KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
VERSLO INFORMATIKOS KATEDRA

Viktorija Sprainytė

Asinchroninė sąsaja internetinėse sistemose ir jos kūrimo metodologija

Magistro darbas

Darbo vadovas
doc. dr. V.Pilkauskas

Kaunas, 2007

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
VERSLO INFORMATIKOS KATEDRA

Viktorija Sprainytė

Asinchroninė sąsaja internetinėse sistemose ir jos kūrimo metodologija

Magistro darbas

Recenzentas
doc. dr. D.Rubliauskas
2007-05-28

Darbo vadovas
doc. dr.V.Pilkauskas
2007-05-28

Atliko
IFM-1/1 gr. stud.
Viktorija Sprainytė
2007-05-28

Kaunas, 2007

Turinys

LENTELĖS

PAVEIKSLAI

Asynchronous interface in web systems and methodology for creating it

SUMMARY

Presented work covers main web modelis: classic and asynchronous (AJAX) and their working schemes. The author pays great attention to asynchronous interfaces – their working principles, existing software solutions and problems which may occure when developer tries to adopt them.

The research work describes the range of implementation possibilities: branches of technology, usable formats, their pros and cons. Prototype model for creating asynchronous web systems and it‘s features are provided. Author introduces what problems may occur in implementation phase and what influence this technology may have on users‘, developers‘ and whole web‘s security.

1. Įvadas

Nuo interneto atsiradimo jame saugomos informacijos turinys, tipas ir saugojimo būdas smarkiai keitėsi. Gana ilgą laiką vartotojo naršyklė bendraudavo su internetiniu serveriu klasikiniu būdu – kiekvienas vartotojo veiksmas naršyklėje siųsdavo užklausą į serverį ir grąžindavo rezultatus atgal. Toks statinis interneto modelis labai ribojo tiek vartotojų, tiek interneto puslapių kūrėjų darbą ir galimybes.

Ilgainiui pasidarė svarbu ne tik pati informacija, bet ir kaip ji atvaizduojama – kaip būdas pritraukti vartotoją ar kuo geriau išspręsti konkrečią problemą. Pradėta ieškoti naujų metodų informacijos atsiuntimui iš internetinio serverio vartotojui.

AJAX viena iš tokių technologijų (technologijų rinkinys) pakeitęs supratimą apie tai kaip gali atrodyti informacinė sistema ir ką ji gali padaryti. Technologijos realizacija, paremta asinchronine sąsaja, labai skiriasi priklausomai nuo naršyklių. Sukurta aibė įvairiausių įrankių ir bibliotekų padedančių sumažinti veikimo skirtumus skirtingose interneto naršyklėse, padaryti tokias internetines sistemas kuo panašesnes į paprastas programas, kuriose programos veikimo sklandumas nepriklauso nuo interneto greičio. Pati technologija nebuvo susieta su konkrečiais duomenų formatais ar programavimo kalba, internetinių sistemų kūrėjams palikta laisvė pasirinkti jas patiems, priklausomai nuo turimos darbo patirties su konkrečiomis technologijomis ir įgūdžių.

Manoma, kad nauja technologija atvėrė daug saugumo spragų, bet iki tol internetas taip pat nebuvo saugus. Konkrečiais atvejais priemonių turi imtis tiek standartų kūrėjai, tiek naršyklių programuotojai, sistemų projektuotojai bei paprasti vartotojai, naudojantys AJAX technologija paremtas internetines sistemas.

Darbe aprašyti ir palyginti pagrindiniai internetiniai modeliai: klasikinis ir asinchroninis (AJAX). Dėmesys darbe sutelktas į asinchronine sąsajas – veikimo principus, egzistuojančius programinės įrangos sprendimus, problemas, su kuriomis gali susidurti programuotojas.

Tiriamajame darbe pateikiamas šios technologijos realizavimo galimybių spektras: technologijos atmainos, naudojami formatai, jų teikiami privalumai ir trūkumai. Nepamirštama su kokiais sunkumais gali susidurti programuotojas norintis realizuoti tokio tipo sistemą. Pateikiamas prototipinis asinchroninių sistemų kūrimo modelis, bei jo galimybės. Apžvelgiama kokią įtaką vartotojui, programuotojui ir visam Interneto saugumui turi ši nauja technologija.

2. Sinchroninė ir asinchroninė sąsaja

2.1. Sinchroninė sąsajos problematika

Internetas buvo suprojektuotas naršyti HTML dokumentus. To pasekoje, klasikinis internetinių sistemų modelis prisitaikė „paspausk, palauk ir atnaujink“ vartotojo veiksmų paradigmą ir sinchroninį užklausos/atsako komunikavimo mechanizmą.

„Paspausk, palauk ir atnaujink“ vartotojo veiksmų modelis aprašytas tokia seka:

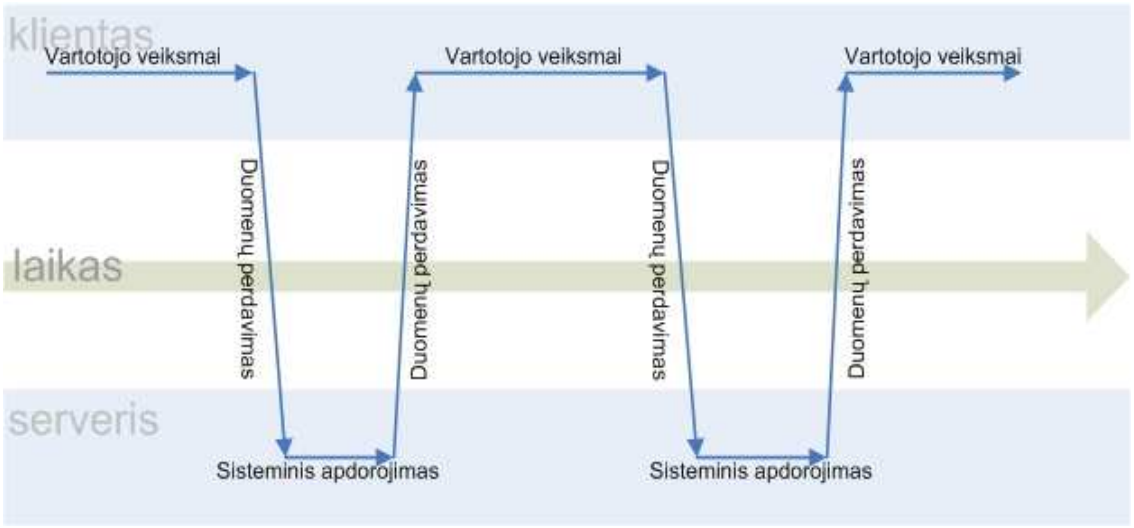
- naršyklė reaguoja į vartotojo veiksmus, pašalina dabartinį HTML puslapį ir siunčia HTTP užklausą atgal į interneto serverį;
- serveris atlieka tokį pat apdorojimą ir grąžina HTML puslapį naršyklei;
- naršyklė atnaujina langą ir atvaizduoja naują HTML puslapį.

Sinchroninis „užklausos/atsako“ komunikavimo modelis:

- naršyklė visada inicijuoja užklausą, o internetinis serveris atsakinėja į tokias užklausas;
- internetinis serveris niekad neinicijuoja užklausų – komunikavimas visada inicijuojamas viena kryptimi;
- užklausos/atsako ciklas yra sinchroninis, kurio metu vartotojas privalo laukti.

Vis gi, šios dvi esminės klasikinio interneto modelio elgsenos neveikia kaip reikia su paprastomis programomis. Programinės įrangos kontekste, klasikinis internetinių sistemų modelis sukuria daug problemų:

- lėtas veikimas dėl „paspausk, palauk ir atnaujink“ modelio;
- operacijos konteksto praradimas atnaujinus puslapį;
- didelė serverio apkrova ir pralaidumo sunaudojimas dėl perteklinių puslapio atnaujinimo veiksmų ir dvikrypčio, realaus laiko komunikavimo galimybės(serverio inicijuotiems atnaujinimams) trūkumas.



1 pav. Sinchroninis internetinių sistemų modelis

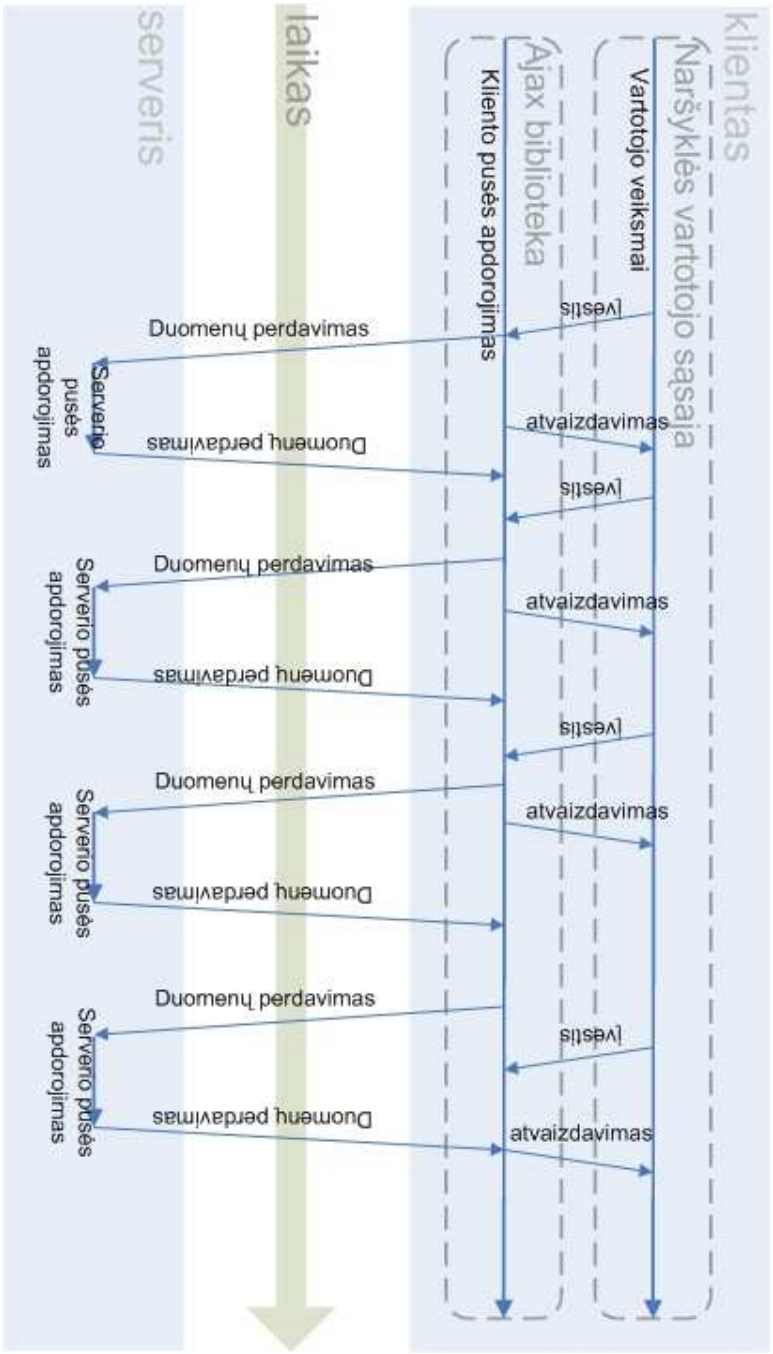
2.2. Asinchroninė sąsaja internetinėse sistemose

Kompiuterio programų kontekste „paspausk, palauk ir atnaujink“ bei „sinchroninė užklausa/atsakas“ sąlygoja, kad internetinės programos yra lėtos, nepatikimos, neproduktyvios ir neefektyvios. Šie du pagrindiniai procesai turi būti pakeisti norint sukurti greitesnes, interaktyvias ir efektyvesnes internetines programos – būtent tai, ką daro AJAX sprendimų modelis.

AJAX modelyje:

- „Dalinis vaizdo atnaujinimas“ pakeičia „paspausk, palauk ir atnaujink“ vartotojo veiksmų modelį. Vartotojui atliekant veiksmus AJAX paremtoje internetinėje sistemoje, atnaujinami tik tie vartotojo sąsajos elementai, kurie turi naują informaciją, o likusi sąsajos dalis paliekama kokia buvo. Dėl „dalinio vaizdo atnaujinimo“ ne tik turime testinį veiksmų kontekstą, bet ir nelinijinę darbo eigą.
- Asinchroninis komunikavimas pakeičia „sinchroninį užklausos/atsako modelį“: AJAX paremtoje sistemoje, užklausa/atsakas gali būti asinchroniniai, atskiriantys vartotojo veiksmus nuo serverio veikimo. Dėl to, vartotojas gali tęsti naudojimąsi programa, kol kliento programa paprašo informacijos iš serverio. Gavus naują informaciją, tik su ja susijusios vartotojo sąsajos dalys yra atnaujinamos.

Kadangi AJAX esmė yra daliniai ekrano atnaujinimai ir asinchroninis komunikavimas, programinis modelis nėra susietas su jokiu specialiu duomenų apsikeitimo formatu (tokiu kaip XML), tam tikra programavimo kalba (kaip Javascript) ar specialiu komunikavimo mechanizmu.



2 pav. Asinchroninis internetinių sistemų modelis

3. AJAX

3.1. Kas yra AJAX?

AJAX terminą pirmąkart viešai panaudojo Jesse James Garrett. Tai įvyko 2005 vasarį. Terminas buvo skirtas apibūdinti aibę technologijų, skirtų asinchroninei naršyklės sąveikai su serveriu. Tada terminas reiškė kad ši technologijų aibė yra sudaryta iš asinchroninio JavaScript ir XML. Tačiau laikui bėgant, vietoj XML galėjo būti naudojamas kitas formatas, o JavaScript pakeistas kita programavimo kalba.

Kad technologijų aibė galėtų būti pavadinta AJAX, ji turi palaikyti šiuos pagrindinius veikimo principus:

- Naršyklė talpina tik programą, ne duomenis;
- Serveris pateikia duomenis, o ne turinį;
- Vartotojo sąveika su programa gali būti sklandi ir nenutrūkstama;

Veikimo atžvilgiu AJAX nėra nauja technologija. Panašiai veikiančios technologijos buvo kurtos tik Internet Explorer naršyklei Windows operacinėje sistemoje jau daugelį metų. Bet iki tol, jos buvo žinomos kaip interneto nuotolinis valdymas (angl. „web remoting“) arba nuotolinis skripto vykdymas (angl. „remote scripting“). Kūrėjai taip pat naudojo įvairius naršyklių priedus (angl. „plug-ins“), Java įskiepius ir paslėptus langus (angl. „hidden frames“), siekdami sukurti panašų funkcionalumą. Kas nuo to laiko pasikeitė, tai XMLHttpRequest objekto palaikymas pagrindinių naršyklių JavaScript aplinkose. Nors šis objektas nėra aprašytas formalioje Javascript technologijos specifikacijoje, visos šiuolaikinės populiariausios naršyklės palaiko jį. Nežymūs skirtumai tarp JavaScript technologijos ir CSS palaikymo tarp naujausios kartos naršyklių, tokių kaip Mozilla Firefox, Internet Explorer ir Safari nėra didelė kliūtis. JavaScript bibliotekos, tokios kaip Dojo, Prototype ir Yahoo User Interface Library skirtos tokiems skirtumams tarp naršyklių JavaScript palaikomumo sumažinti ir parūpinti standartizuotą programavimo modelį.

AJAX paremtos klientinės programos unikalios tuo, kad jos turi puslapiui būdingą valdymo logiką, įtvirtintą JavaScript technologijoje. Puslapis sąveikauja su JavaScript technologija remdamasis tokiais įvykiais kaip dokumento užkrovimas, pelės mygtuko paspaudimas, fokuso atsiradimu/dingimu ar net specialiu skaitikliu. AJAX sąveika leidžia aiškiai atskirti atvaizdavimo logiką nuo duomenų. Paprastas HTML puslapis gali būti persiųstas bitas po bito ir tada atvaizduotas. AJAX sąveikos modeliui reikia kitokios serverio pusės architektūros. Paprastai, serverio pusės internetinės programos yra orientuojamos į HTML generavimą kiekvienai kliento užklausai į serverį. Klientinei pusei atnaujinus puslapį,

HTML puslapis bus sugeneruotas kiekvienai užklausai atskirai. RIA tipo internetinės programos orientuojasi į HTML puslapio gavimą, kuris atstoja šabloną – jame priklausomai nuo kliento „įvykių“ įterpiamas turinys iš XML duomenų , gautų iš serverio pusės komponento.

3.2. AJAX architektūra

Pagrindiniai AJAX elementai:

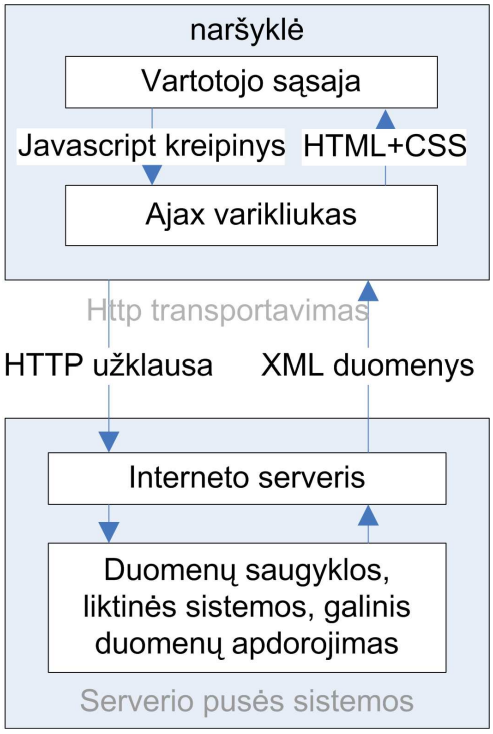
- JavaScript (gali būti pakeistas Java);
- Pakopiniai stiliai (CSS);
- Dokumento objektinis modelis (DOM);
- XMLHttpRequest objektas;

Programinės įrangos aspektu AJAX skiriasi nuo šiuolaikinių internetinių sistemų architektūrų tuo, kad turi kliento pusės varikliuką (biblioteką):

- Vartotojo pusės varikliukas tarnauja kaip terpė tarp vartotojo sąsajos ir serverio;
- Vartotojo veiksmai kviečia kliento pusės varikliuką vietoj puslapių iš serverio;
- XML duomenys perduodami tarp serverio ir kliento pusės varikliuko.

Kliento varikliukas yra pagrindinė AJAX modelio ašis (žiūrėti 4 pav.). Be jo kiekvienas vartotojo veiksmas turėtų nukeliauti iki serverio ir būti apdorotas ten. Vartotojo veiksmai yra stipriai susieti su serverio komunikavimu – kliento varikliukas panaikina šią priklausomybę, Šis variklis, veikiantis interneto naršyklėje, suteikia jai papildomos elegancijos atnaujinant dalį programos lango vietoj viso. Šis variklis taip pat „bendrauja“ su serveriu fone, atsiedamas vartotojo veiksmus nuo komunikavimo su serveriu.

3 pav. Klasikinis internetinių sprendimų modelis



4 pav. Ajax internetinių sprendimų modelis

Tokios dinaminės internetinės programos elgiasi panašiai kaip paprastos programos (darbastalio) nepriklausomai nuo naršyklės įskiepių (priedų) ir tik jai būdingų savybių. Tradiciškai, internetinės programos buvo aibė HTML puslapių, kurie turėjo būti atnaujinami, norint pakeisti dalį puslapio turinio (3 pav). Tokios technologijos kaip Javascript

programavimo kalba ir CSS pakankamai subrendo, kad būtų efektyviai naudojamos kuriant dinamines internetines programas, kurios veikia su dauguma pagrindinių naršyklių.

Naudojant JavaScript, HTML puslapis gali asinchroniškai atlikti užklausas į serverį, iš kurio jis buvo užkrautas, ir gauti turinį, kuris suformuojamas kaip XML, HTML dokumentas, paprastas tekstas ar aprašytas JSON formatu. JavaScript po to gali panaudoti gautus duomenis atnaujinti arba pakeisti HTML dokumento objektinį modelį (DOM).

3.3. AJAX sąveikos pritaikymo atvejai

Nors neatrodo sunku įsivaizduoti AJAX pritaikymo, tačiau iš tikrųjų tai yra daug galingesnis įrankis nei gali atrodyti iš pradžių. Neįmanoma visų pritaikymo atvejų surinkti ir aprašyti, bet galima išskirti pagrindinius:

- formos duomenų validavimas realiu laiku: kai kurie vartotojo įvesti duomenys gali būti validuojami anksčiau nei forma nusiunčiama į serverį;
- automatiškas žodžių užbaigimas: tam tikros duomenų dalys gali būti užpildomos automatiškai vartotojui įvedant juos (pvz., elektroninio pašto adresas, vardas, miesto pavadinimas).
- Užkrovimas pagal užklausimą: priklausomai nuo kliento įvykių, HTML puslapis gali atsisiųsti daugiau duomenų foniniu režimu, kas įtakos naršyklės veikimo greitį (puslapiai bus užkraunami greičiau).
- Ypatingi vartotojo sąsajos valdymo elementai ir efektai: tokie valdymo elementai, kaip „medžiai“, meniu, duomenų lentelės, daugia-funkciniai tekstiniai redaktoriai, kalendoriai, progreso juostos užtikrina geresnę sąveiką su vartotoju ir HTML puslapiais neatnaujinant puslapio rankiniu (paspaudžiant „refresh“ mygtuką) būdu.
- Duomenų atnaujinimas ir duomenų atnaujinimas iš serverio pusės: HTML puslapiai gali užsisakyti naujausių duomenų iš serverio: varžybų rezultatų, akcijų kurso, orų ir t.t. Klientas gali naudoti AJAX technologijas gauti naujausių duomenų aibę neatnaujinant viso puslapio. Užklausos siuntimas nėra labai efektyvus, tam reikia nuolat būti įsitikinus, kad puslapio duomenys yra naujausi. Duomenų atnaujinimas iš serverio pusės per HTTP yra garantuojamas palaikant pastovų ryšį tarp serverio ir kliento (Comet technologija).
- Dalinis duomenų patvirtinimas. HTML puslapis gali patvirtinti formos duomenis kada to reikia be pilno puslapio atnaujinimo.
- „Mashups“: dviejų ar daugiau servisų duomenų, funkcionalumo suliejimas norint gauti naują paslaugą (pvz., Google Maps paslauga).

- Puslapis kaip programa: AJAX technologijos gali būti naudojamos sukurti vieno puslapio programas, kurios atrodo ir veikia kaip darbastalio programos (pvz., Gmail).

Šiuo metu egzistuojančios AJAX pritaikymo tendencijos neatspindi, kam ši technologija bus naudojama ateityje. Tai natūralu, nes tai nauja technologija ir jos galimybės dar pilnai neištirtos.

3.4. Alternatyvios technologijos skirtos AJAX

AJAX jau senokai nebėra vien tik asinchroninis JavaScript ir XML. Bet kuri technologija pagal kurią sukurta internetinė programa atitinkanti AJAX veikimo principus gali būti priskirta šiam internetinių sistemų tipui.

Atsižvelgus į tai, kad AJAX atstovauja internetinių programų modelį, paremtą daliniu vaizdo atnaujinimu ir asinchroniniu komunikavimu, egzistuoja keletas technologijų AJAX kliento varikliuko sukūrimui, iš kurių Javascript, Java ir Flash yra dažniausiai naudojamos. Be to, šios technologijos daugiausiai naudojamos kurti būtent AJAX programas.

3.4.1. Java būdas (Asichroninė Java + XML)

Java paremtas metodas paprastai naudoja naršyklėje esantį Java variklį kliento pusės apdorojimui, pvz vartotojo sąsajos atvaizdavimui, daliniam lango atnaujinimui ir asinchroniniam ryšiui su serveriu užtikrinti. Šiuo būdu sukurtose programose, vartotojo sąsaja gali būti aprašyta XML, o kliento pusės programos logika suprogramuota standartine Java.

3.4.2. Javascript/DHTML metodas (Asinchroninis Javascript+XML)

Jacascript/DHTML būdas paprastai naudoja naršyklėje esančią Javascript biblioteką kliento pusės apdorojimui – daliniam vaizdo atnaujinimui ir asinchroniniam foniniam komunikavimui. Paprastai vartotojo sąsaja yra aprašyta naudojant DHTML ir kliento pusės logika yra vykdoma Javascript kode.

3.4.3. Flash technologija (Asinchroninis ActionScript ir SWF)

Flash technologija paprastai naudoja naršykle paremtą Flash varikliuką su ActionScript biblioteka kliento pusės apdorojimui. Šiuo atveju vartotojo sąsaja aprašoma SWF kliento pusėje, o veikimo logika kliento pusėje suprogramuojama ActionScript programavimo kalba.

3.4.4. Skirtintų technologijų privalumai ir trūkumai

Kiekvienas iš būdų kurti AJAX turi savo privalumų ir trūkumų, todėl vienas ar kitas sprendimas yra tinkamesnis konkretiems sprendimams.

Lentelė Nr. 1 Ajax technologijų palyginimai

	Privalumai	Trūkumai
Java	• kokybiškas ir patikimas veikimas; • lengvas palaikymas dėl objektiškai orientuoto programavimo; • tinkamas programuojantiems Java.	• reikia Java programavimo žinių.
DHTML/ Javascript	• derinasi su egzistuojančiomis HTML internetinėmis programomis ir puslapiais; • tinka mokantiems programuoti DHTML.	• Javascript/DHTML kodas yra sunkiai palaikomas ir neskirtas komandiniam programavimui; • Veikimo ir funkcionalumo apribojimai.
Flash	• Galima naudoti animacijas; • Tinka grafikos dizaineriams.	• ActionScript kodas yra sunkiai palaikomas ir nesuprojektuotas komandiniam darbui; • apribojimai veikimui ir funkcionalumui;

Java technologija įgalina sukurti gerai veikiančias ir funkcionalias programas. Šia programavimo kalba sukurtos programos yra lengvai palaikomos, dėl objektiškai orientuotos Java prigimties. Taip pat, ši technologija tinka programuotojams, programuojantiems Java, bet tuo pačiu tai ir trūkumas, jei šios programavimo kalbos programuotojas nemoka. Istoriškai susiklostė, kad JVM suderinamumas su įvairiomis naršyklėmis buvo svarbi problema šios technologijos plitimui. Vis gi, šiuo metu egzistuoja sprendimai pašalinantys šias problemas.

DHTML/Javascript metodas puikiai derinasi su egzistuojančiomis HTML paremtomis internetinėmis programomis. Šiai technologijai būdinga tai, kad vartotojas atsisiųstų gana daug užimančias Javascript bibliotekas, kurios kompensuotų kliento naršyklių skirtumų trūkumus. Programų interaktyvumas priklausys nuo to, kokią naršyklę vartotojas naudos. Ši technologija gali kelti problemų didelės apimties verslo sistemose dėl sunkumų, kylančių programuojant ir palaikant Javasript kodą, bei DHTML veikimo/funkcionalumo apribojimų.

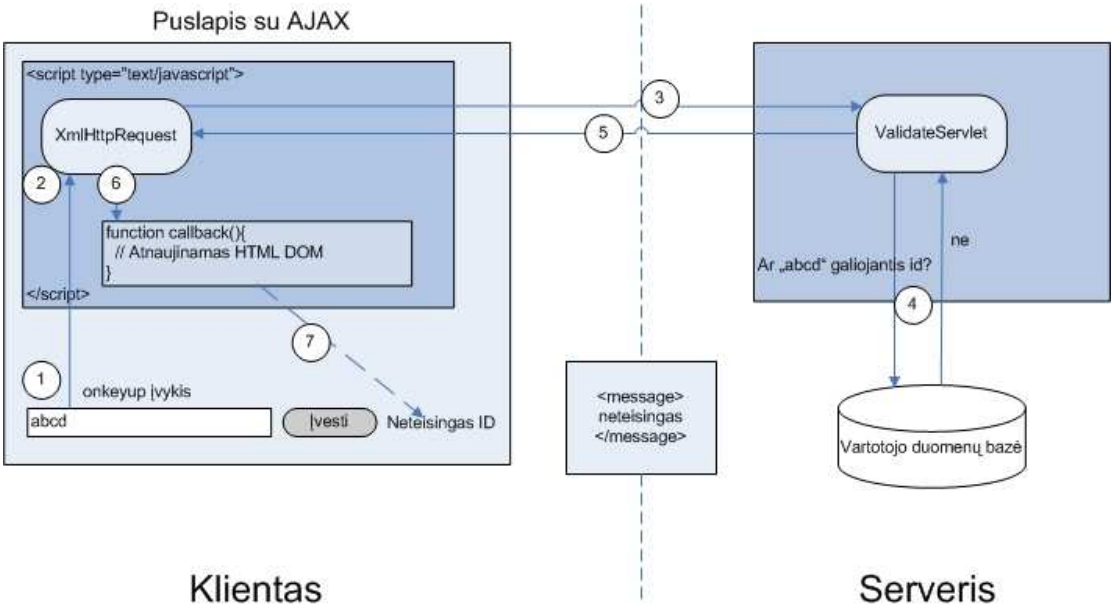
Flash technologija paremtos AJAX programos tinkamiausios norint sukurti išskirtinį dizainą. Dėl to šis būdas mėgstamas grafikos dizainerių. Šia technologija sukurtos sąsajos būna labai interaktyvios, dėl ko Flash varikliukas dar vadinamas „filmų varikliuku“. Pagrindiniai trūkumai būtų ActionScript ir ribotas Flash varikliuko galimybės. Nedidelėms užduotims ActionScript yra labiau tinkamas, bet sudėtingoms programoms – tai keblu ir brangu.

Nėra abejonių, kad kiekviena iš technologijų gali gerai atlikti darbą tam tikroje srityje, kaip ir tai, kad pasirinkus netinkamą būdą – darbas bus varginantis.

Ajax pristato bendrinį programų modelį, kuris suteiktų internetinėms programoms daugiau interaktyvumo, greičio ir sumanumo. Ajax nepririštas prie konkrečios programavimo kalbos, duomenų formato ir tinklo objekto dėl šių savo savybių: dalinio vaizdo atnaujinimo ir asinchroninio komunikavimo.

3.5. Ajax veikimo principas

AJAX sąsajos veikimo principas bus paaiškintas paprastu pavyzdžiu. Esminės metodo savybės nuo pavyzdžio sudėtingumo nepriklauso, todėl aiškinant bus apsiribota paprastu pavyzdžiu. Tarkim, turime internetinį puslapį, į kurį įvedus vartotojo identifikatorių yra nustatoma, ar toks vartotojo id jau egzistuoja ar ne. Vartotojas yra informuojamas su kiekvienu klaviatūros mygtuko paspaudimu (kadangi su kiekvienu paspaudimu pasikeičia įvestas identifikatorius). Žemiau pateiktoje diagramoje pavaizduota tokios programos, sukurtos AJAX principu, veikimo schema.



5 pav. Ajax veikimo principas

Schemoje sunumeruoti veiksmai ir įvykiai:

- Įvykis kliento pusėje (1);
- XMLHTTPRequest objekto sukūrimas ir sukonfigūravimas (2);
- XmlHttpRequest objektas kreipiasi į serverį (3);
- Kreipimąsi apdoroja ValidateServlet funkcija (4);
- ValidateServlet grąžina rezultatą xml formatu (5);
- callback funkcijos kvietimas ir rezultato apdorojimas (6);
- HTML DOM atnaujinimas (7).

Įvykis kliento pusėje

JavaScript technologijos funkcijos yra kviečiamos įvykus kokiam nors įvykiui. Šiuo atveju, funkcija validate() gali būti susieta su onkeyup įvykiu, kuris įvyksta paspaudus nuorodą ar formos komponentą:

```
<input type="text"
           size="20"
             id="userid"
           name="id"
        onkeyup="validate();">
```

*6 pav. Formos komponento **onkeyup** įvykis susietas su validate() funkcija.*

Šis formos elementas iškvies funkciją ***validate()*** kiekvieną kartą vartotojui paspaudus klaviatūros mygtuką formos redagavimo laukelyje.

XMLHTTPRequest objekto sukūrimas ir sukonfigūravimas

```
var req;

function validate() {
   var idField = document.getElementById("userid");
   var url = "validate?id=" + encodeURIComponent(idField.value);
   if (typeof XMLHttpRequest != "undefined") {
       req = new XMLHttpRequest();
   } else if (window.ActiveXObject) {
       req = new ActiveXObject("Microsoft.XMLHTTP");
   }
   req.open("GET", url, true);
   req.onreadystatechange = callback;
   req.send(null);
}
```

7 pav. Validate() funkcija

Validate funkcija sukuria XmlHttpRequest objektą ir kviečia objekto funkciją „open“. Šiai funkcijai reikia perduoti tris parametrus: HTTP metodo tipą (GET arba POST), serverio pusės komponento URL adresą (komponento, su kuriuo objektas bendraus), ir loginį kintamąjį parodantį ar kvietimas į serverį bus atliktas asinchroniškai:

```
XMLHttpRequest.open(String method, String URL, boolean asynchronous)
```

8 pav. Užklausos siuntimas į serverį

Jei sąveika nustatyta į asinchroninę (**true**), turi būti nurodyta grįžtamojo ryšio funkcija. Ši funkcija nurodoma taip kaip parodyta 9 pav.

```
req.onreadystatechange = callback;
```

9 pav. Grįžtamojo ryšio funkcijos nudorymas

XmlHttpRequest objektas kreipiasi į serverį

Pasiekus sakinį req.send(null), bus atliekamas kreipimasis į serverį. HTTP GET atveju, turinys gali būti null arba tuščias. Iškvietus šią funkciją XmlHttpRequest objektui, URL adresas buvo nustatytas objekto konfigūracijos metu. Šiuo atveju, duomenys, kuriuos norima persiųsti yra įtraukti į URL parametrus.

HTTP GET naudojamas tada, kai dvi vienodos užklausos grąžins tuos pačius rezultatus. Kai kurios naršyklės ir internetinės talpyklos naudojant GET metodą, riboja URL adreso ilgį (kartu su parametrais). HTTP POST metodas turėtų būti naudojamas tada, kai siunčiami duomenys paveiks serveryje esančios programos būseną. HTTP POST metodui reikia nurodyti Content-Type antraštę tokiu būdu:

```
req.setRequestHeader("Content-Type", "application/x-www-form-
urlencoded");
req.send("id=" + encodeURIComponent(idTextField.value));
```

10 pav. HTTP antraštės nustatymas

Siunčiant duomenis JavaScript technologijos pagalba, reiktų atkreipti dėmesį į reikšmių kodavimą (tam yra sukurta funkcija encodeURIComponent, kuri turėtų būti naudojama norint užtikrinti, kad tam tikras turinys yra atitinkamos koduotės ir tam tikri simboliai yra teisingai perduodami HTTP užklausos metu).

Kreipimąsi apdoroja ValidateServlet funkcija

Servletas susietas su URI „validate“ patikrina ar vartotojo identifikatorius yra vartotojų duomenų bazėje.

Servletas apdoroja XmlHttpRequest objekto kreipinį lygiai taip pat kaip ir bet kokią kitą HTTP užklausą. Sekantis pavyzdys rodo kaip serverio pusėje apdorojamas id parametras:

```
public class ValidateServlet extends HttpServlet {

    private ServletContext context;
    private HashMap users = new HashMap();

    public void init(ServletConfig config) throws ServletException {
        super.init(config);
```

```
        this.context = config.getServletContext();
        users.put("aaaa","account data");
        users.put("abcd","account data");
    }

    public void doGet(HttpServletRequest request, HttpServletResponse
response)
        throws IOException, ServletException {

        String targetId = request.getParameter("id");

        if ((targetId != null) && !users.containsKey(targetId.trim()))
{
            response.setContentType("text/xml");
            response.setHeader("Cache-Control", "no-cache");
            response.getWriter().write("<message>valid</message>");
        } else {
            response.setContentType("text/xml");
            response.setHeader("Cache-Control", "no-cache");

response.getWriter().write("<message>neteisingas</message>");
        }
    }
}
```

11 pav. Užklausos apdorojimo funkcija serverio pusėje

ValidateServlet grąžina rezultatą xml formatu

Vartotojo identifikatorius „abcd“ jau yra vartotojų sąraše, todėl ValidateServlet funkcija sukurs XML dokumentą su ***message*** elementu ir reikšme „neteisingas“. Sudėtingesniems atvejams gali prireikti DOM, XSLT ar kitų technologijų norint sugeneruoti rezultatą.

```
response.setContentType("text/xml");
response.setHeader("Cache-Control", "no-cache");
response.getWriter().write("invalid");
```

12 pav. Atsako siuntimas iš serverio į klientą.

Programuotojas turi žinoti du dalykus: Content-Type reikšmė turi būti nustatyta į „text/xml“. Antra, Cache-Control turi turėti reikšmę: „no-cache“. XmlHttpRequest objektas

galės apdoroti tik „text/html“ tipo užklausas ir Cache-Control su reikšme „no-cache“ garantuos kad rezultatas nebus saugomas naršyklėje (svarbu, kai ta pati užklausa turi grąžinti skirtingus rezultatus).

Callback funkcijos kvietimas ir rezultato apdorojimas

XmlHttpRequest objektas buvo sukonfigūruotas kviesti callback() funkciją kai XmlHttpRequest objekto savybė readyState pasikeičia. Tarkim, kvietimas į ValidateServlet buvo atliktas ir readyState turi reikšmę 4(kas rodo, kad XmlHttpRequest kvietimas baigtas). HTTP būsenos kodas 200 rodo, kad HTTP sąveika buvo atlikta sėkmingai.

```
function callback() {
    if (req.readyState == 4) {
        if (req.status == 200) {
    // atnaujinti HTML DOM priklausomai nuo gauto pranešimo
        }
    }
}
```

13 pav. callback funkcija

Naršyklės palaiko dokumento objekto atvaizdavimą. JavaScript technologija HTML puslapyje gali prieiti prie DOM bei keisti jį po to kai puslapis buvo užkrautas.

Po sėkmingos užklausos, JavaScript kodas gali modifikuoti HTML puslapio DOM objektą. Xml dokumento, kuris buvo gautas iš ValidateServlet, objekto atvaizdavimas yra prieinamas per XmlHttpRequest objekto lauką responseXML. DOM API turi priemones naudojant JavaScript technologiją eiti per dokumento turinį ir naudoti jį puslapio DOM redagavimui. Tekstinis XML dokumento atvaizdavimas yra prieinamas per req.responseText. Tarkim, užklausa į serverį grąžino tokį rezultatą:

```
<message>
  teisingas
</message>
```

14 pav. Grąžinamas rezultatas

Funkcija parseMessages() apdoroja XML dokumentą, kurį gavo iš ValidateServlet. Ši funkcija iškvies setMessage funkciją, kuriai bus perduota message elemento reikšmė. SetMessage atnaujins HTML DOM‘ą.

```
function parseMessage() {
 var message = req.responseXML.getElementsByTagName("message")[0];
 setMessage(message.childNodes[0].nodeValue);
}
```

15 pav. XML dokumento apdorojimas

HTML DOM objekto atnaujinimas

Su JavaScript galima pasiekti bet kokį elementą, esantį HTML DOM‘e. Geriausiai tai padaryti kaip parodyta 16 pav.

```
document.getElementById("identifikatorius")
```

16 pav. Elemento suradimas pagal identifikatorių.

Kur „identifikatorius“ yra HTML elemento ID atributas. Turint nuorodą į elementą, galima keisti jo atributus, stiliaus savybes bei pridėti ar atimti į elementą įeinančius kitus elementus.

Bendriausias būdas pakeisti elemento turinį yra naudojant jo innerHTML savybę (žiūrėti 17 pav).

```
<script type="text/javascript">
...
function setMessage(message) {
    var mdiv = document.getElementById("userIdMessage");
    if (message == "invalid") {
       mdiv.innerHTML = "<div style=\"color:red\">Neteisingas vartotojo
Id</ div>";
    } else {
       mdiv.innerHTML = "<div style=\"color:green\">Teisingas vartotojo
Id</ div>";
    }
}
</script>
<body>
<div id="userIdMessage"></div>
</body>
```

*17 pav. Elemento turinio keitimas naudojant **innerHTML** elementą.*

HTML puslapio dalys, kurios buvo paveiktos pokyčių, yra atvaizduojamos iš naujo. Jei ***innerHTML*** savyje turi tokius elementus kaip <image> arba <iframe>, jų nurodytas turinys yra sugeneruojamas iš naujo taip pat.

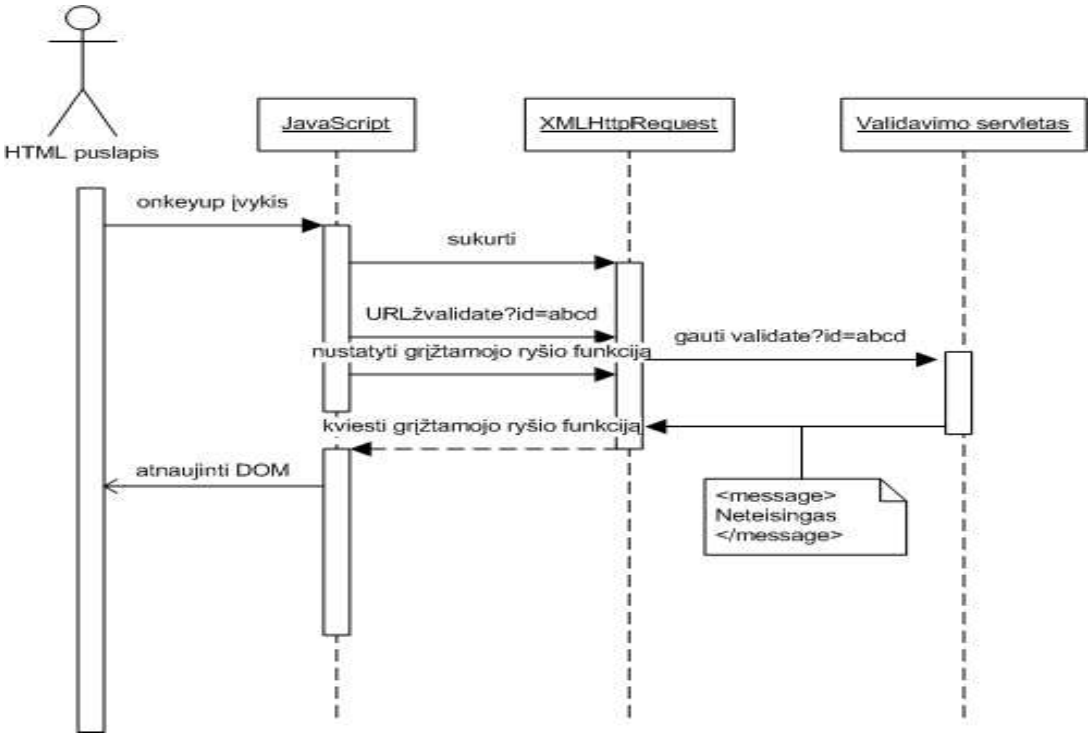
Pagrindinis šio metodo trūkumas yra tas, kad HTML elementai yra tiesiogiai įrašomi į JavaScript kodą, dėl ko pasidaro sunku skaityti, palaikyti ir redaguoti kodą.

Kita priežastis, kodėl redaguojamas HTML DOM'as, tai galimybė sukurti ir įterpti naujus elementus (18 pav).

```
<script type="text/javascript">
...
function setMessage(message) {
     var userMessageElement = document.getElementById("userIdMessage");
     var messageText;
     if (message == "invalid") {
         userMessageElement.style.color = "red";
         messageText = "Invalid User Id";
     } else {
         userMessageElement.style.color = "green";
         messageText = "Valid User Id";
     }
     var messageBody = document.createTextNode(messageText);
     // if the messageBody element has been created simple replace it
otherwise
     // append the new element
     if (userMessageElement.childNodes[0]) {
         userMessageElement.replaceChild(messageBody,
userMessageElement.childNodes[0]);
     } else {
         userMessageElement.appendChild(messageBody);
     }
}
</script>
<body>
<div id="userIdMessage"></div>
</body>
```

18 pav. Programinis puslapio elementų kūrimas ir keitimas.

Aukščiau aprašytą veiksmų seką galima atvaizduoti sekų diagrama, pateikta 19 pav.



19 pav. Ajax sąveikos sekų diagrama

3.6. AJAX duomenų formatai

Populiariausios yra šios AJAX duomenų formatų alternatyvos:

- XML;
- HTML;
- JSON.

3.6.1. XML formatas

XML yra aiškiausias ir savaime suprantamas pasirinkimas naudoti kaip duomenų formatą šiuo atveju. Pirminė XMLHttpRequest objekto paskirtis buvo XML dokumentų importavimas, todėl nenuostabu, kad tai duomenų formatas pagal nutylėjimą.

Jei serveris duomenis grąžintų XML formatu, tai jie galėtų atrodyti taip:

```
<knygos>
        <knyga>
            <pavadinimas>Avies medžioklė</pavadinimas>
            <leidykla>Baltos lankos</leidykla>
            <autorius>Haruki Murakami</autorius>
            <virselis
```

```
src="http://www.baltoslankos.lt/book_images/25_1.jpg"/>
      </knyga>
      <knyga>
            <pavadinimas>Dansu Dansu Dansu</pavadinimas>
            <leidykla>Baltos lankos</leidykla>
            <autorius>Haruki Murakami</autorius>
            <virselis
src="http://www.baltoslankos.lt/book_images/444_1.jpg"/>
      </knyga>
       <knyga>
           <pavadinimas>Norvegų giria</pavadinimas>
           <leidykla>Baltos lankos</leidykla>
           <autorius>Haruki Murakami</autorius>
           <virselis
src="http://www.baltoslankos.lt/book_images/1041_1.jpg"/>
      </knyga>
</knygos>
```

20 pav. Duomenys apie knygas XML formatu.

Tačiau šis formatas nėra patogus atvaizdavimo atžvilgiu – pakankamai daug kodo paprasčiausio sąrašo atvaizdavimui (21 pav):

```
function setDataXML(req)
{
      var books = req.responseXML.getElementsByTagName('knyga');
      for (var i=0;i<books.length;i++)
      {
            var x = document.createElement('div');
            x.className = 'knyga';
            var y = document.createElement('h3');
            y.appendChild(document.createTextNode(getNodeValue(books[i],
               'pavadinimas')));
            x.appendChild(y);
            var z = document.createElement('p');
            z.className = 'moreInfo';
            z.appendChild("Autorius: "+getNodeValue(books[i],'autorius')
               + ', Išleido: ' + getNodeValue(books[i],'leidykla')));
            x.appendChild(z);
            var a = document.createElement('img');
            a.src = books[i].getElementsByTagName(
               'virselis')[0].getAttribute('src');
```

```
            x.appendChild(a);
            document.getElementById('writeroot').appendChild(x);
      }
}

function getNodeValue(obj,tag)
{
      return obj.getElementsByTagName(tag)[0].firstChild.nodeValue;
}
```

21 pav. Duomenų, gaunamų XML formatu, apdorojimo pavyzdys.

Svarbiausias šio formatas privalumas – tai skaitomumas ir suprantamumas. Kitas šio formato privalumas, kad XML yra naudojamas ir kitais atvejais, todėl tai jau įsisavinta technologija. Didžiausias trūkumas, kad reikia papildomo Javascript kodo gautų duomenų atvaizdavimui.

3.6.2. HTML kodo ištraukos

Vienas įdomiausių duomenų formatų - HTML kodo ištrauka. Naudojant šį duomenų formatą, negaunamas visas HTML puslapis, o tik dalis jo. Tiksliau, gaunamas ta dalis, kuri turi būti įterpta.

Pavyzdžiui, serveris grąžina tokią HTML kodo ištrauką:

```
<div class="knyga">
      <h3>Avies medžioklė</h3>
      <p class="autorius"> Autorius: Haruki Murakami</p>
      <img src=”http://www.baltoslankos.lt/book_images/25_1.jpg"/>
</div>
<div class="knyga">
      <h3>Dansu dansu dansu</h3>
      <p class="autorius">Autorius: Haruki Murakami</p>
      <img src="http://www.baltoslankos.lt/book_images/444_1.jpg"/>
</div>
<div class="knyga">
      <h3>HTML Norvegų giria</h3>
      <p class="autorius"> Autorius: Haruki Murakami</p>
      <img src="http://www.baltoslankos.lt/book_images/1041_1.jpg"/>
</div>
```

22 pav. Duomenų pavyzdys XML formatu.

Kodas gautos duomenų ištraukos atvaizdavimui yra labai paprastas: innerHTML savybei priskiriamas XMLHttpRequest objekto responseText reikšmė:

```
function setDataHTML(req)
{
	document.getElementById('knygos').innerHTML = req.responseText;
}
```

23 pav. Duomenų, gautų XML formatu, atvaizdavimas.

Šio duomenų perdavimo metodo pagrindinis privalumas - kodo paprastumas. Be to, tai leidžia gautus duomenis panaudoti ne tik Ajax technologijoje. Tačiau, jei HTML kodo ištrauka turi formos elementų, Internet Explorer naršyklėje, tai sukelia klaidų atsiradimą. Verta paminėti, kad šis duomenų formatas patogus tik esant nedidelėms duomenų struktūroms. Sudėtingėjant grąžinamiems duomenims, sudėtingėja serverio pusėje HTML generuojantis kodas.

3.6.3. JSON

Pagrindinė šio formato idėja buvo tekstą panaudoti kaip Javascript objektą. Gavus tokio tipo duomenis, naudojama Javascript funkcija eval(), kuri paverčia gautus duomenis į tikrą JavaScript objektą, kurį po to galima nuskaityti. Pavyzdžiui, serveris galėtų grąžinti tokius duomenis JSON formatu:

```
{"knygos":[{"knyga":
            {
            "pavadinimas":"Avies medžioklė",
            "leidykla":"Baltos lankos",
            "autorius":"Haruki Murakami",
            "viršelis":"http://www.baltoslankos.lt/book_images/25_1.jpg",
            }
      },
      {"knyga":
            {
            "pavadinimas":"Dansu Dansu Dansu",
            "leidykla":"Baltos lankos",
            "autorius":"Haruki Murakami",
```

```
            "virselis":"ttp://www.baltoslankos.lt/book_images/444_1.jpg",
            }
      },
      {"knyga":
            {
            "pavadinimas":"Norvegų giria",
            "leidykla":"Baltos lankos",
            "autorius":"Haruki Murakami",
            "virselis":"
http://www.baltoslankos.lt/book_images/1041_1.jpg",
            }
      }
]}
```

24 pav. Duomenų JSON formatu pavyzdys.

Kodas duomenų nuskaitymui atrodo panašus (sudėtingumo atžvilgiu taip pat) į tą, kuris naudojamas XML duomenų nuskaitymui:

```
function setDataJSON(req)
{
      var data = eval('(' + req.responseText + ')');
      for (var i=0;i<data.knygos.length;i++)
      {
            var x = document.createElement('div');
            x.className = 'knyga';
            var y = document.createElement('h3');
            y.appendChild(document.createTextNode(
               data.knygos[i].knyga.pavadinimas));
            x.appendChild(y);
            var z = document.createElement('p');
            z.className = 'autorius';
            z.appendChild(document.createTextNode('Autorius: '
               + data.knygos[i].knyga.autorius + ', Išleido: '
               + data.knygos[i].book.leidykla));
            x.appendChild(z);
            var a = document.createElement('img');
            a.src = data.knygos[i].knyga.virselis;
            x.appendChild(a);
            document.getElementById('knygos').appendChild(x);
      }
}
```

25 pav. Duomenų, gautų JSON formatu, apdorojimas

Pagrindinis tokio formato privalumas: jis padeda apeiti Javascript „to paties šaltinio“ politiką, jei JSON dokumentas bus importuotas su <script> žyme. JavaScript neleidžia pasiekti dokumentų (XML ar HTML), kurie nėta tame pačiame serveryje. Vis gi, jei importuojamas JSON formato dokumentas su <script> žyme, ši problema yra apeinama: bet kokie JSON duomenys gali būti įtraukti į bet kurį puslapį. Šiuo metu tai vienintelis duomenų formatas, leidžiantis tokį neribotą priėjimą. Kitas privalumas – kodas JSON duomenims apdoroti yra šiek tiek paprastesnis nei XML duomenys. Vis gi, šis formatas yra sunkiai skaitomas žmonėms, labai svarbu, kad visi skiriamieji ženklai šiuo formatu aprašant duomenis būtų savo vietose.

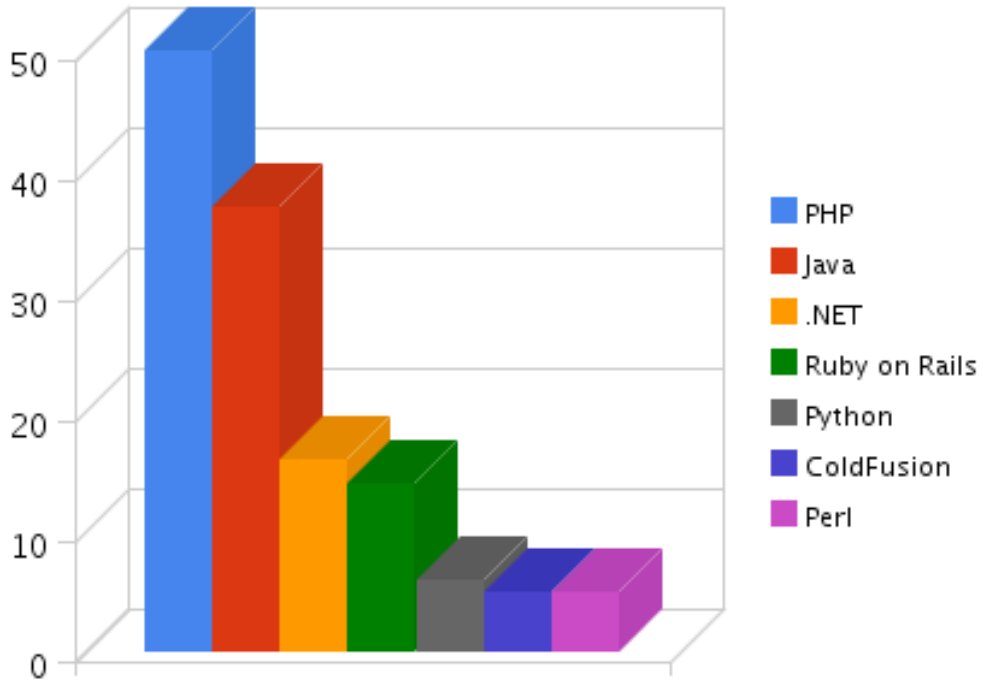
4. Asinchroninės sąsajos įrankiai

4.1. Ajax platformos

Pagal http://www.ajaxian.com portalo atliktos apklausos rezultatus, populiariausios Ajax platformos yra:

- PHP
- Java
- .NET
- Rails
- Python
- ColdFusion
- Perl

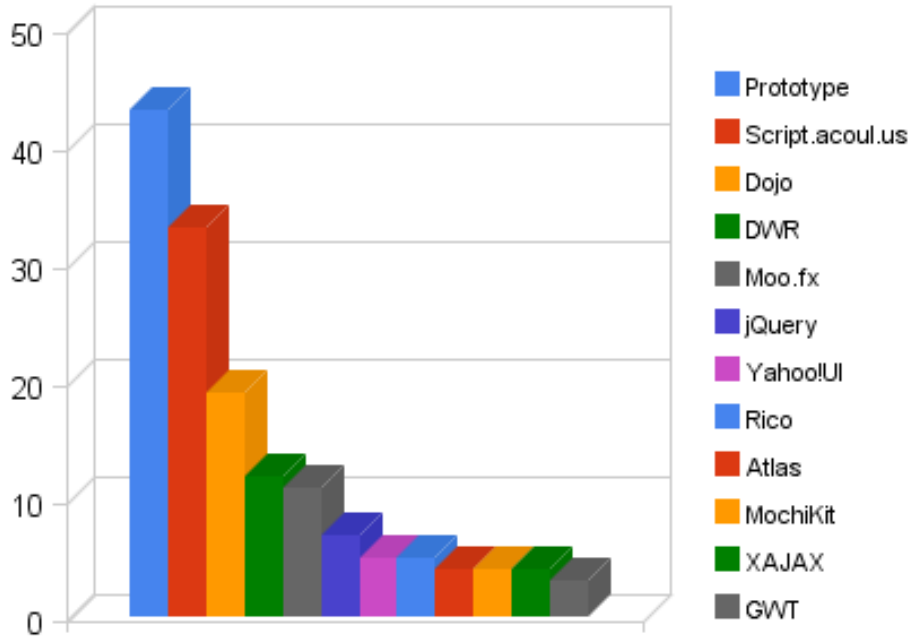
Jų pasiskirstymas pavaizduotas 26 pav paveiksle.



26 pav. Populiariausios AJAX platformos.

4.2. *AJAX bibliotekos*

Portalas http://www.ajaxian.com pateikia populiariausių Ajax bibliotekų rezultatus (žiūrėti 27 pav). Žinomi tokios apklausos rezultatai nėra labai tikslūs, bet jie atspindi šiuolaikines tendencijas šios technologijos pasaulyje, kadangi apklausą atlikinėjęs šaltinis yra vertinamas tos srities profesionalų.



27 pav. Ajax bilbiotekų naudojamumo pasiskirstymas

Egzistuojančias Ajax bibliotekas būtų galima suskirstyti į 2 tipus:

- Vien JavaScript bibliotekos;
- Serveriu paremtos Ajax bibliotekos

Pirmo tipo bibliotekos turi esminius nuotolinio programavimo įrankius ir funkcijas. Antrojo tipo – generuoja Ajax kodą iš serverio pusės. Kiekvienas bibliotekos tipas turi savo pritaikymo sritį ir yra geriausiai tinkamas tam tikrais atvejais. Taip pat galima jas galima tarpusavyje derinti. Vis gi, norint pasirinkti kurio tipo bibliotekas naudoti, reikia žinoti jų privalumus ir trūkumus (Lentelė Nr. 2).

Lentelė Nr. 2 AJAX bibliotekų tipų palyginimai

Bibliotekos tipas	Privalumai	Trūkumai
Serveriu paremtos	Gali susieti serverio objektus	Stipriai susieja JavaScript

	su Javascript objektų atitikmenimis; Sumažina Javascript kodo naudojimą;	kodą su serverio kalba;
Vien JavaScript bibliotekos	Gali susidoroti su keliomis serverio kalbomis; Aiškesnis kliento ir serverio kodo atskyrimas;	Pačiam programuotojui reikia paruošti duomenis grąžinimui (JSON arba XML formatu);

4.2.1. AJAX bibliotekų ir įrankių palyginimas

Žemiau (Lentelė Nr. 3) pateiktos pripažintos AJAX bibliotekos ir įrankiai, bei jų palyginima bendrais bruožais.

Lentelė Nr. 3 Įvairūs AJAX įrankiai ir bibliotekos

Įrankis	Tipas	Privalumai	Trūkumai
Google Web Toolkit (GWT)	Java paremta; integruota Ajax biblioteka	Lengvai plečiamas	Paremtas Java; ne pati geriausia licenzija;
Yahoo UI biblioteka	Javascript paremta; vartotojo sąsajos biblioteka	Plačiai naudojamas; išsami biblioteka; geras palaikymas.	CSS stilius nevaliduojamas; Veikimas skiriasi nuo naršyklės;
Dojo	JavaScript paremta; integruota Ajax biblioteka	Populiarus; geras efektų ir įrankių derinys	Neveikia su Opera naršykle; šykšti dokumentacija.
DWR (Direct Web Remoting)	Java paremta; integruota Ajax biblioteka	Gera integracija su java; išsami dokumentacija; palaiko daugelį naršyklių; automatiškai generuojami testiniai puslapiai	Paremta Java;

JSON-RPC-JAVA	Java paremtas;	Lengva naudoti; Susieja Java objektus su Javascript objektais; Palaiko Internet Explorer, Mozilla, Firefox, Safari, Opera ir Konqueror naršykles;	Sunkiai skaitomas JSON duomenų formatas; Plečiamumo problemos dėl HTTPSession naudojimo;
MochiKit	Javascript paremta; integruota Ajax biblioteka	Suprantama dokumentacija ir demo programos; gerai apgalvota ir draugiška programuotojų atžvilgiu; Palaiko naujausias populiariausių naršyklių versijas;	Labai paprasti biblioteka; Kai kurie efektai yra beta stadijoje;
OpenRico(Rico)	JavaScript paremta; integruota Ajax biblioteka	Gera pusiausvyra tarp Javascript efektų ir Ajax funkcijų; sklandus veikimas.	Nėra jokios dokumentacijos (dėl to sunku išmokti ja naudotis);
Sajax	Javascript paremtas; Integruota Ajax biblioteka;	Palaiko daugumą programavimo kalbų;	Apribotas tik Ajax palaikymu (tiksliau XMLHttpRequest objektu); Nėra vaizdinių efektų bibliotekos; Nėra Java ir JSP palaikymo;
Script.aculo.us	JavaScript paremta; integruotos Ajax ir Vartotojo sąsajos bibliotekos	Puikus efektų ir įrankių rinkinys; Paprastas ir objektiškai orientuotas dizainas; paremta Ruby On Rails platforma (bet	Šykšti Ajax funkcijų aibė;

		nepriklauso nuo jos); Geras bendruomenės palaikymas;	
The X library	Javascript paremta; efektų ir vartotojo sąsajos biblioteka	Geriausi efektai	Ne pati geriausia vartotojo sąsajos biblioteka; užima nemažai vietos;

4.2.2. Prototype biblioteka

Atskiro dėmesio yra verta Prototype biblioteka. Šios bibliotekos populiarumą rodo tai, kad ji labai dažnai yra sudedamoji kitų Ajax bibliotekų dalis. Nors ji yra orientuota į Ruby On Rails platformą, bet gali būti pritaikoma bet kuriai kitai. Būtų galima paminėti tokius privalumus:

- Pakankamai universali (naudojama daugelyje Ajax ir JavaScript bibliotekų);
- Suteikia pakankamai gerą priėjimą prie XMLHttpRequest objekto;
- Gerai dokumentuota;

  Savaime suprantama, kad ji turi ir minusų:
- IE naršyklės palaikymas apribotas 6 versija;
- Neturi vaizdinių efektų bibliotekos;
- Neturi integruoto būdo naudoti Ajax rezultatą kaip XML dokumentą;
- Pakeitimai Object.prototype objekte sąlygoja problemas for/in cikluose;

4.3. Kūrimo įrankių problematika

Nors specializuota programavimo aplinka, norint sukurti asinchroninę sąsaja, nėra būtina, vis gi tokie įrankiai sutrumpina produkto patekimo į rinką laiką. Paprastai jie susieja kuriamą produktą su konkrečia biblioteka, kas gali būti nepageidaujama (ir prieš ką nublanksta visi specializuotos kūrimo aplinkos privalumai). Verta pažymėti, kad nėra vienos geriausios aplinkos, skirtos tokio tipo sistemų kūrimui. Vieniems programuotojams teikiami tokių sistemų privalumai gali tapti trūkumais kitiems programuotojams. Programavimo aplinka tokių sistemų kūrimui nėra esminis elementas.

4.4. Asinchroninės sąsajos generavimas

Asinchroninės sąsajos redaktorius – įrankis, palengvinantis tokių sąsajų kūrimo procesą, padedantis išvengti pasikartojančių veiksmų bei kūrimo procese atsirandančių klaidų, jei kūrimo procesas pakankamai nusistovėjęs.

Galimi sprendimai:

- darbastalio programa;
- internetinė sistema paremta AJAX;

4.4.1. Reikalavimai sistemai

Vartotojas turi galėti:

- keisti sąsajos elementų išdėstymą;
- keisti sąsajos elementų stiliaus parametrus;
- redaguoti sąsajos sugeneruotą kodą;
- priskirti funkcijas konkretiems sąsajos elementų įvykiams;
- įterpti naujus sąsajos elementus, pašalinti senus;
- išsaugoti sukurtą sąsają į per internetą prieinamą talpyklą, taip pat keisti ir šalinti talpykloje esančias sąsajas;

Elementai, kuriuos vartotojas turi galėti įterpti į sąsają:

- duomenų siuntimo/užklausimo elementai:

  - submit tipo HTML elementas <input>;
  - HTML elementas <a>);
- redagavimo elementai:
  - text tipo HTML elementas <input>);
  - HTML elementas <textarea>;
- pasirinkimo elementai:
  - radio tipo HTML elementas <input>;
  - checkbox tipo HTML elementas <input>;
  - HTML elementas <select>;
- HTML elementas <img>;

Elementams galima priskirti tokius įvykius:

- Onkeydown (įvykis, atsirandantis nuspaudus bet kurį klaviatūros mygtuką);
- Onkeypress (įvykis, atsirandantis nuspaudus ir atleidus bet kurį klaviatūros mygtuką);
- Onkeyup (įvykis, atsirandantis atleidus bet kurį klaviatūros mygtuką);
- Onclick (įvykis, atsirandantis paspaudus pelės mygtuką);
- Onmouseover (įvykis, atsirandantis pelės žymekliui atsidūrus virš elemento);
- Onmouseup (įvykis, atsirandantis atleidus pelės mygtuką);

Keičiami elementų parametrai:

- class (elemento klasė
- id (elemento identifikatorius);
- style (elemento stilius);
- title (elemento pavadinimas);

Keičiami elementų stiliaus parametrai:

- background-color (parametras nurodantis elemento fono spalvą);
- color (parametras nurodantis elemento teksto spalvą);
- text-align (elementas nurodantis teksto išlygiavimą; galimos reikšmės: left, right, center, justify);
- font (elementas nurodantis teksto šriftą bei dydį);

Asinchroninės sąsajos redaktoriui būtini tokie įrankiai:

- elementų pasirinkimo juosta;

- grafinės vartotojo sąsajos darbastalis;
- sugeneruoto kodo langas;
- elementų parametrų ir įvykių langas;

Elementų pasirinkimo juosta – tai sąrašas elementų, kuriuos galima įterpti į kuriamą sąsają. Šiuo atveju, tai paprasti HTML elementai, bet taip pat gali būti ir sudėtingesnis atvejai:pvz., save redaguojantis teksto įvedimo laukelis;

Grafinės vartotojo sąsajos darbastalis – langas, kuriame galima įterpti, keisti ir šalinti sąsajos elementus. Visi veiksmai atliekami pele.

Sugeneruoto kodo langas – langas, kuriame matosi sugeneruotas sąsajos kodas. Įterpus naują elementą, paredagavus ar pašalinus seną, šiame lange esantis tekstas atsinaujina automatiškai. Taip pat, sąsają galima redaguoti ir šiame lange. Čia atlikti pakeitimai atsispindi grafinės vartotojo sąsajos darbastalyje.

Elementų parametrų ir įvykių lange galima redaguoti į sąsają įterptų elementų parametrų reikšmes, stiliaus atributus bei su elementais susietus įvykius.

4.4.2. Privalumai ir trūkumai

Galimos tokios generuojamos sąsajos susiejimo su AJAX alternatyvos;

- nenaudoti AJAX bibliotekos;
- naudoti AJAX biblioteką;

Pasirinkus antrąjį variantą turime tokius privalumus:

- greitesnis sistemos sukūrimas;
- patikrintas veikimas;

Tačiau šis metodas nepatogus tuo, kad sugeneruotos sąsajos bus susietos su viena ir ta pačia AJAX biblioteka. Ateityje tai gali tapti suvaržymu, kai prireiks sistema praplėsti tokiu funkcionalumu, kokio bibliotekoje nebus.

Pirmasis variantas atrodo patrauklesnis dėl savo laisvės, bet tuo pačiu yra ir sudėtingesnis:

- ilgesnis sistemos sukūrimo laikas;
- kadangi nebus naudojamos jokios bibliotekos, tokiai sistemai reikės žymiai daugiau testavimo, norint įsitikinti, kad viskas veikia, kaip pridera;
- be to, prireikus naujo funkcionalumo, jį reikės taip pat suprogramuoti ir ištestuoti;

Nepriklausomai nuo to, koks realizavimo variantas būtų pasirinktas, sistemoje bus ir bendrų problemų:

- modelyje nenumatytas serverio pusės kodo generavimas;
- sąsają projektuojantis asmuo turi puikiai išmanyti:
    - kokių duomenų reikia serveryje esančioms funkcijoms;
    - Kokias užklausas siųsti;
    - Kokiu formatu grąžinami rezultatai;
- Redaktorius nesuprojektuotas generuoti saugų sąsajos kodą
- Redaktoriumi sunku generuoti sudėtingas sąsajas;

Asinchroninės sąsajos redaktorius galima alternatyva egzistuojantiems tokios sąsajos kūrimo įrankiams idėjiniame lygmenyje, tačiau vargu ar ji sugebėtų konkuruoti su didesnį biudžetą gaunančiais ir didesnes programuotojų bendruomenes turinčiais sprendimais.

4.5. Technologinės problemos

Vis gi renkantis šią internetinių sistemų realizavimo technologiją, reikia nepamiršti ir šių dalykų:

- Sudėtingumas: Serverio pusės programuotojai turės suprasti, kad atvaizdavimo logika bus privaloma kliento HTML puslapiuose kaip ir serverio pusėje reikalinga sugeneruoti XML turinį, kurio reikės kliento puslapiams;
- XmlHttpRequest objekto standartizavimas: XmlHttpRequest objektas dar nėra JavaScript specifikacijos dalis, tai reiškia, kad veikimas gali keistis priklausomai nuo naršyklės. Geriausia naudoti tokias bibliotekas kaip Dojo, kurios užtikrina kad XmlHttpRequest objektas veiks vienodai visose naršyklėse, net ir senesnėse;
- JavaScript technologijos realizacijos: Ajax sąveika yra stipriai priklausoma nuo JavaScript‘o, kurio veikimas skiriasi skirtingose naršyklėse;
- Testavimas: AJAX programos yra sunkiai testuojamos, kadangi apdorojimo logika yra integruota tiek į klientą, tiek į serverį. Tačiau, egzistuoja naršyklių priedai (pvz., Mozilla Firebug) leidžiantys palengvinti šią užduotį;
- Resursų ir duomenų apsauga: kliento pusės JavaScript kodas gali būti lengvai peržiūrimas bet kurioje naršyklėje;
- Daugelyje naršyklių dinamiškai sukurtas puslapi neužregistruojamas naršyklės istorijos variklyje – paspaudus “atgal” mygtuką rezultatai gali būti ne tokie, kokių tikėtasi.
- Dinamiškai sukurtame puslapyje sunku išsaugoti programos būseną.

4.6. Našumas

4.6.1. JavaScript ir Ajax veikimo sparta

Nesvarbu, kokia sukurta programa būtų naudinga, ji turi veikti pakankama greitai. Kompiuterinės programos veikimo sparta priklauso nuo to, kaip greitai ji veikia ir kiek sistemos resursų (atmintis, CPU krūvis) ji sunaudoja. Savaime suprantama, kad per lėtos programos erzina vartotojus.

JavaScript nėra greita programavimo kalba. Atminties atžvilgiu šios kalbos objektai nėra “lengvi“, o ypač daug atminties reikia DOM objektams.

Ajax veikimo greitis gali būti optimizuojamas trimis būdais:

- for ciklo optimizavimu;
- DOM mazgų prijungimas prie dokumento;
- Žymėjimo per tašką minimizavimas;

For ciklai gali būti optimizuojami iškeliant veiksmus, kurie grąžina/generuoja tas pačias kintamųjų reikšmes, už ciklo ribų. Suprantama, kad optimizavimo rezultatai priklauso nuo konkretaus algoritmo sudėtingumo ir probleminės srities. Gali būti atvejų, kai for ciklų optimizavimas yra neįmanomas.

4.6.1.1. DOM mazgų prijungimas prie dokumento

Norint ką nors atvaizduoti naršyklės lange panaudojant Ajax technologiją, yra sukuriami DOM mazgai ir prijungiami prie dokumento medžio. Kai tik prijungiama dalis tampa dokumento dalimi, DOM mazgas yra atvaizduojamas. Nėra jokio būdo šios savybės išjungimui. Pakartotiniam dokumento atvaizdavimui naršyklės lange, reikia, kad būtų perskaičiuojami įvairūs išdėstymo parametrai, ir tai yra labai brangu laiko atžvilgiu. Jei yra kuriama sudėtinga vartotojo sąsaja, logiškiausia būtų sujungti visus mazgus tarpusavyje ir tik po to surinktą struktūra prijungti prie dokumento. Tokiu būdu, puslapio išdėstymo apdorojimas vyksta tik vieną sykį.

4.6.1.2. Žymėjimo per tašką optimizavimas

JavaScript kalboje kaip ir daugelyje kitų, kintamuosius, esančius giliai objektų hierarchijoje, galima pasiekti „per tašką“, pvz.:

```
Tevas.laikrodis.rodykles.minutes
```

28 pav. Giliai objekto hierarchijoje esančių elementų pasiekimas per tašką.

Šiame pavyzdyje, norint gauti visų trijų rodyklių reikšmes būtų galima tai padaryti taip:

```
var valanduRodykle = tevas.laikrodis.rodykles.valandos;
var minuciuRodykle = tevas.laikrodis.rodykles.minutes;
var sekundziuRodykle = tevas.laikrodis.rodykles.sekundes;
```

29 pav. Laikrodžio rodyklių reikšmių gavimas.

Kiekvieną kartą kai interpretatorius randa taško simbolį, jis peržvelgia visus „tėvo“ kintamuosius. Tokiu būdu, gali būti labai daug tokių kreipinių, iš kurių didžioji dalis kartosis. Šiame pavyzdyje, turėsime devynis tokius kreipinius. Jei pavyzdys būtų perrašytas taip:

```
var rodykles = tevas.laikrodis.rodykles;
var valanduRodykle = rodykles.valandos;
var minuciuRodykle = rodykles.minutes;
var sekundziuRodykle = rodykles.sekunde;
```

30 pav. Optimizuotas laikrodžio rodyklių reikšmių gavimas

Šiuo atveju, turime tik 5 kreipinius. Kitose kalbose, interpretatoriai atlieka patys tokį optimizavimą, tačiau JavaScript veikimas priklauso nuo konkrečios naršyklės.

4.6.2. Atminties valdymas

Atminties nutekėjimo problema yra aktuali daugeliui programavimo kalbų, kurios neturi atminties valdymo mechanizmo – „šiukšlių surinkėjo“(angl. "garbage collector“). JavaScript turi tokį mechanizmą, todėl tai išvaduoja nuo daugelio problemų, kurios galėtų kilti dėl neišvalytos atminties. Tačiau klaidinga manyti, kad programavimo kalbos, turinčios atminties valdymo mechanizmus, negali sukelti atminties nutekėjimo.

Norint išvengti atminties nutekėjimų asinchroninės sąsajos sprendimuose (naudojant Ajax technologiją), patartina laikytis tokių principų:

- stengtis nesukurti ciklinių nuorodų į objektus;
- DOM elementų sunaikinimas: programiniu būdu kuriant/naikinant DOM elementus iškyla grėsmė, kad net kai tie elementai nebus naudojami, jie liks nesunaikinti ir vis tiek užims dalį atminties (bent jau tol kol šiukšlių surinkimo procesas pastebės, kad jie yra nenaudojami).

4.7. Saugumo problemos

Saugumas yra pati opiausia interneto problema. Tinkamas saugumo priemonių pritaikymas internetinėse sistemose gal būti išskirtinė ir stipri sukurto produkto savybė. Jei internetinė sistema yra susijusi su vartotojo piniginės informacijos valdymo, apsipirkimu internetu, saugumo užtikrinimas yra privalomas.

4.7.1. Serverio kilmės politika

Paleidus AJAX programą, internetinis serveris siunčia JavaScript instrukcijų rinkinį į interneto naršyklę, veikiančią kitame kompiuteryje, apie kurį serveris žino labai mažai. Leisdamas naršyklei vykdyti atsiųstas instrukcijas, vartotojas turi pasitikėti programa ir jos autoriais. Kadangi toks pasitikėjimas buvo įvertintas kaip ne visada pagrįstas ir tinkamas, standartų ir internetinių naršyklių kūrėjai įdiegė tam tikras apsaugas.

Viskas kompiuterio kietajame diske yra tik dvejetainių duomenų aibė. Vis gi, tuos duomenys galima suskirstyti į paprastus duomenis ir duomenis-kompiuterio instrukcijas. Paprasti duomenys negali nieko padaryti, kol vykdomasis procesas juos apdoros. AJAX internetiniame modelyje, kliento naršyklėje esanti programa tai kompiuterio instrukcijos, o per tinklą siunčiamas duomenų srautas – paprasti duomenys. Programos kodas yra mobilus, jei jis saugomas viename kompiuteryje, bet gali save persiųsti per tinklą ir būti įvykdytas kitame. Kompiuteris priimantis mobilų kodą, turėtų “apsvarstyti” ar priimti tokį kodą (ypač persiųstą internetu).

Mobilus kodas yra pripažintas potencialia internetinių sistemų grėsme. Dėl šios priežasties, naršyklėje vykdomas JavaScript yra vykdomas specialioje aplinkoje su apribotu priėjimu prie kompiuterinių resursų. AJAX programa negali rašyti ir skaityti iš lokalios dokumentų sistemos. Taip pat ji negali prisijungti prie interneto puslapių, esančių kitur negu pati AJAX programa (dauguma atvejų). Programiniu būdu sugeneruotas IFRAME elementas gali užkrauti puslapius iš kitų interneto vardinių sričių ir įvykdyti juose esantį kodą, bet kodai iš dviejų skirtingų vardinių sričių negali sąveikauti tarpusavyje. Tai vadinama “serverio kilmės” politika.

4.7.2. Bendros saugumo problemos

Ajax nesukuria naujų saugumo spragų internetinių programų aibėje: problemos išlieka tos pačios kaip ir klasikinėse internetinėse sistemose. Vis gi, neegzistuojant bendroms AJAX kūrimo normoms, yra paliekama daug vietos įvairioms saugumo problemoms sukurti. Tai apima tinkamą autentifikavimą, autorizaciją, priėjimo kontrolę ir įvesties validavimą. Labiausiai rūpestį kelia:

- Kliento pusės saugumo elementai: Didžiausias pavojus kyla, kai saugumo priemonės realizuojamos klientinėje pusėje, nes bet kas gali pakeisti kodą, jei jis vykdomas naršyklėje. Saugumo elementai privalo būti realizuoti serveryje.
- Padidintas atakų paviršius: AJAX savaime padidina sistemos sudėtingumą - kuo daugiau sistemoje įterpiamų kodo puslapių, tuo daugiau galimybių panaudoti juos blogiems tikslams, ir daugiau rūpesčių užtikrinti, kad taip nenutiktų.

- Tarpo tarp paslaugų ir vartotojų užpildymas: AJAX gali sėkmingai panaudoti internetines paslaugas (angl. „web services“). Tačiau pradinė jų paskirtis nebuvo skirta sąveikai su paprastais vartotojais. Dėl šios priežasties daugelyje tokių internetinių paslaugų trūksta paprasčiausios apsaugos (autentifikavimas ir autorizacija buvo paliekama kitiems sistemos komponentams).
- Programavimas tarp kelių puslapių (XSS): kodo elementų įterpimas iš kelių skirtingų puslapių visada buvo problema, bet anksčiau klasikiniame interneto modelyje, vartotojas galėjo tai pastebėti. AJAX modelyje, tai padaryti yra sunkiau, nes kliento naršyklė gali siųsti duomenis foniniu režimu vartotojui to net nepastebint.

AJAX sistemų saugumo testavimas yra sudėtingas dėl to:

- Daug sunkiau apibrėžti puslapio būseną;
- Kai kurios užklausos gali būti inicijuojamos tam tikrų skaitiklių.
- Dinaminis DOM elementų atnaujinimas;
- Standartiniai testavimo įrankiai dar nėra pritaikyti suprasti AJAX gaunamų duomenų XML formatu;

Asinchroninių internetinių sąsajų programuotojai turėtų žinoti apie naujausias saugumo spragas, gresiančias tokio tipo sistemoms, o testuotojai įsisavinti naujo tipo (ir ganėtinai skirtingą) sistemų testavimo metodologiją.

5. Išvados

1. Klasikinis internetinis modelis nebe patenkina sistemų kūrėjų ir vartotojų poreikių.
2. Asinchroninė sąsaja nebūtinai yra vienintelis teisingas sprendimas naujiems vartotojų ir sistemų autorių poreikiams patenkinti. Prieš renkantis realizavimo technologiją būtina išnagrinėti visus sprendimo variantus ir jų teikiamą naudą bei trūkumus.
3. AJAX nėra konkreti technologija, tai įvairių technologijų rinkinys. Įvairios AJAX realizacijos atmainos skirtos spręsti konkrečioms (ir gana skirtingoms) problemoms.
4. Egzistuojanti programinės įrangos įvairovė asinchroninei sąsajai realizuoti skirta plačiam internetinių sistemų spektrui: tiek mažoms, tiek didelėms.
5. Sukurtas prototipinis asinchroninės sąsajos generavimo modelis padėjo atskleisti, kad tokių įrankių kūrimas – darbas, reikalaujantis įvertinti daug aspektų: saugumą, sąsajų elgseną skirtingose naršyklėse, funkcionalumo reikalavimus būsimai sąsajai. Tokių projektų realizavimui reikia daug finansinių ir žmogiškųjų išteklių.
6. AJAX gali būti galingas įrankis internetinių projektų realizavimui, jei tinkamai juo naudojamasi. Pati technologija savaime neišsprendžia spartos, atminties ir saugumo problemų. Tai - darbas, reikalaujantis didelių standartų kūrėjų, naršyklių ir sistemų projektuotojų bei testuotojų.

6. Naudota literatūra:

1. AJAX Toolkit Shootout. 2006. [žiūrėta 2007-02-01].
Prieiga per internetą: http://summerofcode.wordpress.com/2006/07/01/ajax-toolkit-shootout/
2. Ajaxian.com 2006 Survey Results. 2006. [žiūrėta 2007-04-11].
Prieiga per internetą: http://ajaxian.com/archives/ajaxiancom-2006-survey-results
3. . Dave Crane, Eric Pascarello with Darren James. Ajax in Action. Greenwich: Manning, 2006.
4. David Talbot. Which AJAX Library Is Right for Me?. 2006 [žiūrėta 2007-03-23]. Prieiga per internetą: http://www.devx.com/AJAXRoundup/Article/33142
5. Denis G. Sureau. JSON - JavaScript Object Notation. 2006. [žiūrėta 2007-05-07]. Prieiga per internetą: http://www.xul.fr/ajax-javascript-json.html
6. Greg Murray. AJAX FAQ for the Java Developer. 2006. [žiūrėta 2007-02-10].
Prieiga per internetą: https://blueprints.dev.java.net/ajax-faq.html
7. Jaswinder S.Hayre, Jayasankar Kelath. Ajax Security Basics. 2006. [žiūrėta 2007-01-27].
Prieiga per internetą: http://www.securityfocus.com/infocus/1868
8. Jesse James Garrett. Ajax: A New Approach to Web Applications. 2005. [žiūrėta 2007-02-07]. Prieiga per internetą:
http://www.adaptivepath.com/publications/essays/archives/000385.php
9. Lietuvos Respublikos Terminų bankas. [žiūrėta 2007-05-14]. Prieiga per internetą:
http://terminai.vlkk.lt
10. Steve Flemming, Beth Stearns. An Introduction to Ajax and Java Studio Creator 2. 2006. [žiūrėta 2007-04-12]. Prieiga per internetą:http://developers.sun.com/jscreator/reference/techart/2/ajax_overview.html

7. Santrumpų ir terminų žodynas

AJAX (Asynchronous JavaScript and XML) – Asinchroninio JavaScript‘o ir XML technologija skirta kurti interaktyvias internetines programas.

CSS (Cascading Style Sheets) – pakopinių stilių aprašymo kalba, skirta HTML kalba rašomų dokumentų stiliams aprašyti.

DOM (Document Object Model) – dokumento objektinis modelis – nuo platformos ir kalbos nepriklausantis standartas HTML ir XML dokumentų atvaizdavimui.

HTML (HyperText Markup Language) – Hipertekstų ženklinimo kalba, dažniausiai naudojama tinklalapiams rašyti.

JSON (JavaScript Object Notation) – JavaScript objektinis žymėjimas.

RIA (Rich Internet Application) – internetinės programos kurios savo funkcionalumu ir savybėmis primena tradicines programas.

XML (Extensible Markup Language) - universali dokumentų ženklinimo kalba, skirta dokumentų struktūrai aprašyti.

XSS (Cross-Site Scripting) – kompiuterių saugumo problemos tipas, kuris atsiranda puslapiuose, leidžiančiuose įterpti kodo elementą iš kito puslapio.