi pomohl XForms validator (http://xformsinstitute.com/validator/), který je prozatím v beta verzi.

4.4. Verze XForms

Do budoucna se počítá s několika verzemi XForms. Verze, které se věnuji v této práci, má označení XForms 1.1 a je ve stádiu CR (C*andidate Rrecommendation*). Kdy by měli základní verze jazyka XForms 1.0, XForms Transitional a XForms 2.0 přejít do konečného stádia doporučení ukazuje následující tabulka.

Specifikace	XForms 1.0	XForms Transitional	XForms 2.0
Recommendation	červen 2008	prosinec 2009	prosinec 2010

4.5. Kostra XHTML dokumentu s použitím XForms

Rozdíly mezi HTML forms a XForms můžeme začít popisovat na jednoduchém příkladě. Na co nejsme u HTML forms zvyklí je, že určité deklarace jsou nutné již v hlavičce XHTML dokumentu. A sice jedná se o element <model>. V něm pak mohou být vloženy další elementy určující způsob odesílání dat a struktura XML.

XForms – head

```
<model>
      <instance xmlns="">
            <jmeno/>
      </instance>
</model>
```

V těle XHTML dokumentu jsou již jen příkazy vkládající formulářové prvky.

```
<input ref="jmeno"><label>Zadej jméno: </label></input>
```

Zcela vypuštěn je párový element `<form></form>`, známý z HTML. Prvek `<label>` je synovský element prvku `<input>` a zobrazí nám panel se slovy Zadej jméno. Prvek `<input>` používá atribut `ref` namísto `name`. I na této jednoduché ukázce je patrné vnořování elementů, jak je známe z XML, nebo XHTML, přičemž vnořování typu ***tučná kurzíva*** je pochopitelně zakázáno.

Ekvivalentní kód v HTML by vypadal takto.

```
<form>
Zadej jméno:<input type="text" name="jmeno">
</form>
```

Oba tyto kódy tedy zobrazí nápis „Zadej jméno“ dále jednořádkové vstupní textové pole. Základní kostra XHTML dokumentu s použitím XForms by pak mohla vypadat následovně.

```
<?xml version="1.0"?>
<h:html xmlns:h="http://www.w3.org/1999/xhtml"
       xmlns="http://www.w3.org/2002/xforms">
<h:head>
    <h:title>Kostra</h:title>
    <model>
        <instance xmlns="">
            <jmeno/>
      </instance>
    </model>
</h:head>
<h:body>
    <h:p>
        <input ref="jmeno"><label>Zadej jméno: </label></input>
    </h:p>

</h:body>
</h:html>
```

Co nás zaujme na tomto úvodním příkladu, je použití prefixu h: v elementech `html`, `head`, `title` a dalších. Právě deklarace jmenných prostorů a využití jednotlivých elementů různými značkovacími jazyky je přístup jazyka XHTML 1.1, jak jsme si jej popsali v jedné z dřívějších kapitol.

Díky prefixům XML procesory poznají, které tagy náleží XHTML a které XForms (viz. úvodní deklarace `xmlns`). V dalších příkladech se setkáme s několika druhy prefixů použitých v jednom souboru. Například tagy patřící XMLSchema pro definování datových typů nebo xml-events pro obsluhu událostí.

Zvláštní postavení má element `<instance xmlns="">`. Ten udává, že následující konstrukce nepatří ani XForms ani HTML. Jedná se o čisté XML. Jako prefix můžeme zvolit jakékoliv písmeno nebo skupinu písmen, přičemž platí, že jednu technologii můžeme nechat bez prefixu.

Dokument s použitím jazyka XForms musí mít příponu *.xhtml, poněvadž je nadstavbou specifikace XHTML 1.1, popř. XHTML 2.0.

4.6. Jak vypadá odeslané XML?

Jak jsme si v úvodu řekli silnou vlastností technologie XForms je odesílání dat ve formě XML. Jako handicap by se zpočátku mohla zdát složitější implementace. Ovšem jak nám konsorcium W3C slibuje na svých stránkách v seznamu často dotazovaných otázek. „Až když začnete vytvářet formuláře, pro které nebylo HTML navrženo, pak se ukáže, že je XForms mnohem jednodušší [4].“

V našem příkladu nás bude zajímat dnešní nálada uživatele. Bude mít na výběr mezi Dobrá, Špatná a Jiná, plus možnost upřesnění nálady v poznámce.

Tento formulář v XHTML tedy by vypadal následovně:

```
<form action='nalada.php' method='post'>
 <label>Dobrá</label>
<input type='radio' name='nalada' value='dobra'
checked='checked' /><br />
 <label>Špatná</label>
<input type='radio' name='nalada' value='spatna' /><br />
 <label>Jiná</label>
<input type='radio' name='nalada' value='jina' /><br />
```

```
 <label>Poznámka:</label>
<input type='text' name='poznamka' /><br />
 <input type='submit' />
</form>
```

Pokud budeme chtít stejný formulář implementovat v XForms, nadeklarujeme si nejprve element `model` v hlavičce XHTML souboru, který nám určí XML strukturu , způsob a cíl odeslání dat.

```
<xf:model>
  <xf:instance>
    <nalada xmlns=''>
      <dnesniNalada/>
      <poznamka/>
    </nalada>
  </xf:instance>
<xf:submission action='nalada.php' method='post' id='submit'/>
</xf:model>
</head>
```

A formulářové prvky napsané v XForms v těle XHTML dokumentu.

```
 <xf:select1 ref='dnesniNalada'>
  <xf:label>Dnešní nálada</xf:label>
  <xf:item>
    <xf:label>Dobrá:</xf:label>
    <xf:value>dobra</xf:value>
  </xf:item>
  <xf:item>
    <xf:label>Špatná:</xf:label>
    <xf:value>spatna</xf:value>
  </xf:item>
  <xf:item>
    <xf:label>Jiná:</xf:label>
    <xf:value>jina</xf:value>
  </xf:item>
 </xf:select1>

 <xf:input ref='poznamka'>
  <xf:label>Poznámka:</xf:label>
 </xf:input>

 <xf:submit submission='submit'>
  <xf:label>Odeslat</xf:label>
 </xf:submit>
```

Odměnou za poněkud náročnější implementaci bude formát dat, který se odešle cílovému skriptu.

```
<nalada>
  <dnesniNalada>dobrá</dnesniNalada>
  <poznamka>Sluníčko svítí, mravenečci se procházejí</poznamka>
</nalada>
```

Tento příklad nám také ukázal jiné použití prefixů a to sice u tagů patřících Xforms. Dále však již budou elementy patřící XForms výhradně bez prefixů a to z důvodu větší přehlednosti.

4.7. Formulářové prvky

Jazyk XForms poskytne všechny ekvivalenty HTML formulářů, plus přichází s některými novými prvky. V předcházejících kapitolách jsme již naznačili, jak se v XForms vytváří některé formulářové kontrolky, nyní se na ně podíváme podrobněji.

Zatímco HTML určuje jak mají kontrolky vypadat, XForms specifikuje, co mají dělat. Jejich vzhled je pak možno nastavit pomocí kaskádových stylů. Navíc na každé výstupní zařízení jinak. Na displej mobilního telefonu jistě můžeme vymyslet nějaké ergonomičtější řešení než na monitor počítače. Nebo naopak v tištěné verzi internetové stránky, bychom měli prvek `<select name="neco" size="1">`, tzv. rozbalovací menu, nahradit vhodnějším.

> *Poznámka: jako první vždy bude uveden stejný formulářový prvek v XHTML, a poté jeho implementace v XForms.*

4.7.1. Jednořádkové vstupní pole

Základní formulářový prvek, sloužící pro vkládání textu. Vedle tlačítek jeden z nejpoužívanějších. Můžeme jej najít v přihlašovacích či registračních dialozích a především při práci s vyhledáváním.

HTML

```
Krátký text: <input type="text" name="kratkyText">
```

XForms

```
<input ref="kratkyText"><label>Krátký text:</label></input>
```

4.7.2. Víceřádkové vstupní pole

Má obdobnou implementaci jako jednořádkové vložení textu. Slouží jako vstup delšího textu na webové stránce.

HTML

```
Dlouhý text: <textarea name="dlouhyText" rows="100"
cols="100"></textarea>
```

XForms

```
<textarea ref="dlouhyText"><label>Dlouhý
text:</label></textarea>
```

Nyní jsme se dostali k tomu, co jsem předesílal v úvodu této kapitoly a to je formátování formulářových prvků pomocí kaskádových stylů. Vlastnosti textové oblasti, jako barva, použité písmo a velikost nastavíme v externím css souboru, a to buď pro všechny tyto prvky stejně, nebo pro každý jinak, přičemž identifikátorem bude právě parametr atributu `ref`.

CSS

```
textarea{ font-family: serif;
      height: 100px;
      width: 100px;
}

textarea[ref=”dlouhyText”]
{ font-family: serif;
      height: 200px;
      width: 200px;
}
```

Připojení kaskádového stylu k souboru v XForms souboru.

```
<?xml version="1.0"?>
<?xml-stylesheet href="03_vlozeniTextu.css" type="text/css"?>
```

4.7.3. Heslo

Vstupní pole pro heslo má v XForms vlastní element `secret.` Při psaní textu se v prohlížeči zobrazují pouze hvězdičky, popřípadě jiné symboly.

HTML

```
Heslo: <input type="password" name="heslo">
```

XForms

```
<secret ref="heslo"><label>Heslo: </label></secret>
```

4.7.4. Přepínač

Umožňují vybrat právě jedno z nabízených možností.

HTML

```
Barva:
Bílá: <input type="radio" name="barva" value="bílá">
Černá: <input type="radio" name="barva" value="černá">
```

XForms

```
<select1 ref="barva" appearance="full">
   <label>Barva:</label>
   <item>
      <label>Bílá:</label><value>bílá</value>
   </item>
   <item>
      <label>Černá</label><value>černá</value>
   </item>
</select1>
```

4.7.5. Zaškrtávací tlačítko

Můžeme zvolit žádné, několik nebo všechny možnosti. Od přepínátek se liší právě elementem `select`. Tím, že se v názvu nevyskytuje číslice jedna, víme, že uživatel bude moci označit více položek.

HTML

```
Objednávám:
<input type="checkbox" name="jidlo" value="snídaně"> snídani
<input type="checkbox" name="jidlo" value="oběd"> oběd
<input type="checkbox" name="jidlo" value="večeře"> večeři
```

XForms

```
<select ref="jidlo" appearance="full">
   <label>Objednávám:</label>
   <item>
      <label>snídani</label><value>snídaně</value>
   </item>
   <item>
      <label>oběd</label><value>oběd</value>
   </item>
   <item>
      <label>večeři</label><value>večeře</value>
   </item>
</select>
```

4.7.6. Rozbalovací menu

Výběr více možností z takovéhoto formulářového prvku se v XHTML řešil přidáním atributu `multiple`. Označení dvou položek v menu se provedlo stisknutím tlačítka myší současně s klávesou Ctrl, případně Shift, pokud jsme chtěli označit i všechny položky mezi dvěma stisknutými. Pokud budeme chtít, aby uživatel mohl zvolit právě jednu položku zapíšeme element `select1`, pokud více položek `select`.

HTML

```
Citroen:
<select name="citroen">
      <option value="AX">AX</option>
      <option value="BX">BX</option>
      <option value="ZX">ZX</option>
```

```
</select>
```

XForms

```
<select1 ref="citroen" appearance="minimal">
<label>Citroen:</label>
<item><label>AX</label><value>AX</value></item>
<item><label>BX</label><value>BX</value></item>
<item><label>ZX</label><value>ZX</value></item>
</select1>
```

4.7.7. Rozbalovací menu podle skupin

Na implementaci prvku zvaném Optgroup byla XML struktura patrná již v HTML formulářových prvcích. To jak jsou položky v menu rozdělovány do skupin řeší XForms podobně. Skupina je vymezena elementem `choices` a titulek skupiny elementem `label`.

V případě, že bychom chtěli, aby byly zobrazeny všechny prvky bez nutnosti rozbalování, přidáme do elementu `select1` nebo `select` atribut `appearance="compact"`.

HTML

```
Auto:
<select name="auto">
   <option selected value="none">None</option>
   <optgroup label="Citroen">
      <option value="AX">AX</option>
      <option value="BX">BX</option>
      <option value="ZX">ZX</option>
   </optgroup>
   <optgroup label="Peugeot">
      <option value="205">205</option>
      <option value="206">206</option>
      <option value="207">207</option>
   </optgroup>
</select>
```

XForms

```
<select1 ref="auto">
```

```
   <label>Auto:</label>
   <item><label>None</label><value>none</value></item>
   <choices>
      <label>Citroen</label>
      <item><label>AX</label><value>AX</value></item>
      <item><label>BX</label><value>BX</value></item>
      <item><label>ZX</label><value>ZX</value></item>
   </choices>
   <choices>
      <label>Peugeot</label>
      <item><label>205</label><value>205</value></item>
      <item><label>206</label><value>206</value></item>
      <item><label>207</label><value>207</value></item>
   </choices>
</select1>
```

4.7.8. Upload

Proměnná, v obou příkladech pojmenovaná soubor, v sobě uchovává cestu k souboru, kterou zvolíme v dialogu pro nahrání souboru.

HTML

```
Vyberte soubor:<input type="file" name="soubor">
```

XForms

```
<upload ref="soubor"><label>Vyberte soubor:</label></upload>
```

4.7.9. Vymazání hodnot

Tlačítko vymaže veškeré hodnoty zadané uživatelem, nebo je inicializuje do výchozích hodnot.

HTML

```
<INPUT type="reset" value="Vymaž hodnoty">
```

XForms

```
<trigger>
   <label>Vymaž hodnoty</label>
   <reset ev:event="DOMActivate"/>
</trigger>
```

Prefix ev: dává internetovému prohlížeči informaci, že následující kód je svázán se jmenným prostorem xml-events, proto je nutná jeho deklarace na začátku XForms dokumentu.

```
<h:html xmlns:h="http://www.w3.org/1999/xhtml"
xmlns="http://www.w3.org/2002/xforms"
xmlns:ev="http://www.w3.org/2001/xml-events">
```

4.7.10. Tlačítka

Stisknutím tlačítka můžeme spustit jakýkoli script Ten deklarujeme v elementu `<trigger>`.

HTML

```
<button onclick="alert('Ahoj světe!')">Script</button>
```

XForms

```
<trigger>
   <label>Script</label>
   <action ev:event="DOMActivate">
      <load resource="javascript:alert('Ahoj světe!')" />
   </action>
</trigger>
```

4.7.11. Tlačítko jako obrázek

Vnořený element `label` může zobrazit nejen text, jak tomu bylo v předchozích ukázkách, ale i příkaz `img` pro vložení obrázku. Uvnitř elementu `<trigger>` kombinujeme XForms a HTML, a proto je nutné rozlišovat jednotlivé příkazy prefixy.

HTML

```
<input type="image" src="img/obr1.gif" ...>
```

XForms

```
<trigger><label><h:img src="img/obr1.gif" /></label></trigger>
```

4.7.12. Rámeček s titulkem

Užitečný příkaz, který formulář opticky sjednotí. Okolo formulářových prvků se zobrazí rámeček s titulkem. Titulek formuláře bude zobrazen vlevo nahoře.

Tento příkaz není testovaným internetovým prohlížečem Firefox s pluginem XForms podporován.

HTML

```
<fieldset>
   <legend>Osobní údaje</legend>
   Křestní jméno: <input name="krestni" type="text">
   Příjmení: <input name="prijmeni" type="text">
</fieldset>
```

XForms

```
<group>
   <label>Osobní údaje</label>
   <input ref="krestni"><label>Křestní jméno:</label></input>
   <input ref="prijmeni"><label>Příjmení:</label></input>
   <input ref="address"><label>Address:</label></input>
</group>
```

4.8. Kontrola výstupu

V následujícím textu se již začnou objevovat prvky webového formuláře, které nemají svůj ekvivalent v HTML Forms. Okamžité zobrazení výsledků formuláře je vhodná vlastnost pro kontrolu dat před samotným odesláním.

Při použití elementu `<output ...>` bude hodnota zobrazena jako text.

```
Hodnota je: <output ref="hod" />
```

Tento prvek nám dokonce umožňuje používat některé matematické operace. V případě, že jinými formulářovými prvky získáme hodnoty pro výšku, šířku a délku tělesa, je možné spočítat jeho objem následujícím zápisem.

```
Objem tělesa je: <output value="vyska * sirka * delka" />
```

4.9. Kontrola rozsahu

V jazyce XForms je možné jednoduchým způsobem určit obor hodnot jednotlivých proměnných, a to použitím elementu `range`. V něm si nastavíme počáteční a koncovou hodnotu a délku kroku. Význam atributu `incremental="true"`, je v tom, že při jeho použití je výstup ihned aktualizován. Pakliže bychom tento atribut vynechali, výstup se aktualizuje až při zmáčknutí tabulátoru, tlačítka myši či při jiné události.

Význam atributu `bind` bude objasněn v kapitole Kontrola nad formulářem. Prohlížeč zobrazuje element `range` jako posuvník.

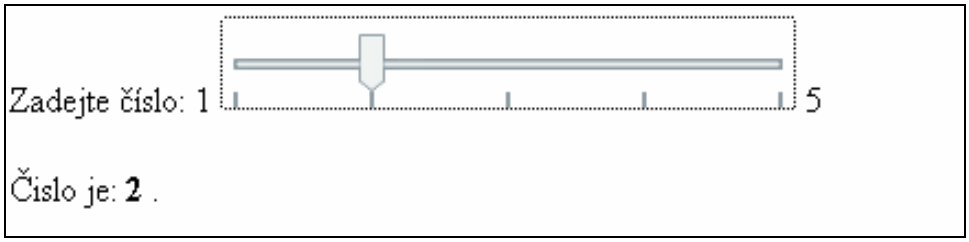
XForms head

```
<model>
     <instance>
        <data xmlns="">
             <cislo></cislo>
        </data>
     </instance>
     <bind id="int" nodeset="/data/cislo" type="xs:integer"/>
</model>
```

XForms body

```
<range bind="int" start="1" end="5" step="1"
incremental="true">
      <label>Zadejte číslo: </label>
</range>
```

```
Číslo je: <h:b><output bind="int"/></h:b>
```



Ukázka elementu `range`.

4.10. Inicializace hodnot

XForms nám dovolují počáteční nastavení hodnot u formulářových prvků, které jsme se si ukázali. Veškerá nastavení se provádějí v hlavičce v XML struktuře a jsou velice jednoduchá. Ukážeme si jak bude vypadat inicializace hodnot na jednom z příkladů, které jsme uvedli dříve.

XForms kontrolka v body zůstává stejná.

```
<select ref="jidlo" appearance="full">
   <label>Objednávám:</label>
   <item>
      <label>snídani</label><value>snidane</value>
   </item>
   <item>
      <label>oběd</label><value>obed</value>
   </item>
   <item>
      <label>večeři</label><value>vecere</value>
   </item>
</select>
```

XML šablona v head

```
<model>
  <instance>
    <data xmlns="">
      <jidlo>snidane vecere
      </jidlo>
    </data>
  </instance>
</model>
```

V XML šabloně jsme pouze přidali hodnotu. V případě, že chceme mít inicializovaných několik hodnot, oddělíme je mezerou.

4.11. Skryté hodnoty

Když bylo potřeba v HTML užívat nějaké neměnné hodnoty, u prvku `input` se nastavilo `type="hidden"` a uživatel nebyl schopen údaj změnit. V XForms nejsou žádné skryté prvky nutné. Pakliže chceme pracovat s nějakou konstantní hodnotou, v XML šabloně si ji deklarujeme, inicializujeme, ale již ji nepropojíme s formulářovými prvky. Tím pádem uživatel nemá možnost tento údaj změnit.

XForms

```
<instance>
      <data xmlns="">
            <predmet>Dotaz z meho webu
            </predmet>
            <telo>
            </telo>
      </data>
</instance>
```

4.12. Získání počátečních hodnot z externího souboru

Počáteční hodnoty nemusíme zadávat v hlavičce XHTML dokumentu, ale je také možné získat je z externího XML dokumentu. V takovém případě element model nebude obsahovat XML šablonu, protože tu zjistí z XML dokumentu. Jediné co zahrnovat je cesta k externímu souboru.

XForms

```
<model>
      <instance src="14_data.xml"/>
</model>
```

Budou-li formulářové prvky strukturované jako dva jednořádkové a jeden několika řádkový textový vstup, resp. tři vstupy, externí XML soubor bude také obsahovat tři vložené elementy s vyplněnými hodnotami.

XForms

```
<input ref='jmeno'>
  <label>Jméno:</label>
 </input>
<h:br />
 <input ref='email'>
  <label>E-mail:</label>
 </input>
<h:br />
<textarea ref="stiznost">
      <label>Stiznost:</label>
</textarea>
```

XML

```
<?xml version="1.0" standalone="yes"?>
<data>
<jmeno>
      Zde zadejte Vaše jméno.
</jmeno>
<email>
      Zde zadejte Váš e-mail.
</email>
<stiznost>
      Zde nepište nic.
</stiznost>
</data>
```

Pro uvedený příklad se XML soubor jmenuje 14_data.xml a je ve stejném adresáři jako XHTML soubor.

4.13. Editace XML

Při editaci XHTML dokumentu můžeme použít také technologii XPath. Ta je mimo jiné adresuje cestu k jakémukoliv elementu nebo atributu XML dokumentu. Tím pádem jsme schopni změnit např. titulek webové stránky. Cesta k titulku ve struktuře XHTML dokumentu by vypadala následujícím způsobem.

XForms

```
<input ref="/h:html/h:head/h:title">...
```

Jako modelový příklad si můžeme představit jednoduché stránky majitele rekreační chaty, který ji na léto pronajímá. Protože rekreanti se střídají často, zveřejňuje na webu informaci o tom zda má volno, či obsazeno. Příklad se skládá se dvou souborů. Soubor s výstupem, který je vystaven návštěvníkům webu se v našem případě jmenuje 15_editaceOutput.xhtml.

XHTML výstup

```
<?xml version="1.0" encoding="UTF-8"?>
<h:html xmlns:h="http://www.w3.org/1999/xhtml"
xmlns="http://www.w3.org/2002/xforms">
<h:head>
      <h:title>Rekreace</h:title>
</h:head>
<h:body>
      <h:p>V chate je momentalne <h:b>obsazeno</h:b>.</h:p>
</h:body>
</h:html>
```

Soubor, ve kterém bude majitel chatky měnit aktuální informace o naplněnosti svého objektu bude vypadat následovně.

XHTML vstup

```
<h:html xmlns:h="http://www.w3.org/1999/xhtml"
xmlns="http://www.w3.org/2002/xforms">
<h:head>
<h:title>Editace</h:title>
<model>
   <instance      src="15_editaceOutput.xhtml"/>
```

```
   <submission action="15_editaceOutput.xhtml"
               method="put" id="change"/>
</model>
</h:head>
<h:body>
<select1 ref="/h:html/h:body/h:p/h:b">
<label>Chata je momentalně: </label>
      <item><label>volná</label><value>volno</value></item>
      <item><label>obsazená</label><value>obsazeno</value></ite
m>
</select1>
<submit submission="change"><label>OK</label></submit>
<h:br />
</h:body>
</h:html>
```

Pakliže chceme tímto způsobem měnit webové stránky musíme je mít v čistém XHTML, protože to dodržuje strukturu XML dokumentu (na rozdíl od HTML). Další podmínkou je, že server musí podporovat metodu put. Offline, tj. na disku tento příklad funguje.

4.14. Metody odesílání

V HTML jsme se mohli setkat s několika způsoby odesílání dat. XForms nechává tři základní způsoby, mění pouze způsob jejich deklarace. Nemusí se psát ve dvou atributech (`method`, `enctype`) ale pouze v jednom (`method`). Následující tabulka uvádí ekvivalenci způsobu odesílání dat.

HTML	XForms
method="get"	method="get"
method="post" enctype="application/x-www-form-urlencoded"	method="urlencoded-post"
method="post" enctype="multipart/form-data"	method="form-data-post"

XForms nabízí ještě několik dalších způsobů odesílání dat. Způsob `method=“post“` posílá výsledky jako XML a `method=“put“` ukládá

výsledky jako XML. Použití metody put jsme si již ukázali v předchozím příkladu.

Jako další použití metody put nám tvůrci XForms nabízejí posílání formuláře a zároveň uložení těchto dat na disk použitím výrazu file:.

XForms – head

```
<submission action="file:zaloha.xml" method="put"/>
```

HTML přístup k odesílání dat znamenal, že data byla odeslána serveru a ten poté odeslal zpět příslušnou webovou stránku. V elementu `submission` však můžeme použít atribut `replace`. Hodnota `replace="instance"` nahradí instanci známým zůsobem. Parametr `replace="none"` nechá stávající dokument bez nahrazení.

4.15. Vícenásobné odesílání

HTML nám dovoloval odesílat data právě na jeden server. XForms nechává možnost odesílat data z formulářů různým serverům nebo různými způsoby. Pokud bychom například chtěli vyhledávaný řetězec předložit různým vyhledávacím enginům.

XForms – head

```
<model>
   <instance><data xmlns=""><q/></data></instance>
   <submission action="http://nejakastranka.com/search"
               method="get" id="com"/>
   <submission action="http://nejakastranka.cz/search"
               method="get" id="cz"/>
</model>
```

Uživatel si pak sám může zvolit způsob vyhledávání.
XForms – body

```
<submit submission="org">
<label>Vyhledat na nejakastranka.org</label></submit>
<submit submission="cz">
<label>Vyhledat na nejakastranka.cz</label></submit>
```

4.16. Více formulářů v jednom dokumentu

Potřebujeme-li v jednom XHTML dokumentu použít více formulářů, v hlavičce dokumentu deklarujeme dva, nebo více elementů model, přičemž každý označíme atributem id=“neco“.

```
<model id="prvni">
  <instance>
    <data xmlns="">
      <a/>
    </data>
  </instance>
</model>
<model id="druhy">
  <instance>
    <data xmlns="">
      <b/>
    </data>
  </instance>
</model>
```

V těle dokumentu pak příslušné formulářové prvky doplníme o atribut model=“něco“.

```
<input ref='a' model="prvni">
  <label>A: </label>
 </input>
<input ref='b' model="druhy">
  <label>B: </label>
 </input>
```

4.17. Kontrola nad formulářem

Pokud jsme v HTML chtěli zakázat činnost některému formulářovému prvku, museli jsme to nastavit napevno ve zdrojovém kódu. Použitím atributů `disabled` nebo `readonly` se kontrolka „vyřadila z provozu“. Pokud jsme však potřebovali změnit její stav, museli jsme se uchýlit ke scriptování.

XForms nám nabízí daleko větší kontrolu na formulářem. V následujících kapitolách si ukážeme vlastnosti XForms, které byly zmíněny v úvodu. Vyžadované položky, omezení datovým typem, možnost vypočítání další hodnoty ze zadaných.

4.17.1. Vyřazení prvků

V HTML se realizovalo použitím atributu `disabled`. Formulářový prvek poté zešedivěl a byl „nezmáčknutelný“. V XForms se stejná funkcionalita řeší zcela jinak, ale výsledek je mnohem elegantnější. Pakliže chceme, uživateli zakázat ovládání některých prvků na základě jeho předchozí volby, prvky se mu nezobrazí. Ačkoliv autoři XForms upozorňují, že prohlížeče budou moci tuto funkcionalitu zobrazovat podle svého.

Následující příklad je formulář mobilního operátora pro výzkum trhu. V základní otázce se zeptá, zda máte mobilní telefon. Odpovědi na výběr jsou Ano, na paušál, Ano na kartu, Nemám. Na základě odpovědi se zobrazí různý formulář. Pokud bude odpověď Ano na paušál, zobrazí se otázka, Kolik měsíčně platíte? Pokud bude odpověď Ano na kartu, bude nás zajímat jak často a v jaké výši si uživatel kupuje kredit. Pokud bude odpověď na základní otázku Nemám, bude uživatel odkázán na webovou stránku.

XForms – head

```
<model>
   <instance><pruzkum xmlns="">
```

```
              <zpusob/>
            <karta/>
            <pausal/>
            <nemam/>
            <jakyKredit/>
            <jakCasto/>
            <kolikPlatite/>
            <navstivte/>
            </pruzkum>
</instance>
   <bind nodeset="jakyKredit" relevant="../zpusob='karta'"/>
   <bind nodeset="jakCasto" relevant="../zpusob='karta'"/>
   <bind nodeset="kolikPlatite" relevant="../zpusob='paušál'"/>
   <bind nodeset="navstivte" relevant="../zpusob='nemám'"/>
</model>
```

Element `pruzkum` nám určuje XML strukturu. Elementem `bind` zajistíme, že uzel s výběrem, jaký kredit a jak často bude mít smysl, pouze pokud zvolíme způsobem placení kartu. Stejně tak informace kolik měsíčně platíme nás bude zajímat pouze pokud odpovíme, že platíme paušálně.

XForms – body

```
<select1 ref="zpusob"><label>Máte mobilní telefon? </label>
   <item><label>Ano, na
kartu</label><value>karta</value></item>
   <item><label>Ano, na
paušál</label><value>paušál</value></item>
   <item><label>Nemám
telefon</label><value>nemám</value></item>
</select1>
<select1 ref="jakyKredit" appearance="full">
   <label>Jak vysoký si kupujete kredit:</label>
   <item>
      <label>250,-</label><value>250</value>
   </item>
   <item>
      <label>500,-</label><value>500</value>
</item>
<item>
      <label>1000,-</label><value>1000</value>
</item>
</select1>
<select1 ref="jakCasto" appearance="full">
   <label>A jak často:</label>
   <item>
      <label>1 týdně</label><value>1tydne</value>
   </item>
```

```
   <item>
      <label>1 za měsíc</label><value>1mesicne</value>
</item>
<item>
      <label>1 za půl roku</label><value>1zaPulRoku</value>
</item>
</select1>
<select1 ref="kolikPlatite" appearance="full">
   <label>Kolik platíte měsíčně: </label>
   <item>
      <label>250,-</label><value>250</value>
   </item>
   <item>
      <label>500,-</label><value>500</value>
</item>
<item>
      <label>1000,-</label><value>1000</value>
</item>
</select1>
<output ref="navstivte"><h:b>Navštivte stránky našeho mobilního
operátora</h:b></output>
```

4.17.2. Nepřepisovatelné kontrolky

Jedná se o podobnou situaci jako v předcházející kapitole. V HTML se přidal atribut `readonly` a pro změnu jeho stavu jsme se museli uchýlit k JavaScriptu. Nastavení prvku jako nepřepisovatelného s použitím XForms se bude opět řešit pomocí elementu `bind`. V atributu `nodeset` si určíme uzel XML struktury, který chceme učinit nepřepisovatelný a v atributu `readonly` nastavíme podmínku. Ta samozřejmě není nutná, respektive nechá se nastavit tak, aby byla vždy splněna.

XForms – head

```
<model>
   <instance><data xmlns="">
      <prepisovani>ne</prepisovani>
      <text>Na okoř je cesta.</text>
   </data></instance>
   <bind nodeset="text" readonly="../prepisovani='ne'"/>
</model>
```

XForms – body

```
<select1 ref="prepisovani" appearance="full"><label>Chcete
umožnit přepisování textu? </label>
```

```
   <item><label>Ano</label><value>ano</value></item>
   <item><label>Ne</label><value>ne</value></item>
</select1>
<h:br />
<input ref="text">
      <label>Text: </label>
</input>
```

Proměnná text, která je předem inicializována, se nechá změnit pouze pokud na zadanou otázku odpovíme Ano.

4.17.3. Povinné vyplnění položky

Další z vlastností, které určuje nastavení elementu `bind` je povinné vyplnění položky, reprezentované atributem `required`. Ten může mít jako parametr hodnotu `true`, nebo jakýkoliv XPath výraz. Záleží na prohlížeči, jak bude zobrazovat povinné položky, ovlivnit to však můžeme použitím kaskádových stylů.

```
<model>
   <instance>
      <data xmlns="">
            <jmeno/>
            <mail/>
            <web/>
      </data>
</instance>
<bind nodeset="jmeno" required="true()"/>
<bind nodeset="mail" required="true()"/>

<submission action="formular.php" method="post" id="submit"/>
</model>
```

4.17.4. Další možná omezení

Přidáním atributu `constraint` můžeme uzlu nastavit další omezení. Uvažujme příklad knihy jízd. Počet najetých kilometrů vozidla v cíli musí být vyšší než při startu jízdy. Abychom zamezili nesmyslnému zadávání dat,

přidáme do uzlu cil, podmínku `constraint`. Symbol `"."` znamená tato hodnota a výraz `>` je zápis symbolu >.

```
<bind nodeset="cil" constraint=". &gt; start"/>
```

Použitím atributu `constraint` můžeme také simulovat atribut `required`. Pokud do uzlu přidáme podmínku, že délka vstupu musí být větší než nula.

```
<bind nodeset="jmeno" constraint="string-length(.) &gt; 0"/>
```

4.17.5. Počítání

Dalším atributem, který můžeme využít v elementu `bind` je `calculate`. Ten nám umožňuje provádět množství aritmetických operací, operací s řetězci, datumy nebo podmíněným větvením.

```
<bind ref="objem" calculate="a * b * c"/>
```

4.17.6. Datové typy

Přidáním definice datového typu do uzlu, můžeme opět zefektivnit práci s formulářem, popřípadě omezit nadbytečné cesty nepoužitelných dat k serveru. Na začátku XHTML dokumentu definujeme prefix a jmenný prostor XML-Schema.

```
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
```

Pakliže chceme, aby byl uzel nesl pouze proměnné typu datum, definujeme datový typ takto.

```
<bind nodeset="datum" type="xsd:date"/>
```

Datových typů je celá řada [5].

Typ xsd:	Vysvětlení
string	řetězec
normalizedString	normalizovaný řetězec
integer	celé číslo
nonPositiveInteger	nekladné celé číslo
negativeInteger	záporné celé číslo
nonNegativeInteger	nezáporné celé číslo
positiveInteger	kladné celé číslo
boolean	logická hodnota
decimal, double	desetinné číslo
date,time, dateTime	datum, čas, datum a čas
anyURI	adresa URI
listItems	seznam řetězců oddělených mezerami
listItem	řetězec bez mezer

Poznámka: Výše uvedené atributy elementu bind, je možno kombinovat. Tímto můžeme nastavit několik podmínek jednomu uzlu.

4.17.7. Switch

Switch, česky přepínač, je rodičovský element struktury, která má různé možnosti využití. Struktura se skládá z bloků `case`, které jsou identifikovány parametry `id` a z tlačítek `trigger`, které slouží k přepínání jednotlivých bloků. Vnořené elementy elementu `trigger` jsou `label,` uchovávající titulek tlačítka, `a toggle.` Příkaz

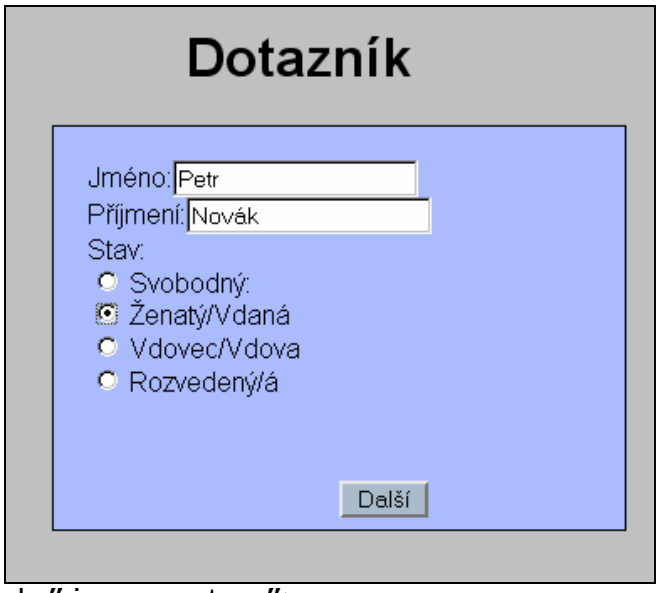
První ukázkový příklad přepínačů bude představovat dotazník. Ten má tři části. V první se zadává jméno, příjmení a rodinný stav, v druhé dosažené vzdělání a řidičský průkaz a v poslední se zobrazí zadané informace. K přepínání mezi okny budou sloužit tlačítka Další a Předchozí, přičemž v posledním okně namísto tlačítka Další bude Odeslat.

XForms

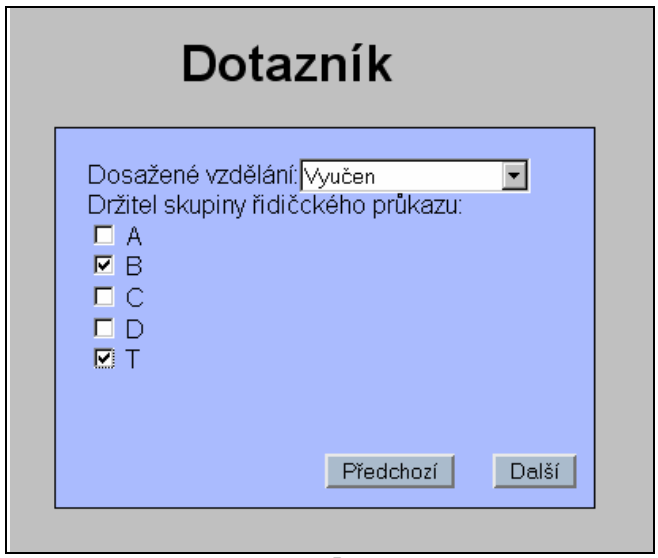
```
<switch>
<case id="jmeno_stav">
 <input ref='jmeno'>
  <label>Jméno:</label>
 </input>
<input ref='prijmeni'>
  <label>Příjmení:</label>
 </input>
<select1 ref="stav" appearance="full">
   <label>Stav:</label>
   <item>
      <label>Svobodný:</label><value>svobodny</value>
   </item>
   <item>
      <label>Ženatý/Vdaná</label><value>zenaty</value>
</item>
<item>
      <label>Vdovec/Vdova</label><value>Vdovec</value>
</item>
<item>
      <label>Rozvedený/á</label><value>rozvedeny</value>
</item>
</select1>

<trigger>
         <label>Další</label>
         <toggle case="vzdelani_ridicak"
ev:event="DOMActivate"/>
</trigger>
</case>
...
<case id="vzdelani_ridicak">
...
</case>
</switch>
```

Deklarace tlačítek v case id=“vzdelani_ridicak“

```
<trigger>
         <label>Předchozí</label>
         <toggle case="jmeno_stav" ev:event="DOMActivate"/>
</trigger>

<trigger>
         <label>Další</label>
         <toggle case="vysledky" ev:event="DOMActivate"/>
</trigger>
```




```
<case id="jmeno_stav">
```




```
<case id="vzdelani_ridicak">
```

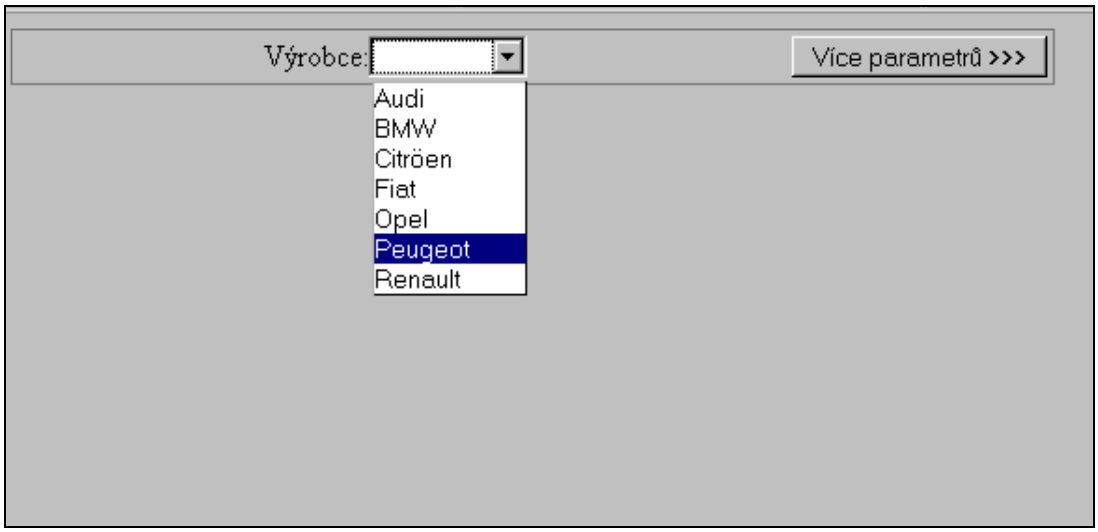
<case id="vysledky">

Jiné možné využití elementu switch je ve výběru množství parametrů. V této ukázce hledáme v databázi inzerátů automobilů. V první řadě nás zajímá výrobce automobilu, někteří uživatelé si ale budou chtít filtrovat výsledky podle stáří automobilu, paliva, stáří inzerátu. Necháme jim proto možnost zobrazit nebo skrýt více parametrů vyhledávání tlačítkem Více, resp. Méně parametrů.

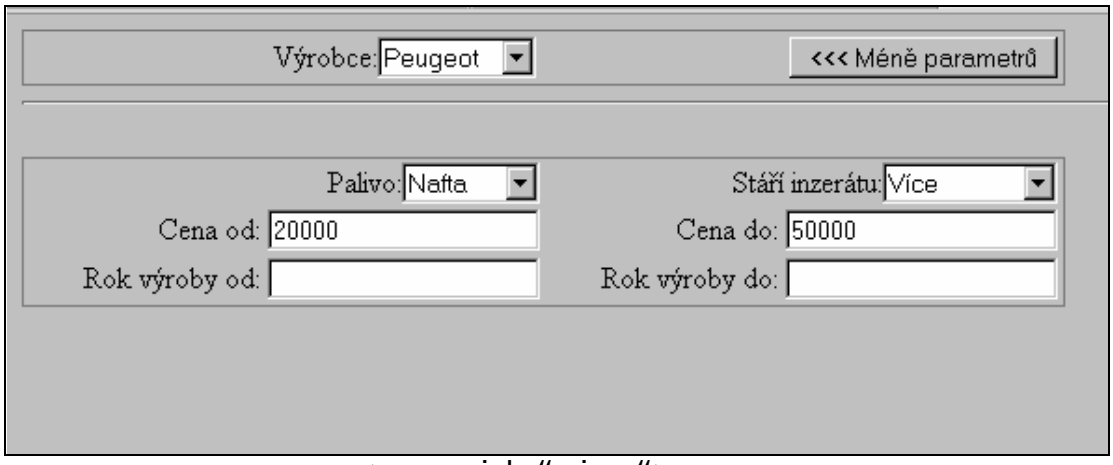
XForms

```
<switch>
   <case id="mene">
      <select1 ref="vyrobce" appearance="minimal">
      <label>Výrobce:</label>
      <item><label>Audi</label><value>Audi</value></item>
      ...
      </select1>
      <trigger>
         <label>Více parametrů </label>
         <toggle case="vice" ev:event="DOMActivate"/>
     </trigger>
   </case>

   <case id="vice">
      <select1 ref="vyrobce" appearance="minimal">
      <label>Výrobce:</label>
      <item><label>Audi</label><value>Audi</value></item>
      ...
      </select1>
```

```
<trigger>
         <label>&lt;&lt;&lt; Méně parametrů</label>
         <toggle case="mene" ev:event="DOMActivate"/>
</trigger>
      <select1 ref="palivo" appearance="minimal">
            <label>Palivo:</label>

      <item><label>Benzín</label><value>Benzín</value></item>
            ...
      </select1>
      <select1 ref="stari" appearance="minimal">
            <label>Stáří inzerátu:</label>
            <item><label>Den</label><value>den</value></item>
            ...
      </select1>

      <input ref="cenaOd"><label>Cena od: </label></input
      <input ref="cenaDo"><label>Cena do: </label></input>
      <input ref="rvOd"><label>Rok výroby od: </label></input>
      <input ref="rvDo"><label>Rok výroby do: </label></input>
   </case>

</switch>
```



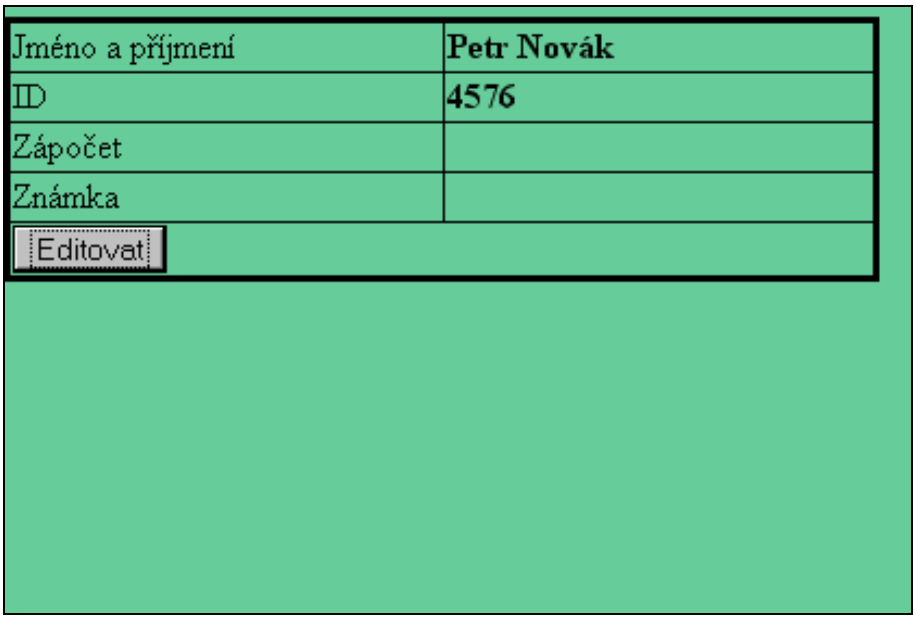
<case id="mene">



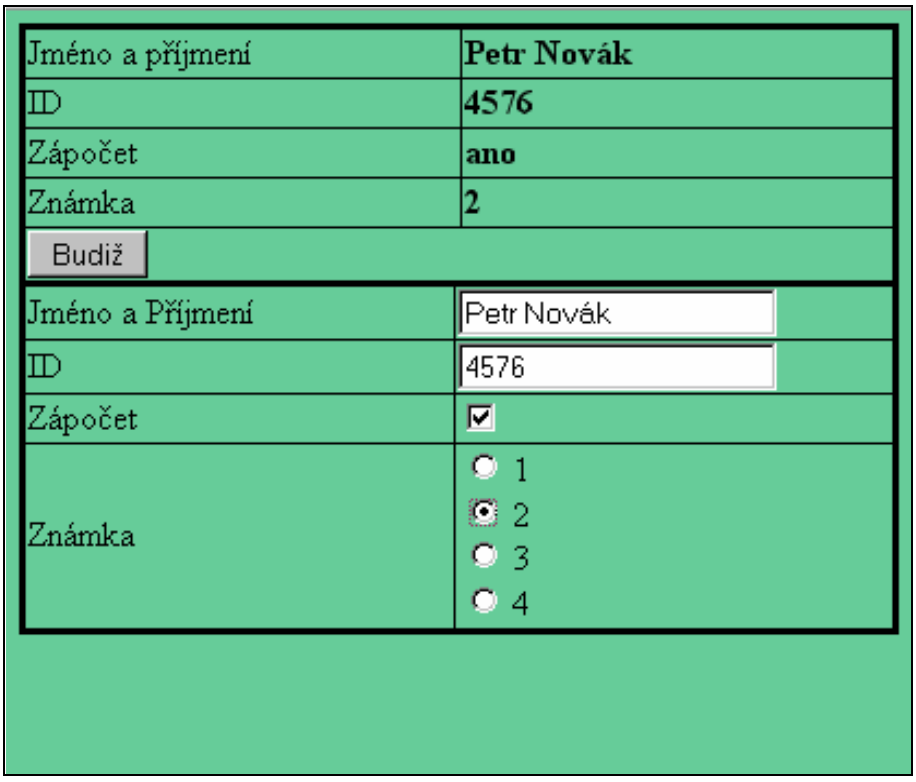
`<case id="vice">`

Poslední ukázkou využití elementu switch bude editace zadaných údajů. Na webové stránce máme tabulku s informacemi o studentovi. Dále je k dispozici tlačítko Editovat. Po jeho stisku se pod tabulkou s údaji zobrazí tabulka editační. V ní máme možnost veškeré údaje změnit. Stisknutím tlačítka Budiž se zobrazí původní tabulka ovšem s editovanými záznamy. Záznamy, které jsou reprezentovány příkazem `<output ref="…">` se dynamicky mění již při zadávání a není potřeba stránku aktualizovat.

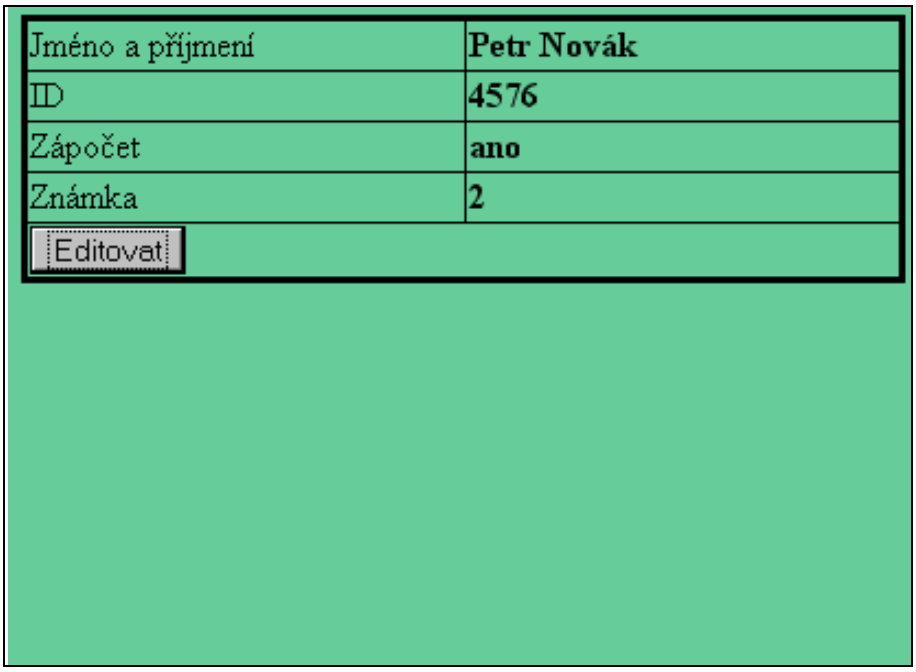
Protože implementace je analogická předcházejícím dvěma ukázkám uvedu pouze ukázku dokumentu bez zdrojového kódu.



Tabulka se zadanými informacemi



Zobrazení editační tabulky po stisknutí tlačítka editovat



Původní tabulka s zeditovanými údaji

4.17.8. Repeat

Element repeat představuje důležitou vlastnost. Je schopen zobrazit XML strukturu podle zadaných kritérií, tím způsobem že prochází jednotlivé uzly XML a vypisuje je. Repeat je párový tag, který má dva atributy. Parametr nodeset určuje uzel XML dokumentu, který má být zobrazen a parametr id uchovává index uzlu. Díky jedinečným indexům jsme schopni pracovat s jednotlivými záznamy.

XForms

```
<repeat nodeset="/zapasy/zapas" id="indexZapasu">
…
</repeat>
```

V ukázce použití elementu repeat budeme spravovat databázi fotbalových zápasů. Každý záznam se skládá ze jména hostů, jména domácích a výsledku zápasu. Dále budeme moci vytvořit nový záznam, editovat jej a smazat.

Všechna data budou uložena v externím XML dokumentu. XML tedy bude mít následující uspořádání.

XML

```
<?xml version="1.0" encoding="UTF-8"?>
<zapasy>
      <zapas>
            <domaci>Žirovnice</domaci>
            <hoste>Počátky</hoste>  ¨
            <vysledek>0:1</vysledek>
      </zapas>
      <zapas>
            <domaci>Kamenice n. Lipou</domaci>
            <hoste>Nová Včelnice</hoste>  ¨
            <vysledek>2:2</vysledek>
</zapas>
</zapasy>
```

Deklarace elementu repeat v těle Xforms dokumentu

```
<repeat nodeset="/zapasy/zapas" id="indexZapasu">
      <input ref="domaci">
            <label>Domácí: </label>
      </input>
      <input ref="hoste">
            <label>Hosté: </label>
      </input>
      <input ref="vysledek">
            <label>Výsledek: </label>
      </input>
</repeat>
```

V hlavičce XForms budeme načítat databázi z XML souboru a ukládat ji tamtéž

```
<model>
      <instance src="21_repeat.xml" />
      <submission id="uloz" method="put" action="21_repeat.xml"
replace="none"/>
</model>
```

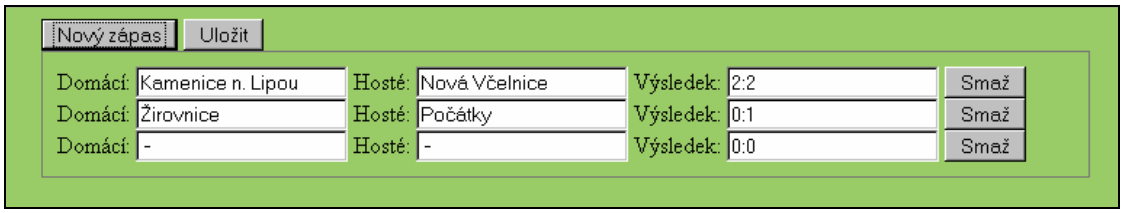
Deklarace tlačítka pro vložení nového záznamu. Element action je svázán se jmenným prostorem xml-events. Vložený element insert v parametru nodeset určuje, který uzel XML je vkládán, a v parametru position a at, kam bude záznam vložen. V našem případě na začátek. Při vkládání na konec

seznamu bychom zadali parametry `position="after"` a `at="count(zapas)"`. Zároveň příkazem setvalue inicializujeme příslušný uzel. Kdybychom tak neučinili jako hodnoty nového záznamu by se nastavili hodnoty posledního výchozího uzlu.

```
<trigger>
   <label>Nový zápas</label>
   <action ev:event="DOMActivate">
      <insert nodeset="zapas" position="before" at="1" />
      <setvalue ref="zapas[last()]/domaci"> - </setvalue>
      <setvalue ref="zapas[last()]/hoste"> - </setvalue>
      <setvalue ref="zapas[last()]/vysledek">0:0</setvalue>
   </action>
</trigger>
```

Smazání záznamu realizujeme přidáním tlačítka Smaž na každý řádek tabulky, tj. do každého cyklu repeat. Protože si každý záznam uchovává jedinečný index, každé jednotlivé tlačítko smaže právě ten řádek tabulky na kterém je zobrazeno.

```
<trigger>
   <label>Smaž</label>
   <delete ev:event="DOMActivate" nodeset="."
at="index('indexZapasu')" />
</trigger>
```



Výstup příkladu na použití elementu repeat

4.18. Nápověda, rada, výstraha

Všechny formulářové prvky, s výjimkou `output` mohou, vedle elementu `label`, obsahovat také jeden z trojice, prvků, která má informativní charakter. Není asi vhodné použít všechny tři najednou pro jeden prvek, ačkoliv zakázané to není.

- Nápověda

Prvek se syntaxí `<help>Text nápovědy</help>`, zobrazí informace, když o ně uživatel požádá. Nápověda se vyvolá stisknutím tlačítka F1, pokud je příslušná kontrolka, ve které se prvek nachází aktivní

- Rada

Krátký text který se zobrazí při najetí myši na příslušný prvek. Obsahuje zpravidla informace nutné k vyplnění prvku. Zapisuje se `<hint>Text rady</hint>`.

- Výstraha

Informace, která se zobrazí při nesprávném vyplnění formulářového prvku.

Všechny tyto prvky budou ukázány na příkladu knihy jízd z kapitoly Další omezení.

```
<input ref="start">
      <label>Počet kilometrů na startu: </label>
      <hint>Zadejte počet kilometrů na tachometru při nástupu
do
      auta</hint>
</input>

<input ref="cil">
      <label>Počet kilometrů v cíli: </label>
      <alert>Pozor, zadáváte méně kilometrů než na
startu</alert>
</input>

<input ref="jmeno"><label>Jméno zaměstance: </label>
      <help>Při vyplňování knihy jízd vyplňte nejprve
      počet kilometrů na startu, poté počet kilometrů v cíli.
      Nakonec zadejte Vaše jméno a uložte formulář.</help>
</input>
```

4.19. Zhodnocení XForms

Po projití celé specifikace značkovacího jazyka XForms si v tomto místě můžeme dovolit jakési zhodnocení. Zásadní nevýhoda je v nulové podpoře

prohlížečů, i když nevýhoda je to relativní a věřím, že v průběhu několika dalších let a verzí prohlížečů se stane tato výtka nepodstatnou.

Konkrétní výhody jazyka byly vyjmenovány výše. Zamyslím se však alespoň nad jednou z nich. Při práci s tímto novým jazykem jsem měl pocit, že vývojový tým skutečně dostál jednomu ze svých slibů a to, že časté skriptovací techniky byly implementovány do značkovacího jazyka. Jsem přesvědčen, že právě tuto vlastnost budoucí autoři webových aplikací ocení.

5. XHTML-Print

XHTML-Print (*eXtensible Hypertext Markup Language for Printing*) je specifikace konsorcia W3C zaměřená na tisk. Její nejnovější verze je v době psaní této práce ve stádiu Proposed Recommendation, což je poslední fáze před konečným doporučením [6].

XHTML-Print je součástí skupiny XHTML jazyků využívajících Modularizace XHTML (http://www.w3.org/TR/2001/REC-xhtml-modularization-20010410/).

Plná specifikace je k dispozici na http://www.w3.org/TR/2006/PR-xhtml-print-20060131/.

5.1.1. Zaměření

Technologie XHTML-Print není zaměřena na přesný tisk, ale na takový, kde je přijatelná určitá proměnlivost ve formátování výstupu, a kde je cílem uchování obsahu. Dále pak na použití v takových situacích, kdy není vhodné nebo možné instalovat ovladače pro tiskárnu.

Cílem XHTML-Print je standardizovat tiskové výstupy z mobilních zařízení a z tiskáren, které nemají celostránkovou vyrovnávací paměť a obecně tisknou odshora dolů a zleva doprava na papír v orientaci na výšku.

5.1.2. Jak XHTML-Print funguje?

V hlavičce html dokumentu určíme, že dokument využívá specifikace XHTML-Print. Tiskárna se pak rozhoduje podle pravidel, která si ukážeme v následujících kapitolách, které elementy jsou pro tisk důležité, které jsou volitelné a které jsou irelevantní.

Je zřejmé, že tiskárna nebude pracovat například s atributem `maxlength`, který určuje maximální možnou délku vstupu u formulářového prvku. Naopak

atribut `rowspan` u elementu `td` musí při tisku brán v potaz, protože určuje kolik následujících buněk tabulky v řádku bude sloučeno.

5.1.3. XHTML-Print dokument

Aby html dokument vyhovoval XHTML-Print specifikaci, musí splňovat některá kritéria.

- kořenový element dokumentu musí být html
- jako výchozí jmenný prostor musí být použito XHTML, tudíž atributem elementu html bude http://www.w3.org/1999/xhtml
- dokument musí obsahovat deklaraci DOCTYPE o použitém DTD (*Document Type Definition*) ve tvaru

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML-Print 1.0//EN"
"http://www.w3.org/MarkUp/DTD/xhtml-print10.dtd">
```

- MIME (*Multipurpose Internet Mail Extension*) musí být nastaveno na "`application/xhtml+xml`"

5.1.4. Podpora

Pro zjištění podpory této specifikace jsem napsal dokument, který obsahuje všechny elementy, které podle pravidel XHTML-Print musí být vytisknuty. Soubor je přílohou bakalářské práce.

Situace s podporou prohlížečů je obdobná jako v předchozích kapitolách. Obdobná v tom smyslu, že dokument má MIME typ `application/xhtml+xml`, a ten jak jsme měli možnost poznat již dříve nepodporuje prohlížeč Microsoft Explorer 7.

5.1.5. Seznam XHTML-Print modulů

Dokument typu XHTML-Print je definován jako sada XHTML modulů. Každý modul obsahuje elementy specifické pro určitou skupinu vlastností, které využíváme při tvorbě webových stránek. Některé moduly se nevyskytují v seznamu základní modularizace, nicméně pro XHTML-Print jsou nutné.

Modul struktury: `body, head, html, title`

Textový modul: `abbr, acronym, address, blockquote, br, cite, code, dfn, div, em, h1, h2, h3, h4, h5, h6, kbd, p, pre, q, samp, span, strong, var`

Hypertextový modul: `a`

Modul seznamů: `dl, dt, dd, ol, ul, li`

Prezentační modul: `b, big, hr, i, small, sub, sup, tt`

Formulářový modul: `form, input, label, select, option, textarea`

Tabulkový modul: `caption, table, td, th, tr`

Obrázkový modul: `img`

Objektový modul: `object, param`

Modul metainformací: `meta`

Modul skriptů: `script, noscript`

Modul style: `style`

Modul link: `link`

Modul base: `base`

5.2. Moduly

V této podkapitole si podrobně rozebereme elementy v jednotlivých modulech a jejich atributy. Každá vlastnost bude obsahovat klíčové slovo, které vyjadřuje vztah ke XHTML-Print.

Klíčové slovo	Popis
musí (must)	podpora tiskárny pro tento atribut je povinná
měl by (should)	podpora tohoto atributu je doporučená, nikoliv vyžadovaná
může (may)	funkcionalita toho atributu je zcela nepovinná
nepoužitelný (N/A)	atribut nebude zohledněn, tiskárna by měla tento atribut ignorovat a nehlásit jej jako chybu
viz. kolekce	tento atribut je součástí kolekcí pro jednoduchou prezentaci a specifikace XHTML-Print jej nijak neupravuje, původní specifikace těchto kolekcí je dostupná na http://www.w3.org/TR/2001/REC-xhtml-modularization-20010410/abstract_modules.html ve článku 5.1 Attribute Collections

Přesný význam a definice klíčových slov je k dispozici na webu Skupiny inženýrských internetových odborníků (http://www.ietf.org/rfc/rfc2119.txt).

5.2.1. Atributy obecně

Obecně atributy mohou spadat do několika kategorií. Pro potřeby XHTML-Print to bude skupina Core, do které patří atributy `class`, `id` a `title`, dále skupina I18N, která specifikuje základní jazyk elementu a nakonec Style pro deklaraci kaskádového stylu.

Bude-li v popisu jednotlivých modulů odkaz na obecné atributy, pak se jejich zpracování řídí následující tabulkou.

Jméno kolekce	Atributy v kolekci	Zpracování
Core	`class`	**musí**

Core	`id`	**musí**
Core	`title`	nepoužitelný
I18N	`xml:lang`	může
Style	`style`	**musí**

5.2.2. Modul struktury

Elementy	Atributy	Zpracování
body	obecně	viz. kolekce
head	I18N	viz. kolekce
head	profile	může
html	I18N	viz. kolekce
html	version	nepoužitelné
html	xmlns	**musí**
title	I18N	viz. kolekce

Pokud se zaměříme na element `title`, vidíme, že je u něj poznámka I18N, viz. kolekce. V tabulce v kapitole 1.2.1 Atributy obecně zjistíme, že I18N, jakožto specifikace základního jazyka elementu může, ale nemusí být tiskárnou zohledněn. Toto však platí pouze pro tisk atributu `xml:lang`.

Samotný element `title`, jak opět vidíme v tabulce kapitoly 1.2.1. je pro tisk nepoužitelný. Je to z toho důvodu, že je určený pro zobrazení textu v hlavní liště okna prohlížeče.

5.2.3. Textový modul

Elementy	Atributy	Zpracování
abbr, acronym, addres	obecně	viz. kolekce
blockquote	obecně	viz. kolekce
blockquote	cite	nepoužitelné

br	Core	viz. kolekce
cite, code, dfn, div, em, h1, h2, h3, h4, h5, h6, kbd, p	obecně	viz. kolekce
pre	obecně	viz. kolekce
pre	xml:space=“preserve“	**musí**
q	obecně	viz. kolekce
q	cite	nepoužitelné
samp, span, strong, var	obecně	viz. kolekce

Z tabulky je patrné, že specifikace XHTML-Print v elementech textového modulu nepřinesla mnoho nového. Většina tagů byla podporována tiskárnou již původní modularizaci.

5.2.4. Hypertextový modul

Elementy	Atributy	Zpracování
a	obecně	viz. kolekce
a	accesskey	nepoužitelné
a	charset	nepoužitelné
a	href	nepoužitelné
a	hreflang	nepoužitelné
a	rel	nepoužitelné
a	rev	nepoužitelné
a	tabindex	nepoužitelné
a	type	nepoužitelné

Hypertextové odkazy, jakožto základní prvky hypertextového dokumentu jsou ze své podstaty nepoužitelné v tištěném výstupu.

5.2.5. Modul seznamů

Elementy	Atributy	Zpracování
dl, dt, dd, ol, ul, li	obecně	viz. kolekce

5.2.6. Prezentační modul

Elementy	Atributy	Zpracování
b, big, hr, i, small, sub, sup, tt	obecně	viz. kolekce

V modulu seznamů a prezentačním modulu, se nacházejí elementy pro formátování textu. Jejich podpora v tiskárnami byla již před XHTML-Print.

5.2.7. Formulářový modul

Elementy	Atributy	Zpracování
form	obecně	viz. kolekce
form	action	nepoužitelné
form	method("get"|"post")	nepoužitelné
form	enctype	nepoužitelné
input	obecně	viz. kolekce
input	accesskey	nepoužitelné
input	checked("checked")	**musí**
input	maxlength	nepoužitelné
input	name	nepoužitelné
input	size	**musí**
input	src	nepoužitelné
input	tabindex	nepoužitelné
input	type("text")	**musí**

input	type("password")	**musí**
input	type("checkbox")	**musí**
input	type("radio")	**musí**
input	type("submit")	**musí**
input	type("reset")	**musí**
input	type("hidden")	**musí**
input	value	**musí**
label	obecně	viz. kolekce
label	acceskey	nepoužitelné
label	for	nepoužitelné
select	obecně	viz. kolekce
select	multiple("multiple")	nepoužitelné
select	name	nepoužitelné
select	size	**musí**
select	tabindex	nepoužitelné
option	obecně	viz. kolekce
option	selected("selected")	**musí**
option	value	nepoužitelné
textarea	obecně	viz. kolekce
textarea	acceskey	nepoužitelné
textarea	cols	**musí**
textarea	name	nepoužitelné
textarea	rows	**musí**
textarea	tabindex	nepoužitelné

Zajímavé se může jevit, že v tabulce formulářového modulu je u atributu `type("hidden")` poznámka **musí**. Samozřejmě, že tento atribut nebude vytištěn, ale bude zpracován v tom smyslu, že bude rozpoznán a ignorován.

Atribut `src` u `input` elementu umožňuje zobrazit tlačítko jako obrázek. V XHTML-Print není podporován, protože image není součástí základního modulu formulářů.

5.2.8. Tabulkový modul

Elementy	Atributy	Zpracování
caption	obecně	viz. kolekce
table	obecně	viz. kolekce
table	summary	nepoužitelné
td,th	obecně	viz. kolekce
td, th	abbr	může
td, th	align("left"|"center"|"right")	**musí**
td, th	axis	nepoužitelné
td, th	colspan	**musí**
td, th,	headers	nepoužitelné
td, th	rowspan	**musí**
td, th	scope("row"|"col")	nepoužitelné
td, th	valign("top"|"middle"|"bottom")	**musí**
tr	obecně	viz. kolekce
tr	align("top"|"middle"|"bottom")	**musí**
tr	valign("top"|"middle"|"bottom")	**musí**

V případě, že v elementech `td` a `tr` chybí atribut `align`, nebo má nesprávné parametry, musí tiskárna reagovat, jako že je zarovnání vlevo. U atributu `th`, pak na střed.

Tiskárna musí zpracovávat hodnoty `top`, `middle` a `bottom` u atributu `valign`. V případě, že atribut chybí, nebo má nesprávné parametry, obsah by měl být ve vertikálním směru zarovnán na střed.

5.2.9. Obrázkový modul

Elementy	Atributy	Zpracování
img	obecně	viz. kolekce
img	alt	**musí**
img	height	**musí**
img	longdecs	nepoužitelné
img	src	**musí**
img	width	**musí**

Podrobnější práce s atributy `width` a `height` respektive s jejich absencí se řídí následujícími pravidly.

- pokud se tiskárna setká s obrázkem v nepodporovaném formátu, musí zobrazit alternativní obsah. Tím je myšlen prázdný prostor vymezený velikostí parametrů `width` a `height`. Zobrazit rámeček okolo obrázku již povinné není.
- `pokud alternativní obsah není k dispozici, obrázek je ignorován a není mu vyhrazen žádný prostor v tiskovém výstupu`
- `jestliže je formát obrázku podporován, ale neobsahuje atributy width a height, tiskárna se musí pokusit tisknout obrázek v jeho původní velikosti.`

5.2.10. Objektový modul

Elementy	Atributy	Zpracování
object	obecně	viz. kolekce
object	archive	nepoužitelné
object	classid	nepoužitelné
object	codebase	**musí**

object	codetype	nepoužitelné
object	data	**musí**
object	declare("declare")	může
object	height	**musí**
object	name	nepoužitelné
object	standby	nepoužitelné
object	tabindex	nepoužitelné
object	type("image/jpeg“)	**musí**
object	width	**musí**
param	id	nepoužitelné
param	name	nepoužitelné
param	type	nepoužitelné
param	value	nepoužitelné
param	valuetype("data"|"ref"|"object")	nepoužitelné

Tiskárna musí podporovat atribut `type("image/jpeg“)`. Jiné formáty obrázků, resp. jiné hodnoty parametru podporovat může.

5.2.11. Modul metainformací

Elementy	Atributy	Zpracování
meta	I18N	viz. kolekce
meta	content	nepoužitelné
meta	http-equiv	nepoužitelné
meta	name	nepoužitelné
meta	scheme	nepoužitenlné

Většina metainfomací o stránce je netisknutelná. Pouze informace o použité znakové sadě, mohou být tiskárnou implementovány.

5.2.12. Modul skriptů

Elementy	Atributy	Zpracování
noscript	obecně	viz. kolekce
script	charset	nepoužitelné
script	defer("defer")	nepoužitelné
script	src	nepoužitelné
script	type	nepoužitelné
script	scheme	nepoužitelné

Skripty jsou programy vykonávané ve spojitosti s webovým dokumentem. Nejčastěji jsou psaná v jazyce JavaScript. Pro tištěnou stránku nejsou důležité a nesmí být tisknuty.

5.2.13. Modul style

Elementy	Atributy	Zpracování
style	I18N	viz. kolekce
style	media	měl by
style	title	nepoužitelné
style	type("text/css")	**musí**
style	xml:space="preserve"	měl by

Tiskárna musí zpracovat obsah elementu `style`, pokud atribut media má hodnotu `print` nebo `all`. Jiný parametr by měla ignorovat. Absence atributu `media` musí být brána jako `media("all")`.

Atribut `type("text/css")` **musí** být zpracován a jiné hodnoty musí být ignorovány.

5.2.14. Modul link

Elementy	Atributy	Zpracování
link	obecně	viz. kolekce
link	charset	**musí**
link	href	**musí**
link	hreflang	může
link	media	**musí**
link	rel("stylesheet")	**musí**
link	rev	nepoužítelné
link	type("text/css")	**musí**

Jestliže tiskárna podporuje zpracování přirozeného jazyka, atribut hreflang musí být zpracován.

Tiskárna musí zpracovat obsah externího CSS souboru, pokud má atribut media hodnotu print nebo all. Pokud má atribut hodnotu projection, tiskárna jej může zpracovat. Dále by měla ignorovat jakýkoliv jiný parametr. V případě, že se atribut media v elementu nevyskytuje, musí jej zpracovat, jako by hodnota byla all. Toto je stejná situace jako v modulu style.

5.2.15. Modul base

Elementy	Atributy	Zpracování
base	href	**musí**

Element base se deklaruje v hlavičce dokumentu a má různé využití. Obecně slouží jako vlastnost, která je společná pro některé další elementy v těle. V testovaném souboru element base určuje absolutní cestu ke všem vloženým obrázkům.

```
<head>
      <base href="http://home.pf.jcu.cz/~zlukya00/BAK/Kap.5-
XHTML-Print/imgs/" />
```

```
</head>
<body>
<img src="1.gif" />
</body>
```

5.3. Výsledky testování

O tom, že Internet Explorer 7 stránku neotevře, pouze ji nabídne ke stažení jsme se zmínili již několikrát. Testovaní specifikace se tedy týkalo pouze prohlížečů Opera a FireFox.

Testování probíhalo způsobem, že jsem porovnával testovací webovou stránku z monitoru počítače se stránkou vytisknutou na papír.Lepší výsledek měl prohlížeč Opera. Veškeré elementy a atributy, které podle specifikace XHTML-Print měly být vytisknuty, vytisknuty byly.

Internetový prohlížeč FireFox nepodporuje jazyk ve dvou bodech. V prvním případě nebyl vytisknut atribut `selected="selected"`, což je zajímavé zvláště v kontextu s tím, že atribut `checked="checked"` vytisknut byl. Druhý nepodporovaný element je však mnohem zásadnější. Jedná se o element `style`. FireFox při tisku nezohlednil globální kaskádový styl. A to i přesto, že měl nastaven `media="all"`.

Poslední část, ve které se testované prohlížeče neshodly je reprezentace alternativního textu obrázku. FireFox vytiskl prostý text, kdežto Opera vytiskla rámeček určený parametry `width` a `height` a v něm alternativní text.

6. Závěr

V bakalářské práci jsem měl za úkol zmapovat některé aktuální značkovací jazyky konsorcia W3C. Jejich výsledkům v testech funkčnosti a podpory jednotlivými prohlížeči se věnuji na konci každé kapitoly.

Co se opakovalo v závěru každé kapitoly, a co je mým nejobecnějším shrnutím z celé práce je, že Microsoft Explorer 7 nepodporuje MIME typ application/xhtml+xml. To je podle mě největší a nejzásadnější problém tohoto oboru. S nulovou podporou Microsoftu se tyto nové značkovací jazyky nikdy nestanou běžným standardem webových stránek, vyhledávače je budou ignorovat a tím pádem je nebudou používat ani webmasteři. Je škoda, že v takto nastavených podmínkách nebudou zatím tyto technologie reálně použity a tím pádem nebudou využity jejich silné a odhaleny jejich slabé stránky.

7. Seznam citací

[1] Tim Berners-Lee *W3C : world wide web consortium* [online]. $1994-2007 v 1.120 2008/02/19 19:12:07 [cit. 2008-03-5]. Text v angličtině. Dostupný z WWW: <http://www.w3.org/People/Berners-Lee/>.

[2] STACHOWIAK, Maciej . *http://webkit.org/ : The WebKit Open Source Project* [online]. 2007. WebKit, 2007 , January 17th, 2007 [cit. 2008-04-04]. Text v angličtině. Dostupný z WWW: <http://webkit.org/blog/89/html-standards-process-returning-from-the-grave/>.

[3] W3C : world wide web consortium [online]. W3C, 1999-2007 , Last updated: 2008/01/10 [cit. 2008-02-20]. Text v angličtině. Dostupný z WWW: <http://www.w3.org/MarkUp/Forms/2003/xforms-faq>.

[4] W3C : world wide web consortium [online]. W3C, 1999-2007 , Last updated: 2008/01/10 [cit. 2008-02-20]. Text v angličtině. Dostupný z WWW: <http://www.w3.org/MarkUp/Forms/2003/xforms-faq>.

[5] KOSEK, Jiří. *Vše o WWW : Kapitola 3. XML schémata* [online]. 1999-2996 , Poslední modifikace: $Date: 2007/12/15 10:48:54 $ [cit. 2008-01-05]. Text v češtině. Dostupný z WWW: <http://www.kosek.cz/xml/schema/wxs.html>.

[6] *W3C : world wide web consortium* [online]. $1994-2007 , Proposed Recommendation 31 January 2006 [cit. 2008-02-27]. Text v angličtině. Dostupný z WWW: <http://www.w3.org/TR/2006/PR-xhtml-print-20060131/>.

8. Seznam použitých zdrojů

[1] SGML
http://www.kosek.cz/clanky/cw/sgml.html

[2] XHTML 2.0
http://www.w3.org/TR/xhtml2/

[3] XHTML Modularization
http://www.w3.org/TR/xhtml-modularization/

[4] Future of HTML
http://www.ibm.com/developerworks/xml/library/x-futhtml2.html

[5] XHTML 2.0 the Latest Trick
http://www.xml.com/pub/a/2002/08/07/deviant.html

[6] HTML 5, XHTML 2
http://webkit.org/blog/89/html-standards-process-returning-from-the-grave/#comment-17195

[7] W3 schools
http://www.w3schools.com

[8] Čurdová Lucie, Wap a jazyk WML
http://home.pf.jcu.cz/~pepe/Diplomky/curdova.doc

[9] Červenka Petr, Dynamické HTML pro Internet Explorer
http://home.pf.jcu.cz/~pepe/Diplomky/cervenka.doc

[10] Mozzila addons
https://addons.mozilla.org/cs/firefox/addon/824

[11] XForms for HTML authors
http://www.w3.org/MarkUp/Forms/2003/xforms-for-html-authors.html

[12] XForms for HTML authors, Part 2
http://www.w3.org/MarkUp/Forms/2006/xforms-for-html-authors-part2.html

[13] XForms validator
http://xformsinstitute.com/validator/
[14] Update Xforms using XForms
http://www.ibm.com/developerworks/xml/library/x-xforms4xforms/?S_TACT=105AGX52&S_CMP=cn-a-x#download
[15] Improving the Web Forms Experience
http://www.w3.org/2004/Talks/10-steven-xformstutorial/
[16] Understanding Xforms: Events and Actions
http://www.oreillynet.com/xml/blog/2006/08/understanding_xforms_events_an.html
[17] Need with XForms
http://www.nabble.com/Need-help-with-XForms-td13918254.html
[18] XForms extensions to XPath
http://www.ibm.com/developerworks/library/x-xformsextensions/#main
[19] XForms 1.0 Frequently asked questions
http://www.w3.org/MarkUp/Forms/2003/xforms-faq.html#advantages
[20] XForms Controls
http://www.orbeon.com/ops/xforms-controls/
[21] XHTML-Print
http://www.w3.org/TR/2006/PR-xhtml-print-20060131/
[22] PeadDr. Petr Pexa, Tvorba WWW a WAP