VYSOKÁ ŠKOLA EKONOMICKÁ V PRAZE

Fakulta informatiky a statistiky

Katedra informačního a znalostního inženýrství

Obor: Aplikovaná informatika

Podpora XML v moderních webových prohlížečích

Bakalářská práce

Kateřina Mansfeldová

Vedoucí práce: Ing. Jiří Kosek

2012

Anotace

Práce nabídne bližší pohled na technologie založené na XML či pracující s XML. Všechny tyto technologie mohou být využívány v prohlížečích. Cílem této práce je poskytnout ucelený přehled podpor v současných moderních prohlížečích, kam budou zahrnovány vzhledem k možnostem i prohlížeče mobilní.

K dosažení cíle budou použity testy daných technologií. Technologie budou vybírány podle jejich předpokládané podpory, následovat bude jejich krátké představení a přiblížení. Předpokladem práce však je, že potenciální čtenáři se v XML technologiích již vyznají.

Annotation

This thesis offers a closer look into technologies based on XML or working with XML. All these technologies can be used in modern browsers. The goal of the thesis is to give a comprehensive overview of XML support in today's modern browsers, with mobile browsers being included.

To reach the object of the thesis, different tests will be used. The technologies are chosen by their expected support, followed by their short introduction and the closer look. The thesis assumes that readers are familiar with XML technologies.

Poděkování

Chtěla bych poděkovat všem, kteří mě až doposud podporovali ve studiu. Zvláštní poděkování patří Ing. Jiřímu Koskovi za cenné rady a pomoc při tvorbě této práce.

Prohlášení

Prohlašuji, že jsem bakalářskou práci na téma *Podpora XML v moderních webových prohlížečích* vypracovala samostatně a použila jsem pouze literaturu uvedenou v přiloženém seznamu. Nemám námitek proti půjčení práce se souhlasem katedry ani proti zveřejnění práce nebo její části.

V Praze dne

Kateřina Mansfeldová

Obsah

Úvod

XML (*Extensible Markup Language*) řadíme mezi značkovací jazyky. V jazyku XML máme do jisté míry volnost, jako například v pojmenovávání elementů a také máme možnost pomocí mnohých prostředků vytvářet vlastní pravidla. To nám umožňuje vytvářet dokumenty, které jsou vytvořeny podle přesných pravidel, jaká potřebujeme. Dokumenty se poté zobrazují tak, jak požadujeme. XML data je možné transformovat do jiných datových formátů.

Pomocí již dříve vytvořených značkovacích jazyků založených na XML je možné vyvářet formuláře, vektorovou grafiku, animace a další. Díky pravidlům, která musí splňovat každé XML, můžeme vytvářet nad XML daty dotazy, exportovat pouze potřebná data či je vkládat. Vzhledem k rozvoji webových aplikací a zpracování dat pomocí prohlížečů si myslím, že by bylo vhodné využívat XML ve větší míře, než je tomu doposud. XML je šikovným nástrojem se širokou škálou možností, a přitom není jeho náročnost příliš vysoká.

Úskalím využití jazyka je to, že každý webový prohlížeč se jinak vypořádá se zobrazením dat či danou technologii vůbec nepodporuje. Proto je vhodné si před použitím otestovat funkčnost v jednotlivých prohlížečích.

Abych případným zájemcům, kteří se rozhodnou využít XML, ušetřila čas, či malou měrou podpořila využití XML v prohlížečích, rozhodla jsem se zaměřit na několik standardů souvisejících s XML a budu je testovat v paletě současně nejpoužívanějších webových a mobilních prohlížečů.

Pro přiblížení daných technologií se bude jednat v první řadě o zobrazení XML, které nám může být nápomocné k přehlednějšímu zobrazení kódu a k odhalení chyb v kódu, při jejichž hledání by nám mohly prohlížeče pomoci. Dále bych se zajímala o různé transformace XML do uživatelsky přívětivější formy (s pomocí CSS (*Cascading Style Sheets*) a XSLT (*Extensible Stylesheet Language Transformations*). Při využití CSS nezměním jazyk daného dokumentu, pouze upravím jeho vzhled, na rozdíl od XSLT, kde změním jazyk (z XML do HTML (*HyperText Markup Language*), XHTML (*eXtensible HyperText Markup Language*) atd.). Vzhled je také ovlivněn, ale to je dáno přednastaveným formátováním daných jazyků a samozřejmě i zde můžu využít CSS k osobitějšímu naformátování.

Pokud hovořím o využití ve webových prohlížečích, bude mě zajímat vložení celého XML či jeho části do (X)HTML. Vzhledem k této problematice jsem si vybrala DOM (*Document Object Model*), Data Islands, XMLHttpRequest, XInclude (*XML Inclusions*) a XPath (*XML Path Language*). Pomocí těchto nástrojů můžu manipulovat s daty, vytvářet dotazy nad daty a samozřejmě data vkládat.

Abych měla jistotu, že data budou mít formu, kterou požaduji, vytvořím si pomocí různých schémat pravidla, která určují například jména elementů v dokumentu, která mohou či musí být použita. Schémata se budu zabývat jen okrajově při načítání externích entit. Avšak pro kontrolu XML podle daného schématu je nutná validace, kterou se také budu zabývat. Ke správnému zobrazení patří i podpora MIME (*Multipurpose Internet Mail Extensions*) typů a podpora jmenných prostorů. Práci nám může usnadnit i podpora XML Base či xml:id, obdoba těchto

atributů existuje i v HTML. Avšak z mého pohledu je v obou případech lepší využít XML variantu. Proto v mé práci najdete kapitoly zabývající se těmito atributy.

A nakonec se budu zabývat jazyky založenými na XML, které do webových stránek přináší grafiku (SVG), animace (SMIL), formuláře (XForms), interaktivnost (XBL) a v neposlední řadě například i usnadnění (MathML).

V průběhu práce se vždy zmíním o daném tématu, přiložím testovací kód a nakonec očekávaný výstup testovacího kódu. Výsledky testu v nejpoužívanějších webových prohlížečích uvedu na konci své práce.

Kapitola 1

Zobrazení XML bez stylů

Zobrazení XML bez jakéhokoliv stylu by mělo vypadat jako kód, který jsme napsali. Měl by být vidět kód, protože prohlížeče neví, co dané značky znamenají, zda třeba element head neznačí typ hlavy, protože daný dokument je o výzkumu tvaru lidského těla. Prohlížeče většinou dokáží zobrazit XML jako přehledný kód. Za přehledný považuji zobrazení ve stromové struktuře, zvýrazněný a případně s možností rozbalování jednotlivých elementů. XML se zobrazí pouze v případě, že je struktura XML bezchybná. V opačném případě kód není zobrazen.

Kód pro testování zobrazení

kucharka.xml

```
<kucharka xmlns="http://mansfeldova.com/namespaces/cooking/1.0">
 <recept id="0001">
  <nazev>kakao</nazev>
  <ingredience>
   <prisada>
    <mnozstvi zpusob_mereni="hrnek">1</mnozstvi>
    <druh>mléko</druh>
   </prisada>
   <prisada>
    <mnozstvi zpusob_mereni="lžička">3</mnozstvi>
    <druh>kakao</druh>
   </prisada>
  </ingredience>
  <postup>Mléko ohřejeme a vmícháme kakao. Lze podávat i studené</postup>
 </recept>
</kucharka>
```

U prohlížečů, které pracují pod operačním systémem Microsoft Windows je standardní stav zobrazení XML s barevným odlišením, zobrazený ve stromové struktuře a s možností rozbalování, jako vidět na obrázku 1.1 – „Správně zobrazené XML“. Ve většině testovaných desktopových prohlížečů je tento stav běžný. Odklonění od tohoto stavu je popsáno v tabulce, která se nachází v kapitole testování prohlížečů 18 – „*Testování prohlížečů*“. U testovaných mobilních prohlížečů se XML zobrazuje jako HTML. Zobrazí se pouze text, který je psaný v elementech.

Obrázek 1.1. Správně zobrazené XML

Chybný XML kód. Chyba je ve špatném koncovém tagu „dlruh“ (místo „druh“) a dále zcela chybí koncový tag „prisada“.

kucharka_chybna.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<kucharka xmlns="http://mansfeldova.com/namespaces/cooking/1.0">
  <recept id="0001">
    <nazev>kakao</nazev>
    <ingredience>
      <prisada>
        <mnozstvi zpusob_mereni="hrnek">1</mnozstvi>
        <druh>mléko</druh>
      </prisada>
      <prisada>
        <mnozstvi zpusob_mereni="lžička">3</mnozstvi>
        <druh>kakao</dlruh>
    </ingredience>
    <postup>Mléko ohřejeme a vmícháme kakao. Lze podávat i studené</postup>
  </recept>
</kucharka>
```

Při testování zobrazení XML s chybou považuji za výchozí stav ohlášení názvu první chyby, vypsání přesné polohy chyby a ukázku části kódu, kde se chyba nachází. Ukázka na obrázku 1.2 – „Správně zobrazené XML s chybou“. Při jiném zobrazení než výchozím popíši odlišnosti v tabulce. 18 – „*Testování prohlížečů*“

Obrázek 1.2. Správně zobrazené XML s chybou

Kapitola 2

XSL a CSS

2.1. Přímé prohlížení XML pomocí CSS

Zobrazení XML pomocí CSS (*Cascading Style Sheet*) a XSL (*Extensible Stylesheet Language*) je pro uživatele mnohem pohodlnější. XSL je více než jen šablona stylů. Skládá se ze tří částí a těmi jsou XSLT (*Extensible Stylesheet Language Tranformations*), XSL-FO (*Extensible Stylesheet Language Formatting Objects*) a XPath. XPath je dotazovací jazyk pro navigaci v dokumentu a budu se jím více zabývat později. XSL-FO nám umožňuje nadefinovat vzhled XML například pro výstup do PDF. Pro moji práci využiji XSLT, pomocí kterého je možné XML transformovat do jiného XML nebo do jiného typu dokumentu, který je podporovaný prohlížeči, jako například HTML či XHTML. XSLT podrobněji rozeberu v následující kapitole. [17]

S CSS máme možnost XML kód pouze naformátovat. Avšak stále je to XML, jen se zobrazuje bez značek a graficky zpracovaný. Na rozdíl od XSLT, CSS byly vytvořeny k formátování. CSS můžeme připojit k XML souboru a nebo XML soubor transformovat pomocí XSLT do HTML bez stylů a následně přidat CSS styl. CSS styl se skládá z pravidel, které tvoří selektor a jedna či více deklarací. Selektor nám ukazuje, co se bude v daném XML upravovat a deklarace nám specifikuje způsob zobrazení. Selektor může ukazovat na jeden či více elementů dohromady, a pokud použijeme znak „*“, styl se aplikuje na celý dokument. V XML, kde používáme mnoho atributů bychom rádi jejich elementy v jistých případech odlišovali. CSS nám nabízí hned několik možností. Můžeme selektorem označit element obsahující zvolený atribut takto: „element[jméno_atributu]“, nebo atribut s určitou hodnotou „element[jméno_atributu=hodnota_atributu]“. Pokud máme atribut ID, pak má pro nás CSS zjednodušení v podobě označení elementu s danou hodnotou ID („element#hodnota_ID“). V kódu budu testovat nejen tyto selektory, ale také podporu jmenných prostorů. K tomuto účelu použiji dvě různé XML (jedno vlastní jeden jmenný prostor a druhé má dva jmenné prostory) a tři styly CSS (první bez jmenného prostoru, druhý se jmenným prostorem a třetí se dvěma prostory). [21]

Bude následovat kód na testování CSS s jedním jmenným prostorem pro testování CSS bez jmenného prostoru a CSS s jedním jmenným prostorem. Pro testování s CSS se jmennými prostory zaměním pouze v daném kódu odkaz „ke_kucharce.css“ za „kucharkaNSglobal.css“. Výsledný testovací kód má název: „css-styl2.xml“.

```
css-styl1.xml
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<?xml-stylesheet type="text/css" href="ke_kucharce.css"?>
 <kucharka xmlns="http://mansfeldova.com/namespaces/cooking/1.0">
```

```
  <recept id="t0001">
   <nazev>kakao</nazev>
   <ingredience>
    <prisada>
     <mnozstvi zpusob_mereni="hrnek">1</mnozstvi>
     <druh>mléko</druh>
    </prisada>
    <prisada>
     <mnozstvi zpusob_mereni="lzicka">3</mnozstvi>
     <druh>kakao</druh>
    </prisada>
   </ingredience>
   <postup>Mléko ohřejeme a vmícháme kakao. Lze podávat i studené</postup>
  </recept>
  <recept id="t0002">
   <nazev>káva s mlékem</nazev>
   <ingredience>
    <prisada>
     <mnozstvi zpusob_mereni="hrnek">0,8</mnozstvi>
     <druh>voda</druh>
    </prisada>
    <prisada>
     <mnozstvi zpusob_mereni="hrnek">0,2</mnozstvi>
     <druh>mléko</druh>
    </prisada>
    <prisada>
     <mnozstvi zpusob_mereni="lzicka">2</mnozstvi>
     <druh>káva</druh>
    </prisada>
   </ingredience>
   <postup>Vodu uvaříme. Do hrnku dáme kávu, zalijeme vodou a dolejeme ▶
mlékem.</postup>
  </recept>
</kucharka>
```

XML kód se dvěma jmennými prostory pro využití CSS se dvěma jmennými prostory

css-styl3.xml

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<?xml-stylesheet type="text/css" href="kucharkaNSglobal2.css"?>
<kucharka xmlns="http://mansfeldova.com/namespaces/cooking/1.0"
    xmlns:ck="http://mansfeldova.com/namespaces/cook/1">
  <recept id="t0001">
      <nazev>kakao</nazev>
      <ingredience>
          <prisada>
```

```
            <mnozstvi zpusob_mereni="hrnek">1</mnozstvi>
            <druh>mléko</druh>
        </prisada>
        <ck:prisada>
            <ck:mnozstvi zpusob_mereni="lzicka">3</ck:mnozstvi>
            <ck:druh>kakao</ck:druh>
        </ck:prisada>
    </ingredience>
    <postup>Mléko ohřejeme a vmícháme kakao. Lze podávat i ▶
studené</postup>
  </recept>
  <recept id="t0002">
    <nazev>káva s mlékem</nazev>
    <ingredience>
        <prisada>
            <mnozstvi zpusob_mereni="hrnek">0,8</mnozstvi>
            <druh>voda</druh>
        </prisada>
        <prisada>
            <mnozstvi zpusob_mereni="hrnek">0,2</mnozstvi>
            <druh>mléko</druh>
        </prisada>
        <prisada>
            <mnozstvi zpusob_mereni="lzicka">2</mnozstvi>
            <druh>káva</druh>
        </prisada>
    </ingredience>
   <postup>Vodu uvaříme. Do hrnku dáme kávu, zalijeme vodou a dolejeme ▶
mlékem.</postup>
  </recept>
</kucharka>
```

Následují CSS kódy, nejdříve CSS bez definovaného jmenného prostoru.

ke_kucharce.css

```
*{
color:black;
width:500px;
margin:auto;
display:block;
}
recept{
border:1px solid red;
background-color:orange;
}
recept[id="t0001"]{
```

```
background-color:yellow;
}
ingredience:before{
content:"Ingredience:";
position:relative;
left:-60px;
}
ingredience{
position:relative;
left:60px;
}
prisada{
display:list-item;
}
ingredience>prisada{
position:relative;
}
mnozstvi:after{
content:"x";
}
druh{
display:inline;
}
mnozstvi[zpusob_mereni=lzicka]{
background-image:url(lzicka.jpg);
background-repeat:no-repeat;
background-position:left;
display:inline;
}
recept#t0002{
color:green;
}
mnozstvi[zpusob_mereni]{
display:inline;
content:"x "attr(zpusob_mereni);
height:10px;
width:10px;
}
recept:nazev{
text-decoration:underline;
text-transform:capitalize;
}
nazev{
text-decoration:underline;
text-transform:uppercase;
}
```

S definovaným globálním jmenným prostorem. Od předchozího CSS se liší pouze prvním řádkem, proto pro zkrácení použiji „(...)“

kucharkaNSglobal.css

```
@namespace "http://mansfeldova.com/namespaces/cooking/1.0";
*{
color:black;
width:500px;
margin:auto;
display:block;
(...)
nazev{
text-decoration:underline;
text-transform:uppercase;
}
```

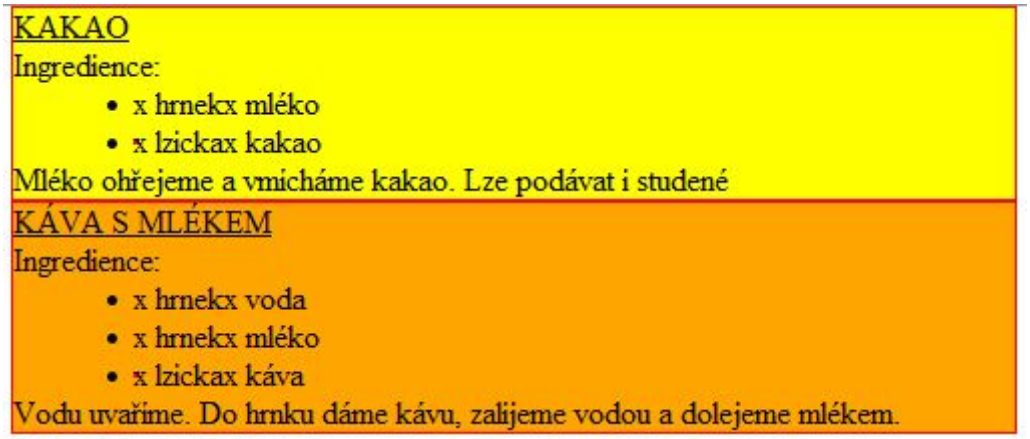
A nakonec CSS se dvěma jmennými prostory

kucharkaNSglobal2.css

```
@namespace "http://mansfeldova.com/namespaces/cooking/1.0";
@namespace ck "http://mansfeldova.com/namespaces/cook/1";
*{
color:black;
width:500px;
margin:auto;
display:block;
}
recept{
border:1px solid red;
background-color:orange;
}
recept[id="t0001"]{
background-color:yellow;
}
ingredience:before{
content:"Ingredience:";
position:relative;
left:-60px;
}
ingredience{
position:relative;
left:60px;
}
ck|prisada{
display:list-item;
}
```

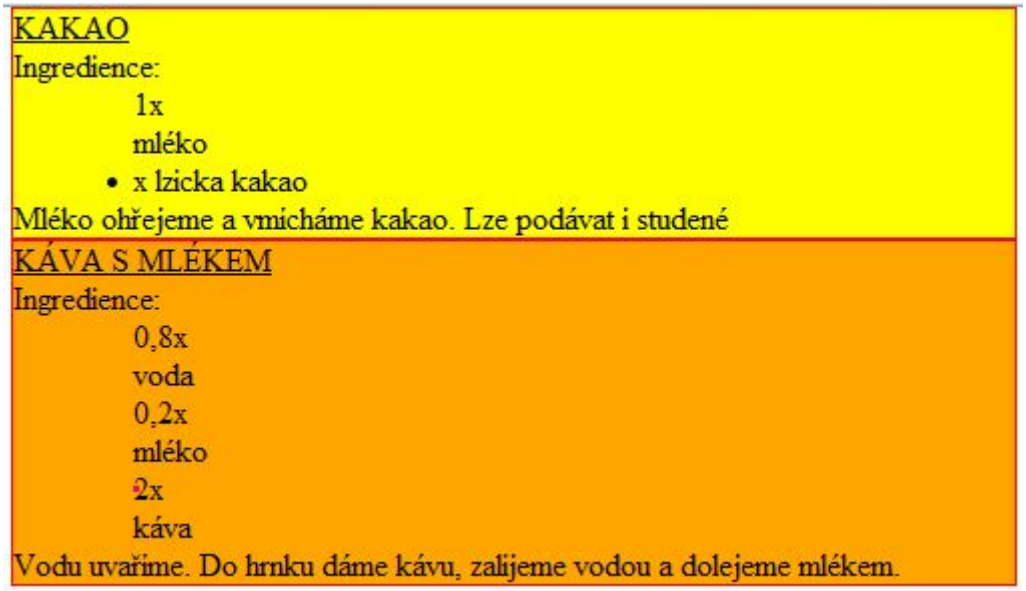
```
ingredience>prisada{
position:relative;
}
mnozstvi:after{
content:"x";
}
ck|druh{
display:inline;
}
mnozstvi[zpusob_mereni=lzicka]{
background-image:url(lzicka.jpg);
background-repeat:no-repeat;
background-position:left;
display:inline;
}
recept#t0002{
color:green;
}
ck|mnozstvi[zpusob_mereni]{
display:inline;
content:"x "attr(zpusob_mereni);
height:10px;
width:10px;
}
recept:nazev{
text-decoration:underline;
text-transform:capitalize;
}
nazev{
text-decoration:underline;
text-transform:uppercase;
}
```

Žádný z testovaných prohlížečů nepodporuje výběr elementu podle id („recept#t0002“), proto jsem usoudila, že výchozí stav bude podpora všeho ostatního kromě výběru elementu podle id. Pouze Opera podporuje vepsání hodnoty atributu pomocí CSS. Žádný z testovaných prohlížečů nemá rozdíl mezi CSS bez jmenného prostoru a CSS s jedním jmenným prostorem připojených k XML s jedním jmenným prostorem. V dokumentacích jsem nenašla pravidlo, že CSS bez jmenného prostoru ignoruje v XML jmenný prostor, ale nenašla jsem ani opak tohoto tvrzení. Ve výsledcích testu zmiňuji tuto vlastnost, avšak nepovažuji ji za nepodporu. Výchozím stavem pro první dva testy považuji následující obrázek: 2.1 – „Správně zobrazené XML s CSS stylem“



Obrázek 2.1. Správně zobrazené XML s CSS stylem

Na následujícím obrázku vidíme, jak bude vypadat s podporou dvou jmenných prostorů.



Obrázek 2.2. Správně zobrazené XML s CSS stylem a podporovou dvou jmenných prostorů

2.2. Podpora XSLT

XSLT, jak jsem již napsala výše, bylo primárně vytvořené pro transformaci XML do jiného XML. Avšak je možné ho použít na transformaci do HTML a do dalších textově založených formátů. Jednodušeji by se dalo říci, že XSLT je jazyk pro změnu struktury a obsahu XML dokumentu.[26]

Momentálně je nejnovější verze XSLT 3.0. Novou funkcí této verze je možnost proudového zpracování. Pokud byl v nižších verzích zpracováván příliš velký dokument, zaplnila se celá virtuální paměť a to znemožnilo provedení transformace. Po podrobném prozkoumání stránek pro podporu a developerských fór jsem zjistila, že je podpora XSLT 3.0 ještě v daleké budoucnosti. Většina prohlížečů nepodporuje ani XSLT 2.0 i přestože finální specifikace byla vydána

v lednu 2007. Podpora XSLT 1.0 by měla být ve většině prohlížečů. A také je ve všech nejznámějších desktopových prohlížečích podporována. Pokud tedy máme XML kód a k němu příslušný XSLT styl, pak pomocí hlavičky v XML dokumentu „<?xml-stylesheet type="text/xsl" href="styly.xsl"?>“ je přidán k dokumentu a další práci dělá prohlížeč za nás. [12][13]

Následuje styl, pomocí kterého budu transformovat XML.

styl.xsl

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" ▶
version="1.0"
    xmlns:cook="http://mansfeldova.com/namespaces/cooking/1.0">
  <xsl:output method="html" encoding="UTF-8"/>
  <xsl:template match="/">
      <html>
          <head>
              <title>
                  <xsl:value-of select="cook:kucharka/cook:nazev"/>
              </title>
              <style type="text/css">
                  body { color:black;
                  }
                  h1 { color:red;
                      text-decoration:underline;
                  }
                  h2 { text-decoration:none;
                  }
                  ol { color:orange;
                  }</style>
          </head>
          <body>
              <h1>
                  <xsl:value-of select="cook:kucharka/cook:nazev"/>
              </h1>
              <xsl:apply-templates select="cook:kucharka/cook:recept"/>
          </body>
      </html>
  </xsl:template>
  <xsl:template match="cook:recept">
      <h2>
          <xsl:apply-templates select="cook:nazev"/>
      </h2>
      <ol>Ingredience <xsl:apply-templates select="cook:ingredience"/>
      </ol>
      <xsl:apply-templates select="cook:postup"/>
  </xsl:template>
  <xsl:template match="cook:postup">
```

```
      <h3>POSTUP:</h3>
      <p><xsl:value-of select="."/></p>
  </xsl:template>
  <xsl:template match="cook:ingredience">
      <xsl:for-each select="cook:prisada">
          <xsl:choose>
              <xsl:when test="cook:mnozstvi/@zpusob_mereni='hrnek'">
                  <li><xsl:value-of select="cook:mnozstvi"/> hrnek ▶
<xsl:value-of select="cook:druh"/></li>
              </xsl:when>
              <xsl:when test="cook:mnozstvi/@zpusob_mereni='lzicka'">
                  <li>
                      <xsl:value-of select="cook:mnozstvi"/>lžička ▶
<xsl:value-of select="cook:druh"/></li>
              </xsl:when>
              <xsl:otherwise/>
          </xsl:choose>
      </xsl:for-each>
  </xsl:template>
</xsl:stylesheet>
```

Kód k transformaci

kucharka-xsl.xml

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<?xml-stylesheet type="text/xsl" href="styl.xsl"?>
<kucharka xmlns="http://mansfeldova.com/namespaces/cooking/1.0">
  <nazev>Nápoje</nazev>
  <recept id="t0001">
      <nazev>kakao</nazev>
      <ingredience>
          <prisada>
              <mnozstvi zpusob_mereni="hrnek">1</mnozstvi>
              <druh>mléko</druh>
          </prisada>
          <prisada>
              <mnozstvi zpusob_mereni="lzicka">3</mnozstvi>
              <druh>kakao</druh>
          </prisada>
      </ingredience>
      <postup>Mléko ohřejeme a vmícháme kakao. Lze podávat i ▶
studené</postup>
  </recept>
  <recept id="t0002">
      <nazev>káva s mlékem</nazev>
      <ingredience>
```

```
            <prisada>
                <mnozstvi zpusob_mereni="hrnek">0,8</mnozstvi>
                <druh>voda</druh>
            </prisada>
            <prisada>
                <mnozstvi zpusob_mereni="hrnek">0,2</mnozstvi>
                <druh>mléko</druh>
            </prisada>
            <prisada>
                <mnozstvi zpusob_mereni="lzicka">2</mnozstvi>
                <druh>káva</druh>
            </prisada>
        </ingredience>
        <postup>Vodu uvaříme. Do hrnku dáme kávu, zalijeme vodou a dolejeme ▶
mlékem.</postup>
    </recept>
</kucharka>
```

Výsledek této transformace by měl vypadat takto: 2.3 – „Výstup testu XSLT transformace“

Obrázek 2.3. Výstup testu XSLT transformace

Transformace provedená tímto způsobem proběhne jen jednou. Pro statické aplikace to je vyhovující, ale v případě interaktivnějších aplikací je třeba provádět transformace v závislosti na požadavcích uživatele. V rámci efektivnosti by bylo vhodné provádět transformaci přímo na klientovi. JavaScriptové rozhraní není pro tyto účely jednotné pro všechny prohlížeče. Tento problém lze vyřešit pomocí knihovny Sarissa, která vyřeší implementační problémy. Knihovna Sarissy je volně přístupná ke stažení na stránkách: http://sourceforge.net/projects/sarissa/files/. Následující kód je demonstrací tohoto řešení.[24]

```
transformace-sarissa.html
<!DOCTYPE html SYSTEM ▶
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head><title>XSLT pomocí sarissy</title>
      <script type="text/javascript" ▶
src="http://mansfeldova.com/xml/sarissa-full-0.9.9.5/gr/abiss/js/sarissa/sarissa.js"></script>
      <script type="text/javascript">
          var xslDoc = Sarissa.getDomDocument();
          xslDoc.async = false;
```

```
            xslDoc.load("styl.xsl");
            var xslProc = new XSLTProcessor();
            xslProc.importStylesheet(xslDoc);
            var xmlDoc = Sarissa.getDomDocument();
            xmlDoc.async = false;
            xmlDoc.load("kucharka-xsl.xml");
            function test()
            {
            document.getElementById("xslVloz").innerHTML =
            new ▶
XMLSerializer().serializeToString(xslProc.transformToDocument(xmlDoc));
            }
        </script>
    </head>
    <body onload="test()">
        <div id="xslVloz"></div>
    </body>
</html>
```

Výstup by měl vypadat stejně jako je tomu u transformace pro statické aplikace.

2.3. Podpora XPath

XPath se řadí mezi dotazovací jazyk nad XML daty. Poskytuje nám možnost vybírat si uzly ze stromové prezentace XML. Již dříve jsem zmínila, že XPath je z rodiny XSL a je velice užitečný pro XSLT. Tento jazyk není založen na notaci XML. Jeho popis cest k elementům je podobný cestám, které popisuje operační systém k souborům. XPath využívá predikáty, čímž jazyk zvyšuje specifikaci uzlů. Predikát je Booleovský výraz zapisovaný do závorek „[]“ a umístěných na konec kroku. K dispozici je také aritmetika. Zde si musíme dát pozor na znak značící operaci dělení. Dělení se znázorňuje jako div nikoli jako „/“. Pro jazyk XPath je „/“ klíčové znaménko pro práci s uzly a určuje podle umístění a počtu, zda se bude jednat o relativní či absolutní cestu. Pokud by byl dotaz jen „/“, pak je vybrán celý dokument, nad kterým je dotaz zpracováván.[22]

XPath je velice mocným nástrojem s vysokým využitím a to nejen v XSLT. Budu testovat jeho podporu pomocí JavaScriptu s využitím rozhraní XMLHttpRequest. XMLHttpRequestu je věnována 10. kapitola. Test vypadá následovně:

```
xpath.html
<!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 4.01//EN'>
<html>
  <head>
    <title>Test XPath </title>
    <meta http-equiv="content-type" content="text/html;charset=utf-8"/>
    <script type="text/javascript">
     function loadXMLDoc(dname) {
      if (window.XMLHttpRequest) {
       xhttp=new XMLHttpRequest(); }
```

```
      else {
       xhttp=new ActiveXObject("Microsoft.XMLHTTP"); }
      xhttp.open("GET",dname,false);
      xhttp.send("");
      return xhttp.responseXML;
      }
    xml=loadXMLDoc("kucharkaVetsi.xml");
    function vyraz(path){
     if (window.ActiveXObject) {
      var nodes=xml.selectNodes(path);
      for (i=0;i<nodes.length;i++) {
        document.write(nodes[i].childNodes[0].nodeValue);
        }
       }
     else if (document.implementation && ▶
document.implementation.createDocument) {
     var nodes=xml.evaluate(path, xml, null, XPathResult.ANY_TYPE, null);
      if (nodes.resultType==1){
        document.write(nodes.numberValue);}
      else {
        var result=nodes.iterateNext();
      while (result) {
        document.write(result.childNodes[0].nodeValue);
        result=nodes.iterateNext();
      } } }
    }
  </script></head>
  <body>
   Nazev kucharky: <script ▶
type="text/javascript">vyraz("/kucharka/nazev");</script><br/>
   Počet ingrediencí měřených na hrnky: <script ▶
type="text/javascript">vyraz('count(//mnozstvi[@zpusob_mereni="hrnek"])');</script><br/>
   Počet lžiček používaných v kuchařce: <script ▶
type="text/javascript">vyraz('sum(//mnozstvi[@zpusob_mereni="lzicka"])');</script><br/>
  </body>
</html>
```

Výstup by měl vypadat jako na následujícím obrázku: 2.4 – „Výstup testu XPathu“

Název kuchařky: Nápojová kuchařka
Počet ingrediencí měřených na hrnky: 3
Počet lžiček používaných v kuchařce: 5

Obrázek 2.4. Výstup testu XPathu

Jak je z obrázku patrné, zjišťuji název kuchařky, dále podle hodnoty atributu „zpusob_mereni“ u elementu „mnozstvi“ sčítám počet elementů s hodnotou „hrnek“ a nakonec sčítám hodnoty elementů „mnozstvi“ s atributem „lzicka“.

Kapitola 3

Data Islands

XML Data Islands (*datové ostrůvky*) jsou prostředkem pro vkládání XML dat do webových stránek. Data se mohou vkládat přímo mezi HTML kód a nebo ho lze připojit externě. XML je uzavřeno v tagu „xml“ a musí mu být přiděleno id. Tato část se shoduje, ať už jsou Data Islands psané v kódu či zda je přidáváme z externího souboru. Vložené XML by se nemělo zobrazovat do té doby, dokud se na něj neodkážeme. Můžeme vyvolávat hodnoty elementů XML v elementu „span“, kde do atributu „datasrc“ vložíme id přiřazené vkládanému XML a do atributu „datafld“ vložíme jméno atributu. Můžeme využívat i složitější struktury XML, v HTML je pak náročnější vypsat vnořený element. V kódu níže je to demonstrováno na elementu „prisada“. Data Islands jsou podporovány pouze IE prohlížeči.

Pokud bychom chtěli pracovat s XML daty v jiných prohlížečích, můžeme využít HTML5 či XHTML, kde pracujeme s XML díky DOMParseru. XML se zde pomocí tagu „script“ vloží do (X)HTML. Je pro nás výhodnější vkládat XML data do (X)HTML pomocí scriptu. V jeho případě je podpora daleko větší, předpokladem je podpora u všech prohlížečů (včetně IE). Při vkládání do XHTML musíme dávat pozor, aby vše v elementech „script“ bylo vnořeno do elementu „<![CDATA]]>“.[15][14]

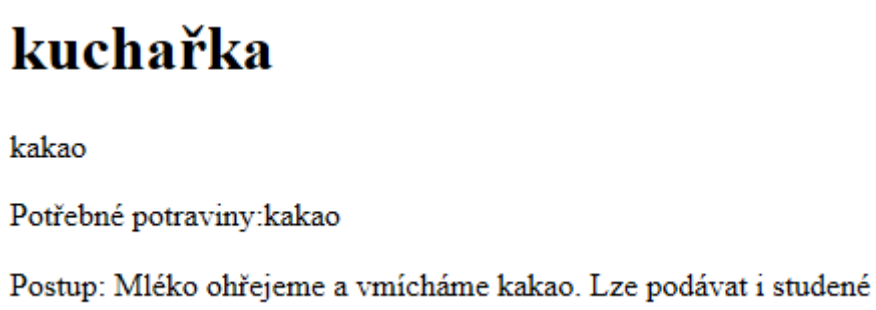
Testovat budu Data Islands vložená do HTML kódu a také vložení pomocí elementu script do HTML5 a XHTML. Nejdříve otestuji Data Islands :

dataIsland.html

```
<!DOCTYPE html SYSTEM ▶
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Test data islands
    </title>
    <meta http-equiv="content-type" content="text/html;charset=utf-8"/>
  </head>
  <body>
    <xml id="recept">
      <recept>
        <nazev>kakao
        </nazev>
        <mnozstvi zpusob_mereni="lzicka">3</mnozstvi>                    ▶
```

```
        <druh>kakao
        </druh>
      <postup>Mléko ohřejeme a vmícháme kakao. Lze podávat i studené
        </postup>
      </recept>
    </xml>        <h1>kuchařka</h1>
    <span datasrc="#recept" datafld="nazev">
    </span>
    <br/>
    <p>Potřebné potravinygg:
      <span datasrc="#recept" datafld="druh">
      </span>
      <br/>
      <p>Postup:
        <span datasrc="#recept" datafld="postup">
        </span>
  </body>
</html>
```

Výstup by měl vypadat takto:



Obrázek 3.1. Výstup testu na Data Islands

Následuje test na vložení XML pomocí elementu script do HTML5 s využitím DOMParseru:

```
data.html
<!DOCTYPE html>
<html lang="cs">
 <head>
<meta http-equiv="content-type" content="text/html;charset=utf-8"/>

 <title>XML Data Block Demo</title>
  <script id="kucharka" type="application/xml">
   <kucharka xmlns="http://mansfeldova.com/namespaces/cooking/1.0">
    <recept id="0001">
     <nazev>kakao</nazev>
```

```
     <ingredience>
      <prisada>
       <mnozstvi zpusob_mereni="hrnek">1</mnozstvi>
       <druh>mléko</druh>
      </prisada>
      <prisada>
       <mnozstvi zpusob_mereni="lžička">3</mnozstvi>
       <druh>kakao</druh>
      </prisada>
     </ingredience>
    <postup>Mléko ohřejeme a vmícháme kakao. Lze podávat i studené</postup>
   </recept>
  </kucharka>
 </script>
<script>
function runDemo() {
var kucharka = document.getElementById("kucharka").textContent;
var parser = new DOMParser();
var doc = parser.parseFromString(kucharka, "application/xml");
var prisady = ▶
doc.getElementsByTagNameNS("http://mansfeldova.com/namespaces/cooking/1.0", ▶
"prisada");
var druh = ▶
prisady[0].getElementsByTagNameNS("http://mansfeldova.com/namespaces/cooking/1.0", ▶
"druh")[0].textContent;
var nazev = ▶
doc.getElementsByTagNameNS("http://mansfeldova.com/namespaces/cooking/1.0", ▶
"nazev")[0].textContent;
document.body.textContent = "Na " + nazev + " potřebujeme " + ▶
prisady.length + " přísady. Jedna z nich se nazývá: " + ▶
prisady[0].textContent + ".";
}
</script>
</head>
<body onload="runDemo()";>        Zde nefunguje
</body>
</html>
</html>
```

Výsledek kódu

Na kakao potřebujeme 2 přísady. Jedna z nich se nazývá: mléko.

Obrázek 3.2. Výsledek kódu s využitím DOMParseru

Poslední kód na testování podpory Data Islands je podobný předchozímu, upravený pro XHTML. Následující kód by měl mít stejný výstup jako kód předchozí.

```
data.xhtml
<!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="en">
<head>
<meta http-equiv="content-type" ▶
content="application/xhtml+xml;charset=utf-8"/>
<title></title>
<script id="kucharka" type="application/xml"><![CDATA[
  <kucharka xmlns="http://mansfeldova.com/namespaces/cooking/1.0">
    <recept id="0001">
      <nazev>kakao</nazev>
       <ingredience>
        <prisada>
          <mnozstvi zpusob_mereni="hrnek">1</mnozstvi>
         <druh>mléko</druh>
        </prisada>
        <prisada>
         <mnozstvi zpusob_mereni="lžička">3</mnozstvi>
         <druh>kakao</druh>
        </prisada>
       </ingredience>
      <postup>Mléko ohřejeme a vmícháme kakao. Lze podávat i ▶
studené</postup>
     </recept>
  </kucharka>
  ]]&gt;
  </script>
  <script type="text/javascript">
    <![CDATA[
function runDemo() {
    var kucharka = document.getElementById("kucharka").textContent;
    var parser = new DOMParser();
    var doc = parser.parseFromString(kucharka, "application/xml");
    var prisady = ▶
doc.getElementsByTagNameNS("http://mansfeldova.com/namespaces/cooking/1.0", ▶
"prisada");
    var druh = ▶
prisady[0].getElementsByTagNameNS("http://mansfeldova.com/namespaces/cooking/1.0", ▶
"druh")[0].textContent;
    var nazev = ▶
doc.getElementsByTagNameNS("http://mansfeldova.com/namespaces/cooking/1.0", ▶
```

```
"nazev")[0].textContent;
    document.body.textContent = "Na " + nazev + " potřebujeme " + ▶
prisady.length + " přísady. Jedna z nich se nazývá: " + druh + ".";
}
]]&gt;
  </script>
  </head>
  <body onload="runDemo()">
    Zde nefunguje
  </body>
</html>
```

Kapitola 4

Podpora MIME typů

MIME (Multiporpose Internet Mail Extension) bylo původně používáno, jak je vidět z názvu, pro rozšíření možností elektronické pošty. Toto rozšíření umožňovalo posílat elektronickou poštou nejen textové zprávy psané v kódu ASCII, ale i animace, obrázky, zvuk a další. Princip výměny zpráv se rozšířil brzy i za hranice elektronické pošty a teď jej využívá nejen HTTP protokol. Hlavička MIME má několik hlaviček. Já se budu zabývat pouze o MIME type (formálně jde o položku jménem MIME Content-Type). Hodnota Content-Type určuje typ přenášených dat. Jde o znakový řetězec, který se skládá ze dvou částí. První z nich určuje základní typ dat a druhá jej upřesňuje. V této době existuje sedm možných hodnot základního typu: text, multipart, message, application, image a audio. Každý z těchto typů může být upřesněn podtypem. [18][20]

Pokud chceme ovlivňovat MIME typy posílaných souborů, musí náš server umožňovat nastavení pomocí konfiguračního souboru „.htacess“. Do něj vepíšeme například: „AddType text/xml .xxx“. Tím serveru řekneme, že soubor s příponou „.xxx“ je typu „text/xml“, a server se podle toho zachová. Rozhodla jsem otestovat, jak se zachovají prohlížeče, pokud jím pošlu bezchybný XML soubor a XML soubor s chybou. Oba tyto soubory budu postupně testovat se třemi příponami, které definuji v „.htaccess“, jak je znázorněno níže.

```
.htaccess
      AddType text/xml .xxx
      AddType application/xml .yyy
      AddType application/xhtml+xml .zzz
```

Následuje kód bez chyby:

```
MIME/kuchMime.(xxx; yyy; zzz)
<kucharka xmlns="http://mansfeldova.com/namespaces/cooking/1.0">
  <recept id="0001">
     <h1><nazev>kakao</nazev></h1>
      <ingredience>
          <prisada>
              <mnozstvi zpusob_mereni="hrnek">1</mnozstvi>
              <druh>mléko</druh>
          </prisada>
          <prisada>
```

```
            <mnozstvi zpusob_mereni="lžička">3</mnozstvi>
            <druh>kakao</druh>
        </prisada>
    </ingredience>
    <postup>Mléko ohřejeme a vmícháme kakao. Lze podávat i ▶
studené</postup>
  </recept>
</kucharka>
```

Pokud v prohlížeči funguje podpora MIME typů, zobrazí se soubor ve dvou případech jako XML (popsáno v první kapitole). Ve třetím případě - soubor „kuchMime.zzz“ by se měl zobrazit jako XHTML stránka s nadpisem „kakao“, zde jsem použila element h1. V případě chybného XML, jehož kód bude následovat, by se měla ve všech třech případech zobrazit stránka s chybou (i toto bylo popsáno v první kapitole).

```
MIME/kucha.(xxx; yyy; zzz)
<?xml version="1.0" encoding="UTF-8"?>
<kucharka xmlns="http://mansfeldova.com/namespaces/cooking/1.0">
    <recept id="0001">
        <h1><nazev>kakao</nazev></h1>
    <ingredience>
        <prisada>
            <mnozstvi zpusob_mereni="hrnek">1</mnozstvi>
            <druh>mléko</druh>
        </prisada>
        <prisada>
            <mnozstvi zpusob_mereni="lžička">3</mnozstvi>
            <druh>kakao</dlruh>
    </ingredience>
    <postup>Mléko ohřejeme a vmícháme kakao. Lze podávat i studené</postup>
    </recept>
</kucharka>
```

Kapitola 5

Validace XML

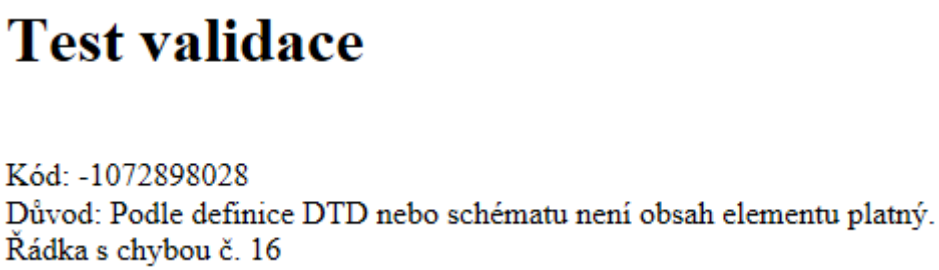
Již několikrát jsem zde zmínila „volnost“ při psaní XML. V praxi je často žádané a potřebné vytvářet několik obdobných XML dokumentů, na které se aplikuje CSS či XSL. Tím pádem bychom dokumentům rádi zajistili stejnou či v jistých mezích podobnou strukturu. Mohli bychom se o to postarat „ručně“, ale to by bylo příliš namáhavé a zdlouhavé, nehledě na problémy, které by vznikaly při vytváření podobných dokumentů více lidmi. Zde by narůstala nemožnost vytvoření obdobných dokumentů. Abychom měli striktně daná pravidla jaké elementy a atributy můžeme v dokumentu používat, využijeme schémat. Těmito schématy si vlastně vytvoříme vlastní značkovací jazyk, který má syntaxi XML a používá jen určené značky. Těchto jazyků již existuje mnoho. Nejznámějším jazykem je XHTML.

Schémat existuje několik, jmenovitě DTD (*Document Type Definition*), W3C (*World Wide Web Consortium*) XML Schéma, RELAX NG (*Regular Language for XML Next Generation*) a Schematron. Pro potřeby využití více schémat v jednom dokumentu využijeme NVDL (*Namespace Validation and Dispatching Language*) schématu. DTD má výhodu v široké podpoře aplikací. Vlastní však také řadu nevýhod, jako je nemožnost určení datového typu pro obsah elementů a atributů a minimální podpora jmenných prostorů. W3C XML schéma je po DTD nejpoužívanějším jazykem s podporou velkých firem a spoustou open-source projektů. W3C XML schémata mají poněkud složitou specifikaci a můžeme narazit na různé úskalí či nejasnosti v jazyku. Pokud bychom chtěli elegantní schéma, pak je RELAX NG ten pravý. Je velmi jednoduchý na naučení i na použití a jeho popularita roste. Tyto jazyky (RELAX NG, W3C XML schéma, DTD) vytváří gramatiky pro nové jazyky. Pokud chceme jazyk, který nám postihne zcela jinou oblast, měli bychom zvolit Schematron. Schematron nám umožňuje vytvořit pravidla, která musí být pro daný dokument splněna.[7]

Pro validaci XML oproti DTD či jiným schématům je potřeba parser. Webové prohlížeče většinou neumí validovat dokument oproti DTD. U prohlížeče Internet Explorer to však možné je. Vytvoříme si HTML stránku, která volá parser Exploreru. Ten nám provede validaci kódu v daném XML. Funkcí „validateOnParse“ zapínáme či vypínáme validaci. Je zde také přidán kód pro zobrazení části kódu, důvodu chyby a čísla řádky s výskytem chyby, jako je vidět na obrázku níže 5.1 – „Zobrazení validace chybného XML“

```
validaceIE.html
<!DOCTYPE html SYSTEM ▶
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
```

```
    <title>Validace</title>
    <meta http-equiv="content-type" content="text/html;charset=utf-8"/>
   </head>
   <body>
    <h1>Test validace</h1>
    <script type="text/javascript">
       var xmlDoc = new ActiveXObject("Microsoft.XMLDOM");
       xmlDoc.async="false";
       xmlDoc.validateOnParse="true";
       xmlDoc.load("kucharka_chybna.xml");
       document.write("<br />Kód: ");
       document.write(xmlDoc.parseError.errorCode);
       document.write("<br />Důvod: ");
       document.write(xmlDoc.parseError.reason);
       document.write("<br />Řádka s chybou č. ");
       document.write(xmlDoc.parseError.line);
     </script>
   </body>
</html>
```



Obrázek 5.1. Zobrazení validace chybného XML

Kapitola 6

Načítání externích entit

Entita je jednotka informace, která má přiřazené jméno, aby mohla být identifikována. Používá se v případech, kdy je pro nás výhodné složit jeden dokument z více dokumentů (práce více lidí na jednom dokumentu, lepší přehlednost, snadnější editace) nebo když potřebujeme používat určitou informaci na několika místech najednou či pokud informace odpovídají jinému datovému formátu než je XML. Entita je deklarována v DTD pomocí tagu <!ENTITY>. V kódu pak odkazujeme na tuto entitu pomocí symbolu „&;“ (&jménoEntity;). Na entitu se můžeme odkazovat v dokumentu několikrát a dokonce můžeme vytvořit hierarchii entit. Avšak není dovoleno, aby entita odkazovala sama na sebe. Existují tři druhy externích entit: obecná, parametrická a binární. Obecná entita slouží pro použití v XML, na parametrické entity se dá odkazovat pouze v DTD a binární entita je využívána pro vkládání jiných než textových entit. Binární entita se liší od textové deklarace tím, že za umístění souboru vepíšeme klíčové slovo NDATA a jméno formátu. Reference na binární entitu má podobu prázdného elementu s atributem, který je shodný s názvem entity. Pokud tedy chceme použít binární entitu, musíme nejen definovat entitu, ale i příslušný prázdný element.[27]

Následuje kód na testování. V DTD jsem si nadefinovala entitu čaj: „<!ENTITY caj SYSTEM 'caj.xml' >“. Entita čaj vloží do kódu místo zástupné entity „&caj;“ tento kód:

```
caj.xml
<?xml version="1.0" encoding="UTF-8"?>
<recept id="caj" >
 <nazev>Čaj</nazev>
 <ingredience>
  <prisada>
   <mnozstvi zpusob_mereni="hrnek">1</mnozstvi>
   <druh>voda</druh>
  </prisada>
  <prisada>
   <mnozstvi zpusob_mereni="lzicka">3</mnozstvi>
   <druh>med</druh>
  </prisada>
  <prisada>
   <mnozstvi zpusob_mereni="pytlik">1</mnozstvi>
   <druh>čaj podle chuti</druh>
  </prisada>
```

```
 </ingredience>
 <postup>Vodu uvaříme, vložíme čaj a necháme 5minut louhovat. Sáček vyjmeme ▶
a přidáme med. Zamícháme, podáváme.</postup>
</recept>
```

Budeme vkládat do následujícího dokumentu:

kucharka_entity.xml

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE kucharka [
<!ELEMENT  kucharka (recept+,obrazek?)>
<!ELEMENT  recept (nazev,ingredience,postup)>
<!ELEMENT nazev (#PCDATA) >
<!ELEMENT  ingredience (prisada+)>
<!ELEMENT prisada (mnozstvi,druh) >
<!ELEMENT mnozstvi (#PCDATA) >
<!ELEMENT druh (#PCDATA) >
<!ELEMENT postup (#PCDATA) >
<!ENTITY caj SYSTEM 'caj.xml' >
<!ATTLIST recept id ID #REQUIRED>
<!ATTLIST mnozstvi zpusob_mereni CDATA #REQUIRED >
<!ATTLIST kucharka xmlns CDATA #FIXED ▶
'http://mansfeldova.com/namespaces/cooking/1.0' >]>
<kucharka xmlns="http://mansfeldova.com/namespaces/cooking/1.0">
  <recept id="t0001">
    <nazev>kakao</nazev>
    <ingredience>
      <prisada>
        <mnozstvi zpusob_mereni="hrnek">1</mnozstvi>
        <druh>mléko</druh>
      </prisada>
      <prisada>
        <mnozstvi zpusob_mereni="lzicka">3</mnozstvi>
        <druh>kakao</druh>
      </prisada>
    </ingredience>
   <postup>Mléko ohřejeme a vmícháme kakao. Lze podávat i studené</postup>
  </recept>
  <recept id="t0002">
    <nazev>káva s mlékem</nazev>
    <ingredience>
      <prisada>
        <mnozstvi zpusob_mereni="hrnek">0,8</mnozstvi>
        <druh>voda</druh>
      </prisada>
      <prisada>
```

```
        <mnozstvi zpusob_mereni="hrnek">0,2</mnozstvi>
        <druh>mléko</druh>
      </prisada>
      <prisada>
        <mnozstvi zpusob_mereni="lzicka">2</mnozstvi>
        <druh>káva</druh>
      </prisada>
    </ingredience>
    <postup>Vodu uvaříme. Do hrnku dáme kávu, zalijeme vodou a dolejeme ▶
mlékem.</postup>
  </recept>&caj;
</kucharka>
```

Pokud by daný prohlížeč podporoval externí entity, pak by výsledek vypadal takto:

```
<kucharka>
  <recept id="t0001">
    <nazev>kakao</nazev>
    <ingredience>
      <prisada>
        <mnozstvi zpusob_mereni="hrnek">1</mnozstvi>
        <druh>mléko</druh>
      </prisada>
      <prisada>
        <mnozstvi zpusob_mereni="lzicka">3</mnozstvi>
        <druh>kakao</druh>
      </prisada>
    </ingredience>
    <postup>Mléko ohřejeme a vmícháme kakao. Lze podávat i studené</postup>
  </recept>
  <recept id="t0002">
    <nazev>káva s mlékem</nazev>
    <ingredience>
      <prisada>
        <mnozstvi zpusob_mereni="hrnek">0,8</mnozstvi>
        <druh>voda</druh>
      </prisada>
      <prisada>
        <mnozstvi zpusob_mereni="hrnek">0,2</mnozstvi>
        <druh>mléko</druh>
      </prisada>
      <prisada>
        <mnozstvi zpusob_mereni="lzicka">2</mnozstvi>
        <druh>káva</druh>
      </prisada>
```

```
    </ingredience>
    <postup>Vodu uvaříme. Do hrnku dáme kávu, zalijeme vodou a dolejeme ▶
mlékem.</postup>
  </recept>
  <recept id="caj" >
    <nazev>Čaj</nazev>
    <ingredience>
      <prisada>
        <mnozstvi zpusob_mereni="hrnek">1</mnozstvi>
        <druh>voda</druh>
      </prisada>
      <prisada>
        <mnozstvi zpusob_mereni="lzicka">3</mnozstvi>
        <druh>med</druh>
      </prisada>
      <prisada>
        <mnozstvi zpusob_mereni="pytlik">1</mnozstvi>
        <druh>čaj podle chuti</druh>
      </prisada>
    </ingredience>
    <postup>Vodu uvaříme vložíme čaj a necháme 5minut louhovat. Sáček ▶
vyjmeme a přidáme med. Zamícháme, podáváme.</postup>
  </recept>
</kucharka>
```

Kapitola 7

XML Base

XML Base definuje proces vyhodnocení relativních URI (*Uniform Resource Identifier*) odkazů spolu se syntaxí explicitní kontroly báze URI elementů v dokumentu. Běžně je výchozím stavem, když se v dokumentu nalezené relativní odkazy berou relativně vzhledem k bázi URI původní entity. Tento stav však není často žádoucím a právě toto řeší atribut xml:base, kterým lze nadefinovat výchozí bázi URI libovolnému elementu a jeho potomkům v dokumentu. Je-li hodnota atributu xml:base relativní URI, bude se vyhodnocovat vzhledem k aktuální bázi URI a to buď k výslovně zadané v atributu xml:base u předka či se dědí z celé entity. Pokud je možné pracovat se jmennými prostory, pak je prefix xml automaticky svázán se jmenným prostorem http://www.w3.org/XML/1998/namespace.[23]

```
base.xhtml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
    xml:base="http://mansfeldova.com/">
    <head>
        <title>Testování xml:base</title>
    </head>
    <body>
        <a href="kucharka.xml">Kuchařka</a>
        <a xml:base="xml/MIME/" href="kuchMime.xxx">další kuchařka</a>
    </body>
</html>
```

Výsledkem tohoto kódu bude stránka pouze se dvěma odkazy, kde jeden odkazuje na „http://mansfeldova.com/kucharka.xml“ a druhý na „http://mansfeldova.com/xml/MIME/kuchMime.xxx“.

Uplatnění najde XML Base také v SVG (*Scalable Vector Graphics*) - o kterém budu hovořit ve 13. kapitole. Konkrétněji u elementu „xlink“. XLink (*XML Linking Language*) umožnuje vložit element do XML, aby vytvořil a popsal vztahy mezi zdroji. Můžeme ovšem vytvářet i hypertextové odkazy.[4] V kódu, který bude následovat, se budu třikrát odkazovat na jeden obrázek. Pokud je XML Base podporovaný v daném prohlížeči, zobrazí se dva růžové čtverečky a jeden obrázek oznamující neexistující odkaz na obrázek, jak je vidět na následujícím obrázku

7.1 – „Správné zobrazení stránky s podporou XML Base v SVG“. V případě nepodpory XML Base se zobrazí jen jeden růžový čtvereček a dva zastupující znaky.

```
base-svg.xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.0//EN" ▶
"http://www.w3.org/TR/2001/REC-SVG-20010904/DTD/svg10.dtd">
<svg xmlns="http://www.w3.org/2000/svg" ▶
xmlns:xlink="http://www.w3.org/1999/xlink"
   viewBox="00 100 100">
  <image xlink:href="lzicka.jpg" width="100" height="10"/>
  <g xml:base="http://mansfeldova.com/">
   <image xlink:href="lzicka.jpg" width="50" height="500"/>
    <g xml:base="xml/">
     <image xlink:href="lzicka.jpg" width="1000" height="100"/>
    </g>
  </g>
</svg>
```

Obrázek 7.1. Správné zobrazení stránky s podporou XML Base v SVG

Kapitola 8

Manipulace s XML prostřednictvím DOM

Standard DOM (Document Object Model) je projekcí XML infosetu. Tento model reprezentuje infoset jako strom uzlů. Procesor generuje při zpracování ve vnitřní paměti strom, který odpovídá strukturou zpracovávanému dokumentu. S takto vytvořeným modelem můžeme dále pracovat a tak zjišťovat případně i měnit obsah daného XML dokumentu.

Toto rozhraní je ze všech rozhraní pro práci s XML nejflexibilnější, avšak je to vykoupeno jeho náročností při zpracování větších dokumentů. Na vrcholu DOM struktury je DOMNode. Metody a vlastnosti třídy DOMNode jsou přístupné ve všech uzlech. V určitých uzlech můžeme využít metody či vlastnosti navíc. S rozhraním DOM se pracuje mimo jiné i v JavaScriptu. [24]

Testovací kód s pomocí XMLHttpRequest, který přiblížím později (v 10. kapitole):

```
dom-xhr.html
<!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 4.01//EN'>
<!DOCTYPE html SYSTEM ▶
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Testování manipulace s daty prostřednictvím DOM</title>
   <meta http-equiv="content-type" content="text/html;charset=utf-8"/>
   <script type="text/javascript">
    function nactiXML(cesta){
      if (window.XMLHttpRequest) {
        xhttp=new XMLHttpRequest();
         }
      else {
        xhttp=new ActiveXObject("Microsoft.XMLHTTP");
        }
    xhttp.open("GET",cesta,false);
    xhttp.send();
    return xhttp.responseXML;}
   nazev="kucharka.xml"
   xml=nactiXML(nazev);
    if(xml.nodeType==9){
     document.write("Dokument byl načten správně<br/>");
     nazev=xml.getElementsByTagName("nazev")[0].childNodes[0].nodeValue;
```

```
      document.write("Název receptu: "+nazev+"<br/>");
      klonovani=xml.getElementsByTagName("prisada")[0].cloneNode(true);
      xml.getElementsByTagName("prisada")[1].appendChild(klonovani);
      prisady=xml.getElementsByTagName("druh");
     document.write("První přísada: "+prisady[0].childNodes[0].nodeValue ▶
+ "<br/>");
     document.write("Druhá přísada: "+prisady[1].childNodes[0].nodeValue ▶
+ "<br/>");
      document.write("Naklonovaná první přísada: ▶
"+prisady[2].childNodes[0].nodeValue + "<br/>");
      }
     else{
     document.write("Dokument " + cesta + " nebyl úspěšně načten.<br/>");
      }
  </script>
 </head>
 <body></body>
 </html>
```

Dalším kódem převádím textový řetězec na XML a následně v něm testuji vše jako je tomu u načítání XML ze souboru.

```
DOM-text.html
<!DOCTYPE html SYSTEM ▶
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title></title>
    <meta http-equiv="content-type" content="text/html;charset=utf-8"/>
    <script type="text/javascript">
    function loadXMLString(txt){
     if (window.DOMParser){
       parser=new DOMParser();
       xmlDoc=parser.parseFromString(txt,"text/xml");
      }
     else {
      xmlDoc=new ActiveXObject("Microsoft.XMLDOM");
      xmlDoc.async=false;
      xmlDoc.loadXML(txt);
     }
     return xmlDoc;
    }
    </script>
</head>
<body>
  <script type="text/javascript">
```

```
text="<kucharka><recept><nazev>kakao</nazev><ingredience><prisada><mnozstvi>1</mnozstvi>
<druh>mléko</druh></prisada><prisada><mnozstvi>3</mnozstvi><druh>kakao</druh>
</prisada></ingredience><postup>Mléko ohřejeme a vmícháme kakao. Lze ▶
podávat i studené</postup>
</recept><hodnoceni>5</hodnoceni></kucharka>";
    xml=loadXMLString(text);
    if(xml.nodeType==9){
     document.write("Dokument byl načten správně<br/>");
     nazev=xml.getElementsByTagName("nazev")[0].childNodes[0].nodeValue;
     document.write("Název receptu: "+nazev+"<br/>");
     klonovani=xml.getElementsByTagName("prisada")[0].cloneNode(true);
     xml.getElementsByTagName("prisada")[1].appendChild(klonovani);
     prisady=xml.getElementsByTagName("druh");
    document.write("První přísada: "+prisady[0].childNodes[0].nodeValue ▶
+ "<br/>");
    document.write("Druhá přísada: "+prisady[1].childNodes[0].nodeValue ▶
+ "<br/>");
     document.write("Naklonovaná první přísada: ▶
"+prisady[2].childNodes[0].nodeValue + "<br/>");
    }
    else{
     document.write("Dokument " + cesta + " nebyl úspěšně načten.<br/>");
    }
    </script>
 </body>
</html>
```

Výsledky by měly vypadat takto: 8.1 – „Správné zobrazení stránky s podporou DOM“. Nejprve jsem testovala, zda byl dokument správně načten tím, že jsem otestovala jeho první element, kde má být hodnota „nodeType“ rovna „9“. Tato hodnota reprezentuje začátek dokumentu. NodeType vrací číselnou hodnotu, které jsou přidělené jednotlivým typům uzlu. Dále vypisuji hodnoty z kuchařky a na závěr testuji klonování elementu. Přidala jsem navíc přísadu obsahující „mléko“.

Dokument byl načten správně
Název receptu: kakao
První přísada: mléko
Druhá přísada: kakao
Naklonovaná první přísada: mléko

Obrázek 8.1. Správné zobrazení stránky s podporou DOM

Kapitola 9

XForms

XForms jsou formuláře založené na XML. XForms mohou kontrolovat zadávaná data (porovnávání dat a kontrola datového typu) a nepovolí odeslání formuláře, pokud nejsou vyplněné povinné položky. Což je v praxi velice užitečné. Uživatel má okamžitou odezvu, zda je všechno v pořádku. Zadávaná data umí XForms odesílat jako XML data. Ve formulářích XForms můžeme kombinovat již existující technologie jako například XPath a XML schémata. Tím je pro lidi, kteří již tyto technologie znají, snažší se je naučit. Dalšími výhodami je nezávislost na platformě, jednoduché vytvoření komplikovanějších formulářů a široká přístupnost (například i pro slepecké čtečky).[10]

Dle mého názoru jsou XForms šikovným nástrojem, bohužel se však opět setkávám v současné době s nepodporou XForms ze strany prohlížečů. Dokonce bych řekla, že pokud prohlížeče podporovaly XForms, tak v současné době podporu zastavily nebo se jí úplně zbavily. Uvedu konkrétní příklad. Prohlížeč Mozilla Firefox podporoval XForms do verze 3.6 (v roce 2012 je aktuální verze 11.0) a to jen s pomocí rozšíření Mozilla XForms. I této době můžeme využívat XForms díky XSLTForms. Jak název napovídá, formulář se vytvoří pomocí XSLT stylů. Stáhneme si příslušné styly, umístíme je na server a můžeme je využívat. Styly a ukázkové příklady jsou volně přístupné na této adrese: http://www.agencexml.com/xsltforms.htm.[9]

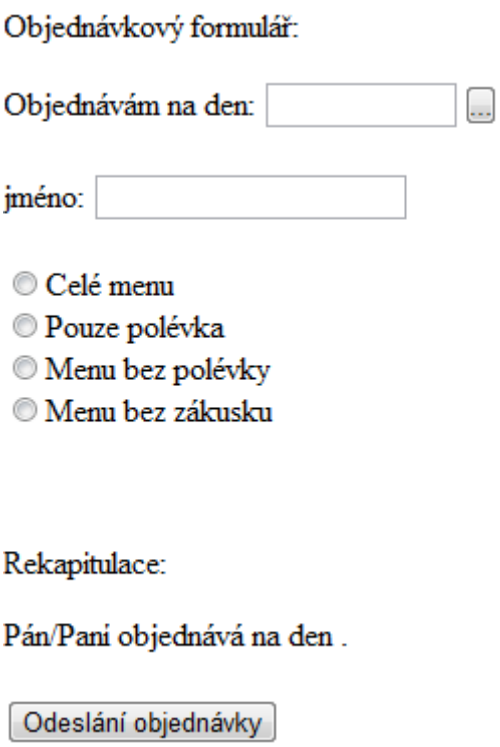
Rozhodla jsem se testovat XSLTForms. V mém případě jsem vytvořila jednoduchý formulář pro objednávku jídla na určitý den.

```
form.xml
<?xml-stylesheet href="xsltforms/xsltforms.xsl" type="text/xsl"?>
<html xmlns="http://www.w3.org/1999/xhtml" ▶
xmlns:xf="http://www.w3.org/2002/xforms"
    xmlns:ev="http://www.w3.org/2001/xml-events"
    xmlns:xs="http://www.w3.org/2001/XMLSchema">
<head>
 <title>Testování XSLTForms</title>
 <xf:model>
  <xf:instance>
   <instance  xmlns="">
    <datum/>
    <NaJmeno/>
    <jidla type="xs:string" />
     <seznamJidel>
```

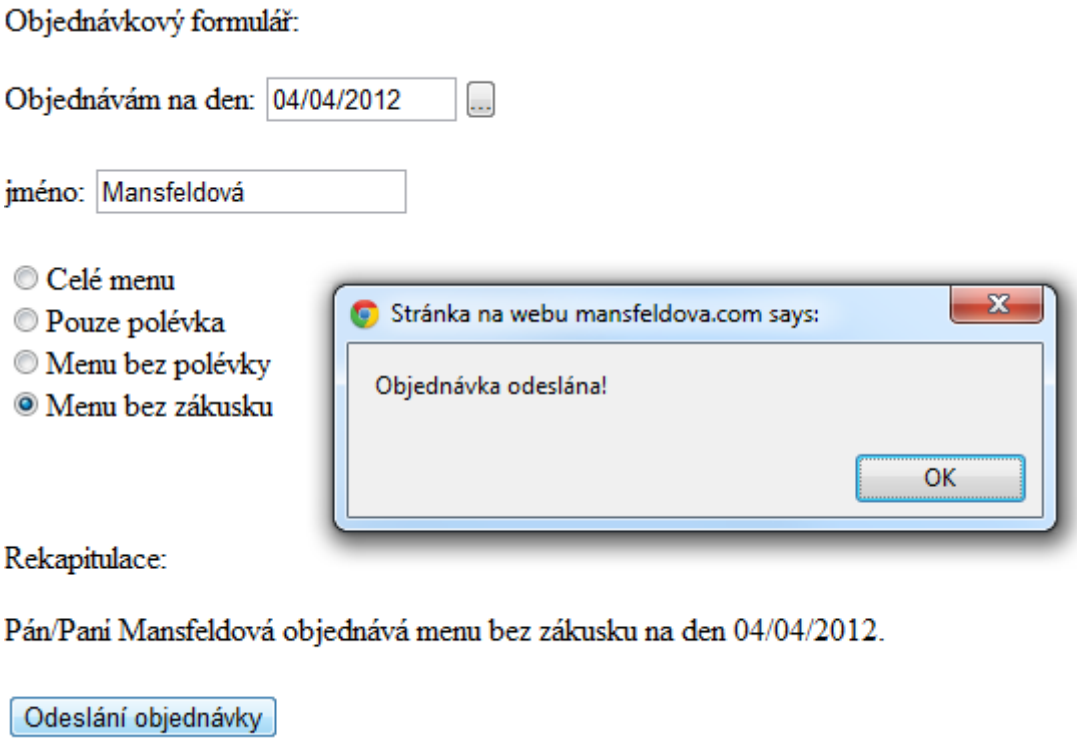
```
      <item>
       <label>Celé menu</label>
       <value>celé menu </value>
      </item>
     <item>
      <label>Pouze polévka</label>
      <value>pouze polévku</value>
     </item>
     <item>
      <label>Menu bez polévky</label>
      <value>menu bez polévky</value>
     </item>
     <item>
      <label>Menu bez zákusku</label>
      <value>menu bez zákusku</value>
     </item>
    </seznamJidel>
   </instance>
  </xf:instance>
  <xf:bind nodeset="datum" type="xs:date" />
 </xf:model>
</head>
<body>
 <p>Objednávkový formulář:  </p>
  <xf:input ref="datum">
   <xf:label>Objednávám na den: </xf:label>
  </xf:input>
  <br />
  <xf:input ref="NaJmeno" incremental="true">
   <xf:label>jméno: </xf:label>
  </xf:input>
  <br />
  <xf:select1 ref="jidla" appearance="full" incremental="true">
   <xf:itemset nodeset="../seznamJidel/item">
    <xf:label ref="label" />
    <xf:value ref="value" />
   </xf:itemset>
  </xf:select1>
  <br/>
  <p>Rekapitulace:</p>
  <xf:output value="concat('Pán/Paní ', NaJmeno, ' objednává ')"/>
  <xf:output ref="jidla"/>
  na den <xf:output ref="datum"/>.<br/>
  <xf:trigger>
   <xf:label>Odeslání objednávky</xf:label>
    <xf:message level="modal" ev:event="DOMActivate">Objednávka ▶
```

```
odeslána!</xf:message>
   </xf:trigger>
 </body>
</html>
```

Před vkládáním dat vypadá stránka s formulářem následovně: 9.1 – „Zobrazení XSLTForms před zadáním dat“. Po vyplnění polí a odeslání objednávky by mohla stránka vypadat třeba takto: 9.2 – „Zobrazení XSLTForms po zadání dat“.



Obrázek 9.1. Zobrazení XSLTForms před zadáním dat



Obrázek 9.2. Zobrazení XSLTForms po zadání dat

Kapitola 10

XMLHttpRequest

XMLHttpRequest API umožňuje zasílání dat mezi klientem a serverem. Samotné části jména XMLHttpRequest jsou mírně zavádějící. Přesněji tedy objekt XMLHttpRequest podporuje všechny textové formáty včetně XML. Požadavky lze zasílat nejen pomocí HTTP a HTTPS, ale některé implementace podporují i jiné protokoly. A nakonec poslední část názvu představuje veškerou aktivitu spojenou s HTTP požadavky.[11]

Tento objekt je hojně využíván AJAXem (*Asynchronous JavaScript and XML*). AJAX umožňuje aktualizovat pouze část stránky a není zde nutnost vykreslovat celou stránku znova. XMLHttpRequest vytvořila společnost Microsoft a byl implementován jako ActiveX komponenta. V této době je již standardizován sdružením W3C. Při použití musíme mít na paměti, že pokud chceme zachovat zpětnou kompatibilitu, musíme použít buď nativní implementaci nebo ActiveX komponentu.[24]

Zvolila jsem si jednoduchý test, který nám po stisku tlačítka zobrazí text vytažený z „kucharka.xml“.

request.html

```
<!DOCTYPE html SYSTEM ▶
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="content-type" content="text/html;charset=utf-8"/>
  <title>Test na XMLHttpRequest</title>
  <script type="text/javascript">
  function ukazat(){
   if (window.XMLHttpRequest){
    xmlHttp=new XMLHttpRequest();
   }
   else  {
    xmlHttp=new ActiveXObject("Microsoft.XMLHTTP");
   }
    xmlHttp.open("GET","kucharka.xml",false);
    xmlHttp.send();
   document.getElementById("vysledek").innerHTML=xmlHttp.responseText; ▶

     }
```

```
  </script>
 </head>
 <body>
  <p>Po kliknutí na tlačítko se zobrazí XML.</p>
  <input type="submit" value="Zobrazit XML" onclick="ukazat();"/>
  <span id="vysledek"></span>
 </body>
</html>
```

Nejprve se nám zobrazí HTML stránka s textem „Po kliknutí na tlačítko se zobrazí XML.“ a tlačítkem. Po stisku tlačítka se zobrazí výpis z XML.

Kapitola 11

xml:id

Xml:id nám umožňuje vložit jedinečný identifikátor jakémukoliv prvku v dokumentu. Každý prvek může mít maximálně jeden tento identifikátor. Pokud by se xml:id vyskytovalo v jednom dokumentu vícekrát stejné, tak by mělo dojít k xml:id chybě. Tato chyba sice není fatální, ale v rámci interoperability je důležité ji neignorovat.[19] Hodnota atributu xml:id je omezená, začínat musí podtržítkem či písmenem a poté mohou následovat pomlčky, podtržítka, tečky, číslice či písmena. Xml:id se využívá za účelem identifikace pro různé dotazovací jazyky (XPath), rozhraní (DOM) nebo se na ně můžeme odvolávat v XInclude.[24]

XML dokument:

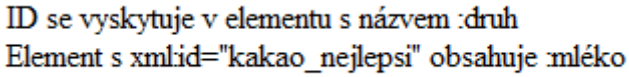
```
kucharkaID.xml
<kucharka xmlns="http://mansfeldova.com/namespaces/cooking/1.0">
  <recept >
    <nazev>kakao</nazev>
    <ingredience>
      <prisada>
        <mnozstvi zpusob_mereni="hrnek">1</mnozstvi>
        <druh xml:id="kakao_nejlepsi">mléko</druh>
      </prisada>
      <prisada>
        <mnozstvi zpusob_mereni="lžička">3</mnozstvi>
        <druh>kakao</druh>
      </prisada>
    </ingredience>
   <postup>Mléko ohřejeme a vmícháme kakao. Lze podávat i studené</postup>
  </recept>
</kucharka>
```

Dotazování pomocí DOM

```
getID.html
<!DOCTYPE html SYSTEM ▶
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
```

```
<head>
 <title>Testování podpory xml:id</title>
  <meta http-equiv="content-type" content="text/html;charset=utf-8"/>
  <script type="text/javascript">
   function nactiXML(cesta){
     if (window.XMLHttpRequest) {
       xhttp=new XMLHttpRequest();
        }
     else {
       xhttp=new ActiveXObject("Microsoft.XMLHTTP");
       }
    xhttp.open("GET",cesta,false);
    xhttp.send();
    return xhttp.responseXML;}
   nazev="kucharkaID.xml"
   xml=nactiXML(nazev);
     ID=xml.getElementById("kakao_nejlepsi");
     document.write("ID se vyskytuje v elementu s názvem: " + ▶
ID.nodeName+"<br/>");
    document.write("Element s xml:id=&quot;kakao_nejlepsi&quot; obsahuje ▶
:"+ID.childNodes[0].nodeValue);
  </script>
 </head>
 <body></body>
 </html>
```

Při podpoře xml:id se má zobrazit text jako je vidět na obrázku 11.1 – „Ukázka podpory xml:id“, kde jsou důležitá slova druh a mléko, která jsou vypisována funkcí.



Obrázek 11.1. Ukázka podpory xml:id

Kapitola 12

XInclude

XInclude umožňuje sloučení několika dokumentů. Používá standardní syntaxi XML a funguje na úrovni infosetu. Procesor, který rozpoznává XInclude, nahradí odkaz odkazovaným obsahem. XInclude se zpracovává až po parsování, tudíž pokud vkládaný dokument má jiný jmenný prostor než dokument do kterého vkládáme, bude vloženému textu přidělen původní jmenný prostor.

XInclude definuje jmenný prostor http://www.w3.org/2001/XInclude. V kódu se XInclude využívá jako element include, který má atributy href, parse, xpointer, encoding, accept a accept-language. S tímto elementem je spojen element fallback. Element fallback se využívá v případech, kde soubor či odkaz v elementu include nelze nalézt. Fallback může být použit v jednom elementu include vícekrát.[2]

XInclude není v prohlížečích podporován, avšak si můžeme opět vypomoci. V tomto případě je to pomocí XSLT transformace. Pomocí vytvořené transformace by měl XInclude fungovat. XSLT transformaci jsem zakomponovala do již vytvořeného XSLT, použitého v kapitole 2.2. Transformace vypadá následovně:

```
styl-xi.xsl
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" ▶
version="1.0"
    xmlns:cook="http://mansfeldova.com/namespaces/cooking/1.0"
    xmlns:xi="http://www.w3.org/2001/XInclude">
    <xsl:output method="html" encoding="UTF-8"/>
    <xsl:template match="/">
 <html>
   <head>
      <title>
         <xsl:value-of select="cook:kucharka/cook:nazev"/>
      </title>
      <style type="text/css">
        body { color:black; }
        h1 { color:red;
             text-decoration:underline; }
        h2 { text-decoration:none; }
        ol { color:orange;}
      </style>
```

```
    </head>
    <body>
      <xsl:choose>
          <xsl:when test="//xi:include">
             <xsl:apply-templates  mode="xinclude"/>

          </xsl:when>
          <xsl:otherwise>
              <xsl:apply-templates/>
          </xsl:otherwise>
      </xsl:choose>
      <h1>
          <xsl:value-of select="cook:kucharka/cook:nazev"/>
      </h1>
      <xsl:apply-templates select="cook:kucharka/cook:recept"/>
  </body>
 </html>
</xsl:template>
<xsl:template match="cook:recept">
    <h2>
        <xsl:apply-templates select="cook:nazev"/>
    </h2>
    <ol>Ingredience <xsl:apply-templates select="cook:ingredience"/>
    </ol>
    <xsl:apply-templates select="cook:postup"/>
</xsl:template>
<xsl:template match="cook:postup">
    <h3>POSTUP:</h3>
    <p><xsl:value-of select="."/></p>
</xsl:template>
<xsl:template match="cook:ingredience">
    <xsl:for-each select="cook:prisada">
        <xsl:choose>
            <xsl:when test="cook:mnozstvi/@zpusob_mereni='hrnek'">
                <li><xsl:value-of select="cook:mnozstvi"/> hrnek ▶
<xsl:value-of select="cook:druh"/></li>
            </xsl:when>
            <xsl:when test="cook:mnozstvi/@zpusob_mereni='lzicka'">
                <li><xsl:value-of select="cook:mnozstvi"/>lžička ▶
<xsl:value-of select="cook:druh"/></li>
            </xsl:when>
            <xsl:otherwise/>
        </xsl:choose>
    </xsl:for-each>
</xsl:template>
```

```
<xsl:template match="node()" mode="xinclude">
    <xsl:copy
        ><xsl:call-template name="xinclude-copy-attributes"
        /><xsl:apply-templates select="node()" mode="xinclude"
        /></xsl:copy>
</xsl:template>
<xsl:template name="xinclude-copy-attributes">
    <xsl:for-each select="./@*">
        <xsl:copy-of select="."/>
    </xsl:for-each>
</xsl:template>
</xsl:stylesheet>
```

Kód na který se daná transformace použije:

```
xin1.xml
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<?xml-stylesheet type="text/xsl" href="styl-xi.xsl"?>
<kucharka xmlns="http://mansfeldova.com/namespaces/cooking/1.0" ▶
xmlns:xi="http://www.w3.org/2001/XInclude">
  <nazev>Test XInclude</nazev>
  <xi:include href="kucharka.xml" parse="xml">
   <xi:fallback><p>kucharka.xml neexistuje</p></xi:fallback>
  </xi:include>
</kucharka>
```

Výsledek by měl vypadat následovně:

Test XInclude

kakao

Ingredience

1. 1 hrnek mléko
2. 3lžička kakao

POSTUP:

Mléko ohřejeme a vmícháme kakao. Lze podávat i studené

5

Test XInclude

Obrázek 12.1. Výsledek XInclude

Kapitola 13

SVG

SVG (*Scalable Vector Graphics*) je jazyk pro popis dvourozměrné vektorové grafiky, statické i dynamické animace. Formát SVG je základním otevřeným formátem pro vektorovou grafiku na internetu. SVG může obsahovat i rastrovou grafiku. Při vektorovém vykreslování systém zachová kvalitu obrazu i při mnohonásobném zvětšení. SVG dokument lze složit z několika dokumentů.[25]

Pro různé vektorové mapky a obrázky je SVG mnohem úspornější než bitmapové formáty jako GIF (*Graphics Interchange Format*) nebo JPEG (*Joint Photographic Experts Group*). SVG umožňuje zobrazení základních tří typů grafických objektů, a to vektorové grafické tvary, obrázky a text. Každý element i atribut může být animován. Výhodou je, že obrázek ve formátu SVG lze vyhledávat, indexovat, scriptovat a komprimovat.

Kód pro testování:

```
svg.xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.0//EN" ▶
"http://www.w3.org/TR/2001/REC-SVG-20010904/DTD/svg10.dtd">
 <svg xmlns="http://www.w3.org/2000/svg" width="100%" height="100%">
  <circle cx="100" cy="50" r="40" stroke="black" stroke-width="2" ▶
fill="blue"/>
  <text x="0" y="30" fill="red" transform="rotate(30 20,40)">SVG</text>
  <defs>
   <linearGradient id="grad1" x1="0%" y1="0%" x2="100%" y2="0%">
    <stop offset="0%" style="stop-color:rgb(255,255,0);stop-opacity:1"/>
    <stop offset="100%" style="stop-color:rgb(255,0,0);stop-opacity:1"/>
   </linearGradient>
  </defs>
  <ellipse cx="250" cy="70" rx="85" ry="55" fill="url(#grad1)" />
  <text fill="#ffffff" font-size="45" font-family="Verdana" x="200" ▶
y="86">XML</text>
  <rect x="300" y="20" width="25" height="25" style="fill:blue">
   <animate attributeType="CSS" attributeName="opacity" from="1" to="0" ▶
dur="5s" repeatCount="indefinite" />
  </rect>
  </svg>
```

Dále budu testovat v HTML5

svg5.html

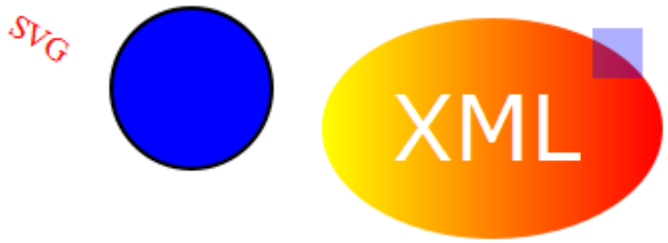
```
<!DOCTYPE html>
<html lang="cs-cz" dir="ltr">
  <head>
    <meta charset="UTF-8">
      <title>SVG v HTML5</title>
    </head>
    <body>
      <svg xmlns="http://www.w3.org/2000/svg" width="100%" height="100%">
        <circle cx="100" cy="50" r="40" stroke="black" stroke-width="2" ▶
fill="blue" />
        <text x="0" y="30" fill="red" transform="rotate(30 ▶
20,40)">SVG</text>
        (...)
        <animate attributeType="CSS" attributeName="opacity" from="1" ▶
to="0" dur="5s" repeatCount="indefinite" />
      </rect>
    </svg>
  </body>
</html>
```

Další test bude v rámci XHTML, kde použiji DTD, které je pro XHTML, SVG i MathML (MathML se bude týkat následující kapitola)

svg.xhtml

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1 plus MathML 2.0 plus SVG ▶
1.1//EN" "http://www.w3.org/2002/04/xhtml-math-svg/xhtml-math-svg.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
    <title>SVG v XHTML</title>
  </head>
  <body>
      <svg width="100%" height="100%">
        <circle cx="100" cy="50" r="40" stroke="black" stroke-width="2" ▶
fill="blue" />
        <text x="0" y="30" fill="red" transform="rotate(30 ▶
20,40)">SVG</text>(...)
        <animate attributeType="CSS" attributeName="opacity" from="1" ▶
to="0" dur="5s" repeatCount="indefinite" />
      </rect>
    </svg>
  </body>
</html> ▶
```

Při správném zobrazení se zobrazí diagonální text: „SVG“, modrý kruh vytažený černou barvou, elipsa vyplněná přechodem od žluté až po červenou a nápisem: „XML“ a vpravo nahoře na elipse malý modrý čtverec, který se během pěti sekund pomalu ztratí. Názorně na obrázku 13.1 – „Správné zobrazení SVG“



Obrázek 13.1. Správné zobrazení SVG

Kapitola 14

MathML

Jazyk MathML (Mathematical Markup Language) je využíván pro správné zobrazení matematických vzorců. Nezachycuje pouze strukturu, ale i obsah vzorců. Pro tyto účely využívá MathML různé značky. Prezentační MathML pro čitelnost člověkem a sémantické MathML pro porozumění programy. Cílem MathML je kódování matematických vzorců vhodných pro vědeckou komunikaci, a dále také usnadnění konverze do jiných formátů například i do Braillova písma.

V této době existuje verze MathML 3.0, která má lepší přístupnost pro jazyky píšící zprava doleva. Nejčastěji se MathML využívá ve webových prohlížečích a vzdělávacím softwaru. S MathML 3.0 je také úzce spojené CSS pro MathML.[5]

Kód na testování prohlížečů

```
math.xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE math PUBLIC "-//W3C//DTD MathML 3.0//EN" ▶
"http://www.w3.org/Math/DTD/mathml3/mathml3.dtd">
  <math xmlns="http://www.w3.org/1998/Math/MathML">
    <mrow>
      <mn>0</mn>
      <mo>=</mo>
      <mrow>
        <msup>
          <mrow>
            <mi>a</mi>
            <mo>&#8290;</mo>
            <mi>x</mi>
          </mrow>
          <mn>2</mn>
        </msup>
        <mo>+</mo>
        <mi>b</mi>
        <mo>&#8290;</mo>
        <mi>x</mi>
        <mo>+</mo>
        <mi>c</mi>
      </mrow>
```

```
      </mrow>
      <mspace linebreak="newline" />
      <mrow>
        <mi>x</mi>
        <mo>=</mo>
        <mfrac>
          <mrow>
            <mrow>
              <mo>-</mo>
              <mi>b</mi>
            </mrow>
            <mo>±</mo>
            <msqrt>
              <mrow>
                <mrow>
                  <msup>
                    <mi>b</mi>
                    <mn>2</mn>
                  </msup>
                  <mo>-</mo>
                </mrow>
                <mrow>
                  <mn>4</mn>
                  <mo>&#8290;</mo>
                  <mi>a</mi>
                  <mo>&#8290;</mo>
                  <mi>c</mi>
                </mrow>
              </mrow>
            </msqrt>
          </mrow>
          <mrow>
            <mn>2</mn>
            <mo>&#8290;</mo>
            <mi>a</mi>
          </mrow>
        </mfrac>
      </mrow>
  </math>
```

Tento kód budu testovat i v HTML5, pro zkrácení kódu (kód MathML bude stále stejný) uvedu „(...)“ pro již známý kód.

14.1 – „Vzorec MathML“

```
math5.html
<!DOCTYPE html>
```

```
<html lang="cs-cz" dir="ltr">
  <head>
    <meta charset="UTF-8">
      <title>MathML v HTML5</title>
    </head>
    <body>
      <math xmlns="http://www.w3.org/1998/Math/MathML">
        <mrow>
        (...)
        </mrow>
      </math>
    </body>
</html>
```

Ještě vyzkouším kód v XHTML, kde musím použít specifické DTD.

```
math.xhtml
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1 plus MathML 2.0//EN" ▶
"http://www.w3.org/Math/DTD/mathml2/xhtml-math11-f.dtd">
 <html>
  <head>
    <title>MathML v XHTML</title>
  </head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
  <body>
     <math>
      <mrow>
       (...)
      </mrow>
    </math>
  </body>
</html>
```

Kód MathML vždy zobrazuje kvadratickou rovnici a výpočet neznámé „x“ v kvadratické rovnici. V ideálním případě by měl výsledek kódů vypadat takto:

$$0 = ax^2 + bx + c$$

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Obrázek 14.1. Vzorec MathML

Kapitola 15

Jmenné prostory

Jmenné prostory nám zajišťují jednoznačnou identifikaci elementů v dokumentu. Již víme, že názvy elementů v XML dokumentech si můžeme vytvořit samy. A pokud XML dokumenty nevytváříme jen pro svoje potřeby, vznikají zde nejednoznačnosti. Například element značka může určovat název výrobce v internetovém obchodě a nebo název dopravní značky v dokumentech pro autoškolu. Pokud danou značku použijeme se jmenným prostorem, máme jistotu, že nedojde ke kolizi mezi názvy elementů. Jmenný prostor má podobu URI (*Uniform Resource Identifier*) adresy a ta zajišťuje možnost celosvětově vytvářet unikátní URI. Tato adresa nemusí ani existovat, je to pouze identifikátor a aplikace pracující s XML se nikdy z této adresy nesnaží získat dokument.[24]

Pro využití jmenných prostorů můžeme použít jednu ze dvou syntaxí. První možností je určení výchozího jmenného prostoru. Deklarujeme tedy jmenný prostor přidáním atributu xmlns, do kterého napíšeme patřičnou (jedinečnou) URI, element a elementy obsažené v něm pak budou patřit do tohoto jmenného prostoru. Druhou možností je určení prefixu, který zastupuje jmenný prostor (xmlns:prefix="URI"). Název prefixu je libovolný. Prefix se pak musí psát před názvem každého elementu (prefix:element), kterému chceme přiřadit daný jmenný prostor. Zápisu „prefix:element“ se říká kvalifikované jméno elementu a společně nám jednoznačně identifikují element.[24]

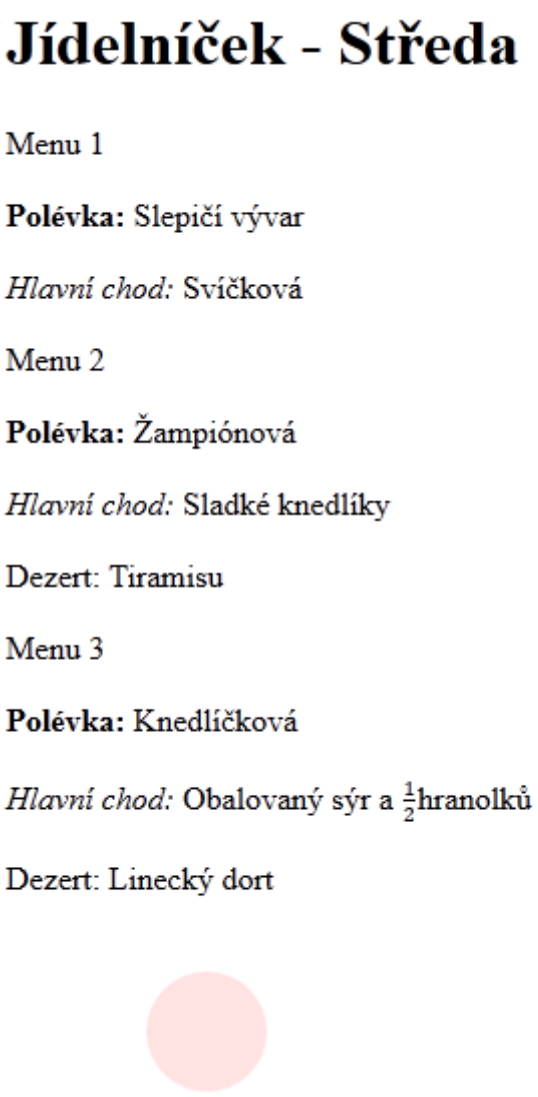
Následujícím kódem budu testovat jmenné prostory:

```
jmenne.xml
<?xml version="1.0" encoding="UTF-8"?>
<jidelnicek xmlns="http://mansfeldova.com/restaurace/jidelnicek/1.0" ▶
xmlns:html="http://www.w3.org/1999/xhtml">
  <html:html>
    <html:title>Jídelníček</html:title>
    <html:head></html:head>
    <html:body>
      <html:h1>Jídelníček - Středa</html:h1>
      <menu id="1">
        <html:p>
          <h1>Menu 1</h1>
        </html:p>
        <html:p>
          <html:b>Polévka:</html:b>
```

```
    <polevka>Slepičí vývar</polevka>
  </html:p>
  <html:p>
    <html:em>Hlavní chod:</html:em>
    <hlavni_jidlo>Svíčková</hlavni_jidlo>
  </html:p>
</menu>
<menu id="2">
  <html:p>
    <h2>Menu 2</h2>
  </html:p>
  <html:p>
    <html:b>Polévka:</html:b>
    <polevka>Žampiónová</polevka>
  </html:p>
  <html:p>
    <html:em>Hlavní chod:</html:em>
    <hlavni_jidlo>Sladké knedlíky</hlavni_jidlo>
  </html:p>
  <html:p>Dezert:
    <dezert>Tiramisu</dezert>
  </html:p>
</menu>
<menu id="3">
  <html:p>
    <h3>Menu 3</h3>
  </html:p>
  <html:p>
    <html:b>Polévka:</html:b>
    <polevka>Knedlíčková</polevka>
  </html:p>
  <html:p>
    <html:em>Hlavní chod:</html:em>
    <hlavni_jidlo>Obalovaný sýr a
      <math xmlns="http://www.w3.org/1998/Math/MathML">
        <mrow>
          <mfrac>
            <mrow>
              <mn>1</mn>
            </mrow>
            <mn>2</mn>
          </mfrac>
        </mrow>
      </math>hranolků</hlavni_jidlo>
  </html:p>
  <html:p>Dezert:
```

```
          <dezert>Linecký dort</dezert>
        </html:p>
      </menu>
      <html:p>
        <svg xmlns="http://www.w3.org/2000/svg" width="300px" ▶
height="100px" viewbox="0 0 300 100">
          <circle cx="100" cy="50" r="30" stroke-width="2" fill="red" >
          <animate attributeType="CSS" attributeName="opacity" from="1" ▶
to="0" dur="5s" repeatCount="indefinite" />
          </circle>
        </svg>
      </html:p>
    </html:body>
  </html:html>
</jidelnicek>
```

Testuji zde hned několikrát, zda prohlížeč umí pracovat se jmennými prostory. Využívám zde stylového přednastavení XHTML stránek, tím mám na mysli, že jsem si v XML zvolila elementy h1, h2, h3 jako označení menu. Pokud by prohlížeč nahlížel na celý dokument jako na dokument XML, pak by text v elementu „h1“ byl stejně velký jako element „html:h1“. Jmenný prostor HTML jsem nadefinovala prefixem html. Ale při podpoře jmenných prostorů bude velikost slova v elemementu h1 jiná, než slova v elementu html:h1. Dále jsem zkoušela vložit do jmenného prostoru XML (element hlavni jidlo) kód, který má pomocí MathML zobrazit „1/2“. A jako poslední jsem do prefixem deklarovaného jmenného prostoru XHTML vložila animovaný kruh díky jazyku SVG. Výsledná stránka by měla vypadat takto: 15.1 – „Správné zobrazení stránky s podporou jmenných prostorů“



Obrázek 15.1. Správné zobrazení stránky s podporou jmenných prostorů

Kapitola 16

XBL

XBL (XML Binding Language) je jazyk pro popis vazeb. Těmito vazbami se připojují elementy z jiných dokumentů. Je to mechanismus pro základ běžných prezentací a interaktivnost jednotlivých elementů, jejich připojením na jednotlivé definice takzvaným vázáním.

Navázáním na elementy může být připojené CSS, DOM či deklarací v XBL, kde se tyto elementy spojují se specifickými selektory, které jsou naimplementovány pomocí specifické vazby. Připojené elementy se nazývají vázané elementy. Vazba specifikuje jejich nové chování či jejich prezentaci. V žádném případě nemění sémantiku dokumentu ani jeho význam. XBL pracuje ve jmenném prostoru „http://www.w3.org/ns/xbl“.[16]

Abychom mohli využívat XBL v prohlížečích, musíme si opět přizvat na pomoc JavaScriptovou knihovnu. Knihovnu včetně příkladů je možné volně stáhnout na stránce http://code.google.com/p/xbl/. Stačí ji připojit k HTML, vytvořit XBL soubor a dále se fantazii meze nekladou. Propojení mezi XBL a daným HTML elementem zajišťuje CSS pomocí deklarace „binding“. V následujícím příkladu jsem vázala hlavně CSS styly a vzniklo tak interaktivní menu.[1]

```
xbl.html
<!DOCTYPE html SYSTEM ▶
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
  <head>
    <title>Test prohlížeče XBL</title>
   <meta  http-equiv="content-type" content="text/html,charset='utf-8'"/>
    <script src="http://mansfeldova.com/xml/xbl/xbl.js" ▶
type="text/javascript" ></script>
    <style type="text/css">
         .mouseover-mouseout {  binding: ▶
url('xblko.xml#mouseover-mouseout');  }
        .mouseenter-mouseleave { binding: ▶
url('xblko.xml#mouseenter-mouseleave'); position:relative;
        padding:10px;  border: 1px solid LightGray; margin:20px;}
        .click {binding: url('xblko.xml#click');}
        .dblclick { binding: url('xblko.xml#dblclick'); }
    </style>
 </head>
 <body>
```

```
    <h1>Středeční menu</h1>
    <div class="mouseenter-mouseleave" id="1">
     <div class="mouseenter-mouseleave" id="2">
       <div class="mouseenter-mouseleave" id="2.1">
         <div class="mouseover-mouseout" ><h2>Menu 1</h2></div>
          <p><b>Polévka: </b>Slepičí vývar</p>
          <p><em>Hlavní chod: </em>Svíčková</p>
       </div>
       <div class="mouseenter-mouseleave" id="2.2">
        <div class="click"><h2>Menu 2</h2></div>
         <p><b>Polévka: </b>Žampiónová</p>
         <p><em>Hlavní chod: </em>Sladké knedlíky</p>
         <p>Dezert: Tiramisu</p>
       </div>
       <div class="mouseenter-mouseleave" id="2.3">
        <div class="dblclick"><h2>Menu 3</h2></div>
         <p><b>Polévka: </b>Knedlíčková</p>
         <p><em>Hlavní chod: </em>Obalovaný sýr a hranolky</p>
          <p>Dezert: Linecký dort </p>
        </div>
    </div>
    </div>
 </body>
</html>
```

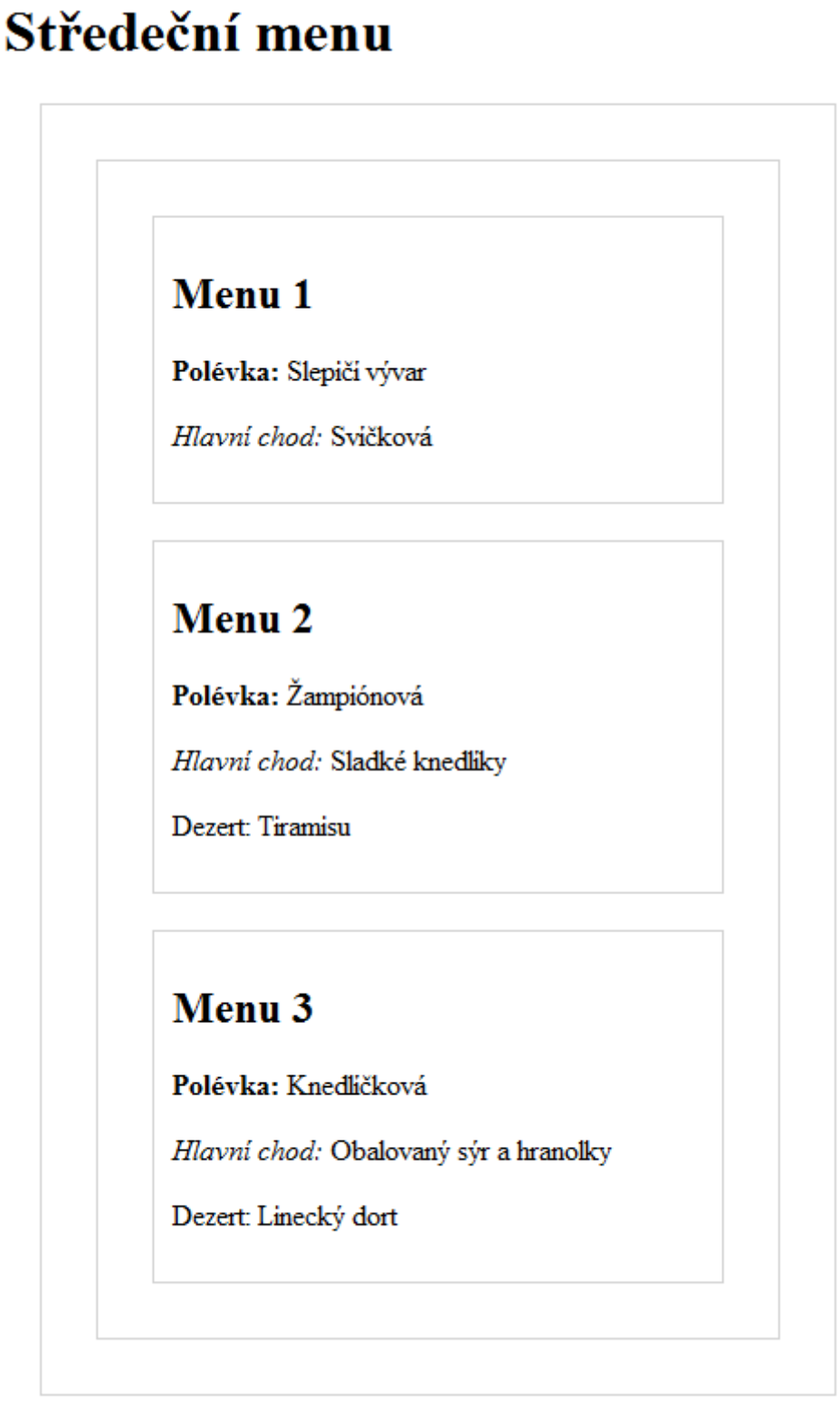
Následuje kód navázaného XBL.

xbl.xml

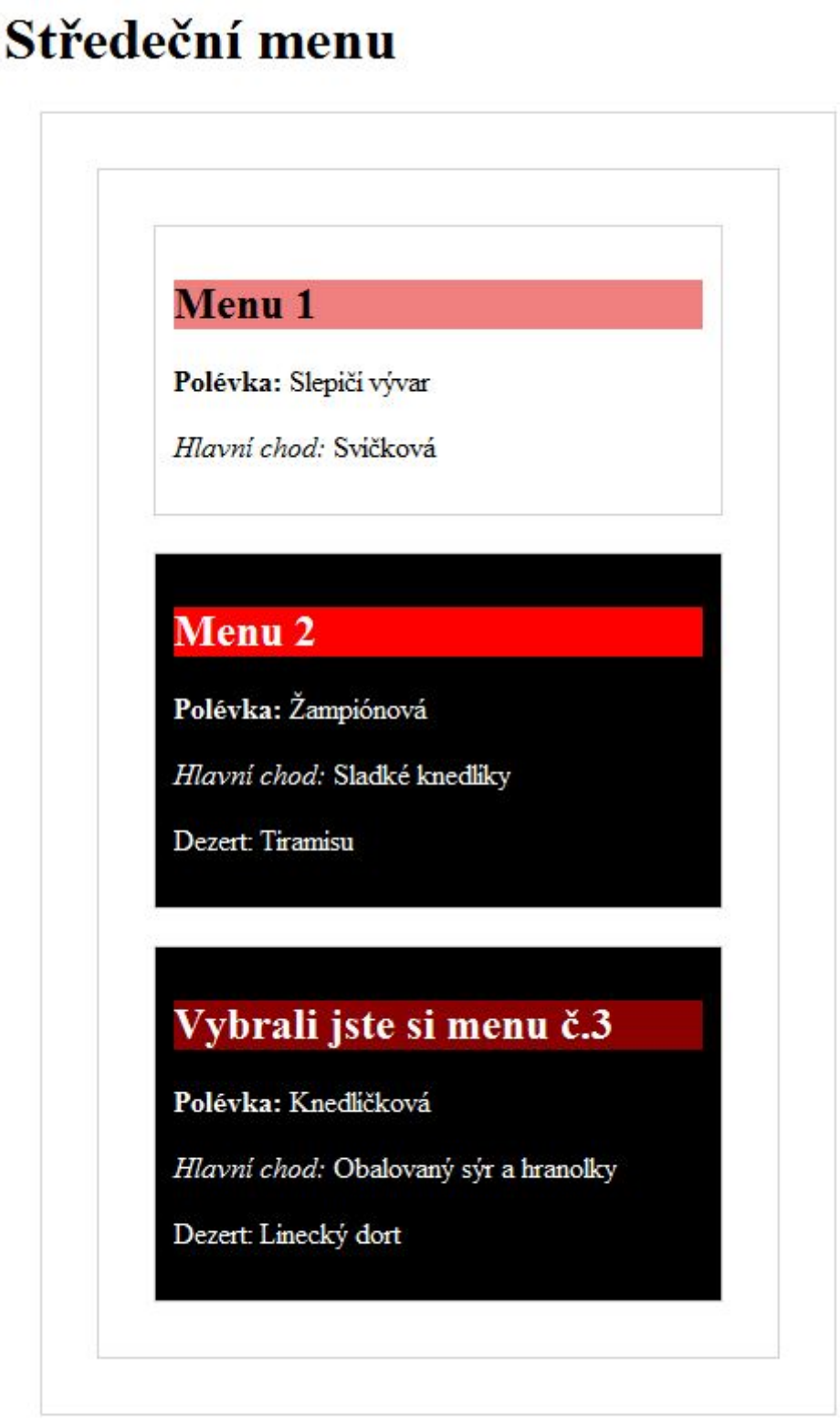
```
<xbl xmlns="http://www.w3.org/ns/xbl" xmlns:xbl="http://www.w3.org/ns/xbl">
<binding id="mouseover-mouseout">
    <handlers>
        <handler event="mouseover">
            this.boundElement.style.backgroundColor = 'lightcoral';
        </handler>
        <handler event="mouseout">this.boundElement.style.backgroundColor
 = '';</handler>
    </handlers>
</binding>
<binding id="mouseenter-mouseleave">
    <handlers>
        <handler ▶
event="mouseenter">this.boundElement.style.color='black';</handler>
       <handler event="mouseenter">this.boundElement.style.backgroundColor
 = 'white';</handler>
        <handler event="mouseleave">this.boundElement.style.color ▶
='white';</handler>
```

```
        <handler event="mouseleave">this.boundElement.style.backgroundColor
 = 'black';</handler>
    </handlers>
</binding>
<binding id="click">
    <handlers>
        <handler event="click">
            this.boundElement.style.backgroundColor = 'red';
        </handler>
    </handlers>
</binding>
<binding id="dblclick">
    <handlers>
        <handler event="dblclick">
            this.boundElement.style.backgroundColor = 'darkred';
          this.boundElement.innerHTML = "&lt;h2&gt;Vybrali jste si menu ▶
č. 3&lt;/h2&gt;";
        </handler>
    </handlers>
</binding>
</xbl>
```

Použila jsem akce, které reagují na akce myší. Každý ze tří nadpisů menu reaguje na jinou akci. První reaguje na přejetí, druhé na kliknutí a třetí na dvojklik. U prvních dvou se jen změní barva pozadí, třetí po dvojkliku zamění nadpis „Menu 3“ za „Vybrali jste si menu č. 3“. K tomu ještě jsem vytvořila k menu do sebe 3x zanořený obal menu, který mění barvu pozadí na černou, pokud jsme mimo daný obdélník. Pokud se vyskytujeme v jakémkoliv obdélníku, pak má on i obdélníky pod ním bílou barvu pozadí. Následuje obrázek znázorňující stránku po načtení 16.1 – „Načtení stránky s XBL“, dále pak obrázek demonstrující aktivní XBL. 16.2 – „Správné zobrazení stránky s podporou XBL“



Obrázek 16.1. Načtení stránky s XBL



Obrázek 16.2. Správné zobrazení stránky s podporou XBL

Kapitola 17

SMIL Timesheets

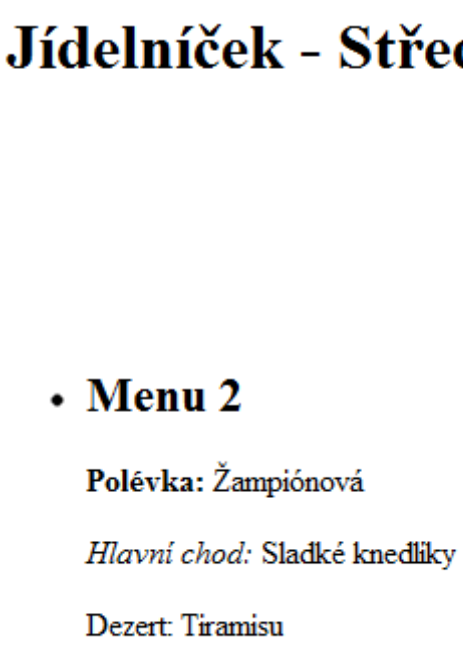
SMIL Timesheets (*Synchronized Multimedia Integration Language*) je XML jazyk pro časování nejen SMIL elementů a atributů, ale je možné ho použít na mnoho jiných XML. Samotný SMIL je jazyk pro integraci animací, videí, hudby či obrázků a dají se díky němu vytvořit multimediální prezentace. Když budeme využívat SMIL s pomocí CSS, tak nám tato dvojice poskytne časově multimediální prezentace, které se dají použít na více dokumentů zároveň. Časování elementu se nastavuje pomocí atributu „dur“ a pokud některý element z jeho dětí má atribut dur s delším trváním (nebo čas běhu elementu začal později, tudíž se stejným časem je trvání elementu delší než jeho rodič), pak je tento element zastaven předčasně. Časování je možné nastavit jako začátek běhu, konec běhu a dobu běhu. O tom, že si je SMIL Timesheets s CSS blízký nás přesvědčí i fakt, že pro identifikaci elementu v dokumentu používají stejný princip. Elementy v dokumentu se označují atributem class nebo atributem id v elementu div. Na takto označené elementy se poté odkazujeme ve SMIL Timesheets pomocí elementu item v atributu select, tak jak jsme zvyklí z CSS, kde pro id vkládáme před daný název „#“ a pro název z class „.“.[8][6]

Vzhledem k malé podpoře mezi prohlížeči si můžeme pomoci knihovnou „timesheets.js“ která nám umožňuje využívat SMIL podle W3C. S pomocí této knihovny jsem testovala SMIL Timesheets. Knihovna je volně přístupná ke stažení na stránce: http://wam.inrialpes.fr/timesheets/.[3] Následuje kód:

```
time.xhtml
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML+SMIL //EN" ▶
"http://www.w3.org/2001/SMIL20/WD/xhtmlplussmil.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" ▶
xmlns:smil="http://www.w3.org/2001/SMIL20">
  <head>
    <title></title>
    <script type="text/javascript" ▶
src="http://mansfeldova.com/xml/smi/timesheets.js"></script>
    <style type="text/css">
        ul li {
        position: relative;
        top: 0;
        opacity: 0;
        filter: alpha(opacity=0);
        -moz-opacity: 0;
```

```
        -khtml-opacity: 0;
        }
        li[smil=active] {
        opacity: 1;
        filter: alpha(opacity=100);
        -moz-opacity: 1;
        -khtml-opacity: 1;
        }
        </style>
  </head>
  <body>
      <h1>Jídelníček - Středa</h1>
      <ul  id="jidelnicek" smil:timeContainer="seq" ▶
smil:repeatCount="indefinite">
          <li id="1" smil:dur="1s" begin="0:01">
          <p>  <h1>Menu 1</h1></p>
          <p>  <b>Polévka: </b>Slepičí vývar</p>
          <p>  <em>Hlavní chod: </em>Svíčková</p>
      </li>
          <li id="2" smil:dur="1s" >
          <p>    <h2>Menu 2</h2></p>
          <p>    <b>Polévka: </b>Žampiónová</p>
          <p>   <em>Hlavní chod: </em>Sladké knedlíky</p>
          <p>Dezert: Tiramisu</p>
      </li>
          <li id="3" smil:dur="1s" >
          <p>    <h3>Menu 3</h3></p>
          <p>    <b>Polévka: </b>Knedlíčková</p>
          <p>    <em>Hlavní chod: </em>Obalovaný sýr a
            <math xmlns="http://www.w3.org/1998/Math/MathML">
            <mrow><mfrac><mrow><mn>1</mn></mrow><mn>2</mn></mfrac></mrow>
            </math>hranolků</p>
          <p>Dezert: Linecký dort </p>
      </li>
          </ul>
  </body>
</html>
```

Výsledkem tohoto kódu by mělo být cyklické zobrazování a skrývání jednotlivých menu po vteřině. Následující obrázek byl zachycen při zobrazení pouze menu č. 2.



Obrázek 17.1. Správné zobrazení stránky s podporou SMIL Timesheets

Kapitola 18

Testování prohlížečů

V následujících tabulkách budu testovat odchýlení od očekávaných stavů. Pro lepší přehlednost tabulek jsem si zvolila následující zkratky. Internet Explorer 8 - IE8; Internet Explorer 9 - IE9; Chrome 14 - Ch; Opera 11.52 - Od; Mozilla Firefox 12 - F; Safari 5.1 (na OS Mac X Lion 10.7.3) - Sd; Safari 5.1 (mobilní) - Sm; Opera Mini 6.5 (mobilní) - Omi; Opera Mobile 11.5 (mobilní) - Omo; Integrovaný prohlížeč OS Android 2.1 (mobilní) - An; Integrovaný prohlížeč OS Symbian S60 (mobilní) - Sy.

Tabulka 18.1. Testování prohlížečů 1/

Prohlížeče	Zobrazení XML bez stylů	Zobrazení chybného XML bez stylů	Zobrazení XML se styly CSS
IE8	Zobrazeno správně	Nevypíše přesnou polohu chyby	Zde podpora dopadla nejhůře; nepodporuje výběr elementu podle atributu a jeho hodnoty, funkci vepsání textu před/za daný element, vepsání hodnoty atributu; nepodporuje ani jmenné prostory; CSS se zobrazí i přesto, že nemá nastavený jmenný prostor na dokument se jmenným prostorem
IE9	Zobrazeno správně	Chybu ignoruje; zobrazí jako kód HTML (než narazí na chybu); pokud si zobrazíme konzoli pro vývojáře, pak je nám ukázána první chyba v kódu i s celým kódem	Nepodporuje vypsání hodnoty atributu; CSS se zobrazí i přesto, že nemá nastavený jmenný prostor na dokument se jmenným prostorem
Ch	Zobrazeno správně	Ke zvolenému výchozímu stavu přidá prohlížeč kód zpracovaný jako HTML do chyby	Nepodporuje vypsání hodnoty atributu; CSS se zobrazí i přesto, že nemá nastavený jmenný prostor na dokument se jmenným prostorem
Od	Není možné sbalit elementy	Navíc zobrazuje specifikaci a možnost zpracovat dokument jako HTML, při zvolení této možnosti se chyba zcela ignoruje a vypíše všechny texty	Bezchybná podpora; jako jediná vypisuje pomocí CSS hodnotu atributu; CSS se zobrazí i přesto, že nemá nastavený jmenný prostor na dokument se jmenným prostorem
F	Zobrazeno správně	Výchozí stav	Nepodporuje vypsání hodnoty atributu; CSS se zobrazí i přesto, že nemá nastavený jmenný prostor na dokument se jmenným prostorem
Sd	Zpracováno jako text HTML	Ke zvolenému výchozímu stavu přidá prohlížeč kód zpracovaný jako HTML do chyby	Nepodporuje vypsání hodnoty atributu; CSS se zobrazí i přesto, že nemá nastavený jmenný prostor na dokument se jmenným prostorem

Sm	Zpracováno jako text HTML	Ke zvolenému výchozímu stavu přidá prohlížeč kód zpracovaný jako HTML do chyby	Nepodporuje vypsání hodnoty atributu; CSS se zobrazí i přesto, že nemá nastavený jmenný prostor na dokument se jmenným prostorem
Omi	Zpracováno jako text HTML	Zpracuje dokument hned jako HTML, chyba se zcela ignoruje a vypíše všechny texty	Bezchybná podpora; jako jediná vypisuje pomocí CSS hodnotu atributu; CSS se zobrazí i přesto, že nemá nastavený jmenný prostor na dokument se jmenným prostorem
Omo	Zpracováno jako text HTML	Výchozí stav	Bezchybná podpora; jako jediná vypisuje pomocí CSS hodnotu atributu; CSS se zobrazí i přesto, že nemá nastavený jmenný prostor na dokument se jmenným prostorem
An	Zpracováno jako text HTML	Ke zvolenému výchozímu stavu přidá kód zpracovaný jako HTML do chyby	Nepodporuje vypsání hodnoty atributu; CSS se zobrazí i přesto, že nemá nastavený jmenný prostor na dokument se jmenným prostorem
Sy	Nezobrazuje XML	Nezobrazuje XML	Nezobrazuje XML

Tabulka 18.2. Testování prohlížečů 2/

Prohlížeče	XSLT	XSLT transformace pomocí Sarissy	XPath dotazy
IE8	Úplná podpora	Nepodporuje	Podpora výpisu elementu; nepodpora funkcí
IE9	Úplná podpora	Nepodporuje	Podpora výpisu elementu; nepodpora funkcí
Ch	Úplná podpora	Nepodporuje	Úplná podpora
Od	Úplná podpora	Úplná podpora	Úplná podpora
F	Úplná podpora	Úplná podpora	Úplná podpora
Sd	Úplná podpora	Nepodporuje	Úplná podpora
Sm	Úplná podpora	Nepodporuje	Úplná podpora
Omi	Úplná podpora	Úplná podpora	Úplná podpora
Omo	Úplná podpora	Úplná podpora	Úplná podpora
An	Nepodporuje	Nepodporuje	Nepodporuje
Sy	Nezobrazuje XML	Nepodporuje	Nepodporuje

Tabulka 18.3. Testování prohlížečů 3/

Prohlížeče	Data Islands	Data Islands pomocí scriptu; v XHTML	Podpora MIME
IE8	Plná podpora	V HTML nepodporuje, v XHTML podporuje	Korektně zobrazené typy text/xml a application/xml; application/xhtml+xml zobrazeno jako XML, ale ignorovalo značku nadpisu (element h1), u chybného souboru pouze výpis do chyby
IE9	Plná podpora	Plná podpora	U chybných XML zobrazeno jako HTML do chyby; u bezchybných XML korektně zobrazené typy text/xml a application/xml, application/xhtml+xml zobrazeno jako HTML, ale ignorovalo značku nadpisu (element h1)
Ch	Nepodporuje - zobrazí text daného xml v místě vložení XML	Plná podpora	U všech variant zobrazeno jako XML; u bezchybných jako kód XML, u chybných výpis chyby
Od	Nepodporuje - nezobrazí text v místě vložení XML	Plná podpora	U všech bezchybných XML zobrazeno jako XML; u chybných výpis chyby v případě text/xml a application/xml, u application/xhtml+xml se vypíše jako HTML s podporou elementu h1,chybu v tomto případě neohlásí
F	Nepodporuje - zobrazí text daného xml v místě vložení XML	Plná podpora	U všech variant zobrazeno jako XML; u bezchybných jako kód XML, u chybných výpis chyby
Sd	Nepodporuje - zobrazí text daného xml v místě vložení XML	Plná podpora	Předpokládám, že jsou všechny varianty zobrazeny jako XML; bezchybné XML se zde zobrazuje jako text HTML, u chybných se zobrazí výpis chyby

Sm	Nepodporuje - zobrazí text daného xml v místě vložení XML	Plná podpora	Předpokládám, že jsou všechny varianty zobrazeny jako XML; bezchybné XML se zde zobrazuje jako text HTML, u chybných se zobrazí výpis chyby
Omi	Nepodporuje - nezobrazí text v místě vložení XML	Plná podpora	Předpokládám, že jsou všechny varianty zobrazeny jako XML; bezchybné XML se zde zobrazuje jako text HTML, chybné se zobrazí jako XHTML a ignorují chybu
Omo	Nepodporuje - nezobrazí text v místě vložení XML	Plná podpora	U bezchybných předpokládám zobrazení jako XML, dokumenty s chybou posílané s text/xml a application/xml zobrazí stránku s chybou, application/xhtml+xml zobrazí jako HTML
An	Nepodporuje - zobrazí text daného xml v místě vložení XML	Plná podpora	Předpokládám, že jsou všechny varianty zobrazeny jako XML; XML se zde zobrazuje jako text HTML
Sy	Nepodporuje - zobrazí text daného xml v místě vložení xml	Plná podpora	Podporuje při bezchybných souborech, u chybných souborů zobrazí špatně application/xhtml+xml

Tabulka 18.4. Testování prohlížečů 4/

Prohlížeče	Validace	Externí entity	XML Base, XML Base v SVG
IE8	Úplná podpora	Úplná podpora	Nepodporuje
IE9	Úplná podpora	Nepodporuje	Nepodporuje
Ch	Nepodporuje	Nepodporuje	Nepodporuje samotné XML Base, podporuje v SVG
Od	Nepodporuje	Nepodporuje	Úplná podpora
F	Nepodporuje	Nepodporuje	Úplná podpora
Sd	Nepodporuje	Nepodporuje	Nepodporuje samotné XML Base, podporuje v SVG
Sm	Nepodporuje	Nepodporuje	Nepodporuje samotné XML Base, podporuje v SVG
Omi	Nepodporuje	Nepodporuje	Úplná podpora
Omo	Nepodporuje	Nepodporuje	Úplná podpora
An	Nepodporuje	Nepodporuje	Nepodporuje samotné XML Base, nepodporuje v SVG - nepodporuje SVG
Sy	Nepodporuje	Nepodporuje - nezobrazuje XML	Nepodporuje

Tabulka 18.5. Testování prohlížečů 5/

Prohlížeče	Manipulace s XML prostřednictvím DOM, načtení pomocí XMLHttpRequest	XForms	XMLHttpRequest
IE8	Plná podpora	Plná podpora	Plná podpora
IE9	Plná podpora	Plná podpora	Plná podpora
Ch	Plná podpora	Plná podpora	Plná podpora
Od	Plná podpora	Plná podpora	Plná podpora
F	Plná podpora	Plná podpora	Plná podpora
Sd	Plná podpora	Plná podpora	Plná podpora
Sm	Plná podpora	Plná podpora	Plná podpora
Omi	Plná podpora	Částečná podpora - ignoruje data vložená do textového pole	Plná podpora
Omo	Plná podpora	Plná podpora	Plná podpora
An	Plná podpora	Nepodporuje	Plná podpora
Sy	Plná podpora	Nepodporuje - nezobrazuje XML	Plná podpora

Tabulka 18.6. Testování prohlížečů 6/

Prohlížeče	xml:id	XInclude	SVG v XML
IE8	Nepodporuje	Nepodporuje	Nepodporuje - zobrazí kód
IE9	Nepodporuje	Nepodporuje	Nepodporuje animace
Ch	Nepodporuje	Nepodporuje	Plná podpora
Od	Plná podpora	Nepodporuje	Plná podpora
F	Nepodporuje	Nepodporuje	Plná podpora
Sd	Nepodporuje	Nepodporuje	Plná podpora
Sm	Nepodporuje	Nepodporuje	Plná podpora
Omi	Plná podpora	Nepodporuje	Nepodporuje animace
Omo	Plná podpora	Nepodporuje	Plná podpora
An	Nepodporuje	Nepodporuje	Nepodporuje, zobrazí pouze text „SVG XML“
Sy	Nepodporuje	Nepodporuje - nezobrazuje XML	Nepodporuje - nezobrazuje XML

Tabulka 18.7. Testování prohlížečů 7/

Prohlížeče	SVG v HTML5	SVG v XHTML	MathML v XML
IE8	Nepodporuje	Nepodporuje animace	Nepodporuje - zobrazí kód s prefixem jmenného prostoru
IE9	Nepodporuje animace	Nepodporuje animace	Nepodporuje - zobrazí kód
Ch	Úplná podpora	Úplná podpora	Nepodporuje - zobrazí kód
Od	Úplná podpora	Úplná podpora	Podporuje - mezi znaky, které mají být těsně u sebe dělá velké mezery
F	Úplná podpora	Úplná podpora	Podporuje - nepodpora pouze odřádkování
Sd	Úplná podpora	Úplná podpora	Úplná podpora
Sm	Úplná podpora	Úplná podpora	Úplná podpora
Omi	Nepodporuje animace	Nepodporuje animace	Zobrazí čísla bez zlomku, zobrazí znaménko ±, neodřádkuje, nezobrazí mocninu/odmocninu
Omo	Úplná podpora	Úplná podpora	Zobrazí čísla bez zlomku, zobrazí znaménko ±, neodřádkuje, nezobrazí mocninu/odmocninu
An	Nepodporuje, zobrazí pouze text „SVG XML“	Nepodporuje, zobrazí pouze text „SVG XML“	Zobrazí čísla bez zlomku, zobrazí znaménko ±, neodřádkuje, nezobrazí mocninu/odmocninu
Sy	Nepodporuje - zobrazí text „SVG XML“	Nepodporuje - zobrazí text „SVG XML“	Nepodporuje - nezobrazuje XML

Tabulka 18.8. Testování prohlížečů 8/

Prohlížeče	MathML v HTML5	MathML v XHTML	Jmenné prostory
IE8	Zobrazí čísla bez zlomku, zobrazí znaménko ±, neodřádkuje	Zobrazí čísla bez zlomku, zobrazí znaménko ±, neodřádkuje	Nepodporuje
IE9	Zobrazí čísla bez zlomku, zobrazí znaménko ±, neodřádkuje, nezobrazí mocninu/odmocninu	Zobrazí čísla bez zlomku, zobrazí znaménko ±, neodřádkuje, nezobrazí mocninu/odmocninu	Úplná podpora
Ch	Zobrazí čísla bez zlomku, zobrazí znaménko ±, neodřádkuje, nezobrazí mocninu/odmocninu	Zobrazí čísla bez zlomku, zobrazí znaménko ±, neodřádkuje, nezobrazí mocninu/odmocninu	Úplná podpora
Od	Podporuje - mezi znaky, které mají být těsně u sebe dělá velké mezery	Podporuje - mezi znaky, které mají být těsně u sebe dělá velké mezery	Úplná podpora
F	Podporuje - nepodpora pouze odřádkování	Podporuje - nepodpora pouze odřádkování	Úplná podpora
Sd	Úplná podpora	Úplná podpora	Úplná podpora
Sm	Úplná podpora	Úplná podpora	Úplná podpora
Omi	Zobrazí čísla bez zlomku, zobrazí znaménko ±, neodřádkuje, nezobrazí mocninu/odmocninu	Zobrazí čísla bez zlomku, zobrazí znaménko ±, neodřádkuje, nezobrazí mocninu/odmocninu	Úplná podpora
Omo	Zobrazí čísla bez zlomku, zobrazí znaménko ±, neodřádkuje, nezobrazí mocninu/odmocninu	Zobrazí čísla bez zlomku, zobrazí znaménko ±, neodřádkuje, nezobrazí mocninu/odmocninu	Úplná podpora
An	Zobrazí čísla bez zlomku, zobrazí znaménko ±, neodřádkuje, nezobrazí mocninu/odmocninu	Zobrazí čísla bez zlomku, zobrazí znaménko ±, neodřádkuje, nezobrazí mocninu/odmocninu	Úplná podpora
Sy	Zobrazí čísla bez zlomku, zobrazí znaménko ±, neodřádkuje, nezobrazí mocninu/odmocninu	Zobrazí čísla bez zlomku, zobrazí znaménko ±, neodřádkuje, nezobrazí mocninu/odmocninu	Nepodporuje - nezobrazuje XML

Tabulka 18.9. Testování prohlížečů 9/

Prohlížeče	XBL	SMIL Timesheet
IE8	Plná podpora	Nepodporuje
IE9	Plná podpora	Nepodporuje
Ch	Plná podpora	Plná podpora
Od	Plná podpora	Nepodporuje
F	Plná podpora	Plná podpora
Sd	Plná podpora	Plná podpora
Sm	Plná podpora - neberu v úvahu přejíždění myší	Plná podpora
Omi	Plná podpora - neberu v úvahu přejíždění myší	Nepodporuje
Omo	Plná podpora - neberu v úvahu přejíždění myší	Nepodporuje
An	Plná podpora - neberu v úvahu přejíždění myší	Nepodporuje
Sy	Plná podpora - neberu v úvahu přejíždění myší	Nepodporuje

Závěr

Moje práce byla tvořena s myšlenkou podpory XML v prohlížečích. Tato myšlenka mě provázela celou prací. Pokud některá technologie není podporována v prohlížečích nativně, snažila jsem se zjistit, jak je možné danou technologii využít tak, jak ji definuje její specifikace jen s pomocí nahrání veřejně šiřitelného JavaScriptu či pomocí XSLT. V průběhu práce jsem měla pocit, že webové prohlížeče příliš nepodporují XML technologie.

Závěr mé práce mě ovšem mile překvapil. Prohlížeče hojně podporují XML. Podle testů hodnotím prohlížeč Mozilla Firefox 12 jako prohlížeč nejvíce podporující XML technologie. Mezi testovanými mobilními prohlížeči si nejlépe vedla Opera Mobile 11.5, což se dalo očekávat už jen proto, že Opera se mezi desktopovými prohlížeči jeví jako další optimální volba prohlížeče po Firefoxu. Zklamáním pro mě bylo snížení podpory v novější verzi prohlížeče Internet Explorer oproti starší verzi.

Na druhou stranu jsem nečekala tak rozsáhlou podporu u prohlížečů spouštěných v operačním systému Macintosh. V globálním měřítku prohlížeče spíše podporují XML technologie nežli naopak. Vzhledem k tomu, že většina technologií není podporována úplně ve všech prohlížečích, je vhodné některé technologie použít v případě, kdy můžeme zaručit práci s nimi a jejich zobrazení v konkrétním prohlížeči.

S přihlédnutím k rychlosti rozvoje technologií je možné tuto práci průběžně aktualizovat a rozšiřovat o jiné (nové) XML technologie.

Příloha č. 1

Na přiloženém CD jsou všechny testovací kódy, co se v mé práci vyskytly. Testy jsou momentálně také k dispozici na stránce http://mansfeldova.com/xml/testy.html

Příloha č. 2

Seznam zkratek

XML - Extensible Markup Language
CSS - Cascading Style Sheets
XSLT - Extensible Stylesheet Language Transformations
HTML - HyperText Markup Language
XHTML - eXtensible HyperText Markup Language
DOM - Document Object Model
XInclude - XML Inclusions
XPath - XML Path Language
MIME - Multipurpose Internet Mail Extensions
SVG - Scalable Vector Graphics
MathML - Mathematical Markup Language
SMIL Timesheets - Synchronized Multimedia Integration Language
XLink - XML Linking Language
URI - Uniform Resource Identifier
AJAX - Asynchronous JavaScript and XML
GIF - Graphics Interchange Format
JPEG - Joint Photographic Experts Group
DTD - Document Type Definition
HTTP - Hypertext Transfer Protocol

Literatura

[1] ILINSKY, Sergey. *Xbl - xbl.js.* [online] [cit. 2012-04-25]. Dostupný na WWW: <http://code.google.com/p/xbl/>.

[2] MARSH, Jonathan; David ORCHARD a Daniel VEILLARD. *XML Inclusions (XInclude) Version 1.0 (Second Edition).* The World Wide Web Consortium. [online] [cit. 2011-12-20]. Dostupný na WWW: <http://www.w3.org/TR/xinclude/>.

[3] CAZENAVE, Fabien. *timesheets.js.* [online] [cit. 2012-04-25]. Dostupný na WWW: <http://wam.inrialpes.fr/timesheets/>.

[4] DEROSE, Steve; Eve MALER a David ORCHARD. *XML Linking Language (XLink) verze 1.0.* The World Wide Web Consortium [online] [cit. 2012-04-29]. Dostupný na WWW: <http://www.w3.org/TR/xlink/>.

[5] CARLISLE, David; Patrick ION a Robert MINER. *Mathematical Markup Language (MathML) Version 3.0* The World Wide Web Consortium [online] [cit. 2011-12-10]. Dostupný na WWW: <http://www.w3.org/TR/MathML3/>.

[6] GRIMMICH, Šimon. *SMIL - jazyk pro multimediální prezentace | Interval.cz.* The World Wide Web Consortium [online] [cit. 2012-03-02]. Dostupný na WWW: <http://interval.cz/clanky/smil-jazyk-pro-multimedialni-prezentace/>.

[7] KOSEK, Jiří. *Kapitola 1. Úvod.* [online] [cit. 2011-12-02]. Dostupný na WWW: <http://www.kosek.cz/xml/schema/uvod.html>.

[8] VUORIMAA, Petri; Dick BULTERMAN a Pablo CESAR. *SMIL Timesheets 1.0.* The World Wide Web Consortium. [online] [cit. 2011-11-22]. Dostupný na WWW: <http://www.w3.org/TR/timesheets/>.

[9] <agenceXML>. *XSLTForms - <agenceXML>.* [online] [cit. 2012-04-22]. Dostupný na WWW: <http://www.agencexml.com/xsltforms.htm>.

[10] KREJČÍ, Richard. *Grafika.cz - Encyklopedie publikačních formátů: XForms.* [online] [cit. 2011-11-18]. Dostupný na WWW: <http://www.grafika.cz/art/webdesign/encx-forms.html>.

[11] KESTEREN, Anne van. *XMLHttpRequest Level 2.* The World Wide Web Consortium [online] [cit. 2012-04-18]. Dostupný na WWW: <http://www.w3.org/TR/XMLHttpRequest/>.

[12] KAY, Michael. *XSL Transformations (XSLT) Version 2.1.* The World Wide Web Consortium [online] [cit. 2011-11-22]. Dostupný na WWW: <http://www.w3.org/TR/2010/WD-xslt-21-20100511/>.

[13] KAY, Michael. *XSL Transformations (XSLT) Version 2.0.* The World Wide Web Consortium [online] [cit. 2011-11-22]. Dostupný na WWW: <http://www.w3.org/TR/xslt20/>.

[14] Microsoft. *Author XML Data Islands.* [online] [cit. 2012-03-12]. Dostupný na WWW: <http://msdn.microsoft.com/en-us/library/windows/desktop/ms763762(v=vs.85).aspx>.

[15] Mozilla Developer Network. *Using XML Data Islands in Mozilla - MDN.* [online] [cit. 2012-02-12]. Dostupný na WWW: <https://developer.mozilla.org/en/Using_XML_Data_Islands_in_Mozilla>.

[16] HICKSON, Ian. *XML Binding Language (XBL) 2.0.* The World Wide Web Consortium [online] [cit. 2011-10-27]. Dostupný na WWW: <http://www.w3.org/TR/xbl/>.

[17] QUIN, Liam. *The Extensible Stylesheet Language Family (XSL).* The World Wide Web Consortium [online] [cit. 2011-10-25]. Dostupný na WWW: <http://www.w3.org/Style/XSL/>.

[18] Jiří. PETERKA, *MIME type.* [online] [cit. 2011-11-1]. Dostupný na WWW: <http://www.earchiv.cz/a97/a704k130.php3>.

[19] MARSH, Jonathan; Daniel VEILLARD a Norman WALSH. *Xml:id Version 1.0.* The World Wide Web Consortium. [online] [cit. 2011-11-5]. Dostupný na WWW: http://www.w3.org/TR/xml-id/>.

[20] DOSTÁLEK, Libor. *MIME - Multipurpose Internet Mail Extension.* [online] [cit. 2011-11-01]. Dostupný na http://www.cpress.cz/knihy/tcp-ip-bezp/CD-dalsi/CD-mime/mime.htm>.

[21] CASTRO, Elizabeth. *XML pro World Wide Web.* Brno: SoftPress, 2001. ISBN: 80-86497-07-0.

[22] POKORNÝ, Jaroslav. Irena MLÝNKOVÁ, Martin NEČASKÝ, Karel RICHTA, Kamil TOMAN a Vojtěch TOMAN. *XML technologie, principy a aplikace v praxi.* Praha: Grada, 2008. ISBN: 978-80-247-2725-7.

[23] SKONNARD, Aaron a Martin GUDGIN. *XML - pohotová referenční příručka. Referenční příručka programátora ke XML, XPath, XSLT, XML Schema, SOAP a dalším.* Praha: Grada, 2006. ISBN: 80-247-0972-4.

[24] KOSEK, Jiří. *PHP a XML.* Praha: Grada, 2009. ISBN: 978-80-247-1116-4.

[25] HUB, Miloslav. *Technologie internetu - XML, distanční opora.* Pardubice: Univerzita Pardubice, 2010. ISBN: 978-80-7395-306-5.

[26] KAY, Michael. *XSLT 2.0 and XPath 2.0 Programmer's Reference 4th Edition.* Canada: Wiley Publishing, Inc., 2008. ISBN: 978-0-470-19274-0.

[27] BRADLEY, Neil. *XML kompletní průvodce.* Praha: Grada, 2000. ISBN: 80-7169-949-7.