

VILMA RIŠKEVIČIENĖ

---

# OBJEKTINIS PROGRAMAVIMAS

---

---

Mokymo(si) rinkinys





VILMA RIŠKEVIČIENĖ

---

# OBJEKTINIS PROGRAMAVIMAS

---

Mokymo(si) rinkinys

Parengta pagal 2007–2013 m. Žmogiškųjų išteklių plėtros veiksmų programos  
2 prioriteto „Mokymasis visą gyvenimą“ (projekto kodas Nr. VP1-2.2-ŠMM-07-K-01-027)  
priemonę „Studijų kokybės gerinimas, tarptautiškumo didinimas“

Spausdinti rekomendavo Marijampolės kolegijos Inžinerijos ir informacinių technologijų  
katedra 2012 m. balandžio 11 d. posėdžio sprendimu (protokolas Nr. 7)

Sudarė

*Vilma Riškevičienė*

Redagavo

*Nijolė Bagdonienė*

Recenzavo

informatikos inžinerijos magistrė *Vida Undzėnienė*

Marijampolės švietimo centro lektorė

# TURINYS

PRATARMĖ.....	7
IVADAS .....	8
1. OBJEKTINIO PROGRAMAVIMO PRINCIPAI .....	9
1.1. Klasės.....	9
1.2. Objektai ir klasių panaudojimas .....	12
1.3. Paveldėjimas .....	12
2. PHP KALBOS ISTORIJA IR PASKIRTIS.....	14
2.1. PHP privalumai.....	14
2.2. PHP kalbos sintaksė.....	15
2.3. Kintamieji ir kintamųjų tipai .....	15
2.4. Teksto rašymas naršyklės lange.....	18
2.5. PHP operatoriai.....	20
2.6. Sąlygos sakiniai .....	21
2.7. PHP Switch ir ciklo sakiniai.....	22
2.8. PHP funkcijos .....	23
2.8.1. Eilučių tvarkymo funkcijos .....	25
2.8.2. PHP datos ir laiko funkcijos .....	28
2.8.3. PHP masyvų funkcijos (array functions).....	29
2.9. Specialieji PHP masyvai.....	32
2.9.1. \$_GET masyvas.....	32
2.9.2. PHP \$_POST masyvas .....	33
2.9.3. PHP \$_REQUEST masyvas .....	34
2.10. Formų duomenų patikrinimas.....	35
2.11. Duomenų išsaugojimas serveryje .....	37
2.12. Formų apdorojimas – Formos atsakymo siuntimas el. paštu.....	40
2.12.1. Formos atsakymo siuntimas el. paštu.....	40
2.12.2. Antraštės .....	40
2.12.3. El. pašto adreso teisingumo tikrinimas.....	43
2.12.4. El. laiško siuntimas, panaudojant formas duomenis .....	43
2.12.5. El. laiško siuntimo testavimas .....	44
2.13. Formos duomenų įrašymas į failus .....	44
2.13.1. Skaitymas ir rašymas į failus.....	44
2.13.2. Tekstinė duomenų bazė .....	45
2.14. PHP slapukai („Cookies“) .....	46
2.15. PHP sesijos .....	46
3. JAVASCRIPT KALBOS PASKIRTIS .....	48
3.1. Pagrindiniai JavaScript elementai .....	48
3.2. Operatoriai .....	49
3.3. Sąlygos tikrinimo operatoriai .....	51
3.4. Ciklo operatoriai .....	52
3.5. JavaScript objektai.....	53

3.5.1.	Objektas Array .....	53
3.5.2.	Objektinis naršyklės modelis. JavaScript objektų hierarchija .....	54
3.5.3.	Objektas Date .....	55
3.5.4.	Objektas Function .....	57
3.5.5.	Objektas Math .....	57
3.5.6.	Objektas String .....	58
3.5.7.	Objektas Navigator .....	58
3.5.8.	Objektas Window .....	59
3.5.9.	Objektas Location .....	64
3.5.10.	Objektas Event .....	65
3.5.11.	Objektas Screen .....	65
3.5.12.	Objektas Document .....	66
3.6	Baziniai JavaScript įvykiai .....	68
3.6.1.	Įvykis OnAbort .....	69
3.6.2.	Įvykis OnBlur .....	69
3.6.3.	Įvykis OnChange .....	69
3.6.4.	Įvykis OnClick .....	70
3.6.5.	Įvykis OnError .....	72
3.6.6.	Įvykis OnFocus .....	72
3.6.7.	Įvykis OnLoad .....	73
3.6.8.	Įvykiai OnMouseOver ir OnMouseOut .....	73
3.7.	Išskleidžiamieji sąrašai .....	73
3.8.	Intarpų funkcijų derinimas .....	77
4.	PRAKTINĖS UŽDUOTYS .....	80
4.1.	Praktinė užduotis. HTML pagrindai .....	80
4.2.	Praktinė užduotis. PHP pradmenys, PHP kintamieji .....	87
4.3.	Praktinė užduotis. PHP masyvai. PHP ciklai, eilutės .....	89
4.4.	Praktinė užduotis. PHP skaitymas ir rašymas į/iš failus .....	94
4.5.	Praktinė užduotis. Loterija .....	95
4.6.	Praktinė užduotis. Formos ir jų apdorojimas .....	98
4.7.	Praktinė užduotis. PHP funkcijų rašymas .....	100
4.8.	Praktinė užduotis. Vartotojų sesijos .....	101
4.9.	Praktinė užduotis. JavaScript pradmenys, JavaScript kintamieji .....	103
4.10.	Praktinė užduotis. JavaScript operacijos .....	104
4.11.	Praktinė užduotis. JavaScript funkcijos .....	107
4.12.	Praktinė užduotis. JavaScript masyvai .....	110
4.13.	Praktinė užduotis. JavaScript datos ir laiko objektai, įvykiai .....	113
4.14.	Praktinė užduotis. JavaScript naršyklės, langai .....	117
4.15.	Praktinė užduotis. JavaScript formos .....	119
4.16.	Praktinė užduotis. JavaScript objektų masyvo kūrimas .....	121
	LITERATŪROS ŠALTINIAI .....	124

## PRATARMĖ

Mokymo(si) rinkinys skirtas ne tik Marijampolės kolegijos Verslo ir technologijų fakulteto bei kitų aukštųjų mokyklų studentams, bet ir visiems, norintiems išmokti objektinio programavimo pagrindų.

Šioje knygoje aprašomi objektinio programavimo ypatumai, supažindinama su dinaminių tinklalapių kūrimo pagrindais: *PHP* ir *JavaScript* technologijomis. Pateikiami nuoseklūs praktinių darbų aprašymai. Šiuo metu *PHP* programiniai intarpai naudojami milijonuose interneto svetainių. Tai pažangi ir plačiai paplitusi objektinio programavimo kalba, kuri vartojama aprašyti įvairiems sudėtingiems interneto technologijų procesams. *JavaScript* kalba skirta interaktyviems tinklalapiams kurti ir turi priemonių naršyklei valdyti.

## IVADAS

Mokomoji knyga skirta studentams, besimokantiems objektinio programavimo dalyko, taip pat tiems, kurie nori plačiau susipažinti su objektinio programavimo principais, interneto technologijomis.

Knygą sudaro keturi skyriai, kurių kiekvienas suskirstytas į poskyrius. Pirmuose trijuose skyriuose pateikta teorinė medžiaga, gausiai iliustruota pavyzdžiais, ketvirtajame – praktinės užduotys bei savarankiško darbo užduotys.

Pirmajame skyriuje nagrinėjami objektinio programavimo principai. Pateikiami pagrindiniai objektinio programavimo terminai.

Antrasis mokomosios knygos skyrius skirtas supažindinti su interneto serveriuose vykdomų procesų aprašymo kalba *PHP*. Aptariamose kodo įterpimo į *HTML* dokumentą galimybės, nagrinėjama *PHP* sintaksė, kintamųjų aprašymo taisyklės, valdymo struktūros, duomenų masyvo tvarkymo priemonės.

Trečiajame skyriuje supažindinama su *JavaScript* priemonėmis, nagrinėjami kodo įterpimo į *HTML* dokumentą principai ir taisyklės, aiškinama sintaksė, sąlygos ir ciklo sakiniai, valdymo struktūros, pateikiami dialoginių langų kūrimo pavyzdžiai.

Ketvirtasis mokomosios knygos skyrius skirtas praktinėms užduotims. Kiekvienos praktinės užduoties pabaigoje yra pateikiama savarankiško darbo užduočių.

Metodinio leidinio medžiaga parengta naudojantis Antano Vidžiūno, Daivos Vitkutės „Interneto paslaugos ir svetainių kūrimas“, Liudviko Kaklauskio „Tinklalapiai ir jų kūrimas“, P. Širvinsko „PHP pamokos pradedantiesiems“, J. N. Robbins „Tinklalapių dizainas. (X)HTML kalbos, pakopinių stilių ir tinklalapių grafikos pradžiamokslis“ ir internetiniais šaltiniais, kurie nurodyti literatūros sąrašė.



# 1. OBJEKTINIO PROGRAMAVIMO PRINCIPAI

Objektinis programavimas *OOP* (angl. *Object-Oriented Programming*). Pagrindinė OOP idėja – duomenų ir funkcijų, atliekančių veiksmus su šiais duomenimis apjungimas į vieną visumą–objektą:

- objekto duomenys objekto išorėje tiesiogiai nėra prieinami: jei reikia kokių nors duomenų iš objekto, reikia kviesti funkciją – objekto metodą;
- jei reikia keisti objekto duomenis, taip pat reikia kviesti funkciją – objekto metodą.

*PHP* kalba jau pakankamai ištobulinta ir joje yra viskas, ko reikia *OOP* programavimui atlikti. Pagrindiniai objektinio programavimo terminai:

*Objektas* – duomenų ir susijusio funkcionalumo talpinimas į vientisus vienetus. Objektai padeda pasiekti moduliškumą ir nusako objektinės programos struktūrą.

*Abstrakcija* – galimybė programuoti nežinant konkrečių detalių apie informaciją.

*Inkapsuliacija (informacijos slėpimas)* – užtikrina, kad objekto naudotojas negali pakeisti objekto būsenos nenumatytu būdu. Tik objekto vidiniai metodai turi galimybę keisti objekto būseną. Objektas pateikia interfeisą (sąsają), nusakančią galimybes manipuluoti objektu.

*Polimorfizmas* – objektiškai orientuotos kalbos nekviečia paprogramių, bet siunčia pranešimus, todėl konkretus į pranešimą reaguojantis metodas priklauso nuo objekto, ne nuo siuntėjo. Tai leidžia dirbti su bendresnio tipo objektais (pvz., paukštis), nežinant konkretaus tipo (pvz., strutis ar pingvinas), dėl to nereikia kiekvienam konkrečiam tipui rašyti atskiro kodo.

*Paveldėjimas* – objektų organizavimas, specializuojant egzistuojančius bendresnius tipus, papildant ar iš dalies pakeičiant funkcionalumą.

## 1.1. Klasės

*Klasė* – rinkinys funkcijų ir kintamųjų, susijusių tarpusavyje.

*Savybės* – klasėje naudojami kintamieji.

*Metodai* – klasėje naudojamos funkcijos.

*Konstruktorius* – funkcija įvykdoma sukuriant klasę (sukuriant objektą).

*Klasės sintaksė* – apibrėžiama žodžiu *class*, tada nurodomas klasės pavadinimas ir vėliau laužtiniuose skliaustuose { } rašomas visas klasės tekstas.

Pvz.

```
// pirmiausia – klasė darbui su vartotojų duomenimis,  
// kurie saugomi duomenų bazėje  
class Vartotojai {  
function SukurtiVartotoja($vardas, $slaptazodis, $el_pastas) {  
$query = "insert into vartotojai ...";
```

```

// ... funkcija vartotojo sukūrimui
}
function VartotojoDuomenys($id) {
    $query = "select * from vartotojai where id = $id";
    // ... iš duomenų bazės paimami vartotojo duomenys
}
function PrisijungimoDuomenys($vardas, $slaptazodis) {
    $query = "select id from vartotojai
    where vardas = '". $vardas. "'
    and slaptazodis = '". $slaptazodis. "'";
    // ... patikriname, ar vartotojas įvedė
    // teisingus prisijungimo duomenis
}
}
/* ----- */
// antra klasė - naudingų nuorodų talpinimas ir grupavimas
class Nuorodos {
    function ĮterptiNuoroda($pavadinimas, $kategorija,
    $adresas, $aprasymas) {
        $query = "insert into nuorodos ...";
        // ... įterpia nuorodos įrašą į duomenų bazę
    }
    function NuoroduKategorijos() {
        $query = "select * from nuorodu_kategorijos order by pavadinimas";
        // ... ištraukia iš duomenų bazės nuorodų kategorijas pagal abėcėlę
    }
    function IdomiausiosNuorodos() {
        $query = "select nuorodos.pavadinimas, count(*) as ivertinimai
        from nuorodu_ivertinimai
        join nuorodos on nuorodu_ivertinimai.nuorodos_id = nuorodos.id
        group by nuorodos.pavadinimas order by ivertinimai
        limit 0,10";
        // ... sudėtingesnė užklausa į duomenų bazę –
        // TOP 10 nuorodų pagal lankytojų įvertinimus
    }
}
/* ----- */
// ir galiausiai – klasė, susijusi su komentarais
class Komentarai {
    function ĮterptiKomentara($nuoroda_id, $vartotojas_id, $komentaras) {
        $query = "insert into nuorodu_komentarai ...";
        // ... įterpia vartotojo komentarą prie nuorodos
    }
    function NaujausiKomentarai() {
        $query = "select * from nuorodu_komentarai
        order by id desc limit 0,10";
    }
}

```

```
// ... ištraukia iš duomenų bazės 10 naujausių komentarų
}
function NuorodosKomentarai($nuoroda_id) {
    $query = "select * from nuorodu_komentarai
    where nuoroda_id = $nuoroda_id
    order by id desc limit 0,10";
    // ... ištraukia iš duomenų bazės komentarus
    // prie konkrečios nuorodos
}
}
```

Be funkcijų (arba kitaip – klasės metodų), klasė gali turėti ir savo vidinius kintamuosius arba klasės atributus:

Pvz.

```
// pavyzdys – klasė, skirta darbui su duomenų baze
class DuomenuBaze {
    var $serveris = "localhost";
    var $vartotojas = "admin";
    var $slaptazodis = "admin12345";
    var $duomeniu_baze = "projektas";
    var $prisijungimas;
    function Prisijungimas() {
        $this->prisijungimas = mysql_connect($this->serveris,
        $this->vartotojas, $this->slaptazodis);
        mysql_select_db($this->$duomeniu_baze);
        // ... funkcija prisijungimui prie duomenų bazės serverio
    }
}
```

Taip pat svarbu, kaip kreiptis į tuos kintamuosius. Tam vartojamas žodelis *this* bei simbolių junginys *→*. Lygiai taip kreipiamasi ir į klasės metodą, jeigu jį reikia iškviesti iš kito metodo:

Pvz.

```
// pavyzdys – klasė, skirta darbui su duomenų baze
class DuomenuBaze {
    var $serveris = "localhost";
    var $vartotojas = "admin";
    var $slaptazodis = "admin12345";
    var $duomeniu_baze = "projektas";
    var $id;
    var $rezultatas;
    function Prisijungimas() {
        $this->id = mysql_connect($this->serveris,
        $this->vartotojas, $this->slaptazodis);
        mysql_select_db($this->$duomeniu_baze);
        // ... funkcija prisijungimui prie duomenų bazės serverio
    }
}
```

```

}
function Uzklausa($query) {
    $this->Prisijungimas();
    $this->rezultatas = mysql_query($query);
    // ... tolimesnis užklauskos rezultatų apdorojimas
}
}

```

## 1.2. Objektai ir klasių panaudojimas

Bendru atveju, jeigu reikia panaudoti funkciją, ji yra iškviečiama – parašomas jos pavadinimas ir parametrai. Naudojantis klasėmis, yra kiek kitaip: prieš tai reikia sukurti klasės objektą, o tada jau galima atlikti veiksmus su tuo objektu:

```
$db = new DuomenuBaze();
```

Kintamajam priskiriamas pasirinktos klasės apibrėžimas. Tas kintamasis vadinamas klasės objektu. Vėliau būtent su tuo objektu bus atlikti veiksmai. Galima sukurti kelis tos pačios klasės objektus ir su jais dirbti atskirai.

Pvz.

```

$db1 = new DuomenuBaze();
$db1->duomenu_baze = "pirmas_projektas";
$db1->Prisijungimas();
// kitas tos pačios klasės objektas
$db2 = new DuomenuBaze();
$db2->duomenu_baze = "antras_projektas";
$db2->Prisijungimas();

```

Sukūrus objektą, galima ne tik iškviešti jo metodus, bet ir priskirti reikšmę jo atributams – kintamiesiems, nuo kurių priklauso metodų iškvietimo parametrai.

## 1.3. Paveldėjimas

Dar viena svarbi objektinio programavimo galimybė – kurti klases, kurios paveldi viena kitą.

Sakykime, yra klasė *Transportas*, kurią panaudojant bus dirbama su duomenimis apie transporto priemones. Projektui išsiplėtus, reikia atskiros funkcionalumo krovininiams automobiliams, ir atskirų funkcijų motociklams. Bet dalis iš tų funkcijų vis tiek turi būti bendros. Sprendimas atrodo taip:

```

// bendra klasė, skirta
// bendra klasė, skirta duomenims apie transportą
class Transportas {
    function RegistruotiTransporta() {
        // ... funkcija transporto priemonės registravimui
    }
}

```

```

}
// ... dar keletas funkcijų, skirtų bendrai transporto priemonėms
}
// klasė krovininiams automobiliams
class Krovininiai EXTENDS Transportas {
var $krovinio_mase;
function KrovinioMase() {
// ... funkcija, skirta tik krovininiam transportui
}
}
// klasė motociklams
class Motociklai EXTENDS Transportas {
var $motociklu_klase;
function MotocikloKlase() {
// ... funkcija, skirta tik motociklams
// klasių objektų sukūrimas
function $krovininis = new Krovininiai();
$krovininis->KrovinioMase();
$krovininis->RegistruotiTransporta();
function $motociklas = new Motociklai();
$motociklas->MotocikloKlase();
$motociklas->RegistruotiTransporta();
}
}

```

Specialiu žodeliu *extends* nurodoma, kad kuriama klasė paveldima iš kitos klasės ir taip pat paveldi visas jos savybes bei objektus. Dviejų sukurtų paveldėtų klasių objektai naudoja savo individualias funkcijas, taip pat ir bendras „tėvo“ klasės funkcijas.

## 2. PHP KALBOS ISTORIJA IR PASKIRTIS

*PHP* kalba pradėta vartoti 1994 metų rudenį, kai Rasmus Lerdorfas nusprendė susikurti priemonių sistemėlę, kuri palengvintų asmeninio puslapio kūrimo darbą. Savo sistemos versiją 1.0 ir idėjas jis pristatė 1995-ais metais konferencijoje pranešime „*Personal Home Page Tools*“. Toliau viskas sparčiai vystėsi: 1995 metų viduryje atsiranda antroji Lerdorfo sistemos versija (*PHP/FI version 2*). *FI* – pagrindinė naujovė – *Formų (blankų) Interpretatorius*. *PHP/FI* integravosi į *Apache Web* serverį, naudojo standartinę *Apache API*. *PHP* dirbo greičiau, nei *CGI/Perl* skriptai, jos galimybės artėjo prie *Perl* kalbos teikiamo serviso, plito kalbos galimybės bendrauti su duomenų bazėmis.

1997 metų pabaigoje Zeev Suraski ir Andi Gutmans perrašo *PHP* realizaciją, ištaiso pasitaikiusias senos versijos klaidas ir pagreitina *PHP* skriptų vykdymą.

1998 viduryje atsiranda nauja versija, pavadinta *PHP3*, kuria 1999 metais jau naudojami apie 106 registruotų *www* tinklo serverių.

1999 metų pabaigoje perrašoma *PHP3* realizacija, kurios variklius pavadinamas *ZendEngine*, ir išleidžiama nauja versija *PHP4*. Prie *PHP* atskirų modulių kūrimo prisijungia masė entuziastų, sukuriamas grafikos modulis *gd* ( autoriai: Rasmus Lerdorf, Stig Bakken, Jim Winstead, Jouni Ahto), ryšio tarp kompiuterių palaikymo ir tvarkymo modulis *sockets* (Chris Vandomelen, Sterling Hughes, Daniel Beulshausen) ir daug kitų.

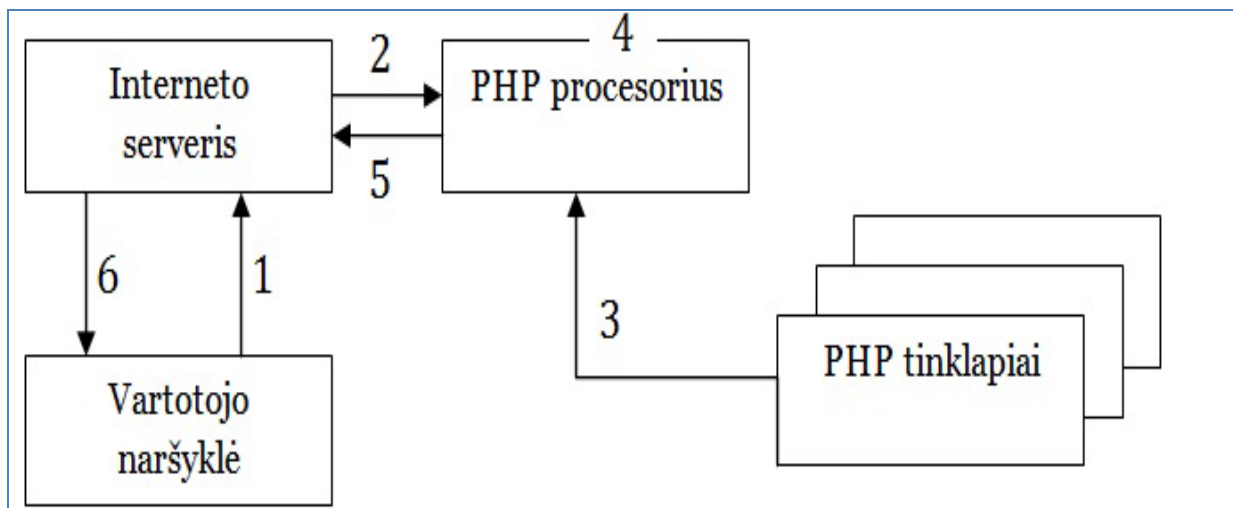
2004 metais atsirado *PHP5* versija. Naujoje versijoje praplėstos objektinio programavimo galimybės.

### 2.1. PHP privalumai

*PHP* kalba turi šiuos privalumus:

- *PHP* kalbos programos gali būti vykdomos įvairių operacinių sistemų (*Windows*, *Unix*, *Linux* ir t. t.);
- atvirojo kodo principas;
- dauguma sintaksės taisyklių ir naudojamų struktūrų perimtos iš populiarių programavimo kalbų (*C++*, *JAVA*);
- pritaikyta aprašyti informaciniams ryšiams su duomenų bazių valdymo sistema *MySQL*.

Šiuo metu *PHP* programiniai intarpai naudojami milijonuose interneto svetainių. Dabar tai galinga objektinio programavimo kalba, kuria galima aprašyti įvairius sudėtingus interneto technologijų procesus. Naujausią *PHP* diegimo programų paketą galima parsisiųsti iš svetainės <http://www.php.net> ir įdiegti serveryje. Tinklapių failai su *PHP* programų intarpais saugomi failuose su vardo plėtiniu *php*.



2.1.1 pav. Tipinis PHP dokumentų užklausų aptarnavimo ciklas

Tokių užklausų tipinį aptarnavimo ciklą iliustruoja 2.1.1 pav.

1. naršyklės adreso langelyje nurodyto *PHP* tinklalapio paieškos užklausos siuntimas į serverį;
2. užklausos perdavimas *PHP* intarpų aptarnavimo programų paketui (*PHP* procesoriui);
3. tinklalapio skaitymas iš diskinės serverio atminties;
4. tinklalapyje įrašytų *PHP* instrukcijų analizė ir vykdymas;
5. tinklalapio su *PHP* instrukcijų įvykdymo rezultatais perdavimas į serverį;
6. tinklalapio siuntimas į naršyklės kompiuterį.

## 2.2. PHP kalbos sintaksė

*PHP* (*PHP hypertext preprocessor*) – tai kodų kūrimo kalba, kuri pradžioje buvo orientuota tik į internetą, o ilgainiui tapo programavimo kalba. *PHP* kodai yra interpretuojami ir vykdomi serverio pusėje.

## 2.3. Kintamieji ir kintamųjų tipai

Viena svarbiausių ir esminių programavimo kalbos struktūrų yra kintamieji. Pagrindinė kintamojo paskirtis – saugoti duomenis, kiekvienas kintamasis skripto vykdymo momentu įgyja tam tikrą reikšmę, kuri, vykdant skriptą, gali keistis. Kintamieji gali būti gaunami iš *HTML* formų, yra sukuriami automatiškai programos vykdymo pradžioje. Jie atlieka labai svarbų vaidmenį, nes perduoda vartotojo įvestą informaciją. Kintamasis apibrėžiamas nurodant jo vardą. Išskirtinis *PHP* kintamojo požymis yra ženklas "\$". Kintamųjų tipai:

*Integer*; // sveikojo skaičiaus tipas

*Floating-point numbers*; // slankaus kablelio skaičiai, arba kitaip – realieji skaičiai

*String*; // eilutės tipas

*Array*; // struktūrinio kintamojo tipas – masyvas

*Object*; // objekto tipas (objektiniam programavimui)

Dažniausiai kintamojo tipas yra nustatomas ne paties programuotojo, o skripto kompiliatoriaus. Jis pats priskiria kintamajam tinkamą tipą. Jei reikia keisti kintamojo tipą programos metu, tai galima padaryti naudojant *settype()* funkciją.

*Integer* (sveikųjų skaičiaus tipas). Šis tipas priskiriamas kintamajam šiais atvejais:

*\$a*= 1234; # *dešimtainis skaitmuo*

*\$a*= -123; # *neigiamas skaitmuo*

*\$a*= 0x12; # *šešioliktainis skaičiaus pavidalas (lygus 18 dešimtainėje sistemoje)*

*Floating-point numbers* (slankaus kablelio skaičiai, arba kitaip – realieji skaičiai). Du būdai, vaizduojantys realųjį skaičių:

*\$a*= 1.234;

*\$a*= 1.2e3;

*String* (eilutes tipas). *String* tipas gali būti vaizduojamas programavimo kalbose dviem būdais:

„Viengubose“ kabutėse pvz.: *a*\$='Testas';

„Dvigubose“ kabutėse pvz.: *a*\$="Kitas testas";

*Array* (struktūrinio kintamojo tipas – masyvas). Vienmatis masyvas, kurį galima įsivaizduoti kaip vieną ilgą seką elementų, kurie susiję savo kintamųjų tipais.

Pvz.

savaitės dienų masyvas: ["pirm", "antr", "trec", "ketvr", "penkt", "sest", "sekm"]

*\$savdienos*[0]="pirm";

*\$savdienos*[1]="antr";

*\$savdienos*[2]="trec";

*\$savdienos*[3]="ketvr";

*\$savdienos*[4]="penkt";

*\$savdienos*[5]="sest";

*\$savdienos*[6]="sekm";

Pastaba. *PHP* masyvo pirmojo elemento indeksas yra 0, o ne 1.

Į vienmatį masyvą įrašyti elementus galima ir šiuo būdu:

*\$savdienos*[ ]="pirm"; // 0 elementas

*\$savdienos*[ ]="antr"; // 1 elementas

*\$savdienos*[ ]="trec"; // ir t. t.

*\$savdienos*[ ]="ketvr";

*\$savdienos*[ ]="penkt";

*\$savdienos*[ ]="sest";

*\$savdienos*[ ]="sekm";

Dvimatis masyvas (matrica) vaizduojamas panašiai:

*\$skaicius*[1][2] = *\$kitas*; // dvimatis masyvas



Daugiamačiai (trimačiai ir t.t.) masyvai vaizduojami analogiškai:

```
$masyvas[1][2][3] = $kitas; // trimatis masyvas
```

*Object* (objekto tipas). Sukurti objektą ir vėliau jį naudoti galima taip:

```
Class objektas
```

```
{
```

```
Function parasyti()
```

```
{
```

```
echo "SVEIKAS PASAULI";
```

```
}
```

```
$naujas = & new objektas;
```

```
$naujas-> parasyti();
```

```
}
```

*PHP* kalboje nėra griežtos kintamųjų tipų kontrolės. Kintamųjų tipai nustatomi priklausomai nuo jiems suteiktų reikšmių tipo, o jeigu vėliau kintamajam suteikiama kito tipo reikšmė, pakinta ir kintamojo tipas. Kintamojo tipas nurodo, kaip jo reikšmė saugoma kompiuterio atmintinėje, kokius veiksmus jai leidžiama taikyti.

Programos dažnai būna vaizdesnės, kai jose naudojamos konstantos. Tai vardais nurodomos specialios paskirties reikšmės, kurias programoje keisti draudžiama. Konstantas sukuria funkcija *define*:

```
Define ("Konstantos vardas", Reikšmė);
```

Siekiant, kad konstantų vardai geriau išsiskirtų iš programų teksto, joms rekomenduojama skirti didžiosiomis raidėmis rašomus vardus ir juose nenaudoti priešdėlio *\$*. Konstantų aprašymo pavyzdžiai:

```
define ("KAINA", 100.25);
```

```
define ("KIEKIS", 10);
```

```
define ("PVM", 0.18);
```

Aprašant duomenų mainus su išore naudojamos simbolių eilutės, o aritmetinius skaičiavimus galima taikyti tiksliai skaičiams, todėl aktuali duomenų tipų keitimo problema. Programavimo kalbose su griežta tipų kontrole tipų keitimo veiksmams turi rūpintis ir juos aprašyti programuotojas. Programose *PHP* kalba plačiai naudojamas automatinis tipų keitimas, kai duomenų interpretavimo būdas priklauso nuo konteksto. Dviviečiuose aritmetiniuose veiksmuose taikomos tokios automatinio tipų keitimo taisyklės:

- jei vienas argumentas yra realusis skaičius, o antrasis – sveikasis skaičius, sveikasis skaičius pertvarkomas į realųjį tipą;
- jei vienas argumentas yra realusis skaičius, o antrasis – simbolių eilutė, simbolių eilutė pertvarkoma taip, kad jos reikšmė atitiktų kito argumento reikšmės tipą.

Šios taisyklės leidžia taikyti aritmetinius veiksmus simbolių eilutėmis aprašomoms reikšmėms. Į skaičių galima pertvarkyti bet kurią eilutę, kurios pradžioje įrašyta skaičiaus reikšmė. Jei toliau eilutėje yra raidės arba kiti simboliai, jie atmetami. Jeigu pradinis eilutės

simbolis nėra skaitmuo, ji keičiama skaičiumi 0. Aritmetinių veiksmų, kurių vienas argumentas yra simbolių eilutė, skaičiavimo pavyzdžiai:

```
"9 Lt" - 1;           // Sveikasis skaičius 8
"0.55 kg" * 2;        // Realusis skaičius 1.1
"9 m." - 0.5;         // Realusis skaičius 8.5
"1E3 taškų" + 1;      // Realusis skaičius 1001
```

Kai pageidaujama tiesiogiai apibrėžti kintamajam suteikiamos reikšmės tipą, naudojamas tipo keitimo operatorius. Jo aprašymo sintaksė tokia:

*(Parinktas reikšmės tipas) Reikšmė*

Prenkamas reikšmės tipas nurodomas pagrindiniais kalbos žodžiais: *integer* (sveikasis), *double* (realusis) ir *string* (simbolių eilutė). Kintamųjų reikšmių tipo nurodymo pavyzdžiai:

```
$skiekis = 0;           // Automatiškai parenkamas sveikasis tipas
$suma = 0.00;          // Automatiškai parenkamas realusis tipas
$viso = (double) $skiekis; // Tiesiogiai nurodomas realusis tipas
$tekstas = (string) $suma; // Tiesiogiai nurodomas eilutės tipas
```

*PHP* programose taip pat tenka aprašyti įvairius tekstų tvarkymo veiksmus. Šiam tikslui naudojamas tašku žymimas dvivietis eilučių sujungimo operatorius ir ištisas tekstų tvarkymo funkcijų rinkinys. Sujungimo operatoriaus rezultatas yra simbolių eilutė, kurioje prie kairiojo argumento eilutės prijungiama dešiniojo argumento eilutė. Jeigu kuris nors operatoriaus argumentas yra skaičius, jis automatiškai pertvarkomas į ekvivalentinę simbolių eilutę ir tada taikomas sujungimo veiksmas. Eilučių sujungimo operatoriaus taikymo pavyzdžiai:

```
$a = "Firma";
$b = " 'Aukso veršis' ";
$pvadinimas = $a.$b; // Rezultatas: "Firma 'Aukso veršis' "
$n = 5;
$s = 'Gauta: ' . $n . ' kg '; //Rezultatas: 'Gauta: 5 kg'
$t = 3 . 2.74 // Rezultatas: "32.74"
```

## 2.4. Teksto rašymas naršyklės lange

Vienas svarbiausių *PHP* programų privalumų tas, kad jomis galima aprašyti serveryje rengiamų duomenų įterpimą į *HTML* dokumentus ir jų perdavimą naršyklių programoms. Tokiems veiksams aprašyti naudojamas operatorius *echo* ir funkcijos *print*, *printf* bei *print\_r*. Kreipiniai į funkcijas rašomi taip:

*Funkcijos vardas (Argumentas arba jų sąrašas) ;*

Į naršyklę perduodamus duomenis paprasčiausia aprašyti operatoriumi *echo*:

*Echo Simbolių eilučių sąrašas;*

Simbolių eilutės yra tarp paprastų arba dvigubų kabučių įrašyti bet kurie naršyklės palaikomo alfabeto simbolių rinkiniai, pavyzdžiui:

```
echo "Labas, Jonai!"; // Vienos eilutės perdavimas naršyklei
```

```
echo 'Labas, ', 'Jonai!'; // Dviejų eilučių perdavimas naršyklei
```

Paprastose ir dvigubose kabutėse rašomų simbolių eilučių savybės labai panašios, tačiau ne identiškos. Pavyzdžiui, dvigubomis kabutėmis išskirtose eilutėse galima įterpti kintamųjų reikšmes, o kito tipo eilutėse to daryti negalima. Jeigu simbolių eilutėje būtina įterpti jos ribas žymintį simbolį (") , tai galima padaryti rašant jį su priešdėliu \, pavyzdžiui:

```
echo "Firma "Microsoft" "; // Sintaksės klaida
```

```
echo "Firma \"Microsoft\" "; // Teisingas simbolių " įterpimas
```

Funkcija *print* skiriasi nuo operatoriaus *echo* tuo, kad jai gali būti perduodama tikrai viena eilutė. Be to, funkcijos grąžinama loginė reikšmė praneša, ar argumento eilutė naršyklės lange įrašyta sėkmingai. Todėl tikrinant funkcijos *print* grąžinamą reikšmę, galima nustatyti, ar pavyko perduoti duomenis į naršyklės kompiuterį.

Naudojant funkciją *printf* galima nurodyti naršyklei perduodamų duomenų rašymo būdus – formatus. Šios funkcijos aprašymo sintaksė tokia:

```
printf (Šablonas, Šablone įterpiamos reikšmės);
```

Šablonas yra teksto eilutė su specialiomis funkcijai perduodamų reikšmių įterpimo būdą aprašančiomis žymėmis, kurios aprašo reikšmėms skiriamų laukų dydžius bei kitas savybes:

```
%[Užpildo simbolis] [Ženklas] Lauko dydis Lauko tipo simbolis
```

Šiame įterpimo žymės struktūros aprašyme laužtiniai skliaustai žymi neprivalomus jos elementus. Kaip matyti iš sintaksės aprašymo, kiekvienos įterpimo žymės pradžioje rašomas privalomas simbolis %. Toliau gali būti nurodomas užpildo simbolis, rašomas įterpiamų duomenų neužimtoje lauko dalyje, ir ženklas: + arba –. Abu šie elementai neprivalomi. Kai įterpimo žymėje nenurodomas užpildo simbolis, užpildui naudojamas tarpo simbolis. Ženklo simbolio prasmė priklauso nuo įterpiamame lauke rašomos reikšmės tipo. Jei ši reikšmė yra skaičius, simbolis + nurodo, kad ženklas reikalingas net ir tada, kai reikšmė teigiama. Simbolių eilutėms ženklas + nurodo, kad reikšmės tekstas turi būti rašomas kairiajame lauko krašte. Tai taip pat yra ir tipinis simbolių eilučių lygiavimo būdas, parenkamas tada, kai ženklo simbolio žymėje nebūna. Ženklo simbolis – nurodo, kad eilutės reikšmė rašoma dešiniajame lauko krašte.

Teksto ir sveikųjų skaičių reikšmėms skiriamo lauko dydis nurodomas vienu skaičiumi, kuris nusako minimalų reikšmėms skiriamo lauko dydį. Jeigu laukas per mažas, jis automatiškai padidinamas taip, kad įterpiama reikšmė tilptų, o jeigu per didelis, likusioje dalyje rašomi užpildo simboliai. Realiesiems skaičiams skiriamo lauko dydį aprašo du tašku atskirti skaičiai: pirmasis nurodo viso lauko dydį, o antrasis – trupmeninei daliai skiriamų simbolių skaičių.

Dažniausiai naudojami įterpiamų reikšmių tipų žymėjimo simboliai aprašyti 2.4.1 lentelėje. Jų taikymo pavyzdžiai:

```
printf('Kodas: %3d.\ 5); // Reikšmė: Kodas 005
$metai= 2004; $m= 9; $d= 30; // Kintamųjų reikšmių aprašymas
printf('%04d-%02d-%02y', $metai, $m, $d); //Reikšmė: 2004-09-30
printf('Išlaidos: %5.2f Lt.', 9.2); // Išlaidos: 9.20 Lt.
```

2.4.1 lentelė. Įterpiamų reikšmių tipų žymėjimo simboliai

Simbolis	Žymimo reikšmės tipo savybės
<i>c</i>	Simbolis, kurio kodą nurodo įterpiama reikšmė
<i>d</i> arba <i>i</i>	Sveikasis skaičius
<i>e</i> arba <i>f</i>	Realaus tipo skaičius su trupmenine dalimi
<i>s</i>	Simbolių eilutė

Funkcija *print\_r* naudojama pateikiant naršyklės ekrane sudėtingos struktūros duomenis. Jos argumentai gali būti ne tik skaičių ir teksto reikšmės, bet ir sudėtingesnės duomenų struktūros – masyvai bei objektai. Todėl ši funkcija taip pat naudojama ir programoms derinti, tarpiniams jų darbo rezultatams parodyti.

## 2.5. PHP operatoriai

Priskyrimo, palyginimo ir loginiai operatoriai pateikti 2.5.1, 2.5.2, 2.5.3 lentelėse.

2.5.1 lentelė. Priskyrimo operatoriai

Operatorius	Aprašymas	Pavyzdys	Rezultatas
+	Sudėtis	x=2 x+2	4
-	Atimtis	x=2 5-x	3
*	Daugyba	x=4 x*5	20
/	Dalyba	15/5 5/2	3 2.5
%	Modulis (dalybos liekanos)	5%2 10%8 10%2	1 2 0
++	Padidinimas vienetu	x=5 x++	x=6
--	Pamažinimas vienetu	x=5 x--	x=4

2.5.2 lentelė. Lyginimo operatoriai

Operatorius	Aprašymas	Pavyzdys
==	Yra lygu	5==8 grąžina <i>false</i> (netiesa)
!=	Nėra lygu	5!=8 grąžina <i>true</i> (tiesa)

<code>==</code>	Yra lygu (tikrina ne tik reikšmes, bet ir kintamųjų tipus)	<code>\$a = '1';</code> // kadangi kabutėse, tai bus ne skaičius, o teksto tipo <code>\$b = 1;</code> // skaičiaus tipo kintamasis <code>\$a == \$b</code> grąžina <i>false</i> (netiesa)
<code>!=</code>	Nėra lygu (tikrina ne tik reikšmes, bet ir kintamųjų tipus)	<code>\$a = '1';</code> // kadangi kabutėse, tai bus ne skaičius, o teksto tipo <code>\$b = 1;</code> // skaičiaus tipo kintamasis <code>\$a != \$b</code> grąžina <i>true</i> (tiesa)
<code>&lt;&gt;</code>	Nėra lygu	<code>5 &lt;&gt; 8</code> grąžina <i>true</i> (tiesa)
<code>&gt;</code>	Yra daugiau už	<code>5 &gt; 8</code> grąžina <i>false</i> (netiesa)
<code>&lt;</code>	Yra mažiau už	<code>5 &lt; 8</code> grąžina <i>true</i> (tiesa)
<code>&gt;=</code>	Yra daugiau arba lygu už	<code>5 &gt;= 8</code> grąžina <i>false</i> (netiesa)
<code>&lt;=</code>	Yra mažiau arba lygu už	<code>5 &lt;= 8</code> grąžina <i>true</i> (tiesa)

2.5.3 lentelė. Loginiai operatoriai

Operatorius	Aprašymas	Pavyzdys
<b>&amp;&amp;</b> <i>And</i>	Ir	<code>x=6</code> <code>y=3</code> <code>(x &lt; 10 &amp;&amp; y &gt; 1)</code> grąžina <i>true</i> (tiesa)
<b>  </b> <i>Or</i>	Arba	<code>x=6</code> <code>y=3</code> <code>(x == 5    y == 5)</code> grąžina <i>false</i> (netiesa)
<b>!</b>	Ne	<code>x=6</code> <code>y=3</code> <code>!(x == y)</code> grąžina <i>true</i> (tiesa)

## 2.6. Sąlygos sakiniai

PHP kalboje naudojami sąlygos tikrinimo sakiniai:

- *If* sakiny – skirtas vykdyti kodą tik tada, kai sąlyga grąžina reikšmę *true*;
- *If... else* sakiny – skirtas naudoti, jei atitinka sąlygą, tada vykdyti vieną kodą, kitu atveju – kitą;
- *If... elseif... else* sakiny – skirtas naudoti, kai turimos kelios sąlygos, jei atitinka pirmą – vykdomas pirmas kodas, jei antrą – antras, kitu atveju *else* dalies kodas;
- *Switch* sakiny – panašus į *if... elseif... elseif... else...* ir skirtas pasirinkti vieną iš kelių kodo blokų.

Pvz. Bus spausdinama „Šiandien penktadienis!“, jei kodas bus vykdomas penktadienį:

```
<html>
<body>
<?php
    $d=date("D");
    if ($d == "Fri") {
        print "Šiandien penktadienis!";
    }
```

```

}
?>
</body>
</html>

```

Pvz. Jei kodas bus vykdomas penktadienį, parašys „Šiandien penktadienis!“, jei sekmadienį – „Šiandien sekmadienis!“, jei kitą dieną, parašys „Sveiki!“:

```

<html>
<body>
<?php
$d=date("D");
if ($d= "Fri") {
print "Šiandien penktadienis!";
} elseif ($d= "Sun") {
print "Šiandien sekmadienis!";
} else {
print "Sveiki!";
}
?>
</body>
</html>

```

## 2.7. PHP Swith ir ciklo sakiniai

Ciklas skirtas patikrinti daug sąlygų ir parinkti vieną iš kelių kodo blokų. Labai panaši struktūra į *if... elseif... elseif... else...* sąlygos sakinį.

*Switch* sakinio sintaksė:

```

switch (n) {
case 1:
    kodas, kurį vykdome, jei kintamasis n yra lygus 1;
    break;
case 2:
    kodas, kurį vykdome, jei kintamasis n yra lygus 2;
    break;
default:
    kodas, kurį vykdome, jei kintamasis n nėra lygus nei 1, nei 2;
}

```

Vietoje *n* rašomas kintamasis, jo reikšmė tikrinama su visomis reikšmėmis, kurios aprašytos su *case*. Po žodelio *case* yra rašoma galima kintamojo reikšmė, tada dvitaškis ir po dvitaškio kodas, kuris bus vykdomas. Kai norima, kad ciklas nebebūtų toliau vykdomas, parašoma *break*;

- *while* – ciklas, kuris kartoja kodą tol, kol jame parašyta sąlyga yra *true*;

- *do.. while* – ciklas, kuris vieną kartą įvykdo kodą, tada patikrina sąlygą, jei ji yra *true* – toliau veikia kaip paprastas *while*;

- *for* – ciklas, kuris įvykdo kodą nurodytą kiekį kartų;
- *switch* – ciklas, panašus į *if... elseif... elseif... else...*;
- *foreach* – ciklas, kuris vykdo kodą kiekvienam masyvo elementui.

*while* ciklo sintaksė:

```
While (sąlyga) {  
    Kodas, kuris bus vykdomas;  
}
```

*do.. while* ciklo sintaksė:

```
do {  
    kodas, kuris bus vykdomas;  
} while (sąlyga);
```

*For* ciklo sintaksė:

```
for (pradinė_reikšmė; sąlyga; padidinimas) {  
    kodas, kuris bus vykdomas  
}
```

*Pradinė\_reikšmė* – šioje vietoje dažnai nurodomas kintamasis ir jo pradinė reikšmė. Tai taip pat gali būti bet koks *PHP* kodas, kuris bus įvykdytas prieš pradedant ciklą. *Sąlyga* – šioje vietoje yra tikrinama, ar tęsti ciklą: jei joje įrašytas kodas grąžina reikšmę *true* (tiesa), tada cikle esantis kodas bus įvykdomas. Jei sąlyga grąžina reikšmę *false* (netiesa), tada ciklas yra nutraukiamas. *Padidinimas* – šioje vietoje dažniausiai yra padidinama pradinės reikšmės reikšmė. Tai taip pat gali būti bet koks *PHP* kodas, kuris bus vykdomas kiekvieno ciklo pabaigoje.

**P a s t a b a.** Kiekvienas parametras gali būti tuščias, tačiau visada turi būti du kabliataškiai, kurie atskiria pradinę reikšmę nuo sąlygos ir padidinimo. Tuščias ciklas *for( ; ; )* yra begalinis.

*foreach* ciklo sintaksė:

```
foreach ( $masyvas as $reiksme ) {  
    kodas, kuris bus vykdomas  
}
```

Ciklo metu kiekvieną elementą galima pasiekti per kintamąjį *\$reiksme*.

## 2.8. PHP funkcijos

*PHP* kalba turi labai daug bazinių funkcijų:

- masyvo funkcijos;

- kalendoriaus funkcijos;
- datos funkcijos;
- katalogų / aplankų funkcijos;
- klaidų funkcijos;
- failų sistemų funkcijos;
- filtrų funkcijos;
- *FTP* funkcijos;
- *HTTP* funkcijos;
- *MAIL* funkcijos;
- matematinės funkcijos;
- *MySQL* funkcijos;
- *String* / eilučių formatavimo funkcijos;
- archyvų / *zip* funkcijos.

Vartotojas gali kurti savo funkcijas. Funkcija turi turėti pavadinimą, kuris negali sutapti su jau rezervuotais *PHP* funkcijų pavadinimais ir pačių sukurtų kitų funkcijų pavadinimais. Funkcijos pavadinimas negali prasidėti skaičiumi. Funkcijų vardų užrašymas galimas didžiosiomis ir mažosiomis raidėmis, taip pat pabraukimu (*\_\_funkcijosVardas()*). Funkcija gali (bet neprivalo) turėti parametrus, kurie rašomi skliausteliuose po pavadinimo ir išskiriami kableliu. Jeigu parametrų nėra perduodama, rašomi tiesiog tušti skliausteliai ( ). Funkcijos kodas rašomas laužtiniuose skliaustuose { ir }.

Funkcijos sintaksė:

```
function funkcijosVardas()
{
    kodas, kuris bus vykdomas;
}
```

Pvz.

```
<?php
    function manoAdresas(){
        echo "Miestas, Gatvės 45 g.";
    }
    echo "Labas, mano adresas yra : "
    manoAdresas();
?>
```

Pvz. Funkcija su parametrais:

```
<?php
    function manoAdresas($butoNr){
        echo "Miestas, Gatvės 45 g. Buto numeris : ". $butoNr;
    }
    echo "Labas, mano adresas yra : ";
```



```
manoAdresas("24");
```

```
?>
```

Pvz. Funkcija su parametrais:

```
<?php
```

```
function manoAdresas($butoNr, $symbolis){  
    echo "Miestas, Gatvės 45 g. Buto numeris : " . $butoNr . $symbolis . "<br />";  
}  
echo "Labas, mano adresas yra : ";  
manoAdresas("24", ".");  
echo "Labas, mano adresas yra : ";  
manoAdresas("24", "?");  
echo "Labas, mano adresas yra : ";  
manoAdresas("24", "!");  
echo "Labas, mano adresas yra : ";  
manoAdresas("24", "!!!");
```

```
?>
```

Pvz. Funkcija – kintamasis sugrąžinimas:

```
<?php
```

```
function skaiciuok($a, $b){  
    $suma = $a+$b; // pirmame kintamojo $suma apskaičiuoja  
    $suma = $a . "+" . $b . "=" . $suma; // antrame suformatuojame eilutę  
    // ir parodome koks buvo veiksmas  
    return $suma;  
}  
echo skaiciuok("5", "6");
```

```
?>
```

### 2.8.1. Eilučių tvarkymo funkcijos

Analizuojant iš formų gaunamus duomenis ir aprašant naršyklėms grąžinamų tinklalapių tekstą, tenka atlikti įvairius teksto tvarkymo veiksmus. Tokiems veiksmams aprašyti skirtas simbolių eilučių tvarkymo funkcijų rinkinys, aprašytas 2.8.1.1 lentelėje. Dažniausiai šios funkcijos naudojamos pertvarkant eilutėse esančius simbolius ir analizuojant jų struktūrą.

2.8.1.1 lentelė. Eilučių tvarkymo funkcijos

Kreipinys	Funkcijos grąžinama reikšmė
<i>strcasecmp(arg1, arg2)</i>	Analogiška funkcijai <i>strcmp</i> , tiksliai vienodai interpretuoja didžiąsias ir mažąsias raides
<i>strcmp(arg1, arg2)</i>	Teigiama, kai <i>arg1</i> > <i>arg2</i> , neigiama, kai <i>arg1</i> < <i>arg2</i> , ir 0 – kai šios eilutės vienodos
<i>string strtok(arg1, arg2)</i>	Eilutės <i>arg1</i> žodis (fragmentas), kurio pabaigą nurodo vienas iš eilutėje <i>arg2</i> išvardytų skirtukų (kartojant funkciją vien tiksliai su skirtukų eilute parametrų sąraše, iš anksčiau tvarkytos eilutės atskiriami nauji jos fragmentai)
<i>strlen(arg)</i>	Skaičius, kuris nurodo argumento eilutės <i>arg</i> simbolių skaičių

<i>strpos(arg1, arg2)</i>	Skaičius, kuris eilutės <i>arg2</i> pirmojo įterpimo eilutėje <i>arg1</i> vietos indeksą, arba reikšmę <i>false</i> , kai <i>arg2</i> nėra eilutės <i>arg1</i> dalis
<i>strrpos(arg1, arg2)</i>	Analogiška funkcijai <i>strpos</i> , tiksliai vienodai interpretuoja didžiąsias ir mažąsias raides
<i>str_replace(arg1, arg2, arg3)</i>	Pertvarkyta eilutė <i>arg3</i> , kurioje argumento <i>arg1</i> nurodomi fragmentai pakeičiami argumento <i>arg2</i> nurodomu fragmentu
<i>strstr(arg1, arg2)</i>	Eilutė <i>arg2</i> , jeigu ji yra eilutėje <i>arg1</i> , ir <i>false</i> , jeigu ten jos nėra
<i>substr(arg1, p, n)</i>	Ilgio <i>n</i> argumento eilutės <i>arg1</i> dalis, pradedant simboliu su indeksu <i>p</i> (kai neprivalomas argumentas <i>n</i> nenurodomas, atskiriamos dalies pabaiga sutampa su eilutės <i>arg1</i> pabaiga)
<i>substr_replace(arg1, arg2, p, n)</i>	Pertvarkyta eilutė <i>arg1</i> , kurioje <i>n</i> simbolių, pradedant simboliu su indeksu <i>p</i> , keičiami eilutės <i>arg2</i> simboliais (kai neprivalomas argumentas <i>n</i> nenurodomas, pakeitimas atliekamas iki eilutės <i>arg1</i> pabaigos)
<i>trim(arg)</i>	Pertvarkyta argumento eilutė, kurios pradžioje ir pabaigoje pašalinami tarpo simboliai
<i>strtoupper(arg)</i>	Pertvarkyta argumento eilutė, kurioje visos mažosios raidės pakeistos didžiosiomis
<i>strtolower(arg)</i>	Pertvarkyta argumento eilutė, kurioje visos didžiosios raidės pakeistos mažosiomis
<i>Stri_tags(\$input)</i>	Pertvarkyta argumento eilutė, kurioje pašalintos <i>HTML</i> kalbos žymės

Pavyzdžiui, interneto parduotuvių svetainėse reikia iš užsakymų formos laukelyje įvesto pageidaujamo prekių sąrašo atrinkti atskirus prekių pavadinimus. Tam naudojama funkcija *strtok*, kurios pirmasis argumentas nurodo analizuojamą eilutę, o antrasis – ją sudarančių elementų skirtukų sąrašą. Funkcijos reikšmė grąžina pirmąjį analizuojamos eilutės elementą. Jeigu pageidaujama išrinkti iš eilutės ir kitus elementus, reikia pakartotinai kreiptis į funkciją, nurodant tiksliai skirtukų rinkinį. Funkcija „atsimena“, kokia anksčiau tirtos eilutės dalis buvo apdorota ir tęsia jos likusios dalies analizę. Kai išrenkami visi tiriamos eilutės elementai, funkcija grąžina tuščios eilutės reikšmę. Atsižvelgiant į šią savybę, galima aprašyti visų jos elementų išrinkimą ciklu *while*.

Pvz.

```
$pavadinimas = strtok($eil, ",");
while ($pavadinimas != ""){
    echo $pavadinimas, "<br>";
    $pavadinimas = strtok(",");
}
```

Pavyzdyje aprašoma, kaip iš eilutės *\$eil* išrenkami ir parodomi naršyklės ekrane kableliais atskirti jos elementai. Atskiriamo elemento reikšmė perduodama kintamajam *\$pavadinimas* ir, kol tai nėra tuščia eilutė, operatoriumi *echo* perduodama į naršyklės ekraną. Aprašytas eilutės elementų išrinkimo būdas tinka tiksliai tada, kai tvarkomoje eilutėje nebūna greta įrašytų dviejų skirtukų, nes tada taip pat grąžinama tuščios eilutės reikšmė, kurios negalima atskirti nuo eilutės pabaigos požymio.

Renkant statistinius duomenis ir atliekant duomenų paiešką, dažnai tenka tikrinti, ar simbolių eilutėse yra tam tikras žodis arba kuris nors kitas požymis. Taip tikrinti lengviausia naudojant funkciją *strstr*, kuri grąžina reikšmę *false*, kai antrojo argumento eilutė nėra pirmojo argumento eilutės dalis, o kitu atveju grąžina antrojo argumento reikšmę. Šios funkcijos naudojimo pavyzdys:

```
$fp = "Geras miestas Kaunas";
$tempo = strstr($fp, "Kaunas");
if ($tempo )
    echo "Eilutėje pavadinimas ", $tempo, "yra.";
else
    echo "Eilutėje pavadinimo ", $tempo, "nėra."
```

Vietoje funkcijos *strstr* galima naudoti funkciją *strpos*, kuri grąžina antrojo argumento pirmojo įterpimo pirmojo argumento eilutėje vietos indeksą arba loginę reikšmę *false*, kai antrojo argumento eilutė nėra pirmojo argumento dalis. Kai tikrinamas fragmentas yra analizuojamos eilutės pradžioje, funkcijos grąžinamas indeksas yra reikšmė 0, kuri interpretuojama kaip *false*. Todėl, naudojant *strpos* vietoj *strstr* funkcijos grąžinamai reikšmei patikrinti, reikia taikyti ne lygybės veiksmą *==*, o tapatumo kontrolės veiksmą, kuris žymimas trijų simbolių grupe *===*. Šis veiksmas leidžia atskirti sveikąją reikšmę 0 nuo loginės reikšmės *false*, nes jos yra skirtingų tipų. Pvz.

```
$vardas = "Vytautas Didysis";           // Tiriama eilutė
$rezultatas = strpos($vardas, "V"); // Raidės "V" indeksas eilutėje
if ($rezultatas == false                ) // Tapatumo kontrolė
    echo "Raidės V varde ", $vardas, " nėra.";
else
    echo "Raidės indeksas: ", $rezultatas;
```

Naudojant funkciją *strpos* kartu su eilučių fragmentų atskyrimo funkcija *substr* ir kitomis eilučių tvarkymo funkcijomis, galima aprašyti sudėtingus eilučių analizės uždavinius. Pavyzdžiui, konkretaus žodžio pasikartojimų skaičiaus eilutėje skaičiavimas aprašomas taip:

```
$sk = 0;                               // Žodžio pasikartojimų skaitiklis
$fp = "Kaunas, Kaunas ir tikrai Kaunas"; // Tiriama eilutė
$tempo = strpos($fp, "Kaunas")          // Žodžio vietos indeksas
while (!( $tempo === false )) {          // Ar žodis eilutėje yra?
    $fp = substr($fp, ($tempo+6));        // Iširtos dalies atmetimas
    $tempo = strpos($fp, "Kaunas");       // Žodžio vieta likusioje eilutės dalyje
    $sk ++;                               // Žodžio pasikartojimų skaičiavimas
}
echo "$sk"                               // Pasikartojimų skaičius
```

## 2.8.2. PHP datos ir laiko funkcijos

Darbai su data PHP turi savo funkciją *date* (formatas). Funkcija naudojama laiko ir datos parodymui įvairiais formatais.

Formatai:

Y – metai, pvz.: 2012

y – metai, pvz.: 12

M – mėnuo, pvz.: Bal

m – mėnuo, pvz.: 10

D – savaitės diena, pvz.: Penkt

l – savaitės diena, pvz.: Penktadienis

d – diena, pvz.: 27

z – metų diena, pvz.: 299

H – valandos 24 val. formatu

h – valandos 12 val. formatu

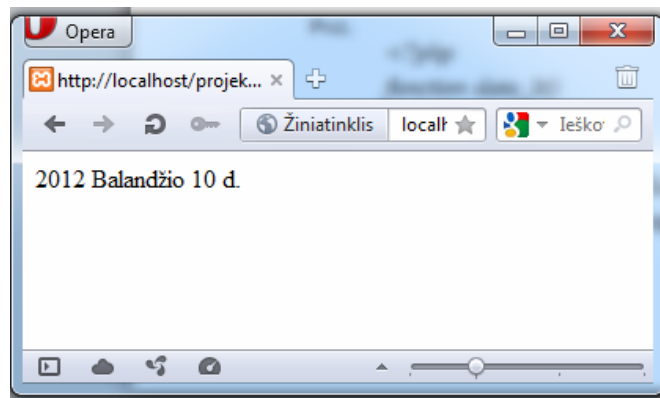
i – minutes, pvz.: 5

s – sekundes, pvz.: 40

a – am/pm

Pavyzdyje pateiktas kodas, kuris į interneto naršyklę perduos einamąją datą (2.8.2.1 pav.):

```
<?php
function data_lt()
{
    $menesis = date('n');
    $mas_menesiai = array("Sausio", "Vasario", "Kovo", "Balandžio", "Gegužės",
"Birželio",
"Liepos", "Rugpjūčio", "Rugsėjo", "Spalio", "Lapkričio", "Gruodžio");
    $data = date('Y');
    $data .= $mas_menesiai[$menesis-1];
    $data .= date('j \d. ');
    return $data;
}
echo data_lt();
?>
```



2.8.2.1 pav. Vaizdas naršyklėje

### 2.8.3. PHP masyvų funkcijos (array functions)

PHP kalboje yra nemažai funkcijų, skirtų masyvams kurti, ir jos dažnai smarkiai palengvina kodo rašymą. Masyvams skirtos funkcijos aprašytos 2.8.3.1 lentelėje.

2.8.3.1 lentelė. Masyvų funkcijos

Funkcija	Aprašymas
<code>array()</code>	Sukuria naują masyvą
<code>array_change_key_case()</code>	Grąžina tą patį masyvą, tik visas reikšmes pertvarko didžiosiomis arba mažosiomis raidėmis
<code>array_chunk()</code>	Išskaido masyvą į gabalus
<code>array_combine()</code>	Sukuria naują masyvą, naudojant vieną masyvą raktams (indeksams), kitą – reikšmėms
<code>array_count_values()</code>	Grąžina masyvą su kiekvienos reikšmės buvimo vieta
<code>array_diff()</code>	Palygina dviejų masyvų reikšmes ir grąžina skirtumus
<code>array_diff_assoc()</code>	Palygina dviejų masyvų raktus (indeksus) ir reikšmes, grąžina skirtumus
<code>array_diff_key()</code>	Palygina dviejų masyvų raktus (indeksus) ir grąžina skirtumus
<code>array_diff_uassoc()</code>	Palygina dviejų masyvų raktus (indeksus) ir reikšmes pagal vartotojo pateiktą funkciją, grąžina skirtumus
<code>array_diff_ukey()</code>	Palygina dviejų masyvų raktus (indeksus) pagal vartotojo pateiktą funkciją ir grąžina skirtumus
<code>array_fill()</code>	Užpildo masyvą reikšmėmis
<code>array_filter()</code>	Filtruoja masyvo elementus pagal vartotojo pateiktą funkciją
<code>array_flip()</code>	Apkeičia vietomis masyvo raktus (indeksus) su jų reikšmėmis. Reikšmės tampa indeksais ir atvirkščiai
<code>array_intersect()</code>	Palygina dviejų masyvų reikšmes ir grąžina sutapimus
<code>array_intersect_assoc()</code>	Palygina dviejų masyvų raktus (indeksus) ir reikšmes, grąžina sutapimus
<code>array_intersect_key()</code>	Palygina dviejų masyvų raktus (indeksus) ir grąžina sutapimus
<code>array_intersect_uassoc()</code>	Palygina dviejų masyvų raktus (indeksus) ir reikšmes pagal vartotojo pateiktą funkciją, grąžina sutapimus
<code>array_intersect_ukey()</code>	Palygina dviejų masyvų raktus (indeksus) pagal vartotojo pateiktą funkciją ir grąžina sutapimus
<code>array_key_exists()</code>	Patikrina, ar masyve yra toks raktas (indeksas)

<b><i>array_keys()</i></b>	Grąžina visus masyvo raktus (indeksus)
<b><i>array_map()</i></b>	Siunčia visas reikšmes vieną po kitos į vartotojo pateiktą funkciją, kuri grąžina naujas reikšmes
<b><i>array_merge()</i></b>	Sulieja keletą masyvų į vieną
<b><i>array_merge_recursive()</i></b>	Sulieja keletą masyvų į vieną
<b><i>array_multisort()</i></b>	Rūšiuoja kelių dimensijų masyvus
<b><i>array_pad()</i></b>	Įterpia į masyvą nurodytą skaičių elementų su nurodytomis reikšmėmis
<b><i>array_pop()</i></b>	Ištrina paskutinį masyvo elementą
<b><i>array_product()</i></b>	Apskaičiuoja masyvo reikšmių sandaugą
<b><i>array_push()</i></b>	Prideda vieną ar kelias reikšmes prie masyvo pabaigos
<b><i>array_rand()</i></b>	Grąžina vieną ar kelis atsitiktinius raktus (indeksus) iš masyvo
<b><i>array_reduce()</i></b>	Grąžina masyvą kaip <i>string</i> tipo kintamąjį, pagal vartotojo pateiktą funkciją
<b><i>array_reverse()</i></b>	Grąžina tą patį masyvą su atvirkščia tvarka išdėstytais jo elementais
<b><i>array_search()</i></b>	Ieško masyve nurodyto rakto (indekso) ir grąžina jo reikšmę
<b><i>array_shift()</i></b>	Panaikina masyvo pirmąjį elementą ir grąžina jo reikšmę
<b><i>array_slice()</i></b>	Grąžina masyvą pasirinktomis dalimis („supjausto“ jį)
<b><i>array_splice()</i></b>	Panaikina ir pakeičia naujais nurodytus masyvo elementus
<b><i>array_sum()</i></b>	Grąžina masyvo reikšmių sumą
<b><i>array_udiff()</i></b>	Palygina masyvo reikšmes pagal vartotojo pateiktą funkciją ir grąžina masyvą su rezultatais
<b><i>array_udiff_assoc()</i></b>	Palygina masyvo raktus (indeksus) ir reikšmes pagal vartotojo pateiktą funkciją ir grąžina masyvą su skirtumais
<b><i>array_udiff_uassoc()</i></b>	Palygina masyvo raktus (indeksus) ir reikšmes pagal vartotojo pateiktas funkcijas (viena reikšmėms tikrinti, kita raktams) ir grąžina masyvą su skirtumais
<b><i>array_uintersect()</i></b>	Palygina masyvo reikšmes pagal vartotojo pateiktą funkciją ir grąžina masyvą su sutapimais
<b><i>array_uintersect_assoc()</i></b>	Palygina masyvo raktus (indeksus) ir reikšmes pagal vartotojo pateiktą funkciją ir grąžina masyvą su sutapimais
<b><i>array_uintersect_uassoc()</i></b>	Palygina masyvo raktus (indeksus) ir reikšmes pagal vartotojo pateiktas funkcijas (viena reikšmių tikrinimui, kita raktams) ir grąžina masyvą su sutapimais

<b><i>array_unique()</i></b>	Panaikina pasikartojančias reikšmes turinčius masyvo elementus iš masyvo
<b><i>array_unshift()</i></b>	Prideda vieną ar kelis elementus į masyvo pradžią
<b><i>array_values()</i></b>	Grąžina visas masyvo reikšmes
<b><i>array_walk()</i></b>	Pritaiko vartotojo pateiktą funkciją kiekvienam masyvo elementui (vienmačiam masyvui)
<b><i>array_walk_recursive()</i></b>	Pritaiko vartotojo pateiktą funkciją kiekvienam masyvo elementui (keliamajam masyvui)
<b><i>arsort()</i></b>	Pertvarko masyvo elementus pagal jų reikšmes (pagal abėcėlę nuo Z iki A ). Indeksai išlieka šalia reikšmių tokie pat, kaip buvę
<b><i>asort()</i></b>	Pertvarko masyvo elementus pagal jų reikšmes (pagal abėcėlę nuo A iki Z ). Indeksai išlieka šalia reikšmių tokie pat, kaip buvę
<b><i>compact()</i></b>	Sukuria masyvą, kuris kintamųjų pavadinimus priskiria raktams (indeksams), o jų reikšmes – masyvo reikšmėms
<b><i>count()</i></b>	Suskaičiuoja, kiek elementų turi masyvas, grąžina skaičių. Taip pat galima naudoti su objektais, grąžina objekto parametrų skaičių
<b><i>current()</i></b>	Grąžina dabartinio masyvo elemento reikšmę
<b><i>each()</i></b>	Grąžina dabartinio masyvo elemento raktą (indeksą) ir reikšmę iš masyvo. Rodyklę ( <i>pointer</i> ) perkelia į kitą elementą
<b><i>end()</i></b>	Perkelia masyve esančią rodyklę ( <i>pointer</i> ) į paskutinį elementą
<b><i>extract()</i></b>	Įkelia kintamuosius iš masyvo į veikiamąją simbolių lentelę
<b><i>in_array()</i></b>	Patikrina, ar nurodyta reikšmė yra masyve
<b><i>key()</i></b>	Suranda rakto (indekso) vietą masyve
<b><i>krsort()</i></b>	Išrikiuoja masyvą pagal raktą (indeksą) atgaline tvarka
<b><i>ksort()</i></b>	Išrikiuoja masyvą pagal raktą (indeksą) normalia tvarka
<b><i>list()</i></b>	Priskiria kintamiesiems tokias reikšmes, kokios buvo masyvo elementų reikšmės
<b><i>natcasesort()</i></b>	Grąžina masyvą išrikiavus pagal reikšmes natūralia tvarka, neatsižvelgiant į didžiąsias ar mažąsias reikšmes. A yra tapatu a, kaip ir Z z ar B b t. t.
<b><i>natsort()</i></b>	Grąžina masyvą išrikiavus pateiktąjį pagal reikšmes natūralia tvarka
<b><i>next()</i></b>	Perkelia masyvo rodyklę ( <i>pointer</i> ) į kitą elementą
<b><i>pos()</i></b>	Tas pats kaip <i>current()</i>
<b><i>prev()</i></b>	Perkelia masyvo rodyklę ( <i>pointer</i> ) į prieš tai buvusį elementą

<b><i>range()</i></b>	Sukuria masyvą pagal intervalą ir jį užpildo reikšmėmis
<b><i>reset()</i></b>	Perkelia masyvo rodyklę ( <i>pointer</i> ) į pirmą masyvo reikšmę
<b><i>rsort()</i></b>	Rikiuoja masyvą atbuline tvarka, raktų (indeksų) tvarka išlieka ta pati (atsiriša nuo perrikiuotų reikšmių)
<b><i>shuffle()</i></b>	Sumaišo masyvo elementus
<b><i>sizeof()</i></b>	Tas pats kaip <i>count()</i>
<b><i>sort()</i></b>	Išrikiuoja masyvą, indeksų tvarka išlieka tokia kaip buvo (atsiriša nuo perrikiuotų reikšmių)
<b><i>uasort()</i></b>	Išrikiuoja masyvą pagal reikšmes ir vartotojo funkciją, raktai (indeksai) lieka pririšti prie reikšmių
<b><i>uksort()</i></b>	Išrikiuoja masyvą pagal raktus (indeksus) ir vartotojo funkciją, raktai (indeksai) lieka pririšti prie reikšmių
<b><i>usort()</i></b>	Išrikiuoja masyvą pagal reikšmes ir vartotojo funkciją, raktai (indeksai) atsiriša nuo reikšmių

## 2.9. Specialieji PHP masyvai

`$_GET` *GET* būdas, t. y kartu su adresu perduodamų kintamųjų masyvas.

`$_POST` *POST* būdas, t. y *HTTP* užklauso duomenyse perduodamų kintamųjų masyvas.

`$_SERVER` – serverio aplinkos kintamųjų masyvas.

`$_SERVER['PHP_SELF']` – puslapio *URL* adresas.

`$_SERVER['PATH']` – kelias iki vykdomųjų komandų.

`$_ENV` *OS* – aplinkos kintamųjų masyvas.

`$_SESSION` – sesijos klientas-serveris kintamųjų masyvas, kuris saugomas serveryje.

`$_COOKIES` – sesijos klientas-serveris kintamųjų masyvas, kuris saugomas vartotojo kompiuteryje. Gali būti panaudotas kitam klientas-serveris seansui, tai svetainei ar tam puslapiui.

### 2.9.1. `$_GET` masyvas

Šį tipą galima naudoti tada, kai reikia, kad duomenys būtų matomi naršyklės adreso eilutėje. Tai tikslinga paieškos formose, kai yra naudinga paieškos kriterijus matyti naršyklės adreso eilutėje, tada vartotojui paprasta pasidalinti paieškos rezultatais su kitais. Naudojant *get* metodą, reikia nepamiršti, kad adreso ilgis neturi viršyti 2083 simbolių.

Pvz. *HTML* forma su atributu *method* ir reikšme *get*:

```
<form action="html_formos_ivedimas_pvz.php" method="get">
```

*Vardas:*



```
<input type="text">
<input type="submit" value="Siųsti">
</form>
```

PHP failas, kuriam bus pateikta informacija:

```
Jūsų įvestas vardas:<br />
<?php
print($_GET['vardas']);
?>
```

Pateikus formą, naršyklėje bus matomas (jei į įvedimo lauką įvesta: Lukas):

```
Jūsų įvestas vardas:
Lukas
```

Puslapio adresas, rodomas naršyklės adreso lauke, bus panašus į tokį:

```
http://kazkas.lt/html_formos_ivedimas_pvz.php?vardas=lukas
```

### 2.9.2. PHP \$\_POST masyvas

Šiame masyve yra saugomos reikšmės, kurias pateikia vartotojas pasirinkęs formą, kurios atributas *method* turi reikšmę *post*. Informacija, kuri siunčiama *post* metodu, yra visiems matoma, tačiau jai pasiekti reikia specialių papildomų programų. Siunčiamos informacijos per *post* masyvą naršyklės adreso lauke visiškai nebūna matyti. Šis masyvas puikiai tinka slaptiems duomenims persiųsti – prisijungimo vardui, slaptažodžiui ir kt.

**P a s t a b a:** PHP serveryje yra failas *php.ini*. Jame dažniausiai būna PHP serverio nustatymai. Pagal nutylėjimą nustatymas *post\_max\_size* turi reikšmę 8 Mb. Tai reiškia, kad per *post* metodą vienu kartu galima persiųsti daugiausia 8 Mb duomenų.

Pvz. HTML forma su atributu *method* ir reikšme *post*:

```
<form action="html_formos_ivedimas_pvz.php" method="post">
  Vardas:
  <input name="vardas">
  <input type="submit" value="Siųsti">
</form>
```

PHP failas kuriam bus pateikta informacija:

```
Jūsų įvestas vardas:<br />
<?php
print($_POST['vardas']);
?>
```

Pateikus formą, naršyklėje bus matomas (jei į įvedimo lauką įvesta: Lukas):

```
Jūsų įvestas vardas:
Lukas
```

Puslapio adresas, rodomas naršyklės adreso lauke bus panašus į tokį:

### 2.9.3. PHP \$\_REQUEST masyvas

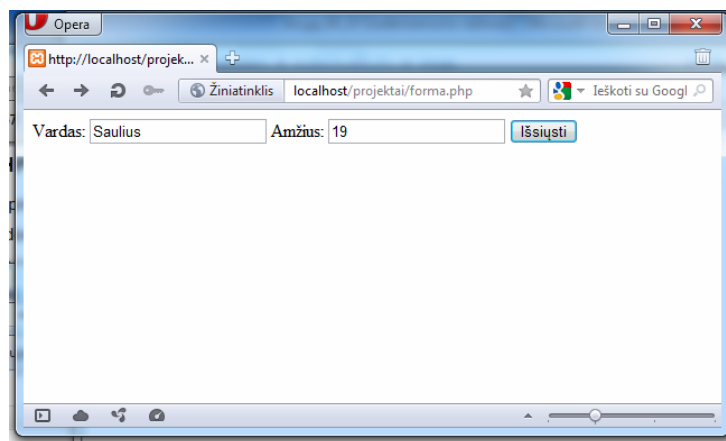
`$_REQUEST` masyvas – tai asociatyvus masyvas, kurio viduje yra `$_GET`, `$_POST` ir `$_COOKIE` masyvų reikšmės. Tai *superglobal* tipo masyvas, pasiekiamas bet kurioje kodo dalyje.

Šį masyvą patogiu naudoti tada, kai tiksliai negalima pasakyti, koku metodu bus gaunami duomenys: ar iš `$_GET`, `$_POST`, ar `$_COOKIE`.

P a s t a b a. Kadangi šis masyvas gauna reikšmes iš `$_GET`, `$_POST` ir `$_COOKIE`, jo reikšmes vartotojai gali pakeisti ir juo pasitikėti negalima. Prieš naudojant reikšmes iš šio masyvo, reikia patikrinti, kad atitiktų tai, ko tikimasi gauti. Pavyzdžiui, jei reikšmė yra iš *HTML* formos, kur vartotojas turėjo įvesti savo vardą, reikia patikrinti, ar tai, kas buvo įvesta, sudaryta tik iš raidžių.

Pvz. Tarkime, kad duomenims apie svetainės lankytojus rinkti naudojama tokia forma (žr. 2.9.3.1 pav.):

```
<html>
<body>
<form action="welcome.php" method="post">
  Vardas: <input type="text" name="fname" />
  Amžius: <input type="text" name="age" />
  <input type="submit" />
</form>
</body>
</html>
```



2.9.3.1 pav. Forma duomenims apie svetainės lankytojus rinkti

Interneto serveryje *PHP* intarpo formuojamas tinklalapio aprašymas, kuriame svetainės lankytojiui pranešama apie formoje įvestus duomenis, atrodo taip (žr. 2.9.3.2 pav.):

```
<html>
<body>
```

```

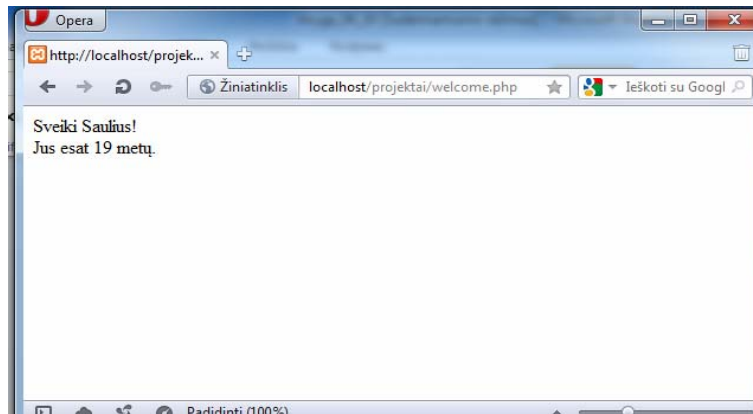
Sveiki <?php echo $_POST["fname"]; ?>!  

Jus esat <?php echo $_POST["age"]; ?> metų.  

</body>  

</html>

```



2.9.3.2 pav. Vaizdas naršyklėje

## 2.10. Formų duomenų patikrinimas

Prieš panaudojant *PHP* intarpuose naršyklėse įrašytus ir parinktus formų duomenis, juos reikia patikrinti. Tikrinant šias reikšmes, naudojami sąlyginio veiksmų parinkimo sakiniai *if*. *PHP* kalboje didesnė tikrinamų sąlygų aprašymo priemonių įvairovė: galima naudoti sąryšių operatorius, logines išraiškas ir *PHP* kalboje apibrėžtas logines funkcijas, kurios aprašytos 2.10.1 lentelėje. Tikrinant, ar formos įvedimo laukelyje buvo įrašyti duomenys, paprastai jo reikšmė lyginama su tuščia eilute, pavyzdžiui:

```

if ($_GET['vardas'] == ""){
    echo "<b>** Neįvedėte vardo! **</b><br>";
    $rezultatas = false; }

```

Tačiau toks tikrinimas ne visuomet geras, nes duomenimis laikoma ir vien iš tarpo simbolių sudaryta eilutė. To galima išvengti tikrinamą reikšmę apdorojant funkcija *trim*, kuri argumento eilutės pradžioje ir pabaigoje panaikina tarpus bei kitus specialius ekrane nerodomus simbolius:

```

if (trim($_GET['vardas']) == ""){
    echo "<b>** Neįvedėte vardo! **</b><br>";
    $rezultatas = false;
}

```

Kai žinomas minimalus įvedamos reikšmės simbolių skaičius, įvedimo kontrolei patogiau naudoti funkciją *strlen*:

```

if (strlen(trim($_GET['adresas'])) < 6){
    echo "Neteisingai įvestas elektroninio pašto adresas";
    exit; // PHP intarpo vykdymo nutraukimas
}

```

Loginės funkcijos, naudojamos kontrolei, pateiktos 2.10.1 lentelėje.

2.10.1 lentelė. Loginės funkcijos

Kreipinys	Reikšmės „true“ sąlyga
<i>is_numeric(arg)</i>	Argumento reikšmė yra skaičius
<i>is_real(arg)</i>	Argumento reikšmė yra sveikasis skaičius
<i>isinteger(arg)</i>	Argumento reikšmė yra realusis skaičius
<i>isset(arg)</i>	Argumento reikšmė yra apibrėžta

Funkcijos *is\_numeric*, *is\_real* ir *is\_integer* įvedimo kontrolei naudojamos tada, kai formos laukeliuose įvestas reikšmes numatoma naudoti skaičiavimuose. Kai formose duomenims įvesti naudojami žymimieji langeliai, į serverį siunčiamos visų pažymėtų langelių reikšmės, o kitų langelių reikšmės lieka neapibrėžtos. Neapibrėžtoms reikšmėms patikrinti naudojama loginė funkcija *isset*. Pavyzdžiui, tarkime, kad į serverį perduodami parodytos formos duomenys, kurios aprašymas toks:

```
<html>
<head></head><body>
<form name="Prekyba">
  <p>Vardas: <input type="text" name="vard" size="15">
  Pavardė: <input type="text" name="pav" size="20"></p>
  <p style="font-weight: bold">
    Pažymėkite pageidaujamas programas:</p>
  <input type="checkbox" name="win" value="ON">Windows XP
  <input type="checkbox" name="office" value="ON">MS Office XP
  <input type="checkbox" name="acrobat" value="ON">
    Adobe Acrobat 5.5 <br><br>
  <input type="submit" value="Užsakyti">&nbsp;
  <input type="reset" value="Trinti"/></form>
</body></html>
```

Formoje parengtas prekių užsakymas gali būti priimtas tikrai tada, kai pažymėtas bent vienas langelis *checkbox*:

```
if ((!isset($_GET['win'])>&& (! isset ($_GET ['office']) ) && (!isset($_GET['acrobat']
)))
  echo "Jūs turite pasirinkti bent vieną programą";
  $rezultatas = false;
}
```

Kitaip tikrinamas parinkimo akučių (*radio buttons*) žymėjimas, nes jų grupėje leidžiama pažymėti tikrai vieną akutę ir visos akutės turi tą patį vardą, pavyzdžiui:

```
<form name="Žemėlapiai">
  <p><b>Pasirinkite miestą: </b><br><br>
  <input type="radio" name="miestas" value="Kaunas">Kaunas
  <input type="radio" name="miestas " value="Klaipėda">Klaipėda
  <input type="radio" name="miestas " value="Vilnius">Vilnius<br><br>
```

</form>

Duomenis įvedus į tokią formą, turi būti kontroliuojama, kad būtų pažymėta bent viena akutė (apibrėžta kintamojo *\$miestas* reikšmė):

```
if (! isset($_GET['miestas'])) {  
    echo "<b>Jūs turite pasirinkti miestą</br><br>";  
    $rezultatas = false;  
}
```

## 2.11. Duomenų išsaugojimas serveryje

Kai programiniai intarpai naudojami tiktai formose įvestiems duomenims apdoroti ir gautiems rezultatams naršyklės lange parodyti, geriau naudoti naršyklės kompiuteryje vykdomų programų intarpus (skriptus). Tada nereikia duomenų siųsti į serverį ir atgal į naršyklę, taupomas brangus serverių ir ryšio kanalų darbo laikas. Serveriuose vykdomų programų privalumai išryškėja tada, kai reikia kaupti įvairiose naršyklėse įvestus duomenis, juos išsaugoti, kad būtų galima vėliau apdoroti arba papildyti iš išorinių šaltinių parinktais duomenimis.

Kaupiant duomenis diskinėje serverio atmintinėje ir parengiant juos vėliau apdoroti, naudojami du būdai: rašymas į tekstinius failus ir rašymas į duomenų bases. Pagrindinis yra antrasis būdas, kuris leidžia įrašytiems duomenims taikyti sudėtingas duomenų bazių analizės ir ataskaitų rengimo priemones, tačiau rašymas į tekstinius failus gerokai paprastesnis. Be to, tai universalus duomenų išsaugojimo būdas, nes tekstinius failus gali apdoroti dauguma taikomųjų programų. Taip pat nesunku parengti ir originalias tokių failų apdorojimo programas.

Prieš įvedant duomenis į failą, jį reikia atverti, parengti rašyti, o jeigu atveriamo failo paskirtame serverio kataloge nėra, jį reikia sukurti. Kiekvienam programoje tvarkomam failui sukuriamas specialus failo kintamasis, kurį su konkrečiu failu susieja funkcija *fopen*:

*Failo kintamasis = fopen("Failo vardas", "Atidarymo būdas");*

Funkcija *fopen* turi du privalomus parametrus. Pirmasis parametras nurodo paprastą arba išplėstą atveriamo failo vardą, o antrasis – jo atidarymo būdą. Paprastas failo vardas naudojamas tada, kai jis būna tame pačiame kataloge, kaip ir jį apdorojanti *PHP* programa. Realiose duomenų rinkimo interneto svetainėse rašyti duomenis į failą, kuris kartu su kitais interneto dokumentais yra atvirame kataloge, nepatartina, nes tokie failai internetu lengvai pasiekiami, todėl jų duomenys nėra saugūs. Geriau duomenis rašyti į specialų katalogą, kuris internetui nėra atviras. Tada naudojamas išplėstas failo vardas, kurio struktūra tokia:

*Kelias loginėje serverio failų struktūroje/Paprastas vardas*

Griežtas failo vietos nurodymas išplėstu vardu taip pat nėra geras, nes interneto serverių failų loginė struktūra laikas nuo laiko keičiama ir tokie pakeitimai su vartotojais nederinami. Todėl *PHP* kalboje numatytas specialus kintamasis *\$DOCUMENT\_ROOT*, kuriam suteikiama kelią į failams skirtą katalogą aprašanti reikšmė. Naudojant šį kintamąjį, failo vieta ir vardas nurodomi taip:

*\$DOCUMENT\_ROOT/./Pakatalogis/Failo vardas*

Kai formomis duomenys renkami iš daugelio svetainės lankytojų, failų rašymo veiksams naudoti būdu *w* negalima, nes tada kiekvienas lankytojas ištrins anksčiau ten buvusių duomenis. Geriau tuščius duomenų failus sukurti jau projektuojant svetainę, o *PHP* intarpuose juos atverti.

Pagrindinės skaitymo bei rašymo funkcijos: *fopen()*, *fgets()*, *fputs()*, *fclose()*, *feof()*; *fopen* – atidaro failą (skaitymo, rašymo arba papildymo veiksmas atlikti). Sintaksė:

*Failo kintamasis =fopen(failo\_pavadinimas, atidarymo būdas);*

Atidarymo būdai gali būti:

'r' – atidaro failą tik skaitymui, žymeklį pastato failo pradžioje;

'r+' – atidaro failą skirtą skaitymui ir rašymui, žymeklį pastato failo pradžioje;

'w' – atidaro failą rašymui, žymeklį pastato failo pradžioje, išvalo failo turinį (ištrina visus buvusių duomenis), o jei failas neegzistuoja, sukuria jį;

'w+' – atidaro failą skaitymui ir rašymui, žymeklį pastato failo pradžioje, failą išvalo, o jei failo nėra, jį sukuria;

'a' – atidaro failą tik rašymui, pastato žymeklį į failo pabaigą, jei failas neegzistuoja, sukuria jį;

'a+' – atidaro failą ir skaitymui ir rašymui, pastato žymeklį į failo pabaigą, jei failas neegzistuoja, sukuria jį.

Pvz.

```
$fp = fopen("/home/katalogas/file.txt", "r");  
$fp = fopen("http://www.mano.lt", "r");  
$fp = fopen("ftp://vartotojas:slaptazodis@mano.lt/", "w");
```

Atidarius failą, atlikus veiksmus, būtinai reikia jį uždaryti, pasinaudojus funkcija *fclose()*.

Pvz.

```
fclose($fp);  
fputs(); – įrašo duomenis į failą,  
fgets(); – nuskaityto nurodyto ilgio eilutę į string tipo kintamąjį,  
$string = fgets($fp,255); // nuskaityto visą eilutę
```

Kai vienu metu tą patį failą naudoja keli vartotojai, gali kilti konfliktinių situacijų. Kad tai išspręsti, galima naudoti funkciją *flock()*. Ši funkcija laikinai sustabdo antro vartotojo darbą ir

palaukia, kol pirmasis baigs dirbti su failu, tada antrajam vartotojui leidžiama pradėti dirbti. Patariama kuo dažniau naudoti šią funkciją dažnai lankomuose puslapiuose.

Pvz.

```
$failas = @fopen('skait.dat','r');  
flock = ($naujas_failas,2);  
$count = fgets($file, 255);  
$count++;  
flock = ($naujas_failas,3);  
fclose($file);
```

Tekstiniai failai – puiki duomenų bazių alternatyva pradedančiajam. Jam nebūtinai nei specialus palaikymas, nei labai daug *PHP* žinių.

Pvz. Informacijos perskaitymas iš tekstinio failo:

```
<?php  
$failas="duomenys.txt";  
$duomenys = fopen($failas, "r");  
$informacija = fread($duomenys, filesize($failas));  
fclose($duomenys);  
echo $informacija;  
?>
```

Pvz. Naujas įrašymas ištrinant visus senus duomenis:

```
<?php  
$informacija="Tekstas, įrašomas į failą";  
$failas="duomenys.txt";  
$duomenys=fopen($failas, "w");  
fwrite($duomenys, "$informacija");  
fclose($duomenys);  
?>
```

Pvz. Įrašymas neištrinant senų duomenų, o tęsiant sąrašą toliau:

```
<?php  
$informacija="Tekstas, įrašomas į failą";  
$failas="duomenys.txt";  
$duomenys=fopen($failas, "a");  
fwrite($duomenys, "$informacija\n");  
fclose($duomenys);  
?>
```

## 2.12. Formų apdorojimas – Formos atsakymo siuntimas el. paštu

### 2.12.1. Formos atsakymo siuntimas el. paštu

Dažnai, formą užpildžius vartotojui, atsakymą jam reikia išsiųsti elektroniniu laišku, arba vartotojo užpildytą formą reikia nusiųsti kam nors kitam. Elektroniniai laiškai *PHP* programavimo kalba siunčiami naudojantis funkcija *mail()*, kuri leidžia siųsti informaciją į el. paštą tiesiai iš kodo.

```
bool mail(string to, string subject, string message, [string additional_headers], [string additional_parameters]);
```

[ ] pažymėti argumentai nėra būtini.

Funkcija *mail()* grąžina *boolean* tipą (*true/false*). Jei *mail()* funkcija grąžina *false*, vadinasi el. laiško išsiųsti nepavyko. Paprasčiausias el. paštu siunčiamo laiško pavyzdys:

```
<?php
    mail(
        webmaster@php.lt Šis el.pašto adresas yra apsaugotas nuo Spam'o, jums reikia įjungti
        Javaskriptą, kad matytumėte tai
        , 'tema', 'laiško turinys');
?>
```

Pirmasis *mail()* argumentas – tai el. pašto adresas, kuriam siunčiamas el. laiškas, kitas argumentas – laiško tema, o trečiasis – turinys. Laiško turinyje, norint nukelti tekstą į naują eilutę, reikia įterpti „\n“.

Pvz.

```
<?
$tekstas = "Mano žinutė\nKita eilutė";
?>
```

Po laiško turinio gali eiti antraštės bei kiti papildomi parametrai, reikalingi el. laiškui išsiųsti.

### 2.12.2. Antraštės

Antraštėse galima nurodyti laiško siuntėją, laiško koduotę, *reply-to* lauką ir t. t. Laiško antraštės yra atskiriamos „\r\n“. Paprasčiausios antraštės, kuri turėtų būti kiekviename laiške, pavyzdys:

```
<?
$header = "Content-type: text/plain; charset=\r\n";
$header .= "From: Vardenis Pavardenis <mano@el_pastas.lt>\r\n";
$header .= "Reply-to: mano@el_pastas.lt\r\n";
mail(
```



```
webmaster@php.lt Šis el.pašto adresas yra apsaugotas nuo Spam'o, jums reikia įjungti
JavaScript, kad matytumėte tai
',tema','laiško turinys',$header);
?>
```

*Content type* nurodoma, kad žinutė bus išsiųsta naudojantis *plain* tekstu (ne *HTML*) bei *windows-1257* koduote. Išsiuntus laišką su tokia antrašte, vartotojai gaus el. laišką, kuriame visos raidės bus lietuviškos.

El. laiškai gali būti siunčiami ir *HTML* formatu, tada antraštė atrodytų taip:

```
$header = "Content-type: text/html; charset=\"windows-1257\"\\r\\n";
```

Dabar el. žinutėje galima naudoti ir *HTML* žymes (pastaba: ne visos pašto dėžutės turi galimybę rodyti el. laiškus *HTML* formatu).

*From* antraštėje nurodoma, nuo ko ir kam siunčiamas laiškas. Siuntėjas/gavėjas gali būti nurodomas tokiais formatais:

```
<?
/*
* From:
mano@pastas.lt Šis el.pašto adresas yra apsaugotas nuo Spam'o, jums reikia įjungti
JavaScript, kad matytumėte tai
```

```
* From:
mano@pastas.lt Šis el.pašto adresas yra apsaugotas nuo Spam'o, jums reikia įjungti
JavaScript, kad matytumėte tai
(Vardenis Pavardenis)
```

```
* From: Vardenis Pavardenis <
mano@pastas.lt Šis el.pašto adresas yra apsaugotas nuo Spam'o, jums reikia įjungti
JavaScript, kad matytumėte tai
>
```

```
* From: "Vardenis Pavardenis" <
mano@pastas.lt Šis el.pašto adresas yra apsaugotas nuo Spam'o, jums reikia įjungti
JavaScript, kad matytumėte tai
>
*/
?>
```

*Reply-to* reikalingas tam, kad būtų nurodyta, kam siųsti atsakymą į išsiųstą el. laišką. Pagal nutylėjimą, jei nėra *reply-to*, atsakymas bus siunčiamas asmeniui, kuris atsiuntė šį laišką.

Kelios papildomos antraštės, kurios gali praversti:

```
$headers .= "X-Priority: 3\\r\\n";
$headers .= "Return-Path: <
```

*mail@server.com Šis el.pašto adresas yra apsaugotas nuo Spam'o, jums reikia įjungti JavaScript, kad matytumėte tai* r\n";

*X-Priority* – dažnai pastebima, kad būna žinutės su šauktuku šone, ten žymimi prioritetai. Jei bus uždėtas prioritetas 1, tai reiškia, kad žinutė yra aukščiausio prioriteto, ir prie el. žinutės gavėjo el. pašto programoje atsiras raudonas šauktukas. Dažniausiai visi siunčia el. žinutes su 3 (normaliu) prioritetu. *Return-path* – tai el. pašto adresas, į kurį bus nukreipiami laiškai, įvykus klaidoms (pvz.: el. laiškas nepasiekė reikiamo serverio, arba tokio el. pašto adreso, adresas jau nebeegzistuoja sistemoje).

Kaip nusisiųsti el. žinutės kopiją sau? Yra trys galimybės:

1. Pats paprasčiausias – tai nurodyti "to" lauką štai taip:

```
<?
  $to = "Vardenis Pavardenis <
  vardenis@pastas.lt Šis el.pašto adresas yra apsaugotas nuo Spam'o, jums reikia įjungti
  JavaScript, kad matytumėte tai
  >" . ", ";
  $to .= "Mano Vardas <
  webmaster@php.lt Šis el.pašto adresas yra apsaugotas nuo Spam'o, jums reikia įjungti
  JavaScript, kad matytumėte tai
  >";
  mail($to,'Laiškas dviems gavėjams', 'Tekstas');
?>
```

2. Savo el.pašto adresą nurodyti "Cc" antraštės lauke:

```
<?
  $header .= "cc:
  webmaster@php.lt Šis el.pašto adresas yra apsaugotas nuo Spam'o, jums reikia įjungti
  JavaScript, kad matytumėte tai
  \r\n";
?>
```

Šiuo atveju visi gavėjai matys, kam buvo siunčiamas gautas el. laiškas.

3. Naudotis antraštės "Bcc" lauku:

```
<?
  $header .= "bcc:
  webmaster@php.lt Šis el.pašto adresas yra apsaugotas nuo Spam'o, jums reikia įjungti
  JavaScript, kad matytumėte tai
  \r\n";
?>
```

### 2.12.3. El. pašto adreso teisingumo tikrinimas

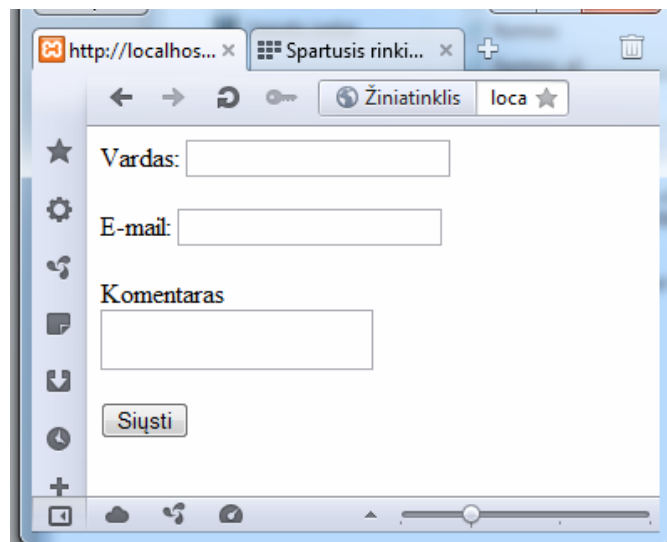
Dažnai formoje prašoma nurodyti, kam išsiųsti atsakomąjį laišką. Todėl svarbu, kad pildantis formą vartotojas teisingai nurodytų savo pašto adresą. Patikrinti, ar pašto adresas yra teisingas, neįmanoma. Tačiau, ar teisingu formatu įvestas, galima. El. pašto adreso formato tikrinimo pavyzdys:

```
if (eregi("[A-Z0-9._%~]+@[A-Z0-9._%~]+\.[A-Z]$", $pastas))  
    echo "'$pastas' yra teisingas!";  
else  
    echo "Klaida, '$pastas' yra neteisingai įvestas adresas";
```

### 2.12.4. El. laiško siuntimas, panaudojant formos duomenis

Norint išsiųsti laišką iš formos, pirmiausia reikia ją sukurti. Sukuriama forma, kurioje būtų įvedamas vardas, el. pašto adresas ir vartotojo pasirinkimas (žr. 2.12.4.1 pav.):

```
<form action="mail.php" method="post">  
    Vardas: <input type="text" name="vardas"><br>  
    E-mail: <input type="text" name="email"><br><br>  
    Komentaras<br>  
<textarea name="komentaras"></textarea><br><br>  
<input type="submit" value="Siųsti">  
</form>  
Rezultatas
```



2.12.4.1 pav. Formos vaizdas

Šiai formai sukuriamas *PHP* skriptas, kuris patikrina, ar teisingai įvestas el. pašto adresas, ir išsiunčia formos autoriui atsakomąjį laišką, kuriame yra komentaro autoriaus vardas, komentaras ir el. pašto adresas:

```
<?  
$vardas=$_POST['vardas'];
```

```

$email=$_POST['email'];
if (eregi("[A-Z0-9._%]+@[A-Z0-9._%]+\.[A-Z]$", $email)==false)
    echo "Klaida, 'Spastas' yra neteisingai įvestas adresas";
else
{
    $komentaras=$_POST['komentaras'];
    $to="
    formosautorius@tinklapis.lt Šis el.pašto adresas yra apsaugotas nuo Spam'o, jums reikia
    įjungti JavaScript, kad matytumėte tai
    "; // formos autoriaus el.pašto adresas
    $zinute="$vardas užpildė komentaro formą. Jis pasakė:\n$komentaras\n\nJo
    el.pašto adresas: $email";
    if(mail($to,"Komentaras",$zinute,"From: $email\n"))
        echo "Ačiū už komentarą.";
    else
        echo "Įvyko klaida, išsiunčiant laišką.";
    }
?>

```

#### 2.12.5. El. laiško siuntimo testavimas

Jei kuriamą formą, kurioje bus atsakymas, siunčiamas el. laiško forma, ir ji bus apdorojama *PHP* skripto, reikia testuoti Windows operacinėje sistemoje, *mail()* funkcija neveiks. Neveiks todėl, kad *php.ini* faile pagal nutylėjimą *SMTP* serveris nurodytas *localhost*, o didžiausia tikimybė yra ta, kad pas vartotoją *localhost SMTP* serverio nėra (nebent naudoja *Windows Server*), todėl ekrane bus matomas tik toks užrašas: *Warning: Failed to Connect in c:\kelias\failas.php on line XX*. Todėl *PHP* negali prisijungti prie *SMTP* serverio. To galima išvengti įrašius *php.ini* faile:

```
SMTP=jums_žinomas_smtp_serverio_adresas.
```

### 2.13. Formos duomenų įrašymas į failus

Dažnai norima formos rezultatus išsaugoti ar juos kaupti. Tam praverčia duomenų bazės, tačiau ne kiekvienas turi serverį su duomenų bazių palaikymu ar moka programuoti *SQL* kalba. Tokiu atveju tenka bandyti išsiversti failais. Paprastoms formoms failai gali pakeisti duomenų bazes. Pateikiama keletas pavyzdžių, kurie turėtų padėti dirbant su failais.

#### 2.13.1. Skaitymas ir rašymas į failus

Informacijos perskaitymas iš tekstinio failo. Pritaikymas: ištisinio teksto nuskaitymas:

```

<?
    $failas="duomenys.txt";
    $duomenys = fopen($failas, "r");
    $informacija = fread($duomenys, filesize($failas));

```

```

fclose($duomenys);
echo $informacija;
?>

```

Naujas įrašymas, ištrinant visus senus duomenis. Pritaikymas: paprasto puslapio skaitliuko skaičiaus įrašymas ir panašiai:

```

<?
$informacija="Tekstas, įrašomas į failą";
$failas="duomenys.txt";
$duomenys=fopen($failas, "w");
fwrite($duomenys, "$informacija");
fclose($duomenys);
?>

```

Įrašymas neištrinant senų duomenų, o tęsiant sąrašą toliau. Pritaikymas: įvairūs *log'ai* ir panašiai:

```

<?
$informacija="Tekstas, įrašomas į failą";
$failas="duomenys.txt";
$duomenys=fopen($failas, "a");
fwrite($duomenys, "$informacija\n");
fclose($duomenys);
?>

```

### 2.13.2. Tekstinė duomenų bazė

Informacijos eilutės perskaitymas iš tekstinio failo. Pritaikymas: objekto aprašymas, jei objektas turi daugiau nei vieną savybę. Pavyzdys:

*Mindaugas|20|185|70*

Pateikta eilutė apibūdina asmens vardą, amžių, ūgį ir svorį. Taigi *duomenys.txt* struktūra turi būti tokia: *tekstas0|tekstas1|tekstas2|tekstas3*. Kodo fragmento esmė tokia, kad yra perskaitoma eilutėje surašyta informacija ir išskirstoma į atskirus atitinkamus kintamuosius *\$laukelis[\$i]*. Šiuo būdu jau įmanoma sukurti panašią į duomenų bazės lentelę:

```

<?
$failas="duomenys.txt";
$simboliu_skaicius = count($failas);
$stulpelis = file($failas);
for($i=0; $simboliu_skaicius > $i; $i++)
    $laukelis = explode("|", $stulpelis[$i]);
echo "$laukelis[0] $laukelis[1] $laukelis[2] $laukelis[3]";
?>

```

Tik kintamasis *\$informacija* turėtų būti:

```

$informacija="Mindaugas|20|185|70";

```

Daug tokių tekstinių eilučių sudaro pilnos duomenų bazės struktūrą. Pavyzdžiui:

```
Mindaugas|20|185|70
Kristina|20|174|50
Rimantas|50|190|80
[..]
Vaida|19|171|53
```

Tokios duomenų bazės struktūros duomenų failo sukūrimas yra paprastas:

```
<?
    $vardas="vardas";
    $amzius="amzius";
    $ugis="ugis";
    $svoris="svoris";
    $informacija=$vardas."|".$amzius."|".$ugis."|".$svoris;
    $failas="duomenys.txt";
    $duomenys=fopen($failas, "a");
    fwrite($duomenys, "$informacija\n");
    fclose($duomenys);
?>
```

## 2.14. PHP slapukai („Cookies“)

Slapukai dažnai naudojami vartotojui identifikuoti. Slapukas – tai nedidelis failas, kurį serveris įdeda į vartotojo kompiuterį. Kiekvieną kartą, kai tas pats kompiuteris interneto naršyklės pagalba kreipiasi į tą patį puslapį, jis išsiunčia ir slapuką. Naudojantis *PHP*, galima atlikti šias operacijas su slapukais:

- slapuko kūrimas. Slapukui sukurti naudojama funkcija *setcookie()*;
- slapuko informacijos išgavimas. Užrašas *\$\_COOKIE* leidžia išgauti informaciją apie slapuką;
- slapuko trynimas. Slapukui ištrinti naudojama ta pati funkcija *setcookie()*.

## 2.15. PHP sesijos

Kompiuteryje galima nustatyti, kada konkretus vartotojas pradėjo darbą ir kada jį baigė. Serveris indentifikuoti vartotojo negali, kadangi *HTTP* protokolas nepalaiko būsenų. Šiai problemai spręsti ir skirtos *PHP* sesijos: jos leidžia išsaugoti vartotojo informaciją serveryje vėlesniam naudojimui. Sesijų informacija yra laikina ir yra sunaikinama tuomet, kai vartotojas palieka interneto svetainę. Sesijų veikimo principas paremtas unikalaus identifikacinio numerio sukūrimu ir suteikimu kiekvienam vartotojui. Tas numeris yra saugomas slapuke arba interneto adrese.

*PHP* sesijai inicializuoti naudojama *session\_start()* funkcija, kuri būtinai turi būti įterpiama dar prieš *html* elementą.

*PHP* sesijos kintamiesiems saugoti naudojamas *\$\_SESSION* kintamasis.

*PHP* sesijai sunaikinti naudojama funkcija *unset()*; norint visiškai sunaikinti sesiją, naudojama funkcija *session\_destroy()*.

### 3. JAVASCRIPT KALBOS PASKIRTIS

Internetiniams puslapiams kurti vartojant vien tik standartinę *HTML* kalbą, tinklalapiai yra statiniai – vartotojas negali turėti įtakos informacijos išdėstymui tinklalapyje. Norint sukurti tikrai interaktyvų tinklalapį, reikalinga dar viena kalba, kuri būtų vartojama naršyklės kontekste, tai intarpo (skripto) kalba.

Programavimo kalba *JavaScript* leidžia naudoti įvairius tekstinės ir grafinės informacijos vaizdavimo būdus, į tinklalapį įterpti įvairius objektus, suteikia dizaineriams daugiau galimybių kurti, neužkraunant serveriui atsakomybės už veiksmus.

Ši kalba skirta interaktyviems tinklalapiams kurti ir turi priemonių naršyklei valdyti. *JavaScript* turi palyginti mažai galimybių darbui su failais ar grafikai valdyti. Programos, sukurtos skripto kalba, negali veikti savarankiškai, jos vykdomos tik naršyklės, suprantančios šią kalbą, kontekste. Programos, sukurtos *JavaScript* kalba, dar vadinamos skriptais, arba scenarijais, ir įtraukiamos į *Web* tinklalapį, kuriame atpažįstamos ir vykdomos kartu su likusiu *HTML* kodu.

Naudojant skripto programas galima išgauti įvairius efektus, kuriuos riboja tik programavimo kalbos galimybės ir naršyklės elementai. Dažniausiai skripto kalbos vartojamos:

- įvairiems dialogo langams bei pranešimams naršyklės informacinėje eilutėje vaizduoti;
- dinamiškam tinklalapio turiniui kurti, kai tinklalapis skaitomas ar jau perskaitytas;
- tinklalapio turiniui keisti;
- formos elementuose vartotojo įrašytos informacijos teisingumui patikrinti;
- navigacijai po kitus puslapius;
- įterptiems *Java-appletams* ir *ActiveX* elementams valdyti;
- žaismingiems elementams kurti (skraidančios snaigės ar panašiai).

*JavaScript* kalboje visi tinklalapio elementai sudėti tam tikra hierarchine tvarka. Kiekvienas elementas yra tam tikras objektas. Kiekvienas objektas turi tam tikrus metodus ir savybes. Baziniai *JavaScript* žodžiai: *break false if null var continue for in true while else function new type of with*.

#### 3.1. Pagrindiniai JavaScript elementai

*JavaScript* kalbos elementų įterpimo taisyklės į *HTML* programą:

```
<script language="JavaScript">  
<!--  
//skripto programa  
//-->  
</script>
```



*JavaScript* kalboje vartojami du komentarų tipai. Pirmasis – vienos eilutės komentarai, prasidedantys simboliu “//”. Antrasis komentarų tipas – keletu eilučių komentaras. Prasideda simboliu “/\*” ir baigiasi simboliu “\*/”.

Kintamieji aprašomi, naudojant *var* išraišką. Jeigu kintamasis aprašytas funkcijoje, jis yra lokalus (reikšmė išlieka funkcijos ribose, išnyksta, kai pasibaigia funkcijos veikimas), jeigu už funkcijos ribų – globalus dokumentui (reikšmė išlieka dokumento ribose, išnyksta jį uždarius). Duomenų tipas priskiriamas automatiškai, priskyrus reikšmę.

Pranešimams spausdinti naudojamos komandos *alert* ir *document.write*.

Pvz.:

```
<script language="Java Script">
    alert('Informacijos išvedimo dialogo lange pavyzdys')
</script>
<script language="Java Script">
    document.write('Informacijos išvedimo naršyklės lange pavyzdys')
</script>
```

Informacijai įrašyti naudojami operatoriai: *alert* ir *prompt*.

Pvz.:

```
<script language="JavaScript">
    if (confirm('Ar norite peržiūrėti kitą puslapį ?'))
        document.write('Peržiūra')
    else
        document.write('Tas pats puslapis')
</script>
<script language='JavaScript'>
    var s
    s=prompt('Įveskite savo vardą', 'Jonas')
    document.write(s)
</script>
```

## 3.2. Operatoriai

Priskyrimo, palyginimo, matematiniai operatoriai ir atliekamos operacijos pateikti 3.2.1, 3.2.2, 3.2.3, 3.2.4, 3.2.5, 3.2.5 lentelėse.

3.2.1 lentelė. Priskyrimo operatoriai

Operatorius	Veiksmas	Pavyzdys
=	Priskiria kintamajam reikšmę	Kint=100
+=	Padidina kintamojo reikšmę nurodytu dydžiu	Kint+=10
-=	Sumažina kintamojo reikšmę nurodytu dydžiu	Kint-=10
*=	Padaugina kintamojo reikšmę iš nurodyto dydžio	Kint*=2
/=	Padaugina kintamojo reikšmę iš nurodyto dydžio	Kint/=5
%=	Padalija kintamojo reikšmę grąžina liekaną	Kint%=3

3.2.2 lentelė. Priskyrimo operandais vykdomos operacijos

Operatorius	Ekvivalenti operacija
$Kint += d$	$Kint = kint + d$
$Kint -= d$	$Kint = kint - d$
$Kint *= d$	$Kint = kint * d$
$Kint /= d$	$Kint = kint / d$
$Kint \% = d$	$Kint = kint \% d$

3.2.3 lentelė. Matematiniai operatoriai

Operatorius	Aprašymas	Pavyzdys
$-$ Reikšmė	Priešingas ženklas	$D = -10$
$Reikšmė1 + reikšmė2$	Sudėtis. Naudojamas ir eilutės jungti	$D = a + 100$ $Str = „abc“ + „def“$
$Reikšmė1 - reikšmė2$	Atimtis	$D = a - 100$
$Reikšmė1 * reikšmė2$	Daugyba	$D = 10 * 10$
$Reikšmė1 / reikšmė2$	Dalyba	$D = a / b$
$Reikšmė1 \% reikšmė2$	Dalyba	$D = a \% 3$
$Reikšmė++$	Kintamojo reikšmė didinama 1	$D++$
$Reikšmė--$	Kintamojo reikšmė mažinama 1	$D--$

3.2.4 lentelė. Palyginimo operatoriai

Operatorius	Operacija
$Dydis1 = dydis2$	Tikrina, ar lygūs operandai
$Dydis1 <> dydis2$	Tikrina, ar nelygūs
$Dydis1 > dydis2$	Tikrina, ar pirmas didesnis už antrą
$Dydis1 >= dydis2$	Tikrina, ar pirmas didesnis už antrą, arba lygūs
$Dydis1 < dydis2$	Tikrina, ar pirmas mažesnis už antrą
$Dydis1 <= dydis2$	Tikrina, ar pirmas mažesnis už antrą, arba lygūs
$Dydis1 \& \& dydis2$	Įvykdo loginę operaciją „ir“
$Dydis1    dydis2$	Įvykdo loginę operaciją „arba“
$!dydis$	Įvykdo loginę inversiją – „ne“

3.2.5 lentelė. Bitų operatoriai (naudojami tik su skaitiniais operandais ir operacijoms su atskirais bitais)

Operatorius	Pavadinimas
$\&$	Bitinė operacija „ir“
$ $	Bitinė operacija „arba“
$\wedge$	Bitinė operacija „ir/arba“
$\sim$	Bitinė operacija „ne“
$<<$	Postūmis į kairę
$>>$	Postūmis į dešinę

3.2.6 lentelė. Operatorių atlikimo eiliškumas

Eil. Nr.	Operatoriai
1	– (unarinis minusas), + (unarinis pliusas), ~ (bitinis NE), ! (loginis NE)
2	*(daugyba), / (dalyba), %
3	+(sudėtis), –(atimtis)
4	<< (postūmis į kairę), >>(postūmis į dešinę)
5	Visos lyginimo operacijos >, >=, <=, <, =,
6	& (bitinis „ir“)
7	^ (bitinis „ir/arba“)
8	(bitinis „arba“)
9	&& (loginis „ir“)
10	(loginis „arba“)
11	?:
12	Priskyrimo operatoriai
13	, (kablelis)

### 3.3. Sąlygos tikrinimo operatoriai

Operatorius *If* (*loginis\_sakinys*). Sakinio *loginis\_sakinys* rezultatas visada turi reikšmę *true* arba *false*. Jei blokuose *if* ir *else* tik po vieną sakinį, tai naudojama tokia konstrukcija:

*If* (*sąlyga*)

*Sakinys*;

*Else*

*Sakinys*;

Jei blokuose daugiau negu vienas sakiny, naudojami figūriniai skliaustai {}:

*If* (*sąlyga*) {

*1sakinys*;

*2sakinys*;

...}

*Else* {

*1sakinys*;

*2sakinys*;

}

Operatorius *Switch* naudojamas, kai sakiny gali įgyti keletą reikšmių. Priklausomai nuo to, atliekami tam tikri veiksmai. Jo sintaksė:

*Switch* (*sakinys*) {

*Case label1*:

{

//veiksmai atliekami tada, kai sakiny įgyja reikšmę, lygią *label1*

}

*case label2*:

{

//veiksmai atliekami tada, kai sakiny įgyja reikšmę, lygią *label2*

}

```
...
default:
{
//veiksmai atliekami tada, kai sakinys neįgyja nei vienos nurodytos reikšmės
}
}
```

Pvz.:

```
Switch (key) {
Case "a" :alert('Jūs paspaudėte klavišą a')
Case "b" :alert('Jūs paspaudėte klavišą b')
Default:alert('Jūs paspaudėte kažkokį klavišą')
}
```

### 3.4. Ciklo operatoriai

Operatorius *For* naudojamas, kai žinoma, kiek kartų reikės kartoti ciklą:

```
For (kint=prad_reiksm;sąlyga;ciklo_žingsnis)
{cikle vykdomų veiksmų seka...}
```

Pvz. Spausdins naršyklės savybes:

```
<html>
<head><title>Java Script pavyzdys</title>
<script language="JavaScript" >
function Test()
{for (i in navigator)
{document.write(i+"="+navigator[i]+"<BR>");}
}
</script>
</head>
<body>
<script language="JavaScript">
Test()
</script>
</body>
</html>
```

Operatorius *While* naudojamas, kai nežinoma, kiek kartų reikės kartoti ciklą:

```
While (sąlyga) {
ciklo sakiniai
}
```

Elementas *break* naudojamas kokių nors veiksmų nutraukimui. Dažniausiai naudojamas blokuose *for*, *for...in*, *while*, kai reikia nutraukti ciklo vykdymą, t. y. išeiti iš ciklo jo neįvykdžius iki pabaigos.

Elementas *Continue* taip pat naudojamas *cikluose for, for...in, while*. Šis elementas nurodo, kad iteracija stabdoma ir pereinama prie kitos iteracijos.

### 3.5. JavaScript objektai

*Objektas* – savybių, metodų, įvykių ir kolekcijų rinkinys, naudojamas objektiniame naršyklės modelyje. Į šiuos objektus kreipiamasi taip:

*objektas.metodas|savybė|kolekcija|savybė*

*Savybė* – kintamasis objekto ribose, kuris gali būti naudojamas kokioms nors reikšmėms gauti ar naujoms reikšmėms nustatyti. Dalis savybių naudojamos tik vieniems ar kitiems duomenims nuskaityti.

*Metodas* – procedūra ar funkcija, priklausanti objektui, ir skirta tam tikriems veiksmams vykdyti arba objektų savybėms valdyti.

*Įvykis* – koks nors vartotojo veiksmas arba naršyklės darbo momentas.

*Kolekcija* – savybių rinkinys, kuriame kintamieji išdėstyti panašia tvarka kaip masyve.

#### 3.5.1. Objektas Array

*JavaScript* kalboje nėra specialios komandų grupės, skirtos atlikti veiksmus su masyvais. Firma *Netscape* sukūrė masyvų naudojimo metodiką panaudodama objektą *array*. Jis turi masyvų sujungimo metodus, rūšiavimą ir perstatymą. Galima sužinoti masyvo dydį. Masyvas – kintamųjų rinkinys, kuriame svarbi elementų tvarka. Pvz., jei masyvo pavadinimas *sar*, tai pirmasis jo elementas bus iškviečiamas *sar[0]*, antrasis *sar[1]* ir t. t. Objekto *array* metodai pateikti 3.5.1.1 lentelėje.

Sukurti objektą *Array* galima dviem būdais:

- konstruktoriumi *new* sukuriamas masyvas, po to nurodomas jo dydis bei jį sudarantys elementai;
- kuriant masyvą galima iš karto aprašyti jo reikšmes.

3.5.1.1 lentelė. Objekto *Array* metodai

Metodas	Aprašymas
<i>Join</i>	Sujungia visus masyvo elementus į vieną eilutę
<i>Reverse</i>	Keičia masyvo elementų tvarką: pirmasis elementas tampa paskutiniu, o antrasis – priešpaskutiniu ir t. t.
<i>Sort</i>	Rūšiuoja masyvo elementus
<i>Concat</i>	Masyvų sujungimas
<i>Slice</i>	Masyvo elementų išrinkimas, nurodant sekos pradžios ir pabaigos simbolius

Pvz.:

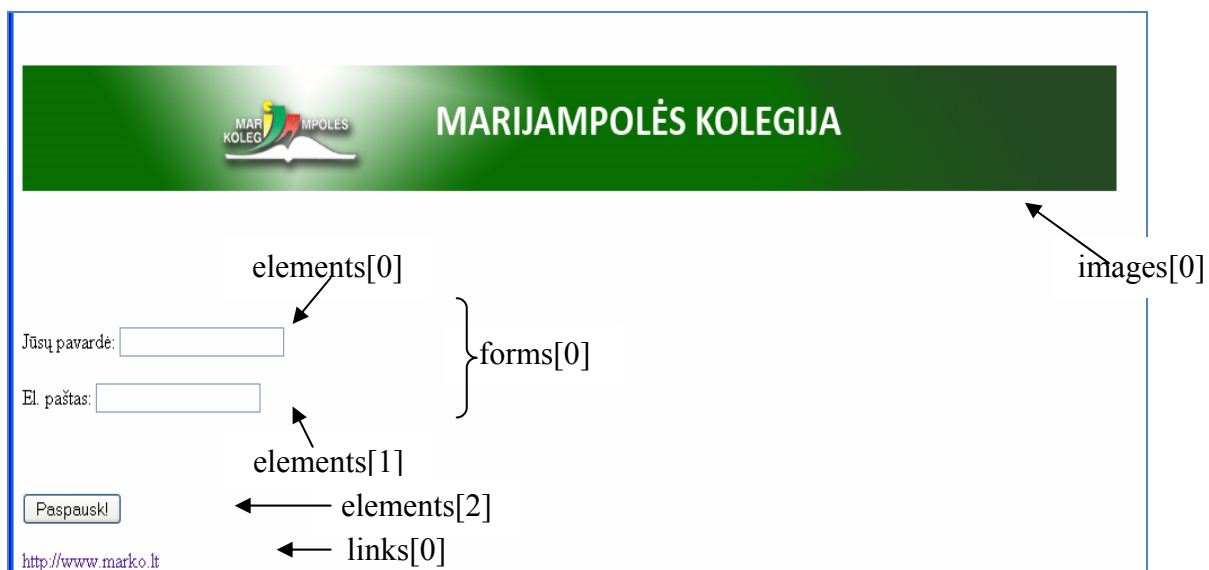
```

<script language="JavaScript">
  a = new Array (2)
  for (i=1;i<3;i++) {
    a[i] = new Array(4)
    for (j=1;j<3;j++){
      a[i][j] = prompt(i+"eilutė"+j+"stulpelis",'Jonas'); }
    }
  for (i=1;i<3;i++) {
    str = "Eilė"+i+":"
    for (j=1;j<3;j++) {
      document.write(i+"eilutė " +j+"stulpelis " +a[i][j], "<br>")
    }
  }
</script>

```

### 3.5.2. Objektinis naršyklės modelis. JavaScript objektų hierarchija

*JavaScript* kalboje visi tinklalapio elementai sudėti tam tikra hierarchine tvarka. Kiekvienas elementas yra tam tikras objektas. Kiekvienas objektas turi tam tikrus metodus ir savybes. *JavaScript* leidžia gana lengvai valdyti tinklalapio objektus, tačiau būtina žinoti ir objektų hierarchiją.



3.5.2.1 pav. Objektinis naršyklės modelis

3.5.2.1 pav. yra vienas paveikslėlis, viena nuoroda ir forma, kuri sudaryta iš dviejų teksto įvedimo laukelių ir mygtuko.

*JavaScript* požiūriu, naršyklės langas – tai objektas *window*. Šis objektas turi savo elementų: informacinę eilutę, tvarka. Naršyklės lango viduje galima įdėti *HTML* dokumentą. Toks tinklalapis yra *document* objektas. Šio objekto savybėms priskiriamos tokios savybės kaip, pvz., tinklalapio fonas. Be išimties visi *HTML* kalbos objektai priskiriami objekto *document*

savybėms. Kaip matyti iš 3.5.2.1 pav., kiekvienas objektas turi savo vardą hierarchinėje struktūroje. Norint kreiptis į kurį nors objektą, reikia žinoti ir jo vietą šioje struktūroje.

Pirmas objektas 3.5.2.1 pav. yra *document*. Paveikslėlis tinklalapyje – *images[0]*, tai reiškia, kad norint kreiptis į šį objektą reikia rašyti *document.images[0]*.

Pavyzdžiui, jei norima sužinoti, kokią tekstą įvedė vartotojas į pirmąjį formos laukelį, tai pirma reikia žinoti kelią iki šio laukelio. Pradedama nuo objekto *document* – *document.forms[0].elements[0]*. Norint sužinoti, ką vartotojas parašė, reikia žinoti, kad teksto įvedimo laukelis turi savybę *value*, kuri kaip tik yra vartotojo įvesta reikšmė. Tam, kad būtų galima perskaityti šią reikšmę, reikėtų rašyti šitaip:

```
name=document.forms[0].elements[0].value,
```

Kintamasis *name* įgis šiame laukelyje įrašytą reikšmę, kurią galima panaudoti programoje. Pavyzdžiui, parašius eilutę

```
alert("Labas"+name),
```

atrodys, lyg programa pasisveikino su vartotojų pagalba. Įrašius vardą Aldona, ekrane atsiras pranešimas „*Labas Aldona*“. Jeigu kuriamas tinklalapis yra labai didelis, tokia hierarchija tampa pernelyg sudėtinga, tokiais atvejais objektams galima priskirti unikalius vardus. Pvz., kai formos vardas yra *Duomenys*, o pirmojo tekstinio lauko pavadinimas *vardas*, tada vietoj užrašo:

```
name=document.forms[0].elements[0].value,
```

galima rašyti:

```
name=document.Duomenys.vardas.value;
```

Šis užrašas reiškia, kad objektas *forms[0]* gauna naują vardą – *Duomenys*. Lygiai taip pat vietoj *elements[0]* galima užrašyti *vardas* (tai rodo užrašas bloko *<input>* atributo vardas). Tai supaprastina programavimą *JavaScript* kalba, kuriant didelės apimties tinklalapius. Dabar galima ne tik perskaityti, kas parašyta lauke, bet ir įrašyti jame naujas reikšmes. Pavyzdžiui, jei į formą vietoj *value* bus įrašytas bazinis įvykis *onClick*:

```
onClick="document.Duomenys.vardas.value= 'Dabar šis laukas aktyvus'; ">,
```

tai perkėlus žymeklį į laukelį *vardas*, jame bus užrašyta: „*Dabar šis laukas aktyvus*“.

### 3.5.3. Objektas Date

Objektas *Date* ir jo metodai naudojami atlikti įvairiems veiksams su datomis ir laiku. Šis objektas turi daug metodų datoms nustatyti, jų reikšmėms gauti, tačiau neturi savybių. Data *JavaScript* kalboje skaičiuojama milisekundėmis nuo 1970 metų sausio pirmos dienos. Vadinasi, viena diena yra sudaryta iš 86400000 milisekundžių. *JavaScript* kalboje galimas skaičių intervalas – nuo -9007199254740991 iki 9007199254740991, vadinasi, galima operuoti 285616 metais.

Sintaksė:

*MyDate=new Date ([parametrai])*

Galimi tokie parametrų variantai:

- jokių parametrų, kintamajam bus priskirta data, kurios formatas „Mėnuo, diena, metai, valandos: minutės: sekundės“, pvz., *someDate=new Date ("Birželis 15,1996")*; Jeigu valandos, minutės ar sekundės nenurodomos, jų reikšmė 0;
- skaičių, kurie sudaro pilną rinkinį: metus, mėnesį ir dieną, pvz., *otherDay=new Date(96,4,15)*;
- skaičių, kurie sudaro pilną rinkinį – metus, mėnesį, dieną, valandą, minutes ir sekundes, pvz., *sameDay=new Date(96,4,15,15,30,0)*.

Objekto *Date* metodus pagal paskirtį galima suskirstyti į šias grupes:

- laiko nustatymas (*set*);
- laiko gavimas;
- laiko ir datos vertimas iš eilutės;
- veiksmai su datomis.

Datai ir laikui nustatyti ir gauti naudojami sekundžių, minučių, valandų, dienų, mėnesio, savaitės ir metų gavimo metodai (žr. 2.5.3.1 lentelę). Pavyzdžiui, metodas *getDay* naudojamas sužinoti, kokia yra savaitės diena, bet metodu *setDay* savaitės diena nėra nustatoma automatiškai.

3.5.3.1 lentelė. *Datos metodai*

Metodas	Aprašymas
<i>getDate</i>	Grąžina einamo mėnesio dieną – skaičių nuo 1 iki 31
<i>getDay</i>	Grąžina savaitės dieną nuo 0 (sekmadienio) iki 6 (šeštadienio)
<i>getHours</i>	Grąžina valandas – sveiką skaičių nuo 0 iki 23
<i>getMinutes</i>	Grąžina minutes – sveiką skaičių nuo 0 iki 59
<i>getMonth</i>	Grąžina mėnesį – sveiką skaičių nuo 0 (sausio) iki 11 (gruodžio)
<i>getSeconds</i>	Grąžina minutes – sveiką skaičių nuo 0 iki 59
<i>getTime</i>	Grąžina skaičių – milisekundėmis nuo 1970 metų sausio 1 dienos 000:00:00 ir nurodytos objekte <i>Date</i> datos
<i>getTimeZoneOffset</i>	Grąžina laiko skirtumą tarp esamo vietinio laiko ir Grinvičo laiko minutėmis
<i>getYear</i>	Grąžina du paskutiniuosius metų skaičius
<i>Parse</i>	Grąžina skaičių milisekundėmis tarp 1970 metų sausio 1 dienos 00:00:00 ir datos eilutėje
<i>Setdate</i>	Nustato mėnesio dieną
<i>SetHours</i>	Nustato valandas
<i>SetMinutes</i>	Nustato minutes
<i>SetMonth</i>	Nustato mėnesio numerį
<i>SetSeconds</i>	Nustato sekundes
<i>SetTime</i>	Nustato laiką
<i>SetYear</i>	Nustato metus



<b><i>ToGMTString</i></b>	Keičia lokalų laiką pagal Grinvičo laiką ir grąžina eilutės pavidalu
<b><i>ToLocaleString</i></b>	Keičia Grinvičo laiką į lokalų laiką ir grąžina eilutės pavidalu
<b><i>UTC</i></b>	Grąžina skaičių milisekundėmis tarp 1970 metų sausio 1 dienos ir datos, nurodytos parametru

Lentelėje 3.5.3.1 išvardinti datos nustatymo metodai ir apribojami skaitiniais intervalais. Jų galiojimo ribos aprašytos 3.5.3.2 lentelėje.

3.5.3.2 lentelė. *Skaitinių intervalų galiojimo ribos*

<b>Reikšmė</b>	<b>Intervalas</b>
<b><i>Sekundės ir minutės</i></b>	0..59
<b><i>Valandos</i></b>	0..23
<b><i>Savaitės dienos</i></b>	0..6
<b><i>Mėnesio diena</i></b>	1..31
<b><i>Mėnuo</i></b>	0..11
<b><i>Metai</i></b>	Metai nuo 1990

### 3.5.4. Objektas Function

Objektas Function naudojamas, kai reikia sukurti *JavaScript* kodo eilutę, kuri skaitoma kaip funkcija. Kaip ir įprasta, funkcija gali turėti keletą parametrų ir tik vieną reikšmę. Aprašant parametrus nebūtina nurodyti kintamųjų tipus. Naujiems objektams kurti naudojamas konstruktorius *new*. Naujos funkcijos aprašymo sintaksė:

```
Function funkcijos_vardas([1parametras] [, 2 parametras] ...[,Nparametras]){
//Funkciją sudarantys operatoriai
}
```

Jei naudojamas konstruktorius *new*, tai funkciją galima aprašyti kaip kintamąjį:

```
Var kintamojo_vardas = new Function([1parametras] [, 2
parametras] ...[,Nparametras]);
```

### 3.5.5. Objektas Math

Objektas *Math* turi visą rinkinį savybių (matematinės konstantos) ir metodų (trigonometrinės ir matematinės funkcijos). Kaip ir kiti objektai, jis gali būti panaudojamas atskirai ir kartu su kitais metodais. Objekto savybės:

*E* – Oilerio konstanta (apie 2.718),

*LN2* – skaičiaus 2 natūralusis logaritmas (apie 0.693),

*LOG2E* – skaičiaus *E* logaritmas, kai pagrindas 2 (apie 1.442),

*LOG10E* – skaičiaus *E* logaritmas pagrindu 10 (apie 1.434),

*PI* – skaičiaus  $\pi$  reikšmė (apie 3.1415),

*SQRT1\_2* – skaičiaus 0.5 kvadratinė šaknis (apie 0.707),

*SQRT2* – skaičiaus 2 kvadratinė šaknis (apie 1.414).

Objekto *Math* metodo argumentas rašomas paprastuose skliaustuose.

3.5.5.1 lentelė. Objekto *Math* metodai

Metodas	Aprašymas
<i>sin, cos, tan</i>	Standartinės trigonometrinės funkcijos, argumentas –adianais
<i>abs</i>	Absoliuti reikšmė – modulis
<i>acos, asin, atan</i>	Arksinusas, arksinusas, arktangentas
<i>exp, log</i>	Eksponentė ir natūrinis logaritmas (pagrindas e)
<i>ceil</i>	Sveikoji skaičiaus dalis, didesnė arba lygi skaičiui
<i>floor</i>	Sveikoji skaičiaus dalis, mažesnė arba lygi skaičiui
<i>min, max</i>	Mažesnis (didesnis iš dviejų) argumentų
<i>pow</i>	Argumento žingsnis
<i>round</i>	Apvalina argumentą
<i>sgrt</i>	Kvadratinė šaknis

### 3.5.6. Objektas *String*

Objektas *String* turi nemažai metodų, skirtų atlikti įvairius veiksmus su eilutėmis, ir vieną savybę (*length*), kurią naudojant galima sužinoti eilutės ilgį.

Panaudojant objektą *String*, kintamasis kuriamas konstruktoriumi *New*:

*Eilute=new String("Mano tekstinė eilutė");*

3.5.6.1 lentelė. Objekto *String* metodai

Metodai	Aprašymas
<i>anchor</i>	Sukuria <i>HTML</i> – elementą <i>anchor</i>
<i>big, blink, bold, fixed, italics, small, strike, sub, sup</i>	Sukuria atitinkamą <i>HTML</i> kalboje parašyto teksto elementą
<i>charAt</i>	Simbolis, esantis nurodytoje eilutės pozicijoje
<i>indexOf, lastIndexOf</i>	Simbolio pozicija nurodytoje eilutėje, paskutinio eilutės simbolio pozicija
<i>Link</i>	Sukuria <i>HTML</i> nuorodą į kitą puslapį
<i>split</i>	Padalina eilutę į eilučių masyvą
<i>substring</i>	Nurodytos eilutės dalis
<i>toLowerCase</i> <i>toUpperCase</i>	Verčia eilutės simbolius į mažąsias arba didžiąsias raides

### 3.5.7. Objektas *Navigator*

Šis objektas pasirenkamas, kai reikia sužinti informaciją apie naudojamos naršyklės modelį ir kitas savybes.

3.5.7.1 lentelė. Objekto Navigator savybės

Savybės	Aprašymas
<i>AppCodeName</i>	Naršyklės vardo kodas
<i>AppName</i>	Naršyklės pavadinimas
<i>AppVersion</i>	Naršyklės versija
<i>UserAgent</i>	Ir naršyklės pavadinimas, ir versija
<i>JavaEnabled</i>	Leidžia sužinoti, ar palaikoma Java kalba
<i>CookieEnabled</i>	Leidžia sužinoti, ar palaikomas cookies
<i>TaintEnabled</i>	Leidžia sužinoti, ar palaikomas <i>taint</i> (tikrai <i>netscape</i> )

### 3.5.8. Objektas Window

Objektas *Window* teikia ir valdo informaciją, esančią lange. Jo savybės pateiktos 3.5.8.1 lentelėje.

3.5.8.1 lentelė. Objekto window savybės

Savybės	Aprašymas
<i>Parent</i>	Gražina „tėvišką“ langą duotajam
<i>Self</i>	Gražina nuorodą į aktyvų langą
<i>Top</i>	Gražina nuorodą į langą, esantį arčiausiai vartotojo
<i>Name</i>	Gražina lango vardą, sukurtą bloke <code>&lt;frame&gt;</code>
<i>Opener</i>	Gražina langą, sukūrusį duotąjį
<i>Closed</i>	Parodo, kad langas uždarytas
<i>Status</i>	Naršyklės informacinės eilutės tekstas
<i>Defaultstatus</i>	Gražina tekstą, esantį informacinėje eilutėje
<i>ReturnValue</i>	Sugražina dialogo lango reikšmę
<i>Document</i>	Gražina nuorodą į objektą <i>document</i>
<i>Event</i>	Gražina nuorodą į globalinį objektą <i>event</i>
<i>History</i>	Gražina nuorodą į objektą <i>history</i>
<i>Location</i>	Gražina nuorodą į objektą <i>location</i>
<i>Navigator</i>	Gražina nuorodą į objektą <i>navigator</i>
<i>Screen</i>	Gražina nuorodą į globalinį objektą <i>screen</i>
<i>ClientInformation</i>	Gražina nuorodą į objektą <i>Navigator</i>

3.5.8.2 lentelė. Objekto window metodai, skirti langų valdymui ir įvairiems veiksmams atlikti

Metodas	Aprašymas
<i>Open</i>	Atidaro naują naršyklės langą
<i>Close</i>	Uždaro aktyvų naršyklės langą
<i>ShowHelp</i>	Išskviečia pagalbos langą
<i>ShowModalDialog</i>	Išskviečia naują langą, kaip modalinį dialogo langą
<i>Alert</i>	Pranešimų langas
<i>Prompt</i>	Dialogo langas
<i>Confirm</i>	Dialogo langas
<i>Navigate</i>	Nuskaityti kitą tinklalapį
<i>Blur</i>	Dingsta tinklalapio efektas
<i>Focus</i>	Perveda koki nors efektą į nurodytą tinklalapį

<b><i>Scroll</i></b>	Persuka tinklalapio turinį horizontaliai ir vertikaliai
<b><i>SetInterval</i></b>	Nustato funkcijos vykdymo intervalą
<b><i>SetTimeout</i></b>	Nustato laiko intervalą, po kurio atliekami kokie nors veiksmai
<b><i>ClearInterval</i></b>	Atšaukia metodo <i>SetInterval</i> vykdymą
<b><i>ClearTimeout</i></b>	Atšaukia metodo <i>Set Timeout</i> vykdymą
<b><i>execScript</i></b>	Vykdo skripto programą

Naujam langui atidaryti naudojami metodai *open* ir *close*. Pavyzdžiui, komanda *Window.open("bandymas.html")* atidarys langą, kuriame bus *HTML* failo *bandymas.html* turinys.

Metodas *open* turi daug papildomų argumentų, kurie leidžia nusakyti lango padėtį, jo dydį ir tipą. Taip pat nurodo, ar langas turi turėti persukimo liniuotę, komandinių mygtukų liniuotę ir t. t. Be to, galima nurodyti ir lango vardą. Jo sintaksė atrodo taip:

*Newwind=window.open(URL, name, features, replace),*

čia: *URL* – failo adresas, kuris rodomas naujame lange; jeigu adresas nenurodomas, rodomas tuščias langas;

*name* – eilutė, nurodanti lango vardą;

*features* – eilutė, nurodanti naujo lango parametrus;

*replace* – nurodo, ar naujas langas įrašomas į *history* sąrašą.

Antrasis argumentas – lango vardas gali būti naudojamas kaip *target* parametras. Taigi, jei žinomas lango vardas, tai jame galima atverti tinklalapį, kurio užrašas atrodo taip:

*<a href="tinklapis.html" target="displayWindow">*

Lango vardas (šiuo atveju *displayWindow*) yra unikalus identifikatorius, kuriuo galima naudotis iš bet kurio naršyklės lango. Naujo lango parametrai, naudojami *features* eilutėje, pateikti 3.5.8.3 lentelėje, naujos langų savybės – 3.5.8.4 lentelėje.

3.5.8.3 lentelė. Naujo lango parametrai

Parametras	Reikšmė	Aprašymas
<b><i>Fullscreen</i></b>	Yes No 1 0	Nurodo lango dydį (per visą ekraną ar ne)
<b><i>Toolbar</i></b>	Yes No 1 0	Ar rodyti naršyklės mygtukų liniuotę
<b><i>Location</i></b>	Yes No 1 0	Ar rodyti adreso eilutę
<b><i>Directories</i></b>	Yes No 1 0	Ar rodyti katalogą
<b><i>Status</i></b>	Yes No 1 0	Ar rodyti naršyklės informacinę eilutę
<b><i>Menubar</i></b>	Yes No 1 0	Ar rodyti meniu liniuotę
<b><i>Scrollbars</i></b>	Yes No 1 0	Horizontali ir vertikali persukimo juosta
<b><i>Resizable</i></b>	Yes No 1 0	Nurodymas, ar gali langas keisti dydį
<b><i>Width</i></b>	Skaičius	Nurodomas lango plotis pikseliais. Minimali reikšmė – 100
<b><i>Height</i></b>	Skaičius	Nurodomas lango aukštis pikseliais. Minimali reikšmė – 100
<b><i>Top</i></b>	Skaičius	Nurodo viršutinio kairiojo lango kampo vertikaliąją koordinatę
<b><i>Left</i></b>	Skaičius	Nurodo viršutinio kairiojo lango kampo horizontaliąją koordinatę

3.5.8.4 lentelė. *Langų naujos savybės*

Parametras	Reikšmė
<i>alwaysLowered</i>	Yes No
<i>AlwaysRaised</i>	Yes No
<i>Dependent</i>	Yes No
<i>Hotkeys</i>	Yes No
<i>InnerWidth</i>	Taškų skaičius (keičia <i>width</i> )
<i>InnerHeight</i>	Taškų skaičius (keičia <i>height</i> )
<i>OuterWidth</i>	Taškų skaičius
<i>OuterHeight</i>	Taškų skaičius
<i>ScreenX</i>	Taškų skaičius
<i>ScreenY</i>	Taškų skaičius
<i>Titlebar</i>	Yes No
<i>z-lock</i>	Yes No

Objektui *window* priklauso ir metodai: *alert*, *prompt*, *confirm*.

Laikmačiai taip pat naudojami objekte *window*. Objektas *window* turi ir du metodus – *setTimeout* ir *setInterval* – naudojamus taimeriams valdyti. Funkcijos *setTimeout* pagalba galima parašyti tokią programą, kuri atliktų tam tikrus veiksmus praėjus nurodytam laikui.

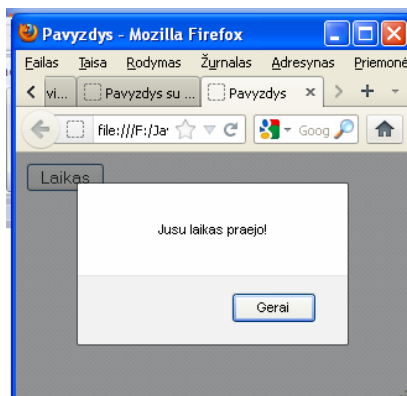
Pavyzdyje (žr. 3.5.8.1) pateikiama programa, kuri tinklalapyje pateiks tik vieną mygtuką – „*Laikas*“, kurį spustelėjus pranešimas funkcijos *laikas()* pagalba išskviečiamas ne iš karto, o praėjus trimis sekundėms. Pirmasis *setTimeout* argumentas nurodo, ką reikia padaryti, šiuo atveju – iškvieisti funkciją *alert*. Būtina įsidėmėti, kad šis argumentas rašomas kabutėse. Antrasis argumentas – laikas milisekundėmis.

Pvz. Funkcija *setTimeout*:

```

<html>
<head>
  <title>Pavyzdys</title>
  <script language="JavaScript">
    //Funkcija timer lauks 3000 milisekundžių
    function laikas() {
      setTimeout("alert('Jūsų laikas praėjo!')", 3000);
    }
  </script>
</head>
<body>
  <form>
    <input type="button" value="Laikas" onClick="laikas()">
  </form>
</body>
</html>

```



3.5.8.1 pav. Vaizdas naršyklėje

Funkciją *setTimeout* galima panaudoti ir reklaminiams paveikslėliams keisti kas tam tikrą laiko intervalą. Tokiu atveju kiekvienas paveikslėlis yra nuoroda į vis kitą tinklalapį. Patogiausia tokią programą realizuoti naudojant masyvus.

Pakartotiniams veiksams atlikti naudojamas metodas *setInterval*. Pavyzdyje kas penkias sekundes spausdinamas pranešimas „Praėjo penkios sekundės“. Funkcija *rodykLaika()* išspausdins ekrane laiką, o funkcija *laikas()* skaičiuoja 5 sekundes ir kiekvieną kartą kreipiasi į save.

Pvz. Funkcija *rodykLaika()*:

```
<html>
<head>
  <title>5 sekundžių laikmatis</title>
</head>
<body>
  <script language = "JavaScript">
function rodykLaika()
  {
    window.alert("Praėjo penkios sekundės")
  }
function laikas()
  {
    window.setInterval("rodykLaika()",5000,"JavaScript")
  }
  laikas()
</script>
</body>
</html>
```

Metodai *clearTimeout* ir *clearInterval* atitinkamai panaikina metodų *setTimeout* ir *setInterval* veikimą.

Pvz. Skaitliuko su funkcija *clearTimeout* vaizdas naršyklėje pateiktas 3.5.8.2 pav.:

```
<html>
<head>
```

```

<script type="text/javascript">
var c=0;
var t;
var timer_is_on=0;

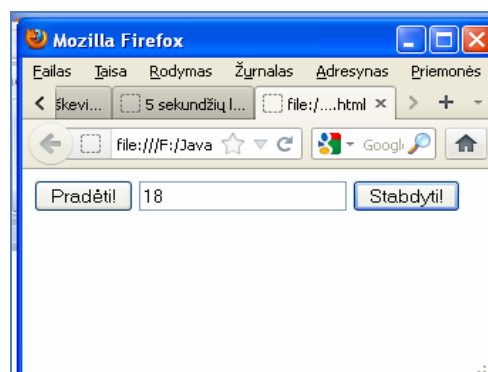
function timedCount()
{
document.getElementById('txt').value=c;
c=c+1;
t=setTimeout("timedCount()",1000);
}

function doTimer()
{
if (!timer_is_on)
{
timer_is_on=1;
timedCount();
}
}

function stopCount()
{
clearTimeout(t);
timer_is_on=0;
}
</script>
</head>

<body>
<form>
<input type="button" value="Pradėti!" onclick="doTimer()" />
<input type="text" id="txt" />
<input type="button" value="Stabdyti!" onclick="stopCount()" />
</form>
</body>
</html>

```



3.5.8.2 pav. Skaitliukas su funkcija `clearTimeout`

Objekto *window* įvykiai pateikti 3.5.8.5 lentelėje.

3.5.8.5 lentelė Objekto *window* įvykiai

Įvykis	Aprašymas
<i>OnBeforeUnload</i>	Įvyksta prieš nuskaitant tinklapį
<i>OnBlur</i>	Įvyksta, kai prarandamas koks nors efektas
<i>OnError</i>	Įvyksta, kai atsiranda klaidų nuskaitant tinklalapį ar grafinį objektą
<i>OnFocus</i>	Įvyksta, kai atsitinka koks nors efektas
<i>OnHelp</i>	Įvyksta, kai paspaudžiamas klavišas F1
<i>OnLoad</i>	Įvyksta, kai nuskaitomas tinklapis
<i>OnResize</i>	Įvyksta, kai pakeičiamas naršyklės lango dydis
<i>onScroll</i>	Įvyksta, kai persukamas lango turinys (persukimo juosta)
<i>OnUnload</i>	Įvyksta, kai pereinama į kitą tinklapį ar baigiamas darbas su naršykle

### 3.5.9. Objektas Location

Šis objektas pateikia informaciją apie aktyvaus tinklalapio *URL* adresą. Jo savybės išvardintos 3.5.9.1 lentelėje.

3.5.9.1 lentelė. Objekto *Location* savybės

Savybė	Aprašymas
<i>Href</i>	Pilnas tinklalapio <i>URL</i> adresas
<i>Hash</i>	Eilutė einanti po simbolio „#“
<i>Host</i>	Adreso <i>hostname</i> dalis: <i>port</i>
<i>Hostname</i>	Adreso <i>hostname</i> dalis
<i>Pathname</i>	Failo pavadinimas
<i>Port</i>	Porto numeris
<i>Protocol</i>	Protokolo vardas
<i>Search</i>	Eilutė einanti po simbolio „?“

Paprastai objektas *location* naudojamas kitiems tinklalapiams nuskaityti. Tam tikslui savybei *href* priskiriama nauja reikšmė. Ši operacija ekvivalenti objekto *window* metodui *navigate*. Pavyzdžiui, komanda *Window.location.href=http://www.naujas.lt/naujas.html* iškviečia tinklalapį *naujas.html*.

3.5.9.2 lentelė. Objekto *Location* metodai

Metodas	Aprašymas
<i>Assign</i>	Nuskaito kitą tinklalapį. Veiksmas ekvivalentus: <i>Window.location.href</i>
<i>Reload</i>	Iš naujo nuskaito aktyvų tinklalapį
<i>Replace</i>	Nuskaito tinklalapį ir pažymi jį <i>history</i> sąrašė



### 3.5.10. Objektas Event

Jis leidžia skripto programai sužinoti detalią informaciją apie praėjusius įvykius ir įvykdyti atitinkamus veiksmus. Pavyzdyje pateikta programa parodo, kaip naudotis savybe *event.type*:

```
<html>
<head>
<title>Pavyzdys</title>
</head>
<body>
<script language="JavaScript">
function vykdyk(){
alert("event.type="+window.event.type)
}
</script>
<a href="tinklapis.html" onMouseOver="vykdyk();">Pereiti į kitą tinklalapį</a>
<form>
<input type="button" value="OK" onClick="vykdyk();">
</form>
</body>
</html>
```

Kai perkeliamas žymeklis ant užrašo *Pereiti į kitą tinklalapį*, iškviesta funkcija *vykdyk()* išspausdins pranešimą *event.type=mouseover*. Paspaudus mygtuką *OK*, bus kitas įvykis *onClick* ir vėl iškviesta funkcija *vykdyk()*, o ekrane atsiras pranešimas *event.type=click*.

### 3.5.11. Objektas Screen

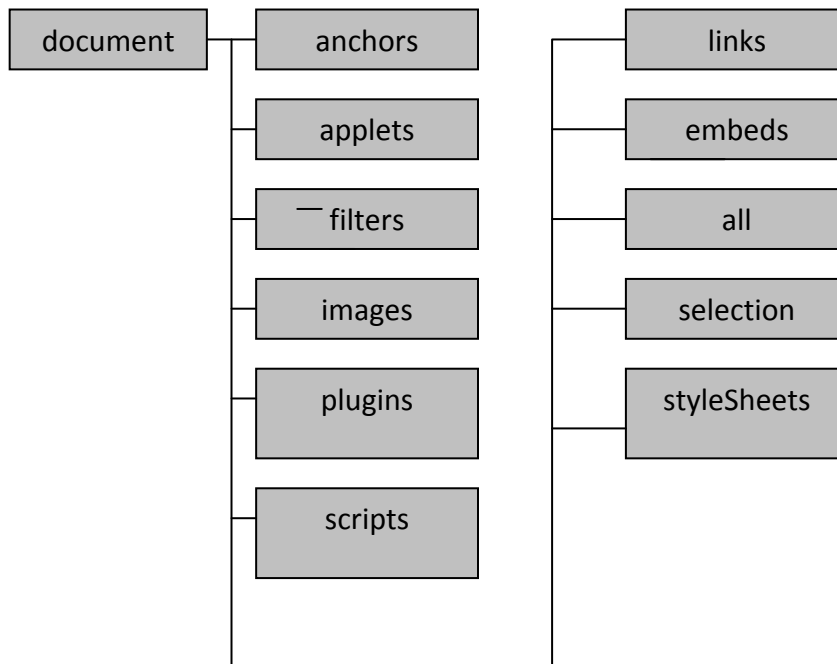
Jis skirtas informacijai apie naršyklę pateikti.

3.5.11.1 lentelė. Objekto Screen savybės

Savybės	Aprašymas	Skaitymas/Rašymas
<i>AvailHeight</i>	Pasiekiamos ekrano srities aukštis pikseliais	Skaitymas
<i>AvailWidth</i>	Pasiekiamas ekrano plotis pikseliais	Skaitymas
<i>BufferDepth</i>	Buferio dydis	Skaitymas/Rašymas
<i>ColorDepth</i>	Maksimalus spalvų skaičius	Skaitymas
<i>Height</i>	Ekrano aukštis pikseliais	Skaitymas
<i>UpdateInterval</i>	Ekrano atnaujinimo intervalas	Skaitymas/Rašymas
<i>Width</i>	Ekrano plotis pikseliais	Skaitymas

### 3.5.12. Objektas Document

Šis objektas naudojamas, kai reikia sužinoti informaciją apie *HTML* dokumentą (žr. 3.5.12.1 pav.).



3.5.12.1 Pav. Objekto Document savybės

Yra įvairių savybių, padedančių sužinoti informaciją apie patį failą, kuriame saugomas *HTML* dokumentas. Pvz. Paskutinę dokumento redagavimo datą galima sužinoti taip:

```
<script language="JavaScript">
document.write(document.lastModified)
</script>
```

3.5.12.1 lentelė. Savybės, susietos su spalvomis

Savybės	Aprašymas
<b><i>bgColor</i></b>	Dokumento fono spalva
<b><i>fgColor</i></b>	Dokumento teksto spalva
<b><i>linkColor</i></b>	Hipersąsajos spalva
<b><i>alinkColor</i></b>	Aktyvios hipersąsajos spalva
<b><i>vlinkColor</i></b>	Anksčiau lankyto puslapio hipersąsajos spalva

Objekto *document* įvykiai naudojami tam, kad tinklalapis būtų aktyvus, patogų valdyti vartotojo veiksmus tinklapyje (klavišų paspaudimas, pelės kursoriaus „vaikščiojimas“, mygtukų paspaudimai ir t. t.) ir atitinkamai į juos reaguoti.

3.5.12.2 lentelė. Objekto Document įvykiai

Įvykiai	Aprašymas
<b><i>onClick</i></b>	Įvyksta, kai spragtelėjamas kuris nors pelės klavišas
<b><i>onDoubleClick</i></b>	Įvyksta, kai spragtelėjamas du kartus pelės klavišas

<b><i>onMouseDown</i></b>	Įvyksta, kai nuspaudžiamas pelės klavišas
<b><i>OnMouseOver</i></b>	Įvyksta, kai pelės žymeklis išeina iš kokio nors elemento srities
<b><i>OnMouseMove</i></b>	Įvyksta, kai juda pelės žymeklis
<b><i>OnMouseOut</i></b>	Įvyksta, kai pelės žymeklis išeina iš elemento srities
<b><i>OnMouseUp</i></b>	Įvyksta, kai atleidžiamas anksčiau nuspaustas pelės klavišas
<b><i>OnDragStart</i></b>	Įvyksta, kai pertempiamas koks nors elementas
<b><i>OnSelectStart</i></b>	Įvyksta, kai pažymimas koks nors elementas
<b><i>OnKeyDown</i></b>	Įvyksta, kai paspaudžiamas klavišas
<b><i>OnKeyPress</i></b>	Įvyksta, kai klavišas nuspaustas ir laikomas
<b><i>OnKeyUp</i></b>	Įvyksta, kai atleidžiamas anksčiau nuspaustas klavišas
<b><i>onHelp</i></b>	Įvyksta, kai nuspaudžiamas F1 klavišas

Objekto *document* kolekcija naudojama, kai reikia atlikti veiksmus su informacija, pateikta tinklalapyje. Pilnas aprašas pateikiamas 3.5.12.3 lentelėje.

3.5.12.3 lentelė *Objekto Document kolekcija*

<b>Kolekcija</b>	<b>Aprašymas</b>
<b><i>All</i></b>	Kolekcija visų blokų ir elementų išdėstytų dokumente
<b><i>Anchor</i></b>	Kolekcija visų anchor elementų
<b><i>Applets</i></b>	Kolekcija visų dokumento objektų: grafinių, apletų ir kt.
<b><i>Embeds</i></b>	Kolekcija visų elementų įterptų į dokumentą bloke <i>&lt;embed&gt;</i>
<b><i>Filters</i></b>	Kolekcija visų filtrų, naudojamų dokumente
<b><i>Frames</i></b>	Rėmelių kolekcija, bloke <i>&lt;frame&gt;</i>
<b><i>Forms</i></b>	Formų kolekcija
<b><i>Images</i></b>	Grafinių objektų kolekcija
<b><i>Links</i></b>	Nuorodų ir žemėlapių kolekcija
<b><i>Plugins</i></b>	Papildomų modulių kolekcija
<b><i>Scripts</i></b>	Visų skripto programų kolekcija
<b><i>styleSheets</i></b>	Stilių kolekcija

Kolekcijos struktūra panaši į masyvo – kiekvienas elementas turi savo indeksą. Kolekcija *all* – tai rinkinys visų blokų ir elementų, išdėstytų tinklalapyje. Kolekciją *frames* galima naudoti tam, kad būtų galima gauti įvairius rėmelių atributus, pavyzdžiui, jų vardus ar rėmelių kitimo atributus. Paprastai rėmeliams suteikiami logiški vardai, kad vėliau į juos būtų lengva kreiptis. Pavyzdžiui, jei rėmelis yra dešinėje, nuoroda į jį gali būti tokia:

*Window.parent.right.location.href="right1.html"*

Kolekcija *forms* naudojama, kai reikia prieiti prie atskirų formos elementų, ir yra gana svarbus informacijos iš vartotojų gauti. Formos ir jų elementai turi vardus, kurie suteikiami atributu *name*. Yra du formos elementų pasiekimo būdai. Pirmasis – pagal indeksą. Informacijai gauti naudojama kolekcija *forms(0)*. Pirmasis elementas visada turi indeksą 0. Formos elementai išdėstyti kolekciijoje *elements*, todėl norint sužinoti, kiek elementų turi forma, reikia užrašyti:

*Document.write("Formos elementų sąrašius".document.forms(0).elements.length)*

Savybės *action*, *method* ir *name* leidžia sužinoti atitinkamus bloko *<forms>* atributus. Norint sužinoti formos elementų tipą, reikėtų parašyti programos fragmentą.

Pvz.:

```
Var ctrls
    Ctrls=document.forms(0).elements
For (i=0;i<ctrls.length;i++)
    {
        document.write(ctrls(i).type,"<br> ")
    }
```

Gautas rezultatas:

```
#0 text
#1 text
#2 submit
#4 reset
```

Patogiau į formos elementus kreiptis pagal jų vardus. Tarkime, formai suteikiamas vardas *form1*, o pirmam elementui *text1*. Norint sužinoti pirmojo laukelio turinį, reikėtų užrašyti taip:

```
Document.write(form1.text1.value
```

Kolekcija *Images* naudojama, kai reikia prieiti prie grafinių tinklalapio elementų. Galima sužinoti kiekvieno objekto plotį, aukštį, saugojimo adresą. Šiuos atributus taip pat galima keisti.

### 3.6. Baziniai JavaScript įvykiai

Įvykiai suprantami kaip veiksmai, kuriuos vartotojas gali atlikti tinklalapyje. Tai gali būti kokio nors objekto paspaudimas pelės klavišu, teksto ar jo dalies pasirinkimas ir t. t. Įvykių rinkinys naudojamas kuriant interaktyvias, dinamiškas programas ir sudaro *Dynamic HTML* technologijos bazę. Baziniai *JavaScript* įvykiai pateikti 3.6.1 lentelėje.

3.6.1 lentelė. Baziniai JavaScript įvykiai

Įvykis	Aprašymas
<b>OnAbort</b>	Naudojamas grafinio objekto įrašymui nutraukti
<b>OnBlur</b>	Naudojamas, kai reikia objekto vaizdą fokusuoti
<b>OnChange</b>	Naudojamas tekstinio lauko reikšmei keisti
<b>OnClick</b>	Naudojamas, kai reikia, kad įvykis įvyktų paspaudus kairįjį pelės klavišą tam tikroje ekrano srityje
<b>OnError</b>	Įvykdomas, kai atsiranda klaidos įrašant puslapį ar grafinį objektą
<b>OnFocus</b>	Naudojamas, kai sufokusuojamas koks nors elementas
<b>OnLoad</b>	Įvykdomas, kai įrašomas puslapis ar grafinis objektas
<b>OnMouseOver</b>	Įvykdomas, kai pelės žymeklis atsiduria nurodyto elemento srityje
<b>OnMouseOut</b>	Įvykdomas, kai pelės žymeklis peržengia nurodyto elemento ribas
<b>OnReset</b>	Įvykdomas, kai paspaudžiamas mygtukas <i>Reset</i>
<b>OnSelect</b>	Įvykdomas, kai tekstiniame lauke pažymimas tekstas
<b>OnSubmit</b>	Įvykdomas, kai paspaudžiamas mygtukas <i>Submit</i>
<b>OnUnload</b>	Įvykdomas, kai pereinama į kitą tinklalapį arba kai baigiamas darbas su naršykle

### 3.6.1. Įvykis OnAbort

Vykdomas, kai nutraukiamas grafinio objekto siuntimas, todėl įvykio apdorojimas vyksta bloke `<img>`, kuriame aprašomi įvairūs grafinio elemento parametrai. Aprašant didelės apimties paveikslo persiuntimo parametrus, komanda `img src` tiesiogiai nurodoma, kaip elgtis, kai vartotojas sustabdys objekto siuntimą. `<Script>` bloke prieš tai turi būti nurodyti mygtukai arba veiksmai, kuriuos vartotojas turi atlikti siekdamas sustabdyti siuntimo procesą. Bendru atveju komanda atrodo taip:

```
<img src= "paveikslas.jpg onAbort= "doAbortff ">
```

čia *paveikslas.jpg* – tai siunčiamas paveikslas, o įvykis *onAbort* aprašo programos veiksmus, kuriuos reikia atlikti, kad siuntimas būtų sustabdytas.

### 3.6.2. Įvykis OnBlur

Įvykis naudojamas, kai reikia patikrinti, ar teisingai vartotojas įvedė informaciją kuriame nors formos laukelyje. Šiuo atveju vartotojas negalės pereiti į kitą lauką tol, kol neįves teisingos informacijos duotajame laukelyje.

Pvz.:

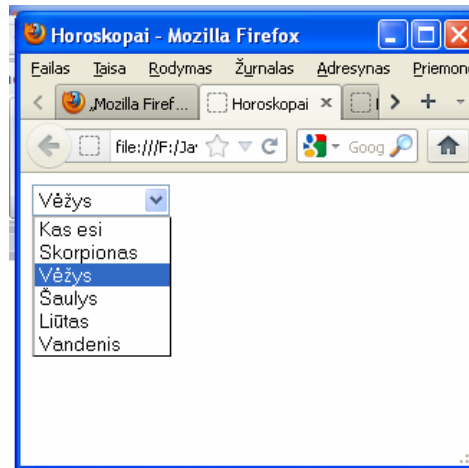
```
<html>
<head>
<script type="text/javascript">
function upperCase()
{
var x=document.getElementById("fname");
x.value=x.value.toUpperCase();
}
</script>
</head>
<body>
Įveskite vardą: <input type="text" id="fname" onblur="upperCase()" />
<p> funkcija suveikia, kai jūs paliekatė įvesties lauką. Funkcija pakeičia
mažąsias raides didžiosiomis.</p>
</body>
</html>
```

### 3.6.3. Įvykis OnChange

Įvykis naudojamas patikrinti, ar kas nors pasikeitė. Iškviečiamas tada, kai pasikeičia kuris nors formos elementas – užpildoma įvesties eilutė, išrenkamas sąrašas ir pan. Jį naudojant galima tikrinti įvestos informacijos teisingumą arba atlikti kitus veiksmus.

*OnChange* ir *HTML* forma sudaro galimybę iš sąrašo pasirinkti tam tikrą nuorodą ir pereiti į kitą tinklalapį.

Pvz. Išsiskleidžiančiame formos sąrašė pateikiami horoskopo ženklų pavadinimai (3.6.3.1 pav.):



3.6.3.1 pav. Išsiskleidžiantis formos sąrašas

Pasirinkus bet kurį ženklo pavadinimą, kviečiama funkcija *keisk()*, kuri konstruoja atitinkamo tinklalapio nuorodą ir jį iškviečia:

```
<html><head>
    <title>Horoskopai</title>
<script language="JavaScript" >
    function keisk() {
        var kur=document.forma.pasirink.options
            [forma.pasirink.selectedIndex].value;
        Window.location. href=kur-+ ". html";}
</script>
</head>
<body>
<form name="forma">
    <select name="pasirink" size="1" onchange="keisk()" >
        <option value="" selected>Kas esi
        <option value="skorp">Skorpionas
        <option value="vezys"> Vėžys
        <option value="saulys"> Šaulys
        <option value="liutas"> Liūtas
        <option value = "vandenis"> Vandenis
    </select>
</form>
</body>
</html>
```

### 3.6.4. Įvykis OnClick

Įvykis įvykdomas, kai paspaudžiamas mygtukas *Submit* ar *Reset* arba kai suaktyvinama kito tinklalapio nuoroda. Pavyzdyje programa paklaus vartotojo, ar tikrai jis nori pereiti į kitą

tinklapi. Programa sureaguos tada, kai vartotojas paspaus kairįjį pelės klavišą su nuoroda *Spausk čia*:

```
Pvz.:
<html>
<head>
    <title>Pavyzdys su "noClick"</title>
<script language= "JavaScript ">
    function doClick()
    {
        if (confirm("Ar tikrai jau išeinate?"))
            return true
        else
            return false
    }
</script>
</head>
<body>
    <a href= "kl.html" onClick= "return doClick()">
        Spausk čia</a>
</body>
</html>
```

Įvykdžius *OnClick*, ekrane pasirodys langas su dviem mygtukais: *OK* ir *Cancel*. Paspaudus *OK*, pereinama į nurodytą tinklalapį, šiuo atveju *kl.html*, jei *Cancel*, liekama tame pačiame.

Programėlė sutrumpės, jei bus panaudota mygtuko *OK* grąžinama reikšmė *true*, o *Cancel* – *false*. Standartinė funkcija *confirm* įgyja tik dvi reikšmes (arba *true*, arba *false*), panaudojus ją galima nebenaudoti operatoriaus *if*:

```
<html>
<head>
    <title>Pavyzdys su "noClick"</title>
<script language= "JavaScript">
    function doClick()
    {
        return confirm("Ar tikrai jau išeinate?")
    }
</script>
</head>
<body>
    <a href= "kl.html" onClick= "return doClick()">
        Spausk čia</a>
</body>
</html>
```

### 3.6.5. Įvykis OnError

Įvykis naudojamas, kai padaroma klaida įrašant dokumentą ar grafinį objektą. Tada galima perduoti suformuluotą pranešimą vartotojui. Pateiktame pavyzdyje programa, panaudodama *HTML* konstrukciją *img src*, tikrina, ar objektas įvestas teisingai. Jei ne, tai funkcija *doError()* spausdina pranešimą lange *Grafinio objekto įrašymo klaida*, vietoj paveikslo užrašomas alternatyvus tekstas:

```
<html>
<head>
<title> Pavyzdys su onError</title>
<script language= "JavaScript">
    function doError () {
        alert("Grafinio objekto Įrašymo klaida");
    }
</script>
</head>
<body>
    <img src = "c.windows/images/paveikslas.jpg"
    onError= "doError()" alt = "deja, paveikslėlio nematysite ">
</body>
</html>
```

### 3.6.6. Įvykis OnFocus

Įvykdomas tada, kai persiunčiamas tinklalapio elementas. Juo gali būti langas, rėmelis, formos elementas. Naudojamas, kai vartotojui norima padėti ar kaip nors kitaip sureaguoti į jo veiksmus. Pavyzdžio bloke, panaudojant *HTML* kalbos konstrukciją *form*, išskviečiamas langas. Jame pateikiamas komentaras, siūlantis įrašyti savo elektroninio pašto adresą. Kai perkeliamas žymeklis į šį langą, užrašas panaikinamas:

```
<html>
<head>
<title > Pavyzdys </title >
<script language= "JavaScript">
    function valyk_forma() {
        document.form.form. value=""
    }
</script>
</head>
<body>
    <form name="form">
        <p>< input onFocus "valyk_forma()"name="form"
        size = "12" value=" [Jūsų E-mail ] "></p>
    </form>
</body>
< html>
```



### 3.6.7. Įvykis OnLoad

Įvykdomas, kai baigiamas persiūsti puslapis arba grafinis objektas. Dažniausiai naudojamas, kai reikia įjungti laikrodį arba iškviešti kokią nors funkciją, ir t. t. Pavyzdyje, kol bus siunčiamas puslapis, vartotojas matys pranešimą: *Minutėlę, siunčiama puslapio medžiaga...*:

```
<html>
<head><title>Pavyzdys su onLoad</title><script language = "JavaScript"> function
doLoad() {
alert("Minutėlę, siunčiama puslapio medžiaga... " }
</script></head>
<body onLoad= "doLoad()" "></body></html>
```

### 3.6.8. Įvykiai OnMouseOver ir OnMouseOut

Įvykdomi, kai pelės žymeklis patenka į nurodyto objekto sritį. Pavyzdžio bloke <a> aprašyti du įvykiai: *onMouseOver* ir *onMouseOut*. Patekus žymekliui ant nuorodos, naršyklės informacinėje eilutėje pasirodo pranešimas, paaiškinantis nuorodos paskirtį. Tam naudojama funkcijos *onMouseOver* savybė *window.status*. Panaudojant funkciją *onMouseOut*, spausdinama tuščia eilutė, kuri ištrina ankstesnį užrašą:

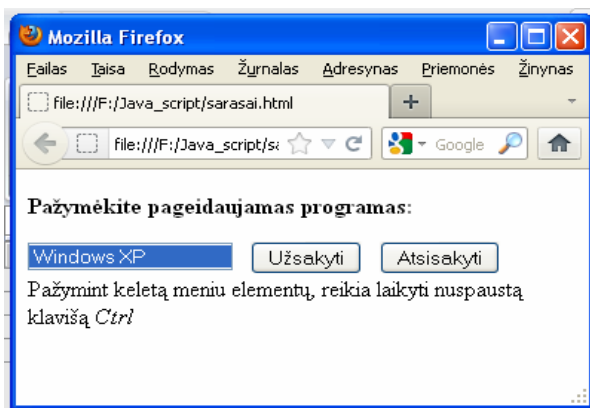
```
<html>
<head>
< title > Pavyzdys </title >
<script language= "JavaScript">
function doMouseOver()
window.status="Pereisite į Hotmail'o tinklą"
function doMouseOut()
window.status = " "
</script>
</head>
<body>
<a href= "http://www.hotmail.com " onMouseOver
="doMouseOver()" " onMouseOut= " doMouseOut()">
Pereikite į Hotmail 'a</a>
</body>
</html>
```

## 3.7. Išskleidžiamieji sąrašai

Programų teikiamoms paslaugoms parinkti naudojama dar viena populiari priemonė – išskleidžiami sąrašai, kurie dar vadinami parinkimo meniu. Pagrindinis jų privalumas yra tas, kad jie ekrane užima mažai vietos. Visos sąrašo savybės aprašomos konteinerio *select* pradinėje gairėje, kurioje galima nurodyti sąrašo vardą (*name*), identifikatorių (*id*), dydį (*size*) ir reikšmių parinkimo būdą – pavienį arba grupinį. Tipinis yra pavienis sąrašo elementų parinkimo būdas,

kai pelės spragtelėjimu galima parinkti tik tai vieną elementą. Tokio parinkimo būdo specialiai prašyti nereikia. Kai leidžiama kartu parinkti kelis elementus, gairėlėje *select* įrašomas atributas *multiple*, o elementų grupė parenkama spragtelint juos pele esant nuspaustam klavišui *<Ctrl>*. Jei šis klavišas nepaspaustas, net ir įrašius atributą *multiple* bus leidžiama pasirinkti tik tai vieną elementą, o parenkant kiekvieną naują elementą, ankstesnis bus grąžinamas į pasyvią būseną.

Sąrašo dydį aprašančio parametro *size* reikšmė nurodo, kiek jo elementų (eilučių) vienu metu rodoma ekrane. Kai leidžiama parinkti tik tai pavienius elementus, greta sąrašo elemento tinklalapyje sukuriamas sąrašo išskleidimo mygtukas ▼. Spragtelėjus pele šį mygtuką, parodomas visas sąrašas, kuriame pasirinktas elementas pažymimas pakeista fono spalva. Jei leidžiamas grupinis parinkimas, vietoje išskleidimo mygtuko sukuriami du sąrašo slinkties valdymo mygtukai, parodyti 3.7.1 pav.



3.7.1 pav. Išskleidžiamasis sąrašas

Atskirų meniu elementų paskirtys aprašomos konteineriuose *option*, o pelės spragtelėjimu parinkus elementą *j* serveri siunčiama pradinėje jo konteinerio gairėlėje *option* nurodyta parametro *value* reikšmė. Pradinis sąrašo elementų parinkimas aprašomas parametru *selected*.

Pavyzdžiui, 3.7.1 pav. rodomas programų parinkimo sąrašas aprašomas taip:

```
<html>
<head/>
<body>
  <form name="Prekyba">
    <p style="font-weight: bold">
      Pažymėkite pageidaujamas programas:</p>
    <select size="1" name="Parinkta" multiple>
      <option value="WinXP" selected>Windows XP</option>
      <option value="Office">MS Office XP</option>
      <option value="Adobe"> Adobe Acrobat 5.5</option>
    </select>&nbsp;
    <input type="submit" value="Užsakyti">&nbsp;
    <input type="reset" value="Atsisakyti">&nbsp;
    Pažymint keletą meniu elementų, reikia laikyti nuspaustą klavišą
```

```

        <span style="font-style:italic">Ctrl</span><br>
    </form>
</body></html>

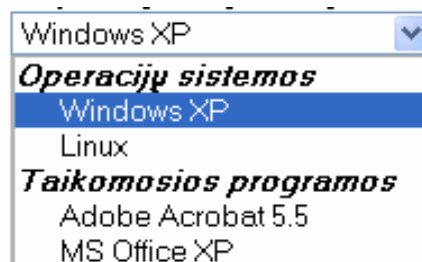
```

Tinklalapio lankytojo parinkto sąrašo elemento indeksą nurodo meniu objekto savybė *selectedIndex*. Visiems konteineryje *select* aprašytiems elementams suteikiamos nuosekliai didėjančios indeksų reikšmės, pradedant nuo nulio. Kai parinktų elementų nėra, savybės *selectedIndex* reikšmė yra *-1*. Tai galima naudoti privalomo meniu elementų parinkimo kontrolės funkcijoje, kurios reikšmė *true* praneša, kad parinktas bent vienas elementas, o reikšmė *false* nurodo, kad tokių elementų nėra:

```

function Tikrinti(){
    if (Prekyba.Parinkta.selectedIndex<0){           // Ar yra parinktų elementų?
        alert("Parinkite bent vieną programą");
        Prekyba.Parinkta.focus();                   // Sąrašo objekto aktyvavimas
        return false;                               // Reikšmė, kai parinkimo nėra
    }
    else return true;                               // Reikšmė, kai yra parinktų
                                                elementų
}

```



3.7.2 pav. Išskleidžiamasis sąrašas su dviem elementų grupėmis

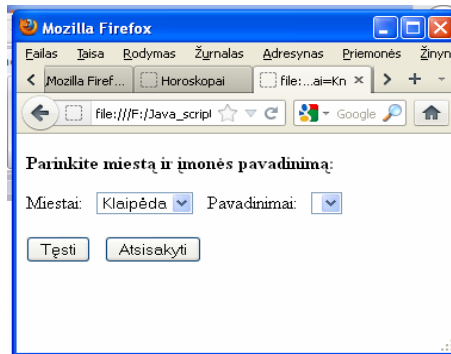
Ilgus parinkimo sąrašus naudoti patogiau, kai jų elementai sugrupuojami ir tokioms grupėms suteikiamos antraštės, kurios atspindi bendras grupės savybes. Grupėms sudaryti naudojami konteineriai *optgroup*, o jų antraštes nurodo parametro, *label* reikšmės. Pavyzdžiui, 3.7.2 pav. parodytas sąrašas su dviem elementų grupėmis aprašomas taip:

```

<select size="1" name="Parinkta">
  <optgroup label="Operacijų sistemos">
    <option value="WinXP">Windows XP</option>
    <option value="Linux">Linux</option></optgroup>
  <optgroup label="Taikomosios programos">
    <option value="Adobe"> Adobe Acrobat 5.5</option>
    <option value="Word"> MS Office XP</option></optgroup>
</select>

```

Dažnai tinklalapiuose pateikiami parinkimo sąrašai, kuriuose rodomi duomenys priklauso nuo lankytojo įvestų duomenų arba atliktų veiksmų. Pavyzdžiui, toks tinklalapis parodytas 3.7.3 pav.



3.7.3 pav. Parinkimo sąrašas

Parinkus jo sąrašė *Miestai* konkretų miesto pavadinimą, gretimame sąrašė *Pavadinimai* pateikiamas tame mieste esančių įmonių sąrašas. Šio tinklalapio aprašymas yra toks:

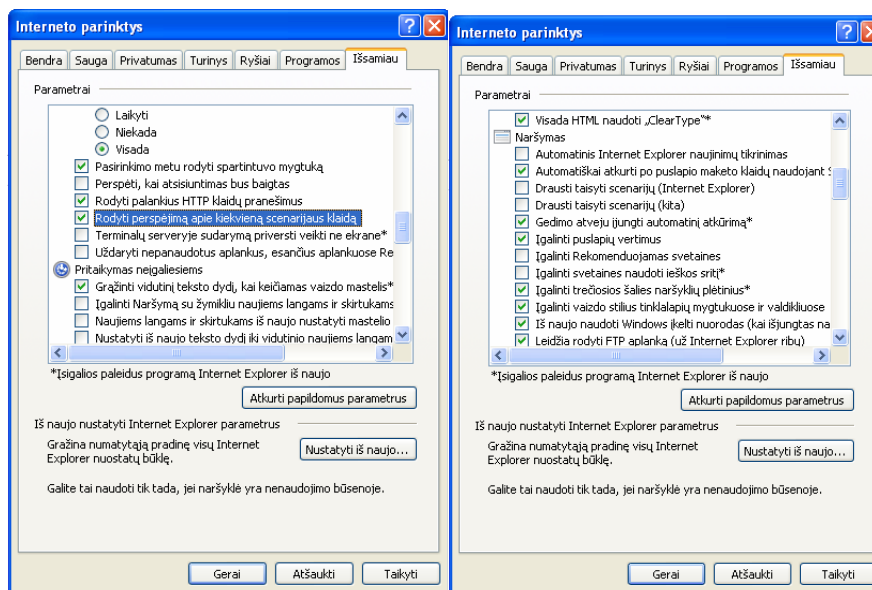
```
<html>
<head>
<script language="JavaScript">
    var mas1 = new Array ('Liteksas', 'Drobė','Audimas');
    var mas2 = new Array ('Burė', 'Gija');
    var mas3 = new Array ('Gaja', 'Vilma', 'Goda');
    var y = new Array (mas1, mas2, mas3);
    var k = 0;
    function Rodyti(vardas){
        Prekyba.Pavadinimai.length= vardas.length;
        for (i=0; i< vardas.length; i++)
            Prekyba.Pavadinimai.options[i]=new Option(vardas[i], vardas[i], 0, 0);}
    function Keisti() {
        k= Prekyba.Miestai.selectedIndex; Rodyti(y[k]);}
</script></head>
<body onLoad = "Rodyti(y[k])" >
<form name="Prekyba" >
<p style= "font-weight: bold">
Parinkite miestą ir Įmonės pavadinimą:</p>
Miestai: &nbsp;
<select size="1" name="Miestai" onChange="Keisti()">
    <option value="Kn" selected>Kaunas</option>
    <option value="Kl">Klaipėda</option>
    <option value="Vl"> Vilnius</option>
</select>&nbsp;
Pavadinimai: &nbsp;
<select size="1" name="Pavadinimai">
</select><br><br>
<input type="submit" value="Tęsti">&nbsp;
<input type="reset" value="Atsisakyti">
</form>
</body></html>
```

Aprašant šio tinklalapio parinkimo meniu tvarkymą, naudojama keletas anksčiau neaptartų programavimo elementų. Įmonių pavadinimams saugoti čia sudarytas dvimatis masyvas  $y$ . Tiesiogiai tokio masyvo *JavaScript* kalba aprašyti neleidžiama. Todėl masyvas  $y$  aprašytas kaip masyvas, kurio elementai taip pat yra masyvai: *mas1*, *mas2* ir *mas3*. Šiems pavadinimų masyvams indeksai parinkti taip, kad jie atitiktų miestų sąrašo indeksus: Kaunas – 0, Klaipėda – 1, Vilnius – 2. Tinklalapio konteineryje *body* aprašytas tuščias sąrašas *Pavadinimai*, kuriame nėra parenkamų elementų. Šį sąrašą užpildo funkcija *Rodyti*, kurios argumentas perduoda jame rašomų pavadinimų masyvą. Iš pradžių sąrašo užpildymą organizuoja tinklalapio atvėrimo įvykis *onLoad*, kuris aprašytas gairėlėje *<body>* ir nukreipia į pavadinimų sąrašą Kauno miesto įmonių pavadinimus. Pakeitus miestų sąrašą parinktą reikšmę gairėlėje *select* aprašytas įvykis *onChange* iškviečia funkciją *Keisti*, kuri pakeičia kintamojo  $k$  saugomo aktyvaus miestų sąrašo elemento indeksą ir perduoda funkcijai *Rodyti* kito miesto įmonių pavadinimų masyvą. Sąrašo papildymą nauju elementu aprašo tokia sintaksinė struktūra:

*Sąrašo vardas.options[Indeksas] =*  
*new Option(Tekstas, Reikšmė, Aktyvumas, Aktyvumas papildant);*

### 3.8. Intarpų funkcijų derinimas

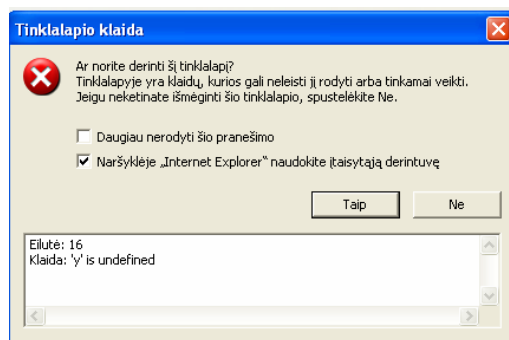
Parengiant bent kiek sudėtingesnes intarpų funkcijas, sunku išvengti klaidų. Todėl, rašant tokias funkcijas, reikia mokėti naudoti jų derinimo priemones. Funkcijų (programų) derinimu vadinamas jose esančių klaidų paieškos ir ištaisymo procesas. Jei skriptams derinti norima naudoti naršyklę, visų pirma reikia įsitikinti, kad joje įdiegtos derinimo priemonės. Pavyzdžiui, naršyklėje *MS Explorer* tai galima sužinoti atvėrus pagrindinio jos meniu komandos *Įrankiai* → *Interneto parinktys* lango kortelę *Išsamiau*, kuri parodyta 3.8.1 pav. Derinant skriptus būtina, kad būtų pažymėtas langelis *Rodyti perspėjimą apie kiekvieną scenarijaus klaidą* ir kad nebūtų žymės langelyje *Drausti taisyti scenarijų* (*Internet Explorer*).



3.8.1 pav. Derinimo priemonių įdiegimas

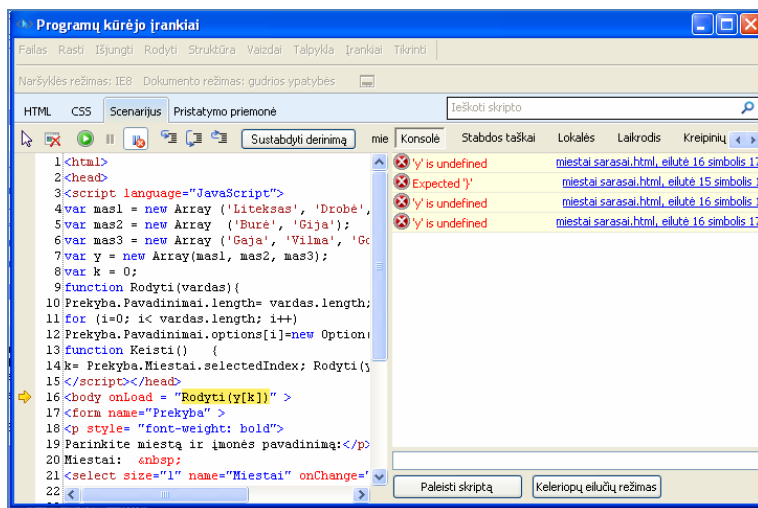
Intarpų funkcijų aprašymuose daromos klaidos būna trijų tipų: sintaksės, vykdymo ir loginės. Sintaksės klaidų atsiranda dėl to, kad neteisingai aprašomos programavimo kalbos struktūros. Jas automatiškai aptinka į naršyklės įkomponuoti programavimo kalbos interpretatoriai. Jų formuojami pranešimai apie klaidas ir atveriami langai įvairių tipų ir atmainų naršyklėse skiriasi, tačiau pagrindiniai klaidų paieškos principai labai panašūs.

Pavyzdžiui, tarkime, kad parengtas tinklalapio aprašymas, kurio programiniame intarpe pamiršta operatoriaus *if* loginę sąlygą įrašyti skliaustuose. Tai sintaksės klaida, kurią naršyklė aptiks tinklalapio atidarymo metu ir parodys ekrane pranešimą apie spėjamą klaidos pobūdį bei pasiūlys tęsti klaidos paiešką (3.8.2 pav.):



3.8.2 pav. Informavimas apie klaidą

Jeigu šiame pranešime pateiktos informacijos pakanka, parinkus naršyklės komandą galima atverti tinklalapio aprašymą *HTML* kalba tekstų rengyklės *Notepad* lange, jį pataisyti ir vėl bandyti atverti naršyklės lange. Norint pasinaudoti specialiomis skriptų derinimo priemonėmis ir gauti išsamesnę informaciją apie klaidas, pranešimo apie klaidą lange reikia paspausti mygtuką *Taip* ir bus atvertas derinimo programų pasirinkimo langas (3.8.3 pav.):



3.8.3 pav. Programų pasirinkimo langas

Nors intarpo aprašyme sintaksės klaidų ir neaptinkama, garantijos, kad jis teisingas, nėra. Dar gali būti vykdymo ir loginių klaidų. Vykdyto klaidų pasitaiko tada, kai pageidaujama atlikti

neleistinus veiksmus, pavyzdžiui, dalyti iš nulio arba sudauginti tekstus. Tokias klaidas ir apytikslę jų vietą taip pat leidžia aptikti automatinio derinimo priemonės.

Sunkiausiai aptinkamos loginės klaidos, kurios atsiranda dėl to, kad intarpo autorius netinkamai parinko uždavinio sprendimo būdą arba atsižvelgė ne į visas galimas situacijas. Tokiais atvejais automatinės derinimo priemonės klaidų neaptinka, tačiau sprendimo rezultatai būna ne tokie, kokių tikimasi. Šias klaidas galima aptikti tikrai organizuojant intarpų testavimą, juose aprašytų procesų bandomąjį vykdymą. Intarpų testavimas atliekamas esant įvairioms jų apdorojamų duomenų reikšmėms, stengiantis sukurti kuo sunkesnes tikrinamos programos darbo sąlygas.

Pastebėjus klaidas, stengiamasi analizuoti jų priežastis. Tai padaryti padeda tarpinių rezultatų parodymas, programos fragmentų išjungimas, paverčiant juos komentarais arba įrašant funkcijų nutraukimo operatorių *return*. Testavimo metodika ir naudojamos priemonės priklauso nuo konkrečių tikrinamų programų atliekamų veiksmų, todėl ir nėra griežtų rekomendacijų, kaip testuoti programas.

## 4. PRAKTINĖS UŽDUOTYS

### 4.1. Praktinė užduotis. HTML pagrindai

**Darbo tikslas:** pagilinti darbo su *HTML* kalba pagrindus.

HTML dokumento struktūra:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<!-- Naudojamo HTML standarto aprašymas -->
<HTML><!-- Dokumento tipas-->
<HEAD><!-- Dokumento antraštė -->
<TITLE>Pavadinimas</TITLE>
...kiti antraštės elementai:
</HEAD>
<BODY>...dokumentas </BODY>
</HTML>
```

Dažniausiai naudojami komandų atributai:

*ALIGN* – elemento išdėstymo orientacija kitų HTML dokumento elementų atžvilgiu arba HTML dokumente.

*HREF* – nuoroda į failą.

*SRC* – failas, kuriame yra informacija, reikalinga elementui sukurti.

*TARGET* – nuoroda į rėmelių dalį.

*BORDER* – rėmelių aplink objektą storis.

*WIDTH* – elemento plotis.

*HEIGHT* – elemento aukštis.

*HSPACE* – horizontalus tarpas tarp elemento ir kitų HTML dokumento elementų.

*VSPACE* – vertikalus tarpas tarp elemento ir kitų HTML dokumento elementų.

*ALT* – tekstas, rodomas vietoje arba šalia elemento.

*BGCOLOR* – elemento fono spalva.

Antraštės elementai:

*<TITLE>Pavadinimas</TITLE>* – Dokumento pavadinimas

*<SCRIPT LANGUAGE="language" SRC="url">...</SCRIPT>* –

Skriptai, hierarchinės stilių lentelės ir t. t. Atributai:

*LANGUAGE* – nurodo skripte naudojamą programavimo kalbą.

*SRC* – nurodo failą, kuriame yra naudojamas skriptas.

*<STYLE TYPE="text/css" TITLE="BrightColours">...</STYLE>*

Naudojamų stilių lentelė. Atributai:

*TITLE* – pavadinimas.

*TYPE* – informacijos tipas.

*<BASE HREF="http://www.myhost.com/" TARGET="default\_target">*



Bazinė nuoroda ir rėmelių dalis.

*<META HTTP-EQUIV="type" NAME="name" CONTENT="info">*

Svarbi informacija apie HTML dokumentą, neperduodama kitu būdu. Atributai:

NAME – informacijos pavadinimas, tipas.

HTTP-EQUIV – informacijos tipas, naudojamas perduodant informaciją HTTP protokolu, gali pakeisti NAME.

CONTENT – pati perduodama informacija.

*<LINK REL="relation" TYPE="doctype" HREF="url" TITLE="name">*

Nurodo ryšį tarp dokumentų. Atributai:

REL – nurodo, kokio tipo yra ryšys.

TYPE – nurodo, kokio tipo yra dokumentas.

Dokumento formavimo elementai:

*<BODY BACKGROUND="image" BGCOLOR="#rrggbb|colour name"*

*TEXT="#rrggbb|colour name" LINK="#rrggbb|colour name" VLINK="#rrggbb|colour name"*

*ALINK="#rrggbb|colour name"> ... </BODY>*

Dokumento turinį „gaubianti“ komanda. Atributai:

BACKGROUND – foninio grafinio vaizdo URL.

TEXT – teksto spalva dokumente, jei nėra nurodyta kita.

LINK, VLINK, ALINK – atitinkamai nelankytų, lankytų ir aktyvių nuorodų spalvos.

*<H? ALIGN="left|center|right"> ... </H?>*

Pavadinimai, ?=(1-6).

*<P ALIGN="left|center|right"> ... </P>*

Pastraipa.

*<DIV ALIGN="left|right|center"> ... </DIV>*

Teksto orientavimas horizontaliai.

*<CENTER> ... </CENTER>*

Centruojamas tekstas.

*<BR CLEAR="left|right|all">*

Perėjimas į naują eilutę. Atributai:

CLEAR – naujos eilutės pradžios atskaitos taškas.

*<NOBR>... </NOBR>*

Neperkeliamas tekstas

*<WBR>*

Teksto galimo perkėlimo vieta.

*<A HREF="URL" TARGET="name|\_self|\_parent|\_top|\_blank" NAME="name"*

*TITLE="name"> ... </A>*

Nuoroda. Atributai:

TARGET – nuorodos rėmelių patalpinimo langas: nurodytasis, tas pats, kuriame yra dokumentas, motininio dokumento langas, visas peržiūros programos langas, naujas langas.

NAME – galimos nuorodos vardas.

*<IMG SRC="url" SRC="url" ALT="name"*

*ALIGN="left|right|texttop|absmiddle|baseline|absbottom" BORDER="pixels" WIDTH="pixels"*

*HEIGHT="pixels" HSPACE="pixels" VSPACE="pixels" LOWSRC="url" USEMAP="url"*

*ISMAP>*

Grafinis vaizdas. Atributai:

USEMAP – nuoroda į žemėlapi.

ISMAP – grafinis vaizdas yra žemėlapis.

LOWSRC – nuoroda į prastės kokybės paveiksluko kopiją.

<PRE WIDTH="simbcount">... </PRE>

Neformatuotinas tekstas.

<BLOCKQUOTE> ... </BLOCKQUOTE>

Citata.

<ADDRESS> ... </ADDRESS>

Kontaktinė informacija apie autorių ir tinklalapį.

<HR ALIGN="left|right|center" SIZE="pixels" WIDTH="pixels|%" NOSHADE  
COLOR="#rrggbb|colour name">

Linija. Atributai:

NOSHADE – „neįspausta“ linija, be šešėlio.

<COMMENT> ... </COMMENT>, <!-- ... -->

Komentarai.

Loginis teksto formatas:

<EM> ... </EM>

Poligrafinis kursyvas.

</STRONG> ... </STRONG>

Paryškintas poligrafinis kursyvas.

<SAMP> ... </SAMP>

Programos išvedimai, skriptai.

<VAR> ... </VAR>

Kintamieji, komandų argumentai.

<CITE> ... </CITE>

Citata, nuoroda į literatūrą.

<DFN> ... </DFN>

Apibrėžimas.

Fizinis teksto formatas:

<TT> ... </TT>

Teletaipo (*monospace*) tekstas.

<I> ... </I>

Pasviręs tekstas (*italic*).

<B> ... </B>

Paryškintas tekstas.

<U> ... </U>

Pabrauktas tekstas.

<STRIKE> ... </STRIKE>

Perbrauktas tekstas.

<BIG> ... </BIG>

Didesnis už aplinkinį tekstas.

<SMALL> ... </SMALL>

Mažesnis už aplinkinį tekstas.

<SUB>...</SUB>

Apatinis indeksas (*subscript*).

<SUP>...</SUP>

Viršutinis indeksas (*superscript*).

<BLINK>...</BLINK>

Mirksintis tekstas.

<FONT SIZE="[+|-]value" COLOR="#rrggbb|colour name"  
FACE="name[,name]...">...</FONT>

Šriftas. Atributai:

SIZE – šrifto dydis, gali būti matuojamas reliatyviai esamo šrifto atžvilgiu.

FACE – šrifto pavadinimas. Naudojamas pirmas sąrašas esantis šriftas.

<BASEFONT SIZE="value" COLOR="#rrggbb|colour name" FACE="name[,name]...">

Bazinis šriftas, kuris naudojamas viso HTML dokumento ribose. Dydis naudojamas kaip pagrindas reliatyviam kitų šriftų dydžiui skaičiuoti.

Struktūriniai HTML elementai:

<MAP NAME="name">

< AREA SHAPE="rect|circle|poly" COORDS="x1,y1,x2,y2,..."  
TARGET="name|\_self|\_parent|\_top|\_blank" HREF="url"|NOHREF">

Kitos AREA komandos.

</MAP>

Grafinis žemėlapis. AREA – sritys su skirtingomis nuorodomis. Atributai:

SHAPE – srities tipas: stačiakampis, apskritimas, daugiakampis.

COORDS – sritį apibrėžiančios koordinatės. Atskaitos taškas: grafinio paveiksluko viršutinis dešinysis kampas. Reikia apibrėžti: stačiakampiui – keturių viršūnių koordinates, apskritimui – centro taško koordinates ir spindulio ilgį, daugiakampiui – viršūnių koordinates.

NOHREF – srityje neveikia jokia nuoroda.

<BG SOUND SRC="url" LOOP="n">

Garsas, skambantis fone. Failų formatai WAV, AU arba MIDI. Atributai:

LOOP – failo kartojimų skaičius. -1 arba *infinite* reiškia „amžiną“ kartojimą.

<EMBED SRC="url" BORDER="pixels" WIDTH="pixels" HEIGHT="pixels"  
HSPACE="pixels" VSPACE="pixels">

Tiesioginis objektų (pvz.: audio ir video failų) įterpimas į HTML dokumentą. Reikalingi priedai ar programos, „suprantančios“ nurodyto failo formatą.

<MARQUEE ALIGN="left|right|top|middle|bottom" BEHAVIOR="scroll|slide|alternate"  
BG COLOR="#rrggbb|colour name" DIRECTION="left|right" HEIGHT="value|value%"  
WIDTH="value|value%" HSPACE="value" VSPACE="value" LOOP="value|–  
1|infinite" SCROLLAMOUNT="value" SCROLLDELAY="value">...</MARQUEE>

Judantis tekstas. Atributai:

BEHAVIOR – judėjimo pobūdis. Tekstas gali judėti nuo vieno peržiūros lango krašto iki kito, judėti „sujungus“ priešingus teksto galus, judėti tik nuo vienos *marquee* lango krašto iki kito.

DIRECTION – judėjimo kryptis.

LOOP – judėjimo ciklą kiekis. Pagal nutylėjimą – begalinis.

SCROLLAMOUNT – taškų kiekis, per kurį pasislenka tekstas.

SCROLLDELAY – „sustojimo“ laikas milisekundėmis tarp kiekvieno teksto pasislinkimo.

```
<APPLET CODEBASE="url" CODE="appletFile" ALT="alternateText"
NAME="appletInstanceName" WIDTH="pixels" HEIGHT="pixels"
ALIGN="left|right|texttop|absmiddle|baseline|absbottom" VSPACE="pixels" HSPACE="pixels"
ARCHIVE="url" MAYSCRIPT>
<PARAM NAME="appletAttribute1" VALUE="value">
```

Kitos PARAM komandos

Alternatyvus HTML komandų rinkinys

```
</APPLET>
```

Java programėlė. Atributai:

CODEBASE – katalogas, kuriame yra programos kodas. Pagal nutylėjimą sutampa su dokumento katalogu.

CODE – failas, kuriame yra sukompiliuotas programos poklasis.

ARCHIVE – papildomų reikalingų failų vieta.

MAYSCRIPT – programa, pasiekama naudojant JavaScript.

```
<UL TYPE=DISC|CIRCLE|SQUARE COMPACT>;
```

```
<LI TYPE=DISC|CIRCLE|SQUARE>
```

Kitos LI komandos

```
</UL>
```

Nenumeruojamas sąrašas. Atributai:

TYPE – sąrašo elemento žymeklio tipas: skritulys, apskritimas, kvadratas.

COMPACT – pateikti sąrašą kompaktiškai.

```
<OL TYPE="A|a|I|i|I" VALUE="startingnumber" COMPACT>
```

```
<LI TYPE="A|a|I|i|I" VALUE="startingnumber">... Kitos LI komandos
```

```
</OL>
```

Numeruojamas sąrašas. Atributai:

TYPE – sąrašų numerių tipas: didžiosios raidės, mažosios raidės, didieji romėniški skaitmenys, mažieji romėniški skaitmenys, arabiški skaitmenys (pagal nutylėjimą).

COMPACT – pateikti sąrašą kompaktiškai.

```
<DL COMPACT>
```

```
<DT> ... <DD>
```

Kitos DT ir DD poros

```
</DL>
```

Apibrėžimų sąrašas. DT – apibrėžiamas terminas. DD – apibrėžimas.

COMPACT – pateikti sąrašą kompaktiškai.

```
<TABLE BORDER="value" CELLSPACING="value" CELLPADDING="value"
WIDTH="value%" HEIGHT="value%" ALIGN="left|right" VALIGN="top|bottom"
BGCOLOR="#rrggbb|colour name" BORDERCOLOR="#rrggbb|colour name"
BACKGROUND="URL">
```

```
<CAPTION ALIGN="top|bottom">...</CAPTION>
```

```
<TR ALIGN="left|right|center" VALIGN="top|middle|bottom|baseline"
BGCOLOR="#rrggbb|colour name" BORDERCOLOR="#rrggbb|colour name">
```

```
<TH|TD WIDTH="value%" HEIGHT="value%" ALIGN="left|center|right"
VALIGN="top|middle|bottom|baseline" NOWRAP COLSPAN="value" ROWSPAN="value"
BGCOLOR="#rrggbb|colour name" BORDERCOLOR="#rrggbb|colour name"
BACKGROUND="URL"> ... </TH|TD>
```

Kitos TH ar TD komandos

```
</TR>
```

Kitos TR komandos

*</TABLE>*

Lentelė:

CAPTION – lentelės antraštė, TR – lentelės eilutė, TH, TD – lentelės ląstelė, TH – labiau skirta pavadinimui. Atributai:

CELLSPACING – ląstelės dydis.

CELLPADDING – atstumas nuo ląstelės turinio iki jos krašto.

ALIGN (lentelei) – slankios lentelės padėtis dokumente.

VALIGN – lentelės ląstelių ar vienos eilutės, vienos ląstelės turinio vertikali orientacija.

BACKGROUND – nuoroda į lentelės ar ląstelės fono grafinio paveiksliuko failą.

BORDERCOLOR – lentelės, eilutės ar ląstelės rėmelių spalva.

NOWARP – ląstelės turinys neperkeliamas.

COLSPAN – ląstelės dalijimas į du stulpelius.

ROWSPAN – ląstelės dalijimas į dvi eilutes.

Formos:

*<FORM ACTION="url" METHOD="get|post" ENCTYPE="MIME type"*

*TARGET="name|\_self|\_parent|\_top|\_blank " >...</FORM>*

HTML Forma, skirta formos elementuose surinktai informacijai nurodytu adresu siųsti.

Atributai:

ACTION – failas, kuriam perduodama informacija.

METHOD – informacijos perdavimo metodas.

ENCTYPE – informacijos kodavimo metodas.

*<INPUT TYPE="text|password|checkbox|radio|image|hidden|submit|reset" NAME="name"*

*VALUE="value" ALIGN="left|right|texttop|absmiddle|baseline|absbottom" CHECKED*

*SIZE="value" MAXLENGTH="value" SRC="url" >*

Formos elementas. Išvaizda ir naudojimas priklauso nuo tipo. Atributai:

TYPE – elemento tipas: tekstinis laukas, slaptažodis, pasirinkimo elementas, išsirinkimo elementas, grafinis vaizdas, nematomas, paslėptas elementas, mygtukas informacijai perduoti į serverį, mygtukas visų formos elementų reikšmėms atnaujinti.

NAME – elemento vardas, perduodamas kaip informacijos pavadinimas į serverį.

VALUE – pradinė elemento reikšmė arba pavadinimai ant submit ir reset tipo mygtukų. Taip pat perduodama į serverį kaip reikšmė (tekstiniams laukams gali būti keičiama).

CHECKED – atributas, skirtas pasirinktiems radio ar checkbox elementams pažymėti (iš vienodo vardo (NAME) radio elementų (gali būti pasirinktas tik vienintelis).

SIZE – matomas tekstinių elementų dydis.

MAXLENGTH – maksimalus simbolių kiekis tekstiniuose elementuose.

SRC – nuoroda į grafinio elemento failą.

*<SELECT NAME="name" SIZE="number" MULTIPLE>*

*<OPTION VALUE="value" SELECTED>*

Kitos OPTION komandos

*</SELECT>*

Formos elementas, skirtas pasirinkimui iš keleto opcijų realizuoti. Atributai:

NAME – elemento vardas, perduodamas kaip informacijos pavadinimas į serverį.

*SIZE* – matomų opcijų kiekis.

*MULTIPLE* – galimybė pasirinkti ne vieną, o keletą opcijų.

*VALUE* – serveriui perduodama reikšmė pasirinkimo atveju.

*SELECTED* – nurodyta opcija bus pasirinkta pagal nutylėjimą.

`<TEXTAREA ROWS="value" COLS="value" NAME="name" WRAP="off|virtual|physical">`

Tekstas:

`</TEXTAREA>` Formos elementas, skirtas didelės apimties tekstui. Atributai:

*NAME* – elemento vardas, perduodamas kaip informacijos pavadinimas į serverį.

*ROWS* – matomų eilučių kiekis.

*COLS* – matomų stulpelių kiekis.

*WRAP* – žodžių perkėlimo į kitas eilutes tipas: eilutės perduodamos tiksliai taip, kaip buvo parašytos, prieš perdavimą eilutės visada sujungiamos į vieną, perkėlimas vykdomas automatiškai.

Rėmeliai:

`<FRAMESET ROWS="pixels|%" *" COLS="pixels|%" *" BORDER="pixels"`

`FRAMEBORDER="yes|no|0" BORDERCOLOR="#rrggbb|colour name">`

`<FRAME SRC="url" NAME="frame_name" SCROLLING="yes|no|auto" NORESIZE`

`FRAMEBORDER="yes|no|0" BORDERCOLOR="#rrggbb|colour name">`

...Alternatyvus HTML elementų rinkinys

`</NOFRAME>`

`</FRAMESET>`

Rėmeliai. *FRAME* komanda formuoja vieną jų dalį, *NOFRAME* dalis skirta rėmelių neatpažįstančioms peržiūros programoms. Atributai:

*ROWS* – peržiūros lango dalijimas į keletą horizontalių dalių. \* ženklas reiškia likutį.

*COLS* – peržiūros lango dalijimas į keletą vertikalinių dalių.

*FRAMEBORDER* – matomi ar nematomi rėmeliai.

*BORDERCOLOR* – rėmelių spalva.

*NAME* – rėmelių dalies vardas.

*SCROLLING* – liniuotė dešiniame rėmelio šone: visada yra, niekada nėra, pagal situaciją.

*NORESIZE* – uždraudimas keisti rėmelių dalies dydį.

## **Užduotis**

Sukurkite internetinę svetainę, vadovaudamiesi šiais reikalavimais:

1. Svetainę turi sudaryti ne mažiau kaip 2 tinklalapiai.
2. Tinklalapiai tarpusavyje turėtų būti sujungti saitais (nuorodomis).
3. Svetainėje turi būti:
  - a) įterpti bent du paveikslai ir kelios horizontalios linijos;
  - b) bent keli objektai (pavyzdžiui, paveikslai) su paaiškinimais (atributas *title*);

- c) sukurtos kelios nuorodos (saitai) į kitas svetaines, konkrečius to paties ar kito svetainės tinklalapio objektus ir bent vienas el. pašto adreso saitas;
- d) sukurtas bent vienas žymėtasis ir bent vienas numeruotasis sąrašas (reikia panaudoti įvairius žymėjimo ir numeravimo stilius);
- e) įterptos bent dvi lentelės;
- f) sukurta forma.

## 4.2. Praktinė užduotis. PHP pradmenys, PHP kintamieji

**Darbo tikslas:** įgyti *PHP* kalbos pradmenis, susipažinti su pagrindinėmis struktūromis, susipažinti su *PHP* kalbos kintamaisiais, juos aprašyti ir naudoti.

**Užduotis.** Kintamieji naudojami informacijai saugoti: pavyzdžiui, teksto eilutėms, skaičiams ar matricoms. Jei kintamasis yra deklaruojamas, jį galite kaskart naudoti savo scenarijuje. *PHP* kintamieji pradedami \$ ženklu/simboliu.

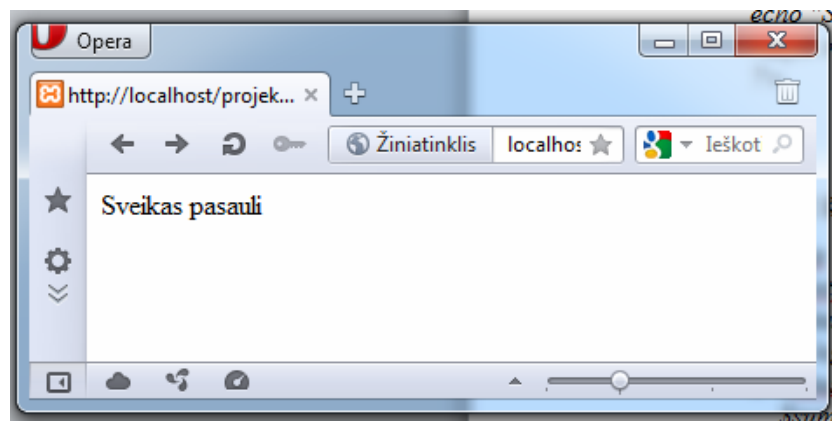
Pirmoji programėlė:

```
<?php
    $kintamasis = reikšmė;
?>
```

Išnagrinėkite ir praktiškai pritaikykite pavyzdžius:

**Pvz. 1.** Pirmoji programėlė (4.2.1 pav.):

```
<?php // PHP failo pradžia.
echo "Sveikas pasauli!"; //Leidimas rašyti tekstą, bei tekstas "Sveikas pasauli!" galite
rašyti ir kitokį tekstą.
?>
```



4.2.1 pav. Pirmoji programėlė

**Pvz. 2.** Kintamųjų priskyrimas:

```
<?php
    $kint = 5; // kintamasis $kint priskiriamas skaičiui 5
    $kitas_kint = 6; // kintamasis $kitas_kint priskiriamas skaičiui 6
```

```

/*loginė užklausa, priskyrimas. Kintamasis $suma buvopriskirtaskintamiesiems $kintir $kitas_kint, kur $kint buvopridėtas priekintamojo $kitas_kint*/
$suma = $kint + $kitas_kint;
echo $suma; // lange išveda: 11
?>

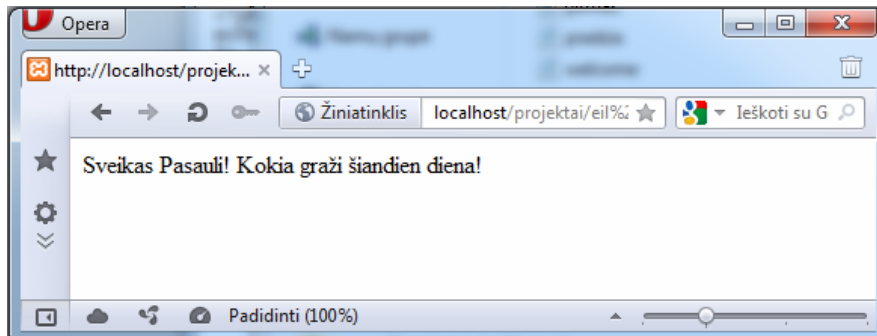
```

**Pvz. 3.** Sąryšio operatorius (žymimas `.`) – tašku). Jis yra naudojamas eilutėms sujungti, tarkime, 2 kintamiesiems (4.2.2 pav.):

```

<?php
$tekstas1 = "Sveikas Pasauli!";
$tekstas2 = "Kokia graži šiandien diena!";
echo $tekstas1 . " " . $tekstas2;
?>

```



4.2.2 pav. Sąryšio operatorius

**Pvz. 4.** Matematiniai operatoriai:

```

?php
//1
echo mano ; $a="1"; echo "yra $a mas<br>"; echo "yra $amas <br>"; echo "yra { $a } mas<br>";

//2
echo $b="100"."<br>";
echo $b[0]. "<br>";
echo $b[strlen($b)-2];
//3
$a = 10;
$b = 2;
$c = 3;
$sum = $a + $b * $c;
echo($sum);
echo $a + $b * $c ;
echo 10 + 2 * 3;
?>

```

**Pvz. 5.** Sąlygos sakiny (4.2.3 pav.):

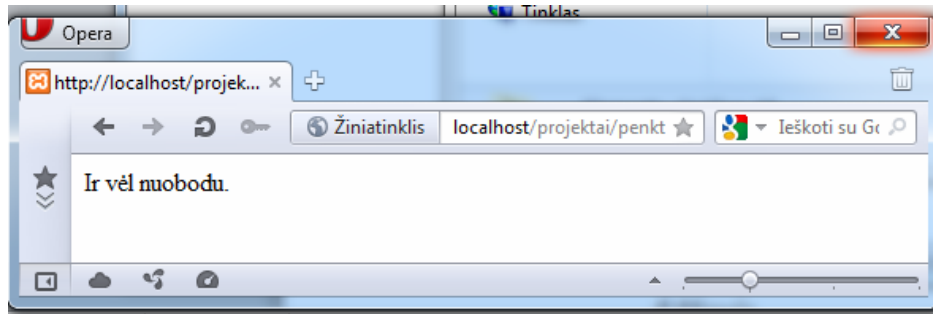
```

<?php
$date = date("D");
if ($date == "Fri")
    echo "Juk Penktadienis. Pašėlkime!";
else

```



?> *echo "Ir vėl nuobodu.";*



4.2.1 pav. *Sąlygos sakiny*s

Naudodamiesi pateiktais pavyzdžiais, sukurkite programas:

1. Parašykite kodą, kuris išvestų į ekraną jūsų pavardę ir vardą.
2. Parašykite kodą, kuris sujungtų dvi duotas eilutes.
3. Parašykite kodą, kuris iš dviejų duotų skaičių išrinktų ir išvestų į ekraną didžiausią.
4. Parašykite kodą, kuris išvestų į ekraną einamąją datą lietuviškai.

### 4.3. Praktinė užduotis. PHP masyvai. PHP ciklai, eilutės

**Darbo tikslas:** įgyti darbo su *PHP* masyvais ir ciklais įgūdžių.

*PHP* masyvas yra trijų rūšių:

- skaitmeninis masyvas (*NumericArray*) – masyvas su numeruotais indeksais;
- asociatyvus masyvas (*AssociativeArray*) – masyve, kur *ID* raktas yra asociatyvus su reikšme;
- daugiamatis masyvas (*MultidimensionalArray*) – tarp masyvo reikšmės gali būti dar vienas masyvas, kuris sudaro „ilgą medį“.

Ciklo sakiniai *PHP* kalboje gali būti:

- *while* – ciklas tikrina sąlygas prieš arba po kiekvieno ciklo pakartojimo ir kartoja ciklą tik tuo atveju, jei sąlyga vis dar netenkinama. Sintaksė:

```
while (sąlyga)  
{  
    kodas, kuris bus vykdomas;  
}
```

- *do...while* – ciklas kartojamas bent vieną kartą, tada pereina į ciklą, kuris kartojamas, kol sąlyga bus teisinga. Sintaksė:

```
do
{
    kodas, kuris yra vykdomas;
}
while (sąlyga);
```

- *for* – ciklo sintaksė šiek tiek sudėtingesnė, nei *while*. Bet kartais naudoti *for* daug naudingiau. Sintaksė:

```
for (ciklo_parametras; sąlyga; parametro_didinimas/mažinimas)
{
    kodas, kuris bus vykdomas;
}
```

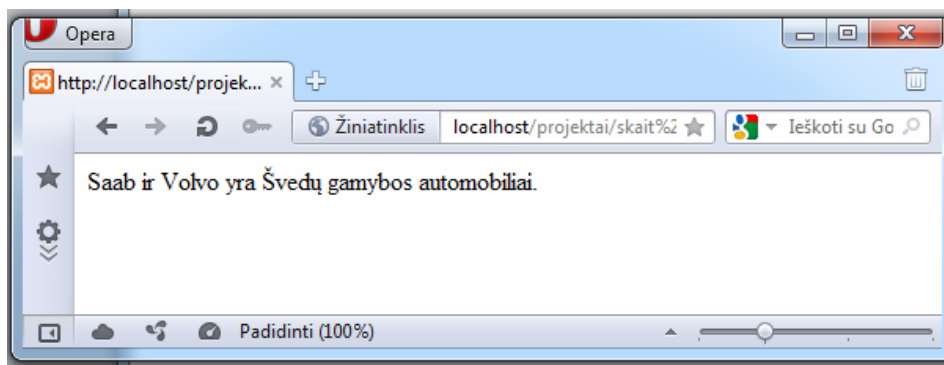
- *foreach* – ciklo kodas naudojamas kiekvieno masyvo elementams apimti. Sintaksė:

```
foreach ($masyvas as $reiksme)
{
    kodas, kuris bus vykdomas;
}
```

**Užduotis.** Išnagrinėkite pavyzdžius ir juos išbandykite:

**Pvz. 1.** Skaitmeninis masyvas (4.3.1 pav.):

```
<?php
    $auto[0] = "Saab";
    $auto[1] = "Volvo";
    $auto[2] = "BMW";
    $auto[3] = "Wollswagen";
    echo $auto[0] . " ir " . $auto[1] . " yra Švedų gamybos automobiliai.";
?>
```

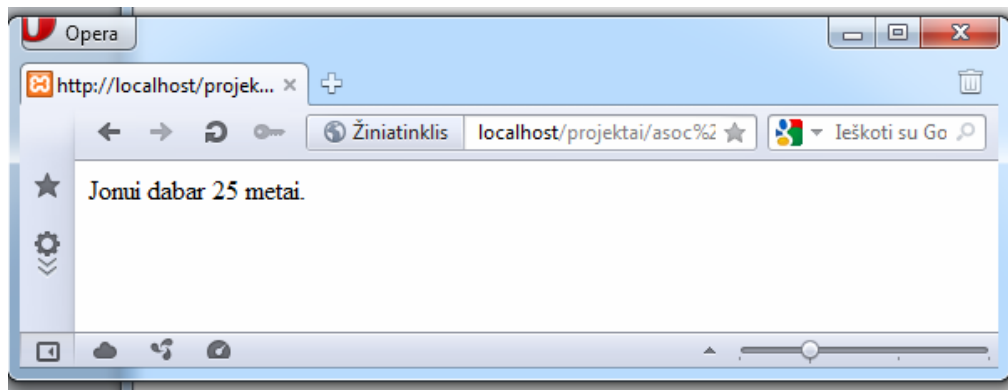


4.3.1 pav. Skaitmeninis masyvas

**Pvz. 2.** Asociatyvus masyvas (5.3.2 pav.):

```
<?php
    $metai["Jonas"] = "25";
    $metai["Petras"] = "27";
    $metai["Antanas"] = "36";

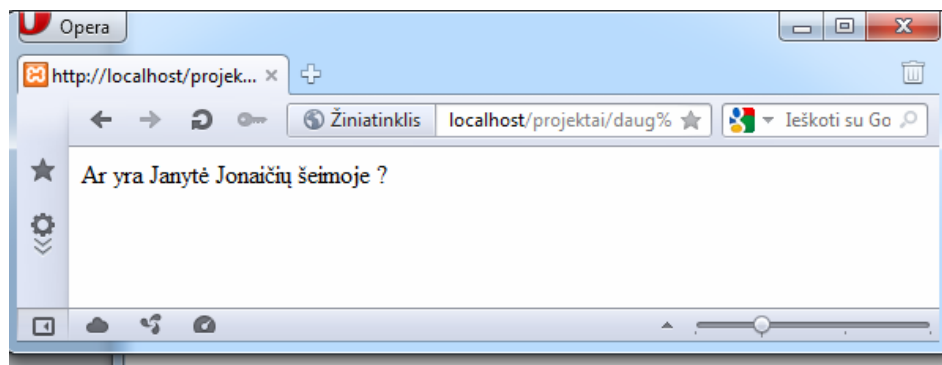
    echo "Jonui dabar " . $metai["Jonas"] . " metai.";
?>
```



4.3.2 pav. Asociatyvus masyvas

**Pvz. 3.** Daugiamatis masyvas (4.3.3 pav.):

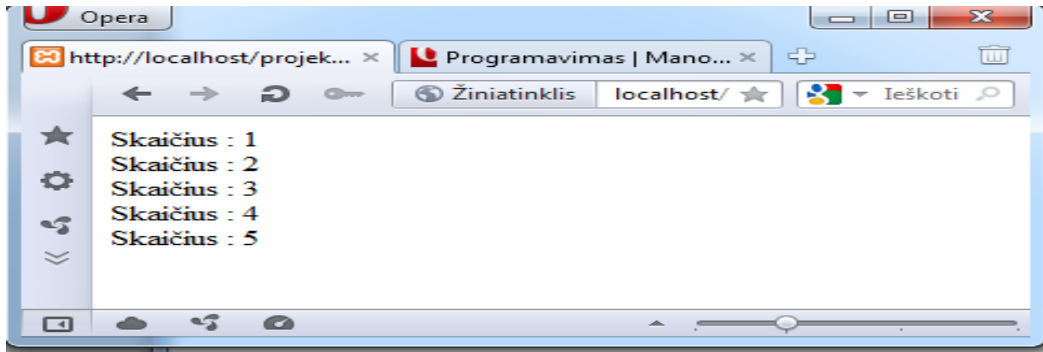
```
<?php
    $seima = array(
        "Jonaičiai" => array ("Jonas", "Janina", "Janytė"),
        "Petraičiai" => array ("Petras"),
        "Antanaičiai" => array ("Antanas", "Antanina"));
    echo "Ar yra " . $seima["Jonaičiai"][2] . " Jonaičių šeimoje ?";
?>
```



4.3.3 pav. Daugiamatis masyvas

**Pvz. 4.** Ciklas *While* (4.3.4 pav.):

```
<?php
    $i = 1;
    while ( $i <= 5 )
    {
        echo "Skaičius : " . $i . "<br />";
        $i++;
    }
?>
```



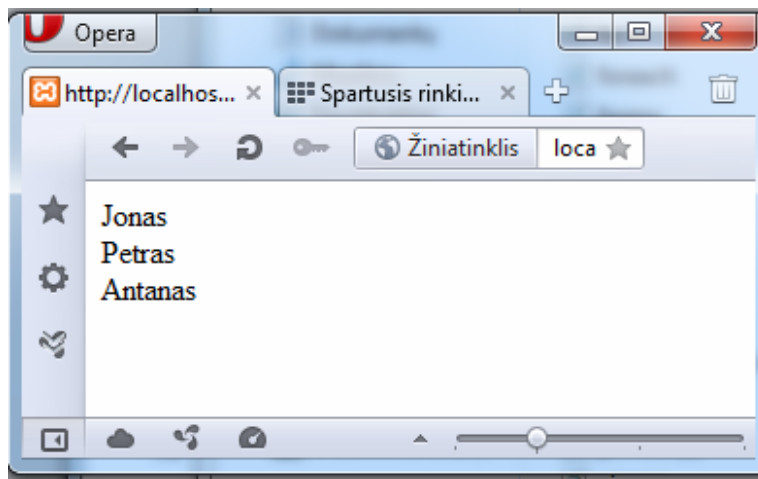
4.3.4 pav. Ciklas *While*

**Pvz. 5.** Ciklas *For* (rezultatas bus tas pats):

```
<?php
    for($i=0; $i <= 5; ++$i){
        echo $i . "<br />";
    }
?>
```

**Pvz. 6.** Ciklas *Foreach* (4.3.5 pav.):

```
<?php
    $duomenys = array("Jonas", "Petras", "Antanas");
    foreach ($duomenys as $vardas){
        echo $vardas . "<br>";
    }
?>
```



4.3.5 pav. Ciklas *Foreach*

**Pvz. 7.** Eilutės. Pavyzdžiai su eilutėmis:

```
<?php
echo substr('abcdef', 1);    // grazinabcdef
echo substr('abcdef', 1, 3); // grazinabcd
echo substr('abcdef', 0, 4); // grazinaabcd
echo substr('abcdef', 0, 8); // grazinaabcdef
echo substr('abcdef', -1, 1); //grazina f
// galim is eilutes pasiimti koki nors elementa kaip is masyvo:
$string = 'abcdef';
```

```

echo $string[0];           // grazina a
echo $string[3];           // grazina d
echo $string[strlen($string)-1]; // grazina f
?>

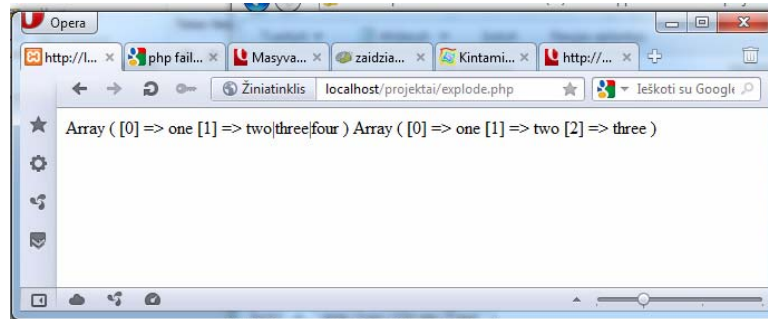
```

**Pvz. 8.** Masyvas (4.3.6 pav.):

```

<?php
$str = 'one|two|three|four';
// teigiamas limitas
print_r(explode('|', $str, 2));
// neigiamas limitas (nuo PHP 5.1)
print_r(explode('|', $str, -1));
?>

```



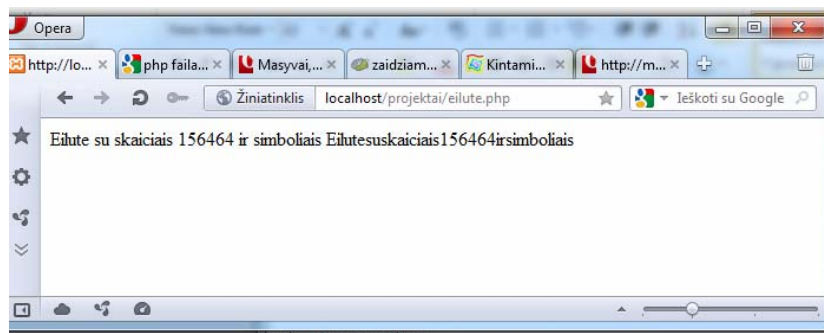
4.3.6 pav. Masyvas

**Pvz. 9.** Iš eilutės panaikinama viskas, išskyrus raides ir skaičius (4.3.7 pav.):

```

<?php
$string = "Eilute su skaiciais 156464 ir simboliais !@#$%^";
$new_string = preg_replace("/[^a-zA-Z0-9\s]/", "", $string);
echo $new_string;
$new_string2 = preg_replace("/[^a-zA-Z0-9]/", "", $string);
echo $new_string2;
?>

```



4.3.7 pav. Vaizdas naršyklėje

Naudodamiesi pateiktais pavyzdžiais, atlikite užduotis:

1. Duotas masyvas A, kuriame surašyti studentų egzaminų pažymiai. Suskaičiuokite studento pažymių vidurkį ir išveskite tik tuos pažymius kurie didesni už 7.
2. Duotas masyvas, kuriame surašyti krepšininkų ūgiai. Pašalinkite iš masyvo tas skaitines reikšmes, kurios mažesnės nei 1,75.
3. Duotas masyvas, kuriame surašyti studentų vardai. Surikiuokite vardus didėjimo tvarka.
4. Duota simbolių eilutė. Suskaičiuokite, kiek joje yra simbolių, ir raides pakeiskite į didžiąsias. Rezultatą išveskite į ekraną.
5. Klaviatūra įvedama simbolių eilutė, kur žodžiai skiriami bent vienu tarpu. Reikia išvesti eilutės žodžius po vieną ekrane, nurodant žodžio pradžios ir pabaigos indeksus.
6. Parašykite programą, kuri atskirtų klaviatūra įvestos eilutės žodžius ir paskirstytų į du masyvus: skaičių ir kitokių žodžių. Abiejų masyvų elementai ir skaičių masyvo suma turi būti parodomi ekrane.

#### 4.4. Praktinė užduotis. PHP skaitymas ir rašymas į/iš failus

**Darbo tikslas:** įgyti skaitymo ir rašymo į/iš failus įgūdžių.

**Užduotis.** Sukurti tekstinį failą *duom.txt*. Įrašykite į šį failą keletą teksto eilučių. Jei pirma ir paskutinė žodžio raidė sutampa, jis pakeičiamas simboliais „#“. Rezultatą įrašykite į failą *rez.txt* ir išveskite į ekraną (4.4.1 pav.).

Programos kodas:

```
<?php
function pakeist($zodis){
    $raide=preg_split('/', $zodis, -1, PREG_SPLIT_NO_EMPTY);
    $kiek_r=count($raide); //suskaičiuojam kiek raidžių yra žodyje
    if($raide[0]==$raide[$kiek_r-1]){ //jei pirma raidė tokia kaip paskutinė
        for($i=0; $i<$kiek_r; $i++) //generuojam eilutę su tiek pat # simbolių
            $raide[$i]="#";
        return implode(" ", $raide);
    }
    else //priešingu atveju, paprasčiausiai grąžinam žodį
        return $zodis;
    }
    $eilute=file('duom.txt'); //nuskaitom dokumento turinį į masyvą
    foreach($eilute as $i=>$zodziai){
        $zodziai=explode(" ", $zodziai); //kiekvieną eilutę suskaldom po žodį
        foreach($zodziai as $j=>$zodis)
            $zodziai[$j]=pakeist($zodis); //kiekvieną žodį tikrinam su funkcija pakeist()
        $eilute[$i]=implode(" ", $zodziai); //žodžius vėl sujungiam į eilutę
```

```

}
$paakeista=implode("", $eilute);
print("<pre>$paakeista</pre>"); //paakeistą eilutę atspausdiname ekrane
$doc=fopen("rez.txt", "w"); //ir įrašome į rezultatų failą
fwrite($doc, $paakeista);
fclose($doc);
?>

```

Funkcija *file()* grąžina dokumento turinį kaip masyvą, o kiekvieną dokumento eilutę kaip atskirą masyvo elementą.

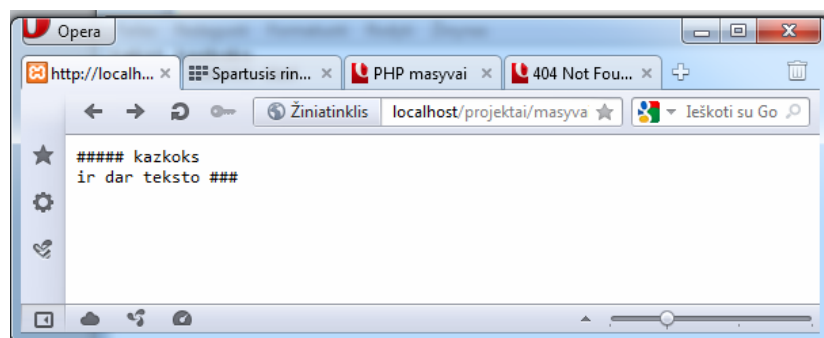
Funkcija *explode()* suskaldo eilutę pagal nurodytą skirtuką, t. y. yra " " (tarpas) ir grąžina kiekvieną jos dalį masyve.

*implode()*, priešingai nei *explode()* – sujungia masyvo elementus į eilutę.

*preg\_split()* atlieka tą pačią funkciją kaip ir *explode()*, tik skirtukui naudojama paprastoji išraiška (*regular expression*), kuri suskaldo žodį po raidę.

Sakykime, tekstinio failo *duom.txt* turinys:

*tekst kažkoks*  
*ir dar teksto dad*



4.4.1 pav. Vaizdas naršyklėje

Naudodamiesi pavyzdžiu, sukurkite programas:

1. Tekstiniame faile surašyti: studento vardas, ūgis ir svoris, sukurkite rezultatų failą ir į jį įrašykite studento vardą ir apskaičiuotą kūno masės indeksą.
2. Duomenų faile surašyti skaičiai. Reikia duomenis iš failo surašyti į masyvą ir išvesti į ekraną. Suskaičiuokite masyvo elementų sumą ir išveskite į ekraną.
3. Duomenų faile surašyti skaičiai. Reikia duomenis iš failo surašyti į masyvą. Masyvo elementų reikšmės įrašomos į rezultatų failą.

## 4.5. Praktinė užduotis. Loterija

**Darbo tikslas:** pagilinti darbo su *PHP* kalba įgūdžius.

**Užduotis.** Sakykime, vyksta loterija, kurios metu išridenami 6 skaičiai, kiekvienas iš jų yra vienaženklis – nuo 0 iki 9. Skaičiai gali kartotis ir tarpusavyje nesusiję. Lošėjas viename biliete bando spėti visą šešetuką. Jeigu atspėja visus šešis skaičius (pagal jų pozicijas), tada laimi didįjį prizą. Kad jis laimėtų, minimaliai užtenka atspėti 2 skaičius. Sukurkite *PHP* skriptą, kuriam iš naršyklės adreso eilutės pateikiami parametrai – šeši skaičiai, kuriuos spėja lošėjas. Skriptas sugeneruoja lošimą – šešis skaičius atsitiktine tvarka – ir parodo ekrane visus skaičius, kiek iš jų atspėta bei koks yra laimėjimas.

Padarykite prielaidą, kad adresas bus formuojamas programai pateikiant parametrus *x1*, *x2*, *x3*, *x4*, *x5* ir *x6*, kuriuos ir reikia nuskaityti. Įrašykite šias skaičių reikšmes į masyvą. Tai galima padaryti atliekant veiksmus su vienintele kodo eilute:

```
$spejimas = array($_GET['x1'], $_GET['x2'], $_GET['x3'], $_GET['x4'],  
$_GET['x5'], $_GET['x6']);
```

Tačiau nepamirškite patikrinti, ar į adreso eilutę įvesti reikiami skaičiai ir ar tikrai nėra klaidų. Dėl to kiekvieną kintamąjį reikia patikslinti, ar tai skaičius nuo 0 iki 9. Panaudokite funkciją *is\_int()*, kuri nustato, ar kintamasis yra sveikasis skaičius:

```
$spejimas = array();  
if (is_int($_GET['x1']) && $_GET['x1'] <= 9) {  
    $spejimas[] = $_GET['x1'];  
}  
if (is_int($_GET['x2']) && $_GET['x2'] <= 9) {  
    $spejimas[] = $_GET['x2'];  
}  
if (is_int($_GET['x3']) && $_GET['x3'] <= 9) {  
    $spejimas[] = $_GET['x3'];  
}  
if (is_int($_GET['x4']) && $_GET['x4'] <= 9) {  
    $spejimas[] = $_GET['x4'];  
}  
if (is_int($_GET['x5']) && $_GET['x5'] <= 9) {  
    $spejimas[] = $_GET['x5'];  
}  
if (is_int($_GET['x6']) && $_GET['x6'] <= 9) {  
    $spejimas[] = $_GET['x6'];  
}
```

*Lošimo generavimas.* Reikia sukurti tuščią masyvą, o tada šešis kartus įvykdyti ciklą, kuriame bus generuojamas atsitiktinis skaičius. Kiekvieną iš jų įrašykite į masyvą bei patikrinkite su spėjimo masyvo elementu – jeigu sutampa, tai padidinkite teisingų skaičių kintamąjį vienetu:

```
$atspeta = 0;  
$losimas = array();  
for ($i=0; $i <= 5; $i++) {
```



```

$skaičius = rand(0,9); // rand() – atsitiktinis skaičius tarp 0 ir 9
$losimas[] = $skaičius;
if ($skaičius == $spejimas[$i]) { $atspeta++; }
}

```

Prisiminkite, kad *PHP* kalboje masyvų numeravimas prasideda nuo 0, o ne nuo 1, tad ir ciklas vykdomas nuo 0 iki 5 pozicijų.

*Informacijos išvedimas į ekraną.* Išveskite spėjamus skaičius, tada lošimo skaičius ir galiausiai rezultatus – kiek skaičių atspėta ir ar jūs laimėjote:

```

echo "Spėjami skaičiai: " . implode(" ", $spejimas) . "<br />";
echo "Lošimo skaičiai: " . implode(" ", $losimas) . "<br />";
echo "Atspėta skaičių: " . $atspeta . "<br />";
if ($atspeta >= 2) {
echo "Jūs laimėjote";
} else {
echo "Jūs nelaimėjote";
}

```

*Pilnas skripto PHP kodas.* Programa realizuota. Bet norisi ją šiek tiek pajvairinti. O kaip atrodytų programa, jeigu lošimų būtų ne 1, o 10? Arba 100? Tam tikslui parašykite dar vieną ciklą. Pilnas programos kodas atrodo taip:

```

$spejimas = array();
if (is_int($_GET['x1']) && $_GET['x1'] <= 9) {
$spejimas[] = $_GET['x1'];
}
if (is_int($_GET['x2']) && $_GET['x2'] <= 9) {
$spejimas[] = $_GET['x2'];
}
if (is_int($_GET['x3']) && $_GET['x3'] <= 9) {
$spejimas[] = $_GET['x3'];
}
if (is_int($_GET['x4']) && $_GET['x4'] <= 9) {
$spejimas[] = $_GET['x4'];
}
if (is_int($_GET['x5']) && $_GET['x5'] <= 9) {
$spejimas[] = $_GET['x5'];
}
if (is_int($_GET['x6']) && $_GET['x6'] <= 9) {
$spejimas[] = $_GET['x6'];
}
$losimu_skaicius = 100;
$laimingi_bilietai = 0;
for ($losimai=1; $losimai <= $losimu_skaicius; $losimai++) {
$atspeta = 0;
$losimas = array();
for ($i=0; $i <= 5; $i++) {

```

```

$skaičius = rand(0,9);
// funkcija rand() grąžina atsitiktinį skaičių,
// šiuo atveju intervale 0-9
$losimas[] = $skaičius;
if ($skaičius == $spejimas[$i]) { $atspeta++; }
echo "Spėjami skaičiai: " . implode(" ", $spejimas) . "<br />";
}
echo "Lošimo skaičiai: " . implode(" ", $losimas) . "<br />";
echo "Atspėta skaičių: " . $atspeta . "<br />";
if ($atspeta >= 2) {
echo "Jūs laimėjote";
$laimingi_bilietai++;
} else {
echo "Jūs nelaimėjote";
}
}
echo "<hr />"; echo "Laimingų bilietų: " . $laimingi_bilietai;

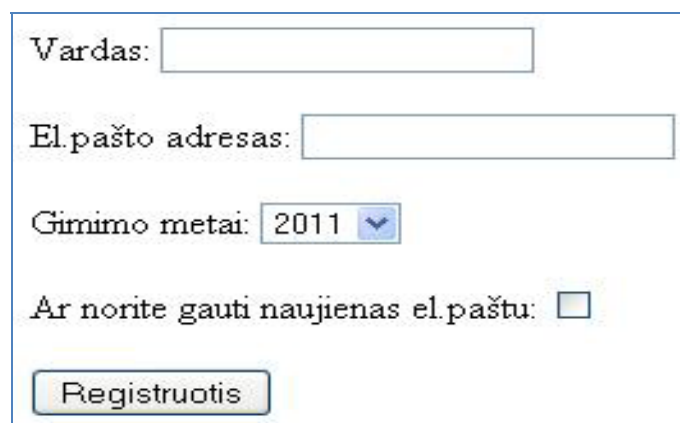
```

Ekrane matyti kiekvieno lošimo statistika pačioje pabaigoje – kiek bilietų laimėta iš visų lošimų, jeigu skaičiuotumėte, kad turėjote tik po vieną kiekvieno lošimo bilietą su tais pačiais skaičiais.

## 4.6. Praktinė užduotis. Formos ir jų apdorojimas

**Darbo tikslas:** pagilinti gebėjimus kurti ir apdoroti formas.

**Užduotis.** Sukurti formą ir apdoroti duomenis. Naudodamiesi *HTML* kalba, sukurkite tokią formą:



Vardas:

El. pašto adresas:

Gimimo metai:

Ar norite gauti naujienas el. paštu: ☐

4.6.1 pav. Forma

Formos kodas *HTML* kalba (4.6.1 pav.):

```

<form action="rezultatas.php" method="post">
Vardas: <input type="text" name="vardas" />
<br /><br />

```

```

El.pašto adresai: <input type="text" name="pastas" />
<br /><br />
Gimimo metai:
<select name="metai">
<option value="2011">2011</option>
<option value="2010">2010</option>
<option value="2009">2009</option>
<option value="2008">2008</option>
<option value="2007">2007</option>
<option value="2006">2006</option>
</select>
<br /><br />
Ar norite gauti naujienas el. paštu:
<input type="checkbox" name="naujienos" />
<br /><br />
<input type="submit" name="submit" value=" Registruotis " />
</form>

```

Turėdami *PHP* skriptą užpildytą formą, apdorokite jos duomenis, t. y. išveskite į ekraną ir patikrinkite teisingumą. 4.6.1 pav. pavaizduotos formos užpildymo skriptas *rezultatas.php*:

```

echo "Jūsų įvestas vardas: " . $_POST['vardas'] . "<br />";
echo "Jūsų įvestas el. pašto adresai: " . $_POST['pastas'] . "<br />";
echo "Jūsų gimimo metai: " . $_POST['metai'] . "<br />";
echo "Ar norite užsisakyti naujienas: ";
if ($_POST['naujienos'] && $_POST['naujienos'] == 'on') {
echo 'taip';
} else {
echo 'ne';
}

```

Užpildykite formą (4.6.2 pav.):



The screenshot shows a web form with the following elements:

- A text input field labeled "Vardas:" containing the text "Remigijus".
- A text input field labeled "El. pašto adresai:" containing the text "remigijus@remka.lt".
- A dropdown menu labeled "Gimimo metai:" with "2010" selected.
- A checkbox labeled "Ar norite gauti naujienas el. paštu:" which is checked.
- A "Registruotis" button at the bottom.

4.6.2 pav. Formos pildymas

Pagal anksčiau sukurtą *PHP* kodą turite gauti rezultatą (4.6.3 pav.).

Jūsų įvestas vardas: Remigijus  
Jūsų įvestas el. pašto adresas: remigijus@remka.lt  
Jūsų gimimo metai: 2010  
Ar norite užsisakyti naujienas: taip

4.6.3 pav. Formos patikrinimo rezultatas

Analogiškai duotajam pavyzdžiui sukurkite ir apdorokite formą (4.6.4 pav.).

Laukai, kuriuos turite užpildyti, kad mes galėtume atlikti Jūsų užsakymą, pažymėti žvaigždute ("\*\*")

Vardas:  \*

Pavardė:  \*

Telefonas:  \*

Fax:  El. pašto adresas:

Pasirinkite prekę: 

Valiklis "Stebuklas"

Skalbiklis "OMO"

Šampūnas "Pantene Pro-V"

Mūsų adresas: " Prekės paštu" Gatve 7, Miestas, pašto indeksas

telefonas: (8-27) 123456 fax: (8-27) 123456

4.6.4 pav. Formos pavyzdys

## 4.7. Praktinė užduotis. PHP funkcijų rašymas

**Darbo tikslas:** įgyti gebėjimus rašyti funkcijas *PHP* kalba.

**Užduotis.** Parašyti funkciją, kuri suskaičiuotų mokinio pažymių aritmetinį vidurkį ir išvestų tik tuos pažymius, kurie aukštesni už 8:

```
functionPirmunai($pazymiu_masyvas) {  
    $suma = 0;  
    for ($i=0; $i < count($pazymiu_masyvas); $i++) {  
        if ($pazymiu_masyvas[$i] > 8) {  
            echo $pazymiu_masyvas[$i] . " ";  
        }  
        $suma = $suma + $pazymiu_masyvas[$i];  
    }  
    // tikriname, ar masyvas ne tuščias, nes iš 0 dalinti negalima  
    if (count($pazymiu_masyvas) > 0) {  
        $vidurkis = $suma / count($pazymiu_masyvas);  
    } else {  
        $vidurkis = 0;  
    }  
}
```

```
return $vidurkis;
}
```

Funkcijos iškvietimas ir jos panaudojimas, kai kiekvienam parametrai priskiriamos reikšmės pagal nutylėjimą:

```
function SkaiciausKvadratas ($skaicius = 2) {
    return $skaicius * $skaicius;
}
$kvadratas = SkaiciausKvadratas(5); // funkcija grąžina 25
$kvadratas = SkaiciausKvadratas();
// funkcija grąžina 4, nes skaičiuoja kvadratą iš 2
```

## 4.8. Praktinė užduotis. Vartotojų sesijos

**Darbo tikslas:** sukurti vartotojų prisijungimų ir registracijos sistemą.

**Užduotis.** Sukurti prisijungimų ir registracijos sistemą, kad galėtumėte autorizuoti vartotojus.

Iš esmės yra du būdai, kaip saugoti vartotojų prisijungimo sesijas: tai *saušainiukai* (*cookies*) bei paprastos *PHP* sesijos. Kadangi labiau paplitęs antrasis būdas, būtent jį ir naudosime.

Sesijos sukūrimas ir panaudojimas

Darbas kuriant vartotojų sesijas susideda iš keturių etapų:

- sesijos sukūrimas;
- sesijos kintamųjų užpildymas;
- sesijos kintamųjų panaudojimas;
- sesijos užbaigimas.

Kad išnaudotumėte su sesijomis susijusias galimybes, pačioje skripto pildymo pradžioje turite pradėti sesiją:

```
session_start();
```

Sesija iškviesta, ir galima ją naudoti. Kiekvienai vartotojo sesijai serveryje sukuriamas kintamųjų masyvas, kuris saugomas tik tos sesijos gyvavimo metu. Dėl to, lankytoji pereinant nuo puslapio prie puslapio, galima atsekti, koks vartotojas, kokios jo teisės, kokius puslapius jis jau aplankęs ir t. t.

Visa tai atliekama, įrašant duomenis į specialų sesijos masyvą `$_SESSION`, masyvo elementams priskiriant reikšmes:

```
] $_SESSION['vartotojo_role'] = "1";
// pažymi, kad vartotojo rolė yra pirma (pvz., administratorius)
$_SESSION['puslapiai']++;
```

*// pažymi, kad lankytojas aplankė dar vieną puslapį (+1)*

Masyvas `$_SESSION` su išsaugotomis reikšmėmis yra prieinamas visiems *PHP* skriptams, kol tas vartotojas neatsijungia nuo sistemos ir neužbaigia sesijos.

Kai vartotojas atsijungia nuo sistemos, reikia jo sesiją būtinai panaikinti. Kitaip gali kilti problemų: prie to paties kompiuterio atsisėdęs kitas žmogus prieis prie to vartotojo sesijos be slaptažodžio. Panaikinimas atliekamas su funkcija *session\_destroy()* be parametrų.

Sesijos panaudojimo pavyzdys – autorizacijos mechanizmas. Pvz. Sakykime, jūs turite savo nedidelę turinio valdymo sistemą ir jums reikia, kad prie jos prisijungtų tik administratorius su tam tikru vartotoju ir slaptažodžiu. Prisijungimo forma atrodytų taip:

A login form with two input fields labeled 'Vardas' (Name) and 'Slaptažodis' (Password), and a 'Prisijungti' (Login) button.

4.8.1 pav. *Prisijungimo forma*

Iš prisijungimo formos (4.8.1 pav.) į *PHP* skriptą perduodami trys formos parametrai – kintamieji: *login*, *password* ir *submit*. Naudojant formos kintamųjų perdavimo metodą *POST*, sutrumpintas *PHP* skripto tekstas atrodys taip:

```
session_start();
$error = ""; // klaidos tekstas - bendru atveju šis kintamasis tuščias
if($_POST['submit'] == "Prisijungti") {
    // jeigu paspaustas mygtukas "Prisijungti"
    if($_POST['login'] == "admin" && $_POST['password'] == "admin123") {
        $_SESSION['username'] = "admin";
    } else {
        $error = "Neteisingi prisijungimo duomenys";
    }
}
if($_SESSION['username'] == "admin") {
    echo "Sveiki, jums prieinamos administratoriaus funkcijos...";
    // parodomas administratoriaus meniu
} else {
    echo "<form> ... </form>"; // užkraunama visa prisijungimo forma
    if($error != "") { echo $error; }
}
```

## 4.9. Praktinė užduotis. JavaScript pradmenys, JavaScript kintamieji

**Darbo tikslas:** įgyti *JavaScript* kalbos pradmenis, susipažinti su pagrindinėmis struktūromis, su *JavaScript* kalbos kintamaisiais, juos aprašyti ir išmokti naudoti.

*JavaScript* kalbos elementai įterpiami į *HTML* kodą:

```
<scriptlanguage="JavaScript">
```

```
<!--
```

```
...
```

```
//skripto programa
```

```
...
```

```
//-->
```

```
</script>
```

**Užduotis.** Išnagrinėkite pavyzdžius ir išbandykite praktiškai:

**Pvz. 1.** Pranešimams spausdinti naudojamos komandos *alert* ir *document.write*:

```
<scriptlanguage="JavaScript">
```

```
    alert('Informacijos išvedimo dialogo lange pavyzdys')
```

```
</script>
```

```
<scriptlanguage="JavaScript">
```

```
    document.write('Informacijos išvedimo naršyklės lange pavyzdys')
```

```
</script>
```

Informacijai įrašyti naudojami operatoriai *alert* ir *prompt*.

**Pvz. 2.** Informacijos išvedimas:

```
<scriptlanguage="JavaScript">
```

```
    if (confirm('Ar norite peržiūrėti kitą puslapį ?'))
```

```
        document.write('Peržiūra')
```

```
    else
```

```
        document.write('Tas pats puslapis')
```

```
</script>
```

**Pvz. 3.** Informacijos išvedimas (4.9.1 pav.):

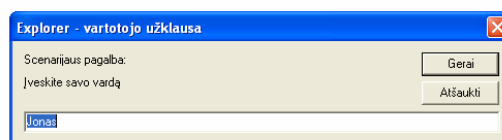
```
<scriptlanguage='JavaScript'>
```

```
    var s
```

```
    s=prompt('Įveskite savo vardą', 'Jonas')
```

```
    document.write(s)
```

```
</script>
```

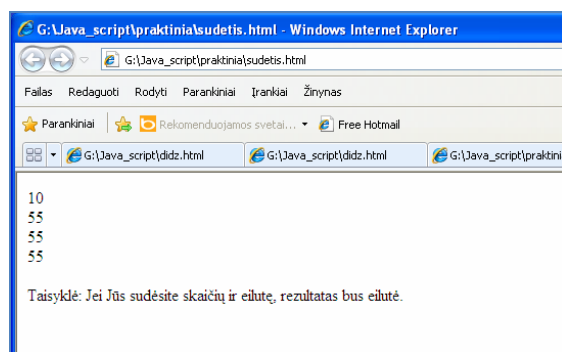


4.9.1 pav. Informacijos išvedimas

**Pvz. 4.** Matematiniai operatoriai (4.9.2 pav.):

```
<html>
<body>

<scripttype="text/javascript">
var x;
x=5+5;
document.write(x);
document.write("<br />");
x="5"+"5";
document.write(x);
document.write("<br />");
x=5+"5";
document.write(x);
document.write("<br />");
x="5"+5;
document.write(x);
document.write("<br />");
</script>
<p>Taisyklė: Jei Jūs sudėsite skaičių ir eilutę, rezultatas bus eilutė.</p>
</body>
</html>
```



4.9.2 pav. Matematiniai operatoriai

Naudodamiesi pateiktais pavyzdžiais, sukurkite skriptus:

1. Parašykite skriptus, kuriuose būtų panaudojamos daugybos, dalybos, atimties operacijos.
2. Išveskite į ekraną savo vardą ir pavardę (naudokite operatorių *Prompt*).

## 4.10. Praktinė užduotis. JavaScript operacijos

**Darbo tikslas:** susipažinti su *JavaScript* kalbos operacijomis, sukurti skriptus, naudojant paprasčiausias operacijas.

**Užduotis.** Išnagrinėkite pavyzdžius ir išbandykite praktiškai:

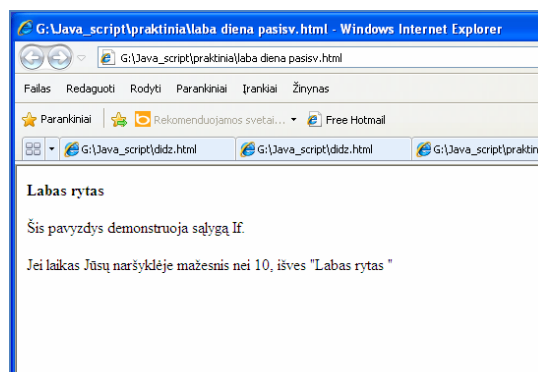
**Pvz. 1.** Sąlygos sakiny (4.10.1 pav.):



```

<html>
<body>
<scripttype="text/javascript">
var d = new Date();
var time = d.getHours();
if (time < 10)
{
document.write("<b>Labas rytas</b>");
}
</script>
<p>Šis pavyzdys demonstruoja sąlygą If.</p>
<p>Jei laikas Jūsų naršyklėje mažesnis nei 10, išves "Labas rytas" </p>
</body>
</html>

```



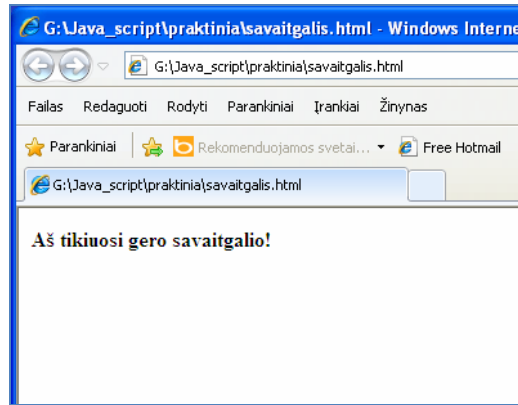
4.10.1 pav. Sąlygos sakiny

**Pvz. 2.** Išrinkimo sakiny (4.10.2 pav.):

```

<html>
<body>
<scripttype="text/javascript">
vard=newDate();
var theDay=d.getDay();
switch (theDay)
{
case 5:
document.write("<b>pagaliau penktadienis</b>");
break;
case 6:
document.write("<b>puiku šeštadienis</b>");
break;
case 0:
document.write("<b>mieguistas sekmadienis</b>");
break;
default:
document.write("<b>Aš tikiuosi gero savaitgalio!</b>");
}
</script>
</body>
</html>

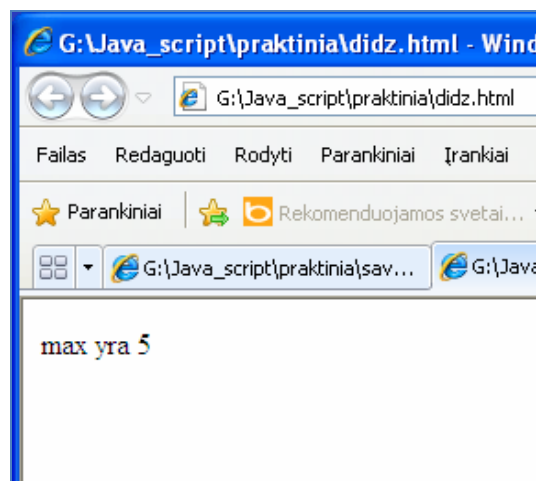
```



4.10.2 pav. Išrinkimo sakiny

**Pvz. 3.** Didesnio skaičiaus iš dviejų duotų radimas (4.10.3):

```
<html>
<body>
<scripttype="text/javascript">
    var a = 5;
    var b = 2;
    if (a > b) {document.write("max yra " + a);}
    else {document.write("max yra " + b);}
</script>
</body>
</html>
```



4.10.3 pav. Maksimalaus skaičiaus paieška

Naudodamiesi pavyzdžiais, sukurkite skriptus:

1. Pakeiskite 1 pavyzdį: jei laikas jūsų naršyklėje didesnis nei 10 (t. y. priešingu atveju), būtų parašyta „Laba diena“.
2. Papildykite 3 pavyzdį, kad iš trijų duotų skaičių būtų surastas ir išvestas didžiausias.
3. Parašykite skriptus (naudokite ciklo operatorius *for* ir *while*): kintamasis *i* ciklo metu kinta nuo 0 iki 5. Ciklo metu išvedamos visos kintamojo reikšmės (4.10.4 pav.):

```
Skaicius yra 0  
Skaicius yra 1  
Skaicius yra 2  
Skaicius yra 3  
Skaicius yra 4  
Skaicius yra 5
```

4.10.4 pav. Kintamasis kinta nuo 0 iki 5

## 4.11. Praktinė užduotis. JavaScript funkcijos

**Darbo tikslas:** įgyti darbo su JavaScript funkcijomis įgūdžių, parašyti funkcijas.

Funkcijos sintaksė:

```
functionFunkcijos_pavadinimas(parametras1, parametras2...){  
  Funkciją sudarantys operatoriai...  
  return reikšmė;  
}
```

*parametras1, parametras2* – funkcijai perduodami argumentai arba parametrai. *Return* – funkcijos grąžinama reikšmė. Funkcija parametų ir grąžinamų reikšmių gali ir neturėti.

**Užduotis.** Išnagrinėkite ir praktiškai išbandykite pavyzdžius:

**Pvz. 1.** Funkcija be argumentų:

```
<html>  
<head>  
<scripttype="text/javascript">  
functiondisplaymessage()  
{  
  alert("Sveikas pasauli!");  
}  
</script>  
</head>  
  
<body>  
<form>  
<inputtype="button" value="Click me!" onclick="displaymessage()" />  
</form>  
</body>  
</html>
```

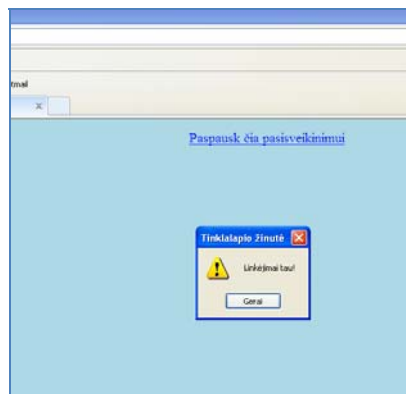
**Pvz. 2.** Funkcijos kvietimas su nuoroda (4.11.1 pav.):

```
<html>  
<head><title>Functions</title>  
<scripttype="text/javascript">
```

```

function greetings(){ // Function defined within <head> tags
document.bgColor="lightblue";
alert("Linkėjimai tau!");
}
</script>
</head>
<body bgcolor="silver">
<div align="center">
<a href="JavaScript:greetings()"><big>Paspauk čia pasisveikinimui</big>
</a><br />
</div>
</body>
</html>

```



4.11.1 pav. Vaizdas naršyklėje

**Pvz. 3.** Funkcijos kvietimas iš įvykio:

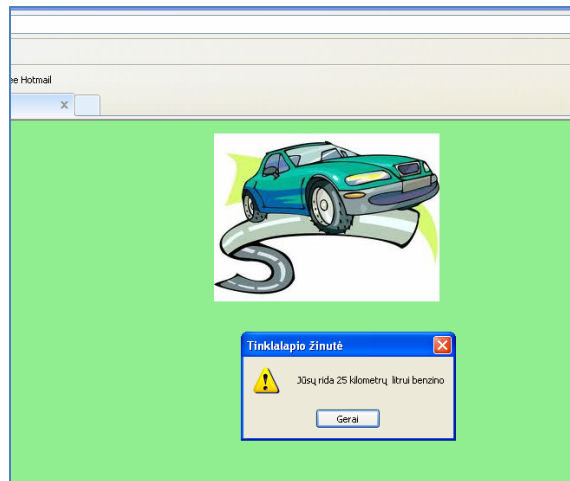
```

<html>
<head><title>FunctionsandEvents</title>
<script type="text/javascript">
function greetings(you){ // Function definition
document.bgColor="lavender";
alert("Sveikininimai ir linkėjimai! " + you);
}
</script>
</head>
<body>
<div align="center">
<form>
<input type="button"
value="Welcomebutton"
onClick="greetings('Danai');">
</form>
</div>
</body>
</html>

```

**Pvz. 4.** Funkcija su argumentais (4.11.2 pav.):

```
<html>
<head><title>ReturnValue</title>
<scripttype="text/javascript">
functionmileage(miles, gas){
returnmiles/gas; // grąžina dalybos rezultatą
}
</script>
</head>
<bodybgcolor="lightgreen">
<fontface="arial" size="+1">
<divalign="center">
<imgsrc="car-wave.gif">
<scripttype="text/javascript">
vardistance=eval(prompt("Kiek kilometrų jūs nuvažiavote?", ""));
varamount=eval(prompt("Kiek litrų benzino jūs sunaudojote?", ""));
var rate = mileage(distance, amount);
// Returnvalueassigned to rate
alert("Jūsų rida"+ rate +" kilometrų 1 litrui benzino");
</script>
</div>
</font>
</body>
</html>
```



4.11.2 pav. *Vaizdas naršyklėje*

Naudodamiesi pavyzdžiais, sukurkite skriptus:

1. Sukurkite funkciją, kuri sujungia 3 kintamuosius ir grąžina reikšmę – tų kintamųjų sąjungą. Rezultatą išveskite į ekraną.
2. Sukurkite funkciją, kuri apskaičiuotų kūno masės indeksą (argumentai: kūno masės ir ūgio reikšmės). Rezultatą išveskite į ekraną.

## 4.12. Praktinė užduotis. JavaScript masyvai

**Darbo tikslas:** įgyti darbo su JavaScript masyvais įgūdžių.

**Užduotis.** Sukurti objektų masyvą *myCars*.

Masyvas – tai specialus kintamasis, kuris tuo pačiu metu gali turėti keletą reikšmių. Sukurti masyvą galima trimis būdais:

1 būdas:

```
var myCars = new Array(); // sukuriamas objektų masyvas
myCars[0] = "Saab";      //
myCars[1] = "Volvo";
myCars[2] = "BMW";
```

2 būdas:

```
var myCars = new Array("Saab", "Volvo", "BMW");
```

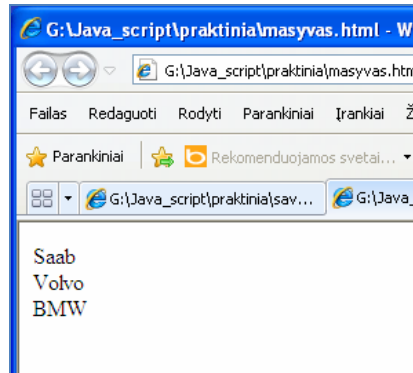
3 būdas:

```
var myCars = ["Saab", "Volvo", "BMW"];
```

Išnagrinėkite ir praktiškai išbandykite pavyzdžius:

**Pvz. 1.** Masyvo elementų išvedimas į ekraną (4.12.1 pav.):

```
<html>
<body>
<script type="text/javascript">
var i;
var mycars = new Array();
mycars[0] = "Saab";
mycars[1] = "Volvo";
mycars[2] = "BMW";
for (i=0; i<mycars.length; i++)
{
document.write(mycars[i] + "<br />");
}
</script>
</body>
</html>
```



4.12.1 pav. Masyvo elementų išvedimas

Masyvų funkcijos:

*concat()* – sujungia kelis masyvus;  
*join()* – sujungia masyvo elementus į eilutę;  
*pop()* – išmeta paskutinį masyvo elementą;  
*push()* – prideda naujų elementų į masyvą;  
*reverse()* – sukeičia masyvo elementus vietomis  
*shift()* – išmeta pirmą masyvo elementą;  
*slice()* – pažymi masyvo elementus;  
*sort()* – rūšiuoja masyvo elementus didėjimo/mažėjimo tvarka;  
*splice()* – prideda elementą į antrą poziciją;  
*toString()* – paverčia masyvą į eilutę;  
*unshift()* – prideda naujų elementų į masyvo pradžią.

**Pvz. 2.** Funkcija *concat()*:

```
<html>
<body>
<scripttype="text/javascript">
varparents = ["Jani", "Tove"];
varbrothers = ["Stale", "Kai Jim", "Borge"];
varchildren = ["Cecilie", "Lone"];
varfamily = parents.concat(brothers, children);
document.write(family);
</script>
</body>
</html>
```

**Pvz. 3.** Funkcija *reverse()*:

```
<html>
<body>
<scripttype="text/javascript">

varfruits = ["Banana", "Orange", "Apple", "Mango"];
document.write(fruits.reverse());
```

```
</script>
</body>
</html>
```

**Pvz. 4.** Funkcija *join()*:

```
<html>
<body>
<scripttype="text/javascript">
varfruits = ["Banana", "Orange", "Apple", "Mango"];
document.write(fruits.join() + "<br />");
document.write(fruits.join("+") + "<br />");
document.write(fruits.join(" and "));
</script>
</body>
</html>
```

**Pvz. 5.** Funkcija *pop()*:

```
<html>
<body>
<scripttype="text/javascript">
varfruits = ["Banana", "Orange", "Apple", "Mango"];
document.write(fruits.pop() + "<br />");
document.write(fruits + "<br />");
document.write(fruits.pop() + "<br />");
document.write(fruits);
</script>
</body>
</html>
```

**Pvz. 6.** Funkcija *push()*:

```
<html>
<body>
<scripttype="text/javascript">
varfruits = ["Banana", "Orange", "Apple", "Mango"];
document.write(fruits.push("Kiwi") + "<br />");
document.write(fruits.push("Lemon", "Pineapple") + "<br />");
document.write(fruits);
</script>
</body>
</html>
```

Naudodamiesi pavyzdžiais, atlikite šias užduotis:

1. Duotas sveikų skaičių masyvas iš dešimties elementų A(10). Rasti didžiausią skaičių ir jį sumažinti perpus. Spausdinti masyvus prieš ir po pakeitimo.



2. Duoti du masyvai  $A(n)$  ir  $B(m)$ . Masyvų didžiausius elementus sukeisti vietomis, o tų elementų aritmetinį vidurkį užrašyti abiejų masyvų gale. Spausdinti masyvus prieš ir po veiksmų.
3. Duotas masyvas  $A(10)$ . Gauti masyvo teigiamų elementų sandaugą, apskaičiuoti neigiamų elementų sumą ir nulinių – kiekį.
4. Duotas masyvas  $A(n)$ . Išrikiuoti masyvo elementus didėjimo tvarka ir mažiausią iš jų – pašalinti.

#### 4.13. Praktinė užduotis. JavaScript datos ir laiko objektai, įvykiai

**Darbo tikslas:** įgyti darbo su *JavaScript* datos ir laiko objektais įgūdžių.

Objektas *Date* naudojamas įvairiems veiksmams su laiku ir datomis atlikti. Objektas turi daug metodų datai nustatyti, reikšmėms gauti.

*Date* objektai yra kuriami panaudojant *Date()* konstruktorių.

```
newDate() // currentdateandtime  
newDate(milliseconds) //millisecondssince 1970/01/01  
newDate(dateString)  
newDate(year, month, day, hours, minutes, seconds, milliseconds)
```

Datos iniciavimo pavyzdžiai:

```
var today = newDate()  
var d1 = newDate("October 13, 1975 11:13:00")  
var d2 = new Date(79,5,24)  
var d3 = new Date(79,5,24,11,33,0)
```

**Užduotis.** Išnagrinėkite ir praktiškai išbandykite pavyzdžius:

**Pvz. 1.** Sukuriamas *Date* objektas su konkrečia data (2010 vasario 14):

```
var myDate=newDate();  
myDate.setFullYear(2010,02,14);
```

**Pvz. 2.** Nustatomas *Date* objektas 5 dienas į priekį:

```
var myDate=newDate();  
myDate.setDate(myDate.getDate()+5);
```

**Pvz. 3.** Šiandienos datos palyginimas su 2010 vasario 14 diena:

```
var myDate=newDate();  
myDate.setFullYear(2010,02,14);
```

```

vartoday = newDate();

if (myDate>today)
{
    alert("Todayisbefore 14th January 2010");
}
else
{
    alert("Todayisafter 14th January 2010");
}

```

**Pvz. 4.** Datos ir laiko nustatymo rodymas puslapyje:

```

<html>
<body>
<scripttype="text/javascript">
vard=newDate();
document.write(d);
</script>
</body>
</html>

```

**Pvz. 5.** Laikrodžio nustatymo rodymas puslapyje (4.13.1 pav.):

```

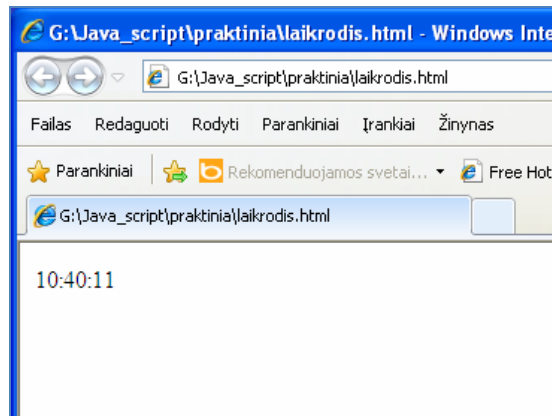
<html>
<head>
<scripttype="text/javascript">
functionstartTime()
{
    vartoday=newDate();
    varh=today.getHours();
    varm=today.getMinutes();
    vars=today.getSeconds();
    // add a zero in front of numbers < 10
    m=checkTime(m);
    s=checkTime(s);
    document.getElementById('txt').innerHTML=h+":"+m+":"+s;
    t=setTimeout('startTime()',500);
}
functioncheckTime(i)
{
    if (i<10)
    {
        i="0" + i;
    }
    return i;
}

```

```

}
</script>
</head>
<bodyonload="startTime()">
<divid="txt"></div>
</body>
</html>

```



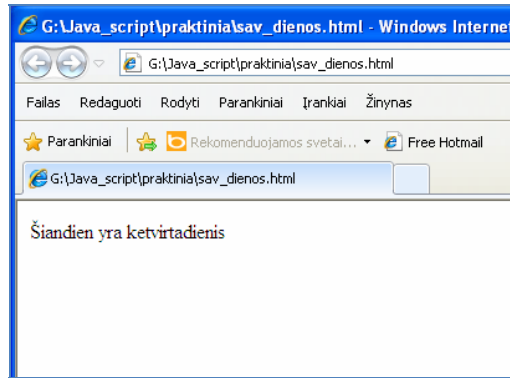
4.13.1 pav. *Laikrodžio nustatymas*

**Pvz. 5.** Savaitės dienos rodymas puslapyje (4.13.2 pav.):

```

<html>
<body>
<scripttype="text/javascript">
vard=newDate();
varweekday=new Array(7);
weekday[0]="sekmadienis";
weekday[1]="pirmadienis";
weekday[2]="antradienis";
weekday[3]="trečiadienis";
weekday[4]="ketvirtadienis";
weekday[5]="penktadienis";
weekday[6]="šeštadienis";
document.write("Šiandien yra " + weekday[d.getDay()]);
</script>
</body>
</html>

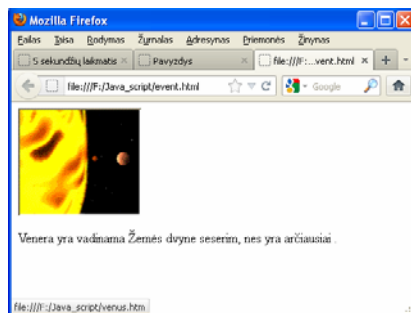
```



4.13.2 pav. Savaitės dienos nustatymas

**Pvz. 6.** Programos kodas, kai užėjus pele ant paveikslėlio pasikeičia užrašas. Priklausomai nuo to, kur bus pelės žymeklis, matysite kitą planetų apibūdinimą. Paspaudus pelės klavišą, bus iškviestas paveikslėlis su planetos atvaizdu (4.13.3 pav.):

```
<html>
<head>
<scripttype="text/javascript">
functionwriteText(txt)
{
document.getElementById("desc").innerHTML=txt;
}
</script>
</head>
<body>
<imgsrc="planets.gif" width="145" height="126" alt="Planets"
usemap="#planetmap" />
<mapname="planetmap">
<areashape="rect" coords="0,0,82,126"
onmouseover="writeText('Saulė ir Jupiteris yra didžiausi mūsų Saulės sistemos
objektai.')"
href="sun.htm" target="_blank" alt="Sun" />
<areashape="circle" coords="90,58,3"
onmouseover="writeText('Merkurijaus planetą yra sunku nagrinėti iš Žemės, nes ji yra
taip arti Saulės.')"
href="mercur.htm" target="_blank" alt="Mercury" />
<areashape="circle" coords="124,58,8"
onmouseover="writeText('Venera yra vadinama Žemės dvyne seserim, nes yra arčiausiai
.')"
href="venus.htm" target="_blank" alt="Venus" />
</map>
<p id="desc">Nuveskite pelės žymeklį virš saulės ir planetų ir pamatysite skirtingus
aprašymus.</p>
</body>
</html>
```



4.13.3 pav. *Vaizdas naršyklėje*

Naudodamiesi pavyzdžiais, atlikite užduotis:

1. Parašykite kodą, kuris, paspaudus mygtuką, įterptų į puslapį einamąją datą, laiką, savaitės dieną ir metus lietuviškai.
2. Sukurkite programą, kuri parodytų, kiek dienų liko iki jūsų gimtadienio.
3. Sukurkite programą, kuri reaguotų į pele atliekamus veiksmus.

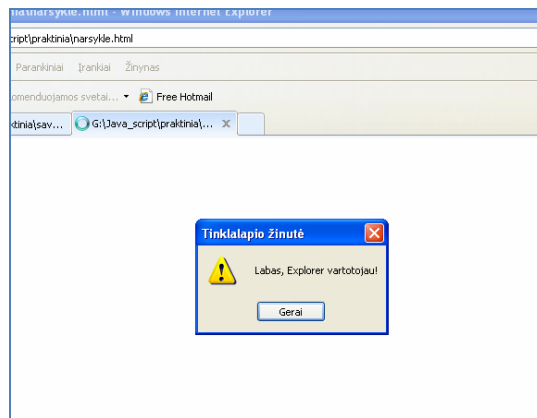
#### 4.14. Praktinė užduotis. JavaScript naršyklės, langai

**Darbo tikslas:** įgyti darbo su *JavaScript* naršyklėmis įgūdžių.

**Užduotis.** Išnagrinėkite ir praktiškai išbandykite pavyzdžius:

**Pvz. 1.** Kodas parodo naršyklės pavadinimą (4.14.1 pav.):

```
<head>
<script language="javascript">
<!--
var browserName=navigator.appName;
if (browserName=="Netscape")
{
alert("Labas Netscape vartotojau!");
}
else
{
if (browserName=="Microsoft Internet Explorer")
{
alert("Labas, Explorer vartotojau!");
}
else
{
alert("Kokia čia naršyklė??");
}
}
}
//-->
</script>
</head>
```



4.14.1 pav. Naršyklės pavadinimas

**Pvz. 2.** Kodas parodo naršyklės versijos numerį:

```
<head>
<scriptlanguage="javascript">
<!--
varbrowserVer=parseInt(navigator.appVersion);
if (browserVer >= 4)
{
alert("Yourbrowserisnewenoughformysite.");
}
else
{
alert("Ithinkyouneed a newbrowser!");
}
//-->
</script>
</head>
```

**Pvz. 3.** Paspaudus mygtuką *Atidaryti*, naujame lange atverčia tinklalapį *www.marko.lt*:

```
<html>
<head>
</head>
<body>
<scripttype="text/javascript">
function Atidaryti() {
window.open("http://www.marko.lt")
}
</script>
</head>
<body>
<inputtype=buttonvalue="Atidaryti langą" onclick="Atidaryti()" />
</body>
</html>
```

1. Parašykite kodą, kuris parodytų jūsų naršyklės pavadinimą, versiją, naudojamą kalbą ir kt. informaciją.
2. Parenkite tinklalapį, su kuriuo kartu atsivertų pagalbinis pasisveikinimo langas. Po 5 sekundžių pagalbinis langas turi būti uždaromas.

**Darbo tikslas:** įgyti darbo su *JavaScript* formomis įgūdžių.

**Pvz. 1.** Naudojant formų elementus, interneto svetainių lankytojų registravimo tinklalapis aprašomas (4.15.1 pav.):

The screenshot shows a Mozilla Firefox browser window. The address bar displays the local file path: `file:///F:/Java_script/praktinia/formos.html`. Below the address bar, there are several bookmarks: "Most Visited", "Getting Started", "Latest Headlines", and "galerija.swf". The main content area shows a registration form with the following elements:

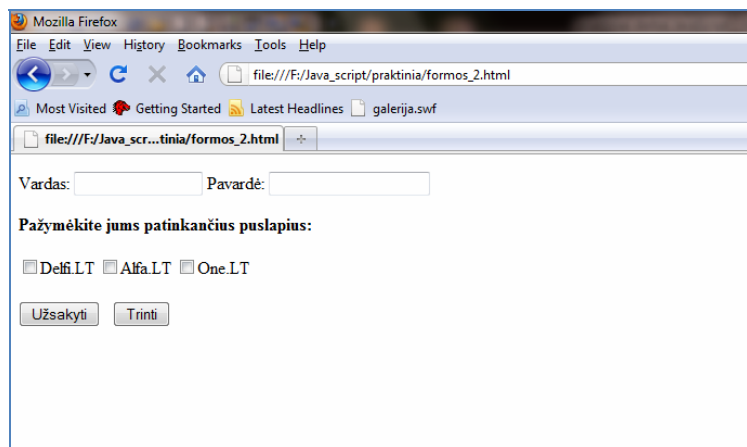
- Labels: "Vardas:" (Name) and "Pavardė:" (Surname).
- Input fields: Two empty text boxes for entering the name and surname.
- Label: "E-Paštas:" (Email).
- Input field: One empty text box for entering the email address.
- Buttons: Two buttons labeled "Registruotis" (Register) and "Trinti" (Delete).

Šioje formoje spragtelėjus *reset* tipo mygtuką *Trinti*, bus ištrinami visi teksto laukeliuose įrašyti duomenys, o *submit* tipo mygtukas *Registruotis* valdo šių duomenų perdavimą konteinerio *form* parametro *action* nurodomai programai. Kai šio parametro reikšmė yra elektroninio pašto adresas, formos elementų pavadinimai ir parinktos jų reikšmės siunčiamos parametro nurodytu adresu. Pvz:

```
<formname=Siuntimas e-paštu" action="mailto:Tavomail@Mail.LT"
method="POST" title="Test" entype="text/plain">
```

**Pvz. 2.** Parinkimo akutės ir žymimieji langeliai (4.15.2 pav.):

```
<html>
<head>
</head>
<body>
<formname="Prekyba">
<p>Vardas: <inputtype="text" name="vard" size="15">
Pavardė: <inputtype="text" name="pav" size="20"><br>
<p><b>Pažymėkite jums patinkančius puslapius:</b><br><br>
<inputtype="checkbox" name="dlf" value="ON">Delfi.LT
<inputtype="checkbox" name="alf" value="ON">Alfa.LT
<inputtype="checkbox" name="one" value="ON">One.LT
<br><br>
<inputtype="submit" value="Užsakyti">&nbsp;
<inputtype="reset" value="Trinti"></p>
</form>
</body>
</html>
```

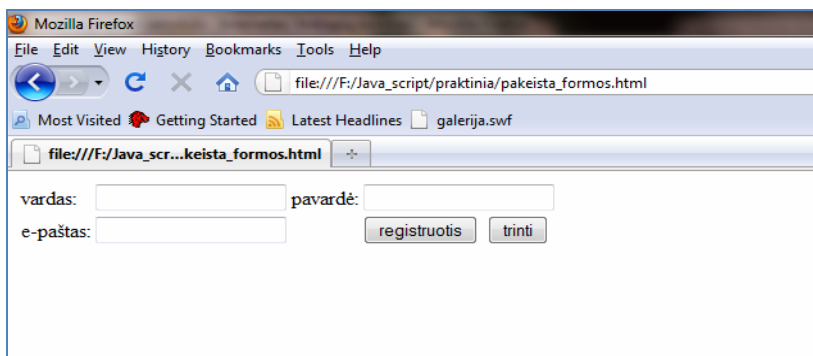


4.15.2 pav. Parinkimo akutės ir žymimieji langeliai

## Užduotys

1. Naudodamiesi lentelių ir eilučių aprašymo priemonėmis, pakeiskite 1 pvz. pateikto tinklalapio aprašymą, kad jo išvaizda būtų kaip 4.15.3 paveikslėlyje:





4.15.3 pav. *Formos pavyzdys*

2. Sukurkite formą, kurios išvaizda būtų kaip 4.15.4 pav.:

**Pažymėkite pageidaujamas OS:**

Windows XP

Pažymint keletą sąrašo elementų, reikia laikyti nuspaušta klavisa *Ctrl*

4.15.4 pav. *Formos pavyzdys*

3. Sukurkite registracijos formą, kurios išvaizda būtų kaip 4.15.5 pav.:

Vardas:

Pavardė:

---

Vartotojo vardas:

Slaptažodis:

---

Lytis: Vyras ☐ Moteris ☐

Amžius:

---

4.15.5 pav. *Formos pavyzdys*

## 4.16. Praktinė užduotis. JavaScript objektų masyvo kūrimas

**Darbo tikslas:** įgyti darbo su JavaScript objektų masyvais įgūdžių.

*JavaScript* objektai – tai vardo ir reikšmės sąrašai. Vardai yra simbolių sekos, o reikšmės gali būti skaičiai, simbolių sekos, loginės reikšmės bei objektai (taip pat masyvai ir funkcijos). *JavaScript* objektai įgyvendinami naudojantis sąrašais, kad reikšmės galėtų būti grąžinamos kuo greičiau. Jeigu reikšmė vardo-reikšmės sąraše yra funkcija, galima laikyti, kad tai metodas. Kai iškviečiamas objekto metodas, metodo kintamasis *this* laiko nuorodą į objektą. Per šį kintamąjį metodas turi priėjimą prie kitų jį iškvietusio objekto metodų.

Visi objektai yra kuriami naudojant konstruktorių – funkciją, kuri parengia objektą darbui. Konstruktoriai *JavaScript* kalboje suteikia tokias galimybes, kokias kitose kalbose suteikia klasės.

Sakykim, aprašytas objektas:

```
personObj.firstname="John";
personObj.lastname="Doe";
personObj.age=30;
personObj.eyecolor="blue";
document.write(personObj.firstname);
```

Aprašyto objekto rezultatas bus *John*.

Naują objektą galima sukurti dviem būdais: tiesioginiu būdu ir naudojant konstruktorių.

**Užduotis.** Išnagrinėkite ir praktiškai pritaikykite pateiktus pavyzdžius:

**Pvz. 1.** Tiesioginiu būdu sukurtas objektas:

```
<html>
<body>
<scripttype="text/javascript">
personObj={firstname:"John",lastname:"Doe",age:50,eyecolor:"blue"}
document.write(personObj.firstname + " is " + personObj.age + " yearsold.");
</script>
</body>
</html>
```

**Pvz. 2.** Objektas, sukurtas naudojant konstruktorių:

```
<html>
<body>
<scripttype="text/javascript">
function person(firstname,lastname,age,eyecolor)
{
this.firstname=firstname;
this.lastname=lastname;
this.age=age;
this.eyecolor=eyecolor;
}
myFather=new person("John","Doe",50,"blue");
document.write(myFather.firstname + " is " + myFather.age + " yearsold.");
</script>
</body>
</html>
```

Naudodamiesi pateiktais pavyzdžiais, atlikite užduotis:

1. Duotas objektų masyvas (studentų sąrašas), kuriame yra tokie laukai: pavardė, vardas, grupės šifras, visų egzaminų įvertinimai. Išvesti į ekraną pirmųjų (pažymių vidurkis  $> 8$ ) ir atsiliekančiųjų (pažymių vidurkis  $< 6$ ) sąrašus.
2. Duotas objektų masyvas (studentų sąrašas), kuriame yra tokie laukai: pavardė, vardas, kursas, visų egzaminų pažymiai. Išvesti į ekraną duomenis: pavardę, vardą, kursą, aštuntukų, devintukų, dešimtukų kiekius.
3. Parašyti programą, kai duotas n įrašų masyvas, kuriame yra informacija apie krepšininkus (įrašo struktūra: krepšininkas.vardas; krepšininkas.pavarde; krepšininkas.ugis). Programa iš įrašų masyvo turėtų pašalinti visus duomenis apie krepšininkus, kurių ūgis mažesnis nei 2 metrai. Išvesti pradinį ir pakeistą įrašų masyvą.

## LITERATŪROS ŠALTINIAI

1. Kaklauskas L. Tinklalapiai ir jų kūrimas. – Šiauliai, 2001. ISBN 9955418508.
2. PHP bendruomenė. [interaktyvus] [žiūrėta 2011-11-17]. Prieiga per internetą: <www.php.lt>
3. PHP komandos. [interaktyvus] [žiūrėta 2011-10-17]. Prieiga per internetą: <http://ik.su.lt/~programavimas/php\_komandos.php>
4. Programavimo pamokos lietuviškai. [interaktyvus] [žiūrėta 2011-09-26]. Prieiga per internetą: <http://manualai.lt/php.html>
5. Programavimo pamokos. [interaktyvus] [žiūrėta 2011-11-17]. Prieiga per internetą: <www.programva.com>
6. Robbins J. N. Tinklapių dizainas. (X)HTML kalbos, pakopinių stilių ir tinklalapių grafikos pradžiamokslis. – Kaunas: Smaltija, 2008. ISBN 9789955707479.
7. Širvinskas P. PHP pamokos pradedantiesiems. Elektroninė knyga [žiūrėta 2011-10-07]. Prieiga per internetą: <http://www.skaitykIT.lt>
8. Vidžiūnas A., Vitkutė D. Interneto paslaugos ir svetainių kūrimas. – Kaunas: Smaltija, 2009. ISBN 9789955707639.
9. w3schools konsorciumas. [interaktyvus][žiūrėta 2011-11-17]. Prieiga per internetą: <http://www.w3schools.com/>
10. Žiniatinklio technologijos. [interaktyvus] [žiūrėta 2011-10-02]. Prieiga per internetą: <http://www.internetotechnologijos.lt>



UDK 004.438(075.8)

Ri-121

ISBN 978-609-422-068-5

Leidiny s skirtas Marijampolės kolegijos Verslo ir technologijų fakulteto ir kitų aukštųjų mokyklų studentams, studijuojantiems objektinį programavimą. Šis mokymosi rinkinys gali būti naudingas visiems, norintiems išmokti objektinio programavimo pagrindų.

VILMA RIŠKEVIČIENĖ

---

# OBJEKTINIS PROGRAMAVIMAS

---

Mokymo(si) rinkinys

Redagavo Nijolė Bagdonienė

Maketavo Neringa Palubinskaitė

Tiražas 40 egz. Užsakymas 313

Leidykla „Piko valanda“, Žemaitės g. 59, LT-68303 Marijampolė

Tel./faks. (8 343) 54 216, el. p. leidyba@pikovalanda.lt, www.pikovalanda.lt