Západočeská univerzita v Plzni

FAKULTA PEDAGOGICKÁ
KATEDRA VÝPOČETNÍ A DIDAKTICKÉ TECHNIKY

DISTANČNÍ KURZ AJAX

DIPLOMOVÁ PRÁCE

Bc. Vlastimil Landsman
Učitelství pro 2. stupeň ZŠ, obor INF-Te

Vedoucí práce: doc. Ing. Václav Vrbík, CSc.

Plzeň, 2013

Prohlašuji, že jsem diplomovou práci vypracoval samostatně s použitím uvedené literatury a zdrojů informací.

Plzeň, 28. červen 2013

……
vlastnoruční podpis

Na tomto místě bych rád poděkoval vedoucímu diplomové práce Doc. Ing. Václavu Vrbíkovi, CSc. za poskytnuté materiály a přínosné podněty ke kurzu i vlastní práci.

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Vlastimil LANDSMAN**
Studijní program:	**N7503 Učitelství pro základní školy**
Studijní obory:	**Učitelství informatiky pro 2. st. ZŠ**
	Učitelství technické výchovy pro 2. st. ZŠ
Název tématu:	**Distanční kurz Ajax**
Zadávající katedra:	**Katedra výpočetní a didaktické techniky**

Zásady pro vypracování:

Vytvořte materiál pro distanční výuku Ajaxu.
1. Vytvořte studijní články, vhodné testy, autotesty, cvičení a úkoly v prostředí ProAuthor.
2. Přílohou diplomové práce bude CD/DVD s dokumentací a off-line verzí navrženého kurzu.

Rozsah grafických prací:

Rozsah pracovní zprávy: **40 - 100 stran + CD**

Forma zpracování diplomové práce: **tištěná**

Seznam odborné literatury:

ASLESON, R., SCHUTTA, N. Ajax : Vytváříme vysoce interaktivní webové aplikace. Brno : Computer Press, 2006. 270 s. ISBN 80-251-1285-3.

HOLZNER, S. Ajax : A Beginners Guide. New York : McGrawHil, 2009. 475 s. ISBN 0-07-159531-7.

Vedoucí diplomové práce: **Doc. Ing. Václav Vrbík, CSc.**
Katedra výpočetní a didaktické techniky

Datum zadání diplomové práce: **4. ledna 2010**

Termín odevzdání diplomové práce: **15. března 2011**

v.z.

Doc. PaedDr. Jana Coufalová, CSc.
děkanka

Západočeská univerzita v Plzni
Fakulta pedagogická
L.S.

Doc. Ing. Václav Vrbík, CSc.
vedoucí katedry

OBSAH

ÚVOD

Tématem diplomové práce je tvorba studijního materiálu pro distanční výuku AJAXu. Cílem distančního kurzu je naučit studenty vytvářet webové aplikace využívající technologie AJAX. Pomocí technologií AJAX se naučí zpracovávat nejen textová data, ale také data ve formátu XML. V závěru kurzu je zařazena práce s JavaScript knihovnami jQuery a knihovnou Google Charts pro vizualizaci grafů. Distanční kurz předpokládá perfektní orientaci v značkovacích jazycích HTML, kaskádových stylech CSS a JavaScriptu**.** Základy nejsou vysvětleny a pro začátečníka, bez znalosti programování a webových technologií, obecně může být problém kurz zdárně absolvovat. Jako samozřejmost je dnes brána v informatice znalost anglického jazyka, alespoň na úrovni čtení technické dokumentace. Příklady jsou proto psány kompletně anglicky s výjimkou komunikace s uživatelem.

Podstatou kurzu není zapamatování všech metod a vlastností, ale získání nadhledu na problematikou AJAXu. Proto je v kurzu zařazeno velké množství praktických příkladů a ukázek, na místo teoretického testování znalostí. Mým názorem je, že dobře okomentovaný zdrojový kód řekne více než tisíce slov. Pro splnění kurzu potřebuje student získat 42 / 80 bodů v 6 samostatných úkolech. Pokud je ani po opravě nedosáhne, body lze ještě získat v závěrečném rozsáhlejším příkladu.

Distanční kurz je postaven na současně nejmodernějších technologiích tvorby webových stránek HTML5 spolu s kódováním utf-8 a zařazeny jsou i velice populární JavaScriptové knihovny jQuery a Google Charts. Standart HTML5 v současnosti ještě není plně schválen organizací W3C[1], ale již dnes je díky zpětné kompatibilitě nasazován na webové stránky. Dnes je vývoj tak překotný, že nemá smysl psát odbornou literaturu, neb již dnem vydání se stává zastaralou a je nutná její revize. Naneštěstí je k dispozici hodně elektronických zdrojů a kurzů dostupných zdarma na internetu.

[1] World Wide Web Consortium (W3C) je mezinárodní konsorcium, jehož členové společně s veřejností vyvíjejí webové standardy pro World Wide Web. Cílem konsorcia je „Rozvíjet World Wide Web do jeho plného potenciálu vývojem protokolů a směrnic, které zajistí dlouhodobý růst Webu“

V první kapitole diplomové práce jsou popsány technologie spadající pod termín AJAX včetně ukázek zdrojového kódu z úvodní kapitoly distančního kurzu. Stejným způsobem je strukturován i distanční kurz. Lehký úvod seznámí studenty se základními prvky AJAXu. V druhé kapitole je rozebrán vývoj distančního kurzu a integrace spouštěče (interpreteru) ukázek do autorského systému ProAuthor. Tomu je věnován prostor hned v úvodu. Ve třetí kapitole jsou podrobněji vysvětleny technologie používané při tvorbě AJAX aplikací včetně okomentovaných ukázek zdrojových kódů použitých v kurzu. Pro potřebu diplomové práce byly zkráceny a přeformulovány. Kromě závěrečného shrnutí je krátce popsán také obsah přiloženého CD.

1 TECHNOLOGIE AJAX

1.1 AJAX

*AJAX (**A**synchronous **J**avaScript **A**nd **X**ML) je obecné označení pro technologie vývoje interaktivních webových aplikací, které mění obsah svých stránek bez nutnosti jejich znovu načítání. Na rozdíl od klasických webových aplikací poskytují uživatelsky příjemnější prostředí, ale vyžadují použití moderních webových prohlížečů.* (1)

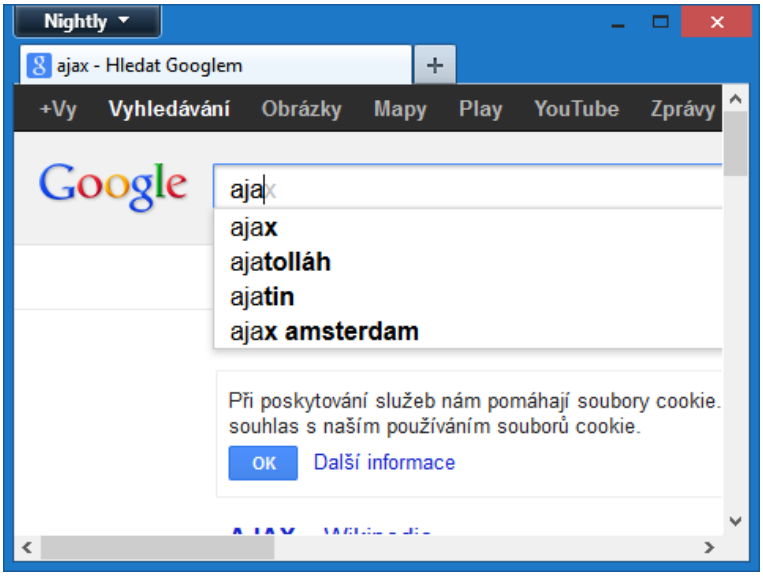
AJAX JE ZALOŽENÝ NA INTERNETOVÝCH STANDARDECH, A VYUŽÍVÁ KOMBINACE TECHNOLOGIÍ:

- HTML (XHTML, HTML5) pro vizuální (textovou) prezentaci informací ve webovém prohlížeči.
- CSS (CSS 1, 2, 3) pro formátování vzhledu informací.
- DHTML / DOM a JavaScript pro dynamické změny obsahu webové stránky.
- XMLHttpRequest pro výměnu dat s webovým serverem, typicky XML, ale je možné použít i čistý (plain) text.

1.1.1 HISTORIE

Termín AJAX se poprvé objevil v dubnu 2005 v článku umístěném na webu Jesse James Garretta[2], nazvaném *Ajax: Nový přístup k webovým aplikacím*. Historicky se podobné techniky používaly již od IE3 se zavedením tagu `<iframe>` a produktech Macromedia Flash4 (dnes Adobe Flash). XMLHttpRequest se však objevil až v Internet Exploreru 5 s využitím Active-X (framework pro sdílení dat mezi aplikacemi). Do povědomí se AJAX dostal díky společnosti Google a jejím využitím ve vyhledávači (našeptáváč), mapách Google maps Gmailu. Dnes je hojně používaný na facebooku nebo youtube a masivně se dostává do povědomí odborníků po celém světě.

[2] http://www.adaptivepath.com/ideas/ajax-new-approach-web-applications



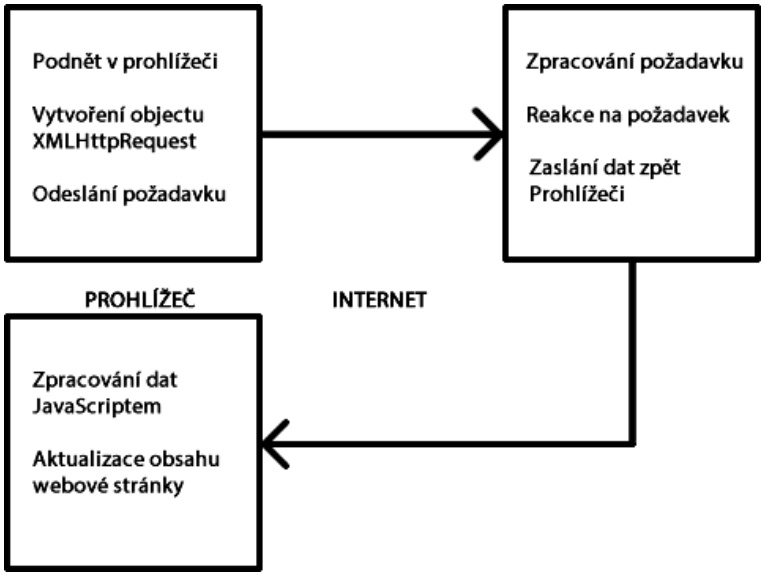
Obrázek 1: Našeptávač Google

VÝHODY

- Odstranění nutnosti znovunačtení a překreslení celé stránky při interakci uživatele (odeslání formuláře apod.).
- Plynulejší práce a přiblížení k desktopovým aplikacím.
- Snížení zátěže na webovém serveru.
- Nezávislost na platformě a prohlížeči.

NEVÝHODY

- V AJAX aplikacích nemusí vždy plně fungovat navigace (tlačítko Zpět, Vpřed), nelze odkázat URL, přidat stránku do oblíbených.
- Může docházet k určité latenci, uživatel se nemusí dovědět o problémech, protože komunikace probíhá na pozadí, záleží na implementaci kódu.
- Vyžadovány jsou moderní prohlížeče, nicméně dnes není problém ani s mobilními zařízeními.



Obrázek 2: Princip AJAX aplikace

1.1.2 PRINCIP AJAX APLIKACÍ

Princip je ilustrován na obrázku 2.

- JavaScript odchytí požadavek na načtení a požádá webový server pouze o změněnou část dat.
- Server požadavek vyhodnotí a data pošle v předem připraveném formátu (nejčastěji XML).
- JavaScript odpověď příjme, zkontroluje a pozmění stránku.

1.2 HTML

***H**yper**T**ext **M**arkup **L**anguage je značkovací jazyk pro vytváření www (**W**orld **W**ide **W**eb) stránek, kterým se publikují dokumenty na internetu. HTML je zjednodušením standartu SGML (**S**tandard **G**eneralized **M**arkup **L**anguage).* (2)

1.2.1 HISTORIE

V roce 1989 spolupracovali Tim Berners-Lee a Robert Caillau na propojeném informačním systému pro CERN, výzkumné centrum fyziky poblíž Ženevy ve Švýcarsku. V té době se pro tvorbu dokumentů obvykle používaly jazyky TeX, PostScript a také SGML. Berners-Lee si uvědomoval, že potřebují něco jednoduššího a v roce 1990 navrhl jazyk HTML a protokol pro jeho přenos v počítačové síti - **HTTP** (**H**yperText **T**ransfer **P**rotocol - přenosový protokol hypertextu). Zároveň také Tim Berners-Lee napsal první webový prohlížeč, který nazval WorldWideWeb.

1.2.2 XHTML

V současnosti je nejvíce rozšířena verze 4.1 z roku 1999. V roce 2000 byl konsorciem W3C standardizován XHTML 1.0 vycházející z XML. Cílem bylo při zpětné kompatibilitě zjednodušit návrh webových stránek tak, aby vyhovovaly standardu XML. Došlo k odstranění překonaných značek a verze XHTML 1.0 Strict předpokládá ještě větší oddělení stylů CSS od dokumentu. XHTML 1.1 je rozšíření XHTML 1.0 o modularizaci.

1.2.3 HTML5

Budoucnost webu patří specifikaci HTML5. Hlavními novinkami je podpora multimédií (2D/3D grafika), nové tagy (prvky) pro audio, video, formuláře (`<range>`) a podpora offline režimu. Některé prvky byly naopak zrušeny. Interpreter i ukázky v kurzu jsou psané dle specifikace XHTML, nicméně celý kurz (mimo část generovanou ProAuthorem) byl následně upraven a validován dle HTML5. Zatím není jisté, jestli vznikne i striktní varianta HTML5, která bude mít blíže k XML podobně jako XHTML.

1.2.4 NĚKTERÉ ZJEDNODUŠENÉ PRVKY HTML5:

Kromě zjednodušení `<!doctype html>` se upravila meta značka formátování. Meta značka může obsahovat informace o stránce, jejím kódování pro prohlížeč. Nepárové značky nemusí být nutně ukončeny lomítkem, ale je to dobrým mravem převzatým z XHTML. Na první řádce meta značka používaná v XHTML na druhém v HTML5:

```
<meta http-equiv="content-type" content="text/html; charset=UTF-8" />
<meta charset="UTF-8" />"><!—zde meta značka HTML5, nad ní klasická XHTML -->
```

Při vložení skriptu do stránky (obvykle do hlavičky `<head>`) nebo připojení ke stránce pomocí `src`, již není nutné uvádět, že se jedná o JavaScript, protože se v současnosti jiný skriptovací jazyk na webu nepoužívá:

```
<script type="text/javascript">/* obsah JavaScriptu v XHTML */</script>
<script>/* obsah JavaScriptu v HTML5 */</script>
```

NOVÉ PRVKY HTML5:

Ve specifikaci HTML5 byly přidány nové zajímavé prvky pro multimédia a formuláře:

Multimediální prvky	Popis
<canvas>	Prvek k vykreslování grafiky za chodu pomocí JavaScriptu
<audio>	Prvek audio obsahu
<video>	Prvek video obsahu
<source>	Prvek s definicí více zdrojů pro audio a video
<embed>	Prvek pro vložení interaktivního obsahu (plugin, flash)
<track>	Prvek definující seznam skladeb pro audio a video

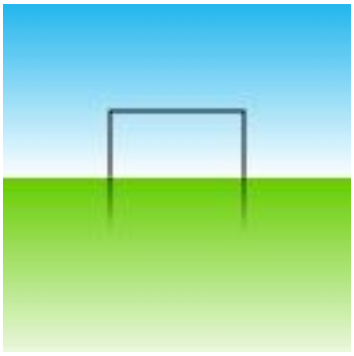
Prvky formulářů	Popis
<datalist>	Prvek k vykreslování grafiky za chodu pomocí JavaScriptu
<keygen>	Prvek pro umístění generových hesel
<output>	Prvek pro výsledek výpočtu
<range>	Prvek posuvníku

Naopak došlo k odstranění překonaných (deprecated) značek, neznamená to však, že je dále nelze používat, při validaci bude vypsána chyba, prvek je překonaný:

- `<acronym>`
- `<applet>`
- `<basefont>`
- `<big>`
- `<center>`
- `<dir>`
- `<font>`
- `<frame>`
- `<frameset>`
- `<noframes>`
- `<strike>`
- `<tt>`

V kurzu je krátká ukázka HTML5 stránky využívající kreslící plátno `<canvas>` a příslušný JavaScript k nakreslení branky (Obrázek 3). Cílem je, aby si studenti vyzkoušeli, zda si jimi používaný prohlížeč rozumí s HTML5 standardem:

```
<!doctype html>
<html lang="cs" dir="ltr">
  <head>
    <meta charset="UTF-8" />
    <title>HTML5 kreslící plátno canvas</title>
    <script>
      function drawShape(objID){ /* JavaScript */ }
    </script>
  </head>
  <body onload="drawShape('canvasID');">
    <canvas id="canvasID"></canvas>
  </body>
</html>
```



Obrázek 3: Branka v HTML5

1.3 XML

*E**x**tensible **M**arkup **L**anguage, rozšiřitelný značkovací jazyk je stejně jako XHTML zjednodušenou podobou staršího jazyka SGML. Ve světě informačních technologií způsobil malou revoluci. Po schválení konsorciem W3C v roce 2008 se rychle stal standardním formátem pro výměnu a sdílení informací. Na rozdíl od HTML byl navržen pro přenášení dat nikoliv k jejich prezentaci. Je dobře čitelný stroji, tak lidmi (tady by se dalo polemizovat). Názvy XML tagů nejsou určeny a je na uživateli, aby si nadefinoval vlastní. XML se skládá ze stromové struktury. Ta začíná kořenovým root prvkem (document), pod ním jsou prvky potomků tzv. child. Prvky mohou mít libovolnou hloubku a podprvky tzv. subchild. Child prvky na stejné úrovni se označují jako sourozenci (siblings). Jednotlivé prvky stejně jako HTML mohou mít textový obsah a atributy.* (3)

1.3.1 VÝHODY A NEVÝHODY XML

PRO PŘENOS DAT:

- Je to formát čitelný člověkem i strojem.
- Podporuje unicode, tj. všechny světové jazyky i další symboly a značky.
- Dokáže reprezentovat nejběžnější datové struktury: záznamy, seznamy, stromy.
- Je sebe popisný - definuje zároveň jména datových položek, jejich strukturu a hodnoty.
- Pevná syntaxe umožňuje jednoduché a efektivní strojové zpracování (parsování).

JAKO ÚLOŽIŠTĚ DAT:

- Je standardizován.
- Hierarchická struktura je vhodná pro většinu druhů dat.
- XML soubor je obyčejný text - nevztahují se na něj žádné patenty.
- Je nezávislý na platformě a to mu zaručuje dobrou přenositelnost.

NEVÝHODY:

- Syntaxe XML může být "nabobtnalá" - snížená přehlednost pro člověka může vést k chybám.
- Soubory mohou být poměrně velké, ale problém řeší komprese (mimo web).
- XML strom může být libovolně hluboký, zvyšuje se zátěž na procesor i paměť.
- XML samo o sobě obsahuje jen textová data - jiné datové typy je nutno definovat.
- Využití pro obecnější než stromové struktury je složitější.

V kurzu je uveden příklad, jak vypadá struktura jednoduchého XML souboru, ten obsahuje kromě prvků i atribut typ vozidla:

```
<?xml version="1.0" encoding="utf-8"?>
<auta>
  <auto typ="osobni">
  <jmeno>Ford</jmeno>
    <rychlost>200</rychlost>
    <rok>2005</rok>
    <cena>500 000</cena>
  </auto>
  <auto typ="osobni">
    <jmeno>Mazda</jmeno>
    <rychlost>220</rychlost>
    <rok>2014</rok>
    <cena>800 000</cena>
  </auto>
  <auto typ="nakladni">
    <jmeno>Citroen</jmeno>
    <rychlost>150</rychlost>
    <rok>2010</rok>
    <cena>1 000 000</cena>
  </auto>
</auta>
```

1.4 DOM A JAVASCRIPT

Java**S**cript (**JS**) je nejčastěji používaná kombinace ve vztahu k DOM. Technologie Microsoft VBScript (na webu) se považuje dnes za překonanou a nepoužívá se.

1.4.1 JAVASCRIPT

JavaScript je multiplatformní, objektově orientovaný skriptovací jazyk, jehož autorem je Brendan Eich ze společnosti Netscape. Jeho syntaxe patří do rodiny jazyků C / C++ / Java. Slovo Java je však součástí jeho názvu pouze z marketingových důvodů a s programovacím jazykem Java jej vedle názvu spojuje jen podobná syntaxe a interpretace (interpretovaný jazyk). JavaScript byl v červenci 1997 standardizován asociací ECMA. JavaScripty se spouštějí po stažení stránky na straně klienta. Tím se liší od interpretovaných programovacích jazyků PHP nebo ASP, které se spouštějí na straně serveru. Nicméně existují i jeho serverové implementace Node.js a Apache. (4)

1.4.2 DOM

Document Object Model je objektově orientovaná reprezentace popisující strukturu dokumentu HTML nebo XML. DOM umožňuje přístup k dokumentu jako ke stromu, což je datová struktura používaná ve většině XML parserů. Tato technologie, nazývaná GROVE (Graph Representation Of property ValuEs), vyžaduje nahrání celého parsovaného dokumentu do paměti, z čehož plyne, že její optimální použití je tam, kde je k jednotlivým prvkům dokumentu přistupováno v náhodném pořadí nebo opakovaně. Opakem je

sekvenční model SAX, který je rychlejší a méně náročnější na zdroje. V současnosti jsou konsorciem W3C standardizovány 3 úrovně - level. Zásadní je level 1 popisující navigaci v DOM (HTML a XML) dokumentu (resp. jeho stromové struktuře) manipulaci s obsahem (včetně přidávání prvků) a akcemi (klepnutí myši). (5)

V kurzu jsou uvedené příklady nezbytné pro pochopení následující kapitoly XMLHttpRequest, čtení a nastavování hodnot prvků:

```
proměnná = document.getElementById("divID").innerHTML;
/* načte textový obsah prvku s id="divID" do proměnné */

proměnná = document.getElementsByTagName("divID")[0].innerHTML;
/* do proměnné načte textový obsah prvního divu v dokumentu */

document.getElementById(divID).innerHTML = "Ahoj Světe";
/* nastaví text "Ahoj Světe" do prvku uloženém v proměnné divID */

proměnná = document.forms[0].elements["firstname"].value;
/* z formuláře vybere prvního potomka a jeho hodnotu uloží do proměnné */
```

Následuje krátká ukázka, jak lze pomocí DOM a JavaScriptu přidávat do dokumentu prvky nové:

```
var option = document.createElement("option"); // vytvoří prvek option
option.value = i; // nastaví hodnotu na proměnnou i
option.text = i; // nastaví text na proměnnou i
document.getElementById(objID).add(option, null); // připojí prvek
```

1.5 XMLHTTPREQUEST

Základním stavebním kamenem AJAX aplikací je objekt XMLHttpRequest. Všechny moderní prohlížeče ho podporují. Starší prohlížeče IE verze 5 a 6 ho implementují rozdílně a to jako ActiveXObject. V kurzu je předpokládáno použití moderních prohlížečů, IE5, 6 jsou považovány dávno za překonané. XMLHttpRequest objekt se používá pro výměnu dat se serverem na pozadí. To s využitím JavaScriptu přináší možnost aktualizace části webových stránek bez nutnosti jejich znovunačtení. (6)

1.5.1 VYTVOŘENÍ XMLHTTPREQUEST OBJEKTU

Všechny moderní prohlížeče (IE7+, Firefox, Chrome, Safari, a Opera) mají vestavěný `XMLHttpRequest` objekt. Pro vytvoření se používá následující syntaxe:

```
var proměnná = new XMLHttpRequest(); /* založení objektu v moderním prohlížeči */
var proměnná = new ActiveXObject("Microsoft.XMLHTTP");
/* staré verze Internet Exploreru používají ActiveX Object: */
```

Pro obsloužení všech prohlížečů, je možné ověřit, zda podporují XMLHttpRequest objekt. Pokud ne, pokusí se založit ActiveXObject:

```
var XMLHttpRequestObject = false; /* nastavení počátečního stavu (není nezbytné,
nová hodnota je vždy false, null, "", nebo 0) */
if (window.XMLHttpRequest) { /* kód pro IE7+, Firefox, Chrome, Opera, Safari */
  XMLHttpRequestObject = new XMLHttpRequest();
} else if (window.ActiveXObject) { /* kód pro IE5, 6 */
  XMLHttpRequestObject = new ActiveXObject("Microsoft.XMLHTTP");
}
```

1.5.2 ZASLÁNÍ POŽADAVKU NA SERVER

Pro zaslání požadavku na server se používají metody `open()` a `send()` objektu `XMLHttpRequest`. Ve většině případů bude stačit použít jednodušší GET. POST je z důvodu bezpečnosti doporučen pro zasílání formulářových dat a díky limitaci GET také souborů větších rozměrů. `Open()` má jako parametry typ metody, URL (může být xml, txt, php, asp atd.) a volbu, zda bude použit asynchronní dotaz `true | false`. Při vynucení `false` JavaScript nebude pokračovat v provádění kódu, dokud nedostane odpověď od serveru, což může aplikaci zastavit

```
XMLHttpRequestObject.open("GET", URL, true);
XMLHttpRequestObject.send(); // někdy se u GETu uvádí parametr send(null)
```

1.5.3 ZPRACOVÁNÍ ODEZVY SERVERU

Pro získání odezvy ze serveru se používá vlastnost `responseText` nebo `responseXML` objektu XMLHttpRequest. `ResponseText` obsahuje výsledek ve formě string a lze použít následovně:

```
document.getElementById(divID).innerHTML = XMLHttpRequestObject.responseText;
```

Kurz obsahuje funkční ukázku JavaScript kódu, který stáhne, přečte a zobrazí názvy aut z XML souboru vozidel uvedené v kapitole XML:

```
var xmlDoc = XMLHttpRequestObject.responseXML;
/* načte obsah (strom) XML do proměnné */
var car = xmlDoc.getElementsByTagName("jmeno");
/* z XML vytvoří pole prvků jmeno a uloží do proměnné car */
for (var i = 0; i < car.length; i++) {  /* projde celé pole car */
  document.getElementById(objID).innerHTML +=
  car[i].childNodes[0].nodeValue + "<br />";
  /* vloží název auta a odřádkuje */
}
```

1.5.4 ASYNCHRONNÍ ZPRACOVÁNÍ ODEZVY

Událost `onreadystatechange` je vyvolána pokaždé, kdy se změní stav požadavku `readyState`. `ReadyState` obsahuje stav `XMLHttpRequestu`. Pokud je vše v pořádku, nabývá hodnotu 4. Pomocí vlastnosti `status` lze získat HTTP odezvu na požadavek. Výsledek 200 je OK, 404 stránka nebyla nalezena.

```
XMLHttpRequestObject.onreadystatechange = function() {
  if (XMLHttpRequestObject.readyState == 4 && XMLHttpRequestObject.status == 200)
  { /* pokud jsou oba stavy v pořádku */
    document.getElementById(objID).innerHTML = XMLHttpRequestObject.responseText;
    /* do prvku odkázaném v objID se uloží získaná textová data */
  }
}
```

1.6 CSS

*Kaskádové styly (**C**ascading **S**tyle **S**heets) se používají pro popis způsobu zobrazení stránek napsaných v jazycích HTML, XHTML. Jazyk byl navržen organizací W3C, autorem návrhu byl Håkon Wium Lie (technický ředitel firmy Opera, mimo jiné spolupracovník Tima Berners-Lea). Byly zatím vydány dvě specifikace CSS 1 a CSS 2. Dne 7. června 2011 byla dokončena revize CSS 2.1 a pracuje se na verzi CSS 3. Hlavním důvodem používání je možnost oddělit vzhled dokumentu od jeho struktury a obsahu. Starší verze HTML obsahují celou řadu prvků, které nepopisují obsah a strukturu dokumentu, ale i způsob jeho zobrazení. Z hlediska zpracování dokumentů a vyhledávání informací není takový vývoj žádoucí. CSS 3 má úzkou vazbu na budoucí standard HTML5, jež jej má plně využívat. V současné době je hodně webů tvořeno s využitím DHTML (dynamické HTML poháněné JavaScriptem), případně s využitím frameworku jQuery (JavaScriptová knihovna). Trendem je, aby co nejvíce těchto funkčností (většinou animace a grafické prvky) bylo možné tvořit za pomocí webových standardů.* (7)

1.6.1 CSS A AJAX

I přesto, že zkratka CSS stejně jako HTML není součástí termínu AJAX, běžně se používá JavaScriptu k dynamické modifikaci vzhledu webových stránek. V distančním kurzu je ukázka jak efektně upozornit návštěvníka webové stránky na změnu obsahu dokumentu pomocí obarvení textu:

```
document.getElementById("divID").style.color = "#00f";
/* nastaví styl barvy v prvku s "divID" na modrou */
document.getElementById("divID").innerHTML =
XMLHttpRequestObject.responseText;
/* do prvku vloží získaný text */
setTimeout("document.getElementById("divID").style.color = '#000'",
1000);
/* za 1 sekundu nastaví barvu zpět na černou *
```

1.6.2 ZÁVĚR KAPITOLY TECHNOLOGIE AJAX

V úvodní kapitole kurzu byly vysvětleny základy technologií, na kterých jsou AJAX aplikace založeny. Současně si studenti mohli udělat reálnou představu, co je v kurzu čeká díky praktickým ukázkám, kde si vyzkoušeli funkčnost Interpreteru zdrojového kódu. Po úvodní kapitole je zařazen krátký autotest, kde si studenti ověří nabyté teoretické znalosti. Všechny uvedené technologie jsou v distančním kurzu dále prohlubovány pomocí studijních článků, ale hlavně praktických ukázek a cvičení. Každé cvičení obsahuje tipy pro řešení a rozhodl jsem se poskytnout kompletní zdrojové kódy se správnými řešeními příkladů. Tyto zdrojové kódy neobsahují záměrně komentáře a pobízejí studenty k přemýšlení, když už samostatně nedokáží příklad vyřešit. Po cvičení následuje hodnocený složitější úkol. Zde jsou uvedeny pouze tipy pro řešení příkladu, kompletní zdrojový kód má k dispozici pouze lektor nebo tutor kurzu.

2 VÝVOJ DISTANČNÍHO KURZU

2.1 PROAUTHOR

V zadání práce bylo požadováno vytvoření kurzu pomocí autorského systému ProAuthor. ProAuthor je nástroj vyvinutý na Západočeské univerzitě v Plzni. Je určen k tvorbě e-learningových výukových kurzů v distančním vzdělávání. Distanční kurz byl vytvořen ve verzi ProAuthor 7.6.3. Studenti Západočeské univerzity mají možnost získat licenci pro výukové účely zdarma.

Dle popisu autorů (8) mezi jeho hlavní výhody patří:

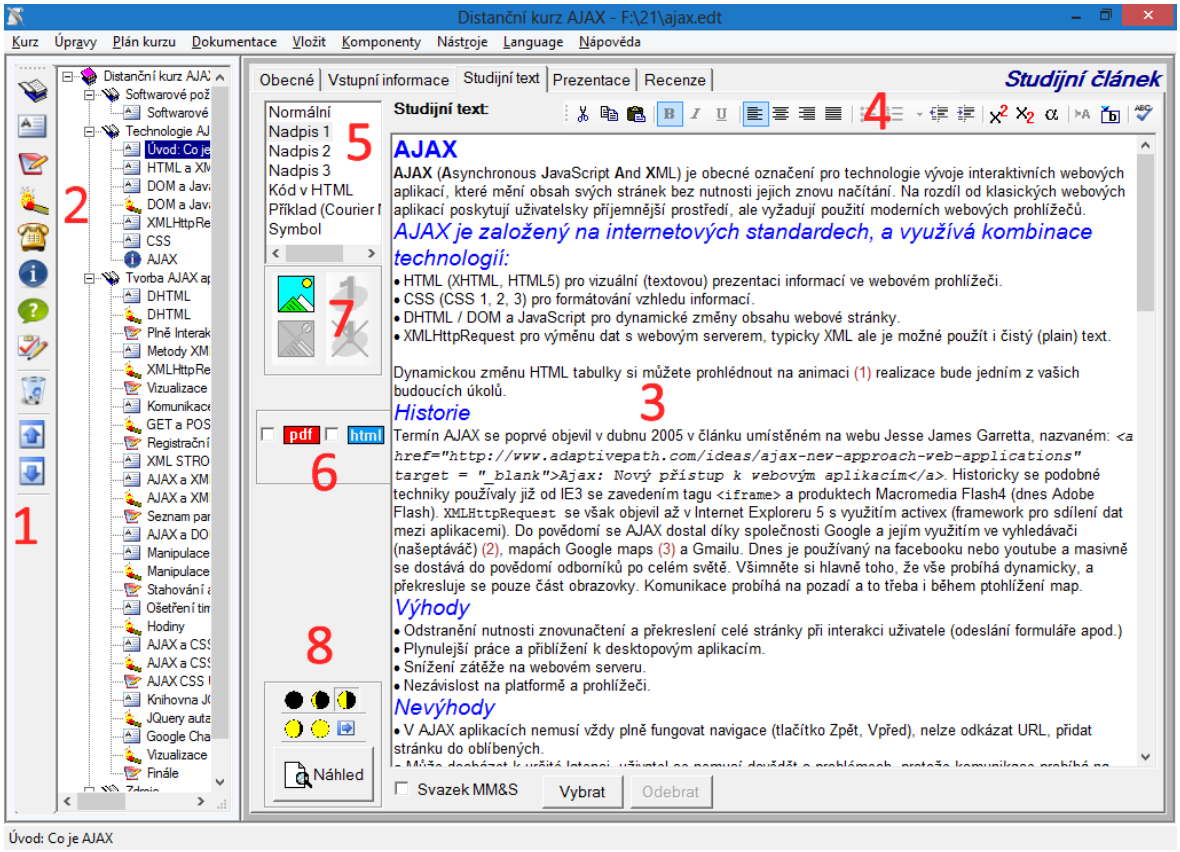
- **Zjednodušení tvorby vzdělávacího obsahu**. Tvůrce kurzu nepotřebuje znalosti programování v HTML. Uživatelské ovládání je zjednodušeno na úroveň jednoduchého textového WYSIWYG[3] editoru.
- **Možnost importu multimediálních nebo programátorky náročných prvků**. Vzhledem k zjednodušení autorského rozhraní má autor možnost definovat multimediální komponenty (flash animace, video, simulace), ty je však nutné vytvořit v externím softwaru.
- **Podpora týmové tvorby, včetně jazykových mutací obsahu kurzu**. Nástroj ProAuthor disponuje automatizovanými nástroji pro slučování jednotlivých částí kurzu od různých autorů a zároveň umožňuje překladateli pracovat na jazykové mutaci obsahu kurzu.
- **Recenzní nástroje**. ProAuthor nabízí autorům a recenzentům nástroje pro vytváření a zpracování recenzních poznámek, včetně možnosti tisku a slučování recenzí od různých autorů.
- **Nástroje pro instrukce a podporu tutorům kurzu**. ProAuthor umožňuje autorům vytvořit nejenom instrukce a doporučení ke studiu kurzů, ale i instrukce a doporučení případným tutorům kurzů, včetně možnosti tisku studijních opor pro účastníky kurzu a tutorských příruček pro tutory kurzu.
- **Podpora standardů a LMS**. Vzhledem k tomu, že se jedná o externí autorský nástroj (samostatnou aplikaci), umožňuje ProAuthor automatický export obsahu kurzu nejen do LMS EDEN, ale i do dalších řídicích systémů, které podporují standard AICC nebo SCORM 1.2.

[3] WYSIWYG je akronym anglické věty „What you see is what you get“, česky „co vidíš, to dostaneš“. Tato zkratka označuje způsob editace dokumentů v počítači, při kterém je verze zobrazená na obrazovce vzhledově totožná s výslednou verzí dokumentu. (9)

2.1.1 UŽIVATELSKÉ PROSTŘEDÍ PROAUTHOR

ProAuthor umožňuje kurz členit do kapitol a do nich vkládat studijní články, autotesty, testy, cvičení a úkoly. Uživatelské prostředí (obrázek 4) je rozděleno na levé vertikální menu *Plán kurzu* (1), umožňující vkládat články či jiné aktivity klepnutím na příslušnou ikonu. Vytvořené aktivity lze organizovat, přesouvat, mazat v připojeném *seznamu aktivit* (2).

V pravé části se nalézá *textový editor* (3). V horní části jsou *nástroje pro formátování* (4) (odstavec, seznam, zarovnání textu, vložení dolního a horního indexu, symbolu apod.). Velmi užitečnou funkcí je kontrola pravopisu, která odhalí nechtěné překlepy, a je velmi pomalá. V levé části editoru jsou k dispozici základní styly nadpisů a odstavců (5) určené pro vytváření struktury článků. Do editoru lze vkládat i HTML kód, ale pak je nutné nastavit vybranému textu styl HTML, jinak se zobrazí tak, jak byl vložen. Místo editoru lze vložit předem vytvořenou HTML stránku neb dokument pdf (6). Pod nimi jsou *ikonky* (7) pro vkládání multimediálního obsahu a odkazu na ně pomocí čísla, jež je nutné

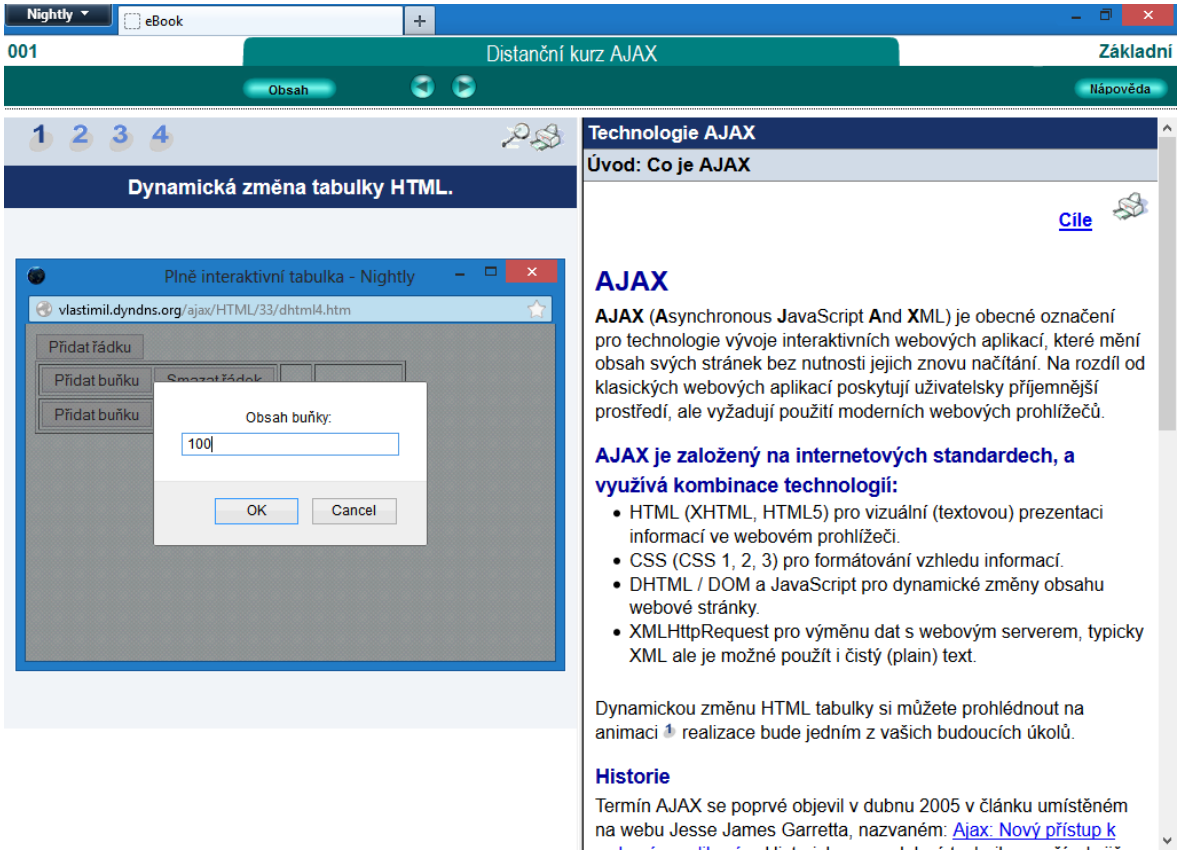


Obrázek 4: Uživatelské prostředí

vyplnit ručně. Multimediální obsah mohou být obrázky tabulky, animace, vzorce, audio video apod.

Speciálním druhem obsahu jsou komponenty *Svazky MM&S,* určené k vložení flash animace nebo HTML kódu. Multimediální komponenty jsou při exportu kurzu zobrazeny v levém rámci a některé lze připojit přímo do studijního textu. Prvky jsou organizovány do skupin. Po nahrání multimediálního obsahu je možné je připojit k článku. Připojené komponenty jsou s náhledem viditelné ve *studijním článku*. Kromě multimediálních komponent lze vkládat text nebo připojovat soubory. V dolní části se nalézá *volba rozvržení šířky animační části* (8). Ještě před vlastním exportem kurzu lze prohlédnout výstup článku pomocí náhledu umístěném hned pod rozvržením. ProAuthor obsahuje nápovědu i s animovanými postupy pro začínající uživatele. Tu lze vyvolat z horního menu, kde lze nalézt duplicitně i jiné prvky.

Výstupem ProAuthoru (obrázek 5) je HTML dokument rozdělený na tři rámce. Horní rámec umožnuje pohyb mezi články, zobrazení obsahu kurzu nebo nápovědy. V levém rámci jsou umístěné multimediální prvky jako obrázky, tabulky a animace. Pokud jich je více, lze se mezi nimi přepínat pomocí ikonek s číslem. V pravém rámci je obsah studijního článku včetně zmenšených ikonek odkazujících na příslušné multimediální prvky. Kromě zvětšení obrázku a tisku je možné zobrazit si ke studijnímu článku cíle výuky a rejstřík, pokud byly vyplněny.



Obrázek 5: Exportovaný kurz

2.1.2 TECHNICKÉ ZHODNOCENÍ NÁSTROJE PROAUTHOR

Program ProAuthor je napsaný v jazyce C#, kde pro ukládání článků používá databázi Microsoft Access. Někteří méně zkušení uživatelé mohou mít problémy s jeho instalací, neb vyžaduje platformu .NET Framework, jež nebývá součástí operačního systému Windows. Část dat (obrázky, animace atd.) je uložena na souborovém systému v nepříliš přehledné struktuře. Jsou zde k dispozici i rtf dokumenty s texty studijních článků.

Program obsahuje velké množství chyb. Tu a tam se objeví chybová hláška, která není ošetřena - nezobrazí se čitelná chyba pro uživatele. Občas je problém s odstraněním multimediálních komponent propojených s články. PA hlásí, že jsou ještě v některém článku použity, ale není tomu tak. Uživatelské rozhraní je značně složité a systém propojování multimédií je méně přehledný. I zkušený uživatel s ním může mít problémy. Flash animace a obrázky jsou při zobrazení zbytečně deformovány. V zobrazení 1:1 se ukáží až po zvětšení pomocí lupy. Nejen proto jsem použil gif animace a rozměry jsem dle doporučení vedoucího práce ponechal na nízké hodnotě 480x360 pixelů.

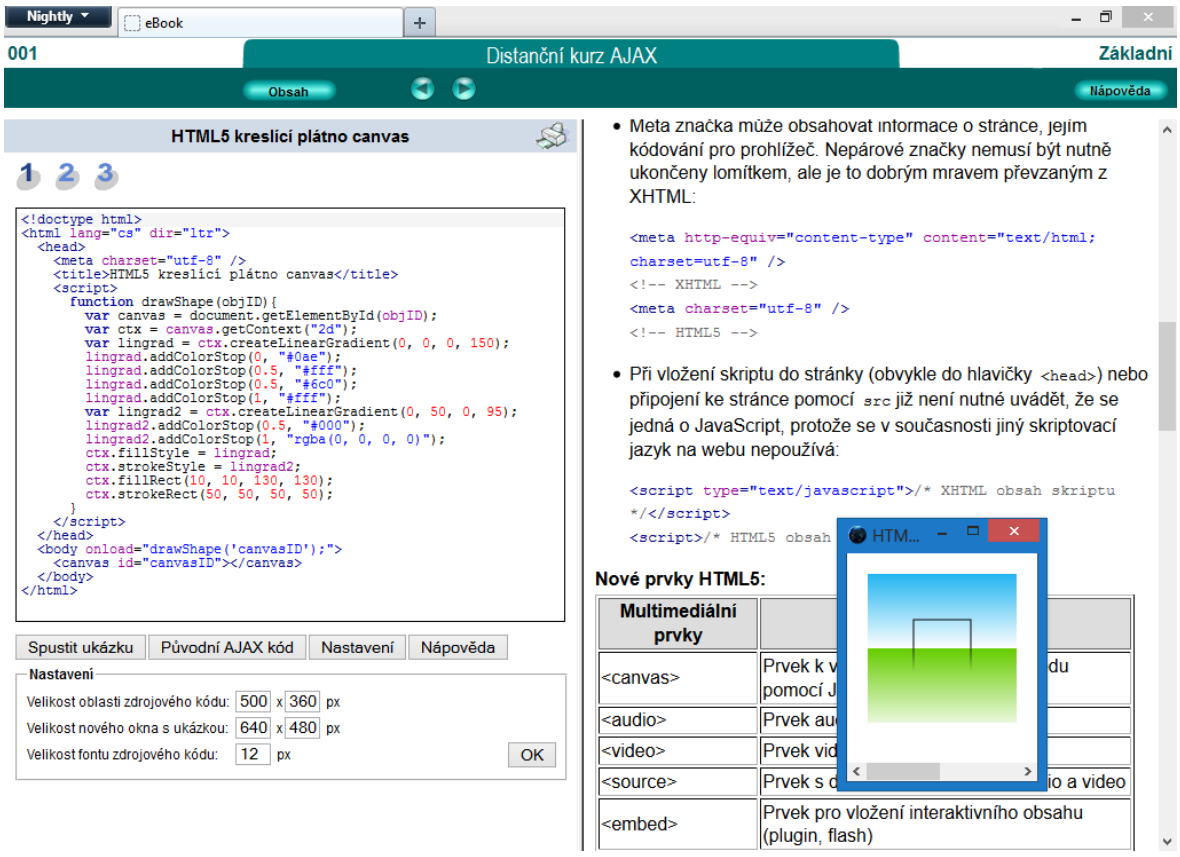
Naprosto zásadní problémy jsou v editoru studijních článků. Velmi často se stává, že nejde nastavit font textu, odrážky nebo se vyskytne nějaké jiné nepředvídatelné chování. Naštěstí se mi nestalo, že by PA zkolaboval a došlo k poškození kurzu nebo databáze. Je k dispozici ukládání do jiné složky, které jsem často používal.

Software je velmi náročný na výkon CPU a disku. Nejvíce je to poznat při exportu kurzu, kdy i na moderních počítačích trvá velmi dlouho, než se kurz publikuje. Předcházet tomu lze rozdělením kurzu na více částí a následné finální spojení a export. Výstup kurzu je ve formě HTML rámců, které se používaly naposled před 10 lety. Použití flash jde tolerovat u animací, ale pro navigační menu není příliš vhodné.

Kdyby se podařilo zmíněné nedostatky odstranit tj. vylepšit uživatelské rozhraní, HTML výstup, podporu multimédii a výkon při zpracování a exportu, byl by ProAuthor použitelnější. Nemohu hodnotit integraci do LMS či jiného systému, protože jsem ho neměl k dispozici.

2.2 SPUSTITELNÉ UKÁZKY KÓDU

Do ProAuthoru jde namísto flash animace (*Svazky MM&S*) vložit HTML stránka. Této vlastnosti jsem využil při implementaci spouštěče ukázek (obrázek 6) neboli interpreteru kódu. Spouštěč mi poskytl vedoucí diplomové práce. Po jeho analýze jsem ho nejdříve mírně upravil, nakonec došlo k jeho úplnému přepsání a přidání nových funkcí. Pro potřeby kurzu se mi jevil dodaný editor nedostatečný, a proto jsem se rozhodl nasadit moderní editor kódu zvýrazňující syntaxi. Kromě modernizace JavaScript a CSS kódu bylo přidáno nastavení, ve kterém lze zvolit velikost oblasti zdrojového kódu, okna s ukázkou a velikost fontu. Nastavení je uloženo do cookies a při dalším spuštění je načteno. Dalším vylepšením je automatické zvětšení oblasti zdrojového kódu při změně velikostí rámce. Jednou z menších změn je možnost zobrazení nápovědy. Ta se ukáže až v případě klepnutí na příslušné tlačítko *Nápověda,* stejně ji lze i schovat.



Obrázek 6: Interpreter zdrojového kódu

Spustitelné a editovatelné ukázky jsou zásadním vylepšením, o který byl vlastní autorský systém PA rozšířen. Díky tomu student ihned vidí, jak se změna ve zdrojovém kódu projeví v jeho aplikaci. Při klepnutí na *Spustit ukázku* je zdrojový kód přeložen do nového okna. Pro funkčnost některých AJAX aplikací je nutné, aby kurz běžel z webového serveru. Jelikož byla požadována i offline verze, je součástí návod, jak spustit kurz offline. Editor má i vlastní historii, a tak je možné se ve zdrojovém kódu vracet, případně jít dopředu. Další možností je návrat k původnímu kódu tlačítkem *Původní AJAX kód*. Mezi jednotlivými příklady lze přepínat tlačítky v záhlaví pod nadpisem ukázky. Na rozdíl od obrázků a animací nejsou funkční odkazy ze studijního článku, ale musí se přepínat ručně.

2.2.1 PROPOJENÍ INTERPRETERU S PROAUTHOREM

Jak již bylo naznačeno, interpreter se vkládá jako HTML stránka pomocí komponenty MM&S, u které je nutné změnit typ souboru ze swf na html. Pomocí této komponenty se vloží pouze první příklad. Název musí být `obr.htm`. Zdrojový HTML kód interpreteru obsahuje odkazy na případné další ukázky a ty je nutné připojit jako soubory, u nich je nutné zaškrtnout v PA možnost: *pouze ukládat (nevytvářet odkaz)* a připojit je ke konkrétnímu článku. Tím je dosaženo uložení všech ukázek do jedné složky. V ukázkách jsou použity odkazy a tlačítka. Aktivní ukázka má zvýrazněnou ikonu. To je zajištěno výměnou obrázku.

Původní zdrojový kód HTML / JavaScript interpreteru jsem přepsal do HTML5, odstranil jsem všechny nadbytečné prvky a pročistil stylopis CSS. Do editoru kódu jsem po prostudování dokumentace[4] implementoval pokročilý strukturní editor syntaxe CodeMirror. CodeMirror kompletně nahradí prvek `<textarea>` vlastním editorem. Pomocí jeho API lze pak získat jeho měnící se obsah. Popis a implementace interpreteru by vystačila na samostatnou diplomovou práci.

[4] http://codemirror.net

2.2.2 EDITOR CODEMIRROR

CodeMirror je JavaScript komponenta, která nabízí editor kódu v prohlížeči webových stránek. K dispozici je velké množství modulů pro různé programovací jazyky. CodeMirror obsahuje velmi rozsáhlé API a díky tomu je velmi přizpůsobitelný pro nasazení na jakékoliv řešení. Díky stylizaci CSS a podpoře grafických témat lze přizpůsobit i vizuálně. Sám jsem si vytvořil pro potřeby kurzu nový stylopis pro barevné zvýrazňování syntaxe tak, aby se co nejvíce podobala nástroji pro tvorbu webových aplikací: Adobe Dreamweaver. Komponenta je k dispozici pod otevřenou licencí (MIT), a tak v nasazení do kurzu nic nebrání, jen je nutné text licence připojit a citovat. (9)

Do kurzu jsem implementoval editor zdrojového kódu nabízející:

- Barevné zvýraznění syntaxe JavaScriptu, HTML, CSS, a XML.
- Částečnou kontrolu syntaxe kódu HTML (chyba je zvýrazněna červeně).
- Automatické doplnění závorek a uvozovek při psaní počátečních znaků: '`[`', '`(`', '`{`', '`"`', nebo '`'`'.
- Automatické uzavírání XML a HTML tagů: '`>`' nebo '`/`'.
- Našeptávač kódu HTML, stačí napsat např. `<in` a použít klávesovou zkratku Shift-Ctrl-Space a kód se automaticky doplní na `<input`.
- Našeptávač kódu JavaScript, stačí napsat `doc` a použít klávesovou zkratku Ctrl- Space a kód se automaticky doplní na document apod.
- Zvýraznění aktuálního řádku s kurzorem.
- Výběr části kódu inverzním zvýrazněním.
- Pohyb v historii psaní CTRL + Y / Z.

Všechny funkce se zapínají pomocí API a příslušných metod a je nutné vždy připojit příslušný JavaScript modul a stylopis CSS. Codemirror umožňuje zapnout číslování řádek, náhled stránky, kontrolu psaní syntaxe a mnohem více. Pro potřeby kurzu byly povoleny jen ty "nejzákladnější" funkce.

Vlastní menu pro ovládání interpreteru se vkládá pomocí JavaScriptu po načtení celé stránky. Výhodou je možnost úpravy jediného souboru. Změny se projeví ve všech ukázkových příkladech najednou. Pro tyto účely byla použita funkce ProAuthoru: *Vložení souborů - G*. Tyto soubory jsou po exportu kurzu dostupné ve složce data. V této složce je nahraný stylopis CSS pro nastavení vzhledu a JavaScripty pro obsluhu interpreteru kódu včetně CodeMirror skriptů. Vše je spuštěno metodou `init()` po načtení HTML kódu.

ZDROJOVÝ KÓD HTML

```
<!doctype html><!-- HTML 5 -->
<html lang="cs" dir="ltr"><!—informace o jazyku a směru textu -->
  <head>
    <title>Distanční kurz AJAX</title>
    <meta charset="utf-8" /><!— doporučené kódování textu -->
    <script src="../data/skript.js"></script><!-- připojení JavaScriptu -->
  </head>
  <body onload="init();"><!-- po načtení stránky se zavolá metoda init() -->
    <h1 id="header">AJAX přístup k prvkům v XML</h1><!-- nadpis ukázky -->
    <div id="links"><!—tisk kódu, odkazy na příklady s ikonami -->
      <a href=""><img onclick="printMe();" id="print" src="../img/tisk.gif"
      alt="Vytisknout ukázku" /></a>
      <a href="obr.htm"><img alt="1" src="../but/but1s.gif" /></a>
      <a href="xml2.htm"><img alt="2" src="../but/but2.gif" /></a>
      <a href="xml3.htm"><img alt="3" src="../but/but3s.gif" /></a>
      <a href="xml4.htm"><img alt="4" src="../but/but4s.gif" /></a>
    </div>
    <textarea id="code">
<!-- sourcecode ########
Mezi komentář se vloží zdrojový kód, protože některé prohlížeče by mohli mít problém
s vnořeným HTML dokumentem obsahující prvek body, head aj. Po načtení stránky je
komentář JavaScriptem nahrazen. CodeMirror nastaví prvku textarea vlastnost
display: hidden a vytvoří si prvek nový, kam přenese původní textový obsah. Textarea
neumožnuje stylizaci vnitřního textu, proto je pro zvýrazňování nepoužitelný.
######## sourcecode -->
    </textarea>
  </body>
</html>
```

VÝPIS JAVASCRIPT METOD

K HTML části interpreteru je připojen JavaScript `skript.js`. Obsahuje metody pro spouštění AJAX ukázek, nastavení oblasti zdrojového kódu, zobrazení nápovědy apod.

```
function init() { ... } // inicializace cookies, uložení původního kódu
function helpMe() { ... } // zobrazení nápovědy
function settings() { ... } // uložení výchozích cookies
function saveSettings() { ... } // uložení nastavení do cookies
function setup(cols, rows, fntsize) { // nastavení velikosti oken dle parametru
function createCookie(name, value, days) { ... } // copyright W3C cookies
function getCookie(c_name) { ... } // copyright W3C cookies
function showNew() { ... } // zobrazení zdrojového kódu do nového okna
function showOrigin() { ... } // návrat k předchozímu kódu
function printMe() { ... } // tisk zdrojového kódu
function printWbutton() { ... } // po dobu tisku skryje tlačítko tisku ze stránky
```

Kompletní zdrojové kódy HTML, JavaScriptu a stylopisu CSS jsou součástí dokumentace k distančnímu kurzu na CD.

2.2.3 KONVENCE PSANÍ ZDROJOVÝCH KÓDŮ

Zdrojové kódy a názvy proměnných jsou psané v anglickém jazyce. Čeština je použita většinou jen k interakci s uživatelem. Pokud se někdy dostanete do praxe, rychle si na to zvyknete. Názvy proměnných a metod jsou psané dle Java / JavaScript konvence[5] oddělené velkými písmeny např.: `document.getElementById(divID).` Vlastní kód je strukturován pomocí dvou mezer a není použit tabulátor. Pokud to příliš neztíží pochopení příkladů, jsou funkce a jejich vlastnosti vnořené do sebe bez mezi proměnných. V JavaScriptu jsou používány mezery v přiřazení pro zvýšení čitelnosti a přehlednosti kódu. Samozřejmostí je mezera za čárkou k oddělení parametrů. V kódu HTML jsem se rozhodl žádné mezery nepoužívat ani u událostí připojených k prvkům. Nebývá to zvykem a některým prohlížečům to způsobuje problémy. Obecně se dá říci, že co kodér, to jiný styl formátování. Výborná doporučení jak psát a strukturovat kódy[6] lze nalézt opět u Googlu.

Někteří kodéři vynechávají odsazení u prvků `head` a `body`. Pokud v kódu nejdou použít uvozovky (`""`), jsou nahrazeny apostrofy (`''`). Na vhodných místech jsou v textu, výjimečně i ve zdrojovém kódu použity `// řádkové komentáře` nebo `/* blokové komentáře */`. V textu hovořím o funkcích a metodách. JavaScript tyto pojmy nerozlišuje a lze s výhradami prohlásit, že funkce ~ metoda. Na rozdíl od funkce je metoda navázána na objekt. Funkce by měla vždy vracet nějakou hodnotu: `return(a + b)` a následně voláním předána např.: `var c = spocitej(1, 2).` Je dobrým zvykem psát před každou proměnnou klíčové slovo `var a = 5`, jinak ji prohlížeč může považovat za globální. Je možné, že se někde v kurzu klíčové slovo "zatoulalo".

[5] http://google-styleguide.googlecode.com/svn/trunk/javascriptguide.xml
[6] http://google-styleguide.googlecode.com/svn/trunk/htmlcssguide.xml

Zdrojové kódy a názvy prvků, metod jsou zvýrazněny jiným fontem, a to i uvnitř článků. Pro zvýraznění zdrojových kódů uvnitř článků jsem použil projekt google-code-prettify[7]. Prettify je JavaScript modul, který umožňuje zvýraznit zdrojový kód v HTML stránce. Při prvním použití na stránce je třeba připojit zdrojový kód se skriptem, případně CSS stylem. CSS styl jsem upravil takovým způsobem, aby byl co nejvíce barevným zvýrazněním podobný editoru CodeMirror.

```
<script src="../data/run_prettify.js"></script>
```

Zvýrazněny budou všechny zdrojové kódy uvedené mezi tagy `<pre>` s CSS třídou `prettyprint`. Pro správné zobrazení HTML prvků, je nutné nahradit závorky `<>` HTML entitami **`&lt;`** a **`&gt;`**. Google-code-prettify lze volně použít dle Apache Licence 2.0 [8].

```
<pre class="prettyprint" style="border: 0px; white-space: pre-wrap;">
&lt;script&gt;
  function alertMe(txt) {
    alert(txt);
  }
&lt;/script&gt;
</pre>
```

Oblasti, kterou chceme zvýraznit, se v PA nastaví styl HTML. Ta se po exportu a spuštění v prohlížeči automaticky zvýrazní. Skript se uloží do globální složky data pomocí funkce *PA Soubory - G*. Google-code-prettify je na použití podstatně jednodušší než CodeMirror, ten by vyžadoval rozsáhlejší úpravy.

[7] http://code.google.com/p/google-code-prettify
[8] http://www.apache.org/licenses/LICENSE-2.0

3 TVORBA AJAX APLIKACÍ

Zde je popsán stručný obsah studijních článků, zadání příkladů, částí zdrojových kódů nebo kompletních ukázek.

3.1 DHTML

DHTML neboli dynamické HTML je kombinace technologií používaných k tvorbě dynamických a interaktivních webových stránek. Těmito technologiemi se většinou myslí HTML, klientský JavaScript, kaskádové styly (CSS) a DOM. DHTML nabízí daleko více pokročilých možností než pouhé stylování pomocí CSS. DHTML bylo v oblasti webových stránek používáno dříve, než se začal používat termín AJAX, a před vlastní standardizací DOM.

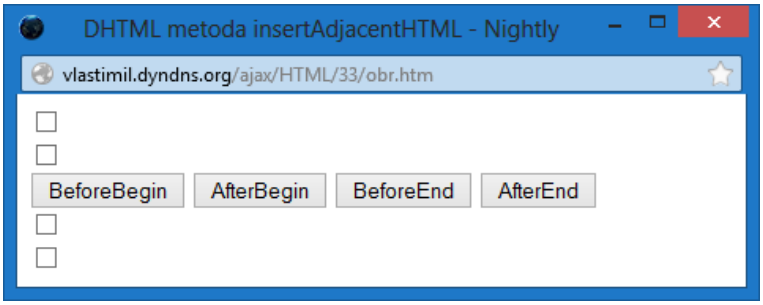
Jsou 2 základní cesty jak modifikovat HTML webovou stránku pomocí metod nebo vlastností. Mezi základní metody DHTML patří:

- `InsertAdjacentHTML()` - přidá HTML vedle již existujícího prvku.
- `InsertAdjacentText()` - přidá text vedle již existujícího prvku.

Metodám lze nastavit, kam mají obsah umisťovat `"BeforeBegin"`, `"AfterBegin"`, `"BeforeEnd"`, nebo `"AfterEnd"`, český význam snad není potřeba dodávat. U těchto 2 metod není zaručená kompatibilita napříč prohlížeči. Např. ve Firefoxu metoda `InsertAdjacentText()` nefunguje a konzole nahlásí neznámou metodu.

V ukázce si studenti mohou vyzkoušet vkládat `checkbox` buttony okolo tlačítek ve formuláři. Do proměnné `place` se předá umístění např.: `"BeforeEnd"`. Tlačítka jsou obaleny pomocí prvku `<div id="divID">`.

```
function addHTML(place) {
  divID.insertAdjacentHTML(place, "<input type=checkbox value='radio' />");
  // dle umístění place se kolem prvku div vloží checkbox buttony.
}
```



Obrázek 7: Metoda InsertAdjacentHTML

3.1.1 AKTUALIZACE WEBOVÝCH STRÁNEK POMOCÍ DHTML VLASTNOSTÍ

DHTML postupně zastarává, proto dnes již všechny vlastnosti používané v minulosti nejsou současnými prohlížeči podporovány:

Vlastnost	Popis
innerText	Umožňuje změnit text mezi počátečním a konečným prvkem (překonáno)
outerText	Umožňuje změnit text včetně prvků (překonáno)
innerHTML	Umožňuje změnit obsah prvku mezi počátečním a koncovým tagem
outerHTML	Umožňuje změnit obsah prvku mezi počátečním a koncovým tagem, text je vynucen jako HTML

Doporučené jsou dvě poslední standardizované vlastnosti `innerHTML` a `outerHTML`. Protože `innerHTML` již bylo probráno v úvodní kapitole, do kurzu byla zařazena ukázka s metodou innerText.

3.1.2 VYTVÁŘENÍ NOVÝCH PRVKŮ POMOCÍ DHTML

AJAX aplikace často potřebují vytvořit nové HTML prvky za chodu, protože předem nevíme, jak budou vstupní data rozsáhlá. Dobrým příkladem může být tabulka, seznam nebo interaktivní formulář reagující na vstupy uživatele. Nicméně není problém interaktivně dle potřeby přidávat nová tlačítka, obrázky či jiné prvky. Vytvoření nového prvku se realizuje metodou `createElement()` a jako parametr se jí předá tag prvku, který chceme vytvořit. Dále lze použít metodu `createTextNode()` pro přidání textového popisu. Po založení prvku se mu nastaví požadované vlastnosti, hodnoty a události `onclick`. Nastavené prvky se připojí do rodičovského prvku a následně pak do dokumentu metodou `insertBefore()`. Podrobněji je tato problematika vysvětlena a rozšířena v kapitole Document Object Model. Ve studijním článku jsou, pospány metody jak interaktivně přidávat prvky formulářů na HTML stránku.

3.1.3 PRÁCE S TABULKAMI V DHTML

V dynamickém HTML, je obsažen objekt `table`, který umožňuje manipulovat s odpovídajícím prvkem `<table>` v HTML dokumentu. Objekt má metody a vlastnosti pro manipulaci s obsahem:

Metody a vlastnosti	Popis
tableObject.rows(index)	Obsahuje pole (kolekci) řádků v tabulce
tableObject.insertRow(index)	Vloží novou řádku, pokud není zadán index, vloží se na konec
tableObject.deleteRow(index)	Smaže řádku, index určuje pozici řádky, která se má smazat
tableObject.cells(index)	Obsahuje index buňky v řádku
tableObject.row(index)	Obsahuje index řádky
tableObject.insertCell(index)	Vloží novou buňku, pokud není zadán index, vloží se na konec řádky
tableObject.deleteCell(index)	Smaže buňku, index určuje pozici v poli, která se má smazat

3.1.4 VEPSÁNÍ JINÉ STRÁNKY DO DOKUMENTU

Jednou ze zajímavých DHTML metod je `document.write()`, která umožňuje vepsat obsah jiné stránky nebo části do současného dokumentu pomocí JavaScriptu. Na rozdíl od dříve popsaných metod, tato nepracuje s malými částmi dokumentu (`text, value`). Využití v AJAX aplikacích bývá různé, dokonce může HTML kód obsahovat pouze tělo body a v závislosti na nějaké podmínce v JavaScriptu je zobrazena příslušná stránka. V odborné literatuře (6) je zmíněn např. jídelní lístek, který se mění v závislosti na denní době. V kurzu je použito toto řešení pro připojení více externích JavaScriptů v interpreteru. Stačí tedy odkázat pouze jeden soubor, ve kterém budou touto metodou vloženy další.

- `document.write("<h1>Nadpis</h1>")` - pokud není script mezi konkrétními tagy, vloží se HTML kód do těla `body`. Prvky hlavičky `head` tj. styly, skripty nebo meta značky nemají na tělo vliv.
- `document.writeln("Prostý text")` - navíc odřádkuje, lze také použít `\n` jako znak nové řádky v textu.

Je třeba si dát pozor na případ, kdy je metoda zavolána ve funkci. Dojde k odstranění všech prvků v těle dokumentu `body`. Prvky vepsané pomocí `document.write()` se nepřepíší.

Všechny uvedené metody a vlastnosti si studenti vyzkouší na 4 praktických příkladech. Následuje závěrečný hodnocený příklad:

NAVRHNĚTE APLIKACI, KTERÁ VYUŽÍVÁ DHTML K VYTVOŘENÍ INTERAKTIVNÍ TABULKY:

- Do prázdné tabulky půjdou přidat nové řádky pomocí tlačítka.
- Do každé řádky půjdou vložit nové buňky, před jejich vložením bude uživatel vyzván k vyplnění jejich textové hodnoty.
- Libovolné řádky půjdou smazat, můžete vyžadovat i potvrzení, zda má dojít k vymazání.
- Po klepnutí na libovolnou buňku bude uživatel vyzván k opravě textové hodnoty v dané buňce, ta se ihned projeví.
- Vypravovaný úkol odešlete k hodnocení.

ŘEŠENÍ:

```
<!doctype html>
<html lang="cs" dir="ltr">
  <head>
   <title>Plně interaktivní tabulka</title>
     <script>
       function createRow(objID) { // metoda pro přidání nové řádky
         var row = document.getElementById(objID).insertRow(0); // založí řádku
         var str1 = "<input type = 'button'
         value = 'Přidat buňku' onclick = 'addCell(this);' />";
         /* připraví tlačítko pro přidání buňky */
         var str2 = "<input type = 'button'
         value = 'Smazat řádek' onclick = 'removeRow(this);' />";
         /* připraví tlačítko pro odebrání buňky */
         row.insertCell().innerHTML = str1 + str2; // vložení objektu do <table>
       }
       function addCell(obj) { // metoda pro přidání nové buňky
         var cell = obj.parentNode.parentNode.insertCell(1); // odkaz na předka
         cell.innerHTML = prompt("Obsah buňky: ", "*"); // dotaz na hodnotu
         cell.addEventListener("click", function(){ replaceText(this); });
         /* připojí událost po kliknutí, spustí obecnou funkci s parametrem */
       }
       function removeRow(obj) { // metoda pro smazání řádky
         obj.parentNode.parentNode.parentNode.removeChild
         (obj.parentNode.parentNode); // odkaz na předka
       }
       function replaceText(obj) { // metoda pro nahrazení obsahu buňky
         obj.innerHTML = prompt("Obsah buňky: ", obj.innerHTML);
       }
     </script>
  </head>
  <body>
    <form>
      <input type="button" value="Přidat řádku"
      onclick="createRow('tableID');" />
    </form>
    <table id="tableID" border="1"></table>
  </body>
</body>
```

V příkladu jsou použity velmi pokročilé metody, ukazatel `this` události `EventListener`, dědičnost. Všechny tyto metody jsou součástí praktických cvičení a jsou uvedeny i v *Tipech na řešení příkladů*. Použití ukazatele `this` je velmi vhodné.

3.2 XMLHTTPREQUEST OBJEKT

V úvodu byl vysvětlen princip `XMLHttpRequest` objektu a bylo upozorněno na rozdílnou implementaci napříč prohlížeči. V budoucnosti se snad vývojáři domluví na jednotném standardu a metodách. Situace se dosti zlepšila s vydáním IE7, ale přesto se někde ještě používá IE6. Je třeba zdůraznit, že na profesionální úrovni je nutné ošetřit všechny výjimky, které mohou při běhu AJAX aplikací vzniknout. V případě, kdy nelze `XMLHttpRequest` objekt založit, je dobré o tom uživatele informovat. V opačném případě se mu zobrazí jen prázdná stránka nebo nesrozumitelná chyba.

U OBJEKTU JE TŘEBA ROZLIŠOVAT METODY:

Metoda	Popis
abort()	Přeruší HTTP request
getAllResponseHeaders()	Vrátí všechny HTTP hlavičky
getResponseHeader()	Vrátí hodnotu hlavičky
open()	Otevře požadavek na server
send()	Pošle požadavek na server
setRequestHeader()	Nastaví formát a hodnotu hlavičky

A VLASTNOSTI, KTERÉ OBSAHUJÍ HODNOTY:

Vlastnost	Popis
channel	Obsahuje kanál, přes který komunikace probíhá
readyState	Obsahuje kód v jakém stavu požadavku
responseText	Obsahuje odezvu ve formě textu
responseXML	Obsahuje odezvu ve formě XML
status	Obsahuje návratový kód požadavku
statusText	Obsahuje HTTP odezvu na požadavek ve formě textu

Jak už bylo v úvodu zmíněno, pro zaslání požadavku na server se používá metoda `open()`. Metodě lze nastavit dodatečné nepovinné parametry, které se mohou v určitých případech hodit

Vlastnost	Popis
method	typ požadavku na server, dle literatury může být GET, POST, PUT, HEAD, PROPFIND
URL	požadovaná URL adresa
asyncFlag	hodnota bool určuje, zda se použije asynchronní (výchozí) nebo synchronní požadavek
userName	uživatelské jméno při poslání požadavku na zabezpečenou stránku
password	heslo při poslání požadavku na zabezpečenou stránku

Použití `asyncFlag = false` není doporučováno, ale má své specifické použití. Pro malé požadavky nebo ukázky s ním nebude problém, JavaScript nebude pokračovat v provádění kódu, dokud nedostane odpověď ze serveru. Pokud je server pomalý aplikace se zasekne nebo zastaví.

AJAX je primárně asynchronní, to znamená, že může provádět více operací na pozadí. Během stahování prohlížeč může interagovat s uživatelem, zatím co probíhá stahování dat. AJAX má metodu jak informovat prohlížeč, že data jsou již připravena. Této metodě se odborně říká `callback()`. Ten již byl použit v několika předchozích příkladech, jen to na první pohled tak nevypadá. Místo metody `callback()` byla ihned definována obecná funkce, která se spustila automaticky po změně stavu:

```
XMLHttpRequestObject.onreadystatechange = function() { ... }
```

3.2.1 READYSTATE A STATUS

Uvnitř funkce musíme zkontrolovat, jestli byla data opravdu stažena a server nevrátil chybný stav. K tomu se používají vlastnosti `XMLHttpRequest` objektu: `readyState` a `status`. Status může nabývat následujících hodnot, nejčastěji se kontroluje na hodnotu 200:

Kód	Popis
200	OK
201	Vytvořeno
204	Bez obsahu
205	Reset obsahu
206	Částečný obsah
400	Špatný požadavek
401	neautorizováno
403	Zakázáno
404	Nenalezeno
405	Nepovolená metoda
406	Neakceptováno
407	Chyba autentizace proxy
408	Timeout požadavku
411	Požadována délka není specifikována
413	Požadavek je příliš dlouhý
413	Požadovaná URL je příliš dlouhá
415	Nepodporovaný typ souboru
500	Interní chyba serveru
501	Neimplementováno
502	Chybná brána
503	Služba nedostupná
504	Timeout brány
505	Nepodporovaná verze protokolu http

ReadyState může nabývat následujících stavů, přičemž nejčastější je kontrola na hodnotu 4:

Kód	Popis
0	Neinicializováno
1	Načítá se
2	Načteno
3	Vyžaduje interakci (heslo apod.)
4	Dokončeno

OŠETŘENÍ ASYNCHRONNÍHO DOTAZU

Ošetření dotazu si studenti vyzkouší ve cvičení, jejich výstupem by měl být skript funkční i ve starším prohlížeči (IE5, 6) s přídatným výpisem neočekávaného stavu. Dále se již v ukázkách kódu bude předpokládat použití moderních prohlížečů a možné chybné stavy nebudou plně ošetřovány.

VZOROVÉ ŘEŠENÍ

```
<!doctype html>
<html lang="cs" dir="ltr">
  <head>
    <script>
      function getData(dataSource, objID) {
        var XMLHttpRequestObject = false; // lze teoreticky vynechat
        if (window.XMLHttpRequest) { // moderní browser
          XMLHttpRequestObject = new XMLHttpRequest();
        } else if (window.ActiveXObject) { // IE5, 6
          XMLHttpRequestObject = new ActiveXObject("Microsoft.XMLHTTP");
        } else { // nepodařilo se založit objekt vůbec
          document.getElementById(objID).innerHTML =
          "Chyba: Váš prohlížeč nepodporuje AJAX";
          return 0; // ukončí metodu getData()
        }
        XMLHttpRequestObject.open ("GET", dataSource); // konfigurace dotazu
        XMLHttpRequestObject.onreadystatechange = function(){ // callback
          if (XMLHttpRequestObject.readyState == 4 /* kontrola readyState */
          && XMLHttpRequestObject.status == 200) { /* kontrola status */
            document.getElementById(objID).innerHTML
            = XMLHttpRequestObject.responseText; // nastaví výstupní text
          } else { // nastaví výstupní text v případě chyby
            document.getElementById (objID).innerHTML
            = "Při stahování dat nastala chyba, server vrátil status: "
            + XMLHttpRequestObject.status; // připojí nestandartní status
          }
        }
        XMLHttpRequestObject.send(); // odeslání asynchronního dotaz
      }
    </script>
  </head>
  <body>
    <form>
      <input type="button" value="Stáhnout"
      onclick="getData('data.txt','divID')" />
    </form>
    <div id="divID"></div>
  </body>
</html>
```

V hodnoceném příkladu mají studenti za úkol napsat aplikaci pro vizualizaci kurzů měn cnb[9]. Výstupem má být HTML tabulka. Měny jsou odděleny pomocí znaku „|". Pro parsování dat je v tipech doporučená metoda `split()`.

3.3 KOMUNIKACE SE SERVEREM

Komunikace se serverem je základ AJAX aplikací. Aby bylo možné reagovat na požadavky aplikace, je třeba umět programovací jazyk používaný na serverech. Dnes se nejčastěji používá jazyk PHP (**P**ersonal **H**ome **P**age od The PHP Group) nebo ASP (**A**ctive **S**erver **P**ages od Microsoft).

Pro funkčnost ukázek a příkladů v kurzu se předpokládá spuštění ze serveru s podporou PHP. Návod jak spustit kurz offline je též součástí.

3.3.1 ZASLÁNÍ DAT METODOU GET

Metoda `GET` umožňuje zaslat serveru text v URL adrese. Pokud chceme zaslat serveru určitý parametr, připojíme ho na konec URL:

```
script.php?parameter1=hodnota
```

Důležité je jméno parametru uvedené za otazníkem, které lze potom zachytit a zpracovat pomocí PHP. V případě, kdy je třeba zaslat parametrů více, oddělují se tyto pomocí `&`, doporučeno je používat `& amp;` (bez mezery jinak je "požráno").

```
http://www.server.com/script.php?firstname=Ferda&amp;lastname=Mravenec
```

Pokud by bylo nutné zaslat celé jméno a příjmení, používá se `+`, mezera není přípustná.

```
http://www.server.com/script.php?fullname=Ferda+Mravenec
```

Parametry jsou přímo viditelné v adresní řádce prohlížeče, což je nepraktické z důvodu bezpečnosti nebo přehlednosti, kdy je parametrů více. Pro tyto případy je lepší použít metodu `POST`.

[9] http://www.cnb.cz/cs/financni_trhy/devizovy_trh/kurzy_devizoveho_trhu/denni_kurz.txt

3.3.2 ZASLÁNÍ DAT METODOU POST

Metoda GET není jedinou možností, jak zaslat data na server. Oproti GETu je POST více zabezpečená (data nejsou v URL) a používá se k zasílání formulářových dat. Zdrojový kód umístěný na serveru je velice obdobný, jen se místo GET použije POST:

```
HttpRequestObject.setRequestHeader('content-type','application/x-www-form-urlencoded');
```

Parametry zasílané POSTem se na rozdíl od GETu, vkládají jako parametr do metody send, připomenu, že GET předává data pomocí URL, tento parametr je tedy null.

```
XMLHttpRequestObject.send("parametr=hodnota")
```

V praktickém cvičení si studenti vyzkouší zaslání dat metodami GET a POST. Cvičení je zadáno:

Vytvořte aplikaci, která bude obsahovat:

- Formulář se vstupními editovatelnými poli Jméno a Příjmení.
- Dvě tlačítka GET a POST.
- Po stisknutí tlačítka GET se data odešlou metodou GET, obdobně při stisknutí POST se odešlou metodou POST.
- Použijte pouze jednu metodu (funkci) getData(), které předejte způsob odeslání.
- Formulář dle metody zašlete na skript get.php respektive post.php.
- Pro testování použijte interpreter v předchozím studijním článku.

PHP SKRIPTY

Pro realizaci jsou připraveny 2 PHP skripty, jeden s metodou GET, druhý s POST:

```
<?php
  echo ("firstname: ".$_GET["firstname"]."<br />");
  echo ("lastname: ".$_GET["lastname"]."<br />");
?>
```

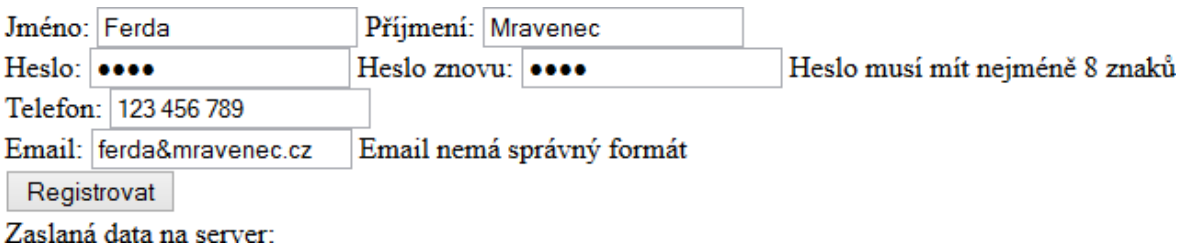
```
<?php
  echo ("firstname: ".$_POST["firstname"]."<br />");
  echo ("lastname: ".$_POST["lastname"]."<br />");
?>
```

3.3.3 ŘEŠENÍ CVIČENÍ METODA GET A POST

```
<!doctype html>
<html lang="cs" dir="ltr">
  <head>
    <script>
      function getData(dataSource, objID, method) {
        /* načte data z vstupních dat formulářů */
        var firstname = document.forms[0].elements["firstname"].value;
        var lastname = document.forms[0].elements["lastname"].value;
        if (method == 'GET') { /* v případě GETU, nastaví předání dat v URL */
          var dataSource = "get.php?firstname=" + firstname
          + "&lastname=" + lastname;
        }
        var XMLHttpRequestObject = new XMLHttpRequest(); // založí objekt
        XMLHttpRequestObject.open(method, dataSource, false); /* předá metodu
        XMLHttpRequestObject.setRequestHeader('content-type','application/x-
        www-form-urlencoded'); // pro POST přidá formát odesílaných dat
        XMLHttpRequestObject.send("firstname=" + firstname
        + "&lastname=" + lastname); // zašle data, metoda bude přetížená
        document.getElementById(objID).innerHTML
        = XMLHttpRequestObject.responseText; // odpověď přidá do dokumentu
      }
    </script>
  </head
  <body>
    <form>
      <label>Jméno: </label>
      <input type="text" id="firstname" />
      <label>Příjmení: </label>
      <input type="text" id="lastname" />
      <input type="button" value="GET"
      onclick="getData('null','divID','GET')" />
      <input type= "button" value="POST"
      onclick="getData('post.php','divID','POST')" />
    </form>
    <div id="divID"></div>
  </body>
</html>
```

HODNOCENÝ ÚKOL

Plné výhody AJAXu si studenti vyzkouší v hodnoceném příkladu, kde mají za úkol vytvořit registrační formulář s „nacionálemi“. Údaje se kontrolují před odesláním na server. V případě problémů se vypíše u chybného údaje zpráva (obrázek 8).



Obrázek 8: Registrační formulář

3.4 XML

Zatím jsme se zabývali pouze stahováním a posíláním textových dat bez jakékoliv struktury. Jak už ale z názvu Asynchronní JavaScript a XML vyplývá, často se používá formát XML. V úvodní kapitole byly vysvětleny klíčové principy a uvedena krátká ukázka. Tvorba struktury XML dokumentu je velice podobná vytváření HTML kódu s tím, že prvky si pojmenováváme dle vlastního uvážení

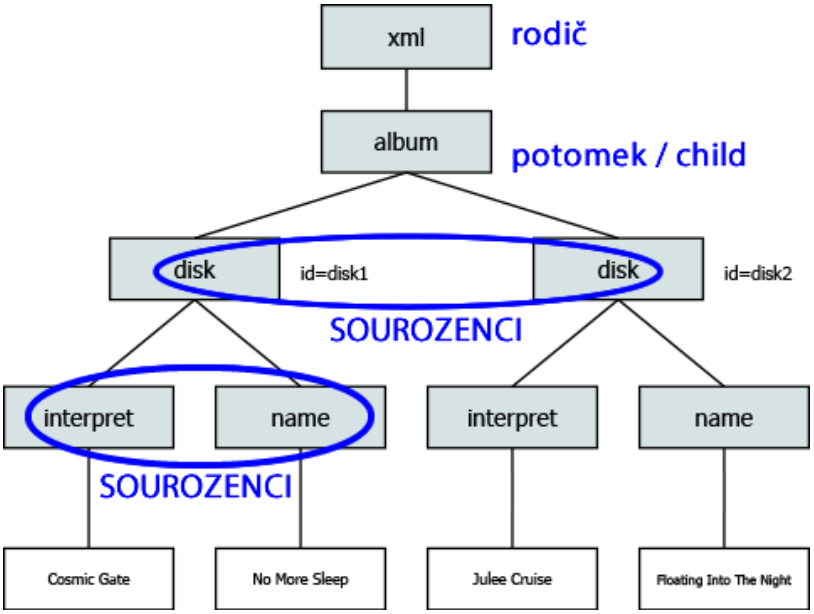
3.4.1 XML JAKO STROM PRVKŮ

Zatímco HTML je potomkem SGML, XHTML je potomkem XML. Pokud si například chceme vést seznam hudebních nosičů, interpretů, přestane nám HTML vyhovovat, nemá prvek `<album>`, `<interpret>` apod. Velice obtížně by se taková data pomocí HTML zpracovávala. Správně strukturované XML tento problém řeší (databáze ovšem také). Software (v našem případě internetový prohlížeč), který XML čte, si sestaví v paměti strom (objekt) odpovídající XML datům, který odráží vzájemné vztahy prvků. Poté lze s takovým stromem dobře manipulovat.

PŘÍKLAD TAKOVÉTO HUDEBNÍ KOLEKCE

```
<?xml version="1.0" ?>
<albums>
  <disk id="disk1">
    <interpret>Cosmic Gate</interpret>
    <name>No More Sleep</name>
  </disk>
  <disk id="disk2">
    <interpret>Julee Cruise</interpret>
    <name>Floating Into The Night</name>
  </disk>
</albums>
```

Vizualizace stromové struktury je na obrázku 9, rodičovský uzel album má dva sourozence `disk`, které jsou pro případ identifikace doplněny o atribut `id`. Tyto uzly (v angličtině node, nody) jsou na stejné úrovni a nazývají se sourozenci (siblings). Můžeme, říčí, že v celé struktuře stromu se používá dědičnost. JavaScript obsahuje vestavěné metody jak přistupovat k jednotlivým uzlům a atributům. Existují textové uzly, uzly s atributy atd. Kompletní seznam je uveden v tabulce, nejčastěji se pracuje s prvními třemi.



Obrázek 9: Stromová struktura XML

Číslo	Typ uzlu
1	Uzel prvků
2	Uzel atributů
3	Uzel textů
4	CDATA speciální data
5	Odkaz na XML entitu uzlů
6	XML entita uzlů
7	XML procesní uzly
8	XML komentační uzly
9	XML dokumentační uzly
10	XML uzly DTD
11	XML uzly částí dokumentu

Daleko častěji se využívá vlastností, ty jsou uvedeny v tabulce:

Vlastnost	Význam
attributes	Atributy uzlů
childNodes	Pole uzlů sourozenců
documentElement	Odkaz na prvek
firstChild	První uzel sourozence
lastChild	Poslední uzel sourozence
localName	Lokální jméno sourozence
name	Jméno uzlu
nextSibling	Následující uzel
nodeName	Jméno uzlu
nodeType	Typ uzlu
nodeValue	Hodnota uzlu
previousSibling	Předchozí uzel

3.4.2 AJAX A XML

XML struktura se nijak zvlášť neliší od HTML, vycházejí vlastně ze stejného předchůdce. Každý XML dokument musí mít povinně v úvodu deklaraci, parametr kódování je volitelný, při neuvedení je obvykle považován za utf-8.

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

Dále již může dokument obsahovat strukturovaná data, názvy tagů avšak nesmí obsahovat mezery a některé znaky jsou nepřípustné. V kurzu je pro potřeby příkladů umístěné jednoduché XML obsahující seznam barev:

```
<?xml version="1.0"?>
<colors>
  <color>red</color>
  <color>green</color>
  <color>blue</color>
</colors>
```

Požadavek na stažení dat se vytváří podobným způsobem jako požadavek na text:

```
XMLHttpRequestObject.responseText / XMLHttpRequestObject.responseXML
```

Výstupem však není textová hodnota, ale objekt obsahující všechny prvky XML dokumentu. Pro jejich získání se používají obvykle 3 řešení:

- Přístup k prvkům pomocí struktury.
- Přímý přístup k prvkům.
- XPath.

PŘÍMÝ PŘÍSTUP K PRVKŮM

Přístup k prvkům pomocí struktury si studenti vyzkouší na hodnoceném příkladu s auty, jedná se o rozšíření úvodního příkladu o práci s atributy, podle nich je možné rozlišit osobní auta od nákladních. XPath je jazyk pro vyhledávání informací v XML dokumentech. Jeho popsání přesahuje rámec kurzu AJAX. Nejčastěji je používaný přímý přístup k prvkům, díky své jednoduchosti. V praktickém cvičení mají studenti za úkol zobrazit seznam barev specifikovaný v XML dokumentu colors.xml. Podle barvy následně obarvit výstupní prvek, podle toho jestli zvolili tabulku nebo seznam.

Studenti mají v praktickém úkolu zadáno:

Doplňte příklad 1 uvedený v minulé kapitole o zpracování získaných XML dat:

- Soubor colors.xml a kostra programu je připravena.
- Výstup zpracujte formou HTML seznamu nebo tabulky.
- Podle barvy uložené v XML obarvěte text (buňku) v seznamu.

ŘEŠENÍ: VYPSÁNÍ A STYLIZACE BAREV Z XML

```
<!doctype html>
<html lang="cs" dir="ltr">
  <head>
    <script>
      function getData(dataSource, objID) {
        var XMLHttpRequestObject = new XMLHttpRequest();
          XMLHttpRequestObject.open("GET", dataSource);
          XMLHttpRequestObject.send(null);
          XMLHttpRequestObject.onreadystatechange = function() {
          if (XMLHttpRequestObject.readyState == 4 &&
          XMLHttpRequestObject.status == 200) {
            /* požadavek je stejný jako u textu, jen se změní zápis na XML */
            var colors =
            XMLHttpRequestObject.responseXML.getElementsByTagName("color");
            /* přímé získání všech prvků color a uložení do pole colors */
            for (var i = 0; i < colors.length; i++) { // v cyklu se projde pole
              document.getElementById(objID).innerHTML += "<li>" +
              colors[i].firstChild.nodeValue + "</li>";
              /* výstupy hodnot nodeValue všech sourozenců firstChild se při
              každém průchodu připojuje k prvku ul, je použit operátor += */
              document.getElementsByTagName('li')[i].style.color =
              colors[i].firstChild. nodeValue; /* pro nastavení barvy se využije
              znalost CSS / JavaScriptu pro nastavení barvy. */
            }
          }
        }
      }
    </script>
  </head>
  <body onload="getData('colors.xml','ulID')" >
    <ul id="ulID"></ul>
  </body>
</html>
```

Pozn.: Kromě vypisování lze po stažení XML a načtení objektu hodnoty i měnit, použije se přiřazení:

```
colors[1].firstChild.nodeValue = "black";
```

3.5 DOCUMENT OBJECT MODEL (DOM)

Na webové stránky nahlíží prohlížeče jako na kolekci objektů (**D**ocument **O**bject **M**odel), který umožňuje přistupovat ke všem prvkům na webové stránce. Ve většině příkladů se vytvářel obsah nový vkládáním respektive nahrazením obsahu prvků `<div></div>` nebo jiných , případně formulářů jejich hodnoty `value`.

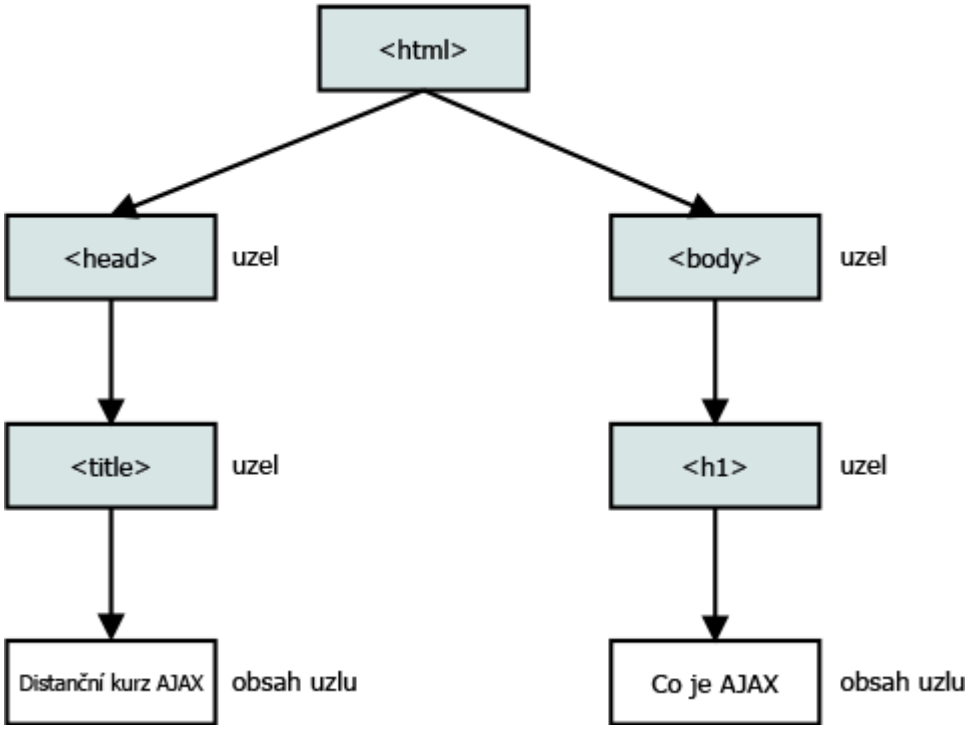
```
document.getElementById(divID).innerHTML = XMLHttpRequestObject.responseText
document.getElementById(divID).value = XMLHttpRequestObject.responseText
```

Kromě tohoto základního přístupu, byly ještě představeny možnosti přidávání obsahu pomocí DHTML metod `insertAdjacentHTML()` a `createElement()`.

3.5.1 HTML JAKO STROM OBJEKTŮ

Stejně jako na XML lze nahlížet na HTML jako kolekci objektů a jejich uzlů ve stromové struktuře. Stromová reprezentace je graficky znázorněna na obrázku 10.

```
<html>
  <head>
    <title>Distanční kurz AJAX</title>
  </head>
  <body>
    <h1>Co je AJAX</h1>
  </body>
</html>
```



Obrázek 10: DOM stromová reprezentace

JavaScript obsahuje vestavěné vlastnosti uzlů (prvků), se kterými lze stejně jako s XML pracovat:

Vlastnost	Význam
attributes	Atributy prvků
childNodes	Pole prvků sourozenců
documentElement	Odkaz na prvek
firstChild	První prvek sourozence
lastChild	Poslední prvek sourozence
localName	Lokální jméno sourozence
name	Jméno prvku
nextSibling	Následující prvek
nodeName	Jméno prvku
nodeType	Typ prvku
nodeValue	Hodnota prvku
previousSibling	Předchozí prvek

DOM pro přístup k prvkům má o něco méně vlastností oproti XML a rozlišuje:

- Prvky (jiné označení: uzly, elementy)
- Atributy
- Textové uzly

Kromě vlastností jsou dostupné JavaScript metody:

Metoda	Popis
replaceNode(node1, node2);	Nahradí prvek 2, prvkem 1
insertBefore(node1, node2);	Vloží prvek 1 před prvek 2
appendChild(node);	Připojí potomka do zadaného prvku

3.5.2 MANIPULACE S PRVKY S VYUŽITÍM DOM

PŘIDÁNÍ NOVÝCH PRVKŮ

V praktických ukázkách si studenti vyzkouší přidání nových tlačítek do HTML dokumentu:

```
var input = document.createElement("input"); // vytvoří nový prvek typu input
input.setAttribute("type", "button"); // nastaví vlastnost typu tlačítko
input.setAttribute("value", "buttonek"); // nastaví textový popisek
input.addEventListener("click", removeButton);
/* připojí metody onclick pro odebrání tlačítka, bude vysvětleno dále */
document.getElementsByTagName("form")[0].appendChild(input);
/* připojí tlačítko do prvního formuláře v dokumentu */
```

Pozn.: Metoda `setAttribute(name, value)` lze použít i k nastavení jiných vlastností, většinou je třeba prvku přidat jednoznačný identifikátor, názvy jsou většinou logicky pojmenovány `setAttribute("id", "divID")`.

ODEBRÁNÍ PRVKŮ

Odebírání prvků probíhá velmi podobně pomocí metody `removeChild()` V příkladu si studenti zároveň vyzkouší obě dvě metody, klepnutím na tlačítko *Přidat tlačítko* se do formuláře přidá nové tlačítko, pokud se na nově vytvořené klepne, tlačítko se odebere. Z formuláře se odebere vždy poslední přidané. Přesné určení je součástí praktického cvičení.

```
var obj = document.getElementsByTagName("form")[0];
/* získá odkaz na objekt prvního formuláře v dokumentu HTML */
obj.removeChild(obj.lastChild); // odebere poslední prvek formuláře
```

NAHRAZENÍ PRVKŮ

Nahrazování prvků je možné realizovat pomocí metod `replaceChild()` a `insertBefore()`. V praktické ukázce jsou tři barevná tlačítka, po klepnutí na libovolné z nich se tlačítko s vybranou barvou přesune na první pozici. V příkladu je použit ukazatel `this`. `This` je ukazuje na aktuální objekt, chová se obdobně, jako kdyby měl objekt svoje unikátní `id`.

V praktickém cvičení si studenti vyzkouší nahrazování prvků tlačítek, po klepnutí na tlačítko dojde k jeho výměně na první pozici. Tlačítka jsou barevně rozlišena.

CVIČENÍ: NA NAHRAZENÍ PRVKŮ

```
<!doctype html>
  <html lang="cs" dir="ltr">
    <head>
      <title>DOM nahrazení prvku v dokumentu</title>
      <script>
        function beFirst(thisObj) { // v metodě je přijat ukazatel na tlačítko
          var obj = document.getElementById("formID"); // odkaz na formulář
          var firstInput = obj.getElementsByTagName("input")[0];
          // odkaz na první tlačítko ve formuláři
          if (thisObj != firstInput) {
          /* porovná zda vybrané tlačítko není první, lze vynechat */
            obj.replaceChild(thisObj, firstInput);
            /* metoda zamění ve formuláři vybraný objekt s prvním */
            obj.insertBefore(firstInput, thisObj.nextSibling);
            /* metoda přidá minulé první tlačítko na konec, jinak by při záměně
            došlo ke ztrátě nahrazovaného tlačítka */
          }
        }
      </script>
    </head>
  <body>
    <form id="formID">
    <!-- každé tlačítko předá v metodě ukazatel this, na objekt (prvek) input -->
    <input type="button" style="background-color: #f30" value="Na první místo"
      onclick="beFirst(this);" />
      <input type="button" style="background-color: #0c6" value="Na první místo"
      onclick="beFirst(this);" />
      <input type="button" style="background-color: #09f" value="Na první místo"
      onclick="beFirst(this);" />
    </form>
  </body>
</html>
```

Problémově zadaný je hodnocený příklad, studenti mají za úkol vytvořit aplikaci, která po klepnutí na jedno ze tří tlačítek zobrazí vždy jeden obrázek. Adresa obrázků je uložena v textových souborech, ty si musí stáhnout AJAXEm a zobrazit pomocí DOM. V tipech pro řešení je způsob jak založit nový obrázek:

```
img = document.createElement("img"); // založení obrázku
img.setAttribute('src', "odkaz na obrázek"); // nastavení atributu odkazu
```

3.6 OŠETŘENÍ TIMEOUTU V AJAXU

V některých případech může server přestat odpovídat na požadavek na stažení dat, pokud nedojde k žádnému chybovému stavu aplikace a potažmo uživatel se o tom nedozví a bude dále čekat na stažení dat. Problém je obtížné v reálných podmínkách nasimulovat, v ukázkovém příkladu si studenti vyzkouší stáhnout neexistující soubor bez ověření statusu a stavu.

3.6.1 ČASOVAČ

K ošetření se používají časovače JavaScriptu. K dispozici jsou dvě metody:

- `setTimeout()`
- `setInterval()`

Na rozdíl od první, druhá se spouští opakovaně každých x milisekund, pro její zastavení je třeba zavolat metodu `clearInterval()`. Aby bylo možné časovač zastavit, je třeba získat referenci na objekt pomocí proměnné:

```
var timer = setInterval(function () { alert("ahoj"), 1000);
/* každou sekundu zobrazí text ahoj, uvedená obecná funkce není potřeba */
clearInterval(timer); // zastaví pravidelné spouštění
```

Pro studenty je připraveno cvičení. Napsat digitální hodiny, které půjdou tlačítkem zastavit, po zastavení tím samým tlačítkem opětovně spustit.

NAPIŠTE APLIKACI, KTERÁ DO DOKUMENTU VLOŽÍ PRVEK, S AKTUÁLNÍM ČASEM.

- Čas se bude každou sekundu aktualizovat.
- Přidejte tlačítko, které čas zastaví.
- Při zastavení se změní význam, funkce i text tlačítka na spuštění a obráceně.
- Pokud se chcete "blýsknout" realizujte takto hodiny analogové.

V tipech na řešení je uveden způsob jak získat aktuální čas, a doporučení použití prvku HTML5 `<canvas>` v případě vykreslení analogových hodin[10].

[10] http://www.dhtmlgoodies.com/tutorials/canvas-clock

ŘEŠENÍ CVIČENÍ: DIGITÁLNÍ HODINY

```
<!doctype html>
<html lang="cs" dir="ltr">
  <head>
    <title>Hodiny</title>
    <script>
      var timer, time;
      function stopClock() { // funkce pro zastavení hodin
        clearInterval(timer); // odstranění časovače
        document.getElementById("btnID").value = "Spustit" // prohodí tlačítka
        document.getElementById("btnID").addEventListener("click", startClock);
        /* změní událost tak, aby se po stisknutí tlačítka spustila metoda pro
        spuštění hodin */
      }
      function startClock() { // funkce pro spuštění časovače hodin
        timer = setInterval("clock()", 1000);
        /* každou sekundu spustí metodu clock() */
        document.getElementById("btnID").addEventListener("click", stopClock);
        /* změní události tak, aby se po stisknutí tlačítka spustila metoda pro
        zastavení hodin */
      }
      function clock() { // funkce pro zjištění času
        time = new Date().toLocaleTimeString(); // vytvoří, převede čas na string
        document.getElementById("inputID").value = time; // vloží čas do inputu
      }
    </script>
  </head>
  <body onload= "startClock();">
    <form>
      <input type="text" size="6" id="inputID" />
      <input type="button" id="btnID" value="Zastavit"
      onclick="stopClock();" />
    </form>
  </body>
</html>
```

3.7 AJAX A CSS

Jak již jste možná poznali v minulých kapitolách, primárním důvodem k použití AJAXu je, že se stránky nemusí aktualizovat při každé změně stavu. To dává uživateli dobrý pocit. Nemusí dlouho čekat na odezvu serveru. Kromě technik umísťování obsahu mezi prvky nebo vytváření prvků nových, existuje ještě jedna možnost, jak ovlivnit změny na webové stránce, a to pomocí CSS stylů, které jsou měněny pomocí JavaScriptu nebo v kombinací s klasicky definovanými styly. Zde není možné uvést všechny CSS styly a vlastnosti, které lze prvkům nastavit, neb toto není učebnice CSS.

3.7.1 ZMĚNA BARVY TEXTU

V úvodním demonstračním příkladu byl uveden způsob, jak změnit barvu textu po stažení. Příklad nebyl zvolen náhodně. Většinou se obarvení textu používá pro informaci k uživateli, že nějaká část dokumentu byla změněna, případně jako výstraha chybně zadaných dat. Metoda `alert()` není úplně vhodná, jelikož její grafickou vizualizaci má na starosti operační systém. K nastavení barvy objektu, který tuto vlastnost umožňuje, stačí připojit `.style.color = barva`. Barvu lze zadat jejím názvem nebo hexadecimálně jako `#rrggbb` (zkráceně `#f00` bude červená, jelikož kanál r (ff) je na maximu 255 a ostatní jsou nulové apod.). V ukázce si studenti vyzkouší propojení timeru a nastavení barvy textu:

```
function getData(dataSource, objID, color) {
  document.getElementById(divID).style.color = color;
  /* prvku nastaví barvu předanou parametrem color */
  document.getElementById(divID).innerHTML = XMLHttpRequestObject.responseText;
  /* do prvku s id předaného v parametru se vloží získaný text z požadavku */
  setTimeout(function () { blinkText(objID); }, 1000);
  /* anonymní obecná funkce, která obsahuje metodu blinkText() s parametrem objID
  s nastavením časovače na 1s */
}
function blinkText(objID) {
  document.getElementById(objID).style.color = "black";
  /* nastaví barvu zpět na černou */
  setTimeout(document.getElementById(objID).innerHTML = "Obsah zprávy", 2000);
  /* za 2s vrátí původní text */
}
```

HTML kód obsahuje tři tlačítka, každé předá barvu metodě v parametru color. V tabulce jsou vypsány další možnosti nastavení barevnosti.

NĚKTERÉ VLASTNOSTI PRO ZMĚNU BARVY A POZADÍ:

Vlastnost	Popis
color	Barva popředí: #rrggbb | colorName
background-color	Barva pozadí: #rrggbb | colorName
background-image	Obrázek pozadí: URL
background-repeat	Způsob zobrazení obrázku: repeat | repeat-x | repeat-y | no-repeat
background-attachment	Určuje, zda dojde k pohybu obrázku: scroll | fixed
background-position	Nastavení pozice obrázku: top | center | bottom | left | right

3.7.2 ZMĚNA FONTU TEXTU

Kromě obarvení staženého textu lze použít i jiné textové efekty, jak upozornit uživatele nebo ozvláštnit svoje aplikace. Je možné aplikovat různé textové efekty pro zvětšení fontu, změnu fontu, podtržení, nastavení tučného písma apod.

Vlastnost	Popis
font-family	Font písma: jméno fontu, případně výčet náhradních
font-style	Způsob vykreslení písma: normal | italic | oblique.
font-weight	Duktus písma: normal | bold | bolder | lighter | aj.
line-height	Výška řádku: px | pt | % | aj.)
font-size	Velikost fontu: px | pt | % | aj.)
text-decoration	Způzosb zvýraznění: none | underline | overline | line-through
text-align	Vycentrování textu: | left | right | center

V ukázce si studenti vyzkouší nastavit efekty písma jako podtržení nebo centrování na stránce pomocí "radio buttonů". Prvky radio musí mít stejný název, aby došlo k jejich seskupení. Vlastnost `checked = "true | false | checked"` určuje výchozí označení. Tuto vlastnost lze číst a vyhodnocovat třeba podmínkou:

```
if (document.getElementById(objID).checked) { ... }
/* pokud má objekt vlastnost checked="checked" /*
```

3.7.3 NASTAVENÍ POZICE PRVKU V DOKUMENTU

Nejčastěji se používá AJAXu a CSS pro nastavování pozice prvků na webové stránce pro různá vyskakovací menu, dialogy, seznamy apod. Lze realizovat dokonce funkce uchopení a táhnutí (drag & drop) libovolného objektu. Podstatnému rozšíření této funkce brání velká složitost kódu, a proto jsou často využívány knihovny jQuery, kde je kód (knihovna) připravena k připojení k webové stránce. HTML5 obsahuje nové metody pro drag & drop.

Pomocí CSS lze nastavovat pozice prvku vlastnosti `position: absolute | relative | fixed` a následné umístit od okraje pomocí odsazení. Je třeba si dát pozor, protože odsazení se udává v pixelech. Ve zdrojovém kódu je nutné vždy připojit k číselné hodnotě text `+ "px";`

CSS VLASTNOSTI PRO NASTAVENÍ POZICE PRVKŮ:

Vlastnost	Popis
position	Nastavení absolutní pozice prvku
top	Odsazení od shora
bottom	Odsazení od spodu
left	Odsazení zleva
right	Odsazení zprava
z-index	Nastavení pořadí překryvu prvků

Přestože se většinou CSS používá k nastavení prvků tlačítek, k tvorbě vysouvacích menu, v ukázce si studenti vyzkouší rozpohybovat "neidentifikovatelný létající objekt". Objekt se ovládá pomocí kurzorové klávesnice nebo přejetím myši přes příslušná tlačítka (obsahují speciální zápis entit `&larr;` pro šipky, které jsou ve výsledku přeloženy interpreterem). Uprostřed se nalézá tlačítko, které vrátí objekt na střed webové stránky. Velikost webové stránky je načtena při spuštění z vlastnosti `window.innerWidth` respektive `window.innerHeight`. V ukázce není ošetřeno vyletění objektu z viditelné oblasti, to bude mimo jiné úkolem samostatné práce.

ZDROJOVÝ KÓD UKÁZKY: UFO, OVLÁDANÉ KLÁVESNICÍ NEBO TLAČÍTKY NA OBRAZOVCE

```
<!doctype html>
  <html lang="cs" dir="ltr">
   <head>
     <title>AJAX a změna pozice prvku pomocí CSS</title>
     <style>
       #imgID { <!-- nastavení absolutní pozice CSS -->
         position: absolute;
       }
       form {
         text-align: center
       }
     </style>
     <script>
       function init(imgID, bodyID) {
         document.getElementById(imgID).style.left
         = window.innerWidth / 2 - 64 + "px";
         /* nastaví pozici doprostřed okna stránky (zleva) */
         document.getElementById(imgID).style.top
         = window.innerHeight / 2 + "px";
         /* nastaví pozici doprostřed okna stránky (zhora) */
         document.getElementById(bodyID).onkeydown = function(event) {
           /* tělu body se připojí událost po stisknutí klávesy */
           switch(event.keyCode) {
           /* přepínač vyhodnotí získaný kód klávesy
           a podle toho zavolá metodu move
           s příslušným parametrem, id, směr, velikost posuvu */
             case 37: move(imgID, false, -5); break;
             /* kód 37 odpovídá kurzorové klávese vlevo, false určí osu y, a -5
             zařídí posuv o -5 px */
             case 38: move(imgID, true, -5); break;
             case 39: move(imgID, false, 5); break;
             case 40: move(imgID, true, 5); break;
           }
         }
       }
       function move(imgID, direction, shift) {
         if (!direction) { /* dle směru se posune zleva nebo shora */
           document.getElementById(imgID).style.left =
          (parseInt(document.getElementById(imgID).style.left) + shift) + "px";
         } else {
           document.getElementById(imgID).style.top =
           (parseInt(document.getElementById(imgID).style.top) + shift) + "px";
            /* textová hodnota se převede na int metodou
            parseInt a přičte se posuv */
         }
       }
     </script>
  </head>
  <body id="bodyID" onload="init('imgID','bodyID');">
    <img id="imgID" src="ufo.png" />
    <form>
      <input type="button" value="↑" onmousemove="move('imgID',true, -5);" />
      <br />
      <input type="button" value="←" onmousemove="move('imgID',false, -5);" />
      <input type="button" value="•" onclick="init('imgID');" />
      <input type="button" value="→" onmousemove="move('imgID',false, 5);" />
      <br />
      <input type="button" value="↓" onmousemove="move('imgID',true, 5);" />
    </form>
  </body>
</html>
```

3.8 KNIHOVNA JQUERY

JQuery je rychlá, malá JavaScriptová knihovna (API) se spoustou možností. Umožňuje manipulovat s HTML dokumenty jednodušeji než psaním čistého JavaScript kódu. Poskytuje metody pro události, animace a je daleko jednodušší na použití. jQuery API knihovna je funkční ve všech moderních prohlížečích. Díky kombinaci všestrannosti a rozšiřitelnosti jQuery mohou lidé psát jednoduché kódy. jQuery není jediná knihovna, existují i další jako Prototype, MooTools, Dojo a spoustu knihoven pro speciální efekty [11]*. JQuerry knihovna je tu s námi již od roku 2006, kdy byla vydána Johnem Resigem na newyorském BarCampu (mezinárodní síť otevřených konferencí a workshopů, zaměřená na nové trendy v internetových aplikacích). V současné době (2013) je v betatestu verze knihoven 2.0.* (10)

STEJNĚ JAKO AJAX UMOŽŇUJE JQUERY:

- Přistupovat k jednotlivým částem stránky.
- Měnit vzhled stránky.
- Rozšiřovat a měnit obsah stránky.
- Reagovat na události uživatele na stránce.
- Přidávat animace a efekty.
- Získávat informace ze serveru bez nutnosti načtení celé stránky.

A to vše podstatněji jednodušším způsobem. Můžeme abstrahovat od určitých rozmanitostí jednotlivých prohlížečů, kdy každý má svou vlastní interpretaci standardů. jQuery přidává abstraktní vrstvu, s níž můžeme překonat jednotlivé rozdíly mezi prohlížeči. Snižuje objem kódu a nesmírně ho zjednoduší a to také díky tzv. zřetězení, kdy můžete určitému prvku přiřadit na sebe navazující akce.

Stejně jako CSS oddělují „zobrazovací“ charakteristiky od struktury HTML, jQuery odděluje „chování“ od struktury HTML. Například místo přímé specifikace `onclick` události v HTML kódu tlačítka, by stránka řízená jQuery napřed našla vhodný prvek tlačítka a potom změnila jeho manipulátor události. Takovéto oddělení chování od struktury se nazývá jako princip nevtíravého JavaScriptu. K dispozici je i jQuery UI (**U**ser **I**nterface) obsahující schémata a nové grafické prvky, které zatím nejsou implementovány ani v HTML 5. Knihovny jQuery jsou hostovány třeba společností Google, a proto není nutné stahovat

[11] http://javascriptlibraries.com

zdrojové kódy přímo na vlastní web. Tím je zaručena vždy aktuálnost použitých knihoven i při změně jejich verze. Na druhou stranu to může vést k neočekávaným problémům při nedostupnosti nebo změnám v chování. V tomto případě je nejlepším řešením umístit jQuery k sobě na web. Pokud nepotřebujeme nahlížet do zdrojového kódu, je datově výhodnější stáhnout komprimovanou verzi knihovny. Ta se označuje jako min, funkčně se neliší, jen jsou odstraněny zbytečné mezery a konce řádek (nepotřebné pro stroj, člověk má problém takovýto nestrukturovaný kód číst).

3.8.1 JQUERY SYNTAXE: $(SELECTOR).ACTION()

- `$` značka použití jQuery.
- `(selector)` výběr HTML prvku, mohou se vybírat pomocí `id`, třídy, názvu, pořadí
- `action()` co se má s prvkem udělat (událost).

3.8.2 PŘIPOJENÍ K WEBOVÉ STRÁNCE

Knihovna jQuery se připojuje stejně jako jakýkoliv jiný script. Pro spuštění kódu není nutné volat metodu `onload()`. JQuery obsahuje vlastní metodu `$(document).ready(function()`, která se spustí po načtení celého obsahu. Podle uvozujícího znaku dolar `$`, jQuery pozná svoje metody. Specifické je i jejich ukončení pomocí dvou závorek a středníku `});`.

3.8.3 METODY PRO VÝBĚR PRVKŮ (SELEKTOR)

- Selektor prvku: `$("p")`
- Selektor #id: `$("#divID")`
- Selektor třídy (.class): `$(".myClass")`

PŘÍKLADY SELEKTORŮ:

```
$("p, a, div, .myClass, #myID")
/* vybere více prvků s id, třídou nebo pomocí tagu */
selectedValue = $("#mySelect option:selected").val()
/* vybere hodnoty option z formuláře select */
$("#divID textarea").hide()
/* vybere potomka textarea v prvku s id="divID" a neprodleně ho schová */
$(this)
/* vybere prvek přímo pomocí ukazatele this */
$("#selectID option:first")
/* vybere prvek select z option pomocí pořadí */
```

3.8.4 JQUERY EFEKTY

JQuery umožňuje prvkům nastavit i různé zajímavé efekty:

Metoda	Popis
hide()	Skrytí prvku
show()	Odkrytí prvku
fadeIn()	Postupné objevení (prolnutí)
fadeOut()	Postupné mizení
fadeToggle()	Dle stavu spustí Out nebo In
slideUp()	Vysunutí prvku
slideDown()	Zasunutí prvku
slideToggle()	Dle stavu dojede k Up nebo Down
animate()	Animace dle parametrů

Každému efektu lze nastavit délku trvání a funkci, která se provede po dokončení, tzv. `callback()`. Místo callbacku lze použít i známou obecnou funkci:

```
$(selector).hide(speed, callback)
/* nastaví metodu callback spuštěnou až zkončí efekt */
$(selector).hide(1000, function() { alert("po schování se za 1s spustila obecná
funkce"); })
/* po schování se spustí obecná funkce */
```

DALŠÍ METODY:

```
$(selector).stop()
/* metoda zastaví prováděnou akci na prvku, může zastavit i provádění efektu */
$("#divID").text("nastavil jsem text").slideUp(2000).slideDown(2000);
/* zřetězení, nejdříve se nastaví prvku text a poté se vysune a následně opět
zasune */
```

3.8.5 JQUERY MANIPULACE S OBSAHEM A ATRIBUTY

jQuery nabízí na DOM založené metody, které zásadním způsobem zjednodušují přístup a manipulaci s prvky a atributy:

Metoda	Popis
text()	Nastavení nebo vrácení textového obsah prvku
html()	Nastavení nebo vrácení obsahu prvku včetně HTML značek
val()	Nastavení nebo vrácení hodnot z prvků formulářů
attr()	Nastavení nebo vrácení atributů prvků

3.8.6 JQUERY PŘIDÁVÁNÍ A ODEBÍRÁNÍ PRVKŮ

Stejně tak využívá jQuery DOM metody k přidávání nebo odebírání prvků z dokumentu:

Metoda	Popis
append()	Vložení obsahu na konec prvku
prepend()	Vložení obsahu na začátek prvku
after()	Vložení obsahu za vybraný prvek
before()	Vložení obsahu před vybraný prvek
remove()	Odstranění rodičovského prvku a jeho potomků
empty()	Odstranění potomky z prvku

3.8.7 JQUERY METODY PRO MANIPULACI S CSS

Metoda	Popis
addClass()	Přidání jednu nebo více CSS tříd k prvku
removeClass()	Odebrání jedné nebo více CSS tříd z prvku
toggleClass()	V závislosti na stavu přidání nebo odebrání CSS tříd z prvku
css()	Nastavení nebo vrácení styly prvku

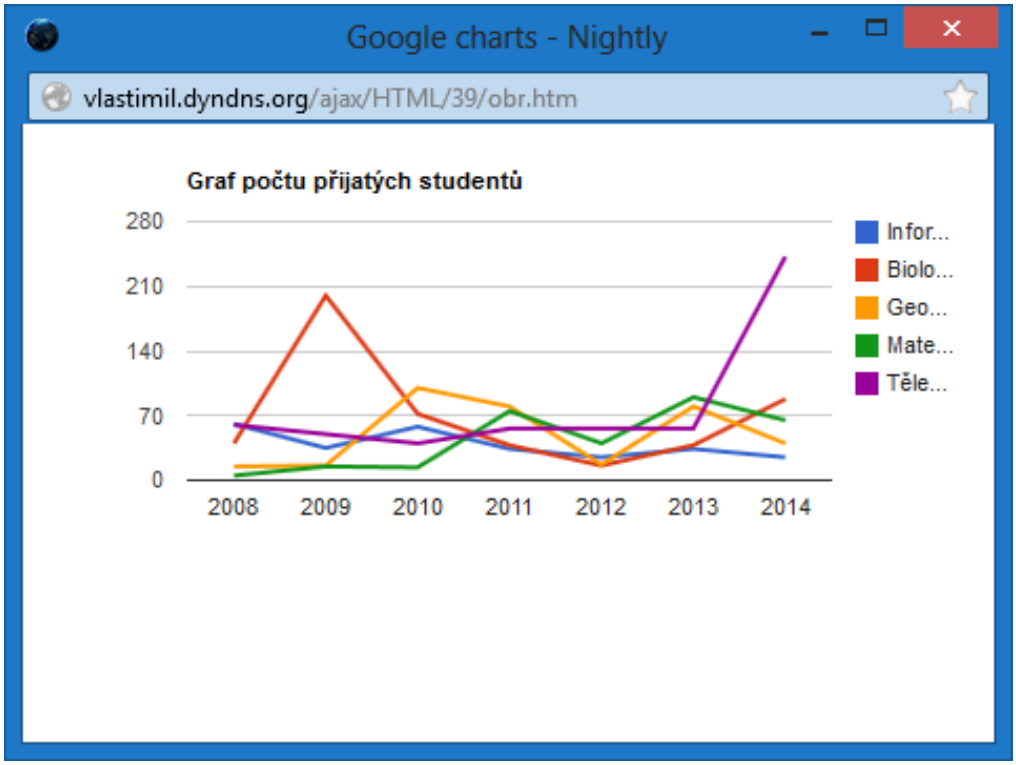
3.8.8 AJAX JQUERY

Nejmarkantnější zjednodušení se dostalo k asynchronním dotazům na data. Programátor již nemusí nic kontrolovat, vše za něj udělá jQuery knihovna. Parsování XML je podobně jednoduché. V ukázce je demonstrováno parsování XML dokumentu s auta.xml.

```
<!doctype html>
  <html lang="cs" dir="ltr">
    <head>
      <title>XML parser jQuery</title>
      <script src="../data/jquery-2.0.2.min.js"></script>
      <script>
        $(document).ready(function () {
        /* funkce se spustí po načtení celé stránky */
          $.ajax ({ /* vytvoří se ajax dotaz */
            type: "GET", /* nastaví se typ dotazu GET */
            url: "auta.xml", /* nastaví se požadovaný soubor */
            dataType: "xml", /* nastaví se typ souboru XML */
            success: xmlParser
            /* při úspěchu se zavolá metoda xmlParser */
          });
        })
        function xmlParser(xml) { /* metodě se předá parametr s objektem xml */
          $("body").append("<a target='_blank'
          href='auta.xml'<auta.xml></a><ul id='targetDiv'></ul>");
          /* založí se prvky <a>; a <ul>; do těla prvku <body> */
          $(xml).find("auto").each(function () { /* pro každé auto */
            $("#targetDiv").append("<li>;" + $(this).find("jmeno").text()
            +  "</li>");
            /* se přidá do dokumentu jméno auta */
          });
        }
      </script>
  </head>
</html>
```

3.9 GOOGLE CHARTS

Google Charts (obrázek 11) je výborné JavaScriptové API pro vizualizaci grafů. Typy grafů, které umí Google vizualizovat, si lze prohlédnout přímo na jeho webu [12]. Najít tam jde playground (pískoviště), kde je možné vybraný graf nasimulovat. Grafy jsou založené čistě na HTML5 a SVG technologii (nicméně Google udržuje kompatibilitu pomocí VML pro starší verze IE). K zobrazení nejsou potřeba žádné pluginy. Přidání grafu do vlastního dokumentu vyžaduje jen několik málo jednoduchých kroků.



Obrázek 11: Google Charts

[12] https://developers.google.com/chart/interactive/docs/gallery

V Ukázce si studenti otestujou statickou (předem nadefinované pole) i dynamickou (data jsou generována cyklem náhodně) definice grafu.

UKÁZKA ZDROJOVÉHO KÓDU: GOOGLE CHARTS

```
google.load("visualization", "1", {packages:["corechart"]});
/* načtení modulů */
google.setOnLoadCallback(drawChart);
/* po načtení všech scriptů se zavolá metoda drawChart */
document.getElementById('chartDiv').style.left = window.innerWidth + "px";
/* zjistí rozměry okna a přizpůsobí velikost grafu */
document.getElementById('chartDiv').style.top = window.innerHeight + "px";
/* zjistí rozměry okna a přizpůsobí velikost grafu */
function drawChart() { /* funkce pro vizualizaci dat */
  data = google.visualization.arrayToDataTable([ /* naplní graf daty */
    ['Rok', 'Informatika', 'Biologie', 'Geografie', 'Matematika', 'Tělesná
    výchova'],
    /* nadpisy sloupců pro jednotlivé spojnice */
    ['2008', 61, 40, 15, 5, 60],
    /* sloupce dat pro rok 2008 (matice) */
    ['2009', 35, 60, 16, 15, 50],
    /* všimněte si, že textové hodnoty jsou rozlišeny od číselných pomocí '' */
    ['2010', 58, 72, 25, 14, 40],
    /* sloupce i řádky jsou odděleny středníky, řádky uzavřeny v hranatých
    závorkách */
    ['2011', 34, 38, 80, 75, 56],
    ['2012', 25, 16, 17, 40, 56],
    ['2013', 34, 38, 80, 75, 56],
    ['2014', 25, 88, 40, 65, 42]
  ]);
  options = {
  /* možnost přidání některých dalších vlastností grafu */
    title: 'Graf počtu přijatých studentů'
    /* přidá nadpis ke grafu */
  };
  chart = new
  google.visualization.LineChart(document.getElementById('chartDiv'));
  /* připojí graf do dokumentu */
  chart.draw(data, options);
  /* vykreslí graf s vlastnostmi */
}
```

Staticky definovaný graf je sice pěkný, ale v AJAX aplikacích bude potřeba načíst do grafu data z XML. V ukázce si studenti vyzkoušejí způsob, jakým se přidávají nové řádky a sloupce do grafu. Načtení dat z XML je pak už jednoduché.

```
data.addColumn('string', 'Year');
/* přidá osu x s roky typu string */
data.addColumn('number', 'Informatika');
/* přidá osu y typu number */
for(i = 0; i < document.getElementById('years').value; i++) {
/* cyklus for omezený hodnotou v prvku input */
  data.addRow([(2004 + i).toString(), Math.floor((Math.random() * 100) + 1)])
  /* přidá rok a naplní náhodným počtem studentů */
}
```

V hodnocené práci mají za úkol přidat dynamicky sloupce a řádky se seznamem všech oborů a počtem přijatých studentů, počet mají stanovit náhodně a dopředu lze zadat období, na které se má statistika počítat.

4 SEZNAM OBRÁZKŮ

5 SEZNAM LITERATURY

1. **Wikipedie: Otevřená encyklopedie.** ***AJAX.*** **[Online] [Citace: 28. 6 2013.] http://cs.wikipedia.org/wiki/AJAX.**

2. Wikipedie: Otevřená encyklopedie. ***HyperText_Markup_Language.*** **[Online] [Citace: 28. 6 2013.] https://cs.wikipedia.org/wiki/HyperText_Markup_Language.**

3. Wikipedie: Otevřená encyklopedie. ***Extensible Markup Language.*** **[Online] [Citace: 28. 6 2013.] http://cs.wikipedia.org/wiki/Extensible_Markup_Language.**

4. Wikipedie: Otevřená encyklopedie. ***JavaScript.*** **[Online] [Citace: 28. 6 2013.] http://cs.wikipedia.org/wiki/Javascript.**

5. Wikipedie: Otevřená encyklopedie. ***Document Object Model .*** **[Online] [Citace: 28. 6 2013.] http://cs.wikipedia.org/wiki/Document_Object_Model.**

6. Holzner, Steven. ***AJAX: A Beginner's Guide.*** **New York : McGraw-Hill Osborne Media, 2008. str. 475.**

7. Wikipedie: Otevřená encyklopedie. ***Kaskádové styly.*** **[Online] [Citace: 28. 6 2013.] http://cs.wikipedia.org/wiki/Kask%C3%A1dov%C3%A9_styly.**

8. Rentel. ***Autorský systém ProAuthor.*** **[Online] [Citace: 28. 6 2013.] http://rentel.cz/rentel/rentelweb.nsf/0/proauthor.**

9. codemirror.net. ***CodeMirror.*** **[Online] [Citace: 6. 28 2013.] http://codemirror.net.**

10. jQuery. ***What is jQuery?*** **[Online] [Citace: 6. 28 2013.] http://jquery.com/.**

11. Wikipedie: Otevřená encyklopedie. ***WYSIWYG.*** **[Online] [Citace: 28. 6 2013.] http://cs.wikipedia.org/wiki/WYSIWYG.**

12. w3schools.com. ***jQuery Tutorial.*** **[Online] [Citace: 28. 6 2013.] http://www.w3schools.com/jquery.**

6 RESUMÉ

Zadáním diplomové práce bylo vytvořit distanční kurz pro výuku AJAXu. Materiál obsahuje vhodné studijní články a odpovídající cvičení a příklady. V kurzu je preferováno praktické ověřování dovedností nad teoretickými znalostmi.

Mým úkolem bylo vytvořit kvalitní výukový materiál dodržující současné webové standardy a konvence pro studenty, kteří již mají předchozí zkušenosti s tvorbou webových aplikací. Díky kurzu studenti získají dovednosti potřebné k tvorbě vysoce interaktivních webových AJAX aplikací. Současně jsem se snažil, aby studenti dostali ucelený přehled problematiky AJAX technologií a budoucích směrech jejího vývoje.

Hlavní část diplomové práce popisuje stručný obsah kurzu, kde jsem chtěl nastínit strukturu studijních článků a připojených aktivit. Velkou část jsem věnoval jak inovacím v editoru zdrojového kódu, tak i způsobu grafického zobrazení článků s cílem, aby práce s distančním kurzem byla pro studenty co nejkomfortnější.

Doufám, že vytvořený distanční kurz AJAXu bude aplikovatelný nejen pro výuku studentů na vysokých školách, ale i pro veřejnost s hlubším zájmem o moderní webové technologie.

7 RESUMÉ (ENGLISH)

By entering this thesis was to create a distance learning course for learning AJAX. Material includes suitable Study articles and appropriate exercises and examples. The course preferred practical skills over verification of theoretical knowledge.

My task was to create high quality educational material observing the current web standards and conventions for students who have prior experience in creating web applications. Thanks to the course, students will gain the skills needed to create highly interactive AJAX web applications. At the same time I tried to make students get a comprehensive overview of AJAX technology issues and future directions of its development.

The main part of the thesis describes a brief synopsis of the course where I wanted to outline the structure of the study cells and associated activities. The great part is devoted to innovation in the source code editor, and graphic way of displaying articles in order to work with the distance of the course was for students as comfortable.

Hope to establish distance learning course AJAX is applicable not only to teach students at universities, but also to the public with a deeper interest in modern web technologies.

8 PŘÍLOHY

CD obsahuje text diplomové práce a vlastní Distanční kurz AJAX. Popis obsahu složek umístěných na CD:

- Text diplomové práce ve složce diplomova_prace.
- Kurz exportovaný do formátu E-book ve složkách e-book1 a e-book2.
- Zdrojové kódy interpreteru kódu ve složce zdrojove_kody.
- Zdrojové soubory AJAX kurzu a zdrojové obrázků a animací ve složce zdroj_kurzu.