Tallinna Ülikool

Veebirakenduste jätkukursus

Jaagup Kippar

2014

Sisukord

Sissejuhatus

HTML5 ja Javaskripti valis veebi arengut suunav W3C põhilisteks tehnoloogiateks, mille peal võiksid veebirakendused lähemate aastate jooksul töötada. Aegade jooksul oli tarvilikke lisapluginaid juba nõnda palju saanud, et uue viisaka ja enamvähem vajalike omadustega veebilehitseja kokkupanek hakkas liialt keeruliseks minema. Selle peale tehti HTMLile põhjalik uuenduskuur ning vaikselt püüab ka Javaskript ühtseid standardeid järgida. Tulemuseks on loodetavasti veeb, mida mitmesugused seadmed taas viisakalt näidata suudavad ning mille kuvamiseks ei pea muuhulgas ka väiksemad seadmed enesesse mitmesuguseid mitmekümne megabaidiseid pluginate käivituskeskkondi laadima.

Javaskript on suhteliselt sarnasena olemas olnud juba 1990ndate keskelt. Samas keerukamaid ja objektorienteeritud lahendusi pole sellega varem kuigi palju kokku pandud. Viimastel aastatel on sedasorti lahendusi hulgem lisandunud ning seetõttu ka vastavad õppematerjalid tarvilikud, et jaguks oskusi ja julgust Javaskripti abil ka keerukamaid veebilahendusi kokku panna.

Meeldetuletus

Alustuseks mõned lihtsad näited Javaskripti abil töötavatest veebirakendustest. Et kelle jaoks tegemist uue keelega või pole lihtsalt kaua kokku puutunud, siis saaks mõningase tunde kätte.

Kalkulaator

Arvutamise jaoks peab ikka olema koht andmete sisse panekuks ning koht tulemuse nägemiseks. Tekstiväli ning div-kiht sobivad selliseks komplektiks küll. Nupuvajutuse peale saab käima panna funktsiooni. Funktsioonid hea defineerida lehe päiseosas, siis nad hilisema kasutuse tarbeks olemas. Elementide poole pöördumiseks sobib käsklus document.getElementById - mida Javaskripti-põhiste lahenduste puhul tuleb päris sageli välja kutsuda. Mõned raamistikud teevad selle käsu küll lühemaks, praegu kasutame pikka kuju.

Näite juures arvutatakse poes oleva letihinna põhjal selles sisalduv käibemaksu osa.

```
<!DOCTYPE html>
<html>
  <head>
     <title>Käibemaksu arvutus</title>
       <meta charset="UTF-8" />
       <script type="text/javascript">
          function arvuta(){
               var letihind=
                   parseFloat(document.getElementById("letihind").value);
               var km=letihind*20/120;
//             km=Math.round(km*100)/100; //ümardus 2 kohta pärast koma.
               km=km.toFixed(2);
               document.getElementById("vastus").innerHTML=km;
            }
       </script>
  </head>
  <body>
     <h1>Katsetuste leht</h1>
       Palun sisesta letihind:
        <input type="text" id="letihind" />
        <input type="button" value="OK" onclick="arvuta();" />
        <div id="vastus">
          Vastuse koht.
        </div>
  </body>
</html>
```

Ülesandeid

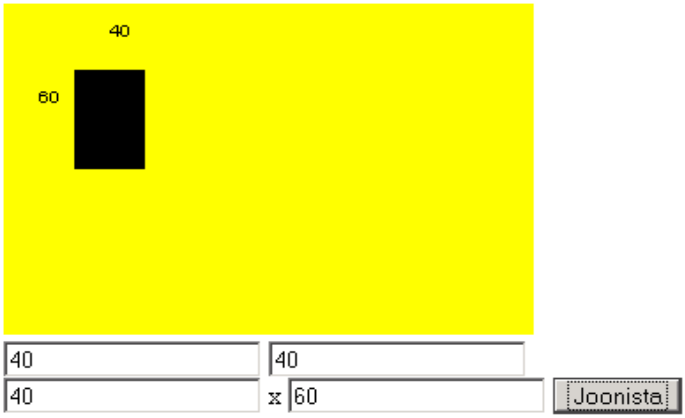
- Pane näide käima
- Koosta kalkulaator valuutavahetusel kättesaadava summa arvutamiseks sisseantud summa, kursi ja komisjonitasu järgi.

Ristkülik ekraanil

HTML5ga koos tuli veebilehele joonistamiseks lõuend ehk canvas. Loojate plaanide järgi saab sellest lähiaastate veebigraafika valitseja. Eks aeg näitab, mis tulevik toob. Aga otseste joonistuskäskudega on küll mugavam jooni ja ringe tekitada, kui selleks kavalalt kombineeritud värvitud taustaga kihte veebilehele võluda. Järgneva joonistuskäsu juures küsitakse lõuend-tahvlilt graafiline kontekst (nagu sulepea) ning tekstikastidest ette antud andmete järgi kuvatakse ekraanile ristkülik.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Joonis</title>
    <script type="text/javascript">
      function joonista(){
         var g=document.getElementById("tahvel").getContext("2d");
         var laius=parseInt(document.getElementById("laiuskast").value);
         var korgus=parseInt(document.getElementById("korguskast").value);
         var vasakult=parseInt(document.getElementById("vasakkast").value);
         var ylalt=parseInt(document.getElementById("ylakast").value);
         g.fillStyle="yellow";
         g.fillRect(0, 0, 300, 200);
         g.fillStyle="black";
         g.fillRect(vasakult, ylalt, laius, korgus); //x, y, laius, kõrgus
         g.fillText(korgus, vasakult-20, ylalt+20);
         g.fillText(laius, vasakult+20, ylalt-20);
      }
    </script>
  </head>
  <body>
    <h1>Joonis</h1>
    <canvas id="tahvel" width="300" height="200"
              style="background-color:yellow"></canvas><br />
    <input type="text" id="vasakkast"  value="40" />
      <input type="text" id="ylakast"  value="40" /><br />
    <input type="text" id="laiuskast"  value="40" /> x
    <input type="text" id="korguskast" value="60" />
    <input type="button" value="Joonista" onclick="joonista()" />
  </body>
</html>
```

Joonis

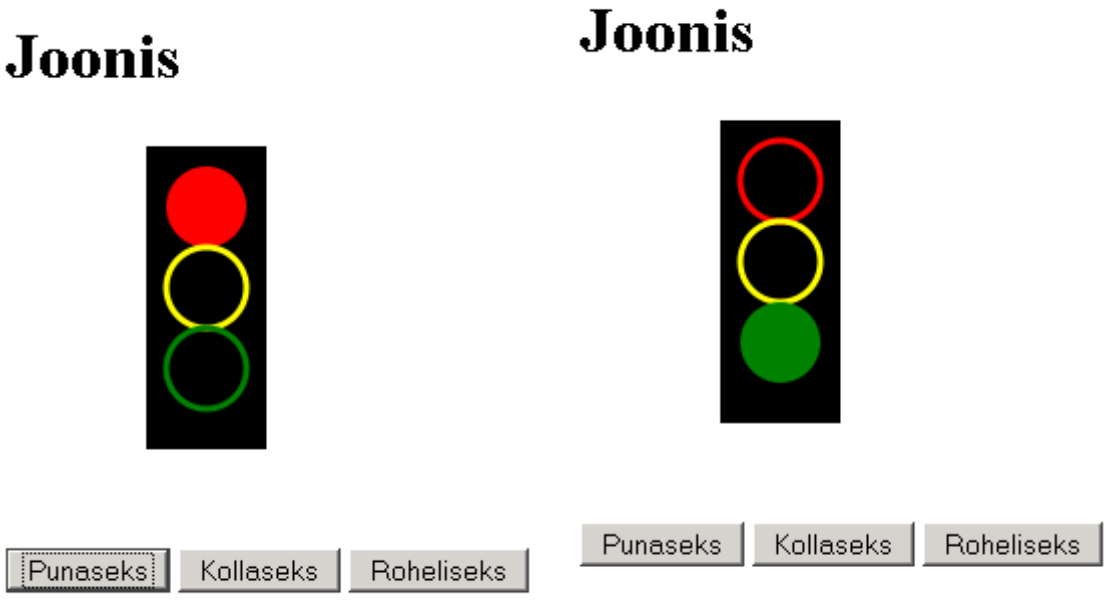


Foor, seisundid

Märgatav osa siinsetest materjalidest tutvustab objektidega majandamist veebilehel. Objekti omapäraks on tema juurde kuuluvad andmed ning samuti käsklused, mis oma tööks andmeid kasutada saavad. Lihtsamatel juhtudel võib sarnase tulemuse saada aga ka tavalisel veebilehel toimetades. Lehtki on suhteliselt eraldiseisev üksus, kus globaalmuutujatena võimalik hoida lehe piires kättesaadavaid andmeid ning neid siis oma käskude juures kasutada. Näitena toodud valgusfoor, kus seisundimuutujas parajasti kirjas põlev värv. Muutuja järgi otstustades joonistatakse sobivasse kohta seda värvi ring. Allpool nuppudega valides saab määrata seisundiks sobiva värvi ning selle peale pannakse pildi uuendamiseks tööle joonistuskäsk.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Joonis</title>
    <script type="text/javascript">
          var seisund="punane";
      function joonista(){
         var g=document.getElementById("tahvel").getContext("2d");
         g.fillStyle="white";
         g.fillRect(0, 0, 300, 200);
         g.fillStyle="black";
         g.fillRect(70, 5, 60, 150);
         g.lineWidth=3;
         g.strokeStyle="red"; g.fillStyle="red";
         g.beginPath();
         g.arc(100, 35, 20, 0, 2*Math.PI, true);
         if(seisund=="punane"){g.fill()} else {g.stroke();}
         g.strokeStyle="yellow"; g.fillStyle="yellow";
         g.beginPath();
         g.arc(100, 75, 20, 0, 2*Math.PI, true);
         if(seisund=="kollane"){g.fill()} else {g.stroke();}
         g.strokeStyle="green"; g.fillStyle="green";
         g.beginPath();
         g.arc(100, 115, 20, 0, 2*Math.PI, true);
         if(seisund=="roheline"){g.fill()} else {g.stroke();}
      }
      function punaseks(){seisund="punane"; joonista();}
```

```
      function kollaseks(){seisund="kollane"; joonista();}
      function roheliseks(){seisund="roheline"; joonista();}
    </script>
  </head>
  <body onload="joonista();">
    <h1>Joonis</h1>
    <canvas id="tahvel" width="300" height="200"
              style="background-color:yellow"></canvas><br />

    <input type="button" value="Punaseks" onclick="punaseks()" />
    <input type="button" value="Kollaseks" onclick="kollaseks()" />
    <input type="button" value="Roheliseks" onclick="roheliseks()" />
  </body>
</html>
```



Ülesandeid

- Pane näide käima
- Koosta taksohinna kalkulaator. Näha on sisseistumistasu ja kilomeetrihind. Väljastatakse sõiduhind vastavalt sisestatud kilomeetrite arvule.
- Lisa kalkulaatorile seisundid öö ja päev. Vastavalt muutub kalkulaatori juures olev pilt. Öösel on hinnad kõrgemad.

Objektid

Tollikalkulaator objektina

Kõigepealt meeldetuletuseks tavaline lihtne kalkulaator.

```
<!doctype html>
<html>
  <head>
    <title>Kalkulaator</title>
        <script>
          function arvuta(){
             var tollidearv=parseFloat(document.getElementById("kast1").value);
                 document.getElementById("vastus").innerHTML=tollidearv*2.54;
          }
        </script>
  </head>
  <body>
     <h1>Arvutamine</h1>
         Tollid:
         <input type="text"   id="kast1" />
         <input type="button" value="Sentimeetriteks" onClick="arvuta();" />
         <div id="vastus"></div>

  </body>
</html>
```

Järgmisena tehakse sisestuskastidest ning arvuta-käsklusest komplekt nimega Arvutaja. Nii on arvuta-käsklusel võimalik this-muutuja kaudu tekstiväljade poole pöörduda ning ei pea keerulisemal lehel enam pikemalt muretsema, et kust andmed kätte saab. Siin näites küll on veel kastide nimed otse koodi sisse kirjutatud, kuid ka selle saab tulevikus paindlikumaks muuta. Lehe sisumärgendi body atribuut onload teatab, et lehe avanemisel tuleb käivitada funktsioon lehealgus(). Selle tulemusena luuakse muutujasse a1 Arvutaja-tüüpi objekt ehk eksemplar. All sentimeetriarvutuse nupu juures sobib tulemuse saamiseks käima panna juba käsklus a1.arvuta(), mis juba teab, millised andmed kust võtta ja kuhu vastus panna.

```
<!doctype html>
<html>
  <head>
    <title>Kalkulaator</title>
        <script>
          var a1;
          function Arvutaja(){
             this.kast=document.getElementById("kast1");
             this.vastusekiht=document.getElementById("vastus");
             this.arvuta=function(){
                    this.vastusekiht.innerHTML=parseFloat(this.kast.value)*2.54;
             }
```

```
          }

          var a1;
          function lehealgus(){
            a1=new Arvutaja();
          }
        </script>
  </head>
  <body onload="lehealgus();">
     <h1>Arvutamine</h1>
         Tollid:
         <input type="text"   id="kast1" />
         <input type="button" value="Sentimeetriteks" onClick="a1.arvuta();" />
         <div id="vastus"></div>

  </body>
</html>
```

Kalkulaator töötab nii nagu ennegi, lihtsalt koodi sees on nüüd koht Arvutaja-objekt oskuste kirjeldamiseks.

Arvutamine

Tollid: 5 Sentimeetriteks
12.7

Järgmise sammuga muudame arvutaja iseseisvamaks - lisame ka oskuse ennast ekraanile kuvada. Ehk siis väljastada näitamiseks vajaliku HTMLi. Kõigepealt luuakse Arvutaja sisse funktsioonid nimega algus ning arvuta. Loomise lõpul käivitatakse funktsioon algus(). Võtmesõna this alguse juures näitab, et tegemist pole lihtsalt alguse-nimelise funktsiooniga (mis võiks ka kusagil mujal loodud olla), vaid Arvutaja objekti külge kuuluva funktsiooniga algus(). Hilisema pöördumise lihtsustamiseks luuakse Arvutaja külge muutuja this.kiht ning järgmise käsuga määratakse kihi sisse HTML-kood. Samuti lisanduvad Arvutaja külge muutujad kast ja vastusekiht. Arvutamisfunktsiooniks arvuta, mille juures siis tollidest sentimeetrid tehakse.

Lehe avamisele body-elemendi atribuut onload ütleb, et tuleb käivitada funktsioon lehealgus(). Seal antakse väärtus eelnevalt loodud muutujale nimega a1. Väärtuseks saab Arvutaja tüüpi objekt, kellele kihi nimeks antakse ette kiht1. Esialgu kannatab sellisena panna vaid ühe arvutaja lehele, sest muutuja nimi a1 on Arvutaja this.algus-funktsioonis a1.arvuta() juures sisse kirjutatud.

```
<!doctype html>
<html>
  <head>
    <title>Kalkulaator</title>
        <script>
          function Arvutaja(kihinimi){
                 this.algus=function(){
                   this.kiht=document.getElementById(kihinimi);
                   this.kiht.innerHTML=
                    "Tollid: <input type='text'   id='kast1' /> "+
                    "<input type='button' value='Sentimeetriteks' "+
                                         "onClick='a1.arvuta();' /> "+
                    "<div id='vastus'></div>";
                    this.kast=document.getElementById("kast1");
                    this.vastusekiht=document.getElementById("vastus");
```

```
                }
                this.arvuta=function(){
                   this.vastusekiht.innerHTML=parseFloat(this.kast.value)*2.54;
                }
                this.algus();
            }

            var a1;
            function lehealgus(){
              a1=new Arvutaja("kiht1");
            }
          </script>
  </head>
  <body onload="lehealgus();">
     <h1>Arvutamine</h1>
     <div id="kiht1"></div>

  </body>
</html>
```

Arvutamine

Tollid: 5 Sentimeetriteks
12.7

Ülesandeid

- Pane näited käima
- Loo sarnaselt näitele kalkulaator ringi pindala leidmiseks raadiuse järgi.

Mitu kalkulaatorit lehel

Üksiku kalkulaatori saab ka niisama veebilehele kirjutada. Kui neid aga vaja panna hulgem ning mitmesuguste parameetritega, siis on objektide loomisest rohkem kasu. Järgmises näites antakse Arvutajale ette kihi id, kuhu end paigutada ja töökorda sättida. Käsklus

```
window[kihinimi+"_kalkulaator"]=this;
```

loob akna ehk lehe peaobjekti külge muutuja, mille nime sees on kihi nimi ja sõna "kalkulaator", ehk siis esimese Arvutaja loomisel

```
new Arvutaja("kiht1");
```

saab objekti nimeks nõnda kiht1_kalkulaator. Nupu loomisel öeldakse nupule, et tuleb sellenimelise objekti juures välja kutsuda käsklus arvuta()

```
"<input type='button' value='Sentimeetriteks'
     onClick='"+kihinimi+"_kalkulaator.arvuta();' /> "
```

Selle järgi teatakse just õigest kohast sisestatud arv võtta, sest

```
this.kast=document.getElementById(kihinimi+"_kast1");
```

paneb arvutaja külge muutuja nimega kast. Pärast arvutuse juures võetakse sealt

```
this.vastusekiht.innerHTML=parseFloat(this.kast.value)*2.54;
```

```
<!doctype html>
<html>
  <head>
    <title>Kalkulaator</title>
        <script>
          function Arvutaja(kihinimi){
             this.algus=function(){
                this.kiht=document.getElementById(kihinimi);
                window[kihinimi+"_kalkulaator"]=this;
                this.kiht.innerHTML=
                    "Tollid: <input type='text'   id='"+kihinimi+"_kast1' /> "+
                    "<input type='button' value='Sentimeetriteks' "+
                    "onClick='"+kihinimi+"_kalkulaator.arvuta();' /> "+
                    "<div id='"+kihinimi+"_vastus'></div>";
                this.kast=document.getElementById(kihinimi+"_kast1");
                this.vastusekiht=document.getElementById(kihinimi+"_vastus");
             }
             this.arvuta=function(){
                    this.vastusekiht.innerHTML=parseFloat(this.kast.value)*2.54;
             }
             this.algus();
          }


          function lehealgus(){
            new Arvutaja("kiht1");
            new Arvutaja("kiht2");
          }
        </script>
  </head>
  <body onload="lehealgus();">
     <h1>Arvutamine</h1>
         <div id="kiht1"></div>
         <div id="kiht2"></div>
  </body>
</html>
```

Arvutamine

Tollid: 3 Sentimeetriteks
7.62
Tollid: 6 Sentimeetriteks
15.24

Ülesandeid

- Pane näide käima
- Loo sarnaselt lehele ka mitu kalkulaatorit ringi pindala leidmiseks raadiuse järgi

Kalkulaatorite parameetrid

Eelnevates näidetes vaadatud kalkulaatorid suutsid hakkama saada vaid ühe operatsiooniga. Kujundus ning arvutuskäik oli juba koodi sisse kirjutatud ning valida sai vaid loodud kalkulaatori asukoha ja nende loomise koguse järgi. Arvutaja saab kirjutada ka paindlikumalt - lisaks kihi nimele antakse järgmises näites ette ka sissestuskasti juurde tulev selgitav tekst, vajutusnupule kuvatav tekst ning arvutuskoefitsient kahe suuruse vahel.

```
<!doctype html>
<html>
  <head>
    <title>Kalkulaator</title>
        <script>
          function Arvutaja(kihinimi, kastitekst, nuputekst, koefitsient){
             this.algus=function(){
               this.kiht=document.getElementById(kihinimi);
               window[kihinimi+"_kalkulaator"]=this;
               this.kiht.innerHTML=
                kastitekst+": <input type='text'   id='"+kihinimi+"_kast1' /> "+
                "<input type='button' value='"+nuputekst+"' onClick='"+
                   kihinimi+"_kalkulaator.arvuta();' /> "+
                "<span id='"+kihinimi+"_vastus'></span>";
               this.kast=document.getElementById(kihinimi+"_kast1");
               this.vastusekiht=document.getElementById(kihinimi+"_vastus");
               this.koefitsient=koefitsient;
             }
             this.arvuta=function(){
              this.vastusekiht.innerHTML=
                   parseFloat(this.kast.value)*this.koefitsient;
             }
             this.algus();
          }


          function lehealgus(){
            new Arvutaja("kiht1", "Eurod", "Dollariteks", 1.3);
            new Arvutaja("kiht2", "Tollid", "Sentimeetriteks", 2.54);

          }
        </script>
  </head>
  <body onload="lehealgus();">
     <h1>Arvutamine</h1>
         <div id="kiht1"></div>
         <div id="kiht2"></div>


  </body>
</html>
```

Tulemusena võib edasi luua juba igasugu kalkulaatoreid, mille puhul arvutus piirdub ühe suhtega ehk korrutus/jagamistehtega.

Arvutamine

Eurod: 11 Dollariteks 14.3
Tollid: 5 Sentimeetriteks 12.7

Ülesandeid

- Pane näide käima
- Loo sama näite põhjal kalkulaator käibemaksu arvutamiseks letihinna järgi
- Koosta sarnaselt kihile pandav kalkulaator, kus näidatakse soovitud arv tekstikaste ning nupuvajutuse peale väljastatakse neisse sisestatud arvude aritmeetiline keskmine.

Andmete talletamine objektis

Eelnevate arvutuste juures anti sisestuse põhjal koheselt vastus ning midagi eraldi säilitada pole vaja. Objekti üheks kasulikuks omaduseks on aga tema olek, ehk siis võime väärtusi oma eluea jooksul säilitada. Siin lihtsama näitena arvuline meelespeetav kogus ning nupud selle koguse suurendamiseks ja vähendamiseks. Tulemuse näitamisks funktsioon kuva() ning nuppude külge kinnitatud funktsioonid suuremaks() ja väiksemaks() soovitud muutuse tarbeks.

```
<!doctype html>
<html>
  <head>
    <title>Kalkulaator</title>
        <script>
          function Laohaldus(kihinimi, kogus){
             this.algus=function(){
                this.kiht=document.getElementById(kihinimi);
                this.kogus=kogus;
                window[kihinimi+"_ladu"]=this;
                this.kiht.innerHTML=
                  "<input type='button' value='&lt;' "+
                     "onClick='"+kihinimi+"_ladu.v2iksemaks();' /> "+
                  "<input type='text' id='"+kihinimi+"_vastus' "+
                     "style='width: 50px' disabled />"+
                  "<input type='button' value='&gt;' "+
                  " onClick='"+kihinimi+"_ladu.suuremaks();' /> ";
                this.vastusekiht=document.getElementById(kihinimi+"_vastus");
                this.kuva();
             }
             this.kuva=function(){
                this.vastusekiht.value=this.kogus;
             }
             this.v2iksemaks=function(){
                this.kogus--;
                this.kuva();
             }
             this.suuremaks=function(){
                this.kogus++;
```

```
                this.kuva();
              }
              this.algus();
            }


            function lehealgus(){
              new Laohaldus("kiht1", 100);
              new Laohaldus("kiht2", 50);
            }
          </script>
  </head>
  <body onload="lehealgus();">
     <h1>Arvutamine</h1>
         <div id="kiht1"></div>
         <div id="kiht2"></div>
  </body>
</html>
```

Arvutamine

Ülesandeid

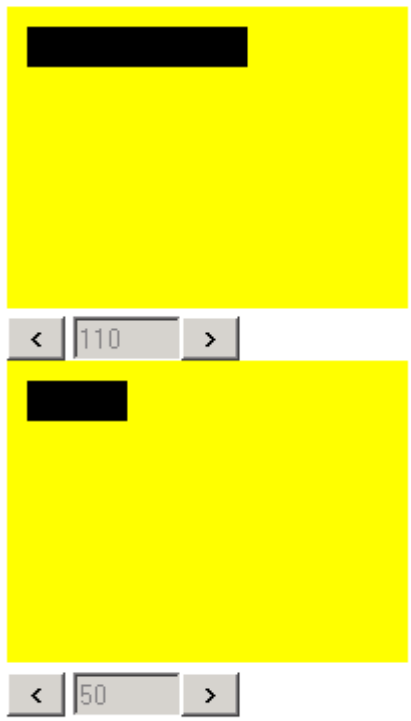
- Pane näide käima
- Lisa objekti kujundusse teine tekstiväli, kus saab määrata, millise koguse võrra olemasolevat väärtust kasvatatakse või kahandatakse
- Tekstivälja asemel saab muudetava suuruse valida rippmenüüst
- Lisa nupp laoseisu nullimiseks

Objekti graafiline väljund

HTML5 juurde kuuluvat Canvast saab sarnaselt sisestuse ja väljundi juures kasutada nagu teisigi veebilehe elemente. Lihtsalt siitkaudu on võimalik väljundit märgatavalt värvilisemaks muuta. Kuvamise funktsioonile lisatakse täiendus: lisaks arvu näitamisele tekstiväljas tehakse lõuendile ka vastava pikkusega kast.

```
<!doctype html>
<html>
  <head>
    <title>Kalkulaator</title>
        <script>
          function Laohaldus(kihinimi, kogus){
             this.algus=function(){
               this.kiht=document.getElementById(kihinimi);
               this.kogus=kogus;
               window[kihinimi+"_ladu"]=this;
               this.kiht.innerHTML=
                 "<canvas id='"+kihinimi+"_joonis' "+
                    "width='200' height='150' "+
```

```
                   "style='background-color: yellow' ></canvas><br />"+
                "<input type='button' value='&lt;' "
                  +" onClick='"+kihinimi+"_ladu.v2iksemaks();' /> "+
                "<input type='text' id='"+kihinimi+"_vastus'"+
                  "style='width: 50px' disabled />"+
                "<input type='button' value='&gt;' "
                    +" onClick='"+kihinimi+"_ladu.suuremaks();' /> ";
                this.vastusekiht=document.getElementById(kihinimi+"_vastus");
                this.joonis=document.getElementById(kihinimi+"_joonis");
                this.kuva();
             }
             this.kuva=function(){
                this.vastusekiht.value=this.kogus;
                var g=this.joonis.getContext("2d");
                g.clearRect(0, 0, 200, 150);
                g.fillRect(10, 10, this.kogus, 20);
             }
             this.v2iksemaks=function(){
                this.kogus--;
                this.kuva();
             }
             this.suuremaks=function(){
                this.kogus++;
                this.kuva();
             }
             this.algus();
          }


          function lehealgus(){
            new Laohaldus("kiht1", 100);
            new Laohaldus("kiht2", 50);
          }
        </script>
  </head>
  <body onload="lehealgus();">
     <h1>Arvutamine</h1>
         <div id="kiht1"></div>
         <div id="kiht2"></div>
  </body>
</html>
```



Ülesandeid

- Pane näide käima
- Lisa tekstiväli näitamaks, mitme ühiku jagu objektis olevat kogust suurendada või vähendada.

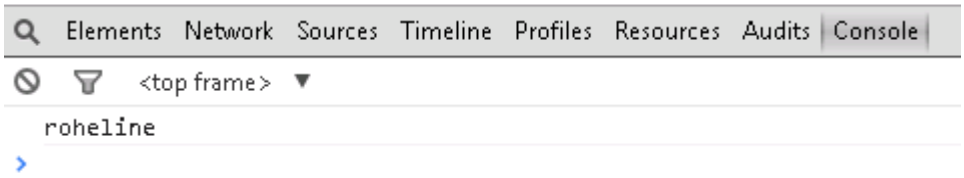
Foor

Objektidega harjumine paistab mõnelegi keeruline olema. Järgnevalt harjumiseks näide, kuidas valgusfoori tööd veebilehel objektina jäljendada. Esmalt lihtsaim variant, kus fooril on üks muutuja ehk väli, mis peab meeles põleva tule värvi. Käsklustega saab seda värvi küsida ning muuta.

```
<!doctype html>
<html>
  <head>
    <title>Foor</title>
    <script>
       function foor(){
          this.tuli="roheline";
          this.kysiTuli=function(){return this.tuli;}
          this.muudaTuli=function(uusTuli){this.tuli=uusTuli;}
       }
       function leheAlgus(){
          var f1=new foor();
          console.log(f1.kysiTuli());
          f1.muudaTuli("kollane");
          document.getElementById("vastus").innerHTML=
            f1.kysiTuli();
       }
    </script>
```

```
  </head>
  <body onload="leheAlgus()">
     <div id="vastus"></div>
  </body>
</html>
```

Käsklus console.log trükib tulemuse testimiseks mõeldud konsooliaknasse (enamasti saab lahti veebilehisejas klahvivajutusega F12)



Lehel kihis näidatud vastus on lihtsalt tekstina seal:

kollane

Mitu foori

Objektitüüp ning objekt ise on eri asjad. Ehk siis ühte tüüpi võib korraga olla mitu objekti. Ning vahel võib mõne tüübi juures töötav objekt ka sootuks puududa. Siin näites siis korraga kaks töötavat foori. Sündides mõlemad rohelis tulega. Edasi muudetakse üks kollaseks ja teine punaseks. Ning taas saab tulemused seest välja küsida:

```
<!doctype html>
<html>
  <head>
    <title>Foor</title>
    <script>
       function foor(){
          this.tuli="roheline";
          this.kysiTuli=function(){return this.tuli;}
          this.muudaTuli=function(uusTuli){this.tuli=uusTuli;}
       }
       function leheAlgus(){
          var f1=new foor();
          var f2=new foor();
          f1.muudaTuli("kollane");
          f2.muudaTuli("punane");
          document.getElementById("vastus").innerHTML=
            "esimene: "+f1.kysiTuli()+
            ", teine: "+f2.kysiTuli();
       }
    </script>
  </head>
  <body onload="leheAlgus()">
     <div id="vastus"></div>
  </body>
</html>
```

esimene: kollane, teine: punane

Ülesandeid

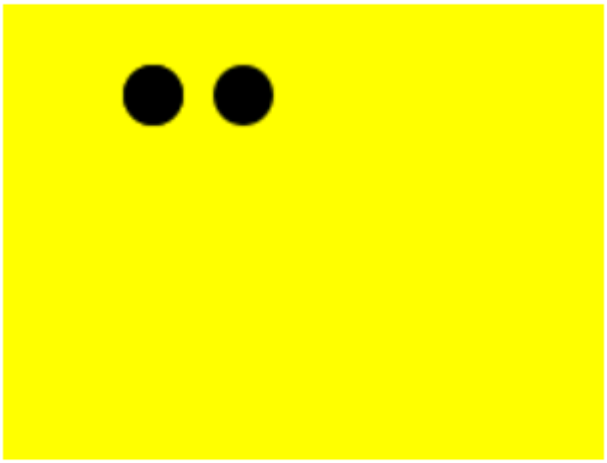
- Pane näited tööle
- Koosta tüüp Lamp, väljaks "seisund" väärtusega true. Koosta käsklus kasSees(), mis siis väljastab seisundi väärtuse, testi.
- Lisa käsklus uusSeisund(asend), mille juures siis vastavalt määratakse seisundi uus väärtus. Testi.
- Lisa käsklus seisundTekstina(), kus siis vastavalt seisundi väärtusena väljastatakse tekstina "põleb" või "ei põle".
- Lisa käsklus vahetaSeisund(), mis siis muutuja "seisund" väärtuse muudab vastupidiseks. Katseta.
- Loo korraga mitu lampi, igaühel oma seisund. Vaheta lampide seisundeid ning veendu, et igaüks näitab õigesti.
- Tee seisundi vahetamiseks ning küsimiseks igale lambile eraldi nupupaar. Katseta ning veendu, et lülitamine töötab õigesti.
- Lisa lambile graafiline liides. Sisselülitatud lambi puhul on näha seest täis ring, väljalülitatud lambi puhul tühi ringjoon.

Kujundite lisamine platsile

Objektitüüpe võib lehel olla mitu. Järgmises näiteks on tüüpideks Pall ja Plats. Pall mõistab oma asukoha meeles pidada ning etteantud graafilise konteksti kaudu joonistada. Plats kuvab end kollase ristkülikuna, tema küljes kujundite massiivis on pallid ning ta suudab nad välja joonistada. Algusnäide koostab platsi kahe palliga ning kuvab nad ekraanile.

```
<!doctype html>
<html>
  <head>
    <title>Kujundid</title>
    <script>
       function Pall(x, y, r){
          this.x=x;
          this.y=y;
          this.r=r;
          this.joonista=function(g){
           g.beginPath();
           g.arc(this.x, this.y, this.r, 0, 2*Math.PI, true);
           g.fill();
          }
       }
       function Plats(kihiId){
          window[kihiId+"_joonis"]=this;
          this.alusta=function(){
             var sisu=
               "<canvas id='"+kihiId+"_tahvel' width='200' height='150' "+
                " style='background-color: yellow' ></canvas><br/>";
             document.getElementById(kihiId).innerHTML=sisu;
             this.kujundid=new Array();
             this.tahvel=document.getElementById(kihiId+"_tahvel");
          }
          this.lisaKujund=function(kujund){
            this.kujundid.push(kujund);
            this.joonista();
          }
          this.joonista=function(){
```

```
            var g=this.tahvel.getContext("2d");
            for(var i=0; i<this.kujundid.length; i++){
               this.kujundid[i].joonista(g);
            }
         }
         this.alusta();
      }

      function algus(){
        kiht1_joonis=new Plats("kiht1");
        kiht1_joonis.lisaKujund(new Pall(50, 30, 10));
        kiht1_joonis.lisaKujund(new Pall(80, 30, 10));
      }
    </script>
  </head>
  <body onload="algus();">
     <div id="kiht1"></div>
  </body>
</html>
```



Ülesandeid

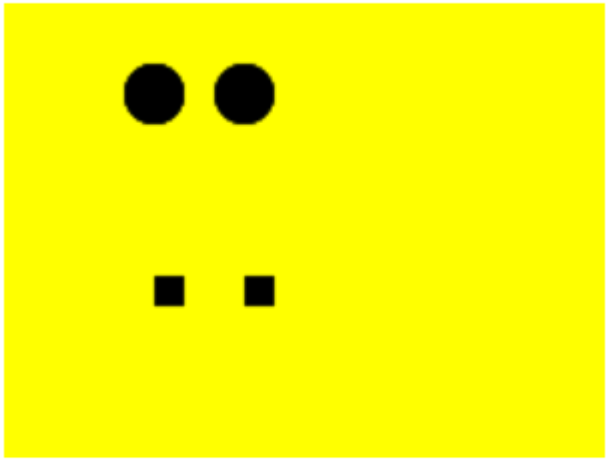
- Pane näide käima
- Katseta mitmesuguste suurustega pallidega
- Koosta tsükkel platsile pallide lisamiseks
- Lisa platsile parameetrid loodava platsi suuruse kohta pikslites
- Lisa platsile äärejoon ja määra selle kaugus servast
- Lisa platsile käsklus soovitud arvu pallide tekitamiseks juhuslikesse kohtadesse äärejoone sisse

Mitmesugused kujundid

Siin näites lisandub Platsile Palli kõrvale kujundiks Ruut. Et Javaskript massiividesse andmete lisamisel piiranguid ei sea, siis saab siin samasse kujundite massiivi panna läbisegi mõlemaid kujundeid. Ning et neil on mõlemal ühine meetod nimega joonista(), siis saab kujundite massiivi nõnda ka välja kuvada. Selleks kutsutakse Platsi joonista-funktsioonis järgemööda tsükli abil välja

kõikide kujundite joonista-funktsioonid.

```
<!doctype html>
<html>
  <head>
    <title>Kujundid</title>
    <script>
       function Ruut(x, y, a){
          this.x=x;
          this.y=y;
          this.a=a;
          this.joonista=function(g){
             g.fillRect(this.x, this.y, this.a, this.a);
          }
       }
       function Pall(x, y, r){
          this.x=x;
          this.y=y;
          this.r=r;
          this.joonista=function(g){
           g.beginPath();
           g.arc(this.x, this.y, this.r,
                          0, 2*Math.PI, true);
           g.fill();
          }
       }
       function Plats(kihiId){
          window[kihiId+"_joonis"]=this;
          this.alusta=function(){
             var sisu=
               "<canvas id='"+kihiId+"_tahvel' width='200' height='150' "+
                " style='background-color: yellow' ></canvas><br/>";
             document.getElementById(kihiId).innerHTML=sisu;
             this.kujundid=new Array();
             this.tahvel=document.getElementById(kihiId+"_tahvel");
          }
          this.lisaKujund=function(kujund){
            this.kujundid.push(kujund);
            this.joonista();
          }
          this.joonista=function(){
             var g=this.tahvel.getContext("2d");
             for(var i=0; i<this.kujundid.length; i++){
                this.kujundid[i].joonista(g);
             }
          }
          this.alusta();
       }

       function algus(){
         kiht1_joonis=new Plats("kiht1");
         kiht1_joonis.lisaKujund(new Pall(50, 30, 10));
         kiht1_joonis.lisaKujund(new Pall(80, 30, 10));
         kiht1_joonis.lisaKujund(new Ruut(50, 90, 10));
         kiht1_joonis.lisaKujund(new Ruut(80, 90, 10));
       }
    </script>
  </head>
  <body onload="algus();">
     <div id="kiht1"></div>
  </body>
</html>
```



Ülesandeid

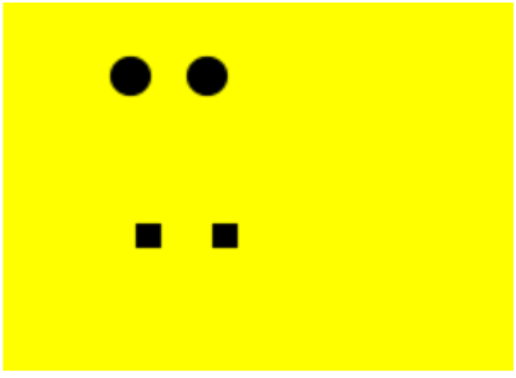
- Pane näide käima
- Paiguta ruute ja palle mitmesugustesse kohtadesse
- Lisa kujundina Puu, mis koosneb võrast ja tüvest. Ette saab anda võra keskkoha, võra raadiuse, tüve paksuse ning pikkuse

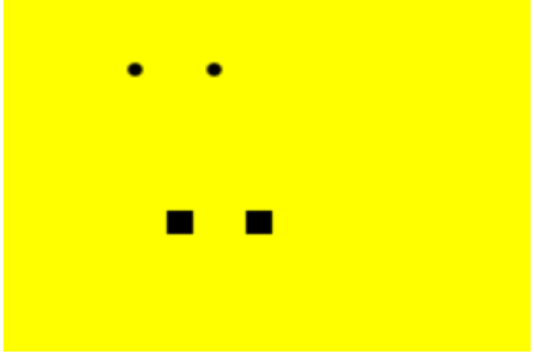
Liikuvad kujundid

Liikumise jaoks lisati mõlemale objektitüübile funktsioon liigu. Ruut suurendab selle peale oma x-koordinaadi väärtust ühe võrra, Pall aga kahendab raadiust 90% peale. Igatahes mõlemad muutuvad liikumise peale nähtavalt. Platsi liikumisfunktsioon käivitab järgemööda kõigi kujundite liikumisfunktsioonid ning edasi Platsi joonistusfunktsiooni, mis omakorda kõik kujundid välja joonistab.

```
<!doctype html>
<html>
  <head>
    <title>Kujundid</title>
    <script>
       function Ruut(x, y, a){
          this.x=x;
          this.y=y;
          this.a=a;
          this.joonista=function(g){
             g.fillRect(this.x, this.y, this.a, this.a);
          }
          this.liigu=function(){
            this.x++;
          }
       }
       function Pall(x, y, r){
          this.x=x;
          this.y=y;
          this.r=r;
          this.joonista=function(g){
           g.beginPath();
           g.arc(this.x, this.y, this.r, 0, 2*Math.PI, true);
           g.fill();
          }
          this.liigu=function(){
```

```
            this.r*=0.9;
          }
        }
        function Plats(kihiId){
          window[kihiId+"_joonis"]=this;
          this.alusta=function(){
            var sisu=
              "<canvas id='"+kihiId+"_tahvel' width='200' height='150' "+
               " style='background-color: yellow' ></canvas><br/>";
            document.getElementById(kihiId).innerHTML=sisu;
            this.kujundid=new Array();
            this.tahvel=document.getElementById(kihiId+"_tahvel");
            setInterval(kihiId+"_joonis.liigu()", 1000);
          }
          this.lisaKujund=function(kujund){
            this.kujundid.push(kujund);
            this.joonista();
          }
          this.joonista=function(){
            var g=this.tahvel.getContext("2d");
            g.clearRect(0, 0, 200, 150);
            for(var i=0; i<this.kujundid.length; i++){
              this.kujundid[i].joonista(g);
            }
          }
          this.liigu=function(){
            for(var i=0; i<this.kujundid.length; i++){
              this.kujundid[i].liigu();
            }
            this.joonista();
          }
          //Lisage kolmandaks kujundiks kukkuv seest tühi ring.
          this.alusta();
        }

        function algus(){
          kiht1_joonis=new Plats("kiht1");
          kiht1_joonis.lisaKujund(new Pall(50, 30, 10));
          kiht1_joonis.lisaKujund(new Pall(80, 30, 10));
          kiht1_joonis.lisaKujund(new Ruut(50, 90, 10));
          kiht1_joonis.lisaKujund(new Ruut(80, 90, 10));
        }
      </script>
    </head>
    <body onload="algus();">
       <div id="kiht1"></div>
    </body>
</html>
```





Ülesandeid

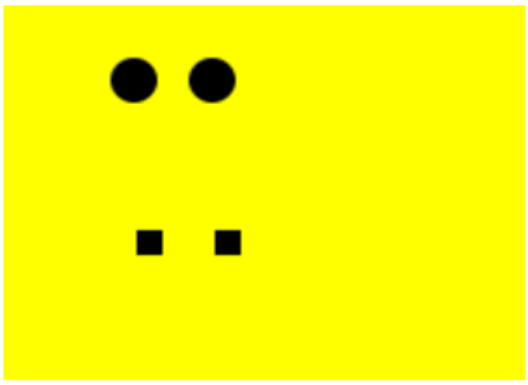
- Pane näide käima
- Pane ruut liikuma vasakule
- Lisa kujundina alla kukkuv tühi ring

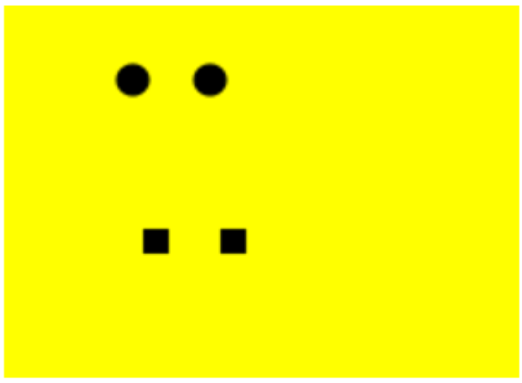
Hiirele reageeerivad liikuvad kujundid

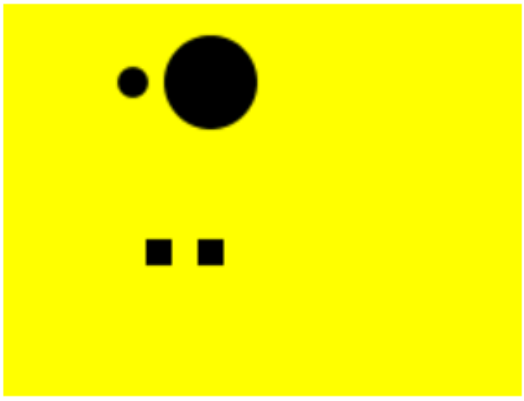
Kui kujundid jälgivad ühist käskude struktuuri, siis saab sellise "hulgikaubanduse" abil neile mitmesuguseid oskusi külge pookida. Siin näites kipuvad ruudud aegamisi paremale ekraanilt välja nihkuma ning ringid väga väikeseks minema. Näitena lisati aga oskused, kus saab hiire abil ruudu jälle lähemale kutsuda ning ringi suuremaks tagasi muuta. Ruut kontrollib, et kui hiire asukoha x-koordinaat on ruudust vasakul, siis leitakse x-i suunaline kaugus hiirest ning peegeldatakse ruut sama kaugele vasakule. Palli puhul aga antakse hiirega palli tabamise puhul lihtsalt taas raadiuseks 20 pikslit. Et hiire jälgimine töötaks, selleks taas püüab plats kõigepealt ise hiiresündmuse kinni.

```
<!doctype html>
<html>
  <head>
    <title>Kujundid</title>
    <script>
       function Ruut(x, y, a){
          this.x=x;
          this.y=y;
          this.a=a;
          this.joonista=function(g){
             g.fillRect(this.x, this.y, this.a, this.a);
          }
          this.liigu=function(){
            this.x++;
          }
          this.hiirAlla=function(hx, hy){
             if(hx<this.x){
               var vahe=this.x-hx;
               this.x=hx-vahe;
             }
          }
       }
       function Pall(x, y, r){
          this.x=x;
          this.y=y;
          this.r=r;
          this.joonista=function(g){
           g.beginPath();
           g.arc(this.x, this.y, this.r,
                         0, 2*Math.PI, true);
           g.fill();
          }
          this.liigu=function(){
            this.r*=0.9;
          }
          this.hiirAlla=function(hx, hy){
             var dx=hx-this.x;
             var dy=hy-this.y;
             var kaugus=Math.sqrt(dx*dx+dy*dy);
             if(kaugus<this.r){this.r=20;}
          }
```

```
      }
      function Plats(kihiId){
         window[kihiId+"_joonis"]=this;
         this.alusta=function(){
            var sisu=
              "<canvas id='"+kihiId+"_tahvel' width='200' height='150' "+
               " style='background-color: yellow' "+
               " onmousedown='"+kihiId+
                           "_joonis.hiirAlla(event)' ></canvas><br/>";
            document.getElementById(kihiId).innerHTML=sisu;
            this.kujundid=new Array();
            this.tahvel=document.getElementById(kihiId+"_tahvel");
            setInterval(kihiId+"_joonis.liigu()", 1000);
         }
         this.lisaKujund=function(kujund){
           this.kujundid.push(kujund);
           this.joonista();
         }
         this.joonista=function(){
            var g=this.tahvel.getContext("2d");
            g.clearRect(0, 0, 200, 150);
            for(var i=0; i<this.kujundid.length; i++){
               this.kujundid[i].joonista(g);
            }
         }
         this.liigu=function(){
            for(var i=0; i<this.kujundid.length; i++){
               this.kujundid[i].liigu();
            }
            this.joonista();
         }
         this.hiirAlla=function(e){
            var tahvlikoht=this.tahvel.getBoundingClientRect();
            var hx=e.clientX-tahvlikoht.left;
            var hy=e.clientY-tahvlikoht.top;
            for(var i=0; i<this.kujundid.length; i++){
               this.kujundid[i].hiirAlla(hx, hy);
            }
         }
         this.alusta();
      }

      function algus(){
        kiht1_joonis=new Plats("kiht1");
        kiht1_joonis.lisaKujund(new Pall(50, 30, 10));
        kiht1_joonis.lisaKujund(new Pall(80, 30, 10));
        kiht1_joonis.lisaKujund(new Ruut(50, 90, 10));
        kiht1_joonis.lisaKujund(new Ruut(80, 90, 10));
      }
    </script>
  </head>
  <body onload="algus();">
     <div id="kiht1"></div>
  </body>
</html>
```







Ülesandeid

- Pane näide käima
- Lisa kujundiks alla kukkuv seest tühi ring. Ringi tabamisel hakkab ta taas tahvli ülaservast kukkuma
- Lisa kujunditele funktsioon kasPihtas(hiireX, hiireY), mis vastab true või false vastavalt sellele, kas etteantud hiire koordinaadid asuvad kujundi seesl. Tee Platsi hiirAlla ümber nõnda, et iga kujundi juures kontrollitakse kõigepealt, et kas hiirega saadi kujundile pihta. Edasi alles hiirega tabamuse puhul kutsutakse välja vastava kujundi kasPihtas. See võimaldab kujundi hiirAlla funktsiooni juurest tabamuse kontrolli välja võtta.

Objektitüübi laiendamine, prototüüp

Küllalt palju saab oma koodi korrastada, kui suhteliselt iseseisvalt toimivad üksused objektitüüpideks ja nende juurde kuuluvateks objektideks kokku koondada. Nii nagu tavalise järjest kirjutatud koodi kokku grupeerimisel funktsioonidesse on võimalik saada mõistlik ülevaade vähemasti kümme korda suuremast lahendusest, nii funktsioonide ja andmete koondamisel objektitüüpidesse ja objektidesse annab see omakorda vähemalt sama suure võidu programmide keerukusega hakkama saamisel. Kuni loodud uusi tüüpe on vaid mõni ning nendest loodud objektid igaüks suhteliselt erisuguste omaduste ja kasutuskohtadega, siis võivad tüübid ise rahumeeli üksteisest erinevad ja sõltumatud olla. Kui aga hakkab tekkima kohti, kus objektid peaksid käituma mõne omaduse suhtes sarnaselt, mõne omas mitte, siis võib koodi ülesehitust otstarbekamaks muuta aidata objektitüübi laiendamine.

Käskluse lisamine

Järgnevas näites luuakse kõigepealt vektori tüüp. Isendimuutujateks x ja y ning meetodiks pikkus. Hiljem lisatakse Vektorile prototüübi kaudu käsklus `tekstina()`, mis siis selle vektori andmed viisakal kujul tekstina tagastab. Vastuskihil näeb funktsiooni töö tulemust.

```
<!doctype html>
<html>
  <head>
    <title>Laiendamine</title>
    <script>
       function Vektor(x, y){
          this.x=x;
          this.y=y;
          this.pikkus=function(){
             return Math.sqrt(this.x*this.x+this.y*this.y);
          }
       }

       Vektor.prototype.tekstina=function(){
         return "("+this.x+", "+this.y+")";
       }

       function leheAlgus(){
         var autokiirus=new Vektor(3, 4);
         document.getElementById("vastus").innerHTML=
          "Tekstina: "+autokiirus.tekstina()+
          " kogukiirus "+autokiirus.pikkus();
       }
    </script>
  </head>
  <body onload="leheAlgus();">
    <div id="vastus"></div>
  </body>
</html>
```

Tekstina: (3, 4) kogukiirus 5

Massiivi viimane element

Tavapärasem on käskluse lisamine juba võõrastele tüüpidele, mida ise muuta ei saa. Näitena

lisatakse viimase elemendi küsimise käsklus Javaskripti süsteemsele klassile Array. Kui muidu on võimalik massiivist eesnimed küsida elementide arvu käsuga eesnimed.length ning viimast eesnime kujul eesnimed[eesnimed.length-1] (sest elementide lugemine algab nullist), siis funktsioonis objekti enese poole pöördumiseks on sõna this. Ehk siis viimase elemendi väärtuse annab this[this.length-1].

```
<!doctype html>
<html>
  <head>
    <title>Laiendamine</title>
    <script>
       Array.prototype.viimane=function(){
          return this[this.length-1];
       }
       var eesnimed=new Array("Juku", "Kati", "Mati");
       function leheAlgus(){
         document.getElementById("vastus").innerHTML=eesnimed.viimane();
       }
    </script>
  </head>
  <body onload="leheAlgus();">
    <div id="vastus"></div>
  </body>
</html>
```

Väljund:

Mati

Ülesandeid

- Lisa Vektorile prototüübi kaudu käsklus korruta(arv). Selle tulemusena korrutatakse vektori mõlemad koordinaadid (this.x ja this.y) vastava arvuga. Midagi ei tagastata return-käsuga. Katseta, andes vektorile algväärtused, paludes neid kahega korrutada ning siis küsides tekstina, et mis väärtused vektori sees nüüd on.
- Lisa käsklus korrutaUueks(arv). Korrutusarvutus käib samuti kummagi koordinaadi kohta. Nüüd aga ei muudeta väljakutsuva vektori andmeid, vaid luuakse käsu töö käigus uus vektor oma koordinaatidega - return new Vektor(this.x*arv, this.y*arv); Katseta.
- Uuri Javaskripti standardtüüpi String. Lisa sellele prototüübi abil teatamaks, mitmest sõnast koosneb selles stringis olev lause. Vihje: käsklus split() jagab lause sõnade massiiviks ning siis saab massiivilt pikkust küsida.

Asukoha põhjal ring

Järgnevas näites pannakse Ringi prototüübiks Asukoha eksemplar. Ehk siis siin näites tehakse Asukohast muutuja a, mille sees koordinaadid 3 ja 5. Ning käsuga

```
var a=new Asukoht(3, 5);
Ring.prototype=a;
```

määratakse, et kõik loodavad ringid saavad oma aluseks Asukoha eksemplari, millest siis ringi

loomisel koopia tehakse. Tulemusena saavad kõik ringid kasutada konkreetse asukoha koordinaate ning samuti töötab käsklus tekstina(), mis koordinaadid viisakasti sulgude vahel välja kuvab.

```
<!doctype html>
<html>
  <head>
    <title>Laiendamine</title>
    <script>
       function Asukoht(x, y){
          this.x=x;
          this.y=y;
          this.tekstina=function(){
            return "("+this.x+", "+this.y+")";
          }
       }
       var a=new Asukoht(3, 5);
       function Ring(){
          this.raadius=10;
          this.pindala=function(){
             return 3.14*this.raadius*this.raadius;
          }
       }
       Ring.prototype=a;
       var r1=new Ring();
       function leheAlgus(){
         document.getElementById("vastus").innerHTML=
           "Ringi asukoht: "+r1.tekstina()+", pindala "+r1.pindala();
       }
    </script>
  </head>
  <body onload="leheAlgus();">
    <div id="vastus"></div>
  </body>
</html>
```

Ringi asukoht: (3, 5), pindala 314

Ülesandeid

- Pane näide tööle.
- Lisa Asukohale käsklus paremale(), mis suurendab x-i väärtust ühe võrra.
- Veendu, et Ringi on võimalik ka selle käskluse abil paremale nihutada, trüki ringi asukoht enne ja pärast nihutamist.

Ülemklassi väljakutse

Eelmises näites oli ringi aluseks alati asukoht koordinaatidega 3 ja 5. Sinna saab küll käsud panna x-i ja y-i asukoha muutmiseks, aga see mõnevõrra tüütu. Hea, kui saab kohe määrata ringi sinna kus ta mõeldud on. Üheks mooduseks on ringi loomise käsu seest välja kutsuda tema aluseks oleva asukoha loomise käsk. Ehk siis luuakse kõigepealt algandmetega asukoht ja määratakse see Ringi protüübiks. Edasi Ringi loomisel antakse x ja y edasi Asukoha konstruktorile, mis hoolitseb uute koordinaatide salvestamise eest.

```
       var a=new Asukoht(0, 0);
       Ring.prototype=a;
       function Ring(x, y, r){
          Asukoht.call(this, x, y);
```

Näide tervikuna:

```
<!doctype html>
<html>
  <head>
    <title>Laiendamine</title>
    <script>
       function Asukoht(x, y){
          this.x=x;
          this.y=y;
          this.tekstina=function(){
            return "("+this.x+", "+this.y+")";
          }
       }
       var a=new Asukoht(0, 0);
       function Ring(x, y, r){
          Asukoht.call(this, x, y);
          this.raadius=r;
          this.pindala=function(){
             return 3.14*this.raadius*this.raadius;
          }
       }
       Ring.prototype=a;
       var r1=new Ring(2, 4, 7);
       function leheAlgus(){
         document.getElementById("vastus").innerHTML=
           "Ringi asukoht: "+r1.tekstina()+", pindala "+r1.pindala();
       }
    </script>
  </head>
  <body onload="leheAlgus();">
    <div id="vastus"></div>
  </body>
</html>
```

Ringi asukoht: (2, 4), pindala 153.86

Ülesandeid

- Pane näide tööle
- Lisage ringile käsklus kasPuutub(teineRing), mis väljastab, et kas üks ring puutub teisega kokku. Katseta toimimist.
- Koosta massiiv viie ringiga. Väljasta, millised ringid puutuvad esimese ringiga kokku.

Käskluse asendamine

Objektidega majandamine võimaldab uue objekti aluseks võtta ka objekti, mille mõned omadused sobivad, teised aga mitte. Siin näites luuakse kõigepealt AlusRuut, kel olemas asukoha

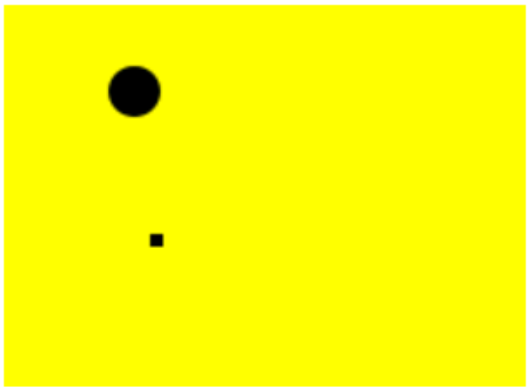
koordinaadid ning kiirusesamm mõlema koordinaadi suunas. Liikumisfunktsiooni käivitusega liigutakse ühe sammu jagu edasi. Joonistuskäsu tulemusena kuvatakse etteantud graafilise konteksti sihtkohale ruut, mille asukohaks alusruudu koordinaadid ning külje pikkuseks viis ühikut.

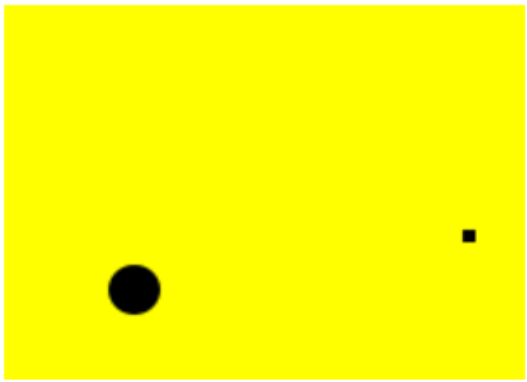
Kui nüüd soovida ringi ehk palli ekraanile joonistada ja seal liigutada, siis alusruudust sobivad kasutada asukoha koordinaadid ja liikumisearvutus, aga joonistamise tulemusena peaks midagi kandilisest ruudust ümaramat ekraanile ilmuma. Nii siin teha saabki. Kukkuva palli juures luuakse joonistuskäsklus uuesti koos ümmarguse ringiga, see asendab ruudu juurest kaasa tulnud kandilise joonistusfunktsiooni.

Platsi peale pannakse AlusRuut ja Pall mõlemad. Neil olemas nüüd nii joonistus- kui liikumisoskus. Pall määrab oma liikumissammuks iga kaadriga kaks ühikut allapoole, alusruudul antakse loomisel paremale liikumise andmed.

```
<!doctype html>
<html>
  <head>
    <title>Kujundid</title>
    <script>
       function AlusRuut(x, y, kiirusx, kiirusy){
          this.x=x;
          this.y=y;
          this.kx=kiirusx;
          this.ky=kiirusy;
          this.joonista=function(g){
             g.fillRect(this.x, this.y, 5, 5);
          }
          this.liigu=function(){
            this.x+=this.kx;
            this.y+=this.ky;
          }
       }
       KukkuvPall.prototype=new AlusRuut(0, 0, 0, 0);
       function KukkuvPall(x, y, r){
          AlusRuut.call(this, x, y, 0, 2)
          this.r=r;
          this.joonista=function(g){
           g.beginPath();
           g.arc(this.x, this.y, this.r,
                          0, 2*Math.PI, true);
           g.fill();
          }
       }
       function Plats(kihiId){
          window[kihiId+"_joonis"]=this;
          this.alusta=function(){
             var sisu=
               "<canvas id='"+kihiId+"_tahvel' width='200' height='150' "+
                " style='background-color: yellow' ></canvas><br/>";
             document.getElementById(kihiId).innerHTML=sisu;
             this.kujundid=new Array();
             this.tahvel=document.getElementById(kihiId+"_tahvel");
             setInterval(kihiId+"_joonis.liigu()", 1000);
          }
          this.lisaKujund=function(kujund){
            this.kujundid.push(kujund);
            this.joonista();
          }
          this.joonista=function(){
```

```
            var g=this.tahvel.getContext("2d");
            g.clearRect(0, 0, 200, 150);
            for(var i=0; i<this.kujundid.length; i++){
               this.kujundid[i].joonista(g);
            }
         }
         this.liigu=function(){
            for(var i=0; i<this.kujundid.length; i++){
               this.kujundid[i].liigu();
            }
            this.joonista();
         }
         this.alusta();
      }

      function algus(){
        kiht1_joonis=new Plats("kiht1");
        kiht1_joonis.lisaKujund(new KukkuvPall(50, 30, 10));
        kiht1_joonis.lisaKujund(new AlusRuut(50, 90, 3, 0));
      }
    </script>
  </head>
  <body onload="algus();">
     <div id="kiht1"></div>
  </body>
</html>
```





Ülesandeid

- Pane näide tööle
- Lisa mitu ruutu erisuguste kiirustega
- Kukkuva palli juures määra kukkumiskiirus juhuslikult vahemikus 1-5 ühikut kaadri kohta
- Lisa katsetamiseks mitu palli.

Klassi prototüübid andmehalduses

Järgnevalt pikem näide, kus eri tegevused eri objektide vahel jaotatud ning võimaldavad kummalgi tüübil selle jaoks mõeldud tegevusele keskenduda ning sellega koodi paremini loetavana hoida. Alustuseks luuakse AndmeHalduse tüüp, kus talletatakse teksti ja arvu paarid ning kust saab hiljem arvud eraldi funktsiooniga välja küsida.

```
<!doctype html>
<html>
  <head>
    <title>Kujundid</title>
    <script>
       function AndmeHaldus(){
          this.andmed=new Array();
          this.lisaKirje=function(tekst, arv){
             this.andmed.push({"t": tekst, "a": arv});
          }
          this.kysiArvud=function(){
             var arvud=new Array();
             for(var i=0; i<this.andmed.length; i++){
                arvud.push(this.andmed[i].a);
             }
             return arvud;
          }
       }

       function algus(){
          var ah=new AndmeHaldus();
          ah.lisaKirje("Juku", 175);
          ah.lisaKirje("Kati", 165);
          document.getElementById("kiht1").innerHTML=ah.kysiArvud();
       }
    </script>
  </head>
  <body onload="algus();">
     <div id="kiht1"></div>
  </body>
</html>
```

Näite väljundiks siis kaks arvu:

175,165

Tabelina näitav laiendus

Andmete hoidjale saab vajadust mööda mitmesuguseid andmete kuvajaid peale ehitada. AndmeTabeli objektis jäetakse meelde teksti ja arvu komplektid. Eraldi käsuga saab välja küsida ka ainult arvud. Olemasolevale objektile saab peale ehitada teise.

```
AndmeTabel.prototype=new AndmeHaldus();
```

ütleb, et igale uuele AndmeTabelile võetakse aluseks AndmeHalduse eksemplar. Ehk siis AndmeTabeli objekt oskab samuti enese sisse teksti ja arvu komplekte koguda ja sealt arve eraldada. Lisaks on tal juures oskus tulemuste kuvamiseks tabelina.

```
<!doctype html>
<html>
  <head>
    <title>Kujundid</title>
    <script>
       function AndmeHaldus(){
          this.andmed=new Array();
          this.lisaKirje=function(tekst, arv){
             this.andmed.push({"t": tekst, "a": arv});
          }
```

```
            this.kysiArvud=function(){
               var arvud=new Array();
               for(var i=0; i<this.andmed.length; i++){
                  arvud.push(this.andmed[i].a);
               }
               return arvud;
            }
         }



         AndmeTabel.prototype=new AndmeHaldus();
         function AndmeTabel(){
            this.HTMLTabelina=function(){
               var t="<table>";
               for(var i=0; i<this.andmed.length; i++){
                 t+="<tr><td>"+this.andmed[i].t+"</td><td>"+
                                this.andmed[i].a+"</td></tr>\n";
               }
               t+="</table>";
               return t;
            }
         }

         function algus(){
            var at=new AndmeTabel();
            at.lisaKirje("Juku", 175);
            at.lisaKirje("Kati", 165);
            document.getElementById("kiht1").innerHTML=at.HTMLTabelina();
         }
      </script>
   </head>
   <body onload="algus();">
      <div id="kiht1"></div>
   </body>
</html>
```

Tabelit võib lehel vaadata

Juku 175

Kati 165

Ülesandeid

- Pane näide tööle
- Lisa AndmeTabeli juurde käsk arvude loetelu näitamiseks (unordered list)

Joonisena näitav laiendus

Samale AndmeHalduse tüübile võimalik peale ehitada teine lahendus, sedakorda joonistusoskusega. Siin kasutatakse arvudena välja küsimise oskust. Joonistustahvel lisatakse funktsioonis etteantud nimega kihile sedakorda Javaskripti DOM-funktsioonide abil.

```
            var tahvel=document.createElement("canvas");
```

```
            tahvel.setAttribute("width",  "300");
            tahvel.setAttribute("height", "200");
            tahvel.style.backgroundColor="yellow";
            document.getElementById(kihinimi).appendChild(tahvel);
```

Üheks mooduseks elementide veebilehele lisamisel on lehe elementide omadus innerHTML, kuhu kirjutatud väärtused teisendatakse brauseri mälus seal olevateks objektideks. Samas on aga võimalik ka objektipuud mööda liikuda ja seal andmeid elemente luua ja ümber paigutada, mis mõnel juhul on mugavam või kiirem. Sõltumata loomismoodusest saab tahvlit ikka ühtemoodi kasutada.

```
<!doctype html>
<html>
  <head>
    <title>Kujundid</title>
    <script>
       function AndmeHaldus(){
          this.andmed=new Array();
          this.lisaKirje=function(tekst, arv){
             this.andmed.push({"t": tekst, "a": arv});
          }
          this.kysiArvud=function(){
             var arvud=new Array();
             for(var i=0; i<this.andmed.length; i++){
                arvud.push(this.andmed[i].a);
             }
             return arvud;
          }
       }

       AndmeJoonis.prototype=new AndmeHaldus();
       function AndmeJoonis(){
          this.arvudJoonisena=function(kihinimi){
             var tahvel=document.createElement("canvas");
             tahvel.setAttribute("width",  "300");
             tahvel.setAttribute("height", "200");
             tahvel.style.backgroundColor="yellow";
             document.getElementById(kihinimi).appendChild(tahvel);
             var g=tahvel.getContext("2d");
             var arvud=this.kysiArvud();
             for(var i=0; i<arvud.length; i++){
                g.fillRect(10, 10+i*20, arvud[i], 10);
             }
          }
       }


       function algus(){
          var aj=new AndmeJoonis();
          aj.lisaKirje("Juku", 175);
          aj.lisaKirje("Kati", 165);
          aj.arvudJoonisena("kiht1");
//          document.getElementById("kiht1").innerHTML=at.HTMLTabelina();
       }
    </script>
  </head>
  <body onload="algus();">
     <div id="kiht1"></div>
  </body>
</html>
```

Tulemusena võibki imetleda andmete graafilist väljundit.

Ülesandeid

- Pane näited tööle
- Kuva AndmeJoonise tulpade juurde ka nimetused ja arvud

Ringjoonega seotud arvutused

Ring kipub vähegi graafilisemate lahenduste juures ikka ette tulema. Olgu siis tegemist elementide paigutamise, liikumise või vaatesuundade arvutamisega.

Ringjoone parameetriline võrrand

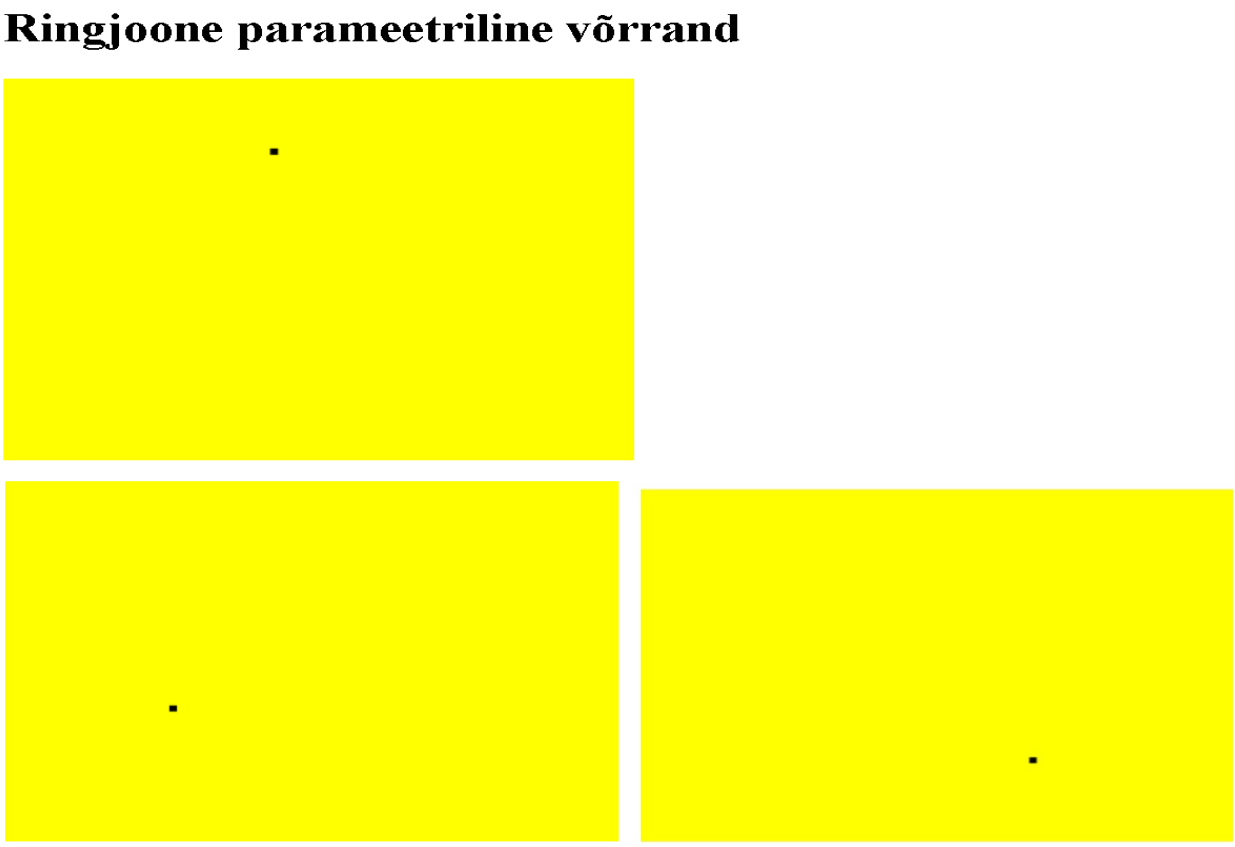
Lisaks mehhaanikas asukoha leidmiseks tarvilikule $x=x_0+v_0t+at^2/2$ ning kaugusearvutuse $\sqrt{x^2+y^2}$ arvutusele on kolmandaks tähtsaks ettetulevaks arvutustehteks ringjoone parameetriline võrrand: $x=x_0+r*\cos(a)$, $y=y_0+r*\sin(a)$. Selle abil saab lihtsamal juhul panna näiteks täpi mööda ekraani ringikujuliselt liikuma.

```
<!doctype html>
<html>
  <head>
     <title>Ringi arvutused</title>
     <script>
       var keskx=200;
       var kesky=150;
       var ringiraadius=100;
       var nurk=0;
       var nurgavahe=0.05; //radiaani
       function liigu(){
          var g=document.getElementById("t1").getContext("2d");
          g.clearRect(0, 0, 400, 300);
          g.fillRect(keskx+ringiraadius*Math.cos(nurk),
                     kesky-ringiraadius*Math.sin(nurk), 5, 5);
```

```
          nurk+=nurgavahe;
        }
      </script>
  </head>
  <body onload="setInterval('liigu()', 50);">
     <h1>Ringjoone parameetriline võrrand</h1>
     <canvas id="t1" width="400" height="300"
       style="background-color: yellow" ></canvas>
  </body>
</html>
```

Ülesandeid

- Käivita näide
- Muuda liikumise keskkohta ja raadiust
- Pane ekraanil korraga ringikujuliselt liikuma kaks täppi



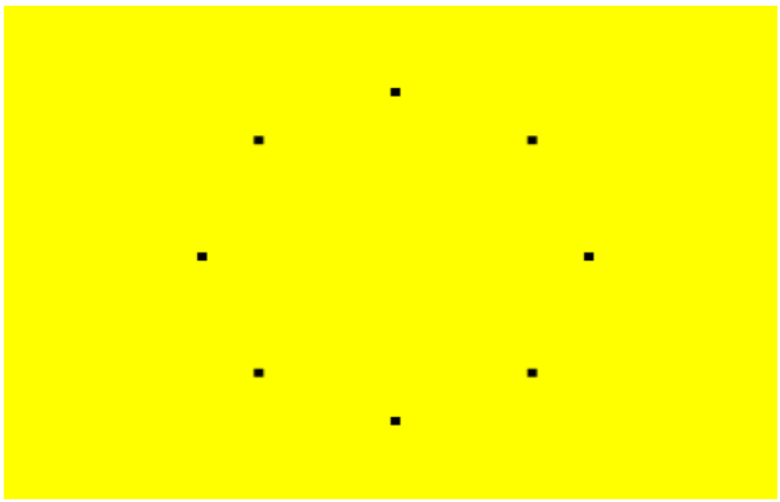
Ruudud ringjoonel

Kui ühekorraga ruudud valmis joonistada ja mitte andmeid muuta ja ekraani tühjendada, siis saab soovitud kujundid ilusti ringi peale paigutada.

```
<!doctype html>
<html>
  <head>
     <title>Ringi arvutused</title>
     <script>
       var keskx=200;
```

```
      var kesky=150;
      var ringiraadius=100;
      var nurk=0;
      var punktidearv=8;
      var nurgavahe=2*Math.PI/punktidearv;
      function joonista(){
         var g=document.getElementById("t1").getContext("2d");
         for(var i=0; i<punktidearv; i++){
           g.fillRect(keskx+ringiraadius*Math.cos(nurk),
                    kesky-ringiraadius*Math.sin(nurk), 5, 5);
           nurk+=nurgavahe;
         }
      }
    </script>
  </head>
  <body onload="joonista();">
    <h1>Ruudud ringjoonel</h1>
    <canvas id="t1" width="400" height="300"
      style="background-color: yellow" ></canvas>
  </body>
</html>
```

Ruudud ringjoonel



Ülesandeid

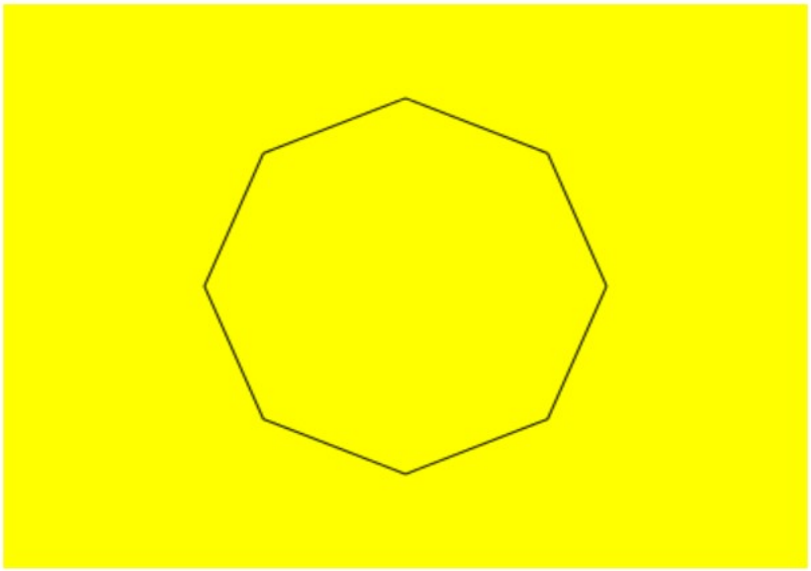
- Käivita näide
- Koosta kella numbrilaud - ringikujuliselt arvud 1-st 12ni.
- Pane iga numbri juurde ka keskkoha poole suunduv joon.

- Eelpool olevat ringjoonel liikumise näidet arvestades püüa tööle panna osutitega kell.

Hulknurk

Järgnevalt veidi pikem näide, kus üheaegselt tegeldakse mitme punkti keeramisega. Korrapärase hulknurga puhul tuleb teada nurkade arvu ning nende kaugust keskpunktist. Edasi saab selle põhjal juba jooned tõmmata. Esimesse punkti tasub minna moveTo-ga, edasi igasse järgmisse nurka saab lineTo abil joone tõmmata kui tahta ühtlaselt ühendatud hulknurka saada. Ning viimasest punktist sammu võrra edasi minnes jõutakse jälle algusesse tagasi.

```
<!doctype html>
<html>
  <head>
     <title>Ringi arvutused</title>
     <script>
       var keskx=200;
       var kesky=150;
       var ringiraadius=100;
       var nurk=0;
       var punktidearv=8;
       var nurgavahe=2*Math.PI/punktidearv;
       function joonista(){
          var g=document.getElementById("t1").getContext("2d");
          g.beginPath();
          g.moveTo(keskx+ringiraadius*Math.cos(nurk),
                   kesky-ringiraadius*Math.sin(nurk));
          nurk+=nurgavahe;
          for(var i=1; i<=punktidearv; i++){
            g.lineTo(keskx+ringiraadius*Math.cos(nurk),
                     kesky-ringiraadius*Math.sin(nurk));
            nurk+=nurgavahe;
          }
          g.stroke();
       }
     </script>
  </head>
  <body onload="joonista();">
     <h1>Ruudud ringjoonel</h1>
     <canvas id="t1" width="400" height="300"
       style="background-color: yellow" ></canvas>
  </body>
</html>
```



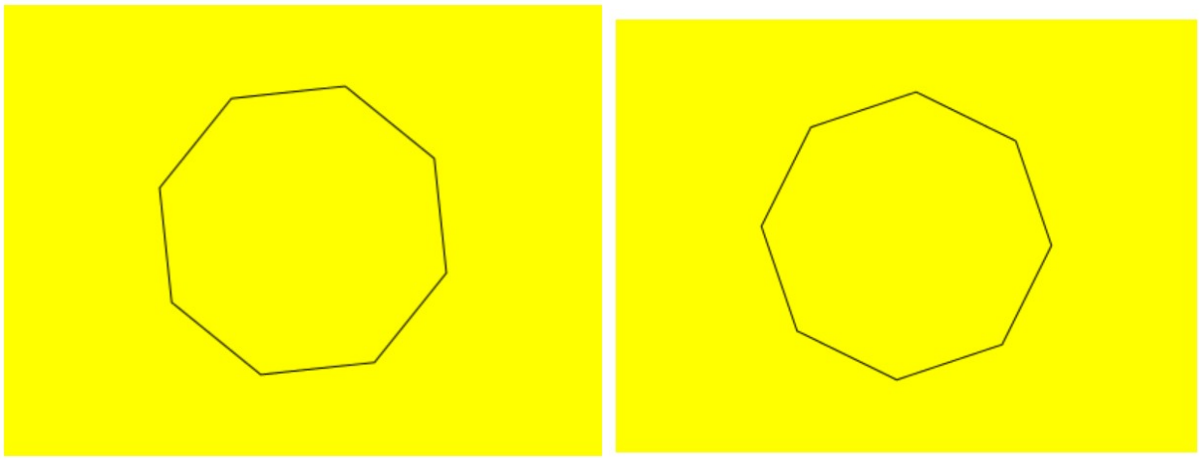
Ülesandeid

- Käivita näide
- Muuda nurkade arvu ja raadiust
- Loo lehele slaiderid raadiuse ja nurkade arvu sujuvaks muutmiseks

Keerlev hulknurk

Hulknurga ühtlaselt keerlema panekuks tuleb esimese punkti algusnurka sujuvalt muutma hakata. Kui teised esimese järgi arvutada, siis hakkabki nõnda kogu hulknurk keerlema.

```
<!doctype html>
<html>
  <head>
     <title>Ringi arvutused</title>
     <script>
       var keskx=200;
       var kesky=150;
       var ringiraadius=100;
       var algnurk=0;
       var keeramissamm=0.1;
       var punktidearv=8;
       var nurgavahe=2*Math.PI/punktidearv;

       function liigu(){
         algnurk+=keeramissamm;
         joonista();
       }

       function joonista(){
          var nurk=algnurk;
          var g=document.getElementById("t1").getContext("2d");
          g.clearRect(0, 0, 400, 300);
          g.beginPath();
          g.moveTo(keskx+ringiraadius*Math.cos(nurk),
                   kesky-ringiraadius*Math.sin(nurk));
          nurk+=nurgavahe;
          for(var i=1; i<=punktidearv; i++){
            g.lineTo(keskx+ringiraadius*Math.cos(nurk),
                     kesky-ringiraadius*Math.sin(nurk));
            nurk+=nurgavahe;
          }
          g.stroke();
       }
     </script>
  </head>
  <body onload="setInterval('liigu()', 100);">
     <h1>Ruudud ringjoonel</h1>
     <canvas id="t1" width="400" height="300"
       style="background-color: yellow" ></canvas>
  </body>
</html>
```



Ülesandeid

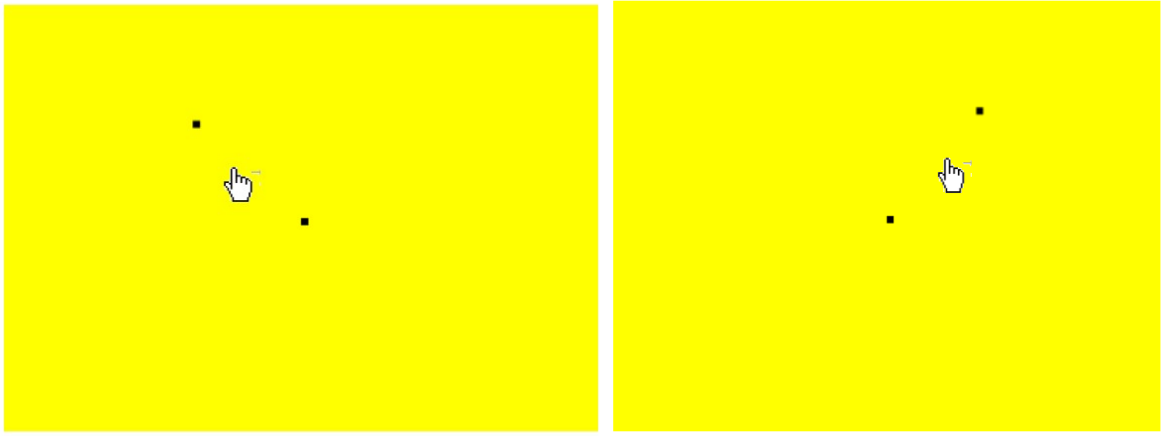
- Pane näide käima
- Võimalda hulknurga pöörlemiskiirust slaideri abil muuta.
- Võimalda ta ka teistpidi pöörlema panna

Nurga määramine

Ümmarguste asjade joonistamisel ning ringi ja kaart pidi liigutamisel on hea ringjoone parameetrilise võrrandi abil nurga, keskpunkti ja raadiuse järgi koordinaate leida. Hiirega rakendust juhtides aga tuleb mõnikord kasuks vastupidine - tuleb leida hiire asukohale vastav nurk mõne punkti suhtes, et saaks ekraanil kujundeid sobivalt pöörata ja liikuma panna. Õnneks on Javaskriptis selle tarbeks kohandatud arkustanensi funktsioon Math.atan2(x, y), mis annab etteantud koordinaatidega punkti paiknemise nurga nullpunkti suhtes. Nõnda on asimuudi leidmine lihtsaks tehtud, tuleb vaid rakendusele sobivad andmed ette sööta. Järgnevas näites näeb ekraani peal kahte punkti. Üks püsib paigal, teise paiknemise suunda selle keskpunkti suhtes saab aga hiirega määrata. Hiire asukoha ja keskpunkti kauguse põhjal leitakse keskpunktist hiire poole suunduv nurk. Sinnapoole püsiva raadiuse kaugusele joonistatakse ringjoone parameetrilise võrrandi abil täpp. Nii saabki hiire abil praegu ühte täppi, aga soovi korral ka tunduvalt keerukamat kujundit ümber keskpunkti pöörata.

```
<!doctype html>
<html>
  <head>
     <title>Nurga leidmine</title>
     <script>
       var keskx=200;
       var kesky=150;
       var ringiraadius=100;
       function hiirLiigub(e){
          var tahvlikoht=document.getElementById("t1").
               getBoundingClientRect();
          var g=document.getElementById("t1").getContext("2d");
          g.clearRect(0, 0, 400, 300);
          g.fillRect(keskx, kesky, 5, 5);
          var hx=e.clientX-tahvlikoht.left;
          var hy=e.clientY-tahvlikoht.top;
          //g.fillRect(hx, hy, 5, 5);
          var nurk=Math.atan2(hy-kesky, hx-keskx);
          g.fillRect(keskx+ringiraadius*Math.cos(nurk),
```

```
                    kesky+ringiraadius*Math.sin(nurk), 5, 5);

      }
    </script>
  </head>
  <body>
    <h1>Ruudud ringjoonel</h1>
    <canvas id="t1" width="400" height="300"
      style="background-color: yellow"
      onmousemove="hiirLiigub(event)"></canvas>
  </body>
</html>
```



Ülesandeid

- Pane näide käima
- Kujunda ekraanile nupp, mida on võimalik koos sellel oleva kriipsuga hiire abil pöörata
- Pane selliseid nuppe ekraanile kaks. Kummagi nupu piires saab just selle nupu pöördenurka sättida.

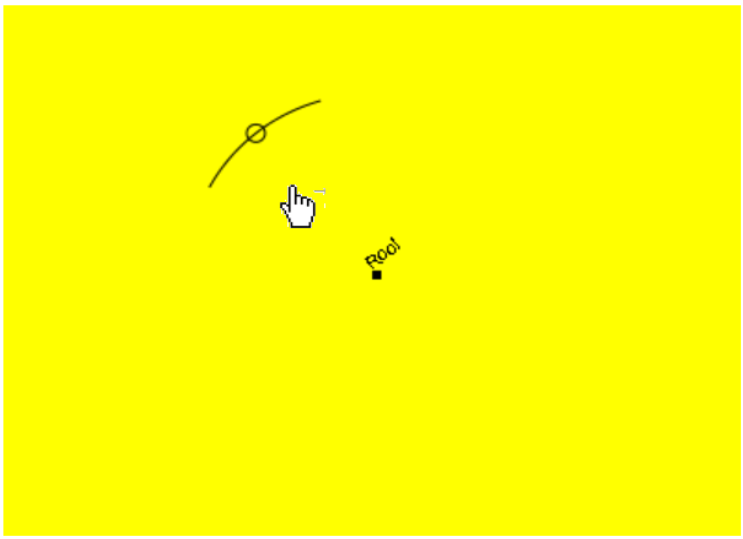
Rool

Täpi keeramise edasiarenduseks on lihtsa rooli keeramine. Hiire asukohast leitud nurga järgi joonistatakse kaar punktist mõlemale poole. Ringjoone parameetrilise võrrandi abil saab ümmarguse täpi kaare keskele soovitud suunda. Teksti pööramiseks rooli keskosas sobib aga reeperi /koordinaatteljestiku nihutus ja pööre. Salvestuskäsklusega talletatakse algne olek, et selle juurde saaks tagasi minna. Edasi translate nihutab nullpunkti rooli keskkoha juurde. Siis on võimlik tekst selle koha juures sobiva nurga alla keerata ja ekraanile joonistada. Ning lõpuks restore-käsklus taastab endise olukorra, et muud joonistuskäsud jälle arusaadavatesse kohtadesse läheksid.

```
g.save();
g.translate(keskx, kesky);
g.rotate(nurk+(Math.PI/2));
g.fillText("Rool", 0, 0);
g.restore();
```

Edasi juba töötav kood tervikuna.

```
<!doctype html>
<html>
  <head>
     <title>Nurga leidmine</title>
     <script>
       var keskx=200;
       var kesky=150;
       var ringiraadius=100;
       function hiirLiigub(e){
          var tahvlikoht=document.getElementById("t1").
               getBoundingClientRect();
          var g=document.getElementById("t1").getContext("2d");
          g.clearRect(0, 0, 400, 300);
          g.fillRect(keskx, kesky, 5, 5);
          var hx=e.clientX-tahvlikoht.left;
          var hy=e.clientY-tahvlikoht.top;
          var nurk=Math.atan2(hy-kesky, hx-keskx);
          g.beginPath();
          g.arc(keskx, kesky, ringiraadius, nurk-0.4, nurk+0.4, false);
          g.stroke();
          g.beginPath();
          g.arc(keskx+ringiraadius*Math.cos(nurk),
                     kesky+ringiraadius*Math.sin(nurk), 5, 0, 2*Math.PI, false);
          g.stroke();
          g.save();
          g.translate(keskx, kesky);
          g.rotate(nurk+(Math.PI/2));
          g.fillText("Rool", 0, 0);
          g.restore();
       }
     </script>
  </head>
  <body onload="setInterval('liigu()', 100);">
     <h1>Ruudud ringjoonel</h1>
     <canvas id="t1" width="400" height="300"
       style="background-color: yellow"
       onmousemove="hiirLiigub(event)"></canvas>
  </body>
</html>
```



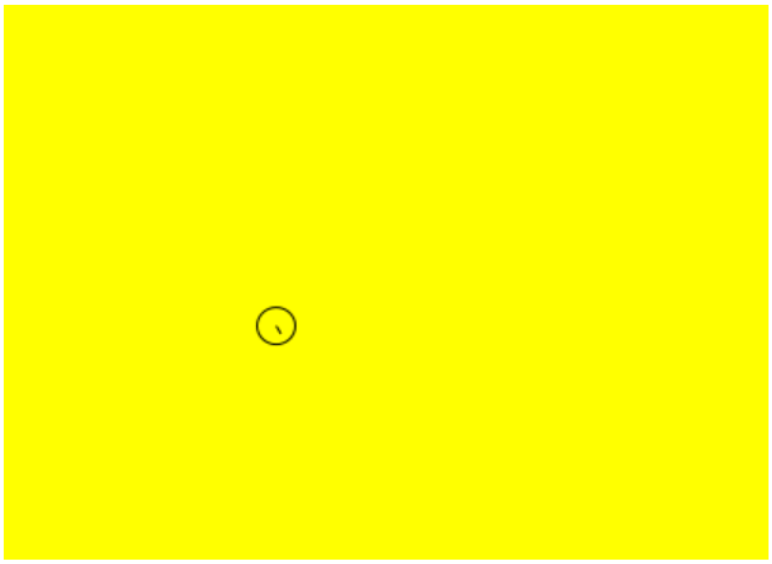
Ülesandeid

- Pane näide käima
- Asenda rooliratta suunda tähistav ringike sobiva nurga alla keeratud R-tähega

Liikur

Tavapäraselt kipume arvutigraafikas eraldi x- ja y-suunda arvestama ja meeles pidama. Samas nurga all liikumise ja pööramiste puhul on mõnigikord mugavam meeles pidada pigem asukohta ning liikumissuuna nurka. Edasised arvutused saab juba nende andmete põhjal teha. Praegusel juhul arvutatakse iga sammu puhul siinus ka koosiinus uuesti, mis üsnagi töömahukad arvutile. Koodi kiirust võimalik siitkaudu kasvatada, aga ka nii paistab liikume täiesti õnnestuma.

```
<!doctype html>
<html>
  <head>
     <title>Ringi arvutused</title>
     <script>

       function Liikur(x, y, kiirus, nurk){
         this.algus=function(){
          this.x=x;
          this.y=y;
          this.kiirus=kiirus;
          this.nurk=nurk;
         }
         this.liigu=function(){
           this.x+=this.kiirus*Math.cos(this.nurk);
           this.y+=this.kiirus*Math.sin(this.nurk);
         }
         this.muudaNurk=function(uusNurk){
            this.nurk=uusNurk;
         }
         this.joonista=function(g){
            g.beginPath();
            g.arc(this.x, this.y, 10, 0, 2*Math.PI, false);
            g.stroke();
            g.beginPath();
            g.moveTo(this.x,this.y);
            g.lineTo(this.x+5*this.kiirus*Math.cos(this.nurk),
                     this.y+5*this.kiirus*Math.sin(this.nurk));
            g.stroke();
         }
         this.algus();
       }

       var auto1=new Liikur(100, 100, 1, Math.PI/3);
       function liigu(){
          var g=document.getElementById("t1").getContext("2d");
          g.clearRect(0, 0, 400, 300);
          auto1.liigu();
          auto1.joonista(g);
       }
     </script>
  </head>
  <body onload="setInterval('liigu()', 50);">
     <h1>Ringjoone parameetriline võrrand</h1>
     <canvas id="t1" width="400" height="300"
       style="background-color: yellow" ></canvas>
  </body>
</html>
```



Ülesandeid

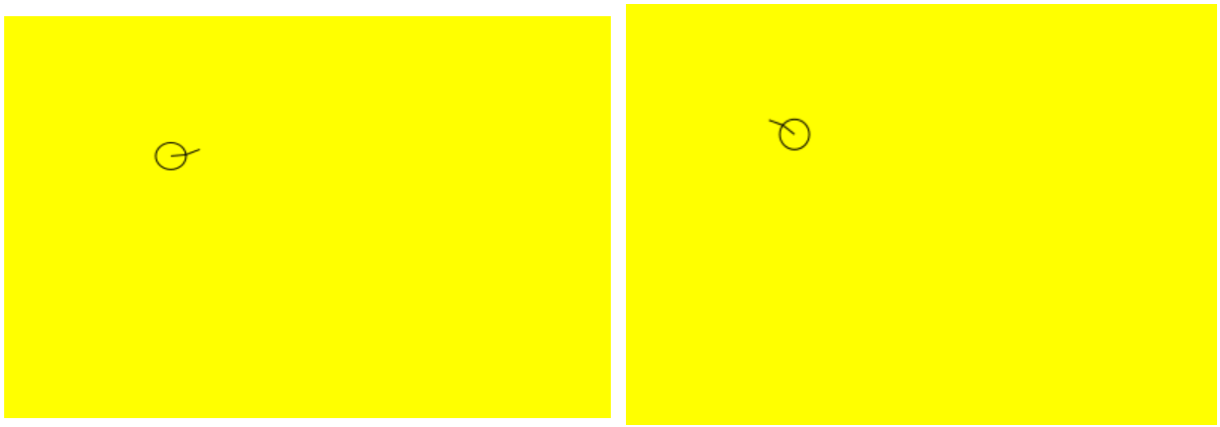
- Pane näide käima
- Lisa ekraanile nupud liikuri keeramiseks päri- ja vastupäeva
- Pane korraga liikuma mitu liikurit
- Joonista joone liikumissuunda näitavasse otsa väike ringike

Keerav liikur

Auto juhtmist jäljendades rooli pööramine ei tähenda mitte korraks sihi muutust, vaid pööratud rool paneb edasisõidu ajal auto pidevalt vastavas suunas pöörama. Sarnast tulemust püütakse ka siin saavutada. Ehk siis kui liikur sõitma pannakse, siis määratakse talle lisaks asukohale, kiirusele ja algnurgale ka nurgamuutus, et kui palju iga sammuga liikur uuesti pöörab. Selliselt tehtud liikur sõidab siis vähegi otsesihist kõrvale kallutatuna mitmesuguse suurusega ringe vastavalt etteantud nurgamuutusele.
Pööramissuuna ja -ulatuse paremaks jälgimiseks on sihtjoone otsas kümne ekraanipunkti suurune juhtlõik, mis selgema nägemise huvides kolmekordse võimendusega annab teada, et kui palju ja millises suunas pööratakse. Pööramisjoone arvutamiseks võetakse liikumisjoone ots ning sealt edasi siis arvutatakse vajaliku nurga ja pööramisjoone pikkuse järgi uued koordinaadid.

```
<!doctype html>
<html>
  <head>
     <title>Ringi arvutused</title>
     <script>

       function Liikur(x, y, kiirus, nurk, nurgamuutus){
         this.algus=function(){
          this.x=x;
          this.y=y;
          this.kiirus=kiirus;
          this.nurk=nurk;
          this.nurgamuutus=nurgamuutus;
         }
         this.liigu=function(){
           this.nurk+=this.nurgamuutus;
```

```
            this.x+=this.kiirus*Math.cos(this.nurk);
            this.y+=this.kiirus*Math.sin(this.nurk);
          }
          this.muudaNurk=function(uusNurk){
             this.nurk=uusNurk;
          }
          this.joonista=function(g){
             g.beginPath();
             g.arc(this.x, this.y, 10, 0, 2*Math.PI, false);
             g.stroke();
             g.beginPath();
             g.moveTo(this.x,this.y);
             //Joone ots
             var jotsx=this.x+10*this.kiirus*Math.cos(this.nurk);
             var jotsy=this.y+10*this.kiirus*Math.sin(this.nurk);
             g.lineTo(jotsx, jotsy);

             var rattapikkus=10;
             var rattanurk=this.nurk+3*this.nurgamuutus;
             var rattaotsx=jotsx+rattapikkus*Math.cos(rattanurk);
             var rattaotsy=jotsy+rattapikkus*Math.sin(rattanurk);
             g.lineTo(rattaotsx, rattaotsy)
             g.stroke();
          }
          this.algus();
        }

        var auto1=new Liikur(100, 100, 1, Math.PI/3, -0.1);
        function liigu(){
           var g=document.getElementById("t1").getContext("2d");
           g.clearRect(0, 0, 400, 300);
           auto1.liigu();
           auto1.joonista(g);
        }
      </script>
  </head>
  <body onload="setInterval('liigu()', 50);">
     <h1>Ringjoone parameetriline võrrand</h1>
     <canvas id="t1" width="400" height="300"
       style="background-color: yellow" ></canvas>
  </body>
</html>
```



Ülesandeid

- Pane näide tööle
- Lisa pööramisnurga suurendamiseks ja vähendamiseks nupud

Rool ja liikur

Autosõidu puhul saab pidevalt sõiduki pööramist muuta. Nii ka siin püütakse nüüd kaks eraldi loodud lahendust kokku ühendada. Rooli juurest saab küsida, et kui palju peaks keerama ning liikuri ülesandeks on siis need andmed vastu võtta ning nende põhjal sujuvalt mööda ekraani liikuda. Suuremalt jaolt saab Rooli ja Liikuri tüübid kokku kopeerida. Ning lihtsalt liikumise juures siis iga sammu puhul küsida rooli käest selle pööratuse nurk ning määrata see autole uueks rooli asendiks ehk nurgamuutuseks iga sammu juures.

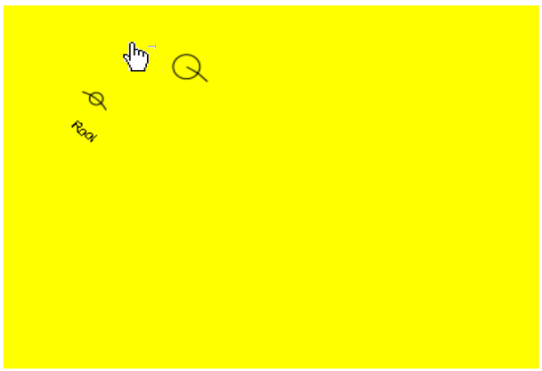
```
auto1.uusRooliAsend(rool1.kysiNurk()/10);
```

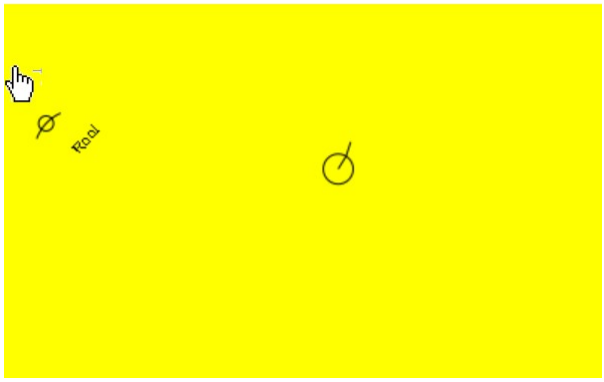
Katseliselt selgus, et suhteliselt mugav oli autot juhtida praeguste parameetrite järgi siis, kui keeramine sammu juures oli kümnendik rooli tegelikust pöördenurgast. Aga eks selliste suhete paika sättimine olegi rakenduse loomise juures mugavuse tekitamise ja katsetamise küsimus.

```
<!doctype html>
<html>
  <head>
     <title>Ringi arvutused</title>
     <script>

       function Liikur(x, y, kiirus, nurk, nurgamuutus){
         this.algus=function(){
          this.x=x;
          this.y=y;
          this.kiirus=kiirus;
          this.nurk=nurk;
          this.nurgamuutus=nurgamuutus;
         }
         this.liigu=function(){
           this.nurk+=this.nurgamuutus;
           this.x+=this.kiirus*Math.cos(this.nurk);
           this.y+=this.kiirus*Math.sin(this.nurk);
         }
         this.muudaNurk=function(uusNurk){
            this.nurk=uusNurk;
         }
         this.uusRooliAsend=function(uusAsend){
            this.nurgamuutus=uusAsend;
         }
         this.joonista=function(g){
            g.beginPath();
            g.arc(this.x, this.y, 10, 0, 2*Math.PI, false);
            g.stroke();
            g.beginPath();
            g.moveTo(this.x,this.y);
            //Joone ots
            var jotsx=this.x+10*this.kiirus*Math.cos(this.nurk);
            var jotsy=this.y+10*this.kiirus*Math.sin(this.nurk);
            g.lineTo(jotsx, jotsy);

            var rattapikkus=10;
            var rattanurk=this.nurk+3*this.nurgamuutus;
            var rattaotsx=jotsx+rattapikkus*Math.cos(rattanurk);
            var rattaotsy=jotsy+rattapikkus*Math.sin(rattanurk);
            g.lineTo(rattaotsx, rattaotsy)
            g.stroke();
```

```
        }
        this.algus();
      }

      function Rool(keskx, kesky, raadius, nurk){
         this.algus=function(){
            this.keskx=keskx;
            this.kesky=kesky;
            this.raadius=raadius;
            this.nurk=nurk;
         }
         this.kysiNurk=function(){
            return this.nurk+Math.PI/2;
         }
         this.joonista=function(g){
             g.beginPath();
             g.arc(this.keskx, this.kesky, this.raadius,
                  this.nurk-0.4, this.nurk+0.4, false);
             g.stroke();
             g.beginPath();
             g.arc(this.keskx+this.raadius*Math.cos(this.nurk),
                   this.kesky+this.raadius*Math.sin(this.nurk),
                     5, 0, 2*Math.PI, false);
             g.stroke();
            g.save();
            g.translate(this.keskx, this.kesky);
            g.rotate(this.nurk+(Math.PI/2));
            g.fillText("Rool", 0, 0);
            g.restore();
         }
         this.uusNurk=function(hx, hy){
            this.nurk=Math.atan2(hy-kesky, hx-keskx);
         }
         this.algus();
      }

      var auto1=new Liikur(100, 100, 1, Math.PI/3, -0.1);
      var rool1=new Rool(50, 100, 30, -Math.PI/2);

      function hiirLiigub(e){
         var tahvlikoht=document.getElementById("t1").getBoundingClientRect();
         var hx=e.clientX-tahvlikoht.left;
         var hy=e.clientY-tahvlikoht.top;
         rool1.uusNurk(hx, hy);
      }

      function liigu(){
         var g=document.getElementById("t1").getContext("2d");
         g.clearRect(0, 0, 400, 300);
         auto1.uusRooliAsend(rool1.kysiNurk()/10);
         auto1.liigu();
         auto1.joonista(g);
         rool1.joonista(g);
      }
    </script>
  </head>
  <body onload="setInterval('liigu()', 50);">
     <h1>Ringjoone parameetriline võrrand</h1>
     <canvas id="t1" width="400" height="300"
       style="background-color: yellow"
       onmousemove="hiirLiigub(event)"></canvas>
  </body>
</html>
```

Jooniselt võib näha, kuidas siis saab rooli abil liikurit mitut moodi sättida.





Ülesandeid

- Pane näide käima
- Muuda pööramise suurust vastavalt rooli asendile.

Ringrajasõit

- Pane auto sõitma ümber täpi, loe mitu ringi on läbitud
- Ringi kraadide järgi saab määrata "väravad", kui kaugelt mingi nurk tuleb läbida. Väravad on näha joonisel ning nende õiget läbimist kontrollitakse ringide kohta punktide arvutamisel.

3D

Kolmemõõtmeliste lahenduste loomiseks on mitmeid raamistikke loodud, kus kasutajal võimalus kujundid ja kaamera sobivasse kohta sättida ning sealtkaudu meeldivat ruumilist pilti vaadata. Veebigraafika üks abiline näiteks three.js nime all. Samas lihtsamat ruumilisuse efekti kannatab täiesti ka harilike kahemõõtmeliste joonistusvahendite abil tekitada

Kaugemal väiksemaks

Arvutijooniste puhul tuleb ikka eristada ekraani- ja maailmakoordinaate. Päris lihtsate jooniste korral võib ette kujutada, et üks piksel ekraanil on üks meeter looduses. Või siis üks piksel vastab inimese pikkuse ühele sentimeetrile. Kõiksugu muude suurenduste puhul aga peab väärtused sobiva koefitsiendiga läbi korrutama. Samuti ei pruugi me tahta arve lugeda joonistusala vasakust ülanurgast, vaid mugavam võib olla arvutada mõnest rohkem joonise keskel olevast punktist - selle väärtuse saab arvutuste juures juurde liita. Matemaatikaõpikus oleme harjunud, et y-telg suundub üles, arvutiekraanil suundub see esmalt alla. Ka selle saab märgimuutusega paika sättida. Kolmanda mõõtme juures tuleb maailmakoordinaatidest ekraanikoordinaatideks arvutades lihtsalt üks tehe juurde - x ja y tuleb kaugusega (z) läbi jagada. Ehk siis sama suur asi kaks korda kaugemal näeb kaks korda väiksem välja. Sobivaks nägemiseks tulevad arvud suurenduskordajaga läbi korrutada. Siin puhul selleks valitud kümme. Muutujad keskx ja kesky tähistavad koordinaatide alguspunkti joonisel. Funktsioone ex ja ey kasutatakse ekraani x-i ja y-i arvutamiseks vastavalt maailmakoordinaatides punkti kolme mõõtme väärtustele.

```
var keskx=200;
var kesky=150;
var suurendus=10;
function ex(px, py, pz){ //x ekraanil
   return keskx+suurendus*px/pz;
}

function ey(px, py, pz){
   return kesky-suurendus*py/pz;
}
```

Joonistamise puhul tuleb siis iga kord sobivas kohas lihtsalt vastavad funktsioonid välja kutsuda

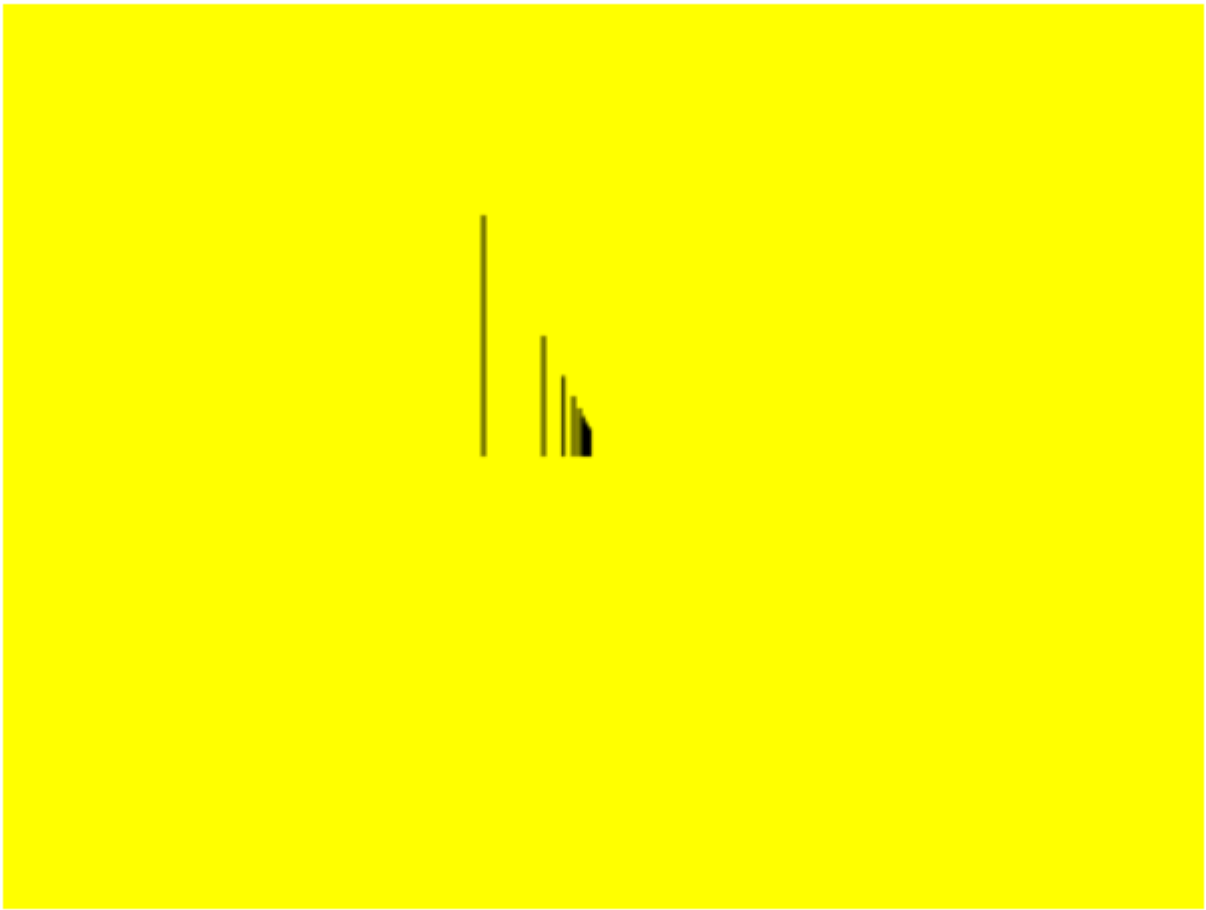
```
g.lineTo(ex(x, y, z), ey(x, y, z));
```

Ehk siis leitakse kolmemõõtmelistele algandmetele vastavad kahemõõtmelised punktid ekraanil. Joonisele tehakse tsükli abil postirida. Posti alumise otsa y on 0, ülemise oma 40 ühikut. Kaugused viiest viiekümne ühikuni.

```
<!doctype html>
<html>
  <head>
    <title>3D</title>
    <script>
       var keskx=200;
       var kesky=150;
       var suurendus=10;
       function ex(px, py, pz){ //x ekraanil
          return keskx+suurendus*px/pz;
       }

       function ey(px, py, pz){
```

```
          return kesky-suurendus*py/pz;
        }

        function joonista(){
          var g=document.getElementById("tahvel1").getContext("2d");
          g.beginPath();
          var x=-20;
          var y=40;
          for(var z=5; z<50; z+=5){
             g.moveTo(ex(x, 0, z), ey(x, 0, z));
             g.lineTo(ex(x, y, z), ey(x, y, z));
          }
          g.stroke();
        }
    </script>
  </head>
  <body onload="joonista();">
     <canvas id="tahvel1" width="400" height="300"
      style="background-color: yellow"/></canvas>
  </body>
</html>
```

Väikese ettekujutuse tulemusena võib lehel täiesti viisakat postirida näha.



Ülesandeid

- Pane näide tööle
- Katseta postide mitmesuguste kõrguste ja vahedega
- Loo sarnane postirida ka vaatajast paremale

- Paiguta postide otsa väikesed ringid nagu tänavalaternad

Kasutaja asukoha muutmine

Kolmemõõtmelises graafikas võivad liikuda kujundid, võib liikuda ka aga kaamera ehk vaataja ise. Kui vaatesuund jääb otse ette, siis polegi arvutamisel muud muret, kui tuleb ainult kasutaja asukoht kujundite asukohast maha lahutada. Ekraani x-koordinaadi arvutamine siis:

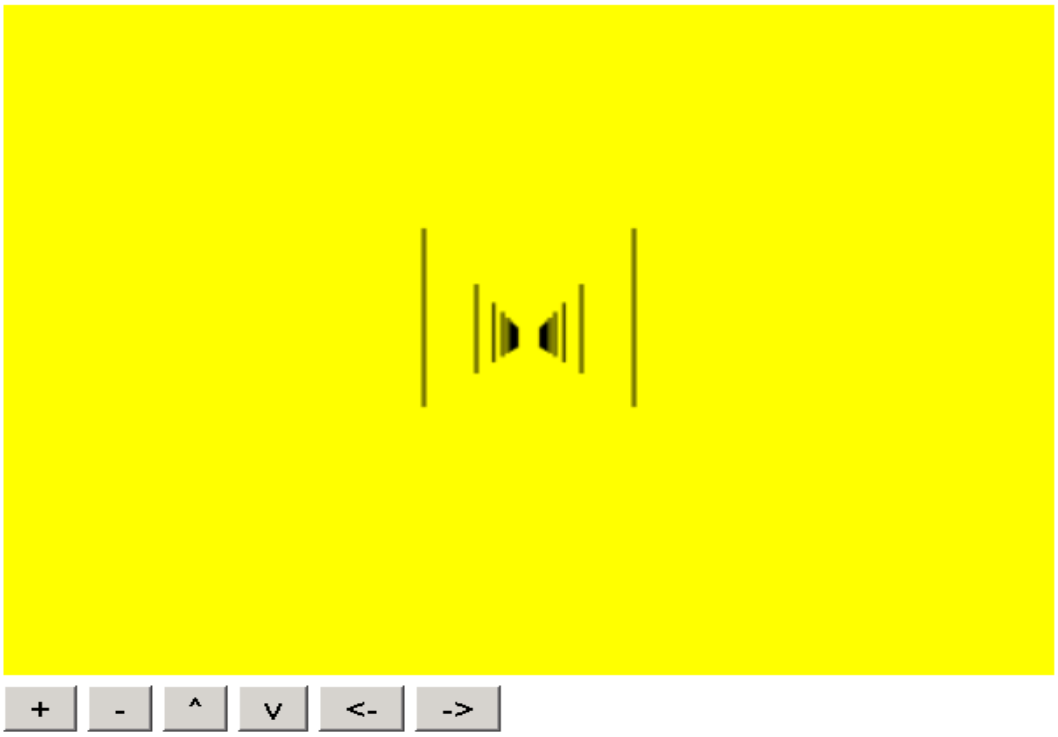
```
function ex(px, py, pz){ //x ekraanil
   return keskx+suurendus*(px-vaatajax)/(pz-vaatajaz);
}
```

Ehk siis nii x- kui z- koordinaadi puhul on vastav tehe tehtud. Mida ettepoole ise liikuda, seda lähemale tulevad kujundid. Mida ülespoole minna, seda enam paistavad kujundid allpool. Vaataja sammu muutuja näitab, kui pikkade hüpetega ta liigub. Juurde nupuriba vaataja andmete muutmiseks ning tekibki juba julgem ruumilise vaatamise tunne.

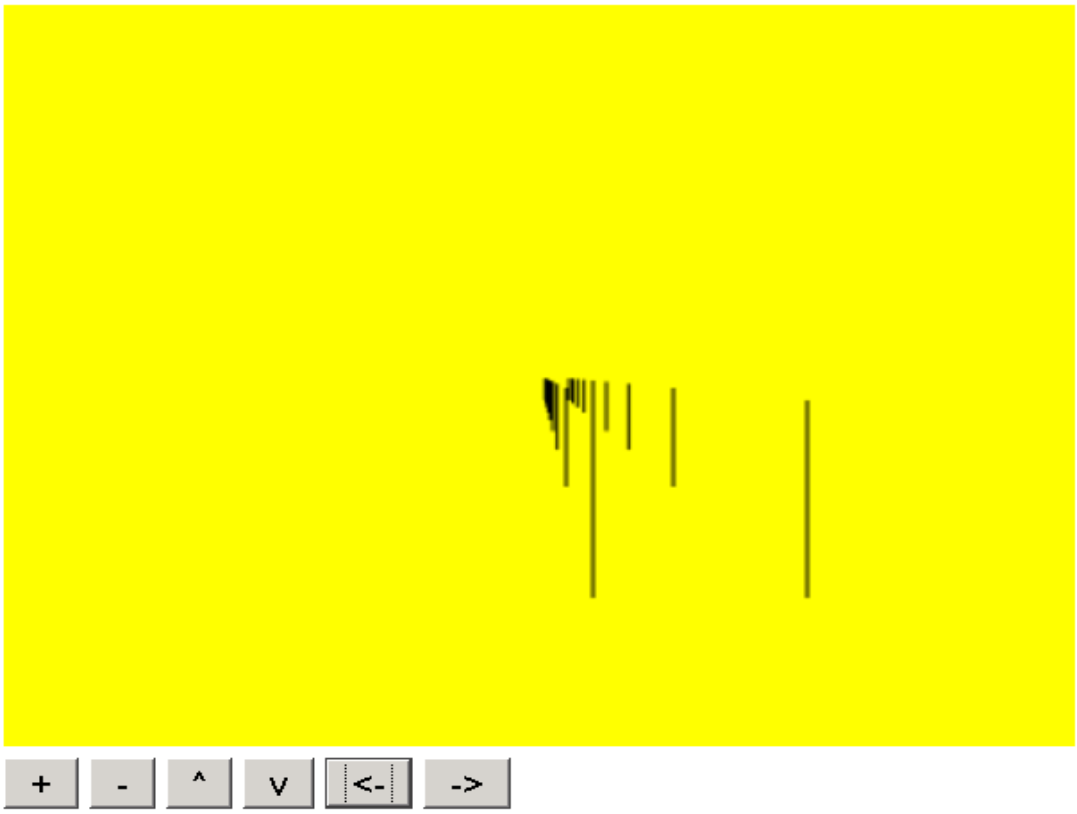
```
<!doctype html>
<html>
  <head>
    <title>3D</title>
    <script>
       var keskx=200;
       var kesky=150;
       var suurendus=10;
       var vaatajax=0;
       var vaatajay=0;
       var vaatajaz=-30;
       var vaatajasamm=5;
       function ex(px, py, pz){ //x ekraanil
          return keskx+suurendus*(px-vaatajax)/(pz-vaatajaz);
       }

       function ey(px, py, pz){
          return kesky-suurendus*(py-vaatajay)/(pz-vaatajaz);
       }

       function joonista(){
         var g=document.getElementById("tahvel1").getContext("2d");
         g.clearRect(0, 0, 400, 300);
         g.beginPath();
         var x=-20;
         var y=40;
         for(var z=5; z<50; z+=5){
            g.moveTo(ex(x, 0, z), ey(x, 0, z));
            g.lineTo(ex(x, y, z), ey(x, y, z));
         }
         //Parema poole postid
         x=20;
         for(var z=5; z<50; z+=5){
            g.moveTo(ex(x, 0, z), ey(x, 0, z));
            g.lineTo(ex(x, y, z), ey(x, y, z));
         }
         g.stroke();
       }

       function vasakule(){vaatajax-=vaatajasamm; joonista();}
       function paremale(){vaatajax+=vaatajasamm; joonista();}
       function yles(){vaatajay+=vaatajasamm;     joonista();}
```

```
      function alla(){vaatajay-=vaatajasamm;     joonista();}
      function edasi(){vaatajaz+=vaatajasamm;    joonista();}
      function tagasi(){vaatajaz-=vaatajasamm;   joonista();}
    </script>
  </head>
  <body onload="joonista();">
     <canvas id="tahvel1" width="400" height="300"
      style="background-color: yellow"/></canvas><br />
     <input type="button" onclick="edasi()" value="+"/>
     <input type="button" onclick="tagasi()" value="-"/>
     <input type="button" onclick="yles()" value="^"/>
     <input type="button" onclick="alla()" value="v"/>
     <input type="button" onclick="vasakule()" value="&lt;-"/>
     <input type="button" onclick="paremale()" value="-&gt;"/>

  </body>
</html>
```

Vaade eest



ning vaade pärast mõningast üles- ja vasakule poole liikumist.



Ülesandeid

- Pane näide tööle
- Koosta kujundiks nelinurkse varjualuse sõrestik - postid ja katusepüramiid. Vaata seda mitmest asukohast
- Koosta kujundiks kuuenurkse varjualuse sõrestik, vaata
- Koosta kujundiks kasutaja määratud nurkade arvuga varjualuse sõrestik

Kolmemõõtmeliste andmete objekt

Üksikuid punkte ja jooni arvutada ja kuvada saab täiesti tavaliste muutujatega. Kui aga vaja koos hakata liigutama hulgast punktidest koosnevaid kujundeid, siis tekib üksikuid punkte tülikalt suurel hulgal. Objektitüüpide ja objektide abil aga on võimalik neid mugavamalt hallata, neile ühekorraga käsklusi jagada. Samuti saab niimoodi mitmemõõtmelised arvutustehted nõnda koodi sisse ära peita, et hilisema töö juures ei pea nii palju arvutusvigade pärast muretsema, kui kood algul korralikult üle kontrollitud sai.

Vektor

Punkt ja vektor mõnevõrra sarnased nähtused. Esimesega küll oleme ehk enam harjunud tähistama asukohta ning teisega liikumist. Aga iseenesest mõlemal on vajalik arv mõõtmeid ning saab külge panna arvutusteks vajalikud käsud, nii et otsest põhjust eraldi tüüpide loomiseks ei ole. Mõistet asukohavektor ka täiesti kasutatakse, nii et las siis siingi piirdutakse ühe tüübiga kolme koordinaadiga määratud suuruse tähistamiseks. Juurde levinud funktsioonid liitmise ja lahutamise kohta. Samuti sisu tekstina väljastamiseks. Kusjuures tähele tasub panna, et praeguse kahe vektori liitmisel tekib uus, kolmas vektor, esialsete vektorite koordinaadid ei muutu.

Objektidest andmete tekstina välja küsimiseks on Javaskriptil olemas mugav JSON (JavaScript Object Notation) - vorming. Andmed paigutatakse looksulgude vahele. Sinna sisse järgemööda tunnuse nimi(võti) ning kooloni taga järgnev väärtus. Nagu näiteks siin algandmed

```
{"x":2,"y":4,"z":6}
```

Käsklus JSON.stringify() teeb objektist sarnase andmetega stringi. Kui mõnikord on vaja andmeid tekstist tagasi terviklikuks objektiks muuta, siis selleks sobib käsklus JSON.parse().

Nüüd aga kolmemõõtmelise vektori kood

```
<!doctype html>
<html>
  <head>
    <title>3D</title>
    <script>
       function Vektor3D(x, y, z){
          this.x=x;
          this.y=y;
          this.z=z;
          this.liida=function(v2){
            return new Vektor3D(this.x+v2.x, this.y+v2.y, this.z+v2.z);
          }
          this.lahuta=function(v2){
            return new Vektor3D(this.x-v2.x, this.y-v2.y, this.z-v2.z);
          }
          this.tekstina=function(){
             return JSON.stringify(this);
          }
       }
       function alusta(){
          var v1=new Vektor3D(2, 4, 6);
          var v2=new Vektor3D(0, 1, 0);
          var v3=v1.liida(v2);
          document.getElementById("kiht1").innerHTML=v3.tekstina();
       }
    </script>
  </head>
  <body onload="alusta();">
    <div id="kiht1"></div>
  </body>
</html>
```

Töö tulemus kihil:

```
{"x":2,"y":5,"z":6}
```

Ülesandeid

- Pane näide tööle
- Lisa vektorile käsklus korruta(arv), mis väljastab uue vektori, kus kõik esialgse vektori koordinaadid on selle arvuga korrutatud. Veendu, et tulemus toimib.

Kahest punktist joon

Vektor hoiab ühe punkti kolm koordinaati ilusti koos. Kui tahta, et joone andmeid saaks ühe

tervikuna käsitleda, siis tuleb omakorda joone kaks otspunkti üheks tervikuks siduda. Siin seda ka tehakse. Joonele luuakse punktide jaoks massiiv, kus kohal 0 on üks otspunkt ning kohal 1 teine.

Arvestades varasemat näidet vaataja asukoha liigutamise kohta, lisatakse eraldi tüüp ja objekt ka vaataja asukoha meelespidamiseks ning vastavalt sellele Joone joonistuskäskluses ekraanikoordinaatide arvutamiseks. Valemid sarnased kui enne, lihtsalt nüüd on need funktsioonide sisse pandud ja funktsioonid omakorda objektiks kapseldatud, et juhtkoodis tekkida võivat segadust vähemaks saada. Kui tükid olemas, siis nende ühendamine võiks juba üsna lihtsalt minna. Praegu luuakse kõigepealt uus Joon, mille otspunktide koordinaatideks saavad kaks Vektor3D-d oma väärtustega. Muud toimetused pandi alusta-funktsiooni sisse, sest tahvlile pole võimalik lehe päises avanemise ajal ligi saada. Küll aga on tahvel kättesaadav pärast lehe laadimist body onload-sündmuse peale käivitatava funktsiooni alusta() sees. Vaatajale määratakse asukoht (hetkel nullpunkt), joonistamise graafiline kontekst ning koordinaatide nullkoha asukoht ekraanil (hetkel tahvli keskkoht). Pärast joonistuskäsklust võikski joon ekraanil näha olla.

```
var j1=new Joon(new Vektor3D(-20, 0, 5), new Vektor3D(-20, 40, 5));
var vaataja;
function alusta(){
   var tahvel=document.getElementById("tahvel1");
   vaataja=new Vaataja(new Vektor3D(0, 0, 0),
     tahvel.getContext("2d"), tahvel.width/2, tahvel.height/2);
   j1.joonista(vaataja);
}
```

Edasi töötav kood tervikuna.

```
<!doctype html>
<html>
  <head>
    <title>3D</title>
    <script>
       function Vektor3D(x, y, z){
          this.x=x;
          this.y=y;
          this.z=z;
          this.liida=function(v2){
            return new Vektor3D(this.x+v2.x, this.y+v2.y, this.z+v2.z);
          }
          this.lahuta=function(v2){
            return new Vektor3D(this.x-v2.x, this.y-v2.y, this.z-v2.z);
          }
          this.tekstina=function(){
             return JSON.stringify(this);
          }
       }

       function Joon(v1, v2){
          this.punktid=new Array();
          this.punktid[0]=v1;
          this.punktid[1]=v2;
          this.joonista=function(vaataja){
             vaataja.g.beginPath();
             vaataja.g.moveTo(
                  vaataja.ex(this.punktid[0]), vaataja.ey(this.punktid[0]));
             vaataja.g.lineTo(
                  vaataja.ex(this.punktid[1]), vaataja.ey(this.punktid[1]));
             vaataja.g.stroke();
          }
       }
```

```
      function Vaataja(asukoht, g, keskx, kesky){
         this.asukoht=asukoht;
         this.g=g;
         this.suurendus=10;
         this.keskx=keskx;
         this.kesky=kesky;
         this.ex=function(p){
            var kaugus=p.lahuta(this.asukoht);
            return keskx+this.suurendus*kaugus.x/kaugus.z;
         }
         this.ey=function(p){
            var kaugus=p.lahuta(this.asukoht);
            return kesky-this.suurendus*kaugus.y/kaugus.z;
         }
      }

      var j1=new Joon(new Vektor3D(-20, 0, 5), new Vektor3D(-20, 40, 5));
      var vaataja;
      function alusta(){
         var tahvel=document.getElementById("tahvel1");
         vaataja=new Vaataja(new Vektor3D(0, 0, 0),
           tahvel.getContext("2d"), tahvel.width/2, tahvel.height/2);
         j1.joonista(vaataja);
      }
    </script>
  </head>
  <body onload="alusta();">
     <canvas id="tahvel1" width="400" height="300"
      style="background-color: yellow"/></canvas>
   </body>
</html>
```

Ülesandeid

- Pane näide tööle
- Joonista nõnda ekraanile kaks Joont
- Lisa Joonele värvi omadus
- Joonista ekraanile kaks eri värvi joont
- Muuda Vaataja asukohta

Joonte kogum

Üksikut joont on tõenäoliselt lihtsam ilma igasuguste objektideta joonistada. Kui mitu joont aga moodustavad uue terviku, siis on see mõistlik uude kesta panna - nii on võimalik seda hiljem korraga joonistada, liigutada või värvida. Kogumi juurde tuleb massiiv millesse andmeid lisada. Kogumile antud joonistuskäsklus käib läbi kõik kogumis olevad jooned ning kutsub välja neist igaühe joonistuskäskluse. Praegu siin juures ka käsklus vaataja tahvli puhastamiseks, aga selle saab mujale panna, kui tekib soov mitu kujundit järjestikku samale tahvlile joonistada.

```
function KujundiKogum(){
   this.kujundid=new Array();
   this.lisaKujund=function(k){this.kujundid.push(k);}
   this.joonista=function(vaataja){
      vaataja.g.clearRect(0, 0, 400, 300);
      for(var i=0; i<this.kujundid.length; i++){
         this.kujundid[i].joonista(vaataja);
      }
   }
}
```

Vaataja juurde sai liikumisoskus. Kõigepealt määrati vaatajale sammu pikkus, et kui palju ta iga korraga liigub.

```
this.vaatajasamm=5;
```

Liikumisfunktsiooni juures liidetakse vaataja asukohale sammuvektor.

```
this.liigu=function(samm){
   this.asukoht=this.asukoht.liida(samm);
}
```

Vastavalt liikumissuunale luuakse siis sammuvektor liikumiseks vaatajasammu jagu vastavas suunas.

```
this.paremale=function(){
        this.liigu(new Vektor3D(this.vaatajasamm, 0, 0));}
```

Lehele lisaks nupud vaataja liigutamiseks sobivas suunas. Kuna Vaataja-tüübist tehti vaataja-nimeline globaalmuutuja, siis saab sellele käsklusi andes lehel olevat vaadet muuta.

```
<input type="button" onclick="vaataja.edasi(); uuenda();" value="+"/>
```

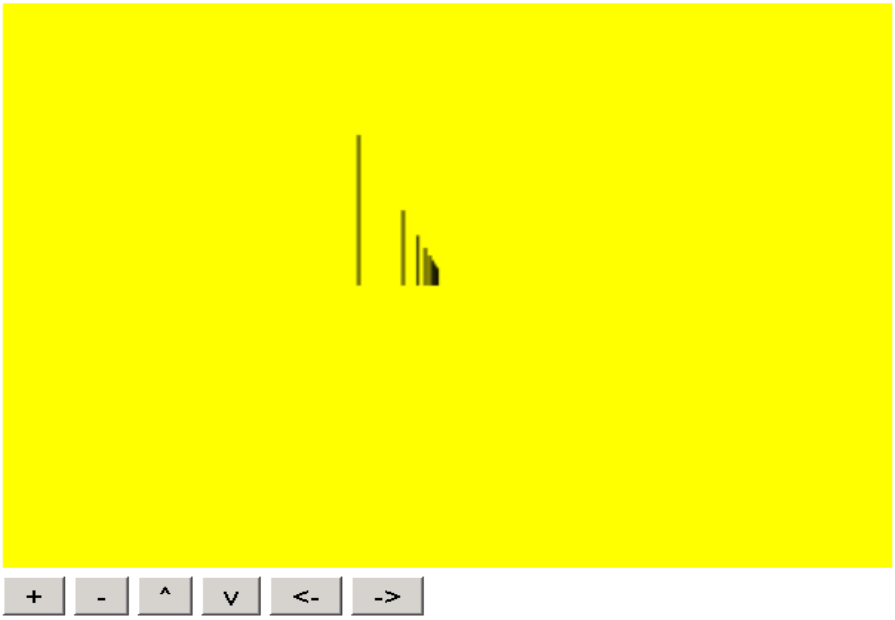
Edasi juba kood tervikuna.

```
<!doctype html>
<html>
  <head>
    <title>3D</title>
    <script>
```

```
function Vektor3D(x, y, z){
   this.x=x;
   this.y=y;
   this.z=z;
   this.liida=function(v2){
     return new Vektor3D(this.x+v2.x, this.y+v2.y, this.z+v2.z);
   }
   this.lahuta=function(v2){
     return new Vektor3D(this.x-v2.x, this.y-v2.y, this.z-v2.z);
   }
   this.tekstina=function(){
      return JSON.stringify(this);
   }
}

function Joon(v1, v2){
   this.punktid=new Array();
   this.punktid[0]=v1;
   this.punktid[1]=v2;
   this.joonista=function(vaataja){
      vaataja.g.beginPath();
      vaataja.g.moveTo(vaataja.ex(this.punktid[0]),
          vaataja.ey(this.punktid[0]));
      vaataja.g.lineTo(vaataja.ex(this.punktid[1]),
          vaataja.ey(this.punktid[1]));
      vaataja.g.stroke();
   }
}

function KujundiKogum(){
   this.kujundid=new Array();
   this.lisaKujund=function(k){this.kujundid.push(k);}
   this.joonista=function(vaataja){
      vaataja.g.clearRect(0, 0, 400, 300);
      for(var i=0; i<this.kujundid.length; i++){
         this.kujundid[i].joonista(vaataja);
      }
   }
}

function Vaataja(asukoht, g, keskx, kesky){
   this.asukoht=asukoht;
   this.g=g;
   this.suurendus=10;
   this.keskx=keskx;
   this.kesky=kesky;
   this.vaatajasamm=5;
   this.ex=function(p){
      var kaugus=p.lahuta(this.asukoht);
      return keskx+this.suurendus*kaugus.x/kaugus.z;
   }
   this.ey=function(p){
      var kaugus=p.lahuta(this.asukoht);
      return kesky-this.suurendus*kaugus.y/kaugus.z;
   }
   this.liigu=function(samm){
      this.asukoht=this.asukoht.liida(samm);
   }
   this.paremale=function(){
         this.liigu(new Vektor3D(this.vaatajasamm, 0, 0));}
   this.vasakule=function(){
         this.liigu(new Vektor3D(-this.vaatajasamm, 0, 0));}
   this.yles=function(){
         this.liigu(new Vektor3D(0, this.vaatajasamm, 0));}
```

```
            this.alla=function(){
                  this.liigu(new Vektor3D(0, -this.vaatajasamm, 0));}
            this.edasi=function(){
                  this.liigu(new Vektor3D(0, 0, this.vaatajasamm));}
            this.tagasi=function(){
                  this.liigu(new Vektor3D(0, 0, -this.vaatajasamm));}
         }

         var vaataja;
         var kogum;
         function alusta(){
            var tahvel=document.getElementById("tahvel1");
            vaataja=new Vaataja(new Vektor3D(0, 0, 0),
              tahvel.getContext("2d"), tahvel.width/2, tahvel.height/2);
            kogum=new KujundiKogum(vaataja);
            for(var z=5; z<50; z+=5){
              kogum.lisaKujund(
                new Joon(new Vektor3D(-20, 0, z), new Vektor3D(-20, 40, z)));
            }
            kogum.joonista(vaataja);
         }

         function uuenda(){
            kogum.joonista(vaataja);
         }
      </script>
   </head>
   <body onload="alusta();">
      <canvas id="tahvel1" width="400" height="300"
       style="background-color: yellow"/></canvas><br />
          <input type="button" onclick="vaataja.edasi(); uuenda();" value="+"/>
      <input type="button" onclick="vaataja.tagasi(); uuenda();" value="-"/>
      <input type="button" onclick="vaataja.yles(); uuenda();" value="^"/>
      <input type="button" onclick="vaataja.alla(); uuenda();" value="v"/>
      <input type="button" onclick="vaataja.vasakule(); uuenda();"
                value="&lt;-"/>
      <input type="button" onclick="vaataja.paremale(); uuenda();"
                value="-&gt;"/>
    </body>
</html>
```

Lehel saab nõnda vaatajaga ringi liikuda.



Ülesandeid

- Pane näide tööle
- Koosta joontest kaks kogumit: kahest joonest T-täht ning kuuest joonest tetraeeder (kolmnurkne püstprisma). Kuva mõlemad ekraanile. Vaata neid kaamera liigutamise abil mitmelt poolt.
- Lisa kogumile omaduseks toon. Kuva kumbki kogum vastava värviga ekraanile.

- Lisa kogumile käsklus teeSamm, mis siis kogumis olevaid kujundeid etteantud vektori jagu edasi liigutab. Lisa lehele nupp ühe ekraanil oleva kogumi liigutamiseks.
- Lisa kogumile muutuja liikumissammu vektorina hoidmiseks. Pane kumbki kujund ühtlase kiirusega tema küljes olevas suunas liikuma.

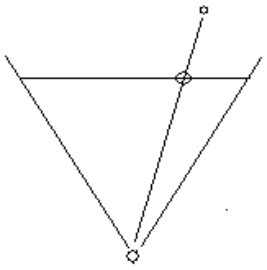
Reaalse mõõtkava arvestamine

Siiani tehtud näidetes jagati kaugustunnetuse saamiseks x ja y-koordinaat lihtsalt kaugusega läbi ning vajadusel korrutati suurenduskordajaga, et pilt enamvähem sobivas suuruses oleks. Tõsielulise olukorra järele tegemisel aga saab andmed ka täpsemalt näiteks meetrites välja arvestada ning siis vaateaval mõttelisel kaugusel olevale kilele projitseerida. Ning edasi otsustada, millise suurendusega projektsioonikile punktid arvutiekraani piksliteks ümber arvutada. Kolmnurga valemite järgi on vaadeldava punkti ühe (nt. x) mõõtme koordinaadi ja silmast kauguse suhe sama kui selle punktini viiva joone lõikekoha kilel ja kile kauguse suhe. Sellise arvutuse järgi õnnestub leida koht kilel

```
(px-vaatajax)*kilekaugus/(pz-vaatajaz)
```

Kuna vaataja ei pruugi olla koordinaatide nullpunktis ning tavajuhul on vaatekile mõõtmed pigem võrreldav ekraanipikslite arvuga sentimeetrites, siis saab kolmemõõtmelised koordinaadid kahemõõtmelisteks arvutada järgnevalt.

```
var suurendus=100; //sentimeetrit meetri kohta
function ex(px, py, pz){ //x ekraanil
   return keskx+suurendus*(px-vaatajax)*kilekaugus/(pz-vaatajaz);
}
```

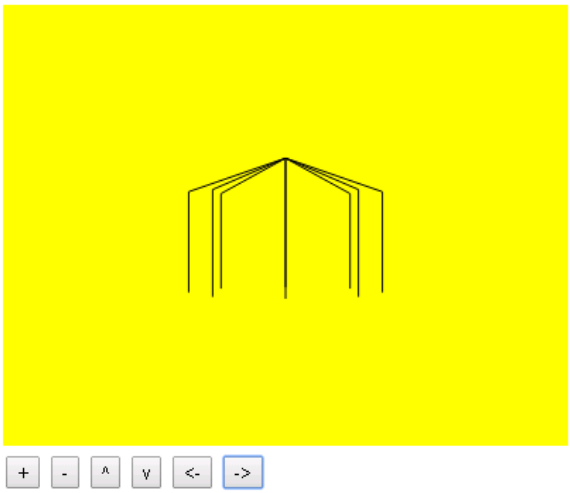


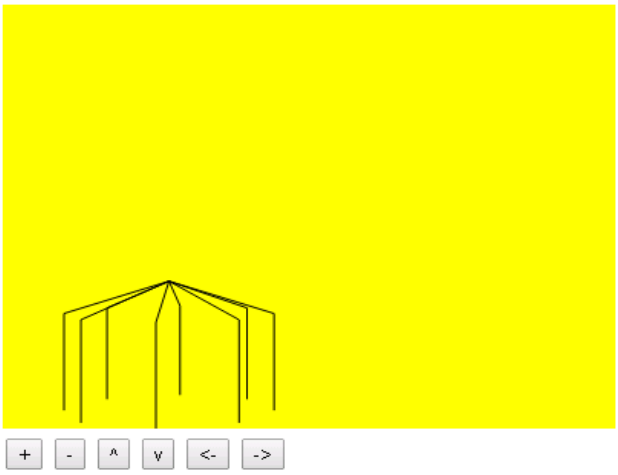
Edasi juba programmikood tervikuna

```
<!doctype html>
```

```
<html>
  <head>
    <title>3D</title>
    <script>
       var keskx=200;
       var kesky=150;
       var suurendus=100; //sentimeetrit meetri kohta
       var vaatajax=0;
       var vaatajay=2;
       var vaatajaz=-25;
       var kilekaugus=8; //meetrit
       var vaatajasamm=2;
       var telkx=0;
       var telkz=0;
       var telgiraadius=3; //meetrit
       var telgikorgus=3; //meetrit
       var tipukorgus=4; //meetrit
       var nurkadearv=8;
       function ex(px, py, pz){ //x ekraanil
          return keskx+suurendus*(px-vaatajax)*kilekaugus/(pz-vaatajaz);
       }

       function ey(px, py, pz){
          return kesky-suurendus*(py-vaatajay)*kilekaugus/(pz-vaatajaz);
       }

       function joonista(){
         var g=document.getElementById("tahvel1").getContext("2d");
         g.clearRect(0, 0, 400, 300);
         g.beginPath();
         var nurgavahe=2*Math.PI/nurkadearv;
         var nurk=0;
         for(var i=0; i<nurkadearv; i++, nurk+=nurgavahe){
            var x=telkx+telgiraadius*Math.cos(nurk);
            var z=telkz+telgiraadius*Math.sin(nurk);
            g.moveTo(ex(x, 0, z), ey(x, 0, z));
            g.lineTo(ex(x, telgikorgus, z), ey(x, telgikorgus, z));
            g.lineTo(ex(telkx, tipukorgus, telkz),
               ey(telkx, tipukorgus, telkz));
         }
         g.stroke();
       }

       function vasakule(){vaatajax-=vaatajasamm; joonista();}
       function paremale(){vaatajax+=vaatajasamm; joonista();}
       function yles(){vaatajay+=vaatajasamm;     joonista();}
       function alla(){vaatajay-=vaatajasamm;     joonista();}
       function edasi(){vaatajaz+=vaatajasamm;    joonista();}
       function tagasi(){vaatajaz-=vaatajasamm;   joonista();}
    </script>
  </head>
  <body onload="joonista();">
     <canvas id="tahvel1" width="400" height="300"
      style="background-color: yellow"/></canvas><br />
     <input type="button" onclick="edasi()" value="+"/>
     <input type="button" onclick="tagasi()" value="-"/>
     <input type="button" onclick="yles()" value="^"/>
     <input type="button" onclick="alla()" value="v"/>
     <input type="button" onclick="vasakule()" value="&lt;-"/>
     <input type="button" onclick="paremale()" value="-&gt;"/>
  </body>
</html>
```

Samuti kaheksanurkse telgi vaated mitmest asukohast.





Andmehaldus

Eesnimede loetelu

Alustuseks näide, kuidas massiivis lehel andmeid hoida ning nad sobivalt välja kuvada. Nimehalduse objekt nagu varasemateski näidetes seotud lehel oleva kihiga. Lisamisfunktsiooni abil saab objekti massiivile eesnimesid juurde lisada, kuvamiskäskluse peale kuvatakse nad ekraanile. Objekti sees hoolitsetakse, et iga muutuse puhul (praegusel juhul andmete lisamisel) uuendataks ka väljundit.

```
<!doctype html>
<html>
  <head>
     <title>Eesnimede haldus</title>
         <script>
             function Nimehaldus(kihinimi){
                   this.algus=function(){
                      this.kiht=document.getElementById(kihinimi);
                          this.andmed=new Array();
                   }
                   this.kuva=function(){
                      var t="<ul>";
                          for(var i=0; i<this.andmed.length; i++){
                             t+="<li>"+this.andmed[i]+"</li>";
                          }
                          t+="</ul>";
                          this.kiht.innerHTML=t;
                   }
                   this.lisaNimi=function(uusnimi){
                      this.andmed.push(uusnimi);
                          this.kuva();
                   }
                   this.algus();
                }
                var h1;
                function lehealgus(){
                   h1=new Nimehaldus("kiht1");
                       h1.lisaNimi("Juku");
                       h1.lisaNimi("Kati");
                       h1.lisaNimi("Ants");
                }
```

```
        </script>
  </head>
  <body onload="lehealgus();">
     <h1>Nimeharjutused</h1>
         <div id="kiht1"></div>
  </body>
</html>
```

Nimeharjutused

- Juku
- Kati
- Ants

Funktsiooni lisaargumendid

Javaskripti funktsiooni omapära on, et sinna sisestatavate argumentide arv pole piiratud. Olemasolevatele võib käivitamisel alati soovitud väärtusi juurde lisada, need saadakse funktsiooni sees kätte muutuja arguments kaudu. Nii siinses näites Nimehalduse loomisel eeldatakse, et antakse ette vähemasti kihi nimi. Edasised parameetrid lisatakse tsükli abil eesnimedena nimeloetellu. Seetõttu hakkab loendur nimega abi ühest, kuna arguments[0] on esimene parameeter ehk kihi nimi.

```
<!doctype html>
<html>
  <head>
     <title>Eesnimede haldus</title>
         <script>
             function Nimehaldus(kihinimi){
                   this.algus=function(){
                      this.kiht=document.getElementById(kihinimi);
                          this.andmed=new Array();
                   }
                   this.kuva=function(){
                      var t="<ul>";
                          for(var i=0; i<this.andmed.length; i++){
                             t+="<li>"+this.andmed[i]+"</li>";
                          }
                          t+="</ul>";
                          this.kiht.innerHTML=t;
                   }
                   this.lisaNimi=function(uusnimi){
                      this.andmed.push(uusnimi);
                          this.andmed.sort();
                          this.kuva();
                   }
                   this.algus();
                   for(var abi=1; abi<arguments.length; abi++){
                        //funktsiooni (ka varjatud) argumendid
                      this.lisaNimi(arguments[abi]);
                   }
                }
                var h1;
                function lehealgus(){
                   h1=new Nimehaldus("kiht1","Juku","Sass", "Ants");
                }
```

```
        </script>
  </head>
  <body onload="lehealgus();">
     <h1>Nimeharjutused</h1>
         <div id="kiht1"></div>
  </body>
</html>
```

Nimeharjutused

- Ants
- Juku
- Sass

Punktide haldus

Esialgu peeti meeles vaid üksikuid väärtusi ehk eesnimesid. Nüüd aga väärtused meeles kirjetena - iga kasutaja juures kirjas ka, et kui palju talle punkte antud on. Andmed ikka massiivis, aga väljastuskujundus loetelu asemel tabelis. Samuti tulemus lisatakse objekti ehk kirjena , väljadeks kasutajanimi ning punkte.

```
<!doctype html>
<html>
  <head>
     <title>Haldus</title>
         <script>
             function Tulemushaldus(kihinimi){
                   this.algus=function(){
                      this.kiht=document.getElementById(kihinimi);
                          this.andmed=new Array();
                   }
                   this.kuva=function(){
                      var t="<table><tr><th>Kasutaja</th><th>Punktid</th></tr>";
                          for(var i=0; i<this.andmed.length; i++){
                             t+="<tr><td>"+this.andmed[i].kasutajanimi+
                               "</td><td>"+this.andmed[i].punkte+"</td></tr>";
                          }
                          t+="</table>";
                          this.kiht.innerHTML=t;
                   }
                   this.lisaTulemus=function(kasutajanimi, punkte){
                      this.andmed.push(
                          {"kasutajanimi": kasutajanimi, "punkte": punkte});
                      this.kuva();
                   }
                   this.algus();
                }
                var h1;
                function lehealgus(){
                   h1=new Tulemushaldus("kiht1");
                       h1.lisaTulemus("Juku", 10);
                       h1.lisaTulemus("Sass", 8);
                }
```

```
        </script>
  </head>
  <body onload="lehealgus();">
     <h1>Nimeharjutused</h1>
         <div id="kiht1"></div>
  </body>
</html>
```

Nimeharjutused

Kasutaja	Punktid
Juku	10
Sass	8

Sortimine

Kui tegemist võistlustulemustega, siis on tähtsus punktide järgi järjestusel. Kuna kirjes mitu välja, siis arvuti omast tarkusest ei tea mille järgi andmed ritta panna. Massiivi elemendis oleva ühe väärtuse korral piisab käsust sort() ning andmed on vastava tunnuse vaikimisi väärtuste järgi ritta pandud - arvud suuruse ning tekstid tähestiku järgi kasvavalt. Mitme tulba puhul tuleb aga öelda, mille järgi järjestada. "Ütlemiseks" on eraldi funktsioon, millele hakatakse sorditavaid andmeridu kahekaupa ette andma. Kui esimene neist peaks olema järjekorras eespool, siis tagastatakse negatiivne arv, kui tagumine eespool, siis positiivne. Kui järjestatava tunnuse järgi pole järjekord tähtis, siis tuleb tagasi anda 0. Kogu selle töö suudab ära teha käsklus

```
this.andmed.sort(function(k1, k2){return k1.punkte-k2.punkte});
```

Edasi juba kood tervikuna.

```
<!doctype html>
<html>
  <head>
     <title>Haldus</title>
         <script>
             function Tulemushaldus(kihinimi){
                   this.algus=function(){
                      this.kiht=document.getElementById(kihinimi);
                          this.andmed=new Array();
                   }
                   this.kuva=function(){
                      var t="<table><tr><th>Kasutaja</th><th>Punktid</th></tr>";
                          for(var i=0; i<this.andmed.length; i++){
                             t+="<tr><td>"+this.andmed[i].kasutajanimi+
                                   "</td><td>"+this.andmed[i].punkte+"</td></tr>
";
                          }
                          t+="</table>";
                          this.kiht.innerHTML=t;
                   }
                   this.lisaTulemus=function(kasutajanimi, punkte){
                      this.andmed.push({"kasutajanimi": kasutajanimi, "punkte":
parseInt(punkte)});
                          this.andmed.sort(function(k1, k2){return k1.punkte-
```

```
k2.punkte});
                            this.kuva();
                    }
                    this.algus();
                }
                var h1;
                function lehealgus(){
                   h1=new Tulemushaldus("kiht1");
                        h1.lisaTulemus("Juku", 10);
                        h1.lisaTulemus("Sass", 8);
                }
        </script>
  </head>
  <body onload="lehealgus();">
     <h1>Nimeharjutused</h1>
        <div id="kiht1"></div>
  </body>
</html>
```

Nimeharjutused

Kasutaja	Punktid
Sass	8
Juku	10

Ülesandeid

- Pane näited käima
- Koosta lehele üks kiht poiste nimede jaoks, teine tüdrukute nimede jaoks, mõlemad sorditakse lisamisel tähestiku järgi
- Lisaks nimedele on inimeste juures kirjas ka pikkused. Andmed sorditakse kõigepealt pikkuste järgi, sama pikkusega inimesed pannakse ritta tähestiku järjekorras
- Loo eraldi objekt, milles võimalik hoida tantsupaaride andmeid. Väljadena kirjas mõlema tantsija nimed ja pikkused. Loo lehele nupp, millele vajutades võetakse nii poiste kui tüdrukute loetelust esimene inimene ning lisatakse tekkinud paar tantsupaaride loetellu, paariks läinud inimesed eemaldatakse üksinda ootajate loetelust.

Salvestamine

Veebilehe kliendirakendustega kipub enamasti olema, et kui aken kinni pannakse, siis on andmed läinud. Või tuleb eraldi serveri poole mõni kaval tükk kirjutada, et tehtu alles jääks. Mõningaid andmeid sai vanasti salvestada küpsistega, kuid enamasti pidi sealjuures piirduma mõne üksiku arvu või tekstiga, et brauseri mahulimiiti ei ületaks. Alates HTML5st kasutada aga muutujad sessionStorage ning localStorage, mille sees ligi viie megabaidi jagu andmeid hoida saab ning seetõttu ei pea salvestamisega liialt kokkuhoidlik olema. Nagu nimigi ütleb, siis sessionStorage's püsivad andmed sessiooni vältel, ehk senikaua kuni brauser lahti on. Muutuja localStorage aga püüab neid pikemat aega säilitada. Neisse muutujatesse saab parameetrite kaudu omi väärtusi tekstina talletada. Lihtsamal juhul saab lehe avamisel kontrollida, kas on talletatud väärtus võtmega punktiseis. Kui jah,siis saab seda küsida. Nagu näha, siis getItem andmete küsimiseks, sarnaselt hiljem setItem andmete salvestamiseks.

Javaskriptil andmete tekstina hoidmiseks on mugav kuju nimega JSON ehk JavaScript Object Notation. Nelja kasutaja nimed ja punktid näevad välja näiteks järgnevalt:

[{"kasutajanimi":"Sass","punkte":8},{"kasutajanimi":"Ants","punkte":9},
{"kasutajanimi":"Mari","punkte":9},{"kasutajanimi":"Juku","punkte":10}]

Andmete tekstiks ja tagasi muutmiseks sai tehtud kaks abistavat funktsiooni:

```
this.andmedTekstina=function(){
   return JSON.stringify(this.andmed);
}
this.andmedTekstist=function(tekst){
   this.andmed=JSON.parse(tekst);
       this.kuva();
}
```

Sealjuures saab mugavasti kasutada brauserisse sisse ehitatud objekti JSON käsklusi parse ning stringify. Edasi saab juba punktid talletada ja vajadusel välja küsida.

```
function salvestaPunktid(){
      localStorage.setItem("punktiseis", h1.andmedTekstina());
}
function loePunktid(){
   if(localStorage.getItem("punktiseis")){
       h1.andmedTekstist(localStorage.getItem("punktiseis"));
    }
}
```

Ning edasi juba kood tervikuna

```
<!doctype html>
<html manifest="punktid.appcache">
  <head>
     <title>Haldus</title>
         <script>
             function Tulemushaldus(kihinimi){
                   this.algus=function(){
                      this.kiht=document.getElementById(kihinimi);
                          this.andmed=new Array();
                   }
                   this.kuva=function(){
                      var t="<table><tr><th>Kasutaja</th><th>Punktid</th></tr>";
                          for(var i=0; i<this.andmed.length; i++){
                             t+="<tr><td>"+this.andmed[i].kasutajanimi+
                                   "</td><td>"+this.andmed[i].punkte+"</td></tr>
";
                          }
                          t+="</table>";
                          this.kiht.innerHTML=t;
                   }
                   this.nimesort=function(k1, k2){
                       if(k1.kasutajanimi<k2.kasutajanimi){return -1;}
                           if(k1.kasutajanimi>k2.kasutajanimi){return 1;}
                           return 0;
                   }
                   this.punktisort=function(k1, k2){return k1.punkte-k2.punkte;}
                   this.lisaTulemus=function(kasutajanimi, punkte){
                      this.andmed.push({"kasutajanimi": kasutajanimi, "punkte":
parseInt(punkte)});
                          this.andmed.sort(this.sortfunktsioon);
                          this.kuva();
                   }
```

```
                    this.andmedTekstina=function(){
                       return JSON.stringify(this.andmed);
                    }
                    this.andmedTekstist=function(tekst){
                       this.andmed=JSON.parse(tekst);
                           this.kuva();
                    }
                    this.algus();
//                  this.sortfunktsioon=this.nimesort;
                    this.sortfunktsioon=this.punktisort;
                 }
                 var h1;
                 function lehealgus(){
                    h1=new Tulemushaldus("kiht1");
                        h1.lisaTulemus("Juku", 10);
                        h1.lisaTulemus("Ants", 9);
                        h1.lisaTulemus("Sass", 8);
                 }
                 function lisaPunktid(){
                    h1.lisaTulemus(document.getElementById("nimekast").value,
                                   document.getElementById("punktikast").val
ue);
                 }
                 function salvestaPunktid(){
                        localStorage.setItem("punktiseis", h1.andmedTekstina());
                 }
                 function loePunktid(){
                    if(localStorage.getItem("punktiseis")){
                           h1.andmedTekstist(localStorage.getItem("punktiseis"))
;
                        }
                 }
          </script>
  </head>
  <body onload="lehealgus();">
     <h1>Nimeharjutused!</h1>
         <div id="kiht1"></div>
         <input type="text" id="nimekast" placeholder="kasutajanimi" />
         <input type="text" id="punktikast" placeholder="punktid" />
         <input type="button" value="Lisa" onclick="lisaPunktid()" /> <br />
         <input type="button" value="Salvesta" onclick="salvestaPunktid()" />
         <input type="button" value="Loe" onclick="loePunktid()" />
  </body>
</html>
```

Ülesandeid

- Pane näide käima.
- Lisa kolmandaks salvestatavaks tulbaks sugu
- Lisa lehele valik, millega saab näha ainult poiste andmeid, ainult tüdrukute andmeid või mõlemaid koos.

Veebiühenduseta rakendus

Veebilehel töötavat rakendust on mugav brauseris ja veebilehelt avada, kuid sugugi pidevalt ei pruugi ühendus olemas olla. Selle mure leevendamiseks on loodud appcache, mis kord avatud lehe andmed brauserisse meelde jätab, nii et lehte saab kasutada ka ühenduse puudumisel. Koos

localStorage-ga moodustavad nad hea komplekti, millega teha täiesti töölauarakendustega võrreldavaid lahendusi. Selliseks hoidmiseks tuleb apache-serveri puhul luua eraldi kirjeldusfail, kus CACHE MANIFEST-sektsioonis on failide loetelu, mis palutakse veebilehitsejal meelde jätta. Soovitatavalt on failis ka trellidega välja kommenteeritud real faili muutmisaeg - lahendus töötab nõnda, et html-fail laetakse uuesti alles siis, kui manifesti faili on muudetud.

punktid.appcache

```
CACHE MANIFEST
# 11:32
punktihaldus5.html
```

Et server teaks appcache-failidele eraldi tähelepanu pöörata, tuleb vastav rida lisada lehtede kuvamise ja tüüpide jagamise eest hoolitsevase faili .htaccess

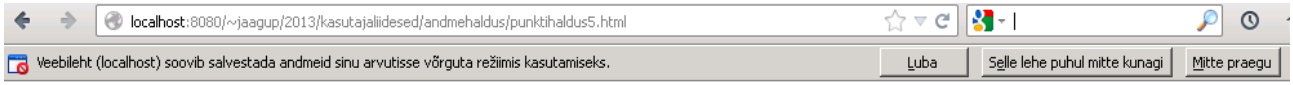
.htaccess

```
AddType text/cache-manifest .appcache
```

Esmasel avamisel küsitakse üle, et kas kasutaja ikka tahab lehte oma lehitseja puhvrisse salvestada. Kui jah, siis edasi saab juba ilma ühenduseta toimetada ning lahenduse lahti ka siis, kui parajasti pole võimalik välismaailmaga sidet pidada.

Brauseri poolt peab samuti märkima, et lehte puhverdatakse. Selleks piisab html-märgendi juures faili määramisest kus kirjas puhverdatavate lehtede loetelu

```
<html manifest="punktid.appcache">
```



Väike läbimäng ka

Avatud leht

Nimeharjutused!

Kasutaja	Punktid
Sass	8
Ants	9
Juku	10

kasutajanimi | punktid | Lisa

Salvesta | Loe

Andmete lisamine

Kasutaja Punktid

Sass 8

Ants 9

Juku 10

Mari | 9 | Lisa

Salvesta | Loe

Salvestusnupule vajutus

Kasutaja Punktid

Sass 8

Ants 9

Mari 9

Juku 10

kasutajanimi | punktid | Lisa

Salvesta | Loe

salvesta

Uue kasutaja lisamine

Kasutaja Punktid

Sass 8

Ants 9

Mari 9

Juku 10

Ants | 11 | Lisa

Salvesta | Loe

Kasutaja olemas

Nimeharjutused!

Kasutaja Punktid

Sass 8

Ants 9

Mari 9

Juku 10

Ants 11

kasutajanimi | punktid | Lisa

Salvesta | Loe

Seis pärast lugemisnupule vajutamist. Kuna Antsu ei salvestatud, siis taastati eelmise salvestuse

aegne seis.

Ülesandeid

- Pane näide tööle.
- Hoia appcache abil salvestatavana lehte, kus lisaks kasutajanimele ja punktidele on salvestatud ka sugu.
- Koosta veebileht, kus kasutaja saab ekraanile tekitada ringe. Ringide koordinaadid talletatakse localStoragesse. Leht puhverdatakse appcache abil

XMLHttpRequest

Javaskripti juures on võime veebilehele taustalt andmeid laadida või neid kasutajale nähtamatult serverisse salvestada. Esialgu kasutati seda üksikute muutuvate uudiste näitamiseks, kuid selle abil luuakse mõnikord ka terveid lehestikke, kus põhilehte vahetamata saab kasutaja palju tarvlikku tehtud. Et andmeid serverist kätte saada, peavad nad kusagil olemas olema. Selleks praeguses näites tekstifail teade.text vajaliku sisuga.

teade.txt

```
Järgmisel nädalal on koolivaheaeg!!
```

Kohene päring

Lühim moodus on paluda andmed failist küsida ning nad omale sobivasse kohta pista. Ühenduse jaoks luuaks muutuja tüübist XMLHttpRequest. Käsuga open määratakse, et millise meetodiga, kust ja millisel moel andmed küsida. GET-päring annab serverile vajadusel aadressireal andmed kaasa, siin aga kasutame seda vaid teksti lugemiseks. Teiseks parameetriks on failinimi. Kolmas parameeter false ütleb, et ühendus ei ole asünkroonne. Ehk siis praegusel juhul kui andmeid küsitakse, siis oodatakse ilusti vastus ära ja vahepeal midagi muud ei tehta. Käsklus send() paneb andmeliikluse tööle, tekstifaili sisu saab kätte loodud objekti muutujast nimega responseText. Ning praegusel juhul see kuvatakse lihtsalt lehele.

```
<!doctype html>
<html>
  <head>
```

```
    <title>Veebiühendus</title>
    <script>
       var xhr=new XMLHttpRequest();

       function loeVeebist(){
          xhr.open("GET", "teade.txt", false);
          xhr.send();
          document.getElementById("kiht1").innerHTML=xhr.responseText;
       }
    </script>
  </head>
  <body>
    <div id="kiht1"></div>
    <input type="button" value="Loe" onclick="loeVeebist()" />
  </body>
</html>
```

Kiire ühenduse ja väikeste andmete puhul on selline lähenemine mugav. Probleemiks aga, et kogu leht hangub andmete laadimise ajaks. Ning kui ühendus aeglane või muul puhul andmete lugemine rohkem aega võtab, siis see tekitab tüütu seisaku.

Ülesandeid

- Pane näide tööle
- Pane lehele kaks nuppu. Igale vajutades saab kätte vastavas failis oleva uudise
- Lisa kolmas nupp. See küsib andmeid PHP lehelt, mis näitab kellaaega

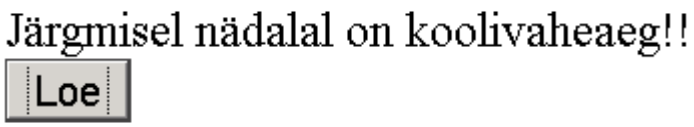
Asünkroonne päring

Hangumise vältimiseks soovitatakse väärtuste küsimiseks üldjuhul asünkroonset päringut. See tähendab, et käsu käivitamisel antakse XMLHttpRequestile sooviavaldus. Ning alles siis, kui andmestik kohale saabunud, käivitub koos sellega funktsioon, kus nendega midagi tegema hakata. Loodud objekti välja onreadystatechange väärtuseks antakse funktsioon mil nimeks praegu andmedSaabusid ning mille ülesandeks saabuvad andmed samuti välja näidata. Välja readyState järgi saab kindlaks teha, milline teade saabus - kas ühenduse loomise, ebaõnnestumise või andmete kohale jõudmise kohta. Viimase koodiks on neli ning seejärel võib saabunud andmetega toimetama hakata.

```
<!doctype html>
<html>
  <head>
    <title>Veebiühendus</title>
    <script>
       var xhr=new XMLHttpRequest();
       xhr.onreadystatechange=andmedSaabusid;

       function loeVeebist(){
           //meetod GET, fail teade.txt, asünkroonne (tagaplaanil)=jah
          xhr.open("GET", "teade.txt", true);
          xhr.send();
       }

       function andmedSaabusid(){
          if(xhr.readyState==4){  //staadium 4, andmed kohal
             document.getElementById("kiht1").innerHTML=xhr.responseText;
```

```
          }
        }
    </script>
  </head>
  <body>
    <div id="kiht1"></div>
    <input type="button" value="Loe" onclick="loeVeebist()" />
  </body>
</html>
```

Tulemus:



Tervitamine serverist

Järgmises näites käivitatakse serveris PHP-fail, mis tervitab eesnime-parameetriga kaasa saadetud nimega tegelast. Andmete saatmiseks GET-meetodiga tuleb vajalikud parameetrid ja väärtused kirjutada avatava failinime lõpu pandud küsimärgi järgi. Et täpi- ja muude salapäraste tähtedega nimed samuti edukalt serverisse kohale jõuaksid, selleks aitab teksti sobivaks sättida javaskripti käsklus encodeURI.

```
<!doctype html>
<html>
  <head>
    <title>Veebiühendus</title>
    <script>
       var xhr=new XMLHttpRequest();
       xhr.onreadystatechange=andmedSaabusid;

       function loeVeebist(){
          xhr.open("GET",
             "tervitus.php?eesnimi="+
                encodeURI(document.getElementById("eesnimi").value),
             true);
          xhr.send();
       }

       function andmedSaabusid(){
          if(xhr.readyState==4){
             document.getElementById("kiht1").innerHTML=xhr.responseText;
          }
       }
    </script>
  </head>
  <body>
    <div id="kiht1"></div>
    Palun eesnimi:
    <input type="text" id="eesnimi" />
    <input type="button" value="Tervita" onclick="loeVeebist()" />
  </body>
</html>
```

tervitus.php

```
<?php
  echo "Tere, $_REQUEST[eesnimi]!";
```

Tere, Juku!
Palun eesnimi: Juku Tervita

Post-meetodiga saatmine

GET-käsklust kasutatakse põhiliselt otsingute juures - sellisel juhul hea, kui teele saadetud otsisõnaga päringule juba esimesest puhverserverist vastuse saab. Kui aga soovitakse, et andmed serverisse kindlalt kohale jõuaksid, nende maht võiks suurem olla või ei taheta, et nad logisse salvestuksid (nagu paroolide juures mõistlik), siis sobib päringumeetodiks pigem POST. See tuleb määrata ühenduse avamise ajal. Samuti tuleb sobiva saatmistüübi määramiseks lisada rida

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

ning kaasapandud andmed ei lähe enam failinime järgi nagu ennist, vaid send-käskluse parameetrina. Muu endiselt sarnane.

```
<!doctype html>
<html>
  <head>
    <title>Veebiühendus</title>
    <script>
       var xhr=new XMLHttpRequest();
       xhr.onreadystatechange=andmedSaabusid;

       function loeVeebist(){
          xhr.open("POST", "tervitus.php", true);
          xhr.setRequestHeader(
              "Content-type","application/x-www-form-urlencoded");
          xhr.send(
            "eesnimi="+encodeURI(document.getElementById("eesnimi").value));
       }

       function andmedSaabusid(){
          if(xhr.readyState==4){
             document.getElementById("kiht1").innerHTML=xhr.responseText;
          }
       }
    </script>
  </head>
  <body>
    <div id="kiht1"></div>
    Palun eesnimi:
    <input type="text" id="eesnimi" />
    <input type="button" value="Tervita" onclick="loeVeebist()" />
  </body>
</html>
```

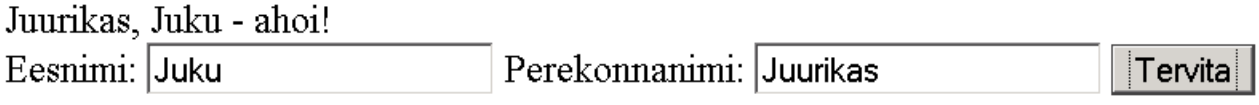
Mitu parameetrit päringus

Eesnime ja perekonnanime mõlema saatmiseks tuleb nad aadressireal &-märgiga eraldada. Mõlemad ilusti ära kodeerida ning jõuavadki kohale. PHP peab muidugi ka mõlema vastuvõtuks valmis olema, et neile reageerida.

```
<!doctype html>
<html>
  <head>
    <title>Veebiühendus</title>
    <script>
       var xhr=new XMLHttpRequest();
       xhr.onreadystatechange=andmedSaabusid;

       function loeVeebist(){
          xhr.open("POST", "tervitus2.php", true);
          xhr.setRequestHeader(
              "Content-type","application/x-www-form-urlencoded");
          xhr.send(
            "eesnimi="+encodeURI(document.getElementById("eesnimi").value)+
             "&perekonnanimi="+
               encodeURI(document.getElementById("perekonnanimi").value));
       }

       function andmedSaabusid(){
          if(xhr.readyState==4){
             document.getElementById("kiht1").innerHTML=xhr.responseText;
          }
       }
    </script>
  </head>
  <body>
    <div id="kiht1"></div>
    Eesnimi:
    <input type="text" id="eesnimi" />
    Perekonnanimi:
    <input type="text" id="perekonnanimi" />
    <input type="button" value="Tervita" onclick="loeVeebist()" />
  </body>
</html>
```

tervitus2.php

```
<?php
  echo "$_REQUEST[perekonnanimi], $_REQUEST[eesnimi] - ahoi!";
```



Ülesandeid

- Pane näited tööle
- Pane leht iga sekundi tagant serverist kella küsima ja veebi näitama
- Serveris on uudised nummerdatud. Kasutaja valib rippmenüüst uudise numbri, selle peale näidatakse talle serverist vastavat uudist

JSON

Algselt Javaskripti andmete hoidmiseks ja ülekandmiseks tekkinud JavaScript Object Notation-vorming on mujalgi populaarseks saanud, PHP juures tema tarbeks eraldi käsklused olemas. Lihtsamaks näiteks massiiv kahe kasutaja ja nende punktidega. PHP käsklus json_encode väljastab tulemused tekstina

```
<?php
  $kasutajad=array();
  $kasutajad["juku"]=7;
  $kasutajad["kati"]=8;
  echo json_encode($kasutajad);
```

Väljund:

```
{"juku":7,"kati":8}
```

Andmeid põhjalikumalt struktueerida tahtes hoitakse tulemusi enamasti kirjete/objektidena. Kirjutamist natuke rohkem, kuid tulemus paindlikum. Kui peaks vaja olema rohkem kui kahe tunnuse hoidmist kirjes, siis saab selle hõlpsasti lisada.

```
<?php
  $kasutajad=array();
  $k=new stdClass();
  $k->kasutajanimi="juku";
  $k->punkte=7;
  $kasutajad[0]=$k;
  $k=new stdClass();
  $k->kasutajanimi="kati";
  $k->punkte=8;
  $kasutajad[1]=$k;
  echo json_encode($kasutajad);
```

Väljund

Väljaspool kõige ümber siis kandilised massiivisulud, edasi iga kirje ümber loogelised sulud.

```
[{"kasutajanimi":"juku","punkte":7},{"kasutajanimi":"kati","punkte":8}]
```

Väljund Javaskripti

JavaScript Object Notationi nimi sellest tulebki, et tulemusi saab vabalt javaskripti muutujaks panna. Sellega saab hakkama rida

```
        var kd=<?php echo json_encode($kasutajad); ?>;
```

Ehk siis eelnevalt nähtud kasutajate loetelu jõuab ühe käsu abil kliendipoolsesse muutujasse. Edasi

näitekood tervikuna

```
<?php
  $kasutajad=array();
  $k=new stdClass();
  $k->kasutajanimi="juku";
  $k->punkte=7;
  $kasutajad[0]=$k;
  $k=new stdClass();
  $k->kasutajanimi="kati";
  $k->punkte=8;
  $kasutajad[1]=$k;
?>
<!doctype html>
<html>
  <head>
    <title>Andmete lugemine</title>
      <script>
        var kd=<?php echo json_encode($kasutajad); ?>;
        function kuvaKasutajad(kihinimi){
          var t="<table><tr><th>Kasutajanimi</th><th>Tulemus</th></tr>";
            for(var i=0; i<kd.length; i++){
               t+="<tr><td>"+kd[i].kasutajanimi+"</td>"+
                 "<td>"+kd[i].punkte+"</td></tr>\n";
            }
            t+="</table>";
            document.getElementById(kihinimi).innerHTML=t;
        }
      </script>
  </head>
  <body onload="kuvaKasutajad('kiht1')">
    <div id="kiht1"></div>
  </body>
</html>
```

Tulemuseks javaskripti ja HTMLi abil kujundatud tabel

Kasutajanimi	Tulemus
juku	7
kati	8

Ülesandeid

- Pane näited tööle
- Koosta PHPs massiiv kataloogis olevate failinimedega (käsklus scandir). Moodusta neist failinimedest JSON-kujul tekst.
- Loe failinimed javaskriptis sisse, moodusta nendest veebilehel loetelu.
- Failinimele vajutamisel küsitakse faili sisu XMLHttpRequesti abil ning näidatakse lehel.

Andmed SQL-tabelis

Järgnevalt veidi pikem näide, kuidas mängurakenduse punkte ja kasutajanimesid serveris asuvas andmetabelis hallata.

Andmete hoidmiseks kahe tulbaga tabel - üks kasutajanime, teine punktide jaoks

```
CREATE TABLE punktihaldus(
  knimi VARCHAR(30) PRIMARY KEY,
  punktidearv INT
);
```

Mõned andmed ka sisse.

```
INSERT INTO punktihaldus VALUES ('malle', 13);
INSERT INTO punktihaldus VALUES ('kalle', 15);
```

PHP-fail andmete küsimiseks andmetabelist ning väljastmiseks JSONina

```
<?php
  $yhendus=new mysqli("localhost", "juku", "kala", "jukubaas");
  function kysiKasutajad(){
    global $yhendus;
      $kask=$yhendus->prepare(
         "SELECT knimi, punktidearv FROM punktihaldus ORDER BY punktidearv");
      $kask->bind_result($knimi, $punktidearv);
      $kask->execute();
      $hoidla=array();
      while($kask->fetch()){
         $k=new stdClass();
         $k->kasutajanimi=$knimi;
         $k->punkte=$punktidearv;
         array_push($hoidla, $k);
      }
      return $hoidla;
  }

  echo json_encode(kysiKasutajad());
```

Tulemus:

```
[{"kasutajanimi":"malle","punkte":13},{"kasutajanimi":"kalle","punkte":15}]
```

Testimise tulemuse järgi saab kontrollida, et andmed tulevad baasist välja.

Edasi näite edasiarendus. Juurde funktsioon uuendaKasutaja, millele antakse kasutajanimi ning uus punktide arv. Kui kasutaja puudub baasist, siis temanimeline rida lisatakse. Kui kasutaja olemas, siis juhul kui uus punktide arv on tal eelmisest suurem, siis tulemus asendatatakse, muul juhul ei tehta midagi. Kasutajate uuendamiseks antakse ette JSON-vormingus tekst kasutajanimede ja punktide kohta. Käsu json_decode abil eraldatakse sealt üksikute kasutajate andmed ning siis igaühega pannakse uuenduskäsklus eraldi käima.

```
<?php
  $yhendus=new mysqli("localhost", "juku", "kala", "jukubaas");
  function kysiKasutajad(){
    global $yhendus;
      $kask=$yhendus->prepare(
           "SELECT knimi, punktidearv FROM punktihaldus ORDER BY punktidearv");
      $kask->bind_result($knimi, $punktidearv);
      $kask->execute();
      $hoidla=array();
      while($kask->fetch()){
         $k=new stdClass();
```

```
        $k->kasutajanimi=$knimi;
        $k->punkte=$punktidearv;
        array_push($hoidla, $k);
     }
     return $hoidla;
  }

  function uuendaKasutaja($knimi, $punktidearv){
     global $yhendus;
       $kask=$yhendus->prepare(
            "SELECT punktidearv FROM punktihaldus WHERE knimi=?");
       $kask->bind_param("s", $knimi);
       $kask->bind_result($p);
       $kask->execute();
       if($kask->fetch()){
          $kask->close();
          if($punktidearv>$p){
               $kask=$yhendus->prepare(
                      "UPDATE punktihaldus SET punktidearv=? WHERE knimi=?");
               $kask->bind_param("is", $punktidearv, $knimi);
               $kask->execute();
            }
       } else {
          $kask=$yhendus->prepare("INSERT INTO punktihaldus VALUES(?, ?)");
            $kask->bind_param("ss", $knimi, $punktidearv);
            $kask->execute();
       }
  }

  function uuendaKasutajad($jsonstr){
     $m=json_decode($jsonstr);
       foreach($m as $k){
         uuendaKasutaja($k->kasutajanimi, $k->punkte);
       }
  }
  uuendaKasutajad(
     '[{"kasutajanimi":"palle","punkte":14},
       {"kasutajanimi":"kalle","punkte":15}]');
  echo json_encode(kysiKasutajad());
```

Nõnda saab pärast andmete uuenduskäsklust serveris olevat uut tulemust vaadata.

Ülesandeid

- Pane näide tööle
- Katseta mitmesuguste kasutajate ja punktiarvudega, jälgi tulemust
- Koosta veebileht, kust saab tarvliku JSON-stringi sisestada, hoolitse, et server selle kätt saaks ja tulemusi arvestaks.

Ajaxi abil lugemine

Punktide haldamiseks nüüd eraldi kliendipoolne rakendus koos eraldiseisva objektiga. Kui andmed leiab localStoragest, kasutab neid, eraldi võimalik ka serverist tulemusi küsida.

```
<!doctype html>
<html>
  <head>
     <title>Haldus</title>
```

```
<script>
    function Tulemushaldus(kihinimi){
          this.algus=function(){
             this.kiht=document.getElementById(kihinimi);
                 this.andmed=new Array();
          }
          this.kuva=function(){
             var t="<table><tr><th>Kasutaja</th><th>Punktid</th></tr>";
                 for(var i=0; i<this.andmed.length; i++){
                    t+="<tr><td>"+this.andmed[i].kasutajanimi+
                     "</td><td>"+this.andmed[i].punkte+"</td></tr>";
                 }
                 t+="</table>";
                 this.kiht.innerHTML=t;
          }
          this.nimesort=function(k1, k2){
              if(k1.kasutajanimi<k2.kasutajanimi){return -1;}
                  if(k1.kasutajanimi>k2.kasutajanimi){return 1;}
                  return 0;
          }
          this.punktisort=function(k1, k2){return k1.punkte-k2.punkte;}
          this.lisaTulemus=function(kasutajanimi, punkte){
             this.andmed.push(
                {"kasutajanimi": kasutajanimi,
                   "punkte": parseInt(punkte)});
                 this.andmed.sort(this.sortfunktsioon);
                 this.kuva();
          }
          this.andmedTekstina=function(){
             return JSON.stringify(this.andmed);
          }
          this.andmedTekstist=function(tekst){
             this.andmed=JSON.parse(tekst);
                 this.kuva();
          }
          this.algus();
//        this.sortfunktsioon=this.nimesort;
          this.sortfunktsioon=this.punktisort;
       }
       var h1;
       function lehealgus(){
          h1=new Tulemushaldus("kiht1");
              h1.lisaTulemus("Juku", 10);
              h1.lisaTulemus("Ants", 9);
              h1.lisaTulemus("Sass", 8);
       }
       function lisaPunktid(){
          h1.lisaTulemus(document.getElementById("nimekast").value,
                       document.getElementById("punktikast").value);
       }
       function salvestaPunktid(){
              localStorage.setItem("punktiseis", h1.andmedTekstina());
       }
       function loePunktid(){
          if(localStorage.getItem("punktiseis")){
              h1.andmedTekstist(localStorage.getItem("punktiseis"));
          }
       }
       var serveriyhendus=new XMLHttpRequest();
       serveriyhendus.onreadystatechange=andmedServerist;
       function loeServerist(){
          serveriyhendus.open("GET", "json4.php", true);
              serveriyhendus.send(true);
       }
```

```
                function andmedServerist(){
                  if(serveriyhendus.readyState==4){
                     alert(serveriyhendus.responseText)
                        h1.andmedTekstist(serveriyhendus.responseText);
                  }
                }
        </script>
  </head>
  <body onload="lehealgus();">
     <h1>Nimeharjutused</h1>
        <div id="kiht1"></div>
        <input type="text" id="nimekast" placeholder="kasutajanimi" />
        <input type="text" id="punktikast" placeholder="punktid" />
        <input type="button" value="Lisa" onclick="lisaPunktid()" /> <br />
        <input type="button" value="Salvesta" onclick="salvestaPunktid()" />
        <input type="button" value="Loe" onclick="loePunktid()" />
        <input type="button" value="Loe serverist" onclick="loeServerist()" />
  </body>
</html>
```

Ülesandeid

- Pane näide tööle
- Pane lehele samaaegselt tööle kaks punktiarvestusobjekti - üks poiste, teine tüdrukute tarbeks
- Koosta ka andmebaasipool nõnda, et kummagi andmeid arvestatakse eraldi

AJAX ka salvestamiseks

PHP poolde juurde täiendus, et kui saabuvad ühe kasutaja andmed, siis vastav kasutaja kas lisatakse baasitabelisse või uuendatakse ta punktiseisu juhul, kui uusi punkte piisavalt palju oli.

edetabel4.php

```
<?php
  $yhendus=new mysqli("localhost", "juku", "kala", "jukubaas");
  function kysiKasutajad(){
     global $yhendus;
     $kask=$yhendus->prepare("SELECT knimi, punktidearv FROM punktihaldus
              ORDER BY punktidearv");
     $kask->bind_result($knimi, $punktidearv);
     $kask->execute();
     $hoidla=array();
     while($kask->fetch()){
       $k=new stdClass();
       $k->kasutajanimi=$knimi;
       $k->punkte=$punktidearv;
       array_push($hoidla, $k);
     }
     return $hoidla;
  }

  function uuendaKasutaja($knimi, $punktidearv){
     global $yhendus;
     $kask=$yhendus->prepare(
             "SELECT punktidearv FROM punktihaldus WHERE knimi=?");
     $kask->bind_param("s", $knimi);
     $kask->bind_result($p); //vana punktide arv
     $kask->execute();
     if($kask->fetch()){
```

```
        $kask->close();
        if($punktidearv>$p){
           $kask=$yhendus->prepare(
              "UPDATE punktihaldus SET punktidearv=? WHERE knimi=?");
           $kask->bind_param("is", $punktidearv, $knimi);
           $kask->execute();
        }
     } else {
       $kask=$yhendus->prepare("INSERT INTO punktihaldus VALUES(?, ?)");
       $kask->bind_param("si", $knimi, $punktidearv);
       $kask->execute();
     }
  }

  if(isSet($_REQUEST["knimi"])){
     uuendaKasutaja($_REQUEST["knimi"], $_REQUEST["punktidearv"]);
  }
  echo json_encode(kysiKasutajad());
```

Javaskriptile juurde täiendus, kus sisestatud andmed saadetakse taustal serverisse, nii et need saab PHP lisada andmebaasi. Samas käivitatakse setInterval'i abil loeVeebist käsklust viie sekundi tagant, et kui peaks mõne teise kliendi kaudu serveris andmeid uuendatama, siis on varsti ka siin uus tulemus näha.

kuvamine4.html

```
<!doctype html>
<html>
  <head>
    <title>Andmete kuvamine</title>
    <script>
       var xhr=new XMLHttpRequest();
       xhr.onreadystatechange=andmedSaabusid;

       function loeVeebist(){
          xhr.open("GET", "edetabel4.php", true);
          xhr.send();
       }

       function andmedSaabusid(){
          if(xhr.readyState==4){
             kd=JSON.parse(xhr.responseText);
             kuvaKasutajad();
          }
       }

       function saadaTulemus(){
          xhr.open("POST", "edetabel4.php", true);
          xhr.setRequestHeader(
             "Content-type","application/x-www-form-urlencoded");
          xhr.send("knimi="+
                encodeURI(document.getElementById("txtknimi").value)+
                  "&punktidearv="+
                     encodeURI(document.getElementById("txtpunkte").value));
       }

      var kd={};
      function kuvaKasutajad(){
         var t="<table><tr><th>Kasutajanimi</th><th>Tulemus</th></tr>";
         for(var i=0; i<kd.length; i++){
            t+="<tr><td>"+kd[i].kasutajanimi+"</td>"+
```

```
                "<td>"+kd[i].punkte+"</td></tr>";
          }
          t+="</table>";
          document.getElementById("kiht1").innerHTML=t;
        }
      </script>
    </head>
    <body onload="loeVeebist();setInterval('loeVeebist()', 5000)">
      <div id="kiht1"></div>
      Kasutajanimi: <input type="text" id="txtknimi" />
      Punkte: <input type="text" id="txtpunkte" />
      <input type="button" value="Saada" onclick="saadaTulemus()" />
      <input type="button" value="Loe" onclick="loeVeebist()" />
    </body>
</html>
```

Ülesandeid

- Pane näited käima
- Vaata toimimist mitmest aknast ning veendu, et ühes tehtud uuendused kajastuvad ka teises.
- Koosta rakendus kasutaja reageerimisaja mõõtmiseks. Mida kiirem tulemus, seda parem. Lehele sisenedes sisestab kasutaja oma kasutajanime. Ta peab vajutama lehele ilmunud kujundile. Aeg salvestatakse baasi. Samaaegselt on näha ka teiste kasutajate parimad ajad. Serveri poolt lisa võimalus tabeli tühjendamiseks uue võistluse tarbeks.

Üldisi ülesandeid

Kordamisküsimused

- Käsklused, funktsioonid ja objektid. Programmi abstraktsioonitaseme tõstmise head küljed ning puudused.
- Objektide kasutusnäiteid veebilehel töötavate komponentide juures.
- Seosed objektide vahel, objektistruktuuri näiteid: kujundid lehel, jalgpall arvutis, kaardimängurakendus
- Objekti prototüüp. Sarnaste oskuste kasutamine mitme objekti juures
- Arvutused nurkade ja ringjoone juures. Liikumine mööda ringjoont, pööramine.
- Kolmemõõtmeline graafika arvutiekraanil. Asukohtade projitseerimine ekraanil, vastavad arvutused
- AJAXi võimalused veebilehtede loomisel. Sünkroonne ja asünkroonne päring. Tegevuste tulemuste salvestamine serveris.
- Andmete salvestamine kliendi arvutis, localStorage, sessionStorage ning appcache. Nende võimaluste kasutamine näitrakenduste juures.

Kolme tasemega ülesanded

Pallide tüübid

* Loo palli tüüp, mil on raadius, asukoht ja liikumissuund ning liikumisarvutuste käsklus. Tekita

paar palli ekraanile, lase neil oma andmete järgi liikuda.
* Loo pallile prototüübi abil alamtüüp, mis liikudes kukub raskusjõu abil allapoole. Pane mõlemat tüüpi pallid läbisegi liikuma.
* Pallidel on alaserv, millest allapoole ei kukuta. Kui üks pall satub kokku teise palliga, siis teine neist hävib ning esimene muutub suuremaks.

Hulknurgad

* Koosta tüüp ruudu jaoks, parameetrina saab ette anda tippude kauguse keskkohast. Joonistamiseks võib kasutada tavalise ristküliku käsklust. Katseta.
* Loo alamtüüp sümmeetrilise hulknurga loomiseks. Ette saab anda nurkade arvu ning kauguse keskkohast. Katseta eksemplare läbisegi.
* Võrreldes eelmisega ei pruugi hulknurk olla sümmeetriline. Tippude nurgad ning kaugused keskkohast antakse loomisel ette. Kujundi nurkade kaugusi keskusest on võimalik muuta. Samuti saab hulknurki kloonida ja ekraanil nihutada.

Jalgratas

* Koosta komponent pöörleva ratta tarbeks, katseta.
* Sama komponendi abil loo kaks suurt ratast ning suur hammasratas.
* Võimalda hiirega suurt hammasratast pöörata. Ühes sellega pöörlevad ka suured rattad, ainult et kaks ja pool korda kiiremini.

Lift

· Lift sõidab üheksakorruselises majas ülevalt alla.
· Liftil on numbrid ühest üheksani ning igal korrusel on kutsenupp. Lift sõidab vajutatud korrusele.
· Lisaks eelmisele on trepikojas lifte kaks. Kutse peale sõidab kohale lähim vaba lift.

Mängukaardid

· Osaliselt üksteise peale joonistatakse kolm juhuslikku mängukaarti.
· Kaarte saab üksteise peale hiirega vedada.
· Kaardid on segatuna laiali laua peal. Sealt saab välja lohistada kaks kaarti ning need ringi pöörata. Kui numbrid on ühesugused, võetakse see paar välja ning leidjale lisatakse punkt.

Automaatsalvestus veebilehel

* Trükitav tekst salvestatakse automaatselt localStorage abil ning näidatakse uuel sisenemisel.
* localStorages salvestatud tekst salvestatakse iga paarikümne sekundi tagant ka serverisse. Uues kohas lehte avades näidatakse serverist tulnud sisu.
* Kui lehe avamisel erinevad serveris olev sisu ning localStorages olev sisu, siis näidatakse kasutajale mõlemat ning küsitakse, et kummaga edasi töötada.

Kolmnurga joonistamine

* Veebilehele joonistatakse kolmnurk. Kolmnurga ühe külje pikkust saab kasutaja muuta.
* Kolmnurga kõigi kolme külje pikkust saab kasutaja eraldi muuta.
* Selliseid muudetava küljepikkusega kolmnurki on lehel kasutaja soovitud arv.

Graafika salvestus

* Loo vahend veebilehel ruutude joonistamiseks ja paigutamiseks.
* Joonistatud andmed saab salvestada serverisse.
* Olemasolevate andmete põhjal saab pildi ekraanil taastada ning seda muuta ja täiendada.

Tähemärkidega kujundus

* Koosta Javaskripti abil klass, mille eksemplarile saab anda ette tähemärgi ning siis selle eksemplari käest soovitud pikkusega selle tähemärgiga jadasid küsida.
* Lisa klassile käsklus, kus käsklusele etteantud tekst ümbritsetakse klassile etteantud tähemärkidest moodustatud ristkülikuga.
* Statistikana peetakse klassis eraldi meeles, millist käsklust ja millal käivitati. Andmeid saab välja küsida.

Asukohtadega jututuba

* Kasutajad saavad jututoas vestelda. Hiirega ekraanile vajutamisel saadetakse edasi ka kasutaja koordinaadid.
* Pildil on näha vestlejate asukohad. Igaüks saab enese nime asukohta hiirega muuta.
* Võrreldes eelmisega kostavad vaid nende kasutajate teated, kes on kuulajast pildil vähem kui 200 ekraanipunkti kaugusel.

Kujundid

* Koosta objekt asukoha (x, y) andmete hoidmiseks. Koosta asukohtadest massiiv. Trüki massiivi andmed ekraanile.
* Koosta objekt, mille sees üheks väljaks on asukohtade massiiv. Lisa objektile käsklus nende andmete järgi joonistamiseks. Vastavalt objekti küljes olevale parameetrile joonistatakse tulemus kas täppidena, ühendatud joontena või seest täidetud alana.
* Koosta objekt, mille sees üheks väljaks on eelnevas punktis kirjeldatud kujundite massiiv. Lisa käsklused kujundite ükshaaval lisamiseks, eemaldamiseks ning komplekti tervikuna ekraanile joonistamiseks.

Hammasrattad

• Joonistatakse kasutaja määratud hammaste arvuga hammasratas.
• Selliseid hammasrattaid joonistatakse kaks ning pannakse üksteisesse hambunult pöörlema.
• Kasutaja annab ette mõlema hambunult pöörleva ratta hammaste arvu.

Valdade valimine

* Koosta massiiv maakondade nimedega. Loo selle massiivi põhjal lehele rippmenüü.
* Lisa lehele andmed igas maakonnas olevate valdade nimedega. Vastavalt maakonna valimisele kuvatakse teises rippmenüüs selle maakonna vallad.
* Maakondade ja valdade tabelid on andmebaasis. Loo vorm ettevõtmisel osalejate andmete sisestamiseks. Igaüks sisestab oma ees- ja perekonnanime, elektronpostiaadressi ning valib maakonnavaliku järgi Javaskripti abil ette tulevast vallast omale sobiva. Osalejate andmed talletatakse serverisse.

Tähtede püüdmine

* Pane ühe sõna tähed ükshaaval ülevalt alla kukkuma.
* Sõnade loetelu on serveris andmebaasis. Sealt valitakse juhuslik sõna ning pannakse lehel tähtede kaupa kukkuma. Kasutaja saab tähti hiirega püüda. Kui kõik tähed käes, näidatakse, et sõna püütud.
* Võrreldes eelmisega kukub tähtede vahel tärne, mida ei tohi püüda. Serveris peetakse arvestust, et millist sõna mitu korda pakutud ning mitu korda kätte saadud.

Lahendusi ja täiendusi

Objektid

Ringi pindala arvutaja objektina

```
<!doctype html>
<html>
  <head>
    <title>Kalkulaator</title>
        <script>
          function Arvutaja(kihinimi){
                 this.algus=function(){
                   this.kiht=document.getElementById(kihinimi);
                   this.kiht.innerHTML=
                    "Raadius: <input type='text'   id='kast1' /> "+
                    "<input type='button' value='Ringi pindala' "+
                                          "onClick='a1.arvuta();' /> "+
                    "<div id='vastus'></div>";
                    this.kast=document.getElementById("kast1");
                    this.vastusekiht=document.getElementById("vastus");
                 }
                 this.arvuta=function(){
```

```
                    var r=parseFloat(this.kast.value);
                    this.vastusekiht.innerHTML=3.14*r*r;
                  }
                  this.algus();
            }

            var a1;
            function lehealgus(){
              a1=new Arvutaja("kiht1");
            }
          </script>
  </head>
  <body onload="lehealgus();">
     <h1>Arvutamine</h1>
     <div id="kiht1"></div>

  </body>
</html>
```

Arvutamine

Raadius: 10 Ringi pindala

314

Kaks pindala arvutajat

```
<!doctype html>
<html>
  <head>
    <title>Kalkulaator</title>
        <script>
          function Arvutaja(kihinimi){
             this.algus=function(){
                   this.kiht=document.getElementById(kihinimi);
                   window[kihinimi+"_kalkulaator"]=this;
                   this.kiht.innerHTML=
                    "Raadius: <input type='text'  id='"+kihinimi+"_kast1' /> "+
                    "<input type='button' value='Ringi pindala' "+
                     "onClick='"+kihinimi+"_kalkulaator.arvuta();' /> "+
                "<div id='"+kihinimi+"_vastus'></div>";
               this.kast=document.getElementById(kihinimi+"_kast1");
                   this.vastusekiht=document.getElementById(kihinimi+"_vastus");
                 }
                 this.arvuta=function(){
                    var r=parseFloat(this.kast.value);
                    this.vastusekiht.innerHTML=3.14*r*r;
                 }
                 this.algus();
          }


          function lehealgus(){
```

```
            new Arvutaja("kiht1");
            new Arvutaja("kiht2");
          }
        </script>
  </head>
  <body onload="lehealgus();">
     <h1>Arvutamine</h1>
         <div id="kiht1"></div>
         <div id="kiht2"></div>
  </body>
</html>
```

Arvutamine

Raadius: 4 Ringi pindala

50.24

Raadius: 6 Ringi pindala

113.03999999999999

Seadistatav kalkulaator

```
<html>
  <head>
    <title>Kalkulaator</title>
        <script>
          function Arvutaja(kihinimi, kastitekst, nuputekst, koefitsient){
             this.algus=function(){
                   this.kiht=document.getElementById(kihinimi);
                   window[kihinimi+"_kalkulaator"]=this;
                   this.kiht.innerHTML=
                    kastitekst+": <input type='text'   id='"+kihinimi+"_kast1' />
> "+
                "<input type='button' value='"+nuputekst+"'
onClick='"+kihinimi+"_kalkulaator.arvuta();' /> "+
                "<span id='"+kihinimi+"_vastus'></span>";
               this.kast=document.getElementById(kihinimi+"_kast1");
                   this.vastusekiht=document.getElementById(kihinimi+"_vastus");
                   this.koefitsient=koefitsient;
                 }
                 this.arvuta=function(){
                    this.vastusekiht.innerHTML=
                      (parseFloat(this.kast.value)*this.koefitsient).toFixed(2);
                 }
                 this.algus();
          }


          function lehealgus(){
            new Arvutaja("kiht1", "Eurod", "Dollariteks", 1.3);
```

```
            new Arvutaja("kiht2", "Tollid", "Sentimeetriteks", 2.54);
            new Arvutaja("kiht3", "Letihind", "Käibemaks", 0.2/1.20);

          }
        </script>
        <meta charset="UTF-8" />
  </head>
  <body onload="lehealgus();">
     <h1>Arvutamine</h1>
         <div id="kiht1"></div>
         <div id="kiht2"></div>
         <div id="kiht3"></div>
  </body>
</html>
```

Arvutamine

Eurod: 10 Dollariteks 13.00

Tollid: 5 Sentimeetriteks 12.70

Letihind: 4 Käibemaks 0.67

Ladu

- Lisa objekti kujundusse teine tekstiväli, kus saab määrata, millise koguse võrra olemasolevat väärtust kasvatatakse või kahandatakse

```
<!doctype html>
<html>
  <head>
    <title>Kalkulaator</title>
        <script>
          function Laohaldus(kihinimi, kogus){
             this.algus=function(){
                   this.kiht=document.getElementById(kihinimi);
                   this.kogus=kogus;
                   window[kihinimi+"_ladu"]=this;
                   this.kiht.innerHTML=
                "<input type='button' value='&lt;' onClick='"+
                    kihinimi+"_ladu.v2iksemaks();' /> "+
                "<input type='text' id='"+kihinimi+
                    "_vastus' style='width: 50px' disabled />"+
                "<input type='number' id='"+kihinimi+
                    "_muutus' style='width: 50px' value='1' />"+
                "<input type='button' value='&gt;' onClick='"+kihinimi+
                    "_ladu.suuremaks();' /> ";
                   this.vastusekiht=document.getElementById(kihinimi+"_vastus");
                   this.muutusekast=document.getElementById(kihinimi+"_muutus");
                   this.kuva();
              }
```

```
                this.kuva=function(){
                   this.vastusekiht.value=this.kogus;
                }
                this.v2iksemaks=function(){
                   this.kogus-=parseInt(this.muutusekast.value);
                   this.kuva();
                }
                this.suuremaks=function(){
                   this.kogus+=parseInt(this.muutusekast.value);
                   this.kuva();
                }
                this.algus();
          }


          function lehealgus(){
            new Laohaldus("kiht1", 100);
            new Laohaldus("kiht2", 50);
          }
        </script>
  </head>
  <body onload="lehealgus();">
     <h1>Arvutamine</h1>
         <div id="kiht1"></div>
         <div id="kiht2"></div>
  </body>
</html>
```

Arvutamine

< 104 1 >

< 50 1 >

- Tekstivälja asemel saab muudetava suuruse valida rippmenüüst

```
<!doctype html>
<html>
  <head>
    <title>Kalkulaator</title>
        <script>
          function Laohaldus(kihinimi, kogus, muutused){
             this.algus=function(){
                   this.kiht=document.getElementById(kihinimi);
                   this.kogus=kogus;
                   this.muutused=muutused;
                   window[kihinimi+"_ladu"]=this;
                   var rippmenyy="<select id='"+kihinimi+"_valik'>";
                   for(var i=0; i<muutused.length; i++){
                      rippmenyy+="<option>"+muutused[i]+"</option>";
                   }
                   rippmenyy+="</select>";
                   this.kiht.innerHTML=
                "<input type='button' value='&lt;' onClick='"+
kihinimi+"_ladu.v2iksemaks();' /> "+
```

```
                "<input type='text' id='"+kihinimi+"_vastus' style='width: 50px'
disabled />"+
                rippmenyy+
                "<input type='button' value='&gt;'
onClick='"+kihinimi+"_ladu.suuremaks();' /> ";
                   this.vastusekiht=document.getElementById(kihinimi+"_vastus");
                   this.muutusevalik=document.getElementById(kihinimi+"_valik");
                   this.kuva();
                  }
                  this.kuva=function(){
                     this.vastusekiht.value=this.kogus;
                  }
                  this.v2iksemaks=function(){
                     this.kogus-=this.muutused[this.muutusevalik.selectedIndex];
                     this.kuva();
                  }
                  this.suuremaks=function(){
                     this.kogus+=this.muutused[this.muutusevalik.selectedIndex];
                     this.kuva();
                  }
                  this.algus();
           }


           function lehealgus(){
             new Laohaldus("kiht1", 100, [1, 5, 10, 30, 100]);
             new Laohaldus("kiht2", 50, [1, 2, 3]);
           }
         </script>
  </head>
  <body onload="lehealgus();">
     <h1>Arvutamine</h1>
         <div id="kiht1"></div>
         <div id="kiht2"></div>
  </body>
</html>
```

Arvutamine

< 130 10 >

< 50 1 >

- Lisa nupp laoseisu nullimiseks

ladu5.html

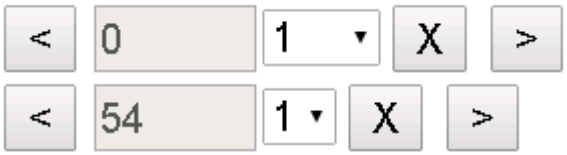
```
<!doctype html>
<html>
  <head>
    <title>Kalkulaator</title>
        <script src="ladu5.js"></script>
        <script>
```

```
          function lehealgus(){
            new Laohaldus("kiht1", 100, [1, 5, 10, 30, 100]);
            new Laohaldus("kiht2", 50, [1, 2, 3]);
          }
        </script>
  </head>
  <body onload="lehealgus();">
     <h1>Arvutamine</h1>
         <div id="kiht1"></div>
         <div id="kiht2"></div>
  </body>
</html>
```

ladu5.js

```
          function Laohaldus(kihinimi, kogus, muutused){
             this.algus=function(){
                   this.kiht=document.getElementById(kihinimi);
                   this.kogus=kogus;
                   this.muutused=muutused;
                   window[kihinimi+"_ladu"]=this;
                   var rippmenyy="<select id='"+kihinimi+"_valik'>";
                   for(var i=0; i<muutused.length; i++){
                      rippmenyy+="<option>"+muutused[i]+"</option>";
                   }
                   rippmenyy+="</select>";
                   this.kiht.innerHTML=
                "<input type='button' value='&lt;' onClick='"+
                   kihinimi+"_ladu.v2iksemaks();' /> "+
                "<input type='text' id='"+
                   kihinimi+"_vastus' style='width: 50px' disabled />"+
                rippmenyy+
                "<input type='button' value='X' onClick='"+
                   kihinimi+"_ladu.nulli();' /> "+
                "<input type='button' value='&gt;' onClick='"+
                   kihinimi+"_ladu.suuremaks();' /> ";
                   this.vastusekiht=document.getElementById(kihinimi+"_vastus");
                   this.muutusevalik=document.getElementById(kihinimi+"_valik");
                   this.kuva();
                }
                this.kuva=function(){
                   this.vastusekiht.value=this.kogus;
                }
                this.v2iksemaks=function(){
                   this.kogus-=this.muutused[this.muutusevalik.selectedIndex];
                   this.kuva();
                }
                this.suuremaks=function(){
                   this.kogus+=this.muutused[this.muutusevalik.selectedIndex];
                   this.kuva();
                }
                this.nulli=function(){
                   this.kogus=0;
                   this.kuva();
                }
                this.algus();
          }
```

Arvutamine



Pallid platsil

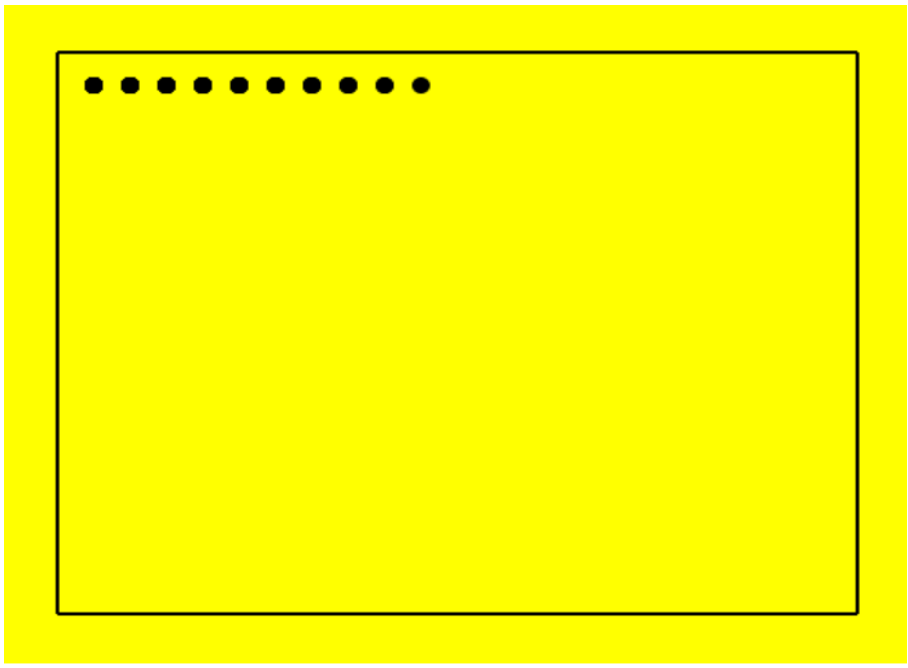
- Koosta tsükkel platsile pallide lisamiseks

Lisa platsile parameetrid suuruse kohta pikslites.

Lisa platsile äärejoon ja määra selle kaugus servast

```
<!doctype html>
<html>
  <head>
    <title>Kujundid</title>
    <script>
       function Pall(x, y, r){
          this.x=x;
          this.y=y;
          this.r=r;
          this.joonista=function(g){
           g.beginPath();
                   g.arc(this.x, this.y, this.r,
                          0, 2*Math.PI, true);
           g.fill();
          }
       }
       function Plats(kihiId, laius, korgus, servakaugus){
          window[kihiId+"_joonis"]=this;
          this.servakaugus=servakaugus;
          this.alusta=function(){
             var sisu=
               "<canvas id='"+kihiId+"_tahvel' width='"+laius+
                   "' height='"+korgus+"' "+
                " style='background-color: yellow' ></canvas><br/>";
             document.getElementById(kihiId).innerHTML=sisu;
             this.kujundid=new Array();
             this.tahvel=document.getElementById(kihiId+"_tahvel");
          }
          this.lisaKujund=function(kujund){
            this.kujundid.push(kujund);
            this.joonista();
          }
          this.joonista=function(){
             var g=this.tahvel.getContext("2d");
             g.beginPath();
             g.moveTo(this.servakaugus, this.servakaugus);
             g.lineTo(this.tahvel.width-this.servakaugus, this.servakaugus);
             g.lineTo(this.tahvel.width-this.servakaugus,
                  this.tahvel.height-this.servakaugus);
             g.lineTo(this.servakaugus,this.tahvel.height-this.servakaugus);
```

```
            g.lineTo(this.servakaugus, this.servakaugus);
            g.stroke();
            for(var i=0; i<this.kujundid.length; i++){
               this.kujundid[i].joonista(g);
            }
         }
         this.alusta();
      }
      var kiht1_joonis, kiht2_joonis
      function algus(){
        //Lisa platsile parameetrid suuruse kohta pikslites
        //Lisa platsile äärejoon ja määra selle kaugus servast
        kiht1_joonis=new Plats("kiht1", 500, 400, 30);
        for(var i=0; i<10; i++){
          kiht1_joonis.lisaKujund(new Pall(50+20*i, 50, 5));
        }
        kiht2_joonis=new Plats("kiht2", 50, 30, 3);
        kiht2_joonis.lisaKujund(new Pall(25, 15, 5));
      }
    </script>
  </head>
  <body onload="algus();">
     <div id="kiht1"></div>
     <div id="kiht2"></div>
  </body>
</html>
```



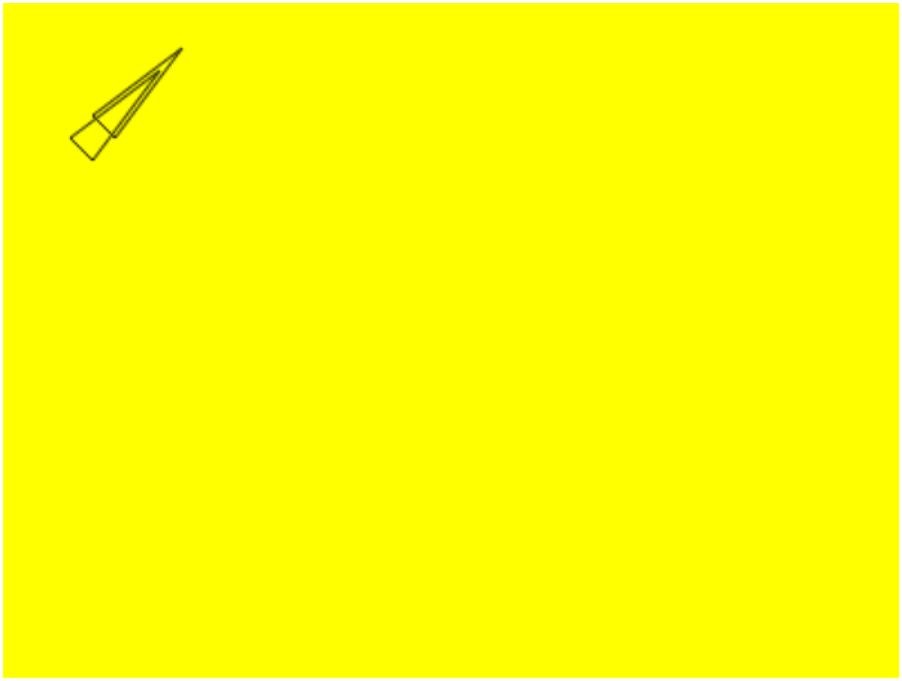
Kahemõõtmelised kujundid

Vektor asukoha ja nihke jaoks. Kahest vektorist Joon ning Joontest Kujundid, mida eraldi liigutada saab. All näites hakkab kumbki kolmnurk omasoodu liikuma.

```
<!doctype html>
<html>
```

```
<head>
   <title>Graafika</title>
   <script>
      function Vektor(x, y){
         this.x=x;
         this.y=y;
         this.tekstina=function(){
            return "("+this.x+", "+this.y+")";
         }
         this.liida=function(v){
            return new Vektor(this.x+v.x, this.y+v.y);
         }
         this.korruta=function(kordaja){
            return new Vektor(this.x*kordaja, this.y*kordaja);
         }
      }

      function Joon(p1, p2){
         this.punktid=new Array();
         this.punktid[0]=p1;
         this.punktid[1]=p2;
         this.tekstina=function(){
            return "Joon: "+this.punktid[0].tekstina()+
               " - "+this.punktid[1].tekstina();
         }
         this.nihuta=function(nihe){
            for(var i=0; i<this.punktid.length; i++){
               this.punktid[i]=this.punktid[i].liida(nihe);
            }
         }
         this.joonista=function(g){
            g.beginPath();
            g.moveTo(this.punktid[0].x, this.punktid[0].y);
            g.lineTo(this.punktid[1].x, this.punktid[1].y);
            g.stroke();
         }
         this.kopeeri=function(){
            return new Joon(this.punktid[0], this.punktid[1]);
         }
      }

      function Kujund(){
         this.kujundid=new Array();
         this.liikumisSamm=new Vektor(0, 0);
         this.lisaKujund=function(k){
           this.kujundid.push(k);
         }
         this.tekstina=function(){
           var t="Kujund (";
           for(var i=0; i<this.kujundid.length; i++){
              t+=this.kujundid[i].tekstina()+"; ";
           }
           return t+")";
         }
          this.muudaLiikumisSamm=function(samm){
               this.liikumisSamm=samm;
          }
          this.liigu=function(){
               this.nihuta(this.liikumisSamm);
          }
          this.nihuta=function(nihe){
            for(var i=0; i<this.kujundid.length; i++){
               this.kujundid[i].nihuta(nihe);
            }
```

```
            }
            this.joonista=function(g){
               for(var i=0; i<this.kujundid.length; i++){
                  this.kujundid[i].joonista(g);
               }
            }
            this.kopeeri=function(){
            var t=new Kujund();
            for(var i=0; i<this.kujundid.length; i++){
                   t.lisaKujund(this.kujundid[i].kopeeri());
            }
                     return t;
          }
        }

        var v1=new Vektor(30, 50);
        var v2=new Vektor(70, 20);
        var v3=new Vektor(40, 60);
        var j1=new Joon(v1, v2);
        var j2=new Joon(v2, v3);
        var j3=new Joon(v3, v1);
        kolmnurk=new Kujund();
        kolmnurk.lisaKujund(j1);
        kolmnurk.lisaKujund(j2);
        kolmnurk.lisaKujund(j3);
            kolmnurk2=kolmnurk.kopeeri();
            kolmnurk.muudaLiikumisSamm(new Vektor(1, 0));
            kolmnurk2.muudaLiikumisSamm(new Vektor(0, 1));
            var g;
            function liiguta(){
               document.getElementById("kiht1").innerHTML=
                     kolmnurk.tekstina()+"<br />"+kolmnurk2.tekstina();
               g.clearRect(0, 0, 400, 300);
               kolmnurk.liigu();
               kolmnurk.joonista(g);
               kolmnurk2.liigu();
               kolmnurk2.joonista(g);
            }

        function leheAlgus(){
           g=document.getElementById("tahvel1").getContext("2d");
               setInterval("liiguta();", 500);
        }
     </script>
  </head>
  <body onload="leheAlgus()">
    <canvas id="tahvel1" width="400" height="300" style="background-color:
yellow">
    </canvas>
    <div id="kiht1"></div>
  </body>
</html>
```

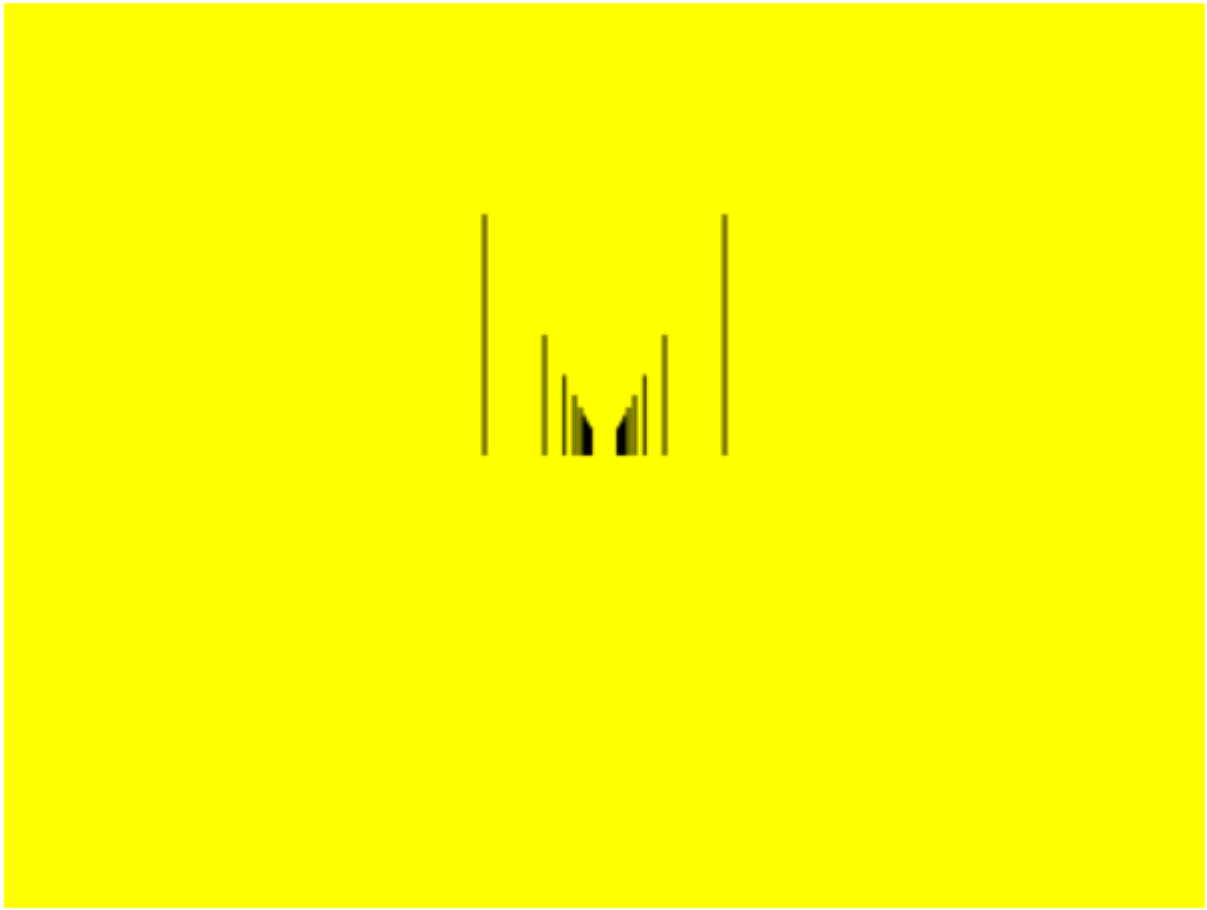


Kujund (Joon: (39, 50) - (79, 20); Joon: (79, 20) - (49, 60); Joon: (49, 60) - (39, 50);)
Kujund (Joon: (30, 59) - (70, 29); Joon: (70, 29) - (40, 69); Joon: (40, 69) - (30, 59);)

Kaks postirida

```
<!doctype html>
<html>
  <head>
    <title>3D</title>
    <script>
       var keskx=200;
       var kesky=150;
       var suurendus=10;
       function ex(px, py, pz){ //x ekraanil
          return keskx+suurendus*px/pz;
       }

       function ey(px, py, pz){
          return kesky-suurendus*py/pz;
       }

       function joonista(){
         var g=document.getElementById("tahvel1").getContext("2d");
         g.beginPath();
         var x=-20;
         var y=40;
         for(var z=5; z<50; z+=5){
            g.moveTo(ex(x, 0, z), ey(x, 0, z));
            g.lineTo(ex(x, y, z), ey(x, y, z));
         }
         //Lisage ka paremale poole postid
         x=20;
         for(var z=5; z<50; z+=5){
            g.moveTo(ex(x, 0, z), ey(x, 0, z));
            g.lineTo(ex(x, y, z), ey(x, y, z));
```

```
        }
        g.stroke();
      }
    </script>
  </head>
  <body onload="joonista();">
     <canvas id="tahvel1" width="400" height="300"
      style="background-color: yellow"/></canvas>
  </body>
</html>
```



Prototüübid

String

```
<!doctype html>
<html>
  <head>
    <title>Laiendamine</title>
    <script>

       String.prototype.tahtedeArv=function(){
          return this.length;
       }

       String.prototype.sonadeArv=function(){
```

```
         return this.split(" ").length;
      }

      function leheAlgus(){
         var lause="Juku tuli kooli";
         document.getElementById("vastus").innerHTML=
                  lause.tahtedeArv()+" tähte, "+
                  lause.sonadeArv()+" sõna.";
      }
    </script>
  </head>
  <body onload="leheAlgus();">
    <div id="vastus"></div>
  </body>
</html>
```

15 tähte, 3 sõna.

Vektor

Käsklus muudaVektoritKorrutades, jätab objekti samaks, lihtsalt muudab selle x- ja y-koordinaati kordaja jagu. Käsklus uusVektorKorrutades jätab vana vektori muutumatuks, käsust tagastatakse uus vektor, mille võimalik hilisem muutmine enam esialgset vektorit ei sega.

```
<!doctype html>
<html>
  <head>
    <title>Laiendamine</title>
    <script>
       function Vektor(x, y){
          this.x=x;
          this.y=y;
          this.pikkus=function(){
             return Math.sqrt(this.x*this.x+this.y*this.y);
          }
       }

       Vektor.prototype.tekstina=function(){
         return "("+this.x+", "+this.y+")";
       }

       Vektor.prototype.muudaVektoritKorrutades=function(kordaja){
         this.x=this.x*kordaja;
         this.y=this.y*kordaja;
       }

       Vektor.prototype.uusVektorKorrutades=function(kordaja){
         return new Vektor(this.x*kordaja, this.y*kordaja);
       }

       function leheAlgus(){
         var autokiirus=new Vektor(3, 4);
         autokiirus.muudaVektoritKorrutades(2);
         document.getElementById("vastus").innerHTML=
          "Tekstina: "+autokiirus.tekstina()+
          " kogukiirus "+autokiirus.pikkus();
         var jalgrattakiirus=autokiirus.uusVektorKorrutades(0.2);
         alert(jalgrattakiirus.tekstina());
         alert(autokiirus.tekstina());
```

```
        }
    </script>
  </head>
  <body onload="leheAlgus();">
    <div id="vastus"></div>
  </body>
</html>
```

(1.2, 1.6)
(6, 8)

Ring

Ringile antakse funktsiooni parameetrina kaasa ligipääs teisele ringile. Funktsiooni sees otsustatakse, kas ringid puutuvad omavahel kokku.

```
<!doctype html>
<html>
  <head>
    <title>Laiendamine</title>
    <script>
       function Asukoht(x, y){
          this.x=x;
          this.y=y;
          this.tekstina=function(){
            return "("+this.x+", "+this.y+")";
          }
       }
       var a=new Asukoht(0, 0);
       function Ring(x, y, r){
          Asukoht.call(this, x, y);
          this.raadius=r;
          this.pindala=function(){
             return 3.14*this.raadius*this.raadius;
          }
          this.kasPuutub=function(r){
             var dx=this.x-r.x;
             var dy=this.y-r.y;
             return Math.sqrt(dx*dx+dy*dy)<this.raadius+r.raadius;
          }
       }
       Ring.prototype=a;
       var r1=new Ring(2, 4, 7);
       var r2=new Ring(2, 80, 4);

       function leheAlgus(){
         document.getElementById("vastus").innerHTML=
           r1.tekstina()+r2.tekstina()+" puutub: "+r1.kasPuutub(r2);
       }
    </script>
  </head>
  <body onload="leheAlgus();">
    <div id="vastus"></div>
  </body>
</html>
```

(2, 4)(2, 80) puutub: false

Ringide massiiv

Ringe on rohkem. Kõik algul oleva ringiga kokku puutuvad ringide muutujad kopeeritakse eraldi massiivi ning sealt trükitakse välja.

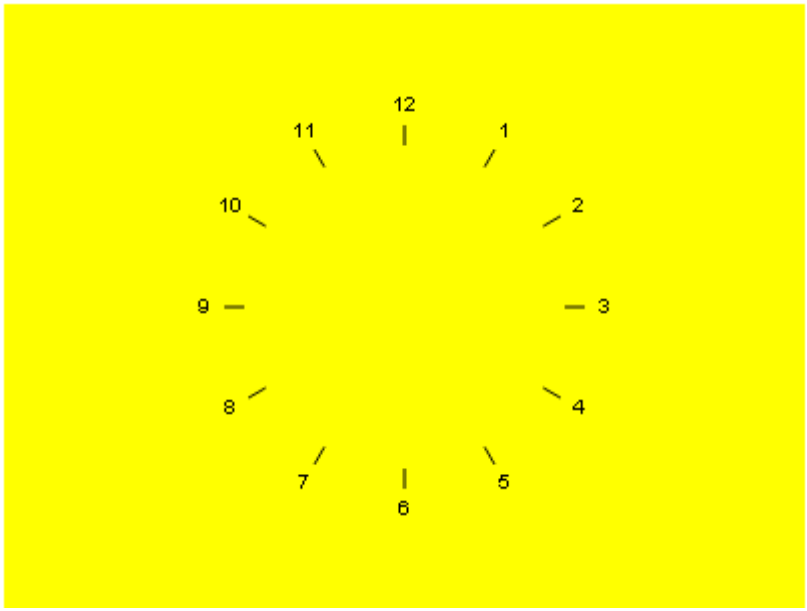
```
<!doctype html>
<html>
  <head>
    <title>Laiendamine</title>
    <script>
       function Asukoht(x, y){
          this.x=x;
          this.y=y;
          this.tekstina=function(){
            return "("+this.x+", "+this.y+")";
          }
       }
       var a=new Asukoht(0, 0);
       function Ring(x, y, r){
          Asukoht.call(this, x, y);
          this.raadius=r;
          this.pindala=function(){
             return 3.14*this.raadius*this.raadius;
          }
          this.kasPuutub=function(r){
             var dx=this.x-r.x;
             var dy=this.y-r.y;
             return Math.sqrt(dx*dx+dy*dy)<this.raadius+r.raadius;
          }
       }
       Ring.prototype=a;

       var ringid=new Array();
       ringid[0]=new Ring(2, 4, 7);
       ringid[1]=new Ring(2, 80, 4);
       ringid[2]=new Ring(2, 8, 4);
       ringid[3]=new Ring(20, 4, 40);
       ringid[4]=new Ring(10, 10, 14);

       function leheAlgus(){
         var puutuvad=new Array();
         for(var i=1; i<ringid.length; i++){
            if(ringid[0].kasPuutub(ringid[i])){
               puutuvad.push(ringid[i]);
            }
         }

         var t="Tabatud: ";
         for(var i=0; i<puutuvad.length; i++){
            t+=puutuvad[i].tekstina();
         }

         document.getElementById("vastus").innerHTML=t;
       }
    </script>
  </head>
  <body onload="leheAlgus();">
    <div id="vastus"></div>
  </body>
</html>
```

Tabatud: (2, 8)(20, 4)(10, 10)

Nurgaarvutused

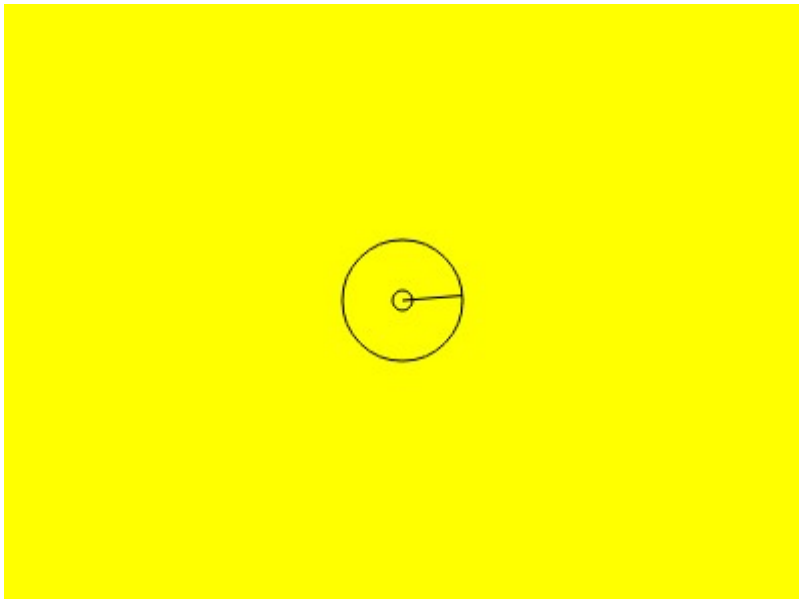
Kella numbrilaud

```
<!doctype html>
<html>
  <head>
     <title>Ringi arvutused</title>
     <script>
       var keskx=200;
       var kesky=150;
         var tekstiraadius=100;
       var ringiraadius=90;
         var siseraadius=80;
       var nurk=Math.PI/2;
       var punktidearv=12;
       var nurgavahe=2*Math.PI/punktidearv;
       function joonista(){
          var g=document.getElementById("t1").getContext("2d");
              nurk-=nurgavahe;
              g.textAlign="center";
              g.textBaseline="middle";
          for(var i=1; i<=punktidearv; i++){
            g.fillText(i, keskx+tekstiraadius*Math.cos(nurk),
                      kesky-tekstiraadius*Math.sin(nurk));
                g.beginPath();
                  g.moveTo(keskx+ringiraadius*Math.cos(nurk),
                      kesky-ringiraadius*Math.sin(nurk));
                g.lineTo(keskx+siseraadius*Math.cos(nurk),
                      kesky-siseraadius*Math.sin(nurk));
                  g.stroke();
            nurk-=nurgavahe;
          }
       }
     </script>
  </head>
  <body onload="joonista();">
     <h1>Ruudud ringjoonel</h1>
     <canvas id="t1" width="400" height="300"
       style="background-color: yellow" ></canvas>
  </body>
</html>
```



Keeratav nupp

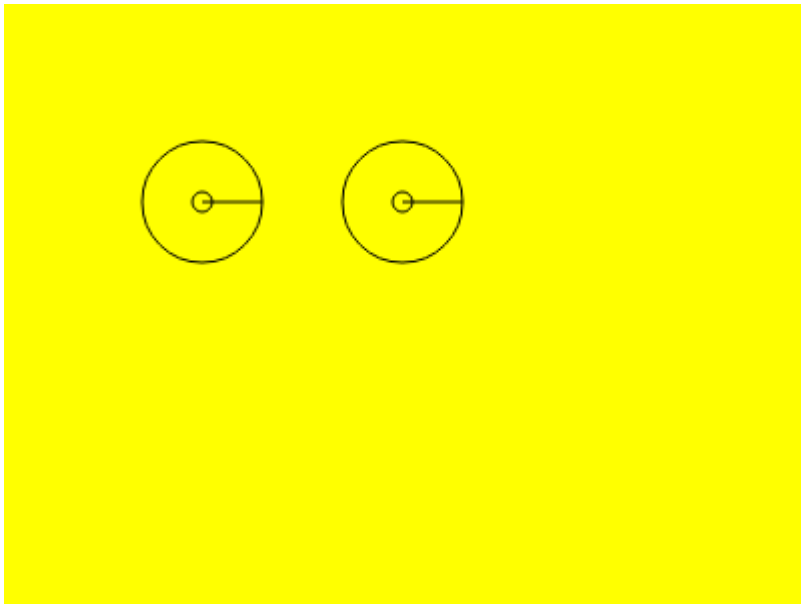
```
<!doctype html>
<html>
  <head>
     <title>Nurga leidmine</title>
     <script>
       var keskx=200;
       var kesky=150;
       var ringiraadius=30;
       function hiirLiigub(e){
          var tahvlikoht=document.getElementById("t1").
               getBoundingClientRect();
          var g=document.getElementById("t1").getContext("2d");
          g.clearRect(0, 0, 400, 300);
          var hx=e.clientX-tahvlikoht.left;
          var hy=e.clientY-tahvlikoht.top;
          var nurk=Math.atan2(hy-kesky, hx-keskx);
              g.beginPath();
              g.arc(keskx, kesky, ringiraadius, 0, 2*Math.PI, true);
              g.stroke();
              g.beginPath();
              g.arc(keskx, kesky, 5, 0, 2*Math.PI, true);
              g.stroke();
              g.beginPath();
              g.moveTo(keskx+ringiraadius*Math.cos(nurk),
                   kesky+ringiraadius*Math.sin(nurk));
              g.lineTo(keskx, kesky);
              g.stroke();
       }
     </script>
  </head>
  <body onload="hiirLiigub(event)">
     <h1>Ruudud ringjoonel</h1>
     <canvas id="t1" width="400" height="300"
       style="background-color: yellow"
       onmousemove="hiirLiigub(event)"></canvas>
  </body>
</html>
```



Nupp objektina

```
<!doctype html>
<html>
  <head>
     <title>Nurga leidmine</title>
     <script>
         var kujundid=new Array();
         function KeeratavNupp(x, y, r){
            this.x=x;
              this.y=y;
              this.r=r;
              this.nurk=0;
              this.joonista=function(g){
                 g.beginPath();
                 g.arc(this.x, this.y, this.r, 0, 2*Math.PI, true);
                 g.stroke();
                 g.beginPath();
                 g.arc(this.x, this.y, 5, 0, 2*Math.PI, true);
                 g.stroke();
                 g.beginPath();
                 g.moveTo(this.x+this.r*Math.cos(this.nurk),
                       this.y+this.r*Math.sin(this.nurk));
                 g.lineTo(this.x, this.y);
                 g.stroke();
              }
         }
        function joonista(){
           var g=document.getElementById("t1").getContext("2d");
           g.clearRect(0, 0, 400, 300);
           for(var i=0; i<kujundid.length; i++){
                 kujundid[i].joonista(g);
              }
        }
          kujundid.push(new KeeratavNupp(100, 100, 30));
          kujundid.push(new KeeratavNupp(200, 100, 30));
     </script>
  </head>
  <body onload="joonista();">
     <h1>Ruudud ringjoonel</h1>
```

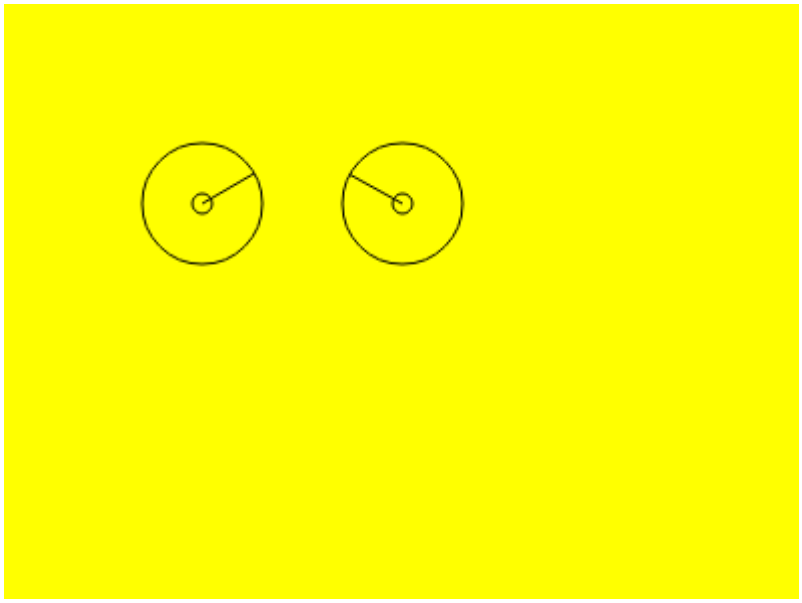
```
    <canvas id="t1" width="400" height="300"
      style="background-color: yellow" ></canvas>
  </body>
</html>
```



Keeratavad nupud objektina

```
<!doctype html>
<html>
  <head>
     <title>Nurga leidmine</title>
     <script>
         function KeeratavNupp(x, y, r){
            this.x=x;
              this.y=y;
              this.r=r;
              this.nurk=0;
              this.joonista=function(g){
                 g.beginPath();
                 g.arc(this.x, this.y, this.r, 0, 2*Math.PI, true);
                 g.stroke();
                 g.beginPath();
                 g.arc(this.x, this.y, 5, 0, 2*Math.PI, true);
                 g.stroke();
                 g.beginPath();
                 g.moveTo(this.x+this.r*Math.cos(this.nurk),
                       this.y+this.r*Math.sin(this.nurk));
                 g.lineTo(this.x, this.y);
                 g.stroke();
              }
              this.kasPihtas=function(hx, hy){
                 var dx=hx-this.x;
                   var dy=hy-this.y;
                   return dx*dx+dy*dy<this.r*this.r;
              }
              this.muudaNurk=function(hx, hy){
                 this.nurk=Math.atan2(hy-this.y, hx-this.x);
              }
         }
```

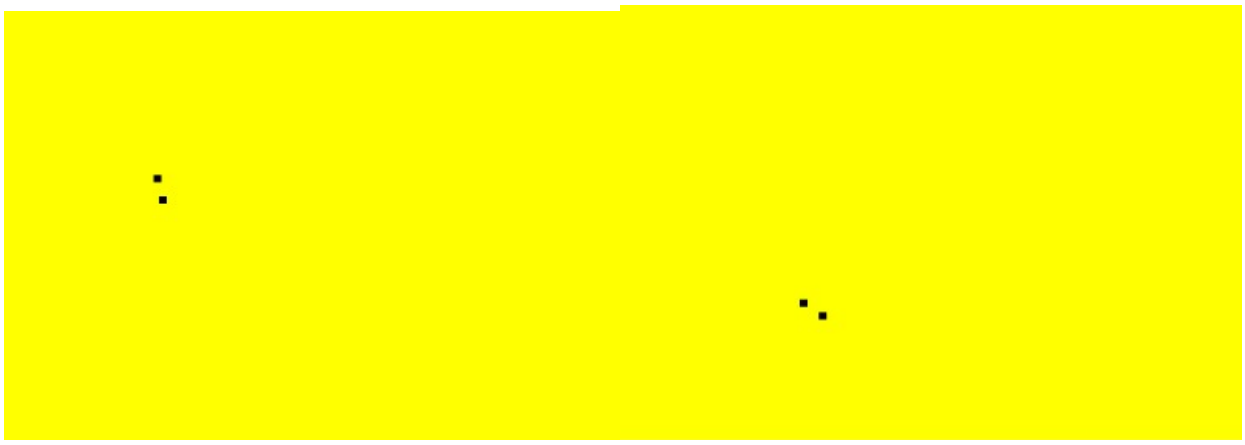
```
        function KujunditeHaldus(tahvliId){
             this.kujundid=new Array();
             this.tahvliId=tahvliId;
             this.joonista=function(){
               var g=document.getElementById(this.tahvliId).getContext("2d");
                 g.clearRect(0, 0, 400, 300);
               for(var i=0; i<this.kujundid.length; i++){
                  this.kujundid[i].joonista(g);
               }
             }
             this.lisaKujund=function(kujund){
                this.kujundid.push(kujund);
             }
             this.hiir=function(e){
               var tahvlikoht=document.getElementById(this.tahvliId).
              getBoundingClientRect();
               var hx=e.clientX-tahvlikoht.left;
              var hy=e.clientY-tahvlikoht.top;
                 for(var i=0; i<this.kujundid.length; i++){
                    if(this.kujundid[i].kasPihtas(hx, hy)){
                       this.kujundid[i].muudaNurk(hx, hy);
                    }
                 }
                 this.joonista();
             }
        }
        var haldus=new KujunditeHaldus("t1");
        haldus.lisaKujund(new KeeratavNupp(100, 100, 30));
        haldus.lisaKujund(new KeeratavNupp(200, 100, 30));

    </script>
  </head>
  <body onload="haldus.joonista();">
    <h1>Ruudud ringjoonel</h1>
    <canvas id="t1" width="400" height="300"
      style="background-color: yellow"
      onmousemove="haldus.hiir(event)"></canvas>
  </body>
</html>
```



Maa ja Kuu

Planeet tiirleb ringjoonel, kaaslane ümber planeedi.

```
<!doctype html>
<html>
  <head>
     <title>Ringi arvutused</title>
     <script>
       var keskx=200;
       var kesky=150;
       var ringiraadius=100;
       var nurk=0;
       var nurgavahe=0.01; //radiaani
       var nurgavahe2=0.3; //radiaani
       var nurk2=Math.PI/2;
       function liigu(){
          var g=document.getElementById("t1").getContext("2d");
          g.clearRect(0, 0, 400, 300);
              var x1=keskx+ringiraadius*Math.cos(nurk);
              var y1=kesky-ringiraadius*Math.sin(nurk);
          g.fillRect(x1, y1, 5, 5);
          g.fillRect(x1+15*Math.cos(nurk2),
                     y1-15*Math.sin(nurk2), 5, 5);
          nurk+=nurgavahe;
          nurk2+=nurgavahe2;
       }
     </script>
  </head>
  <body onload="setInterval('liigu()', 50);">
     <h1>Ringjoone parameetriline vĆµrrand</h1>
     <canvas id="t1" width="400" height="300"
       style="background-color: yellow" ></canvas>
  </body>
</html>
```

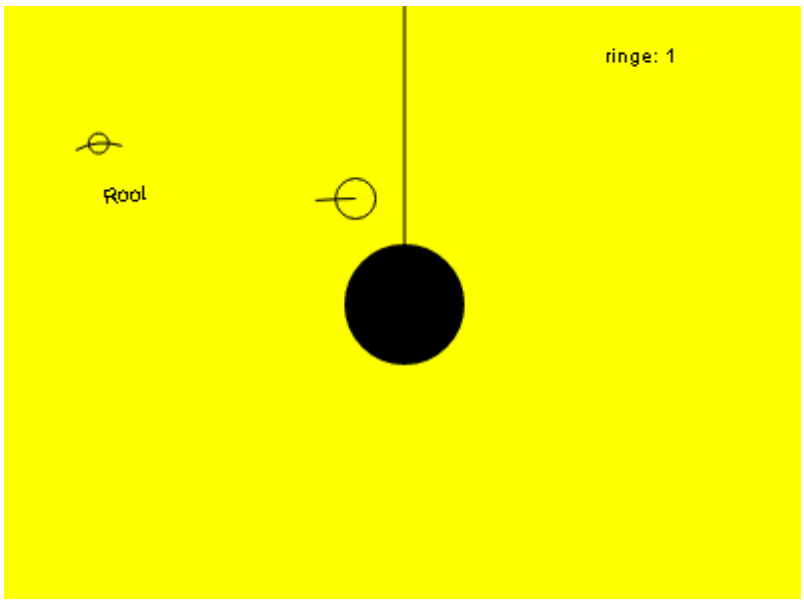


Roolitav liikur ringrajal

```
<!doctype html>
<html>
  <head>
     <title>Ringi arvutused</title>
     <script>
```

```
function Liikur(x, y, kiirus, nurk, nurgamuutus){
  this.algus=function(){
   this.x=x;
   this.y=y;
   this.kiirus=kiirus;
   this.nurk=nurk;
   this.nurgamuutus=nurgamuutus;
  }
  this.liigu=function(){
    this.nurk+=this.nurgamuutus;
    this.x+=this.kiirus*Math.cos(this.nurk);
    this.y+=this.kiirus*Math.sin(this.nurk);
  }
  this.muudaNurk=function(uusNurk){
     this.nurk=uusNurk;
  }
  this.uusRooliAsend=function(uusAsend){
     this.nurgamuutus=uusAsend;
  }
  this.joonista=function(g){
     g.beginPath();
     g.arc(this.x, this.y, 10, 0, 2*Math.PI, false);
     g.stroke();
     g.beginPath();
     g.moveTo(this.x,this.y);
     //Joone ots
     var jotsx=this.x+10*this.kiirus*Math.cos(this.nurk);
     var jotsy=this.y+10*this.kiirus*Math.sin(this.nurk);
     g.lineTo(jotsx, jotsy);

     var rattapikkus=10;
     var rattanurk=this.nurk+3*this.nurgamuutus;
     var rattaotsx=jotsx+rattapikkus*Math.cos(rattanurk);
     var rattaotsy=jotsy+rattapikkus*Math.sin(rattanurk);
     g.lineTo(rattaotsx, rattaotsy)
     g.stroke();
  }
  this.algus();
}

function Rool(keskx, kesky, raadius, nurk){
   this.algus=function(){
      this.keskx=keskx;
      this.kesky=kesky;
      this.raadius=raadius;
      this.nurk=nurk;
   }
   this.kysiNurk=function(){
      return this.nurk+Math.PI/2;
   }
   this.joonista=function(g){
       g.beginPath();
       g.arc(this.keskx, this.kesky, this.raadius,
            this.nurk-0.4, this.nurk+0.4, false);
       g.stroke();
       g.beginPath();
       g.arc(this.keskx+this.raadius*Math.cos(this.nurk),
             this.kesky+this.raadius*Math.sin(this.nurk),
               5, 0, 2*Math.PI, false);
       g.stroke();
      g.save();
      g.translate(this.keskx, this.kesky);
      g.rotate(this.nurk+(Math.PI/2));
```

```
            g.fillText("Rool", 0, 0);
            g.restore();
         }
         this.uusNurk=function(hx, hy){
            this.nurk=Math.atan2(hy-kesky, hx-keskx);
         }
         this.algus();
      }

      var auto1=new Liikur(100, 100, 1, Math.PI/3, -0.1);
      var rool1=new Rool(50, 100, 30, -Math.PI/2);

      function hiirLiigub(e){
         var tahvlikoht=document.getElementById("t1").getBoundingClientRect();
         var hx=e.clientX-tahvlikoht.left;
         var hy=e.clientY-tahvlikoht.top;
         rool1.uusNurk(hx, hy);
      }

      var ringx=200, ringy=150, ringr=30, ringideArv=0;

      function ringLiiklus(){
         var dx=auto1.x-ringx;
         var dy=auto1.y-ringy;
         if(Math.sqrt(dx*dx+dy*dy)<ringr+10){
            auto1.x=100; auto1.y=100;
         }
      }


      function liigu(){
         var g=document.getElementById("t1").getContext("2d");
         g.clearRect(0, 0, 400, 300);
         auto1.uusRooliAsend(rool1.kysiNurk()/10);
         ringLiiklus();
         var vanax=auto1.x;
         auto1.liigu();
         if(auto1.y<ringy){
             if(vanax>ringx && auto1.x<ringx){ringideArv++;}
             if(vanax<ringx && auto1.x>ringx){ringideArv--;}
         }
         auto1.joonista(g);
         rool1.joonista(g);
         g.beginPath();
         g.arc(ringx, ringy, ringr, 0, 2*Math.PI, true);
         g.fill();
         g.fillText("ringe: "+ringideArv, 300, 30);
         g.beginPath();
         g.moveTo(ringx, ringy);
         g.lineTo(ringx, 0);
         g.stroke();
      }
   </script>
</head>
<body onload="setInterval('liigu()', 50);">
   <h1>Ringjoone parameetriline vĆµrrand</h1>
   <canvas id="t1" width="400" height="300"
     style="background-color: yellow"
     onmousemove="hiirLiigub(event)"></canvas>
</body>
</html>
```


ringe: 1
Rool