TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky a mezioborových inženýrských studií

Studijní program: M2612 – Elektrotechnika a informatika
Studijní obor: 3902T005 – Automatické řízení a inženýrská informatika

AJAX na mobilních zařízeních

AJAX on mobile devices

Diplomová práce

Autor: **Jan Kovář**
Vedoucí práce: Ing. Miloš Turek
Konzultant: Ing. Ondřej Raška, MITON CZ, s.r.o.

V Liberci 18. 5. 2007

Prohlášení

Byl(a) jsem seznámen(a) s tím, že na mou diplomovou práci se plně vztahuje zákon č. 121/2000 o právu autorském, zejména § 60 (školní dílo).

Beru na vědomí, že TUL má právo na uzavření licenční smlouvy o užití mé diplomové práce a prohlašuji, že **s o u h l a s í m** s případným užitím mé diplomové práce (prodej, zapůjčení apod.).

Jsem si vědom(a) toho, že užít své diplomové práce či poskytnout licenci k jejímu využití mohu jen se souhlasem TUL, která má právo ode mne požadovat přiměřený příspěvek na úhradu nákladů, vynaložených univerzitou na vytvoření díla (až do jejich skutečné výše).

Diplomovou práci jsem vypracoval(a) samostatně s použitím uvedené literatury a na základě konzultací s vedoucím diplomové práce a konzultantem.

Datum

Podpis

Poděkování

Rád bych touto cestou poděkoval Ing. Miloši Turkovi za vedení této diplomové práce, dále firmě MITON CZ, s.r.o. a především Ing. Ondřeji Raškovi a Jakubu Prokopci za poskytnuté zázemí. V neposlední řadě bych chtěl poděkovat své rodině a přítelkyni za podporu v posledních fázích studia.

Anotace

Diplomová práce se zabývá problematikou nového přístupu k webovým aplikacím, a tím je používání technologie AJAX k tvorbě vysoce interaktivních webových aplikací. Zvláště se zabývá využitím této mladé technologie na mobilních zařízeních.

Úvodní část seznamuje se základy AJAXu a přináší přehled výhod, nevýhod a možností využití AJAXu v standardních webových prohlížečích. Dále se práce věnuje vývoji mobilního webu a jeho současným možnostem. Je zde zpracován přehled moderních mobilních prohlížečů a představeny alternativy pro testování mobilních webových aplikací. Třetí kapitola se zaměřuje na současné možnosti využití AJAXu při tvorbě webových aplikací pro mobilní zařízení.

V poslední části práce je vytvořena mobilní webová aplikace, která využívá technologie AJAX. Tato aplikace je vytvořena pomocí běžně užívaných webových standardů a byla testována v dostupných mobilních prohlížečích, jako Opera Mobile, prohlížeč Nokia S60, mobilní Internet Explorer a Minimo.

Abstract

This Diploma Thesis deals with the problems of a new approach to web applications and that is the use of the AJAX technology to creating highly interactive web applications. It actually deals with the usage of this new technology for mobile devices.

The introductory part introduces the basics of Ajax and bears briefing of the advantages and disadvantages as well as the use and application of AJAX on standard web browsers. Furthermore, the thesis deals with the progress of mobile web and its contemporary capabilities. Here it's processed an overview of modern mobile browsers, as well as introduction to alternative testing methods of mobile web applications. The third chapter focuses on current capabilites of the use of AJAX to creating web applications for mobile devices.

In the last part of the thesis, there is created web application, which applies the AJAX technology. This application is created with the help of commonly used web standards and was tested in accessible mobile browsers, like Opera Mobile, Nokia S60 browser, mobile Internet Explorer and Minimo.

Obsah

Seznam zkratek

AJAX	Asynchronous JavaScript And XML
CGI	Common Gateway Interface
CSS	Cascading Style Sheets
DOM	Document Object Model
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
IE	Internet Explorer
JSON	JavaScript Object Notation
J2ME MIDP	Java Platform, Micro Edition Mobile Information Device Profile
MWBP	Mobile Web Best Practices
MWI	Mobile Web Initiative
OMA	Open Mobile Alliance
OS	Operační Systém
PC	Personal Computer
PDA	Personal Digital Assistant
PHP	Hypertext Preprocessor
PPC	Pocket Personal Computer
RDA	Remote Device Access
SDK	Software Development Kit
SSR	Small-Screen Rendering
TCP/IP	Transmission Control Protocol / Internet Protocol
TLD	Top-Level Domain
URL	Uniform Resource Locator
WCSS	Wireless Cascading Style Sheets
WML	Wireless Markup Language
W3C	World Wide Web Consortium
XHTML	eXtensible HyperText Markup Language
XHTML MP	eXtensible HyperText Markup Language Mobile Profile
XML	eXtensible Markup Language
YFT	Yellow Fade Technique

ÚVOD

Internet prošel během své existence velkým vývojem a již dávno tomu není tak, kdy sloužil pouze pro statickou prezentaci informací a obrázků a byl určen převážně pro výměnu dat mezi vědci. Je až překvapující, jak se jeho podoba za tuto dobu rapidně změnila. Internet v dnešní době překypuje nejrůznějšími službami od nesčetného množství klasických webových stránek, kde jsou prezentovány informace, přes webové aplikace jako emailové schránky, elektronické obchody, až po velice dynamické a interaktivní webové aplikace.

Takovéto velice vyspělé, uživatelsky velmi přívětivé, webové aplikace, které se svým vzhledem i chováním v mnohém podobají desktopovým (lokálním) aplikacím, je možné vytvářet díky nejmodernějším technologiím, které jsou v současné době k dispozici. Jedním z těchto prostředků pro tvorbu vysoce interaktivních webových aplikací je poměrně nová technologie nazvaná AJAX.

Na jedné straně máme AJAX – moderní technologii určenou nejmodernějším webovým prohlížečům na osobních počítačích a na druhé straně máme mobilní zařízení (mobilní telefony, kapesní počítače, …), jejichž schopnosti nebývale rychle rostou a které se stále více používají jako klienti webových aplikací. I mobilní web roste rychle a možnosti mobilního připojení k internetu se stále zlepšují. Proč by se tedy nevyzkoušela vhodnost takovéto technologie i pro mobilní zařízení?

Tato diplomová práce byla zadána ve spolupráci s firmou Miton CZ, s. r. o., která poskytuje služby v oblasti webdesignu, webhostingu a distribuci software. Cílem této diplomové práce je zmapovat možnosti současného mobilního webu. Prozkoumat možnosti využití AJAXu v mobilních zařízeních. A nakonec, na základě získaných teoretických znalostí, vytvořit webovou aplikaci, která bude využívat AJAX.

Zpráva je rozdělena do několika kapitol. Úkolem první kapitoly je seznámit se základy a způsoby použití AJAXu a vysvětlit vzájemné souvislosti mezi jednotlivými prvky této technologie. Tato část se převážně zabývá technikami na straně klienta, kde je největší důraz kladen na použití objektu xmlHttpRequest, který umožňuje asynchronní komunikaci mezi klientem a serverem.

Druhá kapitola se zaměřuje na problematiku mobilního webu. V prvé řadě je popsán vývoj a současný stav. Dále jsou představeny rozšířené moderní mobilní prohlížeče a nastíněny jejich schopnosti. Tato kapitola také popisuje některá specifická

omezení mobilních zařízeních a na jejich základě jsou zde stručně popsána některá doporučení pro správné vytváření mobilního webu. Kapitolu uzavírá přehled možností testování webových aplikací určených mobilním zařízením, jako jsou kapesní počítače a tzv. chytré telefony.

Závěrečné části zprávy jsou věnovány v prvé řadě přehledu současných možností AJAXu v mobilních zařízeních a úplně na závěr je popsána vytvořená ukázková aplikace, která pro svou funkci využívá technologie AJAX.

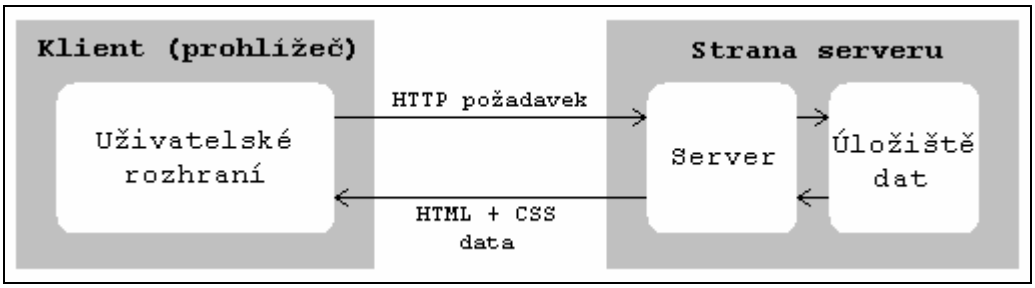
1 AJAX

1.1 Seznámení s AJAXem

1.1.1 Webové aplikace

Dříve než bude vysvětleno, co se skrývá za pojmem AJAX, je dobré připomenout, co jsou webové aplikace a co je jejich přínosem.

Webové aplikace jsou takové aplikace, které se k uživateli dostávají ze serveru prostřednictvím sítě internet. Uživatel s nimi pracuje pomocí klienta (webový prohlížeč). Prohlížeč je v podstatě nástroj, který webovou aplikaci zpracuje a zobrazí. Jak vypadá klasický webový model je patrné na Obr. 1.1.



Obr. 1.1 – *Klasický webový model*

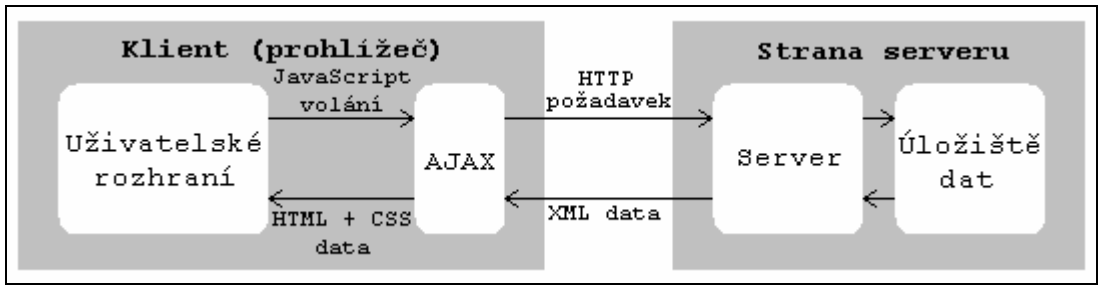
Tento model je typickým příkladem synchronního přenosu typu požadavek/odpověď. Každý požadavek je zpracován serverem a následně je odpověď zaslána klientovi.

Uživatel potřebuje pro práci s webovou aplikací pouze počítač připojený do sítě internet a některý z webových prohlížečů. Pro společnosti je tato skutečnost velice výhodná, protože nemusí instalovat software na mnoha počítačích jednotlivých uživatelů. Cesta, jak dostat aplikaci k uživateli, je pro ně daleko jednodušší a levnější. Dalším významným přínosem těchto aplikací je jejich snadná aktualizace. Jednoduše se aplikace aktualizuje na serveru a uživatel má ihned k dispozici nejnovější verzi. Webové aplikace nejsou omezeny operačním systémem, který uživatel na svém počítači používá. Správně vytvořená aplikace by měla fungovat v kterémkoli moderním prohlížeči.

1.1.2 Co je AJAX?

Termín AJAX se poprvé objevil počátkem roku 2005 v článku Jesse Jamese Garreta s názvem „Ajax: A New Approach to Web Applications“ [1]. Autor zde rozebírá postupný úbytek rozdílů mezi webovými a klasickými klientskými aplikacemi. Původně byl pojem AJAX zkratkou pro Asynchronous JavaScript and XML (asynchronní JavaScript a XML), ale ve skutečnosti zastřešuje více technologií.

Pomocí AJAXu je možné vytvářet daleko dynamičtější webové aplikace. Jednoduše řečeno – AJAX umožňuje volat na pozadí server a podle potřeby získávat data. Jedná se o asynchronní komunikaci, která nikterak nepřeruší práci uživatele. Zatímco uživatel pracuje s webovou stránkou, může dojít k asynchronní komunikaci na pozadí a následně k aktualizaci prohlížené stránky bez nutnosti kompletního znovunačtení stránky. Jak již bylo řečeno, klasické webové aplikace pracují na základě požadavek/odpověď (viz. Obr. 1.1). Dosud byly aktivity na straně serveru a na straně klienta oddělené. Oproti tomu AJAX přináší nový přístup k webovým aplikacím. Na Obr. 1.2 je zobrazen webový model využívající AJAX.



Obr. 1.2 – *Webový model využívající AJAX*

Prvky AJAXu

Bylo řečeno výše, že AJAX zastřešuje více technologií. Nejedná se tedy o nějaký samostatný programovací jazyk, ale o směs technologií, díky kterým lze vytvářet inteligentnější webové aplikace využívající asynchronní komunikaci se serverem.

AJAX se skládá z těchto prvků:

- *HTML/XHTML* a *CSS* – standardně užívané technologie pro prezentaci v prohlížeči.
- *DOM* – umožňuje manipulovat s HTML a XML dokumenty.

- *Objekt* `XMLHttpRequest` – umožňuje vytvořit HTTP požadavek pro asynchronní komunikaci se serverem.
- *XML* – formát pro výměnu dat.
- *JavaScript* – základní složka AJAXu. Představuje hlavní funkcionalitu.
- *Technologie na straně serveru* – nezbytné pro zpracování požadavků.

Výhody a nevýhody AJAXu

Jako každý nástroj či technologie i AJAX s sebou nese řadu možných výhod a samozřejmě i některé nevýhody.

AJAX přináší do vývoje webu tyto možné výhody:

- Umožňuje vytvářet inteligentnější, přívětivější a přístupnější webové aplikace.
- Odstraňuje nutnost znovunačtení a překreslení celé stránky jako u klasického modelu. Během práce uživatele lze aktualizovat jen některé požadované části. To uživateli urychluje práci a šetří čas.
- Vyrovnává zátěž mezi klientem a serverem. Přenáší se menší množství dat (pouze změny).
- Využívá stávající technologie a znalosti vývojářů.

Nevýhody AJAXu:

- Znemožňuje použití tlačítka „Zpět“ ve webovém prohlížeči. Webová stránka se chová jako aplikace. Vše se odehrává na jedné stránce.
- Současně se během práce nemění adresa v prohlížeči. Tudíž není možné tyto stránky v průběhu práce záložkovat.
- Nevhodným použitím zátěž neklesne, protože se naopak zvýší počet HTTP požadavků.
- Dalším možným problémem může být síťová latence. Uživateli nemusí být jasné, že aplikace zpracovává jeho požadavek, a snaží se operaci znovu spustit.
- Uživatel může ve svém prohlížeči zakázat JavaScript, což vede k nefunkčnosti webové aplikace využívající AJAX.

Využití AJAXu

AJAX se stal od svého vzniku velice populární a je oblíbenou a hojně využívanou technologií. Díky svému velkému potenciálu se dnes můžeme setkat s řadou webových aplikací, které tuto technologii plně využívají. Jeho popularita vzrostla hlavně díky společnosti Google, která jej začala používat v mnoha svých aplikacích jako jsou Google Suggest[1] (našeptávač hledaných výrazů – Obr. 1.3), GMail[2] a Google Maps[3].

Obr. 1.3 – *Google Suggest*

S podobnou funkcionalitou jako je Google Suggest se můžeme setkat například i na českém vyhledávacím serveru Seznam.

Internet je plný nejrůznějších aplikací, kde byl AJAX úspěšně nasazen. Je jen na vývojářích, kde a jak ho ve svých webových aplikacích vhodně použijí. Možností využití v různých webových aplikacích je nekonečně mnoho.

[1] www.google.com/webhp?complete=1
[2] www.gmail.com
[3] http://maps.google.com

Některé případy možného použití:

- Dynamická aktualizace jen některých částí webové stránky. Jednoduchým příkladem může být webová stránka obsahující nějakou anketu. Bez AJAXu by musela být znovu načtena celá stránka ihned po hlasování, i když se v ní změnilo jen velmi málo.
- Kontrola zadávaných dat ve formulářích. Díky AJAXu je možné ověřovat data na serveru postupně ihned po zadání jednotlivých položek.
- Našeptávače a funkce automatického dokončování, které jsou obdobou nabízených vyhledávači Google a Seznam.

Kromě těchto několika málo příkladů se na internetu objevuje celá řada zajímavých aplikací využívajících této technologie. Například:

- Plánovací kalendář *Kiko* (www.kiko.com).
- Editory dokumentů jako *ajaxWrite* (www.ajax13.com/en/ajaxwrite), nebo *Zoho Writer* (www.zohowriter.com).
- Vyhledávač zboží na www.yapura.net.
- Komunikační aplikace jako *Meebo* (wwwl.meebo.com), *eBuddy* (www.ebuddy.com), *Kool IM* (www.koolim.com).

AJAX umožňuje vytvářet aplikace, které se svým chováním přibližují klasickým desktopovým aplikacím. Tento fakt vede k tomu, že musíme brát ohled na uživatele, kteří byli doposud zvyklí na klasické chování webových aplikací (požadavek/odpověď). Je důležité, aby bylo uživateli zřejmé, že je jeho požadavek zpracováván, resp. že došlo k aktualizaci některých částí webové stránky. To vedlo k vzniku pomocných technik jako YFT (Yellow Fade Technique). V podstatě jde o to, že změněnou část stránky obarvíme na žluto a tuto barvu pak necháme postupně slábnout. Někomu se může tento přístup zdát až příliš výrazný. Uživatele lze upozornit i jednodušším způsobem, a to např. zobrazením nápisu „Loading…“. Použití těchto pomocných technik záleží čistě na uvážení vývojáře a na druhu webové aplikace.

Díky rozšíření a oblíbenosti vzniklo, a stále vzniká, mnoho knihoven, nástrojů a aplikačních rámců, které usnadňují tvorbu webových aplikací s AJAXem. Tyto nástroje obsahují mnoho zajímavých funkcionalit a díky nim lze celkem jednoduše vytvořit vysoce interaktivní web. aplikace. V síti internet je možné nalézt nesčetné množství nejrůznějších článků, které se týkají problematiky kolem AJAXu. Existuje

řada serverů, které se zabývají vyloženě touto problematikou a které obsahují spoustu informací, užitečných odkazů a návodů. Například:

- www.ajaxian.com
- www.ajaxmatters.com
- www.ajaxprojects.com
- www.ajaxinfo.com

1.2 Objekt XMLHttpRequest

Jádrem AJAXu je objekt XMLHttpRequest umožňující JavaScriptu vytvářet asynchronní požadavky na server. Bez rozšíření podpory tohoto objektu by se dnes o AJAXu pravděpodobně nemluvilo. Původně byl objekt XMLHttpRequest implementován pouze v prohlížeči Internet Explorer 5 společnosti Microsoft v roce 1999. Fakt, že tento prvek nebyl podporován i jinými webovými prohlížeči, vedl k tomu, že jej vývojáři většinou nevyužívali. Tato skutečnost se změnila v době, kdy začal být objekt XMLHttpRequest podporován v prohlížečích Mozilla 1.0 a Safari 1.2. Objekt XMLHttpRequest není standardem W3C. Většina jeho funkcionality je navržena v nové specifikaci DOM Level 3 Load and Save. Je jasné, že pokud se nejedná o standard, může se chování v různých webových prohlížečích lišit. V současné době je implementace objektu XMLHttpRequest v nejmodernějších webových prohlížečích (Firefox, Opera, Internet Explorer, Safari a Konqueror) velmi podobná.[2]

1.2.1 Vytvoření objektu XMLHttpRequest

Dříve než lze odesílat požadavky a zpracovávat odpovědi, je třeba vytvořit instanci objektu XMLHttpRequest. Již v tomto případě narážíme na rozdílnost implementace v různých prohlížečích. Internet Explorer 6 (IE6) a starší verze implementují objekt XMLHttpRequest jako objekt ActiveX. Kdežto u ostatních prohlížečů se jedná o nativní objekt JavaScriptu.

V případě IE6 a starších je možné vytvořit instanci následovně:

```
xmlHTTP = new ActiveXObject("Microsoft.XMLHTTP");
```

U ostatních prohlížečů se instance tvoří tímto způsobem:

```
xmlHTTP = new XMLHttpRequest();
```

Jednoduchá logika pro vytvoření instance

Z výše uvedeného důvodu musí kód JavaScriptu obsahovat jednoduchou rozhodovací logiku, která zajistí vytvoření instance objektu XMLHttpRequest v různých webových prohlížečích.

Vytvoření instance objektu XMLHttpRequest:

```
var xmlHttp;
function vytvorXMLHttpRequest() {
  if (window.XMLHttpRequest) {
    xmlHttp = new XMLHttpRequest();
  }
  else if (window.ActiveXObject) {
    xmlHttp = new ActiveXObject("Microsoft.XMLHTTP");
  }
}
```

Předchozích několik řádků kódu představuje jednoduchou funkci JavaScriptu, která vytvoří instanci objektu XMLHttpRequest nezávisle na prohlížeči. Funkce ověřuje podporu daného objektu v prohlížeči. V případě, že prohlížeč neimplementuje objekt XMLHttpRequest jako nativní objekt JavaScriptu, je tento objekt vytvořen jako nová instance objektu ActiveXObject. Protože Internet Explorer 7 podporuje jak ActiveX, tak i XMLHttpRequest, je doporučeno nejdříve testovat podporu XMLHttpRequest a až poté podporu ActiveX.

Kontrolu, zda-li prohlížeč podporuje objekt XMLHttpRequest, můžeme provést i pomocí funkce `typeof`:

```
if (typeof XMLHttpRequest != "undefined")
  xmlHttp = new XMLHttpRequest();
```

Vytvoření instance pomocí try/catch

Alternativní metoda pro tvorbu instance objektu XMLHttpRequest je uvedena v [3]. Tato metoda je založená na zachytávání a zpracovávání výjimek pomocí `try/catch`. A je zde popsána jako nejlepší způsob jak zajistit, aby metoda dobře fungovala v budoucích prohlížečích a na několika málo řádcích korektně ošetřovala chyby. Pokud dojde v kódu JavaScriptu k chybě, vyhodí se výjimka v podobě objektu

obsahujícího detaily chyby. Tuto výjimku je možné zachytit a zpracovat tak, že není předána do prohlížeče. Syntaxe `try/catch` má následující tvar:

```
try { }
catch (e) { }
```

V bloku `try` je libovolný kód, který může generovat výjimku. V případě chyby se provede kód umístěný v bloku `catch`.

Vytvoření instance objektu XMLHttpRequest metodou `try/catch`:

```
function vytvorXMLHttpRequest()
{
  var xmlHttp;
  // všechny prohlížeče kromě IE6 a starší
  try
  {
    xmlHttp = new XMLHttpRequest();
  }
  catch (e)
  {
    // IE6 nebo starší
    try
    {
      xmlHttp = new ActiveXObject("Microsoft.XMLHTTP");
    }
    catch (e) {}
  }
  if (!xmlHttp)
    alert("Chyba při vytváření objektu XMLHttpRequest.");
  else
    return xmlHttp;
}
```

Lepší objekty pro IE

Jestliže je prohlížečem Internet Explorer 6, je možné se pokusit o použití nejnovější verze prvku ActiveX při tvorbě objektu XMLHttpRequest. Knihovna XMLHttp existuje v mnoha verzích[1]. Dle [3] je nejnovější verze MSXML2.XMLHTTP.6.0. Řešení přináší do funkce, která vytvoří objekt XMLHttpRequest, několik málo řádků navíc.

[1] Podrobnější informace lze nalézt na adrese: http://puna.net.nz/etc/xml/msxml.htm.

Použití nejnovější verze XMLHttp:

```
// IE6 nebo starší
var XMLHttpVersions = new Array('MSXML2.XMLHTTP.6.0',
                                'MSXML2.XMLHTTP.5.0',
                                'MSXML2.XMLHTTP.4.0',
                                'MSXML2.XMLHTTP.3.0',
                                'MSXML2.XMLHTTP',
                                'Microsoft.XMLHttp');
for (var i=0; i<XMLHttpVersions.length && !xmlHttp; i++)
{
  try
  {
    xmlHttp = new ActiveXObject(XMLHttpVersions[i]);
  }
  catch (e) {}
}
```

1.2.2 Metody a atributy

V Tab. 1.1 a Tab. 1.2 jsou uvedeny standardní metody a atributy XMLHttpRequest. K těmto metodám a atributům se přistupuje klasicky pomocí tečkového zápisu. Například:

`xmlHttp.abort();` ,kde `xmlHttp` je instance objektu XMLHttpRequest.

Tab. 1.1 – *Standardní metody XMLHttpRequest*

Metoda	**Popis**
`abort()`	Ukončí aktuální požadavek.
`getAllResponseHeaders()`	Vrátí hlavičky odpovědi jako řetězec.
`getResponseHeader("hlavicka")`	Vrátí jednu hlavičku odpovědi jako řetězec.
`open("metoda", "url")`	Inicializuje parametry požadavku.
`send(obsah)`	Provede požadavek.
`setRequestHeader("hlavicka", "hodnota")`	Nastaví zvolenou hlavičku na zadanou hodnotu.

Tab. 1.2 – *Standardní atributy XMLHttpRequest*

Atribut	Popis
`onreadystatechange`	Používá se jako ukazatel na funkci JavaScriptu, která je vykonána při změně stavu požadavku
`readyState`	Stav požadavku. 0 – neinicializovaný 1 – zavádí se 2 – je zaveden 3 – interaktivní 4 - dokončeno
`responseText`	Vrátí odpověď serveru ve formě řetězce.
`responseXML`	Vrátí odpověď serveru ve formě XML.
`status`	Vrátí stavový kód požadavku získaný od serveru (200 (OK), 404 (Not Found), atd.).
`statusText`	Vrátí zprávu stavu požadavku (OK, Not Found, atd.).

1.2.3 Odeslání požadavku a příjem odpovědi

V předchozích částech byly vysvětleny způsoby vytvoření objektu XMLHttpRequest a byly popsány metody a atributy tohoto objektu. Nyní již lze odeslat požadavek na server a následně přijmout odpověď. Odeslání požadavku může být iniciováno mnoha způsoby, například jednoduchou událostí `onclick` při stisku tlačítka.

Jak odeslat požadavek

Poté, co byl vytvořen objekt XMLHttpRequest, je nutné nastavit parametry následujícího požadavku. Tyto parametry se konfigurují metodou `open`. Tato metoda má dva povinné a několik nepovinných parametrů. První specifikuje metodu, kterou budou zaslána data na server (`GET`, `POST` nebo `PUT`). Druhým je URL v absolutním nebo relativním tvaru, které určuje, kam se požadavek odešle. Třetí parametr udává, zda-li bude požadavek zpracován asynchronně. Hodnota `TRUE` znamená asynchronní přenos (bez čekání na odpověď) a naopak hodnota `FALSE` znamená, že další zpracování bude pokračovat až po přijetí odpovědi.

Požadavek se odešle serveru metodou `send()`. Argument této metody je odeslán jako část těla požadavku. Pokud bude použita metoda `GET`, nebudou v těle požadavku odeslána žádná data a argumentem bude `null`. Mají-li být odeslána nějaká data jako argument metody `send()`, musí být použita metoda `POST`.

Dříve než se pomocí metody `send()` odešle požadavek, je důležité nastavit událost `onreadystatechange` tak, aby byla schopna zpracovat odpověď ze serveru. Pro iniciaci požadavku jsou pak potřeba celkem tři řádky kódu:

```
xmlHttp.open("GET", "test.php", true);
xmlHttp.onreadystatechange = zpracujZmenuStavu;
xmlHttp.send(null);
```

Metoda GET a POST

Jak už bylo řečeno, data se odesílají pomocí metody `GET` nebo `POST`. Ve většině případů potřebujeme serveru předat nějaká data jako součást požadavku. Na základě těchto předaných parametrů pak server vytvoří odpověď. Použije-li se pro předání parametrů metoda `GET`, jsou tyto parametry součástí dotazovaného řetězce URL. Například:

```
xmlHttp.open("GET", "http://localhost/test.php?p1=x&p2=y",
true);
```

Předávány jsou dva parametry, a to `p1` a `p2`.

V případě použití metody `POST` se parametry nepřipojují k URL, ale jsou předávány jako argument metody `send()` takto:

```
xmlHttp.open("POST", "http://localhost/text.php, true);
xmlHttp.send("p1=x&p2=y");
```

Kdy kterou metodu použít, záleží na vývojáři a konkrétním případu. Hlavní rozdíl mezi těmito metodami je, že metoda `POST` odesílá předávané parametry v těle požadavku a metoda `GET` je připojuje k dotazované URL. Z této skutečnosti vyplývá, že metoda `GET` je zcela jistě omezena délkou dat odesílaných v URL. Dle [2] je doporučeno použít metodu `GET`, když zpracování dat nezmění stav datového modelu, což znamená, že by měla být použita při získávání dat. Naopak metoda `POST` je doporučena pro operace, které mění stav datového modelu, např. ukládáním nebo aktualizací dat.

Při použití metody `POST` je nutné nastavit hlavičku `Content - Type` objektu XMLHttpRequest následovně:

```
xmlHttp.setRequestHeader("Content - Type", "application/x-www-
form-urlencoded");
```

Jak přijmout odpověď

V tabulce Tab. 1.2 byly uvedeny dva atributy objektu XMLHttpRequest, které poskytují přístup k odpovědi. Atribut `responseText` poskytuje odpověď ve formě řetězce a atribut `responseXML` poskytuje odpověď ve formě XML.

K odpovědi se většinou přistupuje pomocí funkce zpětného volání, která je vázána na událost `onreadystatechange`. Tato funkce je volána vždy, když dojde ke změně interního stavu požadavku. Stav požadavku udává vlastnost `readyState`, která může nabývat jedné ze čtyř následujících hodnot (také viz Tab. 1.2):
0 – neinicializovaný; 1 – zavádí se; 2 – je zaveden; 3 – interaktivní; 4 – dokončeno.

Jak uvádí [3], některé implementace objektu XMLHttpRequest určité stavové kódy ignorují. Opera odpálí událost pouze pro stavové kódy 3 a 4. A při použití nejnovější verze XMLHttp IE zahlásí stavové kódy 2, 3 a 4.
Typicky vypadá funkce zpracujZmenuStavu tak, jak je naznačeno v následující ukázce.

Funkce zpětného volání:

```
function zpracujZmenuStavu()
{
  //pokračuj, je-li stav požadavku kompletní
  if (xmlHttp.readyState == 4)
  {
    //pokračuj, je-li status OK
    if (xmlHttp.status == 200)
    {
    //získej odpověď jako řetězec
    odpoved = xmlHttp.responseText;
    //zde může být další zpracování odpovědi
    //...
    }
  }
}
```

Ve většině aplikací se k odpovědi přistupuje pokud je stav požadavku kompletní, což znamená, že odpověď byla přijata ze serveru. Navíc se testuje, zda-li má stavový kód požadavku (`status`) získaný od serveru hodnotu ″200″ (OK).

1.3 Používané formáty pro výměnu dat

V jedné z předchozích kapitol bylo řečeno, že odpověď odeslanou serverem lze přijmout formou prostého textu nebo ve formátu XML. Přestože je XML uvedeno v názvu objektu XMLHttpRequest, není podmínkou tento jazyk použít. Pro asynchronní výměnu dat mezi klientem a webovým serverem je možné použít libovolný jiný formát včetně HTML nebo JSON.

1.3.1 XML (eXtensible Markup Language)

XML je velmi rozšířený a používaný značkovací jazyk vyvinutý a standardizovaný konsorciem W3C [4]. Jedná se o jednoduchý otevřený formát založený na prostém textu. Dokumenty XML se podobají dokumentům HTML. Narozdíl od HTML má tento jazyk přísnější syntaxi. Neexistují zde žádné předdefinované značky. Moderní prohlížeče zacházejí s dokumentem XML v souladu se specifikací W3C DOM (více o DOM v 1.4.3).

Ukázka jednoduchého XML dokumentu:

```
<?xml version="1.0" encoding="UTF-8"?>
<studenti>
  <student>
    <jmeno>Jan</jmeno>
    <prijmeni>Kovář</prijmeni>
    <fakulta>M</fakulta>
    <rocnik>5</rocnik>
  </student>
  <student>
    <jmeno>Josef</jmeno>
    <prijmeni>Novák</prijmeni>
    <fakulta>S</fakulta>
    <rocnik>3</rocnik>
  </student>
</studenti>
```

1.3.2 JSON (JavaScript Object Notation)

Jednoduchou a velice často užívanou alternativou k používání XML je formát JSON. Tento formát je založen na podmnožině JavaScriptu. Je člověkem snadno čitelný. Jelikož je tento formát na jazyku nezávislý a v [6] je uveden obsáhlý seznam programovacích jazyků, se kterými jej lze propojit, je JSON vhodný pro výměnu strukturovaných dat mezi různorodými systémy.

JSON je postaven na dvou dobře známých datových strukturách:

- Kolekce párů název/hodnota, které jsou realizované jako objekt.
- Uspořádaný seznam hodnot realizovaný jako pole.

Přehled datových typů, přehlednou grafickou reprezentaci struktury objektu JSON a další informace naleznete v [6]. Následující ukázka představuje příklad zápisu dat ve formátu JSON.

Příklad dat ve formátu JSON:

```
{
  "jmeno": "Josef",
  "prijmeni": "Novák"
  "adresa": {
    "ulice": "Nová 1",
    "mesto": "Liberec"
  },
  "telefon": [
    "111 222 333",
    "444 555 666"
  ]
}
```

Ve srovnání s XML mají data ve formátu JSON často o mnoho menší velikost, což je velkou výhodou.

1.4 Techniky na straně klienta

Pod pojem AJAX spadá několik technologií, které spolupracují na straně klienta. Jednou z technik na straně klienta je i práce s objektem XMLHttpRequest, kterému byla věnována samostatná kapitola. Aby mohl programátor vytvářet webové aplikace s využitím AJAXu, musí tyto jednotlivé technologie dobře znát.

1.4.1 XHTML a CSS

Pro prezentaci v prohlížeči slouží značkovací jazyk HTML, resp. XHTML, a kaskádové styly CSS. Jazyk HTML je základním jazykem pro publikování stránek na internetu. O tomto programovacím jazyku již bylo napsáno mnoho. Specifikace a další informace jsou dostupné na domovské stránce konsorcia W3C (www.w3.org).

Kaskádové styly jsou určeny k formátování HTML, XHTML a XML dokumentů a jsou navrženy organizací W3C. Jejich hlavním smyslem je oddělení vzhledu dokumentu od jeho struktury a obsahu. Umožňují definovat a měnit vzhled webu z jediného souboru (*.css). Kaskádové styly, jako takové, není nutné při tvorbě dynamických webových aplikací používat. Ajaxové webové aplikace budou bez problému pracovat i bez nich. Jejich použití s sebou nese mnoho výhod a je doporučeno. Specifikace a další informace je možné nalézt opět na stránkách W3C.

O HTML a CSS bylo napsáno již mnoho knih a článků na internetu. Jak vytvářet a publikovat stránky pomocí těchto technologií je například v [6].

1.4.2 JavaScript

Základní složkou AJAXu je JavaScript, který vytváří hlavní funkcionalitu na straně klienta. Jedná se o interpretovaný jazyk se schopnostmi objektové orientace, který je podporován všemi moderními prohlížeči. Kód napsaný v JavaScriptu je nahrán do prohlížeče společně s HTML kódem. Lze jej umístit přímo do HTML dokumentu nebo do externího souboru (*.js). Jak je řečeno v [3], součást síly JavaScriptu na straně klienta leží v jeho schopnosti manipulovat s rodičovským HTML dokumentem pomocí rozhraní DOM. Dokumentace k JavaScriptu je k dispozici v [7].

1.4.3 DOM – objektový model dokumentu

Document Object Model je na platformě a jazyku nezávislé rozhraní, které dovolí programům a skriptům dynamicky přistupovat a aktualizovat obsah, strukturu a styl dokumentů. Dokument může být dále zpracován a výsledek tohoto zpracování může být zahrnut zpět do prezentované stránky. Tuto definici uvádí konsorcium W3C na svých webových stránkách (www.w3.org), kde jsou kromě jiného k nalezení i specifikace DOM.

DOM je aplikační rozhraní pro dokumenty HTML a XML a je na programovacím jazyku nezávislý. Má schopnost tyto dokumenty vytvářet, modifikovat, parsovat a vyhledávat v nich. Každý element dokumentu je částí DOM a typicky se k atributům a metodám těchto elementů přistupuje pomocí JavaScriptu. Atributy a metody DOM elementů užitečné pro zpracování těchto dokumentů a jejich popis lze nalézt například v [2] nebo v příloze A.

JavaScript a DOM je možné na straně klienta použít k následujícímu:

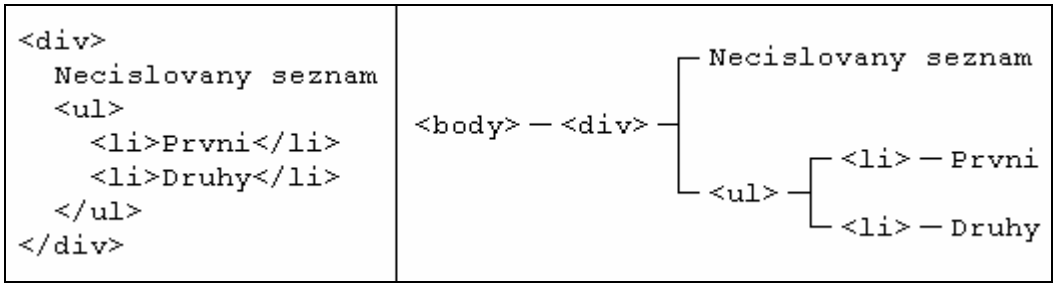
- Zpracování dokumentů XML získaných ze serveru.
- Manipulaci s HTML stránkou během práce uživatele.
- Vytvoření nového dokumentu XML (následně lze odeslat serveru).

1.4.4 Vytváření dynamického obsahu stránky

Díky AJAXu nemusí uživatel během práce čekat, než se celá stránka znovu načte a zobrazí v prohlížeči. Jsou prováděna skrytá asynchronní volání na server a následně na základě odpovědi může dojít ke změně pouze požadované části stránky.

Dynamický obsah stránky pomocí DOM

Jak bylo řečeno výše, DOM lze využít k manipulaci s HTML stránkou během práce uživatele. To znamená, že se dynamicky mění její některé části. Webové stránky jsou reprezentovány v moderních prohlížečích pomocí standardu W3C DOM, a tedy by měly být zobrazeny v různých prohlížečích stejně. DOM si lze představit jako stromovou strukturu stránky. Na obrázku Obr. 1.4 je vpravo zobrazena ukázková struktura, která odpovídá HTML kódu vlevo.



Obr. 1.4 – *Stromová struktura elementů HTML*

Pro vytváření dynamického obsahu je možné použít atributy a metody DOM představené například v [2] a příloze A. V [2] je i několik slov o nekompatibilitě prohlížečů týkajících se těchto metod a atributů.

Vytváří-li se dynamická struktura HTML pomocí DOM, není nutné se starat o formátování textu. Stejně tak není nutné přidávat uzavírací značky manuálně, o to se postará DOM. Uzly lze vytvořit nezávisle a teprve poté rozhodnout o jejich hierarchii.

Dynamický obsah stránky pomocí atributu innerHTML

Jednoduchým způsobem, jak vložit dynamický obsah do stránky, je použití atributu `innerHTML`. Pokud je na straně serveru vytvořen kód HTML jako odpověď na asynchronní volání, lze k této odpovědi přistoupit pomocí atributu `responseText` a následně jednoduše předat prohlížeči pomocí `innerHTML`. Tento atribut je nestandardním atributem elementů HTML, který byl původně implementován pouze v prohlížeči IE. Tento prohlížeč nabízí také jeho nejomezenější podporu. Většina moderních prohlížečů tento atribut podporuje u většiny elementů.

I v tomto případě se znovu využívá DOM. Pomocí metody `getElementById` objektu `document` se přistupuje k jedinečně pojmenovanému elementu (například `<div>`) a dále pomocí atributu `innerHTML` se do tohoto elementu vkládá vytvořený kód HTML.

1.5 Techniky na straně serveru

Předchozí kapitoly byly věnovány technikám na straně klienta, které jsou tou nejdůležitější částí a zajišťují hlavní funkcionalitu webových aplikací s AJAXem. AJAX v podstatě ke své funkci nepotřebuje technologii na straně serveru. Avšak vytváříme-li požadavky na server, potřebujeme také zajistit, že tyto požadavky budou na serveru zpracovány a vyřízeny. K tomu potřebujeme právě některou ze serverových technologií.

Vzhledem k tomu, že AJAX je technika pracující na straně klienta, bude fungovat bez ohledu na to, jakou technologii na serveru používáme, ať už je to Java, .NET, Ruby, PHP nebo CGI. [2]

Na straně serveru není důležité, která z technologií bude použita. Je důležité, aby zde byla vytvořena patřičná funkcionalita, která správně zpracuje požadavek a vytvoří odpovídající odpověď, která bude následně v klientovi zpracována.

1.6 Připojení ke vzdáleným serverům

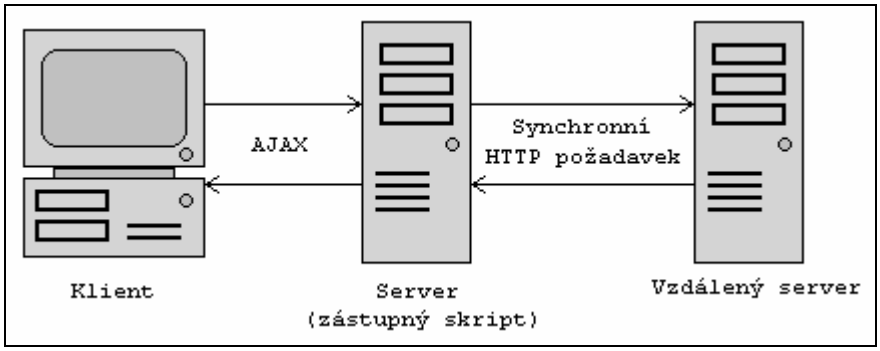
Je důležité se zmínit i o bezpečnosti. Objekt XMLHttpRequest spadá do kategorie zabezpečení prohlížeče. Jakýkoli prostředek požadovaný objektem XMLHttpRequest se musí nacházet ve stejné doméně, ze které pochází volající skript [2]. Jakýkoli cizí server může být nebezpečný a různé prohlížeče se v tomto případě chovají různě.

Dalo by se říci, že asi nejméně bezpečným je v tomto ohledu Internet Explorer, který používá bezpečnostní model založený na zónách. Pokud tento prohlížeč zaznamená požadavek na jiný server, zobrazí dialogové okno s varováním, v němž se uživatele dotazuje na potvrzení této akce[1].

Oproti tomu prohlížeče Firefox[2] a Opera takovýto požadavek přeruší a zobrazí chybové hlášení.

[1] Při standardním nastavení.
[2] Pro prohlížeč Firefox existují některé triky, jak takový požadavek umožnit, ale nedoporučují se.

Pokud opravdu potřebujeme získat nějaká data ze vzdáleného serveru, můžeme toto obejít a použít jiné řešení – zástupný skript. Na Obr. 1.5 je vysvětlen princip zástupného skriptu.



Obr. 1.5 – *Použití zástupného skriptu*

Princip je naprosto jednoduchý. Klient (webový prohlížeč) odesílá asynchronní požadavky na server, kde je zástupný skript, který provede synchronní požadavek na vzdálený server. Poté, co získá požadovaná data, tyto data zpracuje, vytvoří odpověď a odešle klientovi.

2 MOBILNÍ WEB

2.1 Vývoj mobilního webu

V roce 1998 byl organizací WAP Forum definován standard **WAP 1.0** (Wireless Application Protocol). Tento standard byl určen pro bezdrátovou komunikaci a jeho úkolem bylo zpřístupnit internet v mobilních zařízeních, jako jsou mobilní telefony nebo PDA. WAP je ekvivalentem k protokolům v síti internet a je využíván téměř po celém světě[1] k zobrazování tzv. WAP stránek. Tyto stránky jsou vytvářeny pomocí jazyka **WML** (Wireless Markup Language) a přistupuje se k nim pomocí WAP prohlížeče. WML je značkovací jazyk založený na XML, z čehož vyplývá, že má podobnou strukturu jako např. HTML. Obsahuje určitá specifika vyplývající z jeho určení pro mobilní zařízení. WML je implementován v naprosté většině mobilních telefonů.

WAP Forum se stalo součástí Open Mobile Alliance (OMA). Tato organizace vznikla v roce 2002 a stará se o otevřené standardy pro oblast mobilních zařízení.

Nástupcem WAP 1.0 se stal **WAP 2.0**, který je zpětně nekompatibilní. WAP 2.0 přináší namísto WML **eXtensible Hypertext Markup Language Mobile Profile** (XHTML MP) spolu s mobilní specifikací kaskádových stylů **WCSS** (Wireless CSS). WCSS je podmnožinou CSS (definováno W3C). WAP 2.0 je definovaný organizací OMA a kromě změny primárního značkovacího jazyka přináší i podporu internetových komunikačních protokolů, jako TCP/IP a HTTP. XHTML MP je vytvořen speciálně pro mobilní zařízení a je odvozen z XHTML. Jedná se o rozšíření XHTML Basic specifickými doplňky. XHTML Basic je podmnožina XHTML (oba jsou definovány W3C) určená pro zařízení s limitovanými schopnostmi, jako jsou mobilní telefony. Protože je XHTML MP nadřazeno XHTML Basic, zařízení, podporující XHTML MP, podporují i XHTML Basic. Implementace pouze XHTML Basic je vzácná. Specifikace technologií WAP lze nalézt na domovských stránkách OMA (www.openmobilealliance.org).

V roce 2005 se do vývoje mobilního webu zapojilo i konsorcium W3C, a to tím, že založilo Mobile Web Initiative (MWI), pod kterou spadá několik pracovních skupin.

[1] V Japonsku je populárním standardem i-mode od místního převládajícího telefonního operátora DoCoMo.

Jednou z nich je pracovní skupina Mobile Web Best Practices (MWBP), jejímž úkolem je vyvíjet specifikaci Mobile Web Best Practices 1.0[1] , která obsahuje doporučení, jak vytvářet webové stránky pro mobilní zařízení. Podle MWBP 1.0 musí být tyto stránky vytvořeny pomocí XHTML Basic. MWBP definuje na základě těchto doporučení sadu testů W3C MobileOK Basic Tests 1.0[2] pro webové stránky určené mobilním zařízením. W3C připravuje dvě úrovně testů – základní MobileOK Basic a přísnější MobileOK. Testy pro MobilOK ještě nejsou definovány. OMA a W3C spolupracují na mobilní verzi kaskádových stylů – CSS Mobile Profile.

V roce 2006 byla vytvořena generická doména nejvyššího řádu (top-level domain, TLD) **.mobi** (dotMobi), určená výhradně pro mobilní zařízení. Tato doména byla terčem častých kritik, protože byla v nesouladu s W3C. W3C prosazuje, že všechny TLD musí být nezávislé na koncovém zařízení. Reakcí bylo navázání spolupráce s W3C. Webové stránky dotMobi musí být vytvořené v XHTML MP 1.0 nebo pozdější verzi. Více informací a jak vytvářet web. stránky pro doménu dotMobi je na http://pc.mtld.mobi.

V současné době se objevuje řada moderních mobilních prohlížečů, které se svými schopnostmi přibližují svým starším sourozencům z desktopu. Tyto prohlížeče disponují již širokou podporou standardů (včetně nejnovějších technologií jako AJAX), které jsou běžně používány pro tvorbu webových stránek určených desktopovým osobním počítačům. Takovéto mobilní prohlížeče umožňují zobrazit jakoukoli webovou stránku v mobilním zařízení. Více o některých těchto prohlížečích v následující kapitole.

[1] www.w3.org/TR/mobile-bp
[2] www.w3.org/TR/mobileOK-basic10-tests

2.2 Rozšířené moderní mobilní prohlížeče

Tyto prohlížeče jsou určené pro tzv. chytrá mobilní zařízení, jako jsou smartphony a kapesní počítače (PPC, PDA). Tato zařízení jsou vybavena otevřeným operačním systémem (Symbian, Windows Mobile, …), kde jsou tyto prohlížeče součástí software, nebo je lze doinstalovat.

2.2.1 Opera

Společnost Opera Software nabízí webové prohlížeče pro mnoho druhů zařízení od osobních počítačů, mobilních telefonů až po hráčské konzole. Je asi neznámější společností nabízející webové prohlížeče určené pro mobilní zařízení. Opera nabízí uživatelům dva různé mobilní prohlížeče.

Opera Mini

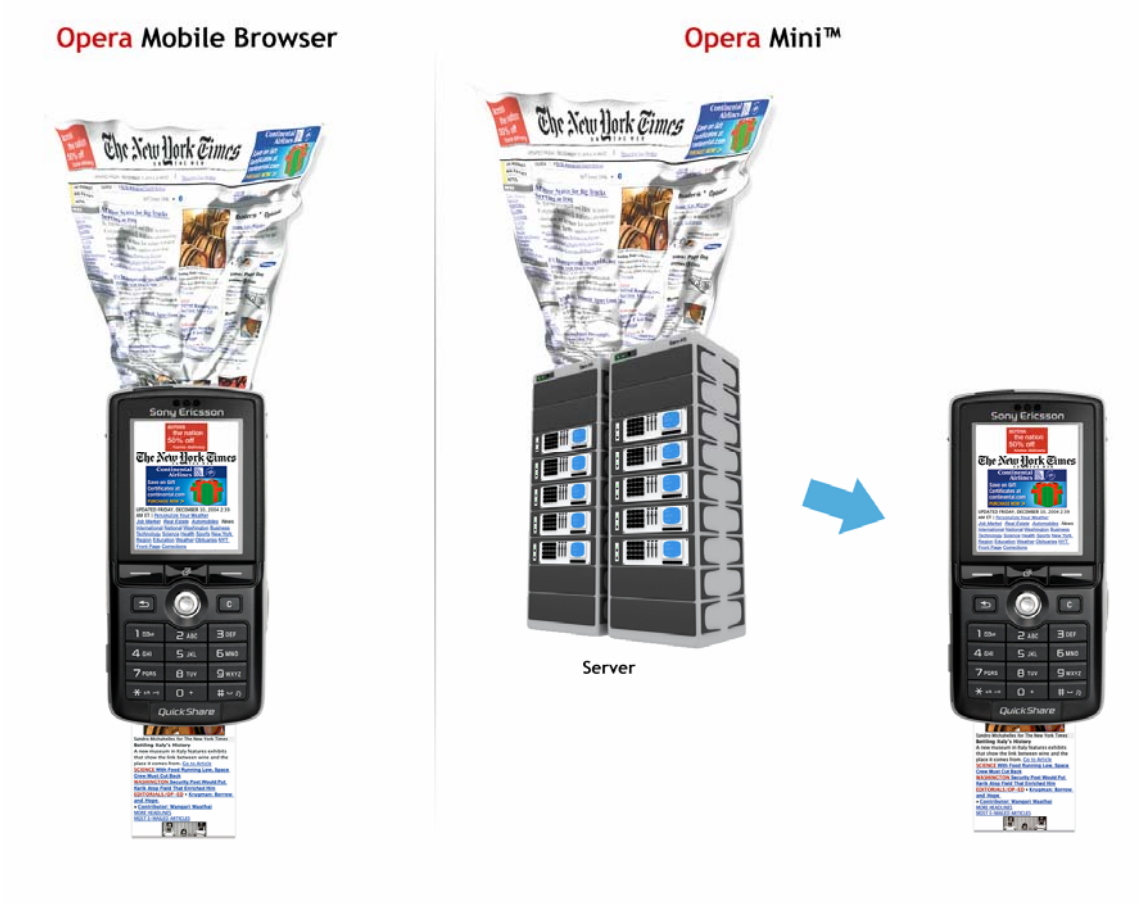
Tato verze mobilního prohlížeče je určená pro mobilní telefony využívající technologii Java (v Čechách jsou tyto mobilní telefony zatím stále nejrozšířenější). Opera Mini využívá vzdáleného serveru, který požadovanou webovou stránku upraví (komprese, redukce velikosti, …) tak, aby ji bylo možné zobrazit. Na Obr. 2.1 je ukázka zobrazení webové stránky vyhledávače Google (www.google.com) v prohlížeči Opera Mini. Poslední verze tohoto prohlížeče je 3.1 a je zdarma ke stažení na stránkách www.operamini.com, kde jsou k dispozici i podrobnější informace.

Obr. 2.1 – *Zobrazení vyhledávače Google v prohlížeči Opera Mini*[1]

[1] Zdroj: www.opera.com.

Opera Mobile

Druhým prohlížečem, který nabízí Opera, je Opera Mobile. Tento prohlížeč je v podstatě zmenšenou verzí prohlížeče používaného v osobních počítačích, což znamená, že by měl podporovat stejné technologie a standardy. V porovnání s prohlížečem Opera Mini (viz Obr. 2.2), umožňuje Opera Mobile zobrazit jakoukoli plnohodnotnou webovou stránku přímo v prohlížeči. Stránky jsou optimalizovány pro malý displej pomocí Small-Screen Rendering (SSR)[1].



Obr 2.2 – *Porovnání zpracování stránek v Opera Mobile a Opera Mini*[2]

[1] SSR lze v prohlížeči vypnout a stránka je zobrazena tak, jak je uživatel zvyklý z prohlížeče na PC.
[2] Zdroj: www.opera.com.

Small-Screen Rendering

Tato technologie vznikla proto, aby byl prohlížeč schopen zobrazit takové webové stránky, které nejsou navrženy s vědomím, že by mohly být zobrazovány i v mobilních zařízeních (v zařízeních s malým displejem). SSR[1] takovéto stránky automaticky upraví a jsou následně zobrazeny.

Primárním zájmem je odstranění nutnosti horizontálního scrolování, což znamená, že obsah bude tak široký, jak je široká obrazovka. SSR při zpracování stránky potlačuje některé aspekty zobrazení stránky: rozvržení, obrázky, tabulky, rámy, velikost písma, zarovnávání. Je-li stránka členěna do několika sloupců, SSR toto rozvržení upraví do jednoho sloupce. Obsah je v tomto sloupci seřazen pod sebou podle toho, jak jsou tyto části řazeny ve zdrojovém kódu.

Velké obrázky SSR upraví na vhodnou velikost a některé vůbec nezobrazí. Obrázky na pozadí nejsou zobrazeny a barva pozadí je nastavena na světle šedou. Barva textu je nastavena na černou.

Na velkých obrazovkách se běžně používá velký rozsah velikosti písma. Naopak na malých obrazovkách je důležité, aby byl text dobře viditelný a čitelný. SSR používá tři velikosti písma: normální pro nadpisy, velká pro hlavní nadpisy a střední pro podnadpisy.

Veškerá zarovnání a pozicování obsahu (absolutně umístěný a plovoucí obsah) jsou ignorována.

Jestliže jsou pro stránku definovány CSS styly určené mobilním zařízením (`media="handheld"`), prohlížeč Opera Mobile nepoužije pro zobrazení SSR a považuje webovou stránku za optimalizovanou pro mobilní zařízení.[8]

Opera Mobile je v současné době ve verzi 8 (podporuje AJAX) k dispozici pro mobilní zařízení (PPC, smartphone) s operačním systémem Windows Mobile (Windows Mobile 2003, Windows Mobile 5 a Windows Mobile 6) a Symbian S60. Na stránkách www.opera.com je možné stáhnout zkušební verze tohoto prohlížeče a případně zakoupit i verzi úplnou.

Po nedávném vypuštění desktopové verze Opera 9, byl společností Opera oznámen i brzký příchod mobilní verze tohoto prohlížeče, která přinese mimo jiné podporu widgets v mobilních zařízeních a inteligentní zoom.

[1] Desktop verze Opery umožňuje simulaci SSR (Shift+F11).

2.2.2 Mobilní Internet Explorer

Již z názvu je zřejmé, že tento mobilní prohlížeč je upravenou verzí prohlížeče Internet Explorer od společnosti Microsoft. Tento prohlížeč je standardně součástí softwarové výbavy mobilních zařízení, které mají jako operační systém některou z verzí mobilních Windows. Podrobné informace o mobilním IE a verzích operačního systému Windows pro mobilní zařízení jsou na stránkách společnosti Microsoft (www.microsoft.com).

2.2.3 Nokia prohlížeč pro S60

Tento mobilní prohlížeč od společnosti Nokia je součástí výbavy každého mobilního telefonu Nokia s OS Symbian S60. V říjnu roku 2006 došlo k odhalení nového prohlížeče, který je součástí S60 3rd Edition, Feature Pack 1. Tento prohlížeč již představuje velice výkonný mobilní prohlížeč s širokou podporou standardů, včetně AJAXu. V dubnu 2007 Nokia oznámila příchod nové verze softwaru – S60 3rd Edition, Feature Pack 2, která přinese podporu mobilních widgets. Více informací na www.s60.com.

2.2.4 Minimo

Název tohoto mobilního prohlížeče je odvozen od „Mini Mozilla“, takže je zřejmé, že se jedná o jakousi zmenšenou mobilní verzi prohlížeče Mozilla, který je používán na PC. Projekt Minimo vzniká pod neziskovou organizací Mozilla Foundation, která podporuje a řídí open source projekt Mozilla. Hlavním vedoucím, který vede tento projekt od jeho vzniku, je Doug Turner. Tento prohlížeč je určen pro mobilní zařízení s OS Windows Mobile. V březnu byla uvolněna verze 0.2. Budoucnost tohoto projektu není zcela jasná. Koncem loňského roku Turner na svém blogu uznal pomalý vývoj tohoto prohlížeče a odhalil, že vývojáři Mozilly zkoumají alternativy pro prohlížení webu v mobilních zařízeních. Současně vyzval kolegy, jestli by někdo nechtěl pokračovat v jeho stopách s projektem Minimo.[9]

2.3 Tvorba mobilního webu

Projektování pro mobilní zařízení je odlišné od projektování pro desktopové osobní počítače. Při návrhu webových stránek pro mobilní zařízení je důležité pamatovat na tyto odlišnosti a některá specifická omezení takovýchto zařízení.

2.3.1 Specifické vlastnosti mobilních zařízení

- **Malý displej** – mobilní zařízení jsou někdy jednoduše označována jako zařízení s malou velikostí obrazovky. Tato velikost je pro různé typy zařízení dosti odlišná. Některé běžné velikosti (šířka x výška v pixelech):

128 x 128 px	320 x 240 px
176 x 208 px	352 x 416 px
176 x 220 px	640 x 200 px
208 x 320 px	640 x 320 px
240 x 320 px	640 x 480 px

- **Omezený uživatelský vstup** – mobilní zařízení obvykle nemají úplnou klávesnici. Disponují pouze velmi omezenou. Některá zařízení, jako PPC, mají klávesnici virtuální (viz Obr. 2.3). Převážně se ovládají pomocí pera a mají jen pár tlačítek.

Obr 2.3 – *Virtuální klávesnice kapesního počítače*

- **Rychlost procesoru** – zejména kvůli velikosti, životnosti baterie a ceně jsou v mobilních telefonech používány pomalé procesory. Kapesní počítače mívají obvykle procesory o něco lepší.

- **Paměť** – mobilní zařízení se většinou vyznačují velmi malou pamětí. Paměť a úložiště dat je jedno a totéž.
- **Rychlost sítě** – hlavně z pohledu mobilního webu je rychlost sítě velmi důležitým faktorem. V podstatě jde o dvě důležité vlastnosti – přenosovou rychlost (bandwith) a latenci neboli zpoždění. Přenosová rychlost udává jaké množství dat lze přenést v daném čase v síti. Dvojnásobně velký soubor zabere dvojnásobek času. Obvykle se udává v bitech za sekundu (b/s nebo bps). Přenosová rychlost je ve skutečnosti o mnoho nižší než její teoretická hodnota. Latence znamená dobu čekání mezi odesláním žádosti a získáním odpovědi (cesta na server a zpět). Toto zpoždění je závislé na síti a je stejné pro jakkoli velký objem požadovaných (přenášených) dat. V bezdrátových sítích může být toto zpoždění až několik sekund.

Tyto specifické vlastnosti jsou zpracovány z [10].

2.3.2 Tipy a doporučení

Na základě omezených možností mobilních zařízení vzniká řada tipů, rad a doporučení jak na tvorbu mobilního webu a čeho se vyvarovat. Těchto doporučení lze nalézt na internetu mnoho a některá tato doporučení zpracovaná z [8] a [10] jsou popsána níže.

- Rozvržení stránky by mělo být navrženo do jednoho sloupce, kde nejdůležitější informace jsou umístěny blízko vrcholu stránky. Stránka by měla obsahovat jen to, co je pro uživatele nejdůležitější, a veškeré přebytečné informace by měly být odstraněny.
- Horizontální scrolování není v mobilních prohlížečích vhodné. Nejlepší způsob je navrhnout stránku tak, aby se přizpůsobovala velikosti obrazovky[1]. Používat tabulky, rámy a absolutní pozicování není dobrý nápad.
- Odstranit nepotřebný a přebytečný kód (jak HTML, tak CSS).
- Používat co nejméně obrázků a na pozadí je nepoužívat vůbec. Obrázky potřebují hodně paměti a představují zvětšení objemu přenášených dat, což se odrazí v rychlosti zobrazení stránky a také i v ceně (významné pro uživatele platícího za množství přenesených dat). Důležité je vždy specifikovat velikost použitého

[1] Stránka by se měla přizpůsobovat šířce obrazovky.

obrázku, a to buď přímo v HTML kódu, nebo pomocí CSS. Stejně tak důležité je i použití alternativního textu. Uživatel si může ve svém prohlížeči stahování obrázků zakázat.

- V CSS je doporučeno používat relativní jednotky (např. `width: 50%`) namísto absolutních (např. `width: 100px`).
- Vyhnout se používání tzv. „pop-up“ oken a dynamickým menu.
- Vzhledem k omezeným uživatelským vstupům se nelze zcela spolehnout na události, které jsou běžně používané na PC. Protože mobilní zařízení nemají myš, události jako `mouseover`, `mouseout`, `mouseup` a `mousedown` nelze použít. Stejně tak `doubleclick` nebo `rightclick`. Události vyvolané klávesnicí (např. `keyup`, `keydown`) nemusí také fungovat, jak by se očekávalo. Jedinou událostí, na kterou se dá spoléhat, je událost `click`.
- Příliš složité skripty a stránky mohou výrazně zatěžovat procesor mobilního zařízení. Není doporučeno používat rozsáhlé procesy na pozadí. Některé procesy jako manipulace s DOM a překreslování stránky mohou procesor výrazně zatížit. Složité a graficky bohaté stránky užívají více paměti.

2.4 Testování

Jestliže vytváříme nějakou aplikaci, je nezbytné ji v průběhu vyvíjení neustále testovat. Neméně důležité je i testování její finální verze. Vezmeme-li v úvahu mobilní zařízení a vývoj mobilních webových aplikací, zjistíme, že testování není tak jednoduché. Pokud chceme testovat webovou aplikaci pro PC, v podstatě nám postačí jeden počítač a několik nainstalovaných prohlížečů. Mobilní zařízení však představují mnoho typů těchto zařízení, různé operační systémy a mnoho druhů mobilních webových prohlížečů.

Ideální možností kde testovat mobilní aplikace, je testovat je přímo na samotném mobilním zařízení. To však nemusí být vždy realizovatelné. Zvlášť pokud je zapotřebí několika různých druhů/typů těchto zařízení. Dále (kap. 2.4.1 a kap. 2.4.2) budou zmíněny některé alternativní možnosti testování mobilních web. aplikací.

2.4.1 Emulátory

První možností jsou emulátory[1]. Na internetu jsou k dispozici emulátory některých mobilních prohlížečů, které umožňují testovat obsah web. stránky vytvořený pomocí WML nebo XHTML MP. Emulátory nejmodernějších mobilních prohlížečů ve většině případů neexistují. Pokud je tedy zapotřebí testovat v těchto prohlížečích, je nutné použít nějaký emulátor mobilního zařízení, do kterého by tyto prohlížeče byly následně nainstalovány.

Microsoft Device Emulator 1.0

Tento emulátor nabízí pro testovací účely společnost Microsoft a standardně je součástí vývojového balíku Visual Studio 2005. V této verzi jde o samostatný software. Součástí emulátoru je PPC a smartphone s OS Windows Mobile 5 (viz Obr. 2.4).

Obr. 2.4 – *PPC a smartphone v Microsoft Device Emulator 1.0*

Emulátor je ke stažení na stránkách společnosti Microsoft (www.microsoft.com).

[1] Emulátor – software, který umožňuje běh programu na jiné platformě, než pro kterou je tento program určen.

2.4.2 Vzdálený přístup k mobilním zařízením

V síti internet existují služby, které umožňují vzdálené připojení k mobilním zařízením. Na těchto zařízeních je pak možné testovat vytvořené mobilní aplikace.

Nokia - Remote Device Access

Na adrese http://apu.ndhub.net se nachází služba Remote Device Access (RDA), kterou nabízí Nokia zdarma registrovaným uživatelům v rámci svého fóra [11]. K připojení k této službě je třeba pouze standardní webový prohlížeč a Java™ Web Start (typicky je součástí Java Runtime Environment). Uživatel má na výběr z několika mobilních telefonů Nokia s OS Symbian S60 (viz Obr. 2.5).

Obr. 2.5 – *Výběr mobilního telefonu pro vzdálený přístup*

Po připojení má uživatel k dispozici obraz displeje v reálném čase a klávesnici pro obsluhu mobilního telefonu (viz Obr. 2.6). Mobilní telefony jsou v offline módu, všechny mají k dispozici připojení k internetu a je možné do nich instalovat aplikace a přenášet soubory.

Obr 2.6 – *Uživatelské rozhraní pro práci se vzdáleným mob. telefonem*

Podobná služba je nabízena například na adrese http://deviceanywhere.com. Je však již plně placená.

3 MOBILNÍ AJAX

Mobilní AJAX, neboli využití AJAXu v mobilních zařízeních, má velký význam. AJAX představuje možnost načítání pouze částí dokumentu, což vede ke snížení objemu přenášených dat. Webová aplikace pak rychleji reaguje a celkově se uživateli může zdát stahování webového obsahu rychlejší.

3.1 Použití v mobilním prohlížeči

Jak bylo naznačeno v kapitole 2, mobilní web prošel od svého vzniku velkým vývojem. Současně dostupné nejmodernější mobilní prohlížeče se svými schopnostmi, hlavně podporou běžně užívaných webových standardů a technologií, téměř vyrovnávají desktopovým prohlížečům. Všechny mobilní webové prohlížeče (mimo prohlížeče Opera Mini) uvedené v kapitole 2.2 podporují AJAX. Jsou jimi Opera Mobile, mobilní IE, prohlížeč Nokia S60 a Minimo. Dále je zde důležité zmínit i další známé mobilní prohlížeče, která disponují podporou pro tuto technologii. Jedním z nich je mobilní prohlížeč NetFront od společnosti ACCESS (www.access-company.com), jehož aktuální verze 3.4 podporuje AJAX. Druhým zde zmiňovaným mobilním prohlížečem je prohlížeč Openwave (www.openwave.com), který podporuje AJAX ve své poslední verzi označené „Mercury Edition“.

První možností jak AJAX použít v mobilních zařízeních je stejný jako na osobních počítačích. K vytvořené webové aplikaci se přistupuje pomocí mobilního prohlížeče. Postupy a použití technologií je tedy stejné. Tato problematika byla popsána v kapitole 1. Důležité je aplikaci navrhovat s vědomím, že je vytvářena pro mobilní zařízení, a držet se některých rad a doporučení (viz kap. 2.3).

Použití existujících aplikačních rámců (frameworků), jako *Prototype* (www.prototypejs.org) nebo *script.aculo.us* (http://script.aculo.us), určených pro usnadnění práce s AJAXem, není v případě mobilních aplikací možné. Zejména kvůli jejich zaměření na desktopové prohlížeče a jejich obsáhlé funkční možnosti (např. drag and drop), které nelze v mobilních zařízeních použít.

3.2 Widgets

Součástí dnešního vyspělého internetu jsou widgets, což jsou malé aplikace typicky vytvořené použitím existujících webových technologií (XHTML, CSS, JavaScript, …) včetně AJAXu. Tyto malé aplikace jsou dnes dobře známé z osobních počítačů. Na Obr. 3.1 je ukázka takové widget aplikace.

Obr. 3.1 – *Widget – hodiny*

Widgets na rozdíl od klasických webových aplikací nejsou načteny po zadání URL v prohlížeči. Jedná se o takové webové aplikace, které jsou nainstalovány přímo v zařízení. To znamená, že všechny zdrojové soubory (CSS, JavaScript, …) a obrázky jsou již v zařízení a nemusí být stahovány.

3.2.1 Opera Platform

Vůbec první podporu AJAXu v mobilních zařízeních přinesla společnost Opera, a to již na konci roku 2005, kdy představila svůj produkt nazvaný „Opera Platform“. Pomocí vývojového balíčku Opera Platform Software Development Kit (SDK) je možné vytvářet malé webové aplikace použitím existujících běžných webových technologií. Tyto aplikace je pak nutné do mobilního zařízení nainstalovat. Výhodou SDK je zpřístupnění přirozené funkcionality mobilního zařízení (zprávy, kalendář, …). Opera Platform podporuje tyto OS: Symbian S60, Windows Mobile 2003 SE a Smartphone. V rámci SDK jsou k dispozici nástroje pro vytvoření instalačních balíčků (*sis pro S60, *cab pro Windows Mobile).

Tento projekt byl velmi zajímavý. V podstatě šlo o první mobilní widgets. Bohužel přibližně v půlce roku 2006 přestal jevit známky činnosti. Byla vydána pouze Beta verze Opera Platform SDK.[12]

3.2.2 Mobilní widgets

V nedávné době došlo k několika významným událostem v této oblasti. Několik poskytovatelů mobilních webových prohlížečů oznámilo příchod podpory mobilních widgets.

V březnu 2007 oznámila společnost Opera brzký příchod mobilního prohlížeče Opera Mobile 9, který bude podporovat widgets.

O něco později v dubnu tohoto roku Nokia oznámila, že platforma S60 bude podporovat widgets v následujícím vydání (S60 3rd Edition, Feature Pack 2) prostřednictvím Web Run-Time Development Environment. Což znamená, že novější chytré mobilní telefony vybavené touto verzí OS budou již widgets podporovat. Nástroje, dokumentace a SDK by měly být k dispozici na podzim 2007.

Tyto dvě společnosti nebyly jediné, které oznámily podporu widgets pro mobilní zařízení. Společnost ACCESS oznámila v únoru tohoto roku vývoj nové verze mobilního prohlížeče – NetFront 3.5, který přinese podporu těchto malých webových aplikací. Další společností, která oznámila podporu widgets v březnu, je Openwave.

Mobilní widgets jsou, stejně jako desktopové, vytvořeny pomocí běžných webových standardů a technologií. XHTML a CSS jsou použity pro prezentaci, JavaScript widget oživuje a AJAX poskytuje prostředky pro asynchronní komunikaci se vzdálenými servery na pozadí. Velkou výhodou mobilních widgets je skutečnost, že jsou umístěny přímo v mobilním zařízení. Nemusí se stahovat prostřednictvím sítě. Pomocí AJAXu lze přenášet jen potřebná data. Ve většině případů mohou widgets přistupovat k API mobilních zařízení, což je další velkou výhodou. Díky tomu lze widget propojit např. s fotoaparátem a s dalšími aplikacemi, jako seznam kontaktů nebo kalendář.

3.3 Další možnosti

Oblast používání AJAXu se teprve vyvíjí a rozrůstá. Je pravděpodobné, že vznikne nebo již vzniká mnoho nástrojů, které umožní vytvářet aplikace pro mobilní zařízení využívající AJAX.

Jedním z těchto nástrojů je mojax, což je framework pro AJAX aplikace pro mobilní zařízení. Tradičně vývojáři kombinují webové standardy a technologie k vytváření AJAX webových aplikací. Mojax tyto způsoby aplikuje do J2ME MIDP aplikací pro mobilní zařízení. Tento projekt je ve fázi vývoje a zatím se jedná jen o Beta verzi a lze nalézt na http://mojax.mfoundry.com.

4 UKÁZKOVÁ APLIKACE

Součástí tohoto projektu bylo vytvoření funkční webové aplikace určené pro mobilní zařízení (chytré telefony, kapesní počítače, …), přičemž tato aplikace měla nějakým způsobem využívat asynchronní komunikaci, kterou poskytuje AJAX. Vzhledem k možnostem, které nabízí současná situace mobilního AJAXu (viz kap. 3), bylo zamýšleno vytvořit takovou webovou aplikaci, která bude využívat běžně užívaných webových standardů a bude určena dostupným moderním mobilním prohlížečům, které podporují AJAX (viz kap. 2.2).

4.1 Návrh aplikace

V prvé řadě bylo důležité rozmyslet, co by vlastně vytvořená aplikace měla dělat. Mělo jít o nějaké vhodné použití a takovou aplikaci, která by se eventuelně dala i na mobilních zařízeních nějakým způsobem využít. Nakonec byl stanoven plán vytvořit vyhledávač zboží. Uživatel nosí mobilní zařízení neustále při sobě, proto by se mu takováto aplikace mohla při výběru zboží a rozhodování o jeho koupi hodit.

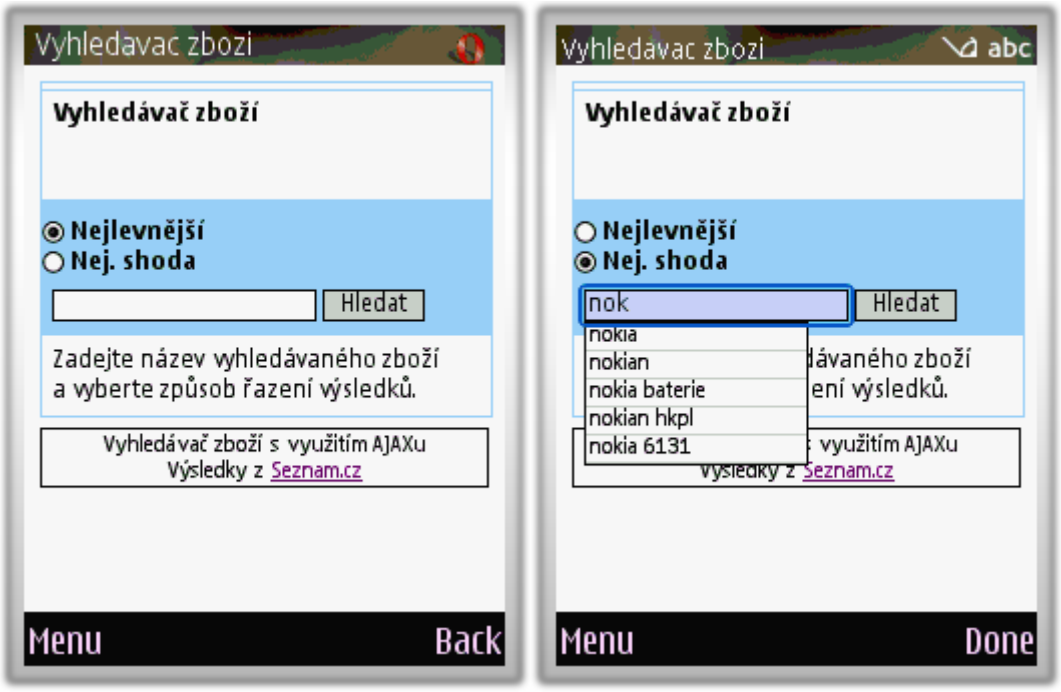
Nyní již nezbývalo nic jiného, než rozhodnout jakou roli v této aplikaci bude hrát AJAX. Funkcionalita vyhledávačů zboží je všem zřejmá. Tyto služby jsou běžně k dispozici na internetu. Pokud by se však uživatel rozhodl použít některý z těchto vyhledávačů, jako například vyhledávač zboží na serveru www.seznam.cz ve svém mobilním zařízení, představovalo by to pro něj dlouhé čekání, než se mu celá tato webová služba zobrazí v prohlížeči. Navíc pro každý vyhledávaný výraz by se musel znovu načíst celý obsah této stránky, včetně výsledků vyhledávání. Z těchto důvodů měla být vytvářená aplikace jednoduchá a úloha AJAXu byla zřejmá. Asynchronního přenosu dat se využije pro získávání výsledků hledání zboží. To ušetří množství přenášených dat. Při prvním vyhledávání budou tyto výsledky zobrazeny v prohlížeči a při každém dalším požadavku na vyhledávání dojde pouze k aktualizaci této části s výsledky. Což znamená, že jediné, co se uživateli v prohlížeči změní, budou zobrazené výsledky.

Dost často jsou součástí těchto vyhledávačů tzv. našeptávače, které se uživateli snaží nabídnout vyhledávaný výraz, který zadává. Tato funkcionalita by měla uživateli urychlit zadávání nebo přinejmenším napovědět vyhledávaný výraz. Jak bylo řečeno

v kap. 1.1.2, je našeptávač typickou funkcionalitou využívající AJAX. Vezmou-li se v úvahu omezené možnosti a rychlost zadávání na mobilních zařízeních, jedná se o funkcionalitu, která by byla velkým přínosem. Proto se jedním z cílů při vytváření ukázkové aplikace – vyhledávače zboží stal i pokus o vytvoření obdobné funkcionality.

4.2 Výsledná aplikace

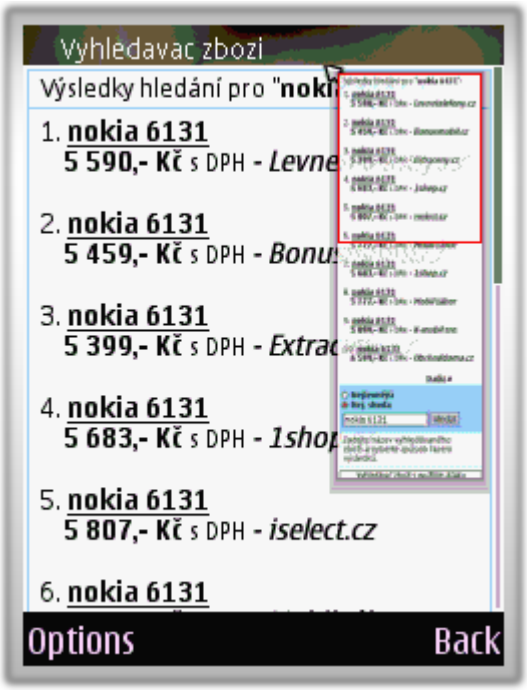
Dříve než bude vysvětlena samotná realizace, je vhodné ukázat jak vypadá výsledná aplikace. Na Obr. 4.1 je nalevo náhled na vytvořenou aplikaci ihned po načtení v mobilním prohlížeči[1] a napravo náhled na funkci našeptávače při zadávání vyhledávaného výrazu.



Obr. 4.1 – *Náhled na vzhled ukázkové aplikace*

Výsledky vyhledávání zboží jsou zobrazeny v horní části, jak je naznačeno na Obr. 4.2[2]. Tento způsob byl zvolen záměrně, aby uživatel po shlédnutí výsledků dospěl až k vyhledávacímu formuláři a nemusel přejíždět přes celý obsah zpět.

[1] V tomto případě Opera Mobile.
[2] Výsledky v prohlížeči Nokia S60.



Obr. 4.2 – *Zobrazení výsledků vyhledávání ve vyhledávači*

4.3 Realizace

Použité technologie:

- Jazyk XHTML 1.0 a kaskádové styly CSS pro prezentaci dat.
- JavaScript pro práci s objektem xmlHttpRequest a dynamické změny pomocí DOM.
- Na straně serveru byl pro vyřizování požadavků použit skriptovací jazyk PHP5.[1]

Pomocníkem pro psaní zdrojového kódu se stal editor PSPad, který je zdarma ke stažení například na www.stahuj.cz. Díky použití tohoto editoru bylo programování o mnoho snažší.

4.3.1 Řešení na straně serveru

Jedním z prvních problémů, které bylo nutné vyřešit, bylo kde vyhledávat zboží a kde brát návrhy pro našeptávač, když nejsou k dispozici příslušné databáze. Tento problém se vyřešil poměrně snadno, a to využitím vyhledávače zboží na serveru www.seznam.cz.

[1] Pro účely testování PHP skriptů byl nainstalován server Apache 2 (http://httpd.apache.org).

Celkem snadno lze z URL v prohlížeči vysledovat HTTP požadavek pro vyhledávání zboží na Seznamu, který má pro následující tvar:

`http://zbozi.seznam.cz/searchScreen?`**`q`**`=nokia+6233&`**`order`**`=cheapest&minPrice=&maxPrice=&`**`offset`**`=10`,

kde nejdůležitějšími parametry jsou `q`, `order` a `offset`. Parametr `q` představuje vyhledávaný výraz neboli klíčové slovo (v tomto případě je to „nokia 6233“), parametr `order` určuje způsob řazení (zde „cheapest“) a parametr `offset` udává v tomto případě posunutí o 10 vyhledaných položek. Zbylé dva parametry `minPrice` a `maxPrice` udávají minimální a maximální cenu zboží a nebyly v aplikaci použity.

Pro zjištění tvaru HTTP požadavku pro našeptávač již musí být použit nějaký program schopný tyto požadavky sledovat.[1] Je to z toho důvodu, že tyto požadavky probíhají na pozadí (AJAX). HTTP požadavek pro našeptávač má tvar:

`http://suggest.zbozi.seznam.cz/?dict=zbozi&`**`phrase`**`=nokia&encoding=utf-8&response_encoding=utf-8`

Nejdůležitějším parametrem je v tomto případě parametr `phrase`, který určuje prefix pro návrhy našeptávače. Další dva parametry určují kódování.

Na straně serveru jsou umístěny dva PHP skripty, které vyřizují asynchronní požadavky vytvořené klientem. Prvním je skript *myServerSide1.php*, který pokud mu jsou předány nezbytné parametry (vyhledávaný výraz, způsob řazení a offset), vytvoří požadavek pro vyhledávání zboží na vzdálený server (dle prvního výše uvedeného tvaru). Následně tento skript přijme od tohoto vzdáleného serveru (server Seznam) odpověď, což v tomto případě představuje XHTML dokument[2]. Poté je volána metoda třídy *Response* v souboru *response.class.php*, která získaná data parsuje pomocí DOM a která vytvoří odpověď. Ta je pak zaslána klientovi (web. prohlížeč).

[1] Použit byl program HttpSpy, který je volně ke stažení na http://www.devolutions.net.
[2] Skriptu je v podstatě vrácen zdrojový kód kompletní webové stránky s výsledky vyhledávání.

Druhým skriptem na straně serveru je *myServerSide2.php*. Tento skript vyřizuje požadavky klienta na návrhy pro našeptávač. Je zde vytvořen požadavek na vzdálený server (opět Seznam). Získaná odpověď má tvar (pro prefix = „n“):

```
var aa = new Array (['nikon','23627'],
['nokia','18552'],
['nod32','17639'],
['nod32 pro','11266'],
['nec','5481'],
['new','5085'],
['náhradní','4840'],
['nokian','4713'],
['nod32 pro ms','4650'],
['nabíječka','4512']);
sg.showList(aa);
```

Tato data ve formě řetězce jsou ve skutečnosti částí kódu JavaScriptu, který v normálním případě webová stránka vyhledávače Seznam ve svém skriptu nějakým způsobem zpracuje. Důležité bylo z tohoto řetězce získat potřebné návrhy pro našeptávač. Řešením je tento řetězec rozčlenit tak, aby zůstaly zvlášť (uložené v poli) jednotlivé návrhy. Následně vytvořená odpověď s návrhy našeptávače je opět zaslána klientovi.

4.3.2 Vyhledávač

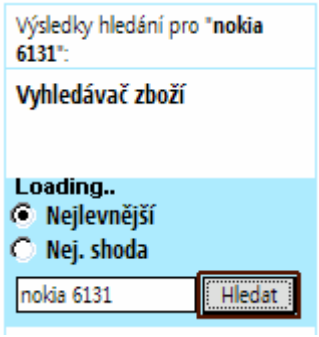
Vytvořená aplikace by se dala rozdělit do dvou funkčních částí. První z nich je samostatný vyhledávač, druhou je funkcionalita našeptávače, která lze do vyhledávače doplnit.

Základním zdrojovým souborem je *index.html*, což je XHTML dokument, který tvoří základní strukturu webové aplikace. Spolu s tímto souborem se v prohlížeči načítá definice kaskádových stylů, která je uložená v souboru *myCSS.css*, a veškerá hlavní funkcionalita představující externě uložený JavaScript (soubor *myJS.js*).

Hlavní částí vyhledávače je formulář, který zahrnuje dva přepínací prvky sloužící pro volbu způsobu řazení vyhledaných položek (od nejlevnější, nebo dle největší shody názvů), vstupní pole pro zadání vyhledaného výrazu a tlačítko pro spuštění vyhledávání.

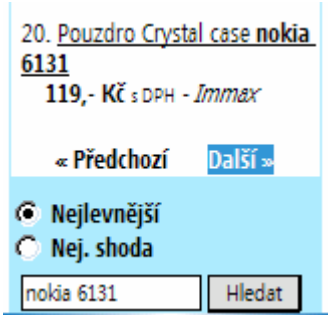
Jak již bylo řečeno, funkcionalitu na straně klienta zajišťuje JavaScript. Ihned po načtení stránky se vytvoří instance objektu `xmlHttpRequest`, která je připravena pro asynchronní komunikaci. Uživatel provede výběr způsobu řazení pomocí

přepínacích prvků. Tyto přepínače jsou obslouženy událostí `onclick`, na kterou je pro každý přepínač navázána funkce v JavaScriptu, která nastaví parametr `order` podle výběru uživatele, a to na hodnotu „cheapest“ (nejlevnější), nebo „relevance“ (největší shoda). Uživatel dále zadá vyhledávaný výraz a stiskne tlačítko „HLEDAT“. Toto tlačítko je obslouženo událostí `onclick` a při jeho stisku je inicializován asynchronní požadavek na server. Požadavek je odeslán včetně všech důležitých parametrů (vyhledávaný výraz, způsob řazení a offset). Server (php skript *myServerSide1.php*) zpracuje požadavek a odešle zpět odpověď. Tato odpověď je HTML kód, který je na straně klienta vložen pomocí atributu innerHTML do elementu `div` (viz kap. 1.4.4). Během získávání výsledků hledání je zobrazena animace „Loading…“ (*loading.gif*), která uživateli naznačuje, že je požadavek opravdu zpracováván (viz Obr. 4.3).



Obr. 4.3 – *Zobrazení animace „Loading…“*

Zobrazení výsledků bylo již naznačeno na Obr. 4.2 v kap. 4.2. Tyto výsledky jsou získávány a zobrazovány po deseti. Stejně je tomu tak i na vyhledávači Seznam, a tak nebylo nutné nic měnit. O kolikátou desítku výsledků se jedná určuje parametr `offset`. Jednotlivé výsledky představují odkazy na příslušné elektronické obchody. Pod výpisem výsledků hledání jsou zpřístupněny odkazy „Předchozí“ a „Další“, které umožňují získat dalších deset nebo předchozích deset výsledků (viz Obr. 4.4). To je prováděno opět pomocí asynchronní komunikace (AJAX).



Obr. 4.4 – *Odkazy „Předchozí“ a „Další“*

4.3.3 Našeptávač

Vytvoření této funkcionality nebylo až tak obtížné, jak se na první pohled zdálo. Našeptávače v desktopových prohlížečích jsou často řešeny tak, že aktualizace návrhů probíhá po stisku klávesy, a současně je testováno, zda došlo ke změně prefixu. V případě mobilních zařízení se nelze spolehnout na události vyvolané stiskem klávesy (viz kap. 2.3.2). Proto byl v tomto případě našeptávač řešen pomocí časovače, který opakovaně spouští funkci, jež detekuje změnu v zadávaném výrazu. Pokud funkce zaznamená změnu, dojde k asynchronnímu požadavku na server a následně k aktualizaci návrhů našeptávače. V mobilních zařízeních nelze jednoduše zařídit, aby se v návrzích našeptávače uživatel pohyboval pomocí šipek nahoru a dolů. Navíc je nutné brát v úvahu, že některá mobilní zařízení používají jako hlavní ovládací prvek pero. Z těchto důvodů je našeptávač realizován jako seznam odkazů. Využívá se toho, že pohyb v mobilních zařízeních, zejména telefonech, je obdobný pohybu pomocí tabulátoru na desktopových web. stránkách (zaměření přeskakuje mezi aktivními prvky, např. odkazy).

Aktualizace návrhů je obdobná aktualizaci výsledků vyhledávání. Opět server vrací odpověď jako HTML kód a pomocí atributu `innerHTML` je vložen. Tentokráte však do elementu `ul`, který tvoří našeptávač. Vkládaný kód, který je vytvořen na serveru, vypadá takto:

```
<li>
  <a href="" onclick="suggest('nikon'); return false;">nikon</a>
</li>
<li>
  <a href="" onclick="suggest('nokia'); return false;">nokia</a>
</li>
…
…
<li>
  <a href="" onclick="suggest('nec'); return false;">nec</a>
</li>
```

Jednotlivé odkazy jsou obslouženy událostí `onclick`. Při vyvolání této události je zavolána funkce `suggest(parametr)`, kde je předávaným parametrem samotný výraz příslušného návrhu. Tímto výrazem je pak nahrazen zadávaný vyhledávaný výraz ve vstupním textovém poli a dojde ke skrytí našeptávače. Pokračuje-li uživatel v zadávání hledaného výrazu, našeptávač je opět zobrazen. Stiskne-li tlačítko „Hledat“, je ukončen případný asynchronní požadavek pro návrhy našeptávače a dojde k asynchronnímu požadavku pro vyhledávání.

V obou případech (vyhledávač a našeptávač) vrací server jako odpověď vytvořený HTML kód, který je pak jednoduše pomoci atributu `innerHTML` vložen tam, kam je určen. Samozřejmě by bylo možné předávat data mezi klientem a serverem např. pomocí XML. Tento způsob by však představoval více JavaScriptu a DOM manipulace na straně klienta, což představuje vetší zátěž procesoru mobilního zařízení.

Vytvořený našeptávač je dobrým pomocníkem zvláště na mobilních zařízeních, kde je zadávání velmi pomalé. Bohužel kvůli vysokým latencím by jeho funkce nemusela být ideální. V některých případech dokonce nemusí tato funkcionalita pracovat podle očekávání. V prohlížeči Nokia S60 dojde k aktualizaci obsahu vstupního textového pole až po jeho opuštění. To znamená, že našeptávač je zobrazen až v době, kdy uživatel opustí zadávací pole. Protože až v tomto momentě je zaznamenána změna vyhledávaného výrazu.

4.4 Testování aplikace

Aplikace byla testována způsoby popsanými v kap. 2.4 – pomocí emulátoru a vzdáleného přístupu k mobilnímu zařízení.

4.4.1 Microsoft Device Emulator 1.0

V prvé řadě byla vytvořená aplikace testována v emulátorech PPC a smartphone v těchto mobilních prohlížečích: mobilní IE, Opera Mobile 8.6 a Opera Mobile 8.65 Beta, Minimo. Mobilní IE je součástí Windows Mobile. Ostatní prohlížeče byly doinstalovány. Vyhledávač v těchto prohlížečích pracoval bez problémů. Některé nepřesnosti v zobrazení v mobilním IE a Minimu jsou způsobeny jejich schopnostmi a podporou CSS.

4.4.2 Nokia – Remote Device Access

Aplikace byla také testována prostřednictví vzdáleného přístupu na mobilním telefonu Nokia N95, ve kterém byla k dispozici nejnovější verze mobilního prohlížeče Nokia S60. Dále byl do toho mobilního telefonu nainstalován prohlížeč Opera Mobile 8.65, kde byla aplikace také vyzkoušena. Až na omezenou funkci našeptávače v prohlížeči Nokia S60 (viz výše kap. 4.3.3) pracovala aplikace správně.

Ukázky z testování v mobilních prohlížečích jsou v Příloze B.

ZÁVĚR

AJAX se stal od svého vzniku fenoménem. Rok 2005 byl rokem, kdy se všichni vývojáři z oblasti internetu učili, co to vlastně je a k čemu je to dobré. Zjistilo se, že pomocí této technologie lze výrazně zkvalitnit webové aplikace. Tyto aplikace jsou již tak interaktivní a uživatelsky přívětivé, že se stále více podobají aplikacím, které je nutné před použitím nainstalovat. A přitom se využívá stávajících, běžně užívaných webových standardů a technologií. To vedlo k obrovskému rozšíření této technologie. AJAX byl zmiňován v každém koutě internetu a každý se ho snažil použít ve své webové aplikaci či stránce. Díky tomu začaly vznikat projekty, které vyvíjely nejrůznější frameworky a nástroje pro usnadnění práce s AJAXem.

O roce 2006 by se dalo říci, že byl rokem, kdy se používání AJAXu ustálilo. V tomto roce vznikla celá řada zajímavých aplikací využívajících asynchronní komunikaci, kterou AJAX umožňuje. Mnoho těchto aplikací využívá vytvořených frameworků.

V současné době je trendem směřovat k opravdu velice interaktivním internetovým aplikacím uvnitř prohlížeče. Objevují se rozsáhlé aplikace, které se svým vzhledem i schopnostmi vyrovnají instalovaným aplikacím.

Oblast mobilního webu zaznamenala za poslední dobu obrovský růst, zejména díky neustále rostoucím schopnostem mobilních zařízení s otevřeným OS, jako jsou smartphony a kapesní počítače. A také díky schopnostem mobilních prohlížečů, z nichž většina podporuje i AJAX. V Čechách ještě sice stále převládají klasické mobilní telefony založené na platformě JAVA, ale uživatelů, kteří používají smartphony neustále přibývá. Podle některých zdrojů, bude během pár let každý čtvrtý mobilní telefon jedním, z tzv. „chytrých“ telefonů.

Tato práce byla založená zejména na současných možnostech AJAXu v mobilních zařízeních. Lze říci, že možnosti využití AJAXu nejsou v této oblasti ještě zcela rozšířené. Teprve nedávno se používání AJAXu ustálilo na osobních počítačích. A následně tuto technologii začalo podporovat i několik moderních mobilních prohlížečů. Pro tyto mobilní prohlížeče lze vytvářet webové aplikace s AJAXem. Takovéto aplikace se však kvůli svému zaměření nemohou chlubit nijak velkou interaktivitou, jako je tomu na desktopových zařízeních, a AJAX zde bude použit spíše pro načítání jen částí dokumentu a ušetření množství přenášených dat.

Dalším významným krokem kupředu je připravovaná podpora widgets v mobilních zařízeních, což uskutečňuje několik společností. Tyto malé webové aplikace mohou využívat asynchronního přenosu dat (AJAXu) a mají celou řadu výhod. Asi největší výhodou je, že tyto aplikace oproti aplikacím v prohlížeči zpřístupňují hardware a další aplikace mobilních zařízení. Což otevírá nové prostory pro mobilní aplikace, které jsou založené na běžných webových technologiích včetně AJAXu.

V rámci dostupných možností byla jako součást této práce vytvořena ukázková aplikace určená mobilním prohlížečům podporujících AJAX. V některých těchto prohlížečích byla otestována a lze říci, že AJAX je v mobilních zařízeních opravdu použitelný. Zvláště pro načítání jen některých částí stránek. Funkcionality typu našeptávač představují mnoho HTTP požadavků a při horším připojení k síti, a zvláště při vysokých latencích může být jejich správná funkce narušena.

Celá práce vznikala ve spolupráci s firmou MITON CZ, s.r.o., která nabízí služby ve světě internetu již řadu let. V oblasti mobilních aplikací nemá příliš zkušeností. Firma neustále zkoumá možnosti rozšíření hranice oblasti uplatnění a zajímá ji vývoj a jeho směr. Tato práce by jí měla poskytnout náhled na současný stav možností využití moderních webových technologií v mobilních zařízeních. Vytvořená ukázková aplikace by mohla být základem pro další vývoj a komerční použití.

Literatura

[1] Garet J. J., Ajax: A New Approach to Web Applications, URL: <www.adaptivepath.com/publications/essays/archives/000385.php>

[2] Asleson R., Schutta N. T.: AJAX – Vytváříme vysoce interaktivní webové aplikace, Computer Press a.s., 2006, ISBN 80-251-1285-3

[3] Darie C., Brinzarea B., Cherecheş-Toşa F., Bucica M.: AJAX a PHP – tvoříme interaktivní webové aplikace PROFESIONÁLNĚ, Toner Press, s.r.o., 2006, ISBN 80-86815-47-1

[4] Bray T., Paoli J., Sperberg-McQueen C. M., XML W3C recommendation, URL: <www.w3.org/TR/2006/REC-xml-20060816>

[5] JSON Homepage, URL: <www.json.org>

[6] Janovský D., Jak psát web, online text, URL: <www.jakpsatweb.cz>

[7] Různí autoři, JavaScript Core Reference – oficiální JavaScript dokumentace, URL: <www.slunecnice.cz/product/JavaScript/Core-Reference-HTML-15>

[8] Různí autoři, Making Small Devices Look Great, My Opera Community, URL: <http://my.opera.com/community/dev/device>

[9] Minimo project page, URL: <www.mozilla.org/projects/minimo>

[10] Různí autoři, Opera Developer Community, URL: <http://dev.opera.com>

[11] Forum Nokia – Resources for mobile application developers, URL: <www.forum.nokia.com>

[12] Opera Platform, URL: <www.opera.com/products/mobile/platform>

[13] Bráza J., PHP 5 – začínáme programovat, Grada Publishing, a.s., 2005, ISBN 80-247-1146-X

[14] Holzner S., JavaScript profesionálně, Mobil Media, a.s., 2003, ISBN 80-86593-40-1

[15] Různí autoři, IEMobile Team Weblog, online text, URL: <http://blogs.msdn.com/iemobile/default.aspx>

[16] Různí autoři, Oficiální stránka PHP, URL: <www.php.net>

[17] Různí autoři, Ajaxian, stránka o problematice AJAX, online text, URL: <www.ajaxian.com>

Příloha A – atributy a metody DOM

Tab. A.1 – *Atributy DOM elementů*

Atribut	Popis
`childNodes`	Vrací pole potomků daného elementu.
`firstChild`	Vrací prvního přímého potomka daného elementu.
`lastChild`	Vrací posledního potomka daného elementu.
`nextSibling`	Vrací element následující ihned za daným elementem.
`nodeValue`	Specifikuje atribut pro čtení/zápis reprezentující hodnotu elementu.
`parentNode`	Vrací rodičovský uzel elementu.
`previousSibling`	Vrací element předcházející daný element.

Tab. A.2 – *Metody DOM elementů*

Metoda	Popis
`getElementById(id) (dokument)`	Vrací element se specifikovanou unikátní hodnotou atributu ID.
`getElementsByTagName(název)`	Vrací pole potomků daného elementu se specifikovaným názvem značky.
`hasChildNodes()`	Vrací Boolean indikující, zda element má nějaké potomky.
`getAttribute(název)`	Vrací hodnotu atributu `název` daného elementu

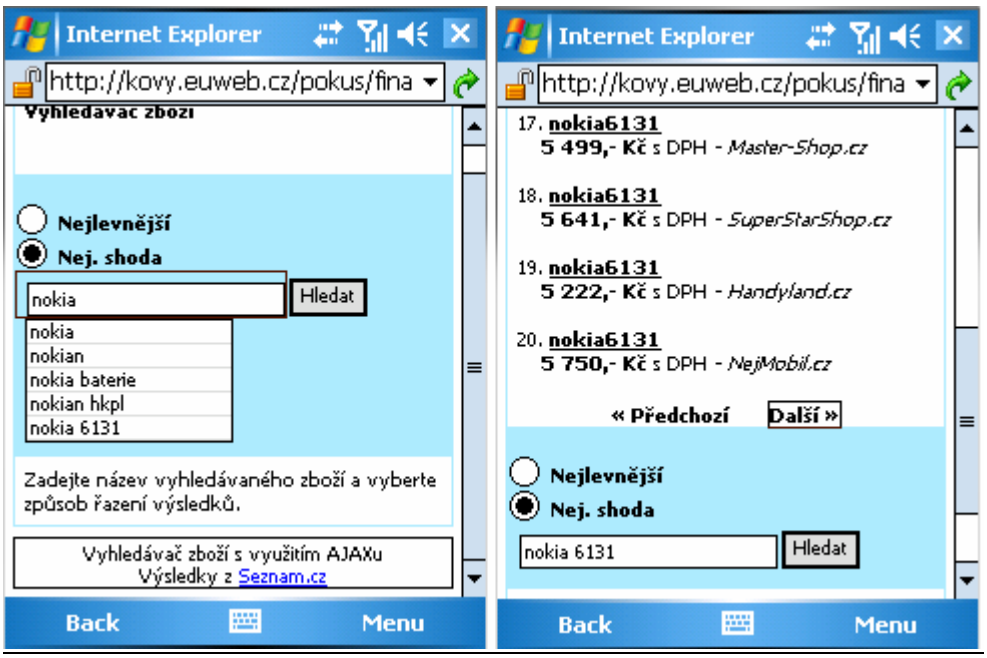
Tab A.3 – *Atributy a metody DOM užitečné pro vytvoření dynamického obsahu*

Atribut/metoda	Popis
`dokument.createElement(název)`	Vytvoří element specifikovaný argumentem `název`.
`Dokument.createTextNode(text)`	Vytvoří uzel obsahující statický text.
`<element>.appendChild(uzel)`	Přidá zadaný uzel do seznamu potomků daného elementu.
`<element>.getAttribute(název)` `<element>.setAttribute(název, hodnota)`	Získá, resp. Nastaví, hodnotu atributu elementu.
`<element>.insertBefore(novýUzel, cílový uzel)`	Vloží uzel zadaný argumentem `novýUzel` před element zadaný argumentem `cílovýUzel` jakožto potomka daného elementu.
`<element>.removeAttribute(název)`	Odstraní atribut z elementu.
`<element>.removeChild(uzel)`	Odstraní uzel ze seznamu potomků elementu.
`<element>.replaceChild(novýUzel, starýUzel)`	Nahradí uzel za nový uzel.
`<element>.hasChildNodes()`	Vrací Boolean indikující, zda element má nějaké potomky.

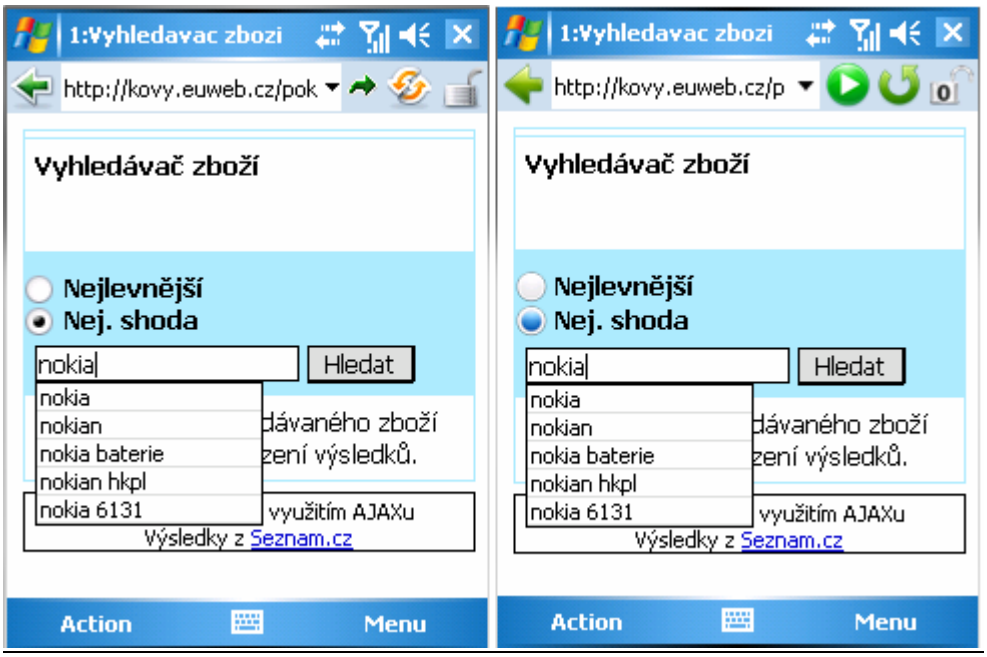
Příloha B – ukázky z testování aplikace

Microsoft Device Emulátor 1.0 – PPC

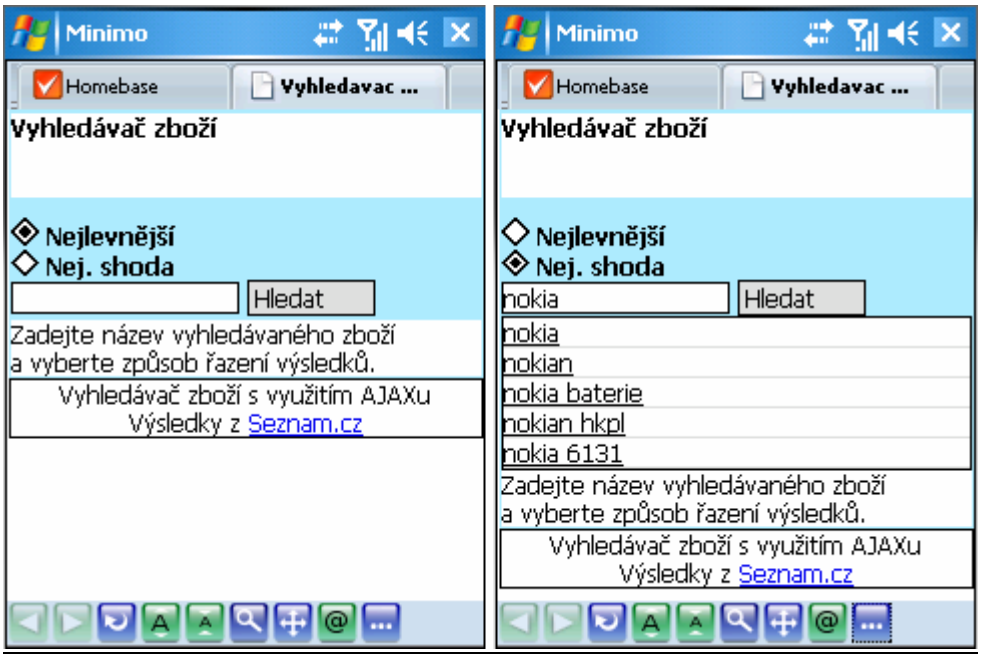
Mobilní IE



Opera Mobile 8.60 a 8.65 Beta

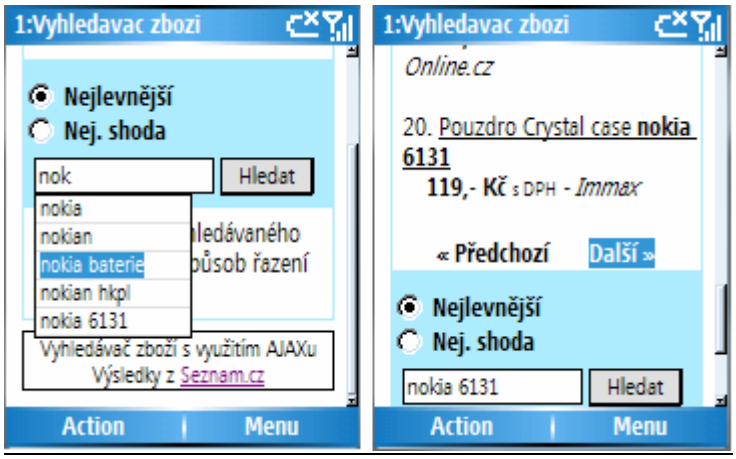


Minimo 0.2



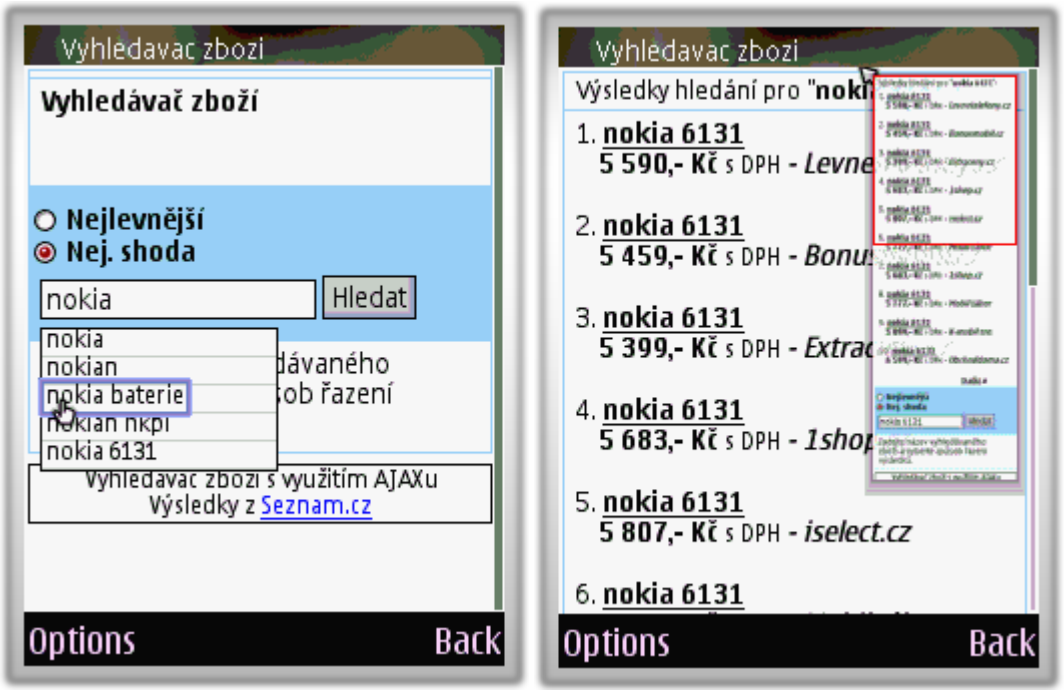
Microsoft Device Emulátor 1.0 – smartphone

Opera Mobile 8.65 Beta



Nokia – Remote Device Access

Nokia S60



Opera Mobile 8.65