Jelgavas novads
Elejas vidusskola

JavaScript izmantošana mājas lapu veidošanā

zinātniski pētnieciskais darbs informātikā

Darbu izstrādāja: 11. klases skolnieks *Arvis Lācis*
Darba vadītājs: informātikas skolotājs *Jānis Tumovs*

2011
Eleja

ANOTĀCIJA

Pētījuma **problēmas būtība** ir sabiedrības neinformētība par JavaScript programmēšanas valodu, tās iespējamo pielietojumu interaktīvu mājas lapu veidošanā. Pētījuma **mērķis** bija iepazīt un izpētīt JavaScript valodas uzbūvi, tās nozīmi mājas lapu veidošanā, izveidot pilnvērtīgus skriptus skolas mājas lapai, kā arī izveidot īsu un pārskatāmu darbu, kas spētu teorētiskā līmenī informēt visus iespējamos interesentus par JavaScript iespējām un nozīmi. Zinātniski pētnieciskā darba galvenie uzdevumi bija atrast darbam nepieciešamo informāciju par JavaScript; no iegūtajiem literatūras, informācijas materiāliem izveidot savstarpēji vienotu, kritērijiem atbilstošu darbu; veikt darba praktisko daļu – izveidot skriptus skolas mājas lapai. Pētījuma galvenie rezultāti iegūti darba gaitā, gan no analizētajiem un aprakstītajiem literatūras avotiem, gan darba autora pētnieciskās darbības laikā. Pētnieciskā darba veikšanai tika izmantota analītiskā, sintēzes un salīdzinošā pētījuma metode. Pētījuma **galvenie rezultāti** ir izveidotais zinātniski pētnieciskais darbs, kas sevī ietver JavaScript pamatsintakses izpēti un aprakstu, kā arī izveidotie JavaScript skripti, kurus var izmantot gan interaktīvu, gan informatīvu mājas lapu elementu veidošanā. **Būtiskākais secinājums** ir, ka JavaScript programmēšanas valodu var viegli pielietot mājas lapu veidošanā, lai tās padarītu vairāk interaktīvas, un ar tās palīdzību var ievērojami samazināt mājas lapas datu, elementu atjaunināšanas laiku.

Atslēgas vārdi: JavaScript, programmēšana, skripti, mājas lapas

ANNOTATION

The meaning of the research is the society's less knowledge about JavaScript programming language, it's using to the interactive web page creation process. **The aim of the research** was to learn and discover the structure of JavaScript, it meaning in web page creation process, create fully working scripts for school's homepage, and also create small and understandable project work, which can be used to inform everybody about the JavaScript language and its main features in an average level. The main tasks of the research was to find needed information about the JavaScript; from collected materials, create connected, well formed work; do practical part – create scripts for the school's web page. The main results were got in research process – from analyzed and described literature and also from authors scientifically activity. In the research process was used analyzing, syntheses and comparing methods. **Main results** about the research is the created scientifically work, which includes research and description of the JavaScript's syntax basics, and also created JavaScript scripts can be used in interactive and informative web page's element creation. **The most important conclusion** is, that JavaScript programming language can be easily used to create web pages, make its more interactive, and with JavaScript programming language it's possible to update web page's data and elements faster and easier.

Keywords: JavaScript, programming, scripts, homepages

SATURS

IEVADS

„Senos laikos kā skaitļošanas līdzekli izmantoja akmentiņus vai nūjiņas. Viduslaikos cilvēki iemācījās rēķināt pierakstot skaitļus stabiņā. Taču tikai pagājušajā gadsimtā tika izgudrots elektroniskais kabatas skaitļotājs (turpmāk – EKS) – ierīce, kuras daudzpusīgās izmantošanas iespējas nav pilnībā apjaustas vēl pat mūsdienās." [23.,3. lpp.][1]

Tomēr ar to viss nebeidzās, EKS tika uzlaboti un pielāgoti cilvēku vajadzībām, līdz beidzot radās pirmie personālie datori, kas spēja izpildīt sarežģītas un apjomīgas programmas, kas bija uzrakstītas kādā no pirmajām programmēšanas valodām – FORTRAN[2] vai COBOL[3]. Līdz ar personālā datora tālāku attīstību, radās arī daudzas jaunas programmēšanas valodas (skatīt 4. tabulu pielikumā). Ap 1962. gadu radās pirmie interneta pamati, kas sākotnēji tika izmantoti tikai militārajām vajadzībām. [2., 3.-4.][10.]

Daudz vēlāk, tikai ap 1990. gadu internets kļuva pieejams plašākai sabiedrībai, iespējams, tikai tad sākās interneta „uzvaras gājiens", radās jaunas, īpaši interneta vajadzībām pielāgotas programmēšanas valodas – HTML, JavaScript, Java, XML, u.c.

Mūsdienās internets tiek ļoti plaši izmantots galvenokārt tā apjomīgo iespēju un pieejamo funkciju dēļ. Internets sniedz visdažādāko informāciju, ļauj izklaidēties, kā arī sazināties ar citiem tā lietotājiem, kā arī ļauj veikt ļoti svarīgus uzdevumus, piemēram, nomaksāt rēķinus, nosūtīt dokumentus. Tomēr tikai retais no interneta lietotājiem spēj pateikt to, kas ir gandrīz ikvienas mājas lapas pamatā – kur un kā ar JavaScript palīdzību ieviesti interaktīvie elementi. Tieši tādēļ, šī zinātniski pētnieciskā darba autors vēlas sniegt daudz plašāku informāciju par JavaScript programmēšanas valodu, tās vēsturi, uzbūvi (sintaksi), un to, kā ar tās palīdzību var uzlabot, padarīt interaktīvu ikvienu mājas lapu. Darba autors šo tematu uzskata par aktuālu un nozīmīgu, jo, pirmkārt, gan jaunieši, gan vecāka gada gājuma cilvēki mūsdienās ļoti daudz izmanto internetu un tā piedāvātās iespējas, otrkārt, latviešu valodā ir neliels informācijas daudzums par JavaScript valodu, iespējams, tieši tādēļ ir tik mazs cilvēku skaits, kas par šo valodu zina, interesējas un arī izmanto.

Savā darbā autors centīsies uzskatāmi parādīt, kā arī izveidot daudzus, pēc iespējas dažādā-

---

1 Šajā un katrā nākamajā norādē pirmā atzīme atspoguļo literatūras avota vietu kopējā sarakstā, bet otrā - lappusi konkrētajā literatūrā.

2 FORTRAN (FORmula TRANslation) – programmēšanas valoda, kas izveidota 1957. gadā un lielākoties tika izmantota zinātniskiem un militāriem mērķiem. [3.,6. lpp.]

3 COBOL (COmmon Bussiness-Oriented Language) - 1959. gadā izveidota programmēšanas valoda, kas tika izmantota biznesa vajadzībām, tiek lietota arī mūsdienās. [3.,6. lpp.]

kus, JavaScript izmantošanas piemērus un paraugus. Pētnieciskajā darbā autors aplūkos un paskaidros dažas nepareizi izveidojušās asociācijas, kas saistītas ar JavaScript nosaukuma nepareizu izprašanu, kā arī ar JavaScript drošumu un iespējamo risku lietotāja datoram.

Darba mērķis: Iepazīt un izpētīt JavaScript uzbūvi, tā nozīmi mājas lapu veidošanā, izveidot pilnvērtīgus skriptus skolas mājas lapai, kā arī izveidot īsu un pārskatāmu darbu, kas spētu teorētiskā līmenī informēt visus iespējamos interesentus par JavaScript iespējām un nozīmi.

Darba uzdevumi:

1. Atrast darbam nepieciešamo informāciju par JavaScript.
2. Analizēt, sagrupēt, konspektēt un apkopot iegūtos materiālus.
3. Veikt darba praktisko daļu – izveidot skriptus skolas mājas lapai.
4. Radīt savstarpēji vienotu un kritērijiem atbilstošu darbu.
5. Novērtēt JavaScript lietderīgumu un pielietojumu.

Darba hipotēze: JavaScript elementu izmantošana padara mājas lapu vairāk interaktīvu un samazina tās atjaunināšanas laiku.

Darbā izmantotās metodes: Darbā autors galvenokārt izmanto analītisko (sadalīšanas) un sintēzes (savienošanas) pētīšanas metodi, taču dažviet autors pievēršas arī salīdzinošajai pētījuma metodei.

Darba teorētiskā bāze: Darbā kopumā izmantoti 27% (7) rakstiskie, literārie avoti un 73% (19) elektronisko resursu avoti. Darbā ir 4 galvenās nodaļas. Autors analizē gan šajā, gan iepriekšējā gadsimtā uzrakstītos literāros darbus un avotus.

Darba struktūra: Darbs sastāv no ievada, 4 nodaļām (3 nodaļām, kas ietilpst darba teorētiskajā daļā un 1 praktiskās daļas nodaļas), secinājumiem, izmantotās literatūras saraksta, anotācijām un pielikuma.

1. INFORMĀCIJA PAR JAVASCRIPT

Tāpat kā ikvienai cilvēku valodai, arī ikvienai programmēšanas valodai ir sava izcelšanās, vēsture un attīstība. Tieši tādēļ, šī darba autors sekojošajās apakšnodaļās centīsies izklāstīt īsu JavaScript valodas vēsturi, sniegs informāciju par JavaScript izmantošanas iespējām; par to, kā iespējams pievienot JavaScript mājas lapai, kā arī aplūkos dažus, ar JavaScript valodu saistītus, nepatiesi izveidojušos, uzskatus.

1.1. Vēsture

JavaScript pirmsākumi meklējami nesenā pagātnē, 20. gadsimta deviņdesmitajos gados, kad pirmos JavaScript pamatus izstrādāja Brendons Eičs (Brendan Eich). Sākotnēji JavaScript tika izstrādāts un paredzēts tikai Netscape pārlūkprogrammām[4], turklāt JavaScript, tā pirmsākumos, tika dēvēts par Mocha. Vēlāk šis nosaukums vairākkārtīgi tika mainīts, piemēram, par LiveScript, līdz beidzot izveidojās patstāvīgs valodas apzīmējums jeb tās nosaukums – JavaScript. Jaunā programmēšanas valoda ieguva šādu nosaukumu, pateicoties Netscape Navigator pārlūkprogrammai, kas pirmoreiz vēsturē sāka plaši izmantot jaunās Java tehnoloģijas un to piedāvātās iespējas, kas tika sniegtas ar kompānijas Sun (Microsystems) noslēgtā līguma ietvaros.

1995. gada decembrī JavaScript bija izstrādāts pietiekami augstā līmenī, lai to varētu iekļaut Netscape pārlūkprogrammas 2.0B3 versijā. JavaScript tika izmantots pirmajās interneta mājas lapās un drīz vien guva ļoti lielu interneta lietotāju un programmētāju atsaucību. JavaScript biežā nosaukumu maiņa daudzos cilvēkos radīja nepareizu uzskatu, ka JavaScript apzīmē tieši tādu pašu jēdzienu kā pavisam cita programmēšanas valoda – Java (skatīt 1.5. apakšnodaļu, lai aplūkotu būtiskākās šo valodu atšķirības). Pētnieciskā darba autoram interesanti šķiet tas, ka šīs nelielās, vairākkārtīgās pārsaukšanas rezultātā, vēl pat mūsdienās ir ļoti daudzi cilvēki, kuri Java un JavaScript uzskata par identiskām valodām.

Laika gaitā JavaScript tika vairākas reizes pārdots un izstrādāts tādās ievērojamās kompānijās, kā Netscape, Sun Microsystems, Mozilla Foundation. Laikā kad notika formalitāšu kārtošana par JavaScript īsto nosaukumu un tā īpašnieku, 1996. gada augustā – kompānija Microsoft izveidoja jaunu JavaScript valodas dialektu, ko nosauca par JScript, lai izvairītos no autortiesību pārkāpšanas. Jaunizveidoto JScript Microsoft ietvēra savā pārlūkprogrammā – Internet Explorer 3.0. Nedaudz vēlāk, ap 1998. gadu, Netscape organizācijai Ecma International iesniedza standartizētu JavaScript versiju, ko nosauca par ESMAScript. Neskatoties uz to, kompānija Mozilla

4 Pārlūkprogramma – datorprogramma, kuras galvenā funkcija ir pārlūkot interneta mājas lapas, kā arī veikt datu apmaiņu starp interneta lietotāju un interneta pakalpojuma sniedzēju (mājas lapu).

Foundation turpināja JavaScript izstrādi.

Lai gan JavaScript mūsdienās tiek plaši izmantots mājas lapu veidošanā, taču tā izstrādāšana vēl turpinās, pēdējā JavaScript versija (1.8.1) tika izstrādāta un prezentēta 2009. gadā. [12.] [13.]

1.2. JavaScript iespējas un priekšrocības

Kas īsti ir JavaScript programmēšanas valoda, ko ar to var paveikt, un kāds no tās ir labums? JavaScript ir programmēšanas valodu, ko dažreiz, tās vienkāršības dēļ, sauc arī par skriptēšanas (*scripting*) valodu. JavaScript valodas galvenās funkcijas ir padarīt mājas lapas interaktīvas, interesantākas un „gudrākas". Ar JavaScript palīdzību ir iespējams gan veikt sarežģītus aprēķinus, gan vienkārši un ātri iegūt informāciju no lietotāja[5] datora (datumu, laiku, pārlūkprogrammas informāciju utt.), apstrādāt lietotāja ievadītos datus, kā arī veikt dažādas darbības, atkarībā no lietotāja rīcības mājas lapā, piemēram, peles klikšķa, taustiņa nospiešanas, elementa ielādes utt. Lai gan JavaScript iespējas ir ierobežotas, jo JavaScript nevar rediģēt informāciju, kas atrodas lietotāja datorā; tomēr ar šo programmēšanas valodu ir iespējams veikt pietiekami daudz, dažādu un sarežģītu operāciju (sīkāk par JavaScript valodas sintakses elementiem un to funkcijām – lasīt 2. nodaļu).

Darba autors par JavaScript valodas galvenajām priekšrocībām izvirza sekojošās:

- **Vienkāršība un saprotamība.** JavaScript valoda ir pietiekami viegla, lai to spētu saprast un izmantot jebkurš mājas lapu veidotājs, kurš labi pārzina HTML[6] un CSS[7] valodu elementus un īpatnības. Tātad, var secināt, JavaScript valodu īsā laikā var apgūt arī cilvēki, kuri nekad iepriekš nav programmējuši.
- **Savienojamība.** JavaScript savā darbībā ātri un vienkārši var izmantot HTML un CSS elementus, bet mājas lapas HTML kods var piekļūt JavaScript kodam; no tā seko, ka šīs abas valodas (JavaScript un HTML) savstarpēji ļoti labi mijiedarbojas.
- **Informācijas iegūšana.** Izmantojot JavaScript, var iegūt informāciju no mājas lapas apmeklētāja, piemēram, pārlūkprogrammas versiju, ekrāna izšķirtspējas lielumu, tādējādi lapas veidotājs ar JavaScript palīdzību var pielāgot savu lapu dažādu lietotāju datoriem, izvairoties no dažādām mājas lapas attēlošanas kļūdām.

5 Šajā projekta darbā ar vārdu „lietotājs" jāsaprot persona, kura izmanto internetu un datoru.

6 HTML – (HyperText Markup Language) hiperteksta iezīmēšanas valoda, kas tiek izmantota, lai mājas lapā attēlotu parastos elementus (tekstu, attēlus, tabulas, u.c.)

7 CSS – (Cascading Style Sheet) īpaši uzrakstīts kods, ar kura palīdzību iespējams daudz ātrāk un vienotāk noformēt mājas lapu galvenokārt tiek izmantots, ja nepieciešams vienlaicīgi noformēt ļoti daudzas mājas lapas sadaļas jeb apakšlapas.

Piemērs:

Lapas apmeklētāja datora ekrānam ir maza izšķirtspēja, JavaScript to nosaka, un pēc lapas veidotāja ievadītajām komandām – pielāgo lapas izmērus atbilstoši ekrāna izšķirtspējai, vai arī novirza lietotāju uz viņa datora ekrānam atbilstošo lapu; sniedz paziņojumu, kas lūdz iestatīt lielāku izšķirtspēju.

Informācijas iegūšanai un apstrādei JavaScript var izmantot arī sīkdatnes[8]. [12.]

1.3. Savienošana ar mājas lapu

JavaScript savienošana ar mājas lapu (HTML kodu) nav sarežģīta, kā jau autors to secina iepriekš, un neprasa lielu piepūli, programmēšanas iemaņas, jo līdzīgi ar mājas lapas kodu tiek savienots arī CSS un citu programmēšanas valodu kodi. Jāatceras, ka viens no galvenajiem HTML un JavaScript sasaistīšanas elementiem ir birkas[9] – `<script type="text/javascript">` un `</script>`.

Mājas lapas sasaistīšanā ar JavaScript, var izmantot sekojošos savienošanas paņēmienus:

1. **JavaScript ievietošana galvenē (starp birkām `<head>` un `</head>`).** Šāds savienošanas paņēmiens ļauj ielādēt JavaScript kodu pirms visiem mājas lapas elementiem. Šāda īpašība ir noderīga, ja mājas lapas apmeklētājam ir lēns interneta savienojums, pēc lapas ielādes, apmeklētājs var veikt visas darbības, negaidot, kamēr ielādēsies kāda JavaScript daļa. Tomēr šādai metodei ir arī nozīmīgs trūkums – mājas lapas ielāde, brīdis līdz pirmo elementu parādīšanās laikam, var būt ļoti ilgs, jo īpaši tad, ja JavaScript kods ir apjomīgs un prasa ilgu ielādes laiku.

Paraugs:

```
<html><head>
<script type="text/javascript">
// JavaScript kods
</script>
</head><body></body></html>
```

2. **JavaScript ievietošana HTML koda ķermenī (starp birkām `<body>` un `</body>`).** Šis paņēmiens ļauj JavaScript ielādēt vienlaicīgi ar visiem mājas lapas elementiem, tādējādi tiek samazināts laiks līdz pirmo elementu parādīšanās brīdim, tomēr šis paņēmiens nav izdevīgs, ja JavaScript un visa mājas lapa ir apjomīga, tad JavaScript kods var netikt ielādēts pilnībā, kā rezultātā var rasties kļūdas mājas lapas attēlojumā un skripta ielāde var

8 Sīkdatne – (cookie) – ir neliela JavaScript izveidota datne, kas spēj saglabāt nelielu teksta veida informāciju, piemēram, lietotāja vārdu, datumu, laiku, u.c.

9 Birka (tag) – HTML valodas elements, kas sniedz komandas, norādes pārlūkprogrammai, piemēram, ielikt atstarpi vai ievietot attēlu, mainīt fona krāsu utt. Birkās var ievietot atribūtus, bet atribūtos vērtības, kas attiecīgi izmaina katras birkas darbības rezultātu.

prasīt daudz ilgāku laiku. [24., 38.-39.][25., 60.-61.]

Paraugs:

```
<html><head></head>
<body>
<script type="text/javascript">
// JavaScript kods
</script>
</body></html>
```

3. **JavaScript izmantošana no attālinātās datnes.** Pēc autora domām, šis ir viens no retāk izmantotajiem JavaScript un HTML sasaistīšanas paņēmieniem galvenokārt tādēļ, ka mājas lapas un JavaScript ielāde šādā gadījumā prasa daudz laika – pārlūkprogrammai jāatrod attālinātās datnes, tās jānolasa un jāapstrādā, un tikai tad jāveic norādīto darbību izpilde. Metode ir izdevīga tikai tad, ja viens un tas pats JavaScript kods tiek izmantots vairākās mājas lapas apakšlapās (sadaļās). Jāpiebilst, ka šādi pievienots JavaScript kods ir daudz vairāk aizsargāts un to ir grūtāk nokopēt. Atšķirībā no abiem iepriekšējiem paņēmieniem, šajā – <script> birkā papildus jānorāda atribūts „*src*", kas satur ceļu līdz attālinātajai datnei.

Paraugs:

```
<html><head>
<script type="text/javascript" src="datne.js"></script>
</head><body></body></html>
```

1.4. JavaScript drošība un saderība ar pārlūkprogrammām

Tāpat kā par ikvienu cilvēku kādreiz ir izveidojušās dažādas baumas, tāpat arī veidojas baumas par cilvēka izgudrojumiem un radītajām lietām, protams, arī par JavaScript programmēšanas valodu. Galvenokārt lielākie strīdi un diskusijas interneta lietotāju vidū ir par to, ka JavaScript ir ļoti nedroša valoda, tā satur vīrusus, spēj ļaunprātīgi piekļūt lietotāja datoram, datnēm, programmām utt. Pēc autora domām, šādi uzskati, iespējams, radušies sekojošo iemeslu dēļ:

1. JavaScript kods tiek lejupielādēts un izpildīts vienlaikus ar HTML kodu, un lietotājs bez īpašām zināšanām un drošības iestatījumiem nevar atcelt JavaScript izpildi, kā rezultātā, lietotājus var pārņemt bailes, ka viņu datorā, bez atļaujas, tiek lejupielādēta un izpildīta jebkāda datorprogramma, šajā gadījumā, skripts. [15.];
2. Autora iepriekš minētās Java un JavaScript valodu neatšķiršanas dēļ – tā kā Java valodā ir iespējams uzrakstīt lietotāja datoram kaitīgas programmas, tad, iespējams, ka kļūdaini šāda paša īpašība tika un tiek piedēvēta arī JavaScript (par Java un JavaScript valodu at-

šķirībām sīkāk lasīt 1.5. apakšnodaļā).

Tomēr pētnieciskā darba autoram jāatzīst, ka iepriekš nosauktie iemesli nav pamatoti un JavaScript nevar veikt ļaunprātīgas darbības ar lietotāja datoru un datiem – nevar nolasīt vai ierakstīt datnes datorā, nevar palaist vai vadīt citas datorprogrammas, nevar izveidot savienojumu ar citiem datoriem. JavaScript var tikai lejupielādēt citu HTML vai JavaScript datni, nosūtīt e-pastus, vai arī iegūt laika un datuma informāciju no lietotāja datora, tādēļ darba autors secina, ka JavaScript valodā nav iespējams uzrakstīt bīstamus datorvīrusus, kas spētu kaitēt lietotāja datoram.

Jāatzīst, ka Microsoft Internet Explorer (turpmāk tekstā – IE) pārlūkprogramma ļauj JavaScript kodam izmantot ActiveX[10] iespējas, kā rezultātā, JavaScript var arī piekļūt un modificēt lietotāja datora datnes, tādēļ IE ir viena no nedrošākajām interneta pārlūkprogrammām.

Līdzīgi kā IE ir pārlūkprogramma, kas ļauj JavaScript izmantot ActiveX, tāpat pastāv arī dažas citas JavaScript elementu atšķirības dažādās pārlūkprogrammās, kas jāņem vērā izstrādājot mājas lapu, lai tā darbotos visos datoros vienādi (lielākoties šīs atšķirības saistītas ar mājas lapas noformēšanas skriptiem).[5.]

Piemēram, lai nomainītu mājas lapas fona krāsu pret melnu, dažādām pārlūkprogrammām jāizmanto dažādi kodi:

- IE – `document.all.fons.style.backgroundColor = "#000000"`;
- Netscape – `document.fons.backgroundColor = "#000000"`;
- Opera un Mozilla – `document.getElementById("fons").style.backgroundColor = "#000000"`.

Protams, ir arī citas JavaScript elementu un sintakses atšķirības dažādās pārlūkprogrammās, taču, par laimi mājas lapu veidotājiem, ir iespējams izveidot JavaScript kodus, kas spēj noteikt lietotāja pārlūkprogrammu un spēj pielāgot izvades rezultātu tieši tai, tādējādi lietotājiem nenākas sūdzēties par to, ka ar vienu pārlūkprogrammu lapu var apskatīt ideāli, bet ar otru viss izskatās nepatīkami un neglīti, kā arī novērojamas citas kļūdas skriptu izpildē.

1.5. Vai JavaScript ir tas pats, kas Java?

Kā jau autors min iepriekš, nosaukuma vairākkārtējas pārdēvēšanas dēļ (Mocha, LiveScript un JavaScript), daudzi cilvēki sāka uzskatīt, un vēl joprojām uzskata, ka JavaScript un Java ir pilnīgi identiskas valodas, kurām nav būtisku atšķirību. Patiesībā ir pavisam otrādi – šīs valodas ir ļoti atšķirīgas, un tām ir tikai dažas kopīgas īpašības. Lielākās un nozīmīgākās JavaScript un Java programmēšanas valodu atšķirīgās un kopīgās īpašības ir apskatāmas 1. tabulā. Kā redzams,

10 ActiveX – Microsoft izveidota programmu sastāvdaļa, kas atvieglo un paātrina citu programmu, programmu komponentu un skriptu izpildes ātrumu un iespējas. [15.]

tad JavaScript ir daudz vienkāršāka un ikvienam pieejamāka programmēšanas valoda nekā Java. [12.][15.]

1. tabula

JavaScript un Java valodu salīdzinājums [12.][15.]

Kopīgās īpašības	Atšķirīgās īpašības
• Ir veidotas no C# programmēšanas valodas sintakses pamatiem. • Daudzi kopīgie elementi, īpaši tie, kas saistīti ar matemātisko un datuma operāciju, funkciju izmantošanu. • Tiek plaši izmantotas mājas lapās.	• JavaScript valodu izveidoja un izstrādāja Netscape programmētājs Brendons Eičs (Brendan Eich), bet Java valodu – Sun (Microsystems). • JavaScript valoda ir dinamiska, t.i., mainīgais var saturēt jebkāda veida elementu, turpretim Java valodā katrs mainīgais ir jādefinē un ir strikti noteikts. • JavaScript valodā rakstīto cilvēka tekstu dators var nolasīt un izpildīt uzreiz, taču Java programmā uzrakstīto tekstu vispirms ir jākompilē[11]. • JavaScript ir daudz vienkāršāka valoda nekā Java, kas savā sarežģītībā ir pielīdzināma C# un C++ valodām. • JavaScript programmēšanas valoda ir atvērtā pirmkoda[12] valoda, bet Java valodas pirmkods var būt arī slēgts, ja tā vēlas programmas autors.

1.6. Daži JavaScript piemēri mājas lapās

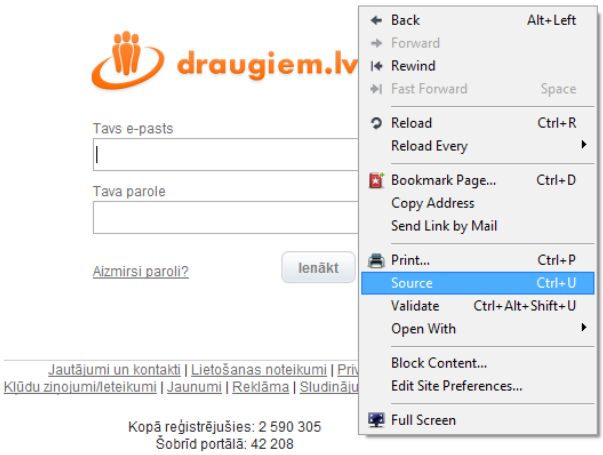
Kā jau autors raksta darba iepriekšējās nodaļās, tad JavaScript kods ir atrodams ikvienā interaktīvā mājas lapā. Jebkuras mājas lapas HTML un JavaScript koda apskatīšana ir ļoti vienkārša – pārlūkprogrammā jāatver nepieciešamā, analizējamā mājas lapa, tad jāveic peles labās pogas klikšķis uz brīvu vietu mājas lapā, t.i., kur nav attēlu, saišu, u.c. interaktīvu elementu, tad jāizvēlas iespēja *Source* vai arī *View Source,* skatīt 1.attēlu) (tas atkarīgs no izmantojamās pārlūkprogrammas), rezultātā HTML, CSS un JavaScript kodi būs redzami.

Kā pirmo piemēru, darba autors aplūko Latvijā vispopulārāko mājas lapu Draugiem.lv (tikai mājas lapas sākumlapu). Redzams, ka pārlūkprogramma attēlo tikai Draugiem.lv logotipu, epasta un paroles ievades laukus, un dažas saites, tātad parastam interneta lietotājam var šķist, ka šajā lapā nav nekā sarežģīta, un viss ir ļoti labi saprotams. Taču, veicot iepriekš aprakstīto koda apskatīšanas paņēmienu, var ieraudzīt, ka patiesībā draugiem.lv sākumlapas kods ir 112 rindiņas

11 Kompilēt – pārveidot mnemonisko kodu (cilvēka rakstīto virkņu kodu) mašīnkodā (datoram saprotamā kodā). [3.,5. lpp.]

12 Pirmkods – programmas sākotnējais, cilvēka rakstītais kods, kas vēlāk tiek kompilēts vai arī tiek tieši izmantots (pārlūkprogrammas tieši nolasa JavaScript kodu).

garš un satur 6042 simbolus. Tikai nelielu daļu, aptuveni 30%, no šī koda veido parastais HTML un CSS kods, bet atlikušo daļu (70%) veido JavaScript (skatīt pielikuma 5. attēlu). Analizējot draugiem.lv kodu, autors novēro, ka JavaScript šeit izmantots, lai atpazītu lietotāja autorizāciju, kā arī, lai paziņotu par pogas „Caps Lock" nospiešanu. Atlikusī skripta daļa tiek izmantota, lai veiktu saziņu, datu apmaiņu starp Draugiem.lv serveri un lietotāja datoru.



1. attēls
Mājas lapas koda apskatīšana pārlūkprogrammā Opera [7.]

Nedaudz atšķirīgāka pieeja JavaScript izmantošanā ir lielākajam Latvijas ziņu portālam – Delfi.lv. Salīdzinot ar Draugiem.lv, Delfi.lv JavaScript kods aizņemt tikai ļoti nelielu daļu no sākumlapas visa koda, pārējo daļu sastāda HTML kods. Delfi.lv JavaScript izmanto, lai veiktu nelielu lapas pielāgošanu lietotāja datoram, kā arī lai saņemtu dažāda veida informāciju no citiem ziņu serveriem. Interesanti, ka Delfi.lv savā mājas lapā izmanto gan JavaScript, gan VBscript programmēšanas valodas. [6.]

Visbeidzot, autors analizē arī Elejas vidusskolas mājas lapas HTML un JavaScript kodu, jo arī sava pētnieciskā darba praktiskajā daļā autors veidos skriptus tieši šai mājas lapai. Salīdzinājumā ar iepriekš minētajām mājas lapām, šajā lapā JavaScript tiek izmantots tikai interaktīvu elementu parādīšanai, kas arī sekmē tik ātru mājas lapas ielādi un tās elementu attēlošanu. Mājas lapā ievietots JavaScript kods, kas parāda attiecīgās dienas vārda dienas gaviļniekus Elejas vidusskolā, kā arī sadaļā *Pasākumi* ievietots ļoti īss, bet efektīvs skripts, kas ļauj lietotājam ātri un interesanti apskatīt dažus fotoattēlus (skatīt 2. attēlu).

2. attēls
Elejas vidusskolas mājas lapā ievietotais skripts, kas maina attēlus [6.]

2. IESKATS JAVASCRIPT SINTAKSĒ

Ikvienai cilvēka valodai ir savs alfabēts, rakstības principi un likumi, sintakse, interpunkcija. Šāda veida uzbūve ir arī jebkurai programmēšanas valodai – alfabēts (izmantojamie simboli), noteikti likumi, sintakse un interpunkcija. Atšķirībā no cilvēku valodas, programmēšanas valodā nedrīkst pieļaut nevienu kļūdu, pat ne vismazāko, jo pretējā gadījumā dators pilnībā vai daļēji var neizpildīt cilvēka doto uzdevumu.

Salīdzinot ar citām programmēšanas valodām, JavaScript valodai ir vienkārša sintakse un uzbūve, turklāt atkarībā no pārlūkprogrammas un JavaScript versijas, pārlūkprogrammas dialekta, ir atļauts arī nedaudz novirzīties no šīm sintakses normām, tomēr oficiāli JavaScript sintaksei ir stingri noteikti nosacījumi, pie kuriem ir vēlams pieturēties, lai patiešām – visas pārlūkprogrammas mājas lapu attēlotu vienādi.

JavaScript sintakse kopumā ir pietiekami plaša un sarežģīta, tādēļ šī darba autors tālākajās apakšnodaļās sniegs tikai virspusīgu ieskatu visā JavaScript sintaksē un uzbūvē, jo autora darba mērķis nav izpētīt visu JavaScript uzbūvi pilnībā, bet gan tikai sniegt ieskatu tajā, kā arī parādīt to, kā JavaScript sintakses elementus var pielietot mājas lapu veidošanā, interaktīvu elementu izveidē.

2.1. Alfabēts un pieturzīmes

JavaScript alfabēts[13] sastāv no daudzām dažādām rakstzīmēm – latīņu alfabēta burtiem, cipariem, kā arī no speciālajām rakstu zīmēm vai apzīmējumiem (papildu informāciju skatīt 2. tabulā). Faktiski, JavaScript savā darbībā var izmantot arī latviešu un citu valodu alfabēta burtus, tomēr šo burtu loma ir visai ierobežota – tie drīkst atrasties tikai mainīgo un simbolu virkņu vērtībās, bet tos nedrīkst izmantot, lai definētu (nosauktu) mainīgos.

Atšķirībā no mūsu, latviešu un citām cilvēku valodām, kurām lielākoties katrs teikums beidzas ar punktu (.), JavaScript viens „teikums" beidzas ar semikolu (;). JavaScript „teikumi" var būt gan ļoti īsi, gan gari, galvenokārt katrs no tiem tiek rakstīts atsevišķā koda rindiņā, taču pēc izstrādātāja gaumes var būt izkārtoti arī savādāk. Līdzīgie mainīgie un virkņu elementi, tāpat kā vienlīdzīgie teikuma locekļi, tiek atdalīti ar komatu (,). Figūriekavas ({}) tiek izmantotas, lai atdalītu JavaScript funkcijas un ciklus, bet ar kvadrātiekavām ([]) tiek atdalīti un norādītu atsevišķi masīvu elementi.

2. tabula

13 Ar vārdu „alfabēts" jāsaprot visas pieļaujamās rakstzīmes, ko JavaScript var brīvi izmantot mainīgo pierakstīšanai, nolasīšanai un citu darbību veikšanai.

JavaScript programmēšanas valodā atļautās rakstzīmes

Apraksts	Piemēri
26 latīņu alfabēta burti	a, b, c…x, y, z; A, B, C…X, Y, Z
10 arābu cipari	0, 1, 2, 3, 4, 5, 6, 7, 8, 9
Pasvītrojuma zīme	_
Matemātisko operāciju zīmes	+ - * / ^
Salīdzināšanas zīmes	< > <> = >= <= == !=
Dažāda veida iekavas	() [] {} <>
Numura zīme	#
Apostrofs un pēdiņas	' ""
Atdalītājzīmes	, ; .
Komentāru zīmes	// /**/

Pēc darba autora domām, nozīmīgas JavaScript pieturzīmes ir arī komentāru zīmes – // un /**/, kas tiek izmantotas, lai pierakstītu dažādus komentārus un skriptus (tas ir lietderīgi, ja JavaScript ir ļoti apjomīgs, vai arī tad, ja skriptu veido vairāki cilvēki). Šādi uzrakstītus komentārus pārlūkprogramma nenolasa un neizpilda. Zīmi // izmanto, ja komentārs ir tikai vienas vai dažu rindiņu garumā, savukārt, /**/ izmanto garāku un apjomīgāku komentāru rakstīšanai.

Komentāru piemēri:

```
// Vienkāršs JavaScript komentārs
```

un

```
/* JavaScript komentārs,
kas ir garāks nekā viena rindiņa.
*/
```

Bez iepriekš nosauktajām pieturzīmēm, JavaScript valodā tiek lietotas arī sekojošas pieturzīmes: atstarpe ( ), apostrofs ('), pēdiņas ("") un punkts (.).

2.2. Mainīgie

JavaScript mainīgais, gluži tāpat kā matemātikā, ir simbolisks apzīmējums kādam mainīgam lielumam – skaitliskai, kā arī teksta veida informācijai. JavaScript valodā mainīgo apzīmē pēc sekojošas formulas: *var* + mainīgā nosaukums = mainīgā vērtība;.

Mainīgo piemēri:

```
var x = 5;                    // Mainīgais „x" ar vērtību 5.
var y = "vārds ";             // Mainīgais „y" ar vērtību „vārds".
```

Mainīgos var apzīmēt ne tikai ar vienu burtu, bet arī ar veselu vārdu vai simbolu virkni (piem., *dati* vai *s5g4m* u.c.), taču, piešķirot mainīgajiem vērtības, ir jāievēro šādi nosacījumi:

- mazie un lielie burti tiek uzskatīti par atšķirīgiem, tas nozīmē, ka, piemēram, mainīgie *x* un *X*, vai arī *dati* un *DaTi* nav vienādi un tie var saturēt pavisam dažādas vērtības;
- mainīgā nosaukumam ir jāsākas ar burtu vai arī ar pasvītrojuma zīmi (_), bet ne ar ciparu.

Visus JavaScript mainīgos var iedalīt divās grupās:

- lokālie mainīgie – mainīgie, kas tiek definēti pēc iepriekš norādītās formulas (izmantojot

vārdu „*var*"), šie mainīgie nevar tikt izmantoti starp dažādām JavaScript funkcijām;

- globālie mainīgie – mainīgie, kas tiek definēti bez vārda „*var*" to sākumā, šie mainīgie var tikt izmantoti starp dažādām JavaScript funkcijām.

Atšķirībā no citām programmēšanas valodām, definējot JavaScript mainīgo, nav obligāti jānorāda tā veids (skaitlis, teksts, datums utt.), turklāt JavaScript mainīgajos var pārdefinēt, t.i., piešķirt tiem pavisam citas vērtības.

Mainīgo pārdefinēšanas piemēri:

```
var dati = "skola";     // Mainīgajam „dati" tiek piešķirta vērtība „skola".
var dati = 4;           // Mainīgajam „dati" tiek piešķirta jauna
                        // vērtība – 4.
var dati = "E-pasts";   // Mainīgajam „dati" tiek piešķirta jauna
                        // vērtība „E-pasts".
```

2.3. Operatori

Operatori ir vienkāršas zīmes, zīmju kombinācijas, kas apzīmē kādu noteiktu matemātiskas vai arī salīdzināšanas darbību. Lielākā daļa biežāk izmantoto JavaScript operatoru un to funkcijas sniegtas 3. tabulā.

3. tabula

JavaScript operatori un to funkcijas (ja y=5 un z=10) [13.][14

Matemātiskais operators	Apraksts	Piemērs ar mainīgajiem	Aprēķināšanas veids	x vērtība
-	Atņemšana	x=y-2;	x=5-2	x=3
+	Saskaitīšana	x=y+4;	x=5+4	x=9
/	Dalīšana	x=y/5;	x=5/5	x=1
*	Reizināšana	x=y*0.5;	x=5*0.5	x=2.5
%	Atlikums	x=y%2;	x=5%2	x=1
++	Pieaugšana	x=++y;	x=5+1+1	x=7
--	Dilšana	x=--y;	x=5-1-1	x=3
Piešķiršanas operators	**Apraksts**	**Piemērs ar mainīgajiem**	**Aprēķināšanas veids**	**y vērtība**
=	Piešķir vērtību	y=y;	y=y=5	y=5
-=	Atņem pats sevi	y-=y;	y=y-y=5-5	y=10
+=	Pieskaita pats sevi	y+=y;	y=y+y=5+5	y=10
/=	Izdala ar sevi	y/=y;	y=y/y=5/5	y=1
*=	Reizina ar sevi	y*=y;	y=y*y=5*5	y=25
%=	Aprēķina atlikumu	y%=y;	y=y%y=5%5	y=0
Salīdzināšanas operators	**Apraksts**	**Piemērs ar mainīgajiem**	**Rezultāts**	
==	Ir vienāds ar	y==8;	Nav patiesība	
===	Ir pilnīgi vienāds (sakrīt vērtība un veids)	y===5; y===”5”;	Patiesība Nav patiesība	

!=	Nav vienāds	y!=8;	Patiesība
>	Ir lielāks nekā	y>8;	Nav patiesība
<	Ir mazāks nekā	y<8;	Patiesība
>=	Lielāks vai vienāds	y>=8;	Nav patiesība
<=	Mazāks vai vienāds	y<=8;	Patiesība
&&	Patiess abos gadījumos	(y < 10 && z > 1);	Patiesība
||	Viens no abiem	(y==6 || z==6);	Nav patiesība
!	Nav	!(y==z);	Patiesība

Kā redzams tabulā, tad operatorus, atkarībā pēc to funkcijām un pielietojuma, autors iedala trīs grupās: matemātiskajos, piešķiršanas un salīdzināšanas operatorus. Jāatzīst, ka pastāv arī citi operatoru iedalīšanas veidi.

Norādītos operatorus galvenokārt izmanto, lai veiktu dažādas darbības ar skaitliskiem mainīgajiem, taču tādus operatoru, kā: +, =, ==, ===, *!=*, &&, ||, *!* - var izmantot arī darbā ar simbolu virknēm (teksta mainīgajiem).

Piemēri ar operatoriem:

```
var x = 25;            // Mainīgajam „x" tiek piešķirta vērtība 25.
var y = 6;             // Mainīgajam „y" tiek piešķirta vērtība 6.
var a = x*y;           // Mainīgais „a" satur vērtību 150, jo 25*6=150.
var b = x%y;           /* Mainīgais „b" satur vērtību 1, jo dalot 25 ar 6,
                       atlikumā iegūs 1.*/
```

2.4. Paziņojuma logi

Paziņojuma logi ir īpaši JavaScript elementi, kas lietotājam parāda informāciju, datus un vērtības, liek attēloto informāciju apstiprināt vai norādīt vai arī liek lietotājam pašam ievadīt atbildi (vērtību).

Paziņojuma logi ir ļoti vienkārši un nespēj attēlot un pieprasīt lietotājam ievadīt daudzveidīgus un sarežģītus datus, taču tie noder gan, lai ātri iegūtu informāciju, gan lai efektīvi brīdinātu lietotāju. Kopumā izšķir 3 paziņojumu logu veidus:

1. Brīdinājuma paziņojumi – tiek izmantoti, lai parādītu vienkāršu, tekstu saturošu logu, brīdinātu vai informētu lietotāju.

Brīdinājuma paziņojuma piemērs:

```
alert("Laipni lūdzu manā mājas lapā!");        // Skatīt pielikuma 6. attēlu.
```

2. Apstiprinājuma logi – tiek izmantoti, lai dotu lietotājam iespēju apstiprināt vai noliegt informāciju. Apstiprinājuma logs satur informatīvu tekstu, kā arī piedāvā divas izvēles iespējas (skatīt pielikuma 7. attēlu) – *Ok* (labi, apstiprināt, piekrist), *Cancel* (atcelt, noraidīt, noliegt). Atkarībā no lietotāja izvēles, JavaScript tālākā darbība var atšķirties. Piemē-

ram, ja lietotājs nospiež pogu *Ok*, tad skripts izpilda vienu darbību virkni vai funkciju, bet ja pogu *Cancel*, tad – atšķirīgu darbību virkni vai funkciju.

Apstiprinājuma loga piemērs:

```
confirm("Vai tiešām vēlaties iziet?");          // Skatīt pielikuma 7. attēlu.
```

3. Ievades logi – tiek izmantoti, lai lietotājam pieprasītu ievadīt informāciju, kas tiek izmantota skripta tālākajā darbībā. Izskata ziņā ievades logi ir līdzīgi apstiprinājuma logiem, taču papildus ir redzams īpašs laukums, kas paredzēts lietotāja atbildes ievadei.

Ievades loga piemērs:

```
prompt("Ievadiet sava mājdzīvnieka vārdu:", "Reksis");
```

2.5. Matemātiskie objekti

Nav noslēpums, ka interaktīvas mājas lapas izveidošanai, nepieciešami arī daudzi, dažādi matemātiski aprēķini – gan vienkārši, gan sarežģīti, piemēram, lai pareizi attēlotu dažādu elementu kustību (leņķiskie aprēķini), lai varētu apstrādāt lietotāja ievadītos datus, veiktu aprēķinus, kas saistīti ar datumiem u.c. Lielākā daļa JavaScript izmantotās matemātikas konstantes, metodes un to apraksti attēloti pielikuma 5. tabulā.

Piemērs:

```
var ievade = prompt("Lūdzu, ievadiet skaitli, no kura vēlaties izvilkt kvadrātsakni ", 1);
var rezultats = Math.sqrt(ievade);
alert("Kvadrātsakne no "+ievade+" ir "+rezultats);
/* Šajā piemērā attēlots tas, kā lietotājam tiek lūgts ievadīt skaitli, tad JavaScript no šī skaitļa izvelk kvadrātsakni un rezultātu izvada lietotājam. */
```

2.6. Sazarojuma un cikliskās konstrukcijas

Ļoti bieži ir nepieciešams, lai JavaScript kods spēj veikt automatizētus lēmumus, kā rezultātā veic vienu vai citu, vai arī nevienu darbību, piemēram, lai spētu analizēt lietotāja ievadi. Šādam nolūkam ļoti labi noder JavaScript sazarojuma konstrukcijas, kas spēj analizēt visdažādākos procesus (mainīgo vērtības, mainīgo garumus, u.c.), turklāt sazarojuma konstrukciju izmantošana ļoti ievērojami samazina JavaScript apjomu vai pat noder gadījumos, kad citādi šādu izpildes rezultātu ar JavaScript nevarētu panākt.

Sazarojuma konstrukcijas var salīdzināt ar dažāda veida matemātiskajiem nosacījumiem, kas lielākoties sākas ar vārdu „*ja*" (if). JavaScript sazarojuma konstrukcijas var iedalīt sekojošās grupās:

- **if (ja)** konstrukcija. JavaScript kods tiek izpildīts tikai tad, ja nosacījums ir patiess, pretē-

jā gadījumā „*if*" konstrukciju saturošā skripta daļa pilnībā tiek ignorēta un skripts turpina izpildīt atlikušās skripta rindiņas, bet ja skriptam vairs nav turpinājuma, tad skripts savu darbību beidz.

if konstrukcijas piemērs:

```
var datums = new Date();              // Mainīgajam „datums" tiek piešķirta
var laiks = datums.getHours();        // datora laika un datuma vērtība.
if (laiks < 10){                      // Pārbauda vai datora pulkstenis rāda
document.write("<b>Labrīt!</b>");}    // mazāk kā 10, ja tā ir, tad tiek
                                      // izvadīts teksts – „Labrīt!".
                                      // Ja pulkstenis rāda vairāk, tad
                                      // teksts netiek izvadīts.
```

- **if… else (ja… citādi)** konstrukcija. Šī konstrukcija ir ļoti līdzīga iepriekš norādītajai, taču šajā gadījumā – ja nosacījums nav patiess, tad tiek izpildīta cita JavaScript daļa.

if… else konstrukcijas piemērs:

```
var datums = new Date();              // Pārbauda vai datora pulkstenis
var laiks = datums.getHours();        // rāda mazāk kā 10, ja tā ir, tad tiek
if (laiks < 10){                      // izvadīts teksts – „Labrīt!".
document.write("<b>Labrīt!</b>");}    // Ja pulkstenis rāda vairāk nekā 10,
else{                                 // tad tiek izvadīts pavisam cits
document.write("<b>Labdien!</b>");}   // teksts – „Labdien!".
```

- **if… else if… else… (ja…, citādi ja…, citādi…)** konstrukcija. Kopumā šī konstrukcija ir tikai divreiz atkārtota iepriekšējā „*ja… citādi*" konstrukcija, taču tā tiek ļoti plaši izmantota mājas lapu veidošanās, jo tā ļauj atlasītu tikai vienu izpildāmo koda daļu no ļoti liela koda un daudziem mainīgajiem.

if… else if… else… piemērs:

```
var datums = new Date();              // Pārbauda vai datora pulkstenis
var laiks = datums.getHours();        // rāda mazāk kā 10.00, ja tā ir, tad
if (laiks < 10) {                     // tiek izvadīts teksts – „Labrīt!".
document.write("<b>Labrīt!</b>");}    // Ja pulkstenis rāda laiku,
else if (laiks>10 && laiks<16){       // kas ir starp 10.00 un 16.00,- tad
document.write("<b>Labdien!</b>");}   // tiek izvadīts teksts – „Labdien!".
else{                                 // Ja neviens no iepriekšējiem
document.write("<b>Labvakar!</b>");}  // nosacījumiem nav spēkā, tad tiek
                                      // izvadīts teksts – „Labvakar!".
```

- **switch (pārslēgšanās, pārlēkšanas)** konstrukcija. Šāda veida konstrukciju parasti izmanto, lai pārslēgtos starp ļoti dažādiem, savstarpēji nesaistītiem nosacījumiem. Parasti šīs konstrukcijas var izveidot arī ar iepriekš aprakstīto *ja…, citādi ja…, citādi…* konstrukciju, tomēr tādā gadījumā ir jāizmanto daudz vairāk „nevajadzīgu" koda daļu, kas padara

JavaScript kodu apjomīgāku un tā lejupielādēšanu un izpildes laiku ilgāku.

switch konstrukcijas piemērs:

```
var skaitlis = prompt("Ievadiet skaitli no 1 līdz 3:",1);
switch(skaitlis) {
case "1":                                   // Lietotāja tiek pieprasīts
document.write("Skaitlis viens");           // ievadīt skaitli no 1 līdz 3.*/
break;                                      // Atkarībā no lietotāja ievades,
case "2":                                   // JavaScript kods piemēro
document.write("Skaitlis divi");            // attiecīgo izvadi.
break;                                      // Ja lietotājs neievada kādu no
case "3":                                   // norādītajiem skaitļiem, tad
document.write("Skaitlis trīs");            // par to tiek paziņots.
break;
default:
document.write("Jūs neievadījāt skaitli no 1 līdz 3.");}
```

Bez sazarojuma konstrukcijām pastāv arī cikliskās konstrukcijas, kuru uzdevums tāpat kā sazarojuma konstrukcijām ir gan samazināt JavaScript koda apjomu, gan paātrināt skripta izpildes laiku un arī atvieglot skriptu veidotāju darbu, jo nav jāveido simtiem rindiņu garš kods, ja tā vietā var uzrakstīt desmit rindiņas garu.

Cikliskās konstrukcijas ir ļoti plaši izmantots elements ikvienā programmēšanas valodā, jo tās nodrošina automātisku, ļoti daudzu vienādu darbību, t.i., ciklu izpildi. Ciklus var izmantot, piemēram, lai no 1000 skaitļiem atrastu tikai vienu, kas der par kāda vienādojuma atrisinājumu, ļautu pārbaudīt vai visi datubāzes ieraksti ir derīgi un atbilst visām prasībām. Lai izveidotu ciklu, izmanto tādu JavaScript elementu kā *„for()”*. *„For”* elementa uzbūve, pieraksts ir šāds: *for(*mainīgais un tā sākuma vērtība; mainīgā beigu vērtība; solis).

Cikla piemērs:

```
for (i=1; i<=100; i++) {          // Šajā piemērā mainīgajam „i” sākuma
document.write(i+" ");}           // vērtība ir 1, beigu vērtība 100, bet
                                  // solis - +1. Šis skripts uz ekrāna izvada
                                  // visus naturālos skaitļus no 1 līdz 100
                                  // (skatīt 8. attēlu pielikumā).
```

Pētnieciskā darba autoram jāatzīst, ka bez šādas, parastās cikliskās konstrukcijas, pastāv arī dažas citas, piemēram, tādas, kas darbojas tik ilgi, kamēr piepildās kāds no nosacījumiem, piemēram, skaitlis x sasniedz vērtību 100, vai arī izlaiž kādu no mainīgā vērtībām (piem., dalīšanas procesos izlaiž nulli, jo ar to nedrīkst dalīt). Šādi cikli tiek izmantoti salīdzinoši reti, tādēļ tos autors savā darbā neaplūko.

2.7. Simbolu virknes, darbības ar virknēm un datu masīvi

Bieži vien, mājas lapu veidotājiem JavaScript daudz vairāk jāizmanto dažādu lietotāju ievažu un tekstu analīzei, nekā parastu matemātisku aprēķinu vai ciklu veikšanai, tādēļ šim nolūkam ir izmantojamas simbolu virknes jeb mainīgie, kas satur tekstu. Daži teksta mainīgo piemēri jau tika minēti 2.2 apakšnodaļā. Simbolu virknes no vienkāršajiem skaitliskajiem mainīgajiem lielākoties atšķiras ar to, ka tās satur nevis skaitļus, bet gan rakstzīmes, t.i., ciparus, burtus u.c., turklāt to vērtības ir jāliek pēdiņās ("").

Simbolu virkņu piemēri:

```
var vards = "dators";      // Virknes „vards" vērtība ir „dators".
var vards2 = "Pēteris"     // Virknes „vards2" vērtība ir „Pēteris", kā
                           // redzams, tad virkņu vērtība var saturēt arī
var vards3 = "5"           // ne-latīņu alfabēta burtus.
var vards4 = "27"          // Šajos gadījumos virkne satur skaitļus, kas
var pedejais_vards = "-7" // sastāv no ciparu rakstzīmēm. Atšķirībā no
                           // skaitliskajiem mainīgajiem, ar šīm virknēm nevar
                           // veikt matemātiskas darbības.
```

Simbolu virkņu definēšana un piešķiršana mainīgajiem vēl neļauj mājas lapas veidotājam iegūt vēlamo rezultātu, lai to izdarītu, jāizmanto daudzas, dažādas virkņu funkcijas, kas var veikt visdažādākās darbības ar virknēm (piemēram, atrast tajās vārdus, aizstāt vārdus vai simbolus, izdzēst lieko utt.). Lūk, pēc autora domām, divas vienkāršākās un praktiskākās virkņu funkcijas [13.][14.]:

- **test()** – šī funkcija lietotāja norādītajā virknē meklē norādīto vārdu vai simbolu, un atkarībā no rezultāta, sniedz atbildi – *true*, ja norādītais vārds vai simbols ir atrasts, bet *false*, ja meklējamais vārds nav atrasts.

test() piemērs:

```
var meklet = new RegExp("z");           // Tiek norādīts meklējamais
var virkne = "Jāpārbauda šī virkne.";   // simbols – „z".
document.write(meklet.test(virkne));    // Teksta mainīgais „virkne", kas
                                  // satur pārbaudāmo tekstu. Šajā gadījumā,
                                  // tiek izvadīts vārds – „false", jo
                                  // virkne nesatur simbolu „z".
```

- **exec()** – šī funkcija ir līdzīga iepriekš norādītajai *„test()"* funkcijai, taču to galvenā atšķirība ir tā, ka pēc pārbaudes tā izvada nevis *true* vai *false*, bet gan meklējamo simbolu, bet ja simbols nav atrasts, tad izvada vārdu – *„null*" (nulle, tukšs).

Bez simbolu virkņu apstrādāšanas un analīzes, tās var izvietot arī grupās un citās virknēs, veidojot datu masīvus. Datu masīvi ir līdzīgi datubāzēm, taču daudz vienkāršāki un ērtāk lietoja-

mi, arī tie var uzglabāt ļoti daudzveidīgus datu. Mājas lapās datu masīvi galvenokārt tiek izmantoti dažādu vārda dienu vai citu vārdu sarakstu veidošanai. JavaScript datu masīvus var uzrakstīt trīs veidos [13.][16.]:

1. piemērs:

```
var dzīvnieki = new Array();          // Tiek definēts datu masīvs.
dzivnieki[0]="Suns";                  // Tiek definētas visas datu masīva
dzivnieki[1]="Kaķis";                 // vērtības.
dzivnieki[2]="Trusis";
```

2. piemērs:

```
var dzivnieki=new Array("Suns","Kaķis","Trusis");
```

3. piemērs:

```
var dzivnieki=["Suns","Kaķis","Trusis"];
```

Kā redzams, tad visos dotajos piemēros tiek definēti trīs dzīvnieki – suns, kaķis un trusis, tādēļ apjoma ziņā visizdevīgākais datu masīvu variants ir norādīts 3. piemērā, lai gan jāatzīst, ka tam ir savi mīnusi, piemēram, nav numurēti datu masīva elementi, kā arī datu masīvs nav pārskatāms, ja tajā ir ļoti daudz elementu, tādēļ šādu datu masīva veidu vislabāk izmantot zināmu datu definēšanai, piemēram, mēneša, dienu nosaukumiem.

Bez šāda datu masīva iedalījuma, datu masīvus var iedalīt arī viendimensiju (redzami augšējos piemēros) un divdimensiju datu masīvos. Divdimensiju datu masīvi ir sarežģītāki, taču ir situācijas, kad ar to palīdzību datus var iegūt daudz vienkāršāk un ērtāk.

Lai iegūtu datus no datu masīva, jāievēro un jāizmanto sekojošā formula: datu masīva nosaukums[kārtas numurs].

Datu iegūšana no datu masīva (no iepriekš norādītajiem piemēriem):

```
document.write(dzivnieki[0]);         // Tiks attēlots vārds „Suns".
document.write(dzivnieki[1]);         // Tiks attēlots vārds „Kaķis".
document.write(dzivnieki[2]);         // Tiks attēlots vārds „Trusis".
```

Arī ar datu masīviem var veikt dažādas darbības, piemēram, tos var sakārtot alfabēta secībā, apvienot, izdzēst utt.

2.8. Notikumi un funkcijas

Lai izveidotu interaktīvu un lietotājam ērtu mājas lapu, noteikti jāizmanto tādi JavaScript elementi, kā notikumi un funkcijas, kas savstarpēji ir ļoti vienoti, jo bez funkcijas norādīšanas HTML un JavaScript kodā nevar ievietot notikumus.

Notikumi ir JavaScript elementi, kas liek izpildīt JavaScript kodu tikai tad, ja lietotājs veic kādu noteiktu darbību, piemēram, veic peles kreiso klikšķi, labo klikšķi, dubultklikšķi, pakustina peli, nospiež kādu taustiņu, pārlādē lapu u.c., tieši tādēļ notikumi ir tik noderīgi, lai veidotu inte-

raktīvu mājas lapu, jo tie ik brīdi var atbildēt uz lietotāja veiktajām darbībām.

Notikuma piemērs:

```
<html><head></head>                    // Šajā gadījumā – ja lietotājs veic
<script type="text/javascript">        // peles kreisās pogas klikšķi, tad tiek
function pele(){                       // parādīts paziņojums, ka lietotājs ir
alert("Jūs tikko veicāt klikšķi ar peli!");}
</script>                              // ka lietotājs ir veicis ir klikšķi.
</head><body onClick="pele()"></body></html>
```

Protams, jāatzīst, ka mājas lapās tik vienkārši skripti, kā piemērā, ir ļoti reti sastopami, un tie ir nedaudz bezjēdzīgi, taču ar šo piemēru palīdzību var labi apskatīt JavaScript notikumu darbību.

Dotā piemēra trešajā rindiņā redzams vārds *„function"*, kas arī ir JavaScript funkcijas elements. Funkcijas tiek izmantotas, lai savstarpēji nodalītu JavaScript koda daļas, līdzīgi, kā tas ir strukturētajās konstrukcijās, taču šajā gadījumā lietotājam jāizpilda kāds nosacījums, lai attiecīgā funkcija darbotos jeb tiktu izpildīta. Funkciju vispārīgi var raksturot šādi: function *funkcijas nosaukums(){}*. Īpaša uzmanība jāpievērš tam, ka funkcija ir jāiekļauj figūriekavās ({}), kā arī, lai funkcija tiktu izpildīta, uz to ir jāizveido norāde citviet, t.i., JavaScript kodā vai arī jāizveido notikums HTML kodā.

2.9. Citi sintakses elementi

Kā jau autors min šīs nodaļas sākumā, tad darbā JavaScript sintakse tiek apskatīta tikai virspusēji, lielāku uzmanību veltot JavaScript iespējām un pielietojumam mājas lapu izstrādāšanā.

Bez iepriekšējās nodaļās norādītajiem valodas elementiem, autors vēlas nedaudz aprakstīt arī sekojošos elementus, kas ir daudz sarežģītāki, nekā jau iepriekš norādītie un arī tiek izmantoti mājas lapu veidošanā:

- **Sīkdatnes.** Sīkdatnes ir īpašs JavaScript sintakses elements, kas ļauj lietotāja datorā saglabāt nelielas datnes, kas satur ļoti nelielu informācijas daudzumu. Saglabātās sīkdatnes JavaScript var nolasīt arī vēlāk, kad lietotājs otrreiz apmeklē mājas lapu. Sīkdatnes lielākoties tiek izmantotas, lai sniegtu informāciju par to, kad lietotājs pēdējo reizi ir apmeklējis attiecīgo mājas lapu (skatīt pielikuma 13. attēlu), taču sīkdatnes var pielietot arī citādi, piemēram, tās var glabāt informāciju par lietotāja veiktajām izmaiņām lapā, un nākošreiz atverot šo lapu, lietotājs šo lapu redzēs tādu pašu, kā iepriekšējā reizē. [13.]
- **Pārlūkprogrammas elementi.** Tiek izmantoti, lai noteiktu lietotāja pārlūkprogrammas nosaukumu, versiju. Šie elementi ir īpaši noderīgi tad, ja izveidotā mājas lapa ir saderīga tikai ar kādu noteiktu pārlūkprogrammu vai arī satur tādus elementus, kurus atbalsta tikai

pašas jaunākās pārlūkprogrammu versiju (par JavaScript un pārlūkprogrammu saderību sīkāk skatīt 1.4. apakšnodaļu).

- **Logu elementi.** Tiek izmantoti, lai noteiktu lietotāja datora ekrāna platumu, augstumu, aizvērtu, atvērtu vai pārlādētu kādu mājas lapas sadaļu, kā arī, lai noformētu mājas lapu, t.i., nomainītu fona, saišu krāsu u.c.
- **Datuma un laika elementi.** Kā jau izsaka nosaukums, tad šie elementi tiek izmantoti, lai noteiktu un apstrādātu lietotāja datora laiku, izvadītu attiecīgo rezultātu no datu masīviem, piemēram, attēlotu attiecīgās dienas vārda dienu gaviļniekus. Daži datuma un laika elementi un to pielietojumi nedaudz tika aplūkoti 2.6. apakšnodaļas piemēros, taču pastāv arī daudzi citi elementi – pārveidotāji (kas ļauj datumu pārveidot skaitliskā mainīgajā), taimeris (kas ļauj JavaScript turpināt savu darbību tikai pēc kāda noteikta laika) utt.

Protams, JavaScript elementi un to iespējas, funkcijas ir daudz, daudz vairāk, un lai tās pilnībā aprakstītu, ar piemēriem, būtu nepieciešamas vismaz 50 lapaspuses. No vienas puses, JavaScript tiek ļoti plaši izmantots, taču, no otras puses, tā sintakses iespējas pagaidām tiek izmantotas tikai nelielā apjomā; lielākoties, visās lapās atrodami vienādi, ne pārāk daudzveidīgi JavaScript kodi. Protams, pareizs ir arī uzskats, ka nav vajadzība veidot īpaši daudzveidīgus un sarežģītus skriptus, ja tādu pašu vai ļoti līdzīgu efektu var sasniegt ar daudz īsākiem un efektīvākiem skriptiem. [4.,8.-10.][21., 33.-35.]

3. JAVASCRIPT IZMANTOŠANA UN NOZĪME CITĀS JOMĀS

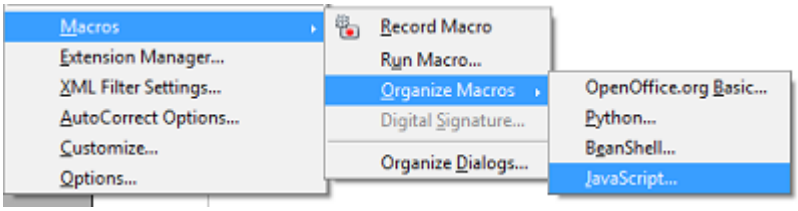
Iepriekšējās nodaļās autors secina, ka JavaScript ir programmēšanas valoda, kas paredzēta mājas lapu veidošanai, interaktīvo elementu izveidošanai, lapas kopējā izskata, funkciju uzlabošanai. Tomēr ir arī daudzas citas jomas, kurās pilnībā vai daļēji var izmantot JavaScript, un par to iegūtās zināšanas.

3. attēls
JavaScript Windows Sidebar rīkos [1.] [9.]

Pirmkārt, JavaScript ir noderīgs cilvēkiem, kuri vēlas iemācīties sarežģītās programmēšanas valodas, lai veidotu datorprogrammas, piemēram, C#, Visual Basic, C++ u.c. JavaScript sniedz ātru ieskatu programmēšanas valodu uzbūvē un to galvenajos elementos. Turklāt mūsdienās ir izveidotas dažas datorprogrammas, kas spēj kompilēt JavaScript kodu, tādējādi izveidojot arī nelielas datorprogrammas, kas spēj izpildīt vienkāršākās darbības un uzdevumus. Šāds risinājums ir neizdevīgs, jo JavaScript nenodrošina visus jaunāko programmēšanas valodu elementus, kā arī pārāk noslogo datora resursus, salīdzinoši ar citām programmēšanas valodām.. [1.][20.]

Otrkārt, JavaScript var izmantot, lai veidotu dažādus rīkus (skatīt 3. attēlu), kā pamatā ir HTML, CSS un JavaScript kodi. Šādu sīkrīku izmantošanu un veidošanu savās programmās nodrošina tādas kompānijas, kā Microsoft (Windows Sidebar), Google (Google Sidebar, Google Chrome Extensions), Yahoo (Widgets), Opera u.c.

Treškārt, JavaScript var izmantot, lai uzrakstītu īsas makrokomandas[14], šādu iespēju, piemēram, piedāvā OpenOffice programmas (skatīt 4. attēlu). Protams, ir iespējams atrast vēl daudzus citus JavaScript pielietojuma veidus, taču tie, pēc zinātniski pētnieciskā darba autora domām, nav tik izplatīti un lietderīgi, lai par tiem aprakstītu detalizētāku informāciju. No iepriekš aprakstītā, autors secina, ka JavaScript programmēšanas valodu var izmantot arī citās jomās.



4. attēls
JavaScript var izmantot makro komandu rakstīšanā [22.]

14 Makrokomanda – īpaši izveidota virkne (maza programma), kas liek datorprogrammai izpildīt noteiktas, lietotāja ievadītas (ierakstītas) darbības, tādējādi atvieglojot un paātrinot lietotāja darbu.

4. PRAKTISKĀ DAĻA – SKRIPTU VEIDOŠANA

Autors par sava darba praktiskās daļas uzdevumu izvirza skriptu veidošanu Elejas vidusskolas mājas lapai – http://www.elejasvsk.lv, uzlabojot jau esošos skriptus, kā arī izveidojot un pievienojot jaunus.

Sava pētnieciskā darba praktiskās daļas īstenošanai, tas ir, skriptu veidošanai, autors izmanto tikai PSPad un Notepad++ lietotnes [16.][19.] Windows 7 operētājsistēmā. Skriptu izveidi autors veic individuāli, savā brīvajā laikā, saskaņojot skriptu izpildes rezultātus ar nepieciešamajiem, izvirzītajiem skriptu darbības mērķiem. Lai izveidotu sarežģītākos skriptus, autors izziņas līmenī izmanto arī literatūras avotos sniegtos piemērus un paskaidrojumus. Pēc skriptu izveidošanas, autors veic to vairākkārtīgu testēšanu, uzlabošanu un papildināšanu. Zinātniski pētnieciskā darba autors par vienu no sava darba lielākajiem izveidotajiem skriptiem uzskata skriptu, kura izpildes rezultāts redzams Elejas vidusskolas galvenajā, sākuma lapā. Skripts izvada gan attiecīgās dienas vārda dienu gaviļniekus skolā, gan attiecīgajā dienā svinamās svētku un atzīmējamās dienas. Jāpiebilst, ka šis skripts izvada arī atlikušo laiku līdz skolas salidojuma datumam – 2011. gada 28. maijā. Skripta gala rezultātu var apskatīt pielikuma 9. attēlā, kā arī apmeklējot Elejas vidusskolas mājas lapu. Šī un citu autora veidoto skriptu pirmkodu var brīvi apskatīt http://arvis.wen9.com/skripti.zip.

Tā kā iepriekš minētais skripts attēlo trīs dažādas lietas, t.i., vārda dienu gaviļniekus skolā, svētku un atceres dienas, kā arī laiku līdz skolas salidojumam, tad visu skriptu veido trīs apjoma ziņā mazāki skripti:

1) Vārda dienu skripts. Iespējams, tieši šī skripta pilnvērtīga izveidošana un noformēšana darba autoram sagādāja vislielākās problēmas, jo, lai gan skripts jau bija izveidots iepriekš un veiksmīgi darbojās skolas mājas lapā, taču tajā esošo skolēnu saraksts bija novecojis, un tieši tādēļ skripta izvade bieži vien bija nepareiza. Galvenais iemesls skripta novecošanai bija skolēnu vārdu saraksts, kas katru gadu mainījās. Visa skolēnu vārdu saraksta pārskatīšana un rediģēšana ir ļoti darbietilpīgs process, tādēļ darba autors, vēl pirms paša vārda dienu skripta izveides, izstrādāja īpašu skriptu – ***Vārdu saraksta veidotājs***, kas jau no esošiem skolēnu sarakstiem, arī parasta teksta virknēm, spēj atrast un analizēt visus vārdus un tos alfabēta secībā sakārto datu masīvā, ko vēlāk var izmantot arī pašā vārda dienu skriptā – vienkārši nomainot iepriekšējā gada skolēnu vārdu datu masīvu, tādējādi tiek ievērojami samazināts mājas lapas kopējais atjaunināšanas laiks, jo vairs nav jāveic manuāla skolēnu vārdu saraksta analizēšana, bet gan viss tiek veikts automātiski; ir jānokopē tikai neliela koda rindiņa, lai skripts arī nākošajā mācību gadā darbotos pilnvērtīgi. Jāņem vērā, ka skripts atrod tikai tos vārdus, kas uzrakstīti pēc latviešu valodas nor-

mām, t.i., vārds sākas ar lielo sākuma burtu un visi pārējie vārda burti ir mazie, atstarpēm starp vārdiem vai citiem simboliem nav nozīmes. *Vārdu saraksta veidotājs*, bez vārdu atrašanas un datu masīva izveidošanas, spēj arī tekstā esošos vārdus sakārtot pēc to sastopamības biežuma, kā rezultātā, daudz parocīgāk ir izveidot Elejas vidusskolā visbiežāk sastopamo skolēnu vārdu sarakstu; atrodamas Elejas mājas lapas sadaļā – „Vecākiem".

Pēc vārdu masīva izveidošanas, darba autors aizstāja veco skolēnu vārdu sarakstu ar jauno, kā arī nedaudz optimizēja jau esošā skripta apjomu, uzlaboja tā funkcionalitāti.

2) Svētku un atceres dienu skripts. Lai gan skripts pēc tā darbības principa šķiet ļoti vienkāršs, jo tikai izvada noteiktajos datumos svinamās svētku un/vai atceres dienas, taču skripta izveide un darbība kopumā ir sarežģīta, jo datoram ir „jāiemāca" atpazīt tādas dienas, kam svinamais datums katru gadu atšķiras, piemēram, iekrīt maija otrajā svētdienā – Mātes diena – un citas, turklāt, piemēram, Lieldienas tiek svinētas pirmajā svētdienā pēc pilnmēness, kas iekrīt noteiktā laika intervālā starp 21. martu un 25. aprīli. Tādēļ ar skripta palīdzību darba autoram vispirms bija jāpanāk, ka dators analizē un atrod pilnmēness datumu, jo JavaScript pamata sintaksē šādas „pilnmēness" funkcijas nav, un tikai tad jāveic visi nepieciešamie aprēķini. Neskatoties uz radušajām problēmām, darba autoram izdevās veiksmīgi realizēt šo skriptu. Jāpiebilst, ka aptuveni pēc 2200. gada Lieldienu aprēķināšanas skriptā var rasties nelielas novirzes, taču ņemot vērā, ka šajā laika intervālā noteikti notiks ļoti daudzas izmaiņas tehnoloģiju jomā, tad darba autors šo skripta darbības ilgumu (~200 gadus) uzskata par pietiekami atbilstošu.

3) Salidojuma skripts. Viens no pēdējiem autora izstrādātajiem skriptiem, kura izstrādāšana aizņēma ievērojamu laiku. Lai gan arī šī skripta pamatuzdevums lietotājiem šķiet pavisam vienkāršs – aprēķināt laiku, kas atlicis līdz Elejas vidusskolas salidojumam, turklāt izvades rezultātu attēlot dienās, stundās un minūtēs. Šajā gadījumā datoram ir „jāiemāca" skaitīt dienas un pareizi aprēķināt laiku līdz brīdim, kad notiks skolas salidojums. Jāņem vērā, ka salidojums notiks plkst. 17.00, nevis pusnaktī, tādēļ šī skripta izveidošana bija sarežģīts un laikietilpīgs process. Skripta izveidē problēmas sagādāja arī pareizu vārdu galotņu noformēšana, piemēram, cilvēkiem ir viegli saprast, ka ir 1 minūte, 11 minūtes un 21 minūte, taču lai šo algoritmu spētu izpildīt dators, jāveic vairāki matemātiskie aprēķini un jāizmanto nosacījumi (ja…, tad…).

Neņemot vērā šos, „lielos" skriptus, darba autors Elejas skolas mājas lapā izveidoja arī dažus mazāka apjoma skriptus:

1) Tabulas rindu krāsas maiņas skripts. Elejas vidusskolas mājas lapas sadaļā „Skolēni", skatīt pielikuma 10. attēlu. Šis skripts tikai iekrāso aktīvo tabulas rindu, uz kuras lietotājs novieto ar peles kursoru, kā rezultātā, tas palīdz lietotāja skatienam nepārlēkt uz citu tabulas rindu, jo īpaši tad, ja tabulai ir daudz kolonu. Skripts labi parāda to, cik JavaScript valodā uzrakstītie

skripti ir lietderīgi interesantu, interaktīvu mājas lapas elementu izveidošanā.

2) Fotoattēlu maiņas skripts. Tā izpildes rezultāts apskatāms skolas mājas lapas „Foto" sadaļā. Šis skripts skolas mājas lapā jau darbojās pirms autora praktiskās daļas uzsākšanas, taču jāatzīst, ka tajā bija dažas nevajadzīgas koda rindiņas, kas nedaudz palēnināja skripta izpildes laiku, tādēļ tās tika izņemtas, citas rindiņas tika rediģētas – fotoattēlu maiņas skripts tika īpaši pielāgots, lai vienlaicīgi būtu universāls vairāku fotoattēlu grupām, t.i., šo skriptu būtu ļoti viegli izmantot arī jebkuriem fotoattēliem.

Skripta kods:

```
m = 1;
function pele(vards,sk,cels){m += 1;
if (m == sk){m = 1;}
document.getElementById(vards).src = "elejabildes/"+cels+m+".jpg";
document.getElementById(vards).title = m+". attēls no "+sk;}
```

3) Skolas vecuma aprēķināšanas skripts. Kopumā ļoti vienkāršs un neliels skripts, kas aprēķina un izvada skolas vecumu pilnos gados; skripts atrodams iepriekš norādītajā arhīva datnē.

Skripta kods:

```
function ev_gadi(){
var datums = new Date();
var gads = datums.getFullYear();
var menesis = datums.getMonth();
var diena = datums.getDate();
var dienu_sk = [-1, 30, 58, 89, 119, 150, 180, 211, 242, 272, 303, 333];
var gads0 = 1880; var diena0 = 1; var menesis0 = 9;
if (dienu_sk[menesis]+diena >= dienu_sk[menesis0-1]+diena0){
var g_starpiba = gads-gads0;}
else{var g_starpiba = gads-gads0-1;}
document.getElementById("ev_gadi").innerHTML = g_starpiba;
setTimeout(ev_gadi(), 1000);}
```

4) E-pasta adrešu aizsardzības skripts („Autori" sadaļā). Skripts galvenokārt izveidots, lai vienkārši un ērti aizsargātu gan skolēnu, gan skolotāju un skolas e-pasta adreses no nonākšanas dažādās mēstuļu, lieko, reklāmas vēstuļu, izsūtīšanas datubāzēs. Skripta darbības princips ir vienkāršs – tas no dotās funkcijas nolasa nepieciešamo informāciju un izveido, uz ekrāna izvada e-pasta adresi. Attiecīgā e-pasta adrese ir redzama tikai atverot mājas lapu pārlūkprogrammā, bet nav atrodama, izmantojot meklēšanas programmas un informācijas iegūšanas rīkus; skripta izpildes rezultāts apskatāms pielikuma 11. attēlā.

Skripta kods:

```
function epasts(id,vards,domens,tema){
```

```
document.getElementById(id).innerHTML = "<a
href='mailto:"+vards+"@"+domens+"?
subject="+tema+"'>"+vards+"@"+domens+"</a>";}
```

5) Krāsu maiņas skripts („Autori” sadaļā). Šis skripts neapšaubāmi pilda tikai vizuālo un interaktīvo funkciju, jo tas vienkārši atkal un atkal ģenerē jeb brīvi izvēlas krāsu, kādā izceļ darba autora vārdu „Arvis Lācis”, ja uz to tiek novietots peles kursors.

Skripta kods:

```
function krasas(){
var sarkans = Math.floor(Math.random()*255);
var zals = Math.floor(Math.random()*255);
var zils = Math.floor(Math.random()*255);
document.getElementById("arvis").style.color =
"rgb("+sarkans+","+zals+","+zils+")";}
```

Darba autors plāno veikt arī turpmāku darbību pie skolas mājas lapas izstrādes, kā arī veikt citus ar JavaScript un HTML valodām saistītus personīgos projektus, tādus kā, piemēram, Vārddienis (skatīt pielikuma 12. attēlu). Plašāku informāciju par Vārddieni ir iespējams atrast lapā http://varddienis.blogspot.com.

Veicot sava zinātniski pētnieciskā darba praktisko daļu, darba autors secina un apstiprina to, ka programmēšanas process ir darbietilpīgs, kā arī ir nepieciešamas labas prasmes, iemaņas arī tādās zinātnēs kā loģikā un matemātikā.

Lai gan autora praktiskajā daļā izveidotie skripti ir veidoti Elejas vidusskolas mājas lapai, taču šos pašus skriptus, to elementus vai daļas var izmantot arī citās mājas lapās, kā arī, piemēram, informātikas mācību stundās, lai iepazīstinātu skolēnus ar JavaScript valodu.

SECINĀJUMI

1. JavaScript ir viena no vienkāršākajām un vieglāk apgūstamajām programmēšanas valodām, ko galvenokārt izmanto interaktīvu elementu izveidošanā mājas lapās.
2. JavaScript programmēšanas valodu var izmantot arī citās jomās, piemēram, lai izveidotu vienkāršu datorprogrammu, vai arī dažāda veida rīkus.
3. JavaScript valodā nav iespējams uzrakstīt bīstamus datorvīrusus, kas spētu kaitēt lietotāja datoram.
4. Programmēšanas process, arī JavaScript valodā, ir darbietilpīgs un ir nepieciešamas prasmes, iemaņas arī tādās zinātnēs kā loģikā un matemātikā.
5. Programmējot, jāņem vērā visi iespējamie skripta izpildes rezultāti un parametri, tā skriptam papildus jāpievieno rindiņas, kas bieži ir daudz garākas nekā skripta pamatfunkcija, piemēram, īsais un garais gads laika aprēķinos.
6. Autors secina, ka viņa sākotnēji izvirzītais mērķis, kā arī uzdevumi ir īstenoti – ir izveidots darbs, kas atbilst izvirzītajiem kritērijiem un noteikumiem, un kurā apkopota svarīgākā un būtiskākā informācija par JavaScript, kā rezultātā, šo zinātniski pētniecisko darbu ir iespējams izmantot arī kā uzziņas materiālu.
7. Autora izveidotais darbs ir pietiekami sarežģīts, tādēļ tas var būt nesaprotams cilvēkiem, kuri datoru izmanto tikai vienkāršu darbību veikšanai (dokumentu rakstīšanai, interneta lietošanai, spēlēm, utt.).
8. Darba autors secina, ka, viņaprāt, darba iesākumā izvirzītā hipotēze ir apstiprinājusies, to pierāda gan interaktīvie, izveidotie piemēri, gan *Vārdu sarakstu veidotājs*, kas ievērojami paātrina mājas lapas vārdu sarakstu atjaunināšanas laiku.

IZMANTOTĀ LITERATŪRA

1. A Vista sidebar gadget to provide weather information [Elektroniskais resurss] – http://www.codeproject.com/KB/gadgets/vistaweatherwidget.aspx – Resurss aprakstīts 2010. gada 31. jūlijā
2. Andžāns A. Informātika 1. daļa (Ievads datorikā) – Rīga: Zvaigzne ABC, 1993.-82 lpp.
3. Balode A. Valoda Turbo Pascal – Programmēšanas pamati – Rīga: Zvaigzne ABC, 2009.-234 lpp.
4. Bucis R. WWW lappuse nav tikai skaista izkārtne // Datorpasaule – Rīga, 1997. / 2 (42) – 8.-10. lpp.
5. CSS Properties to JavaScript Reference [Elektroniskais resurss] – Hardwar.org.uk – http://codepunk.hardwar.org.uk/css2js.htm – Resurss aprakstīts 2010. gada 31. jūlijā
6. DELFI [Elektroniskais resurss] – Delfi.lv – http://www.delfi.lv – Resurss aprakstīts 2010. gada 20. februārī
7. Draugiem.lv [Elektroniskais resurss] – Draugiem.lv – http://www.draugiem.lv – Resurss aprakstīts 2010. gada 20. februārī
8. Elejas vidusskola [Elektroniskais resurss] – Elejas vidusskola – http://www.elejasvsk.lv – Resurss aprakstīts 2011. gada 27. februārī
9. Gadgets [Elektroniskais, lokālais resurss] – Windows Sidebar : Microsoft Corporation – [ASV], 2009. – C:\Program Files\Windows Sidebar\Gadgets
10. Internet [Elektroniskais resurss] – Wikipedia – http://en.wikipedia.org/wiki/Internet – Resurss aprakstīts 2010. gada 20. februārī
11. JavaScript [Elektroniskais resurss] – Vikipēdija – http://lv.wikipedia.org/wiki/JavaScript – Resurss aprakstīts 2010. gada 31. jūlijā
12. JavaScript [Elektroniskais resurss] – Wikipedia – http://en.wikipedia.org/wiki/JavaScript – Resurss aprakstīts 2010. gada 20. februārī
13. JavaScript Tutorial [Elektroniskais resurss] – W3Schools.com – http://www.w3schools.com/JS/... – Resurss aprakstīts 2010. gada 20. februārī

14. JavaScript syntax [Elektroniskais resurss] – Wikipedia – http://en.wikipedia.org/wiki/JavaScript_syntax – Resurss aprakstīts 2010. gada 31. jūlijā

15. Javascript – General Introduction [Elektroniskais resurss] – Quirksmode – http://www.quirksmode.org/js/intro.html – Resurss aprakstīts 2010. gada 31. jūlijā

16. JavaScript.DEF [Elektroniskais, lokālais resurss] – PSPad (v4.5.5) : Jan Fiala – [Čehija], 2011. – C:\Program Files (x86)\PSPad editor\Context\JavaScript.DEF

17. Jelgavas un Ozolnieku novadu informātikas MA [Elektroniskais resurss] – http://www.ima.jrp.lv – Resurss aprakstīts 2011. gada 27. februārī

18. JSLint, The JavaScript Code Quality Tool [Elektroniskais resurss] – JSLint – http://jslint.com – Resurss aprakstīts 2011. gada 19. februārī

19. Langs.model.xml [Elektroniskais, lokālais resurss] – Notepad++ (v5.8.7) : Don Ho – [Ķīna], 2011. – C:\Program Files (x86)\Notepad++\ langs.model.xml

20. Make your javascript a Windows .exe / Stoyan [Elektroniskais resurss] : Phpied.com – http://www.phpied.com/make-your-javascript-a-windows-exe/ – Resurss aprakstīts 2010. gada 16. janvārī

21. Melnūdris A. WWW Eiropas skolās // Datorpasaule – Rīga, 1997. / 2 (42) – 33.-35. lpp.

22. OpenOffice.org Writer [Elektroniskais, lokālais resurss] – OpenOffice.org : Sun Microsystems – [ASV], 2011.

23. Romanovskis T. Elektroniskais skaitļotājs skolā – Rīga: Zvaigzne, 1986.-260 lpp.

24. Vahers K. Ar HTML uz tu // Datorpasaule – Rīga, 1997. / 1 (41) – 38.-39. lpp.

25. Vahers K. Ar HTML uz tu // Datorpasaule – Rīga, 1997. / 4 (44) – 60.-61. lpp.

26. Zinātniskie projekti [Elektroniskais resurss] – http://www.elejasvsk.lv/dokumenti/ZPD_Eleja.doc

PIELIKUMS

4. tabula

Programmēšanas valodu attīstība un rašanās gadi [3.,8. lpp.]

Laiks	Programmēšanas valodas
1951-1955	Izteiksmju kompilatori
1956-1960	FORTRAN, ALGOL, LISP
1961-1965	**COBOL**, SNOBOL
1966-1970	APL, PL\1, **BASIC**, SIMULA
1971-1975	**Pascal, C**, Prolog
1976-1980	Smalltalk, Ada, ML
1981-1985	**Turbo Pascal**, Prolog, Postscript
1986-1990	FORTRAN 90, **C++**, SML
1991-1995	Ada 95, TCL, **Perl, HTML**
1996-2000	**Java, JavaScript, XML**
Treknrakstā atzīmētas valodas, kuras vēl tiek izmantotas mūsdienās.	

5. tabula

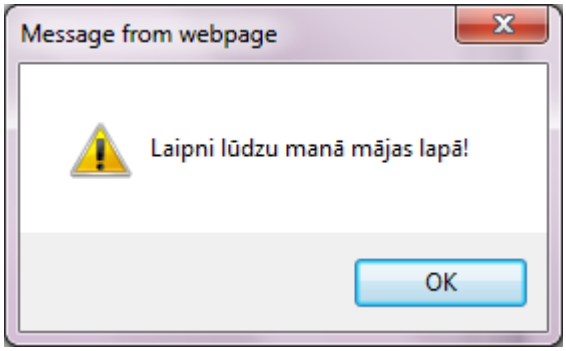
JavaScript matemātiskās konstantes un metodes [13.][14.]

Konstante	Aptuvenā vērtība	Metode	Apraksts	Piemērs
			Absolūtā	

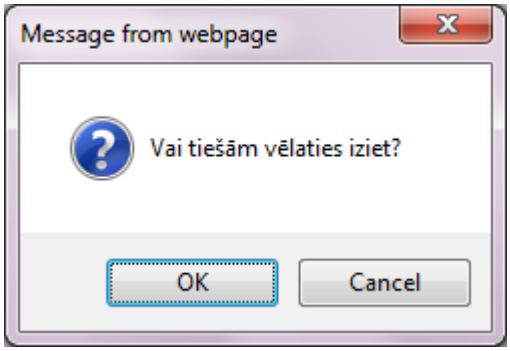
```
<script type="text/javascript" language="Javascript">
	var xmlhttpL=false;

	try { xmlhttpL = new ActiveXObject("Msxml2.XMLHTTP"); }
	catch (e) { try { xmlhttpL = new ActiveXObject("Microsoft.XMLHTTP"); }
	catch (e) { try { xmlhttpL = new XMLHttpRequest(); }
	catch (e) { xmlhttpL = false; }}}

	if (!xmlhttpL && typeof XMLHttpRequest!='undefined') {
	xmlhttpL = new XMLHttpRequest();
	}
	if(xmlhttpL){
	xmlhttpL.onreadystatechange = rqu;
	xmlhttpL.open("GET",'/dtct.html',true);
	xmlhttpL.send(null);
	}
	function rqu() {
	if (xmlhttpL.readyState == 4) {
		if (xmlhttpL.status == 200 || xmlhttpL.status == 304
	}
	}

	document.getElementById('feat1').value = '1';
	document.getElementById('res').value = screen.width+'x'+screen.height;
	if( typeof GetSwfVer != 'undefined' ) document.getElementById('feat3').value = GetSwfVer();


	function capsLock(e){
	 keycode = e.keyCode?e.keyCode:e.which;
	 shift = e.shiftKey?e.shiftKey:((keycode == 16)?true:false);
	 if(((keycode >= 65 && keycode <= 90) && !shift)||((keycode >= 97 && keycode <= 122) && shift))
	 document.getElementById('caps_lock').style.display = 'block';
	 else
	  document.getElementById('caps_lock').style.display = 'none';
	}
	</script>
```

5. attēls

*Draugiem.lv sākumlapas JavaScript kod*s [7.]



6. attēls
Brīdinājuma, paziņojuma loga paraugs



7. attēls
Apstiprinājuma loga paraugs

```
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59
60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87
88 89 90 91 92 93 94 95 96 97 98 99 100
```

8. attēls
Cikla darbības piemērs

9. attēls
Izveidotais skripts, kas parāda gan atlikušo laiku līdz skolas salidojumam, gan vārda dienas, gan svētku un atceres dienas [8.]

Sasniegumi Jelgavas un Ozolnieku novadu mācību priekšmetu olimpiādēs 2009./ 2010. m.g.

Priekšmets	Klase	Skolēns	Skolotājs	Vieta
Latviešu valoda un literatūra	8.	Zaiga Landorfa	Elija Tāraude	1.
Matemātika	8.	Zaiga Landorfa	Vita Cielava	1.
Matemātika	10.	Arvis Lācis	Vita Cielava	1.
Fizika	10.	Arvis Lācis	Jānis Tumovs	2.
Matemātika	5.	Elgars Rihards	Vita Cielava	2.
Latvijas un pasaules vēsture	9.	Eva Zevaliča	Antoņina Trivaškeviča	3.
Vizuālā māksla	4.	Margarita Zaula	Biruta Pundure	1.
Krievu valoda	8.	Anastasija Kazaka	Antoņina Trivaškeviča	1.
Krievu valoda	8.	Arta Pļaveniece	Rodijada Kazaka	1.
Matemātika	3.	Justīne Dižpētere	Baiba Butevica	1.

10. attēls
Izstrādātais skripts, kas iekrāso „aktīvo" tabulas rindu [8.]

11. attēls
E-pasta adrešu aizsardzības skripta izpildes piemērs [8.]

12. attēls
Vārddienis – sīkrīks, kura darbība balstās tikai uz JavaScript un HTML kodu pamata

```
<html>
<head>
<script type="text/javascript">
function getCookie(c_name)
{
if (document.cookie.length>0)
  {
  c_start=document.cookie.indexOf(c_name + "=");
  if (c_start!=-1)
    {
    c_start=c_start + c_name.length+1;
    c_end=document.cookie.indexOf(";",c_start);
    if (c_end==-1) c_end=document.cookie.length;
    return unescape(document.cookie.substring(c_start,c_end));
    }
  }
return "";
}

function setCookie(c_name,value,expiredays)
{
var exdate=new Date();
exdate.setDate(exdate.getDate()+expiredays);
document.cookie=c_name+ "=" +escape(value)+
((expiredays==null) ? "" : ";expires="+exdate.toGMTString());
}

function checkCookie()
{
username=getCookie('username');
if (username!=null && username!="")
  {
  alert('Sveicinati '+username+'!');
  }
else
  {
  username=prompt('Lūdzu ievadiet savu vārdu:',"");
  if (username!=null && username!="")
    {
    setCookie('username',username,365);
    }
  }
}
</script>
</head>
<body onload="checkCookie()">
</body>
</html>
```

13. attēls

Sīkdatnes piemērs, kas pieprasa ievadīt lietotājam savu vārdu, tad to saglabā, un nākošajā lapas atvēršanas reizē parāda paziņojumu. [13.]