JavaScript

Kursuse konspekt

Autor Jüri PUHANG

T a l l i n n 2 0 0 7

Sisukord:

JavaSCRIPT

Sissejuhatus

JavaScript on firma Netscape Communications Corporation poolt väljatöötatud programmeerimise keel, mida kasutatakse dünaamilistel veebilehtedel.
JavaScript on:

* interpreteeriv keel, mida interpreteerib kliendi arvutis weebilehti lugev brauser. JavaScript-is programmi ei kodeerita (ei kompileerita) ümber täidetavaks programmiks, vaid sümbolkeeles programmi lähtetekst loetakse koos HTML-dokumendiga kliendi arvutisse ja brauser täidab (interpreteerib) selle programmi lauseid.
* Objekt-orienteeritud keel, mis tähendab et selle keele lausetega on kättesaadavad HTML-i lehe, brauseri, OP-süsteemi ja kliendi arvutis olevate teiste rakenduste objektid.

JavaScripti arengulugu

Java-projekti algatajaks peetakse James Gosling'it, kes 1991 aastal püüdis C++ keelt täiendada, et see sobiks WWW-ga ja et Internet'i brauser täidaks selles keeles tehtud programme.

Java keel on siiski kompileeriv keel ja selles kirjutatud programm kompileeritakse (kodeeritakse masinkoodi), analoogselt teises programmeerimise keeles kirjutatud programmidele. Java keeles kirjutatud ja kompileeritud programm salvestatakse serverarvutisse ja HTML-i lehelt viidatakse sellele programmile. HTML lehe allalaadimisel laetakse kliendi arvutisse ka kompileeritud programm (Java keeles kirjutatud) ja käivitatakse.

Sama keele baasil on siiski enamkasutatav JavaScript keel, mis keele ülesehituselt on lähedane Java keelega, kuid ei vaja eelnevat kompileerimist (on interpreteeriv keel). Selles keeles kirjutatud programm säilitatakse tekstilisel kujul serverarvutis ja loetakse kliendi arvutisse koos HTML lehega ning seda interpreteerib Internet'i brauser. Erinevad brauserid võivad interpreteerida programme veidi erinevalt ja seepärast on MicroSoft'il oma Java keele teisend JScript.

Samas on nii JavaScript kui ka JScript ECMA[*)]-262 publikatsioonis määratud ECMAScript-ile toetuvad keeled.

[*)] ECMA - Engineering College Magazines Associated.

1. peatükk: Keele ülesehitus

Script-keeles koostatud programm koosneb lausetest ja kommentaaridest.

Lausetes kasutatakse muutujaid, konstante, massiive, avaldisi, operaatoreid, funktsioone ja objekte. Lause võib alata **võtmesõnaga** ja lõpeb **semikooloniga** (;).

Muutujatele, konstantidele, massiividele, funktsioonidele ja objektidele antakse sümbolitest koosnevad nimed ehk identifikaatorid. Nimed algavad tähega või allakriipsutuse (_) märgiga. Nimedes võib kasutada tähti A kuni Z ja a kuni z ning numbreid 0 kuni 9 (ei tohi kasutada täpitähti ö ä ü õ). Nimedes ei tohi kasutada tühikuid. Kui soovitakse kasutada programmi sisust lähtuvalt mitmesõnalisi nimesid, peab sõnad eraldame allakriipsutusega (_) või iga sõna algama suure tähega.
Näiteks: See_on_pikk_nime või SeeOnPikkNimi.

Lausetes võib kasutada tühikuid ning tabulaatorit.

Keeles kasutatakse kolme liiki sulge ( ), [ ], ja { } ning kahte tüüpi tekstieraldajaid: jutumärke “ “ ning ülakomasid ‘ ‘

NB! Ülakoma võib segi minna rõhumarkide ehk apostroofidega (´`) mida ei tohi kasutada

Üherealine **kommentaar** või kommentaar lause lõpus algab `//` märkidega ja lõpeb rea lõpuga.
Mitmerealine kommentaar algab `/*` -iga ja lõpeb `*/` -ga

Kommentaari sisu ei mõjuta programmi käiku.

Kui skript-programm on HTML-i sees, võib selle paigutada HTML-i kommentaarimärkide `<!--` ja `-->` vahele, kuna mõni vanem brauser võib skriptkeeles kirjutatud programmi ekraanile kuvada.

1.1. JavaScript-is programmi paiknevus

JavaScript-is kirjutatud programm võib paikneda:

1) HTML keele algusmärgendites.
2) HTML faili sees märgendite <SKRIPT> vahel.
3) Eraldi failina, mis on tavaliselt laiendiga `.js` ja liidetakse HTML faili juurde.

Näited :

```
1) <H1 onmouseout="language:'JavaScript';
                   this.style.color='red';">
```

2)
```
<SKRIPT LANGUAGE=JAVASCRIPT>
  <!--
    Prorrammi_laused; // Kommentaar
  -->
</SKRIPT>
```

3) HTML keeles fail laiendiga `.html`

```
<HTML>
 <HEAD>
  <TITLE>Teretus</TITLE>
    <SCRIPT language="JavaScript"
          SRC="tere.js"
          type="text/javascript">
    </SCRIPT>
 </HEAD>
 <BODY>
 </BODY>
</HTML>
```

Tekstfail laiendiga `tere.js` , mis on samas kaustas, kus põhifail laiendiga `.html`

```
alert("TERE");
```

1.2. Skript-programmi struktuur ja programmi täitmine

Skript-programmi võib kirjutada üksikute lausetena, mida täidetakse lausete esinemise järjekorras. Programmis võib olla kasutatud tsükleid ja tingimuslikke lauseid, mida korratakse mitu korda või täidetakse teatud tingimusel.

Programmides kasutatakse sageli alamprogramme mingi standardse toimingu sooritamiseks, mille poole saab korduvalt pöörduda. JavaScript-is nimetatakse alamprogramme funktsioonideks (*function*). Programmi laused, mis on funktsioonide sees, täidetakse siis, kui selle funktsiooni poole pöördutakse.

Programmis saab kasutada ka brauserisse sisseehitatud funktsioone (õigemini küll Windows-i funktsioone ja interpretaator pöördub nende poole).

1.3. Muutujad ja nende väärtused

Muutujad tähistatakse, andes neile nimed, mida minetatakse muutujate literaalideks (*literal*). Muutuja nimi algab ladina tähega A kuni Z või a kuni z, kuid esimeseks täheks võib olla allakriipsutus (_). Järgmised märgid võivad ol-

la ka numbrid 0 kuni 9, kuid tühikuid ja erisümboleid ei tohi kasutada.
Literaalides tehakse suurtel ja väikestel tähtedel vahet.

Näiteks: `a123 ; _098 ; bBaA ;` on lubatud literaalid.
`123 _#09 ; c,c ;` ei ole literaalid, millega muutujaid tähistada.
Muutujad `A1` ja `a1` on erinevad muutujad.

Muutujaid ei pea JavaScript-is enne nende kasutamist deklareerima, vaid deklareerimine toimub esimese omistamise operaatoriga.
Näiteks: `b=5`; Toimub koos nii deklareerimine ja väärtuse omistamine.

Siiski on keeles muutujate deklareerimine võimalik ja seda tehakse võtmesõne `var` (sõnast *variable*) abil.
Näiteks: `var b; b=5;`

Muutujate deklareerimine on vajalik, kui kasutatakse palju alamprogramme, kus võivad esineda samanimelised muutujad ei tohi erinevates alamprogrammides olla sama väärtusega.

Andmed, mida muutujatele omistatakse on põhiliselt kolme tüüpi[*)]:
arvud (*Number*), tekst (S*tring*) või loogikaväärtused (*Booleon* – binaarsed).
Binaarse muutuja väärtus võib olla kas Tõene - *true* või Väär - *false*

Andmete tüüpi ei ole samuti vaja deklareerida. Sisemiselt on andmetetüübid siiski olemas, kuid need määratakse programmi kontekstist lähtudes.

Näiteks: `PUU="Kask";` siin muutuja PUU deklareeritakse kui tekstimuutuja ja saab väärtuse Kask.

`a1=25;` siis muutuja a1 on arvuline ja suurusega 25.

Stringid on jutumärkide (“) või ülakomade (‘) vahel olevad märgijadad.
Arvudele väärtuste omistamisel võib kasutada tavalisi kümnendarve, kaheksand-süsteemi arve või kuusteistkümnend-süsteemi arve. Kaheksand-süsteemi arv algab numbriga 0 , ja järgnevad numbrid on 0 kuni 7 , kuusteistkümmendarv algab märkidega 0X või 0x järgnevad numbrid on 0 kuni 9 ning tähed a kuni f või A kuni F. Arvu väärtus ei sõltu sellest, millist omistamise moodust kasutati.
Seega 17 ; `021` ja `0x11` on samaväärsed.

[*)] Muutujate tähised ehk literaalid (*literals*) kajastavad siiski rohkem kui kolme tüüpi muutujaid. Kokku on kuute tüüpi muutujaid: massiiv (*Array Literals*); loogiline (*Boolean Literals*); ujukoma (*Floating-Point Literals*); täisarv (*Integers Literals*); objekt (*Object Literals*); tekstirida (*String Literals*); Andmete seas mida muutujatele omistatakse on veel ka omistamata suurus ja nulli (*Undefined* ja *Nul)l* , millede kasutamist siin ei vaadelda.

Näiteks: kuusteistkümmend-süsteemi arvud on `0x3EFE` ja `0X0`
kaheksand-süsteemi arvud on `022` ja `07737765425`
kümnendsüsteemi arvud on `123` ja `999`

Arvud võivad peale selle olla täisarvud või ujukomaarvud (reaalarvud). Ujukomaarvudes kasutatakse koma asemel punkti. Täisarvud esitatakse ilma punktita ja reaalarvud punktiga või arvu ja kümnendastmetena.
Näiteks `8.55 ; .876` või `2.5e-5` (mis on 0,000025).
Kui arv on ühest väiksem, siis nulle enne punkti ei kirjutata.
Tegelikult säilitab JavaScript kõik arvud reaalarvudena (koos komakohtadega), mida tuleb arvude võrdlemisel arvestada.

Loogikaväärtused tekivad võrdlemise tulemusena, kuid loogikaväärtusele võib väärtuse omistada loogilise valemi abil.

Näiteks: `see=a==5&&b>3` Binaarne muutuja `see` on tõene,
kui a=5 ja b=5 ja väär, kui a=5 ja b=3

2. peatükk: Keele konstruktsioonid

JavaScript-i keele konstruktsioonideks on:

Muutujad, konstandid, avaldised, operaatorid, loogilised ja aritmeetilised tehted, massiivid, objektid, meetodid ja objektide klassid.

2.1. Avaldised ja operaatorid

Omistamise operaatoriga antakse muutujale väärtus.
Näiteks: `a=6;` ,
kuid omistamise operaatori parempoolne osa võib olla avaldis, mis sisaldab muutujaid, konstante ja operaatoreid (tehteid).

Seega avaldiste abil muudetakse programmi täitmise käigus muutujate väärtuseid.

2.2. Tehted ehk aritmeetilised ja loogilised operaatorid

2.2.1. Aritmeetilised tehted ehk avaldised

Tavaliste aritmeetiliste tehete jaoks kasutatakse tehtemärke:
`+` - liitmine; `-` - lahutamine; `*` - korrutamine; `/` - jagamine .

Peale nende on veel: % - jagamine jäägi leidmiseks. `11%2` on 1.
`++` - suurendamine ühe võrra. `a=2; a++;` a on 3.

`--` - vähendamine ühe võrra. `a=2; a--;` a on 1.

Aritmeetilisi tehteid saab teha arvuliste muutujatega. Tekstilisi muutujaid saab ainult kokku liita, mis tähendab kahe teksti järjestiku asetamist.

Kuna muutuja esmakordsel kasutamisel deklareeritakse ka muutuja ja kuna esimeses omistamise operaatoris võib olla valem, siis interpretaator aimab avaldise kontekstist, millist tüüpi muutujat deklareerida.

Näiteks:
```
a="25"+5;        // a on 255 ja tekst,
b=25+5;          // b on 30 ja arv,
c="25"-5;        // c on 20 ja arv,
d="2"+5+5+5;     // d on 2555 ja tekst,
e=25+5+"5";      // e on 305 ja tekst, kuna algul liidetakse arvud
                 // 25 + 5 = 30 ja siis muudetakse tekstiks, et
                 // lisada tekst "5",
f=d-555;         // f on 2000 ja arv, kuna lahutamisel muudeti
                 // tekst 2555 arvuks,
g=d+e;           // g on 2555305 ja tekst, sest d ja e olid tekstid.
```

2.2.2. Võrdlustehted ehk loogilised operaatorid

Võrdlusteheteks on:
võrdne (= =), ei ole võrdne (!=), suurem (>), väiksem (<), suurem või võrdne (>=) ja väiksem või võrdne (<=).

Võrrelda võib teksti (*String*) või arve ja võrdlustulemuseks on loogikaväärtus (*Binary*).

Näiteks: Kui `a=5;` siis `a==5` on Tõene (*True*)
Kui `b=5;` siis `b>6` on Väär (*False*)

Kuigi tavaliselt kasutatakse võrdlustehteid tingimuslikes lausetes, on võimalik võrdlustehte tulemus omistada binaarsele muutujale.

Näiteks: Kui `a=5,` siis `c=a==5;` siin binaarsele muutujale omistatakse väärtus Tõene (*True*)

2.2.3. Loogilised tehted

Binaarsete muutujatega on võimalik sooritada loogilisi tehteid, milledeks on : JA (&& - *Logical AND*), VÕI (|| - *Logical OR*) ja loogiline EI (! - *Logical NOT*). Loogiline EI muudab Tõese Vääraks ja Väära Tõeseks.

Näiteks: Kui `a=5;` siis `c = !(a==5);` c-le omistatakse väärtus Väär.
Kui `a=5;` ja `b=6;` siis `(a==5)&&(b<8)` on Tõene, sest 5=5 ja 6 on väiksem kui 8.

2.2.4. Loogilised tehted bittidega

Arvutis on kõik arvud ja tekstid kodeeritud kahe kahendsüsteemi arvuga “0” ja “1”. Näiteks 9 = 1001 , A = 01000001 ja a = 01100001.

Arvutis olevat infot võib käsitleda kui bittide jada ja sooritada nendega loogilisi tehteid.
Loogilised tehted bittidega on:

JA (& - *Bitwisc AND*), VÕI (| - *Bitwisc OR*), loogiline EI (~ - *Bitwisc NOT*), Välistav VÕI (^ - *Bitwisc XOR*[*)]), Nihe vaasakule (<< - *Lefi Shift*) ja Nihe paremale (>> - *Right Shift*).

Näiteks:
`15 & 9` tulemus 9 (`1111&1001 = 1001`)
`15 | 9` tulemus 15 (`1111|1001 = 1111`)
`15 ^ 9` tulemus 6 (`1111^1001 = 0110`)
`3<<2` tulemus 12 (`0011<<2 = 1100`)

2.3. Laused

Lause võib olla omistamislause või võtmesõnaga algav lause.

Võtmesõna, mida JavaScript keel tunneb, määrab lause ülesehituse.

Laused on: omistamislaused, tingimuslikud laused, valikulaused, tsüklilaused ja katkestamislaused.

*) Välistav VÕI ehk korrutamine moodul 2-ga kus: 0^0 = 0; 0^1 = 1; 1^0 = 1; 1^1 = 0; NB! Märki ^ ei leia eestikeelselt klaviatuurilt, kuid on inglisekeelsel klaviatuuril numbri 6 kohal.

2.3.1. Omistamislause

Omistamislause [**)] on tavaline valem, kus võrdusmärgist vasakul olevale muutujale omistatakse (antakse) parempoolse avaldise väärtus.
Näiteks: `a=2+b;`

Omistamise operaator võib olla ka rekursiivsel kujul, kus vasakul ja paremal on sama muutuja.
Näiteks: `a=a+2;` Sellisel rekursiivsel omistamisel võib kasutada `+=` märk.
Näiteks: `a+=2;`

Peale rekursiivse liitmise (+=) on veel :

`-= -`	lahutamine;	`a -= b;`	on `a = a - b`
`*= -`	korrutamine	`a *= b;`	on `a = a * b`
`/= -`	jagamine	`a /= b;`	on `a = a/b`
`%= -`	jagamise jäägi leidmine	`a %= b;`	on `a = a%b`

2.3.2. Tingimuslik omistamislause

Tingimusliku omistamislausega saab muutujale anda kord ühe või teise väärtuse sõltuvalt sellest, kas tingimus on täidetud (tõene) või ei ole täidetud (väär).

Tingimuslik omistamislause on kujul:

```
var muutuja = (tingimus) ? väärtus1 : väärtus1
```

Kui tingimus on tõene, omistatakse muutujale väärtus1. Kui tingimus on väär, omistatakse muutujale väärtus2.

Näiteks:

```
var iga = (age >= 18) ? "täisealine" :
"alaealine";
```

2.3.3. Tingimuslause

Tingimuslause on kujul:

```
if (tingimus) lause1; või
if (tingimus) lause1; else {lause2;}
```

Kui tingimus on tõene, täidetakse lause1.

Teise kuju korral kui tingimus on tõene, täidetakse lause1. Kui tingimus on väär, täidetakse lause2.

[**)] Omistamise operaator on võrdusmärk (=). Omistamise lause koosneb muutujast, millele see lause väärtuse omistab, omistamise operaatorist ja avaldisest, mille lahendamisel (väljaarvutamisel) leitakse suurus, mis muutujale omistatakse.

Näiteks: `if(a==6) b=9;` Kui `a`-le oli eelnevalt omistatud 6, siis `b`-le omistatakse 9.

2.3.4. Valikulause

Valikulause on kujul:

```
switch (avaldis)
{
   case märgend1 :
      lause1; break;
   case märhendN :
        lauseN; break;
   default :
        lauseA;
}
```

Valikulauses võrreldakse avaldise väärtust järgemööda kõikide märgenditega. Kui väärtus võrdub märgendiga, täidetakse märgendi järel olev lause. Kui avaldise väärtus ei võrdu ühegi märgendi väärtusega, täidetakse lause `default.`

2.3.5. Tsüklilaused

Tsüklilause kordab selles olevat lauset nii kaua, kuni määratud tingimus on tõene.
Tsüklilauseid on kolm kuju: `for`-lause, `do`-lause ja `while`-lause .

`for`-lause on kujul:

```
for (esialgne väärus; tingimus; muutus) lause1 ;
```

For-lause annab muutujale esialgse väärtuse ja seda muudetakse iga tsükliga, kuni tingimus muutub vääraks. Iga tsükliga täidetakse for-lauses olev lause1.

Näiteks: `for (i=4; i<=7; i++) document.write(i+" ");`
kus tsükliliselt väljastatakse arvud 4 5 6 7 Seega tehakse neli tsüklit.

`Do`-lause on kujul:

```
do   {lause1;lause2}  while (tingimus)
```

Do-while lause korral korratakse lauseid senikaua, kuni tingimus on veel tõene. Kontroll toimub tsükli lõpus peale lausete täitmist. Et tsükkel saaks üldse lõppeda, peab kontrollitavat muutujat tsükli sees muutma.

Näiteks:
```
i=4;
do { document.write(i+' ');i++ } while (i<=7);
```
kus tsükliliselt väljastatakse arvud 4 5 6 7 Seega tehakse neli tsüklit.

While-lause on kujul:

```
while (tingimus) {lause1;lause2}
```

While lause korral kontrollitakse tingimust enne lause täitmist. Kui tingimus on veel tõene, täidetakse lause. Et tsükkel saaks üldse lõppeda, peab kontrollitavat muutujat tsükli sees muutma ja seega peab tsüklis olema vähemalt kaks lauset.

Näiteks:
```
i=4;
while (i<=7) { document.write(i+'  ');i++}
```
kus tsükliliselt väljastatakse arvud 4 5 6 7 Seega tehakse neli tsüklit.

2.3.6. Katkestamislaused

Katkestuslausetega lõpetatakse tsükkel, pikem lause või lõpetatakse alamprogramm.

```
continue  märgend;
```
katkestab tsükli täitmise*). **Continue** võib olla täidetav mingil tingimusel
Kui ei ole tsüklit tsüklis, ei ole märgendit vaja kasutada.

Näiteks:
```
a = 4; b = 4;
CC: for (i = 0;i < 5; i++)
{ a++;  continue CC; b++;}
```
kus tehakse küll 5 tsüklit, kuid muutujat b ei suurendata ja tsükli lõpus a= 9 ning b= 4.

```
break märgend;
```
lõpetab tsükli täitmise. Kui eelmisel näitel **continue** asemel on **break** täidetakse üks tsükkel ja tsükli lõpus a=5 ja b=4. Break-i korral märgend pole kohustuslik.

```
return valem;
```
lõpetab funktsiooni täitmise ja võtab parameetriks kaasa valemi väärtuse.

```
märgend : lause;
```
märgendit kasutatakse continue ja break'i lausete korral.

2.3.7. Programmide näited

PROGRAMMINÄIDE 1: IF ELES lause

```
<html>
  <head>
    <title> Näide1 Sisestatud sõna väljastamine
```

*) Tasub märkimist, et `continue`-d saab kasutada ainult tsükli sees. Seega `for`-, `do`- ja `while`-lausetes

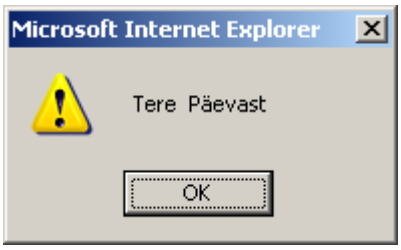
```
    </title>

      <Script language=javascript>
        SI=prompt("Tippida tekst ja OK","sisestada");
      //Pöördumine sisseehitatud funktsiooni poole
        if(SI=="EI"){;}
        else{alert("Tere  "+SI);};
      </script>
  </head>
</html>
```

Näites ei kasutata funktsioone ja esimesena täidetav lause on:

```
SI=prompt("Tippida tekst ja OK","sisestamine");
```

, mis loob sellise dialoogakna.

Peale sisestamist täidetakse tingimuslik lause. Kui sisestatud sõna oli `EI` täidetakse tühi operaator (midagi ei tehta). Teistel juhtudel aga väljastatakse teade, mille tekstiks on `Tere` ja sisestatud tekst.



PROGRAMMINÄIDE 2: prompt ja alert.

```
<html>
  <head>
    <title> Näide2 Sisestatud sõna väljastamine
    </title>

      <Script language=javascript>
        SI=prompt("Tippida tekst ja OK ");
        a=SI!="EI";
        if(a) alert("Tere  "+SI)
      </script>
  </head>
</html>
```

Esimesena täidetav lause on: `SI=prompt("Tippida tekst ja OK ");`
, kus muutujale `SI` omistatakse sisestatud tekst. Kui see tekst on `EI`, saab bi-

naarne muutuja a väärtuse Väär ja tingimuslikku lauset ei täideta (sisestatud teksti ei väljastata).

PROGRAMMINÄIDE 3: Kolm tsüklit 4,5,6,7 väljastamiseks.

```
<html>
  <head>
    <title> Näide3 Kolm tsüklilauset
    </title>
      <Script language=javascript>
        for (i=4; i<=7; i++) document.write(i+" ");
        i=4;
        do { document.write(i+' ');i++ } while (i<=7);
        i=4;
        while (i<=7) { document.write(i+'  ');i++};
      </script>
  </head>
</html>
```

See näide väljastab 4 5 6 7 4 5 6 7 4 5 6 7 . Seega sama info kolme tsükliga ja need on kirjutatud kolmel erineval kujul.

PROGRAMMINÄIDE 4: Arvu teisendamine kahendsüsteemi.

```
<html>
  <head>
    <title> Näide4 Arv kahendsüsteemi </title>
  </head>
      <Script language=jscript>
        a=prompt("Sisestada teisendatav arv","ARV");
        var st=" |";
        b=1; j=1
        do
         {
         for (i=1; i<=8; i++)
           {
           if (a&b) st="1"+st
           else st="0"+st;
           b=b<<1; // Bitt nihkub ühe koha vasakule
           } st=" | "+st; j++;
          }
         while (j<=4) // Neli baiti
        alert(a+" on "+st);
      </script>

</html>
```

Esimesena täidetav lause on: `a=prompt("Sisendada teisendatav arv","ARV")` ja seda neljabaidilist arvu korrutatakse loogiliselt b-ga, (*Bitwisc AND*). b-s on algul 1 ja seda nihutatakse vasakule (seega 1,2,4,8 jne.). Kui vastavas järgus oli "1" siis tekstireale lisatakse sümbol 1. Baitide vahele lisatakse vertikaalne joon.

2.4. Alamprogrammid ehk funktsioonid

Alamprogramm on lausete kogum, millele on antud nimi ja mida saab eraldi välja kutsuda ehk alamprogrammi poole pöörduda. Alamprogrammi on mõtet kasutada, kui selle poole põhiprogrammist pöördutakse mitu korda. Alamprogramm kirjutatakse funktsioonina ja tema kirjeldamiseks kasutatakse võtmesõna `function`* . Alamprogrammis võib kasutada formaalseid parameetreid, mis väärtustatakse peale alamprogrammi poole pöördumist.

```
function FunktsiooniNimi(argument1,argument2,..)
{
     laused, mis kasutavad argumente kui muutujaid
     mis on väärtustatud selle funktsiooni poole
     pöördumisel;
     Võib olla kasutatud 'return'
}
```

Põhiprogrammist pöördutakse selle alamprogrammi poole nii:

```
FunktsiooniNimi(argumen1_väärtus,ardument2_väärtus);
```

Kui kirjeldatud funktsioon on sisuliselt alamprogramm, siis kirjeldatakse teda nii:

```
function AlamprogrammiNimi()
{
```

* Paljudes programmeerimise keeltes on funktsioon ja alamprogramm kaks erinevat mõistet. Alamprogramm on väljakutsutav programmi osa (kasutab põhiprogrammi muutujaid) , kuid funktsioon on programmi osa, mis kasutab funktsioonisiseseid muutujaid ja formaalseid parameetreid. Funktsiooni poole pöördutakse faktiliste parameetritega, mis on selle funktsiooni argumentideks. Funktsioon omakorda omistab formaalsetele parameetritele väärtusi, mis seeläbi satuvad põhiprogrammi. Seega funktsioon on sõltumatu põhiprogrammi muutujatest ja täiesti iseseisev. JavaScript-is alamprogramm kirjutatakse kui funktsioon, kuid sellel parameetrid puuduvad.

```
      laused, mis kasutavad põhiprogrammi muutujaid

}
```

Põhiprogrammist pöördutakse selle alamprogrammi poole nii:

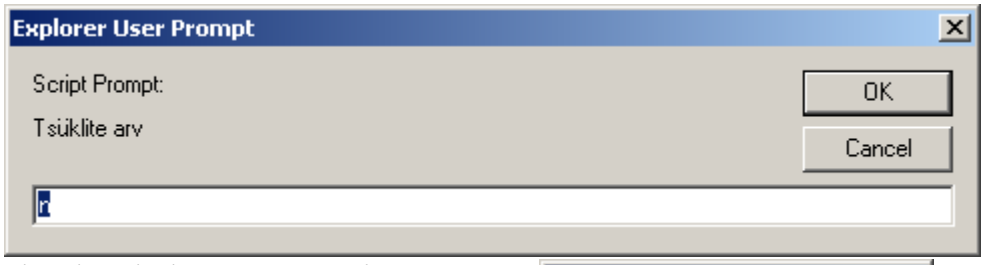
```
AlamprogrammiNimi();
```

2.4.1. Programmi näide

PROGRAMMINÄIDE 5: Funktsiooni sees on tsükkel.

```
<html>
  <head>
   <title> Näide 5 Tsüklite arv
   </title>
    <Script language=javascript>
      function tsyklite_arv(i,n)
      {
       for (j=i; j<=i+n-1; j++) document.write(j+" ");
      }
        SI=prompt("Tsüklite arv","n");
        tsyklite_arv(3,parseInt(SI));
        alert("Tehtud  "+SI+"  tsüklit")
    </script>
  </head>
</html>
```

Näites täidetakse esimese lausena `SI=prompt("Tsüklite arv","n");` ja küsitakse tsüklite arvu. Peale arvu sisestamist ja OK-le



klikkimist, pöördutakse kahe parameetritega (tsükli algus ja tsüklite arv) funktsiooni `tsyklite_arv` poole, mis väljastab iga tsükliga ühe arvu. Tsüklite lõppedes väljastatakse teade: Tehtud `n` tsüklit.

3. Peatükk: Keele objektid

3.1. Objektid, objekti omadused ja meetodid

JaavaScript on objekt-orienteeritud keel ja keele lausetega mõjutatavaid asju käsitletakse objektidena. Objektideks on HTML keeles Weebilehtede märgendid, JavaSkripti objektid, programmi poolt loodud objektid, kuid samuti teiste rakenduste (programmide) objektid. Objekti nimeks(identifikaatoriks) on punktidega eraldatud sõnad. See objekt, mis on HTML keeles ja sisaldab JavaScript-programmi on `document`. Sellel objekti alamobjektideks on HTML-keele märgendid (Näiteks `BODY, FRAME, INPUT, P` jne.) Kõigil neil objektidel on oma järjekorranumber, kuid kui soovitakse neile objektidele osutada väärtusi, on vaja anda neile nimi (`name`*) või neid identifitseerida (`ID`).

Peale HTML keeles moodustatud objektide saab kasutada ka brauserisse sisseehitatud objekte. Nendeks põhilisteks objektideks on:

- massiiv (`Array`);
- kuupäev ja kellaaeg (`Date`);
- matemaatikafunktsioonid ja -konstandid (`Math`);
- tõeväärtusobjekt (`Boolean`);
- alamprogramm (`Function`);
- numbrite määramiseks (`Number`);
- objektide üldiste omaduste määramiseks (`Object`);
- sõnade töötlemist võimaldav objekt (`String`).

Kõigil neil objektidel on omadused ja meetodite abil muudetakse objektide omadusi.

Nagu ülaltpoolt nähtub võtab termin **objekt** kokku mõisted, mille kirjeldamisel terminit **objekt** tingimata kasutada vaja ei ole. Kuna JavaSkript on objekt-orienteeritud keel, siis kasutatakse selle kirjeldamisel vastavat terminoloogiat, mis vajav veidi harjumist.

---

* Nimes(`name`) ja identifikaatoris (`ID`) tohib kasutada ainult tähti A kuni Z ja a kuni z (seega eesti keele täpittähed ja katusega tähed pole lubatud) ja numbreid 0 kuni 9) ning märki alljoonand("_"). Veel võib kasutada poolitusmärki("-"), koolonit(":"), ja punkt(perioodimärk-"."), kuid kuna viimaseid märke pole muutijate tähistamiseks lubatud kasutada, ei tasu neid ka nime andmisel pruukida.
Veel tasub märkida, et nimes ja ID-s on suured ja väikesed täded erinevad (ehk nimi on tõusutundlik).

3.1.1. Objekti omadused

Objektidel on omadused (*properties*). Objekti nime viimane sõna väljendab objekti omasust. Objekti omaduseks on näiteks tekst või arv lahtris, teksti šrift, teksti suurus või teksti värv.

Objekti omaduse üldine kuju on:

```
        rakendusprogramm.objekt.omadus = väärtus;
    või  muutuja = rakendusprogramm.objekt.omadus;
```

Objekti omadusele saab anda uue väärtuse ja objekti omaduse väärtuse võib omistada muutujale.

Näiteks: Kui HTML keeles on loodud tekstilahter, milles on punane tekst SUUR TEKST, suurusega 28pt , siis tuleb HTML keeles kirjutada nii:

```
<input type=text style="font-size:28pt; color:'red';"
value="SUUR TEKST" name="tekstilahter">
```

Kui selle teksti omadusi soovitakse muuta, peab seda tegema JavaScript-i omistamise lausetega ja selle objekti omadustele antakse uued väärtused, kasutades objekti järjekorranumbrit või nime.

Seda tehakse nii:

```
tekst=document.all.tekstilahter.value;//muutujale =omadus
document.all[7].value="UUS TEKST";     //omadusele= väärtus
document.all.tekstilahter.style.color="green";
```

See näide asendab teksti ja teeb teksti roheliseks, kuid varem selles lahtris olnud tekst säilub muutujas `tekst.` Siin programmeerija pidi teadma, et tekstilahter, millele oli antud `ID="tekstilahter"` on seitsmes objekt sellel HTML-keelsel lehel, sest kasutab sama objektile osutamiseks kord `[7]` ja teinekord `tekstilahter .`

3.1.2. Objektid ja meetodid

Funktsioonid on JavaScript-is vaadeldavad kui objektide meetodit.

JavaScript-i keeles on kasutatavad sadu sisseehitatud meetodeid, mis sisuliselt on operatsioonisüsteemi alamprogrammid ja millede poole pöördutakse JavaScripti vahenditega. Nendele alamprogrammide poole pöördumisel võib kaasa anda parameetreid. See alamprogramm (meetod) teeb parameetriga midagi (mingi tehte ja annab vastuse). Vastuse võib omistada muutujale.

```
    rakendus.objekt.meetod(parameeter);
või  muutuja=rakendus.objekt.meetod(parameter);
```

Kui kasutame brauserisse sisseehitatud objekte, peame teadma, mis on selle objekti nimi ja millised on vajaminevad parameetrid, et saada õige tulemust.
Kui kasutame sama asja väljendamiseks sõnu *funktsioon* ja *argument*, siis peame ütlema nii:

JavaScriptist saab pöörduda valmisfunktsioonide poole, kuid peab teadma selle *funktsioon*i nime, mida see funktsioon teeb ja mis on selle funktsiooni argumendid (*argument*).

Samas võib luua ka ise uusi objekte ja meetodeid ehk funktsioone, mis sisuliselt on keerulise struktuuriga andmed ja alamprogrammid. Kasutaja poolt loodud funktsioonid on kasutatavad ainult selles JavaScript programmis, kus nad on loodud.

3.1.3. Objekt `Object` ja

Objekte võib ka programmis ise luua.

Loodud objekt (objekt *object*) on analoogne muutujaga, kuid sellel on korraga mitu väärtust, ehk mitu omadust. Objekt luuakse omistamise operaatoriga.

Objekt luuakse nii:

```
poiss={nimi:"Jaan",pikkus:180,kaal:78,vanus:18};
```

või

```
poiss = new Object;
   poiss.nimi = "Jaan";
   poiss.pikkus = 180;
   poiss.kaal = 78;
   poiss.vanus = 18;
```

Siin `poiss Jaan` on kirjeldatud objektina, millel on neli omadust :
`nimi, pikkus`, `kaal` ja `vanus`

3.1.4. Objekt `function`

Meetod luuakse objektiga `function`

Loodud funktsioonile antakse nimi (*literal*) ja seda nime kasutades saab tema poole pöörduda.
Funktsioon luuakse nii:

```
nimi function(parameetet1,parameeter2,...) {laused};
```

Funktsioon võib kasutada samu muutujaid, mis on põhiprogramm (globaalsed muutujad) või deklareerida muutujaid funktsiooni sees. Viimasel juhul on tege-

mist lokaalse muutujaga ja seda saab kasutada ainule selle funktsiooni sees. Funktsioon võib oma töö lõpus võtta põhiprogrammi kaasa ühe lokaalse muutuja väärtuse, mis omistataks põhiprogrammis globaalsele muutujale.

Põhiprogrammist pöördutakse funktsiooni poole nii:

```
nimi(faktiline_parameeter1,faktiline_parameeter2,...);
```

või

```
muutuja=nimi(faktiline_parameeter1,faktiline_parameeter2,);
```

Objekti omaduseks võib deklareerida ka meetodi, mis tähendab, et selle omaduse poole pöördumisel täidetakse alamprogramm.

Näiteks:

```
function TeataPoisiPikkus(){
      alert(this.nimi+"i pikkus on "+poiss.pikkus);
    }
poiss=new Object;
     poiss.nimi = "Jaan";
     poiss.pikkus = 180;
     poiss.kaal = 78;
     poiss.vanus = 18;
     poiss.teata = TeataPoisiPikkus;
```

3.1.5. Programmide näited

PROGRAMMINÄIDE 6: Funktsiooni kasutamine.

```
<HTML>
  <HEAD>
     <TITLE> ASENDADA TEKST </TITLE>
   <SCRIPT>
     function uustekst()
      {
        document.F1.UUSTEKST.value="UUS TEKST";
      }

   </SCRIPT>
  </HEAD>

  <BODY bgColor=#90e0e0>
    <H2>   TEKSTI ASENDAMINE      </H2>
     <FORM NAME="F1">
<input type="text" name="UUSTEKST" value="VANA TEKST" >
```

```
<BR>
See tekst ei muutu <BR>
<input type="text" name="TEKST"  ><BR>
<input type="button"  value="MUUTA TEKSTI"
onClick="uustekst()">
     </FORM>

   <SCRIPT>
     F1.TEKST.value="TEKST";
   </SCRIPT>

  </BODY>
</HTML>
```

Näites kasutatakse funktsiooni, mille poole pöördutakse nupule MUUDA TEKST klikkides. Põhiosa loob kaks tekstilahtrit, millest esimeses on tekst VANA TEKST ja teise tekstilahtrisse salvestatakse teise (lõpus oleva) skript-iga sõna TEKST. Veel moodustatakse nupp, millele klikkides muudetakse esimese tekstikasti sisu. Tekstilahter on objekt, mis on vormi sees. Tekst on tekstilahtri jaoks objekti väärtus. Kuna kasutusel on ainult üks dokument, siis objekti nime esimese osa `document` võib ära jätta. Täilik nimi on: (`document.all.F1.TEKST.value`)

PROGRAMMINÄIDE 7: Formaalne parameeter ja CASE lause

```
<HTML>
 <HEAD>
  <TITLE> FUNKTSIOONI PROOV </TITLE>
<SCRIPT>
function vm(obj) {
   switch(obj.T1.value)
  { case "sinine": document.bgColor=("blue");break;
    case "punane": document.bgColor=("red");break;
    case "valge": document.bgColor=("#FFFFFF");break;
    default: window.alert("Seda värvi ei tunne");
    }           }

</SCRIPT>
 </HEAD>

 <BODY bgColor=#90e0e0>
  <H1>   VALIKUFUNKTSIOON  </H1>
         <FORM name="F">
  <input type="text" name="T1" size="15">
```

```
  <input type="button" name="button"
  value="MUUTA TAUSTA VÄRVI" onClick="vm(this.form)">

</FORM>
</BODY>
</HTML>
```

Näites kasutatakse funktsiooni, mille kirjutamisel on kasutatud formaalset parameetrit obj. See parameeter saab faktilise (tegeliku väärtuse) peale selle funktsiooni poole pöördumist. Funktsioon võrdleb objekti väärtust eestikeelsete värvinimetustega ja ühtelangemise korral omistatakse dokumendi taustale uus värv. Faktiliseks parameetriks on objekt `this.form` ja this tähendab et see on selle objekti nimi, kus sündmus toimub (seega `document.all`). Sellele lisandub `form` ja formaalne parameeter `obj` saab funktsiooni sees väärtuse `document.all.form`. Switch lauses lisatakse sellele veel `T1.value` (seega `document.all.F.T1.value`)

PROGRAMMINÄIDE 8: Objekti moodustamine

```
<html><head><title>OBJEKT POISS</title></head>
 <body>
   <script>
    poiss={nimi:"Jaan",pikkus:180,kaal:78,vanus:18};
    alert(poiss.nimi+"i pikkus on  "+poiss.pikkus);
   </script>
 </body>
</html>
```

Näites kasutatakse objekti `poiss`, mille omadus `pikkus` väljastatakse meetodiga `alert()`.

PROGRAMMINÄIDE 9: Objekti kasutamine

```
<html><head><title>OBJEKT POISS</title></head>
 <body>
   <script>
   poiss = new Object;
   poiss.nimi = "Jaan";
   poiss.pikkus = 180;
   poiss.kaal = 78;
   poiss.vanus = 18;
   alert(poiss.nimi+"i pikkus on  "+
   poiss.pikkus+"\n"+
```

```
   poiss.nimi+"i kaal on  "+poiss.kaal+"\n"+
   poiss.nimi+"i vanus on  "+poiss.vanus);
   </script>
 </body>
</html>
```

Näites luuakse uus objekt `poiss` ja selle omadustele omistatakse väärtused.

PROGRAMMINÄIDE 10: **Objekt ja meetod**

```
<html><head><title>OBJEKT ja MEETOD</title>
  <script>
    function TeataNimiJaPikkus()
    { alert(this.nimi+"i pikkus on "+poiss.pikkus);
    }
  </script>
</head>
 <body>
   <script>
     poiss=new Object;
     poiss.nimi = "Jaan";
     poiss.pikkus = 180;
     poiss.kaal = 78;
     poiss.vanus = 18;
     poiss.teata = TeataNimiJaPikkus;
     poiss.teata();
   </script>
 </body>
</html>
```

Näites objekti poiss -i omadusele teade väärtuseks meetod (funktsioon) ja selle funktsiooni poole pöördumisel on kättesaadaval selle objekti omadused.

3.2. Matemaatilised funktsioonid ehk `Math` objekt

Matemaatiliste funktsioonide, nagu siinus, cosiinus, ruutjuur ja teised, leidmiseks on brauserisse sisse ehitatud Math objekt. Matemaatiliste funktsioonide leidmiseks kasutatakse selle objekti meetodeid ja ta omadusteks on aga enamlevinud matemaatilised konstandid nagu $\pi = 3{,}1416$; $e = 2{,}71828$ ja teised.

Nende kasutamisel omistatakse muutujale väärtus või kasutatakse vahetult valemites.

```
muutuja = Math.meetod(parameeter);
```

või

```
muutuja = Math.omadus;
```

Matemaatilised meetodid on (*Math.Methods*):

abs | acos | asin | atan | atan2 | ceil | cos | exp | floor | log | max | min | pow | random | round | sin | sqrt | tan
Maremaatilese meetodi omadused on (*Math.Properties*)
E | LN2 | LN10 | LOG2E | LOG10E | PI | SQRT1_2 | SQRT2

Näiteks saab `Math` meetodiga leida siinuse ja cosiinuse väärtuseid

```
var pi     = Math.PI;      Vastus 3.141592653589793
var cosPi  = Math.cos(pi);Vastus -1
var sinPi  = Math.sin(pi);Vastus 1.2246063538223772e-16
```

Siinkohas tuleb meeles pidada, et nurk on radiaanides. Kui nurk on kraadides, tuleb see radiaanideks ümber teisendada.

Selleks tuleb kraadides nurka korrutada π-ga ja jagada 180-ga.
Näiteks nii:

```
var sina  = Math.sin(a*(Math.PI/180));
```

3.2.1. Programmi näited

PROGRAMMINÄIDE 11: Matemaatiliste funktsioonide kasutamine

```
<html>
   <head><title> LEIDA SIINUSE VÄÄRTUS </title>
   </head>
 <body>
   <script>
var pi  = Math.PI;         // Vastus 3.141592653589793
document.write(pi + "<br>");
var cosPi = Math.cos(pi/2);//Vastus 6.123031769111886e-17
document.write(cosPi + "<br>");
var sinPi= Math.sin(pi); // Vastus 1.2246063538223772e-16
document.write(sinPi + "<br>");
document.write(sinPi.toFixed(10) + "<br>");
                                  // Vastus 0.0000000000
a = 30;
document.write(Math.sin(a*(Math.PI/180)).toFixed(10));
                                  // Vastuv 0.5000000000
   </script>
 </body>
</html>
```

Selle näite vastused on sellised

3.141592653589793
6.123031769111886e-17
1.2246063538223772e-16
0.0000000000
0.5000000000

Kuigi sin(π) ja cos(π/2) peaks = 0-iga, saame vastuseks arvuti täpsuse piires nullist erineva arvu. See võib tekitada segadust, kui tuiemust kasutame võrdlustehtes. Sellest väljapääsemiseks saab tulemust ümardada. Kui väljastada arv väiksemate kohtade arvuga, toimub samuti ümardamine.

3.3. Kuupäeva ja kellaaja kasutamine ehk `Date` objekt

Arvuti emaplaadil on reaalaja kell ja CMOS-is on kuupäev mida operatsioonisüsteem haldab. **`Date`** objekti meetoditega saab seda aega küsida.

Kuupäev või kellaaega kas tervikuna või osadena saab omistada muutujale, kui varem on defineeritud Date objekt.

```
          var   tana = new Date();
    kuupaev_kell_osa = Data.getDataMethod();
```

Kuupäeva ja kellaaja objekti meetodid, millega saab aega küsida, on:
getTime() | getSeconds() | getMinutes() | getHours() | getDay() | getDate() | getMonth() | getFullYear() | getYear()*)

Näiteks: aasta, kuu ja nädalapäeva küsimiseks peab kirjutama nii:

```
var tana=new Date();
aeg     = tana.getTime();   // 1166572792930
aasta   = tana.getYear();   // 2006
kuu     = tana.getMonth()+1;// 12
paev    = tana.getDay();    // 3
```

3.3.1. Programmi näide

PROGRAMMINÄIDE 12: Kuupäeva küsimine

```
<html>
    <head><TITLE>Tänane KUUPÄEV</TITLE>
   </head>
<body>
```

*) getTime annab aja millisekundites alates 1 jaanuarist 1970 kell 00.00
getDay annab nädalapäeva järjekorranumbri, kus pühapäev on 0 ja esmaspäev = 1.

```
    <form name="F1">
      TÄNANE  AASTA, KUU ja PÄEV
      <TABLE ID="T2">
  <TR><TD>PROOV</TD><TD><input type="text" name="proov"></TD>
  </TR><TR><TD>AASTA</TD><TD><input type="text" name="aa"></TD></TR>
  <TR><TD>KUU</TD><TD><input type="text" name="kuu"></TD></TR>
  <TR><TD>PÄEV</TD><TD><input type="text" name="paev"></TD></TR>
  <TR><TD>NÄDALAPÄEV</TD><TD><input type="text" name="np"></TD></TR>
      </TABLE>
</form>
<script language="JavaScript">
      tana=new Date();
      document.F1.proov.value=tana.getTime();
      document.F1.aa.value=tana.getYear();
      document.F1.kuu.value=tana.getMonth()+1;
      document.F1.paev.value=tana.getDate();
      document.F1.np.value=tana.getDay();
</script>
    </body>
</html>
```

3.4. Tekstirida ehk objekt `String`

Tekstilõikudega manipuleerimisel kasutatakse `String` objekti.

Kui muutujale omistada tekstiline väärtus, näiteks `tekst = “see”` , siis muutuja `tekst` on `string`. Samaaegselt deklareeritakse muutuja `tekst` objektina*), mille üks omadus on tema väärtus (näites `see`), kuid tekstil on veel teisi omadusi ja objekti `String` meetoditega on võimalik neid omadusi muuta.

Näiteks:
```
var tekst = “see”;
pikkus = tekst.length; // pikkus on 2
document.write(tekst.bold());

document.write(tekst.fontsize(6));//seesee
```

Näites kasutatakse objekti omadust length ja meetodeid bold ja fontsize.

Teksti iga sümbol on 16-bitiline (2 baiti *Unicode*), mis võimaldab kasutada vene, kreeka ja kõigi teiste keelte tähti.

*) Uue objekti `String` võib luua kasutades meetodit `Strind`(), kuid see ei ole kohustuslik. Sellisel juhul:
```
var tekst = new String();
    tekst = "see";
```

3.4.1. Programmi näide

PROGRAMMINÄIDE 13: Teksti omadused ja pikkus (*lenght*)

```
<html>
  <head>
    <title> Teksti omadused
    </title>

  </head>

     <Script language=javascript>
       var tekst = new String();
       tekst = "see";
       pikkus = tekst.length;
       document.write(tekst.bold());
       document.write(tekst.fontsize(6));
       alert("pikkus on  "+pikkus);
     </script>

</html>
```

3.5. Loogiline objekt Bolean ja arvobjekt Number

Kuna JavaScript on objekt-orienteeritud keel, vaadeldakse loogilisi muutujaid ja arve kui objekte, kuigi selleks erilist vajadust ei ole.

`new Boolean(väärtis);`

*Väärtus'*eks on võrdlustehe, mille tulemuseks on Tõde (*True*) või Väär(*False*), või Tõde (*True*) ehk Väär(*False*). Kuna *Booleon* objekte kasutatakse tavaliselt tingimuslikes lausetes ja seal võib samuti kasutada võrdlustehteid, siis lihtsamate ülesannete korral ei ole vaja *Booleon* objekte defineerida.

Samuti võib ka arvmuutujaid käsitleda koi objekte, enne nende kasutamist naid deklareerides.

`new Number(väärtus)`

*Väärtus'*eks on avaldis , mille tulemuseks on arvuline väärtus. Kuna muutujale arvväärtuse omistamisel deklareeritakse see kui numbriline objekt, ei ole eelnev deklareerimine vajalik.

3.5.1. Programmi näide

PROGRAMMINÄIDE 14: Bitwise Operators & Booleon & Number

```
<html>
  <head>
    <title> Näide:Bitwise Operators & Booleon & Number
    </title>
  </head>
     <Script language=javascript>//Bitwise Operators
       a=15&9; b=15|9; c=15^9;

       alert("15&9="+a+"\n15|9="+b+"\n15^9="+c);
     </script>

     <Script language=javascript>//Boolean object
       a=true; b=5==6;           // a=true; b=false;
       //c=5<6&&a;               // c=true&&true
       c = new Boolean(5<6&&a);  // c=true&&true

       alert("a="+a+"\nb="+b+"\nc="+c);
     </script>

     <Script language=javascript>//Number object
       a=1; b=a+2;               // a=true; b=false;
       //c=a-b+1;                // c=true&&true
       c = new Number(a-b+1);    // c=true&&true

       alert("a="+a+"\nb="+b+"\nc="+c);
     </script>
</html>
```

Näites on kolm programmi, millest esimene kuulub 2.2.4 juurde teine ja kolmas 3.5 juurde. Näide demonstreerib erinevaid bittidega ja arvudega tehtavaid tehteid. Programmides, kus on kasutatud objekte Boolean ja Number on teine rida välja kommenteeritud (seega ei toimi). Kui teise rea eest kommentaari märgid `//` ära võtta ja välja kommenteerida kolmas rida on tulemus sama. See näitab, et interpretaator programmi konteksti alusel deklareerib objekti ja määrab selle tüübi.

3.6. Massiivid ehk objekt `Array`

Massiiv on järjestatud ühetüübiliste andmete hulk, milles iga massiivi elemendile on antud järjekorranumber ehk indeks. Indeksid algavad nullist. Massiivile antakse nimi ja indeksi määramiseks kasutatakse nurksulge. Kui massiiv `a` –l on `n` elementi siis neid tähistatakse `a[0],a[1],a[2], ... a[n-1].`

Massiivi loomiseks kasutatakse:

```
new Array();
new Array(elementide_arv_N);
new Array(Element0, element1,… elementN);
```

Esimene lause loob tühja massiivi, milles on 0 elementi.
Teine lause loob massiivi `N` elemendist, kuid elementidele ei omistata väärtuseid. Kolmas lause loob massiivi `N+1` elemendist ja omistab neile ka väärtused.

Kuid massiivi võib luua ka ilma võtmesõna `Array` kasutamata.
Näiteks: `var mas = ["1", "2", "3"];` loob samuti massiivi pikkusega 3, kus `mas[0]`= 1; `mas[1]`= 2 ja `mas[2]`= 3 .

Kui luua tühi massiiv, siis hiljem omistamise lausega on võimalik massiivi elementidele väärtusi omistada ja massiivi pikkust muuta.

Näiteks:

```
var mas New Array();
mas[0] = “null”;
mas[3] = “lõpp”;
```

Siin omistame tühja massiivi elemendile 0 tekstilise väärtuse `null` ja massiivis on üks element. Seejärel omistatakse elemendile 3 (neljas element) väärtus `lõpp` ja massiivis on seejärel neli elementi, kuigi elemendid `mas[1]` ja `mas[2]` on määramatu väärtusega.

Kuma massiiv on objekt, on sellel veel teisi omadusi peale elementide väärtuste ja massiivi mõjutavad meetodid.

`Massiivi_nimi.length` on massiivi pikkus ehk elementide arv massiivis. Massiivi pikkuse võib omistada muutujale ja massiivi pikkusele uue väärtuse omistamisel pikeneb või lüheneb massiiv.

Näites toodud massiivi pikkus on 3. seega:

```
pikkus = mas.length; // pikkus == 3
```

Siin muutuja `pikkus` saab värtuse 3.

3.6.1. Programmi näide

PROGRAMMINÄIDE 15: Massiiv ja massiivi pikkus

```
<html>
  <head>
    <title> Näide:Massiiv
    </title>

      <Script language=javascript>
```

```
        var mas = new Array(1,2,3);
        for (i=0; i<3; i++)
          document.write("mas["+i+"]= "+mas[i]+";  ");
          alert("mas[0]="+mas[0]+"\nmas[1]="+mas[1]
      +"\nmas[2]="+mas[2]+"\nmas.length="+mas.length);
          document.write("<br>");      //Reavahetus*)
        var mm = [1,2,3]; mm[4] = "see"
        for (i=0; i<5; i++)
          document.write("mm["+i+"]= "+mm[i]+";  ");
          alert("mm[0]="+mm[0]+"\nmm[1]="+mm[1]
      +"\x0Amm[2]="+mm[2]+"\u000Amm.length="
      +mm.length);
      </script>
  </head>
</html>
```

Näites on kahte moodi defineeritud kolmeelemendilist massiivi. Massiivile mm on lisatud viies element indeksiga 4 ja selle massiivi pikkuseks on seejärel 5. Neljas element on määramatu väärtusega (`undefined`).

4. Peatükk: Keeles kasutatavad meetodid

Brauserisse on sisse ehitatud hulgaliselt funktsioone ehk alamprogramme, mida nimetatakse meetoditeks. Neid alamprogramme ära kasutades saab kontrollida programmi tööd ja seostada teiste programmide ja rakendustega. Läbi brauseri saab kasutada ka Windows´i funktsioone.

Peale peatükis 3 objektide juurde kuuluvate meetodite on enamvajaminevad järgmised meetodid (funktsioonid):

- alert
- document.write Method
- window.open Method
- parseFloat Method
- parseInt Method

*) `document.write` genereerib Brauseri jaoks HTML keeles teksti ja reavahetuse tekitamiseks vajame HTML märgendit `<br>`. Kui aga `alert` meetodiga moodustatud dialoogaknasse on vaja reavahetust, peab kasutama erisümbolite jaoks nn. Escape sümboleid või kasutada koodi.
Seega reavahetuse kood on `alert`-i jaoks `\n`, `\x0A` või `\u000A`

4.1. Meetod `alert`

See funktsioon loob dialoogakna, mille väljastatav tekst antakse parameetriga. Funktsioon alert on tegelikult Windows´i funktsioon ja kasutatav kõikides keeltes kirjutatud programmides. Selle kasutamiseks tehakse nii:

```
window.alert(teate_sisu);
```

või

```
alert(teate-sisu)
```

Teate sisuks võib olla jutumärkides tekst, muutujad ja muud objektid. Tekstide ja muutujate ühendamiseks kasutatakse pluss (+) märki. Kui soovitakse kasutada erisümboleid (“-jutumärki, \-langjooni või teha reavahetust) tuleb kasutada nn. *escape* meetodit nende esitamiseks, mis seisneb langjoone (\) kasutamises.

Näiteks:

```
a = 125-5;                          //muutuja a=120
alert(“Arvutada\n125-5 = “+a);//väljastatakse tekst,
                                    //reavahetus tekst ja
                                    //vastus.
```

4.2. Meetod `write`

Meetodiga `write` luuakse dokumendi aknasse (see tähendab aktiivse brauseri aknasse) uus sisu. Seda meetodit saab kasutada tulemuste väljastamiseks. Kuna väljastatud infot interpreteerib brauser, kehtivad kõik HTML-i märgendid. Seega selle meetodi abil saab luua suvalise keerukosega uus leht ja seda kohesile näha.

Näiteks:

```
A = 125-5;                                  //muutuja a=120
document.write(“Arvutada <br>125-5 = “+a);//<br> viib
                                            // uuele reale
```

4.3. Meetod `open`

Meetodiga `window.open` avatakse parameetriga määratud kohtas fail. Selleks failiks on sageli HTML fail, kuid võib olla suvalist tüüpi fail, mida läbi brauseri on võimalik avada (.jpg pilt .doc tekst jne.).

Meetodi üldkuju on:

```
Uusaken = window.open( [URL] [,nimi]
[,omadused] [,tagasivõtt])
```

`URL` on avatava faili nimi ja `nimi` on

`_blank`, kui avatakse uues aknas;
`_self`, kui avatakse samasse alnasse.

Näiteks:

```
window.open("pilt.jpg","aken","top=100,left=100,
          height=200,width=300");
```

Meetodil `open` on esimeseks parameetriks avatava faili nimi, teine parameeter määrab koha (Samas tähenduses, mis `target` <a> HTML märgendis), kuhu avatav aken avatakse ja kolmas parameeter avatava akna asukoha ja suuruse.

4.4. Meetod `parseFloat ja parseInt`

Avaldiste kasutamisel ei tarvitse määrata, millist tüüpi arvudega tegemist on. Automaatset teisendamist tekstist arvuks on näidatud 2.2.1.

Kuid vahel võib vaja minna numbritega esitatud arvu (`string`) teisendamist arvmuutujaks. Selleks on meetod parseFloat, mis teisendab teksti (`string`) ujukomaarvuks ja parseInt, mis teisendab teksti (`string`) reaalarvuks.

Seda tehakse nii:

```
parseFloat("314e-2")    // tulemus 3,14
parseFloat("3.14abc")   // tulemus 3,14
parseFloat("-0.0314E+2") // tulemus -3,14
parseFloat("abc")    //Ei saa teisendada ja vastus NaN
```

Seega funktsioon teisendab sellise teksti arvuks, mis algab märgiga + või – ja arvudega. Tavaliste valemite korral funktsiooni `parseFloat` ei ole vaja kasutada. Kui valemis on arv, mida sisestati INPUT tekstiväljana (sõnana), siis on funktsioon `parseFloat` vajalik.

ParseInt on kujul:

```
parseInt(string [,arvusüsteemi_alus])
```

`arvusüsteemi_alus` on arv 2 kuni 35. arvud 0 kuni 9 on arvudena, kuid arvud üle 10-ne märgitakse ladina tähtedega (A = 10, B = 11, …, Z = 35).

Kui arvusüsteemi alust ei ole antud, võib 8-nd, 10-nd või 16-ndsüsteemi anda arvu algusnumbriga nii:

`0x` on kuueteistkümmendsüsteemi tunnuseks;
`0` on kaheksandsüsteemi tunnuseks ja
`1,2,3,4,5,6,7,8,9` on kümnendsüsteemi arv

Neid esitatakse nii:

```
parseInt("0xF");      // tulemus  15
parseInt("017");      // tulemus  15
parseInt("08");       // tulemus  0
parseInt("15.99", 10);// tulemus  15
parseInt("FX5", 16);  // tulemus  15
```

```
parseInt("1111", 2);  // tulemus  15
parseInt("15*3");     // tulemus  15
parseInt("SEE", 8);   // vastus vale ja tulemus NaN
```

Seega parseInt on laialdaste võimalustega funktsioon, mida võim samuti kasutada INPUT tekstiväljas sisestatud arvude teisendamiseks.

KOKKUVÕTE

Programmeerimine on laialdane tegevusala, mis haarab väga paljusid valdkondi arvutite juhtimises. Kuid sõna "programm" ehk ettemääratud tegevust ja "programmeerimine", kui selle tegevuse kavandamine, on kasutatav ka teistel aladel peale arvutiprogrammide koostamise. Programmeerida võib tootmist ja programm võib tähendad kontserdil või raadiosaates ettekandele tulevate palade(saadete) lihtsat järjestust.

Arvutiprogramm on aga käskude jada, mille täitmine reeglina sõltus ettetulevatest asjaoludest. Kuna kaasajal on peaaegu kõik seadmed nii pesumasinad, kohvimasinad, fotoaparaadid, rääkimata personaalarvutitest, digitaalselt juhitavad, on ka arvutiprogrammi ja arvuti mõiste avardunud. Kui varem oli seadmete programmeerimisel kasutatav masinkoodile (seadmes kasutatava mikrokontrolleri käskudele) orienteeritud programmeerimine, siis kaasajal on ka seadmetes kasutusel omamoodi operatsioonisüsteemis ja võimalus koostada programme kõrgkeeltes.

Kõrgkeeltes koostatud programm ei ole arvutis otseselt täidetav, vaid enne täitmist on vaja see kodeerida ümber protsessori käskudeks(masinkoodi) või interpreteerida(ehk läbi mängida) programmi teksti programmi lugemise käigus. Selleks peab arvutis töötama masinkoodis interpretaator, mis loeb programmi lähteteksti lauseid ja täidab neid. Selline protsess on küll sadu kordi aeglasem, kui kompileeritud programmi täitmine, kuid võimaldab kiiremini koostada ja muuta programme, samas säästa aega kompileerimise arvelt. Tavaliselt on interpreteerivad keele "inimsõbralikumad" ja ei vaja programmeeritava seadme põhjalikku tundmist, mis omakorda kiirendab seadmete kasutuselevõttu.

Kogu arvutite ja programmeerimiskeelte ajaloo jooksul on püütud välja mõelda "täiuslikku käsusüsteemi" ja "kõikehaaravat universaalset programmeerimise keelt", mis rahuldaks kõike mõeldavat. Ja jälle on ilmunud uus ja efektiivsem käsustik, arvuti arhitekyuur ja keel. Paistab, et arvutikasutajatel ja programmeerijatel ei jää muud üle, kui kiiresti õppida selgeks uus ja unustada vana, sest sellist lõpplikku arvutit ja keelt ei tule IIALGI.

Seega paistab, et interpreteerivatel süsteemidel ja Script-keeltel on tulevikku ja nende kiire omandamine vajalik ja

SEE TÖÖ ANNAB PIKAKS AJAKS KINDLA LEIVA!

Kaanekujundus J. Puhang

Trükitud Tallinna Polütehnikumis
2007