ProgeTiigri õppematerjal

Tiigrihüppe Sihtasutus

Kliendipoolsed veebirakendused

Jaagup Kippar

2013

Sisukord

Algus

Veebilehestikke on loodud 1990ndatest alates. Peale andmete näitamise/vaatamise on veebis ka üsna algusest peale nendega midagi ka teha saanud. Siinses õppematerjalis püüamegi veebilehti elama panna, võimaldades kasutajal nende taga toimetada. Keskendume kliendi arvutis töötavatele Javaskripti lahendustele, mida saab kohe looma hakata - pole vaja käivitusõigust serveris ega ligipääsu andmebaasile kui keerukamaid ja riskiohtlikumaid võimalusi. Küllalt palju kasulikku ja rõõmsat saab ka kasutaja arvutis tööle panna.

Tekstilehekülg

Internetis saab vaadata kõike, mida sinna sisse pannakse ja mida vaataja lugeda suudab. Olgu see üksik pilt, helilõik või ka lihtsalt tekst. Teksti veebis kättesaadavaks tegemiseks tuleb see paigutada kausta, kus teksti lugeda saab. Lihtsamaks katsetamiseks piisab ka kohaliku masina kataloogist, kust siis hiljem sobiv fail avada.

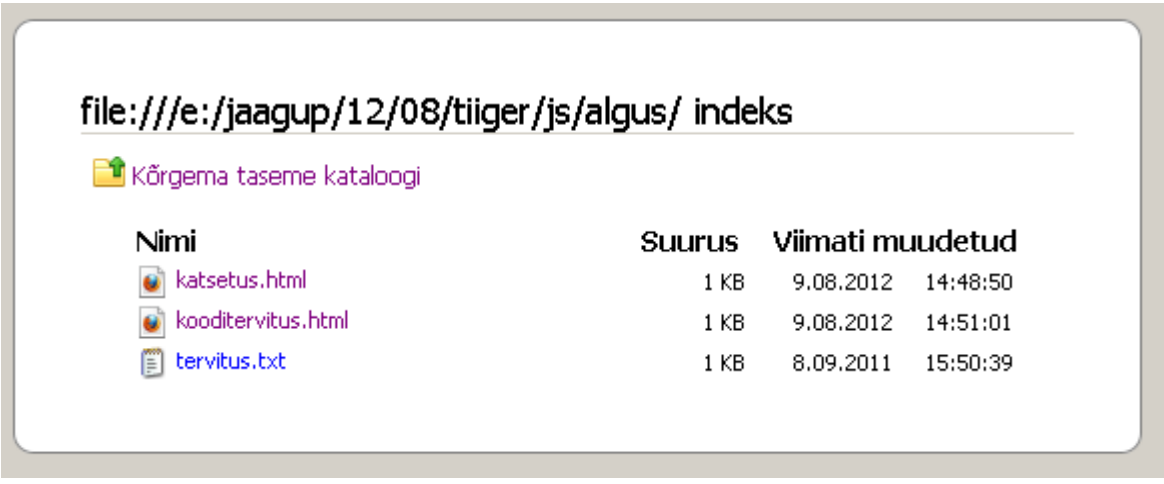
Näites loodi fail nimega

tervitus.txt

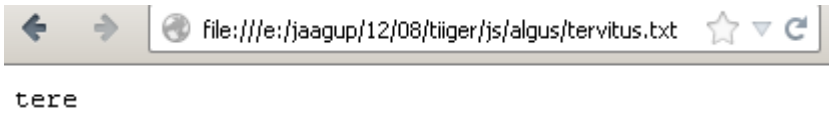
sisuga

tere

Veebilehitsejas kausta vaadates võib näha, et fail on olemas.



Faili avamisel tuleb sisu nähtavale



Ongi esimene tervitav teave edasi antud.

Millega tekstifaile luua - see sõltub juba kasutatavast arvutist ja operatsioonisüsteemist. Linuxi all kirjutajatel on arvatavasti tuttavad redaktorid nimedega vi, joe või pico. Windowsi-masinates kuulub standardkomplekti tarvikute all olev Notepad. Mõnevõrra koodikirjutamist hõlbustab ning treppida aitab eraldi lisatav Notepad++. Aga eks vastavalt maitsele leiab kirjutusprogramme piisavalt. Peab lihtsalt vaatama, et tulemuseks oleks puhtalt salvestatav tekst - sellisena ka edaspidiseid veebilehti kindlam teha.

HTML-leht

Palja tekstiga saab edasi anda jutu sisu. Tühjade ridade ning taandridadega saab väljanägemist parasjagu sättida. Kirjutusmasina-ajastul nendest moodustest täiesti piisas. Nüüd aga on inimesed harjunud vaatama mitmekülgsemat lehekülge nii väljanägemise kui võimaluste poole pealt. Aastaid on veebiga koos käinud HTMLi-nimeline keel, mis koos oma tõusude ja mõõnadega on tänapäevalgi valitsevaks jäänud jõudes nüüd oma viienda versioonini. Järgnevalt üks koodinäide, kus HTMLi abil leht kokku pandud. Mõned seletused sinna juurde.

Esimene rida

```
<!DOCTYPE html>
```

annab veebilehitsejale teada, et millist tüüpi dokumenti oodatakse.

Leht ise jaguneb kaheks suureks osaks: head ning body. Esimeses neist on andmed lehe kohta - hiljem samuti ettevalmistatud koodilõigud. Teise tuleb lehe sisu ise. Andmeks lehe kohta on ka title ehk pealkiri. Edasi soovitavalt kasutatav kooditabel. Täpitähtedega murede vältimiseks oleks see hea sättida utf-8 peale (ning ühtlasi hoolitseda, et ka oma redaktor selles kodeeringus tähed väljastaks). Notepad++ juures kohe eraldi selle jaoks menüü olemas.

Teksti kujundamise jaoks on loodud omaette astmeliste laadilehtede keel nimega CSS. Siin näites antakse teada, et kõik ülemise/esimese taseme pealkirjad joondatakse keskele (`text-align: center`). Kel aga on põhjust kujundusega rohkem tegelda, võib lisateavet hankida veebist nt. http://www.w3schools.com/ õpetuste hulgast. Või siis Tiigrihüppe abil eraldi valmivast veebikujunduse õppematerjalist.

```
<html>
  <head>
     <title>Kujunduse katsetus</title>
       <meta charset="UTF-8" />
       <style type="text/css">
          h1{
              text-align: center;
            }
       </style>
  </head>
```

Sisupoole andmed on lehel üldiselt näha. Kõigepealt siis pealkiri, mis viisakasti `<h1>` ja `</h1>` märgendite vahele pandud. Märgend h1 tähendab esimese taseme ehk suurimat pealkirja (heading 1). Näitamise mõttes loodud loetelu (`<ul>` ehk unordered list), mille sees siis üksikud punktid `<li>`-käsuga (list item). Ning üldjuhul kui mõni märgend algab, siis peab see ka kusagil lõppema. Nii on pealkirja sabas kaldkriipsuga `</h1>` ning loetelu lõpus `</ul>`.

Et HTML on tavaline tekstivorming, siis tuleb kõik lisaandmed seal sees tavalise tekstina edasi anda. Nii ka viide teisele lehele on lihtsalt käsklus nimega `a` (anchor), millel atribuudina küljes `href`

(hyperlink reference) koos aadressiga, kuhu veebilehitseja siis vastava teksti vajutamisel suunatakse.

```
  <body>
     <h1>Katsetuste leht</h1>
       <ul>
          <li>Kooli rõõmus algus</li>
            <li>Veebi <a href="http://www.neti.ee/">otsinguleht</a></li>
            <li>HTMLi <a href="http://validator.w3.org/">validaator</a></li>
       </ul>
  </body>
</html>
```

Ning selgema pildi saamiseks lehe kood tervikuna.

```
<!DOCTYPE html>
<html>
  <head>
     <title>Kujunduse katsetus</title>
       <meta charset="UTF-8" />
       <style type="text/css">
          h1{
              text-align: center;
            }
       </style>
  </head>
  <body>
     <h1>Katsetuste leht</h1>
       <ul>
          <li>Kooli rõõmus algus</li>
            <li>Veebi <a href="http://www.neti.ee/">otsinguleht</a></li>
            <li>HTMLi <a href="http://validator.w3.org/">validaator</a></li>
       </ul>
  </body>
</html>
```

Välja näeb viisakalt ja lihtsalt vormindatud lehena.

Katsetuste leht

- Kooli rõõmus algus
- Veebi otsinguleht
- HTMLi validaator

Viimane validaatori viide on siia juurde täiesti mõttega pandud. Kel lehele esimene pilt ette saadud, võiks kontrollida, et leht ikka igapidi viisakalt kokku pandud on ning vajalikud märgendid oma kohtadel. Muidu võib tulemusena ootamatuid üllatusi juhtuma hakata. Kontroll küllaltki lihtne. Kui enamik korras, näidatakse rohelist vastuslehte. Kui aga märgatavaid muresid süntaksi poole pealt, siis tuleb vastus punane ning tasub hakata ridade kaupa veateateid ja soovitusi lugema. Mõnigikord ei pruugi viga olla sama mis näidatakse, kuid enamjaolt on osutataval real või paar rida kõrgemal midagi viltu läinud, mida tasub siis kohendada.

Ülesandeid

- Tee näited läbi, veendu nende avanemises veebilehitsejas.
- Koosta veebileht enese tutvustamiseks - kus ja millal sündinud, kus koolis käinud, millistest keeltest (kasvõi natuke) aru saad, mis oskustega saad sõpradele kasulik olla.

- Usinamatele: Tee failid võimaluse korral kättesaadavaks mõnes avalikus veebiserveris (nt. zone.ee). Vaata neid ise ning lase oma sõbral vaadata. Muuda sisu natuke ning vaata uuesti tulemust.

Kasutajat tervitav leht

Eelnev leht näeb välja ikka samasugune. Kui tahta lehe sisu muuta, siis tuleb lehe loojal sinna uus jutt kirjutada ning serverisse vaatajatele üles panna.

Lehele kirjutatud programmikood võimaldab aga kasutajale reageerida, lihtsaimal juhul ta nime küsida ning selle järgi tervitada. Selline suhtlus vajab natuke sättimist ette.

Levinud sisestuselemendiks on HTMLi märgend nimega input. Teksti jaoks on ta tüübiks text. Programmikoodi abil elemndi poole pöördumiseks tasub temale määrata ka id. Nõnda siis sobib veebilehe koodi sisse rida

```
<input type="text" id="eesnimi" />
```

HTML-fail kokku näeks välja siis järgmine:

```
<!DOCTYPE html>
<html>
  <head>
     <title>Andmete sisestus</title>
  </head>
  <body>
     <h1>Katsetuste leht</h1>
    Palun eesnimi:
      <input type="text" id="eesnimi" />
  </body>
</html>
```

ja lehele tekib sisestuskast.

Tahtes lehel midagi juhtuma panna, tuleb arvutile teada anda, mille peale midagi toimuma hakkab. Toimetust saab mugavalt käivitada nupu abil, selle jaoks ka sobiv rida lehele.

```
        <input type="button" value="OK" />
```

Kuvamiseks sobib lehele div-kiht id-ga vastus.

```
        <div id="vastus">
          Vastuse koht.
        </div>
```

Kõige tähtsamas lõigus tuleb veebilehitsejale seletada, et mida nupuvajutuse peale tuleb tegema hakata. Veebilehele programmikoodi lisamiseks sobib head-ossa skriptiplokk

```
<script>

</script>
```

Sinna sisse saab siis hakata arvuti jaoks käsklusi kirjutama. Käsklusi ootava lehe blankett näeb välja järgmine:

```
<!DOCTYPE html>
<html>
  <head>
     <title>Andmete sisestus</title>
     <script>

     </script>
  </head>
  <body>
     <h1>Katsetuste leht</h1>
   <div>
    Palun eesnimi:
      <input type="text" id="eesnimi" />
      <input type="button" value="OK" />
   </div>
   <div id="vastus">
     Vastuse koht.
   </div>
  </body>
</html>
```

Soovime, et tervitataks inimest, kelle nimi tekstivälja sisse kirjutatakse. Selleks tuleb eesnimi tekstiväljast välja võtta, juurde lisada tervitussõna ning kogu lugu kasutajale kuvada.

Veebilehe elemendile, millel on id-atribuut, saab ligipääsu küsida Javaskripti käsklusega

```
document.getElementById
```

Elementide sisu küsimiseks ja muutmiseks on neil omadus innerHTML. Kui soovin vastusplokki kirjutada lihtsalt Tere, siis see näeb välja

```
document.getElementById("vastus").innerHTML="Tere";
```

Paljas käsklus aga niisama iseenesest ei tea käivituda. Selleks tuleb käsklus panna nimega funktsiooni sisse ning nupuvajutuse peale see funktsioon tööle panna.

```
      function tervita(){
```

```
        document.getElementById("vastus").innerHTML="Tere";
      }
```

Näites siis tervita on funktsiooni nimi. Ümarsulud kuuluvad funktsiooni nime juurde. Ning järgnevate loogeliste sulgude vahele pannakse käsud, mis funktsiooni käivitamisel tööle pannakse. Nupuvajutuse peale funktsiooni käivitamiseks tuleb nupu onclick-atribuudis öelda, et milline funktsioon käima läheb. Sedakorda siis ainuke mis meil on ehk funktsioon nimega tervita.

```
<input type="button" value="OK" onclick="tervita();" />
```

Pärast täiendusi näeb kool välja järgmine

```
<!DOCTYPE html>
<html>
  <head>
     <title>Programmikoodi katsetus</title>
     <script>
      function tervita(){
               document.getElementById("vastus").innerHTML="Tere";
      }
     </script>
  </head>
  <body>
     <h1>Katsetuste leht</h1>
   <div>
    Palun eesnimi:
      <input type="text" id="eesnimi" />
      <input type="button" value="OK" onclick="tervita()" />
   </div>
   <div id="vastus">
      Vastuse koht
   </div>
  </body>
</html>
```

Enne nupuvajutust paistab vastuse kihil olema "Vastuse koht", pärast programmikoodi abil pandud "Tere".

Nõnda siis

```
document.getElementById("vastus").innerHTML=
  "Tere, "+document.getElementById("eesnimi").value;
```

tulemuseks on, et kihi "vastus" sisuks (innerHTML) saab tervitus inimesele, kelle eesnimi tekstivälja sisestati.

Katsetuste leht

Palun sisesta oma eesnimi: OK
Vastuse koht.

```
<!DOCTYPE html>
<html>
  <head>
     <title>Programmikoodi katsetus</title>
       <meta charset="UTF-8" />
       <script type="text/javascript">
          function tervita(){
               document.getElementById("vastus").innerHTML=
                 "Tere, "+document.getElementById("eesnimi").value;
            }
       </script>
  </head>
  <body>
     <h1>Katsetuste leht</h1>
       Palun sisesta oma eesnimi:
        <input type="text" id="eesnimi" />
        <input type="button" value="OK" onclick="tervita();" />
        <div id="vastus">
          Vastuse koht.
        </div>
  </body>
</html>
```

Katsetuse tulemus näha ka ekraanil.

Ülesanne

* Loo ekraanile kaks tekstivälja: üks eesnime, teine perekonnanime jaoks. Tervita kasutajat mõlemat nime pidi.

Tollid sentimeetriteks

Arvuti nimigi räägib, et masin võiks mõista arvutada. Tänapäeva programmeerimiskeelte puhul

tuleb vahel enne otsida kohta, kuidas masin arvutama panna, aga üldjuhul on see täiesti võimalik.

Järgnev lihtne näide arvutab tolle sentimeetriteks. Kordajaks 2,54 - ühele tollile vastab nõnda palju sentimeetreid. See tähendab, et toll on pikem ning sama pikkuse peale on tolle vähem. Programmeerimiskeeltes kasutatakse koma asemel punkti - see tava Inglise/Ameerika poolt tulnud. Kui kasutajaliideses mõnikord säetakse väljanägemine kasutaja asukohale vastavaks, siis programmeerimiskeele sees on ta ikka ühtmoodi.

Andmeid saab kasutajalt ikka samuti kätte. Tema sisestab numbrid tekstivälja. Tekstivälja küljes on id, mille järgi saab välja ja selle sees oleva väärtuse küsida. ning edasi võib teha juba sobivaid tehteid. Avaldis

```
document.getElementById("tollidearv").value*2.54+" cm";
```

teatab, et võetakse tekstikasti nimega "tollidearv" väärtus ja korrutatakse see 2.45-ga. Juurde liidetakse veel tühik ja cm, mis tähtedena kolmekesi jutumärkide vahel. Nõnda kasutajal pärast ekraanilt parem lugeda, et mida väljastatud vastus tähendab.

```
<!doctype html>
<html>
  <head>
    <title>Arvutamine</title>
        <meta charset="utf-8" />
        <script>
           function arvuta(){
              document.getElementById("vastus").innerHTML=
                    document.getElementById("tollidearv").value*2.54+" cm";
           }
        </script>
  </head>
  <body>
     <h1>Arvutamine</h1>
         <div>
           Mitu tolli:
           <input type="text" id="tollidearv" />
           <input type="button" value="Sisesta" onclick="arvuta()" />
         </div>
         <div id="vastus">

         </div>
  </body>
</html>
```

Tulemus ekraanil:

Ülesandeid

* Loo leht, millega arvutatakse sentimeetreid tollideks. Tolle on sentimeetritest 2.54 korda vähem.
* Koosta uuele lehele kalkulaator, kus poehinna puhul näidatakse, kui palju sellest moodustab käibemaks (20%). Vihje: kui hind letil on 1 euro ja 20 senti, siis 1 euro saab kaupmees ning käibemaksuks läheb 20 senti (20% suuruse maksu puhul).
* Paiguta lehele kaks tekstivälja. Ühte kirjutatakse toote hind, teise ostetav kogus. Väljastatakse vajalik summa.

Kolme arvu aritmeetiline keskmine

Näitena juurde veidi pikema arvutusvalemiga kalkulaator.

Hinna või tollide arvu leidmiseks tekstiväljast saadud väärtusi korrutades saab Javaskript ise aru, et neid tuleb arvudena käsitleda. Tähtede omavahelisest korrutamisest pole üldiselt erilist kasu ning seetõttu korrutamistehte peale muudetakse enne nad sisestatavast tekstist arvutatavateks arvudeks ning korrutamise peale saadakse mõistlik vastus kätte. Kui aga liitmiskäsuna kirjas plussmärk, siis ei tea arvuti, kas ta peaks panema tähti üksteise otsa või liitma leitud arvulisi väärtusi. Ehk siis kas 3+2 on 5 või hoopis 32. Javaskripti puhul valitakse tavajuhul kusjuures viimane vastus. Et arvuti teaks neid arvudena liita ja arvestada, et kolm pluss kaks on viis, selleks aitab käsklus parseInt.

```
var a1=parseInt(document.getElementById("arv1").value);
```

teatab, et tuleb võtta tekstivälja arv1 väärtus ning see parseInt käskluse abil mälus täisarvuks teisendada. Tulemus talletatakse muutuja ehk märksõna a1 alla. Sõna var seal ees näitab, et tegemist uue muutujaga. Nõnda saab ka teised väärtused tekstiväljadest arvuna kätte ning pärast seda võib valemitega toimetama hakata ja lõpuks tulemuse väljastada.

```
<!doctype html>
<html>
  <head>
    <title>Keskmiste arvutamine</title>
        <script>
          function arvuta(){
             var a1=parseInt(document.getElementById("arv1").value);
             var a2=parseInt(document.getElementById("arv2").value);
             var a3=parseInt(document.getElementById("arv3").value);
                 var aritmeetilineKeskmine=(a1+a2+a3)/3;
                 var tulemus="Aritmeetiline keskmine: "+aritmeetilineKeskmine;
                 document.getElementById("vastus").innerHTML=tulemus;
          }
        </script>
  </head>
  <body>
     <h1>Keskmised</h1>
         Arvud: <br />
         <input type="text" id="arv1" />
         <input type="text" id="arv2" />
         <input type="text" id="arv3" />
         <input type="button" value="Arvuta" onclick="arvuta()" />
         <div id="vastus">

         </div>
  </body>
</html>
```

Keskmised

Arvud:

5 | 5 | 2 | Arvuta

Aritmeetiline keskmine: 4

Kalkulaatorite ideid

Üheks Javaskripti levinud rakenduseks on mitmesugused veebilehel töötavad kalkulaatorid. Kui valem on teada, siis pole muud kui andmed sisse ja vastus käes. Et selline oskus edaspidi libedalt läheks, tasub ise vähemalt üks arvutusvalem välja mõelda või üles otsida ning selle kohta kalkulaator teha. Kui omal paremaid mõtteid ei tule, siis mõne idee saab julgesti siit.

- Bensiinimüük
- Kehamassi indeks
- Mõõtühikud - hulgiteisendus
- Vahemaa läbimiseks kuluv aeg
- Söögi kalorsus / toitainete sisaldus
- Elektrienergia arvutused
- Keskmiste arvutamine (aritmeetiline/geomeetriline/mediaan)
- Palgakalkulaator (üldsumma, bruto, neto, sotsiaalmaks, töötuskindlustus)
- Ruutvõrrandi nullkohad (ruutjuure käsk Math.sqrt())
- Kahe muutujaga lineaarvõrrandsüsteemi lahend

Joonistamine

Pilte näitasid veebilehed üsna veebi algusaegadest peale (1990ndatel) Kasutaja soovidest sõltuvate jooniste või animatsioonide loomiseks on aga tulnud mitmesuguseid lisavidinaid kasutada. Varakult tekkisid Java rakendid, siis Flashi abivahendid ning seejärel Silverlighti lisandprogramm.Samuti sai Javaskripti raamistikega nurgataguseid kasutades luua ekraanile mulje joonise tekkimisest - ehkki joone tõmbamiseks näiteks mõnikord tekitati suurel hulgal ruudukujulisi väikesi tekstilõike, värviti lõigukeste taustad ära ning paigutati nõnda, et see kasutajale joonena paistab. Arvestades aga, et veebi loetakse väga mitmesuguste tehniliste võimaluste ja seadistustega masinatega, siis nõnda paljude ja küllalt tugevat riistvara nõudvate tarkvaralahenduste korraga kättesaadavana hoidmine ei kipu õnnestuma. Ikka tuleb teateid, kuidas üks või teine brauser või operatsioonisüsteem jälle mõne lahendusega hakkama ei saa.

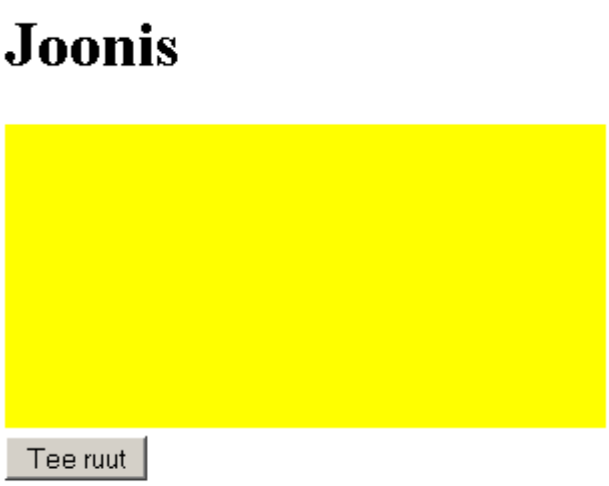
HTML-i viiendat versiooni kavandades otsustati siia vahenditesse sisse panna võimalikult palju hädatarvilikku, mille abil saaks enamiku veebilehtede loomisel tekkivaid soove ära rahuldada. Nõnda on HTML5 igakülgne näitamine seadmele küll veidi keerulisem kui mõne varasema versiooni korral, samas tootjatel on siisk tunduvalt lihtsam teha seadmeid ja tarkvara HTML5 näitamiseks, kui et arvestada laia komplekti mitmesuguste lisandprogrammidega.

Ruut ekraanil

Joonistamiseks ja liigutamiseks lisati HTML5 vahendite hulka element nimega canvas (lõuend). Selle peale on võimalik Javaskripti abil joonistada. Samuti korduva joonistamise/kustutamise abil liikumine tekitada. Lehele saab ta tekitada sarnase käsuga.

```
    <canvas id="tahvel" width="300" height="200"  style="background-
color:yellow"></canvas>
```

Suurus ja taust talle määratud selleks, et lehel selgemalt näha oleks, kus lõuend paistab. Ning id järgi saab hiljem programmikoodiga lõuendi poole pöörduda, et sinna miskit joonistama hakata.



Joonistamise tarbeks siin väike alamprogramm. Tahvlile joonistamise tarbeks tuleb talle kõigepealt ligipääs küsida.

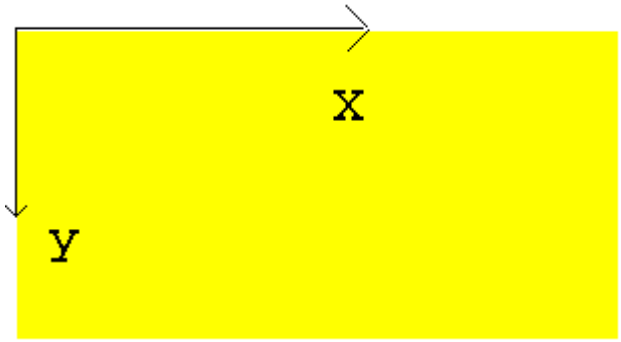
```
        var g=document.getElementById("tahvel").getContext("2d");
```

Edasi võib siis juba joonistuskäskude abil pilti ekraanile kuvama asuda.
Lihtsaimaks näiteks siin ristküliku loomine

```
        g.fillRect(20, 20, 50, 50); //x, y, laius, kõrgus
```

Arvutil programmi abil joonistades tuleb kõik asjad koordinaatide järgi kätte anda, nii ka ristküliku puhul. Programmeerimisgraafika omapäraks on, et koordinaatide nullpunkt asub vasakul ülanurgas. Sealt liigub x-telg paremale ning y-telg alla.

Joonis



Nõnda kokkupandud koodinäite puhul peaks nupule vajutades ekraanile ruut tekkima.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Joonis</title>
    <script type="text/javascript">
      function joonista(){
         var g=document.getElementById("tahvel").getContext("2d");
         g.fillRect(20, 20, 50, 50); //x, y, laius, kõrgus
      }
    </script>
  </head>
  <body>
    <h1>Joonis</h1>

    <canvas id="tahvel" width="300" height="200"
            style="background-color:yellow"></canvas><br />
    <input type="button" value="Tee ruut" onclick="joonista()" />
  </body>
</html>
```

Joonis

Tee ruut

Ülesandeid

* Tutvu ristküliku joonistamise näitega
* Muuda ristküliku suurust ja asukohta
* Tee ekraanile kaks ristkülikut
* Lao ekraanile ristkülikutest koosnev alt laiem torn.

Mõõtude järgi ristkülik

Niisama pildi kuvamiseks pole enamasti vaja programmi kirjutada. Saab valmis pilti näidata või siis seda sobival hetkel vahetada. Kui tahta aga, et kasutaja saaks joonise mõõtmeid muuta, siis võib mõnest koodireast sealjuures kasu tulla. Lihtsamatel juhtudel on küll võimalik sobivad pildid enne valmis teha ning siis vastavalt sisestatud andmetele kasutajale näidata. Kui aga võimalusi palju, siis tuleb ikka vaadata, et mida ja kuidas joonistada.

Andmete sisestamise puhul sobivad samasugused tekstiväljad kui aruvutamise osa juures.

```
    <input type="text" id="laiuskast"  value="40" />
```

```
Nõnda luuakse tekstiväli. Pöördumise jaoks on tal id nimega laiuskast ning
algväärtuseks pannakse sisse talle 40.
```

Kuna arvuti jaoks on tähed ja arvud erinevad, siis tuleb andmetüüp sisselugemisel sobivaks sättida.

```
        var laius=parseInt(document.getElementById("laiuskast").value);
```

tähendab lahtiseletatult, et küsitakse documendi seest laiuskasti-idga element ja sealt tema tekstiline väärtus. Käsklus parseInt muundab teksti arvuks - see tähendab, et 40 pole enam mitte tähed neli ja null, vaid arv nelikümmend, millega soovi korral liita/lahutada saab ning mida võib võtta ja joonise sisendväärtusena. Võrdusmärgi ees olev var laius ütleb, et väljaarvutatud väärtus jäetakse meelde märksõna laius alla ning sealtkaudu on seda võimalik hiljem pruukida. Joonistamise juures lähebki neid kohe tarvis.

```
        g.fillRect(30, 30, laius, korgus); //x, y, laius, kõrgus
```

joonistab etteantud laiuse ja kõrgusega ristküliku. Esimesed 30 ja 30 tähistavad vastavalt kujundi kaugust tahvli vasakust ning ülemisest servast.

Allpool on näidatud ka, kuidas teksti ekraanile kuvada.

```
       g.fillText(korgus, 50, 20);
```

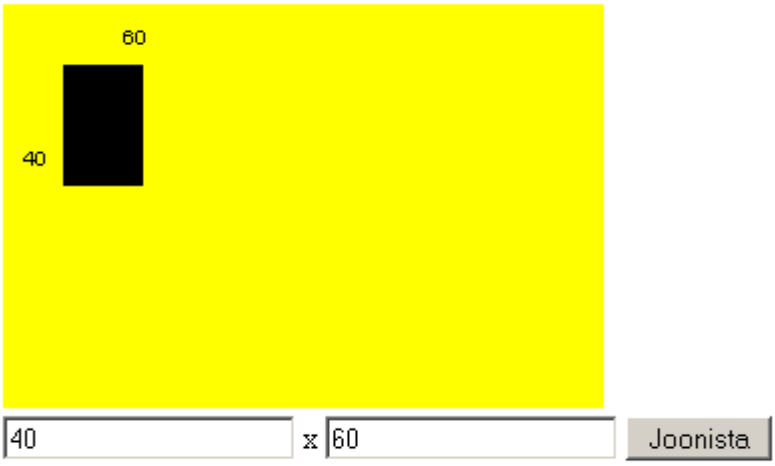
kuvab välja märksõnas ehk muutujas "korgus" oleva väärtuse, paigutades selle alguse tahvli vasakust servast 50 ning ülemisest 20 punkti kaugusele.

Käivitav kood tervikuna

```
<!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);
         g.fillRect(30, 30, laius, korgus); //x, y, laius, kõrgus
         g.fillText(korgus, 60, 20);
         g.fillText(laius, 10, 80);
      }
    </script>
  </head>
  <body>
    <h1>Joonis</h1>
    <canvas id="tahvel" width="300" height="200"
              style="background-color:yellow"></canvas><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>
```

Ning selle järgi valminud joonis.



Ülesandeid

* Tutvu ristküliku mõõtmete määramise näitega.
* Võimalda lisaks mõõtmetele määrata ka ristküliku asukoht ekraanil.
  (Eraldi tekstikastid vasaku ülanurga x- ja y-koordinaadi määramiseks,
  kogu joonistamine jääb ühe funktsiooni sisse.)

* Joonista lehele kaks ristkülikut, kusjuures kummalgi saab määrata vaid laiust.
  Ristkülikud asuvad üksteise all, ning nende vasakud servad kohakuti.

* Joonista lehele kaks ristkülikut, kusjuures kummalgi saab määrata vaid kõrgust. Ristkülikute ülaservad on kohakuti.

* Säti arvutused nõnda, et ristkülikute alaservad on samal kõrgusel,
  pikem ristkülik ulatub ülevalt kõrgemale.

Joon

Joon paistab olema joonistamisele nime andnud. Joone ekraanile kuvamiseks paistab vaja olema veidi rohkem toimetusi kui ristküliku korral, kuid ei midagi võimatut. Joon on tema jaoks joonistustee (Path) osa, kus tuleb üksikuid punkte märkida. Programmikoodiga tuleb käituda, nagu juhitaks pliiatsiotsa. Käsk moveTo ütleb, kuhu koordinaatidele (x ja y) pliiats tõsta, edasine lineTo teatab kuhuni joon tõmmata. Lõpus olev stroke() palub tulemuse ekraanile kuvada.

```
g.beginPath();
g.moveTo(30, 40);
g.lineTo(80, 90);
g.stroke();
```

Et siin näites tehakse alati sama joon ning kasutaja joone andmeid muuta ei saa, pole talle ka joonistamiseks nuppu vaja. Joonistuskäsk pannakse käima kohe, kui leht on kohale jõudnud ja veebilehitsejas ette kuvatud. Ehk siis lehe sisuosa body sündmuse onload peale.

```
  <body onload="joonista()">
```

Muus osas võib jälle rahulikult jälgida, kuidas kood kokku pandud

```
<!DOCTYPE html>
<html>
  <head>
    <title>Joonis</title>
    <script type="text/javascript">
      function joonista(){
         var g=document.getElementById("tahvel").getContext("2d");
         g.beginPath();
         g.moveTo(30, 40);
         g.lineTo(80, 90);
         g.stroke();
      }
    </script>
  </head>
  <body onload="joonista()">
    <h1>Joonis</h1>
    <canvas id="tahvel" width="300" height="200"
              style="background-color:yellow"></canvas><br />
  </body>
</html>
```

Edasi koodi põhjal tekkinud joont imetleda ning selle pealt juba edasisi arendusmõtteid mõlgutada.

Joonis

Ülesandeid

* Tutvu joone tõmbamise näitega
* Tõmba joon ülalt alla
* Tõmba joon vasakult paremale
* Tõmba kolm kõrvuti asetsevat joon
* Joonista kuubikust maja, millel oleks joone abil tõmmatud viilkatus peal.

Joone värv ja paksus

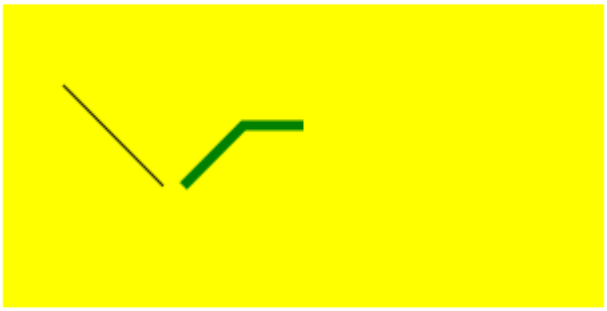
Lihtsalt tõmmatud joon on ühe punkti laiune ning musta värvi. Arvutisse aga tahame vahel ka mitmekesisemaid pilte. Iga Path-i abil tõmmatud joon on oma pikkuses samasuguste omadustega. Eri joonistusteedeks jagatud jooned aga saavad olla igaüks omamoodi. Joone paksuse määrab tunnus lineWidth, joone värvi strokeStyle. Nii on tehtud all olevas näites.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Joonis</title>
    <script type="text/javascript">
      function joonista(){
         var g=document.getElementById("tahvel").getContext("2d");
         g.beginPath();
         g.moveTo(30, 40);
         g.lineTo(80, 90);
         g.stroke();

         g.beginPath();
         g.lineWidth=5;
         g.strokeStyle="green";
         g.moveTo(90, 90);
         g.lineTo(120, 60);
         g.lineTo(150, 60)
         g.stroke();
      }
    </script>
  </head>
  <body onload="joonista()">
    <h1>Joonis</h1>
```

```
    <canvas id="tahvel" width="300" height="200"
                style="background-color:yellow"></canvas><br />
  </body>
</html>
```

Tulemus näha ka joonisel.

Joonis

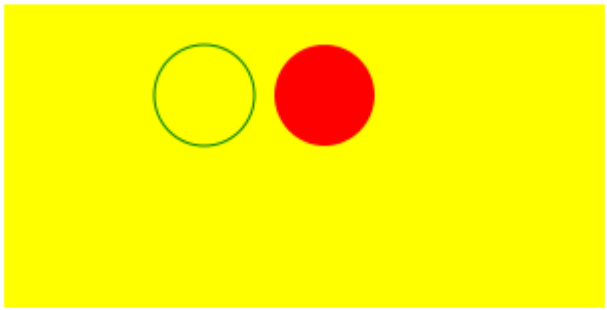


Ringid

Ümaramate kujundite juures aitavad meid ringid - nii seest täidetud kujul kui ringjoonena. Üllatusena küll Javaskriptil lihtsalt ringi loomise käsklust ei ole. Küll aga on universaalne käsklus kaare loomiseks. Kui aga kaare pikkuseks on täisring ehk 360 kraadi ehk 2Pi radiaani, siis näemegi tulemust ringina. Programmeerimise juures tuleb mitmel puhul vajalikke tulemusi kombineerida kättesaadavate võimaluste abil, nii ka siin.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Joonis</title>
    <script type="text/javascript">
      function joonista(){
         var g=document.getElementById("tahvel").getContext("2d");
         g.strokeStyle="green";
         g.beginPath();
         g.arc(100, 45, 25, 0, 2*Math.PI, true);
         g.stroke();

         g.fillStyle="red";
         g.beginPath();
         g.arc(160, 45, 25, 0, 2*Math.PI, true);
         g.fill();

      }
      window.onload=joonista;
    </script>
  </head>
  <body>
    <h1>Joonis</h1>
    <canvas id="tahvel" width="300" height="200"
                style="background-color:yellow"></canvas><br />
  </body>
</html>
```

Joonis



Nõnda on põhiliste kujundite joonistamise oskus olemas. Edasine sõltub suurelt jaolt juba oma fantaasiast ja ekraanipunktide lugemise jaksust. Nõnda saab mitmesuguseid jooniseid luua, kus kasutajalt tulevad andmed sisse ning nende põhjal tehakse tema soovidele vastav joonis.

Ülesandeid

* Pane ringi joonistamise näide käima
* Joonista valgusfoor
* Kasutaja saab nupule vajutamisega määrata, milline tuli fooris põleb
* Joonista automudel, mille pikkust ja kõrgust saab kasutaja määrata

Hiir

Arvuti ilma hiireta on nagu kevad ilma kuldnokkadeta - selline on ühe arvutigraafika õpetaja ütlus ning mitut pidi paistab tal õigus olema. Midagi vähegi liikuvamat ette võttes aitab hiir elu märgatavalt mugavamaks teha.

Hiire koordinaatide püüdmine

Lihtsaim näide hiire toimimise kohta võiks olla järgmine: Tahvli külge kinnitatakse sündmus onmousedown, mille peale palutakse käivitada funktsioon nimega vajutus. Parameetriks antud muutuja event on kasutada veebilehe elementide juures ning sealtkaudu saab lugeda välja hiire asukoha ekraanil. Y-koordinaadiks siis e.clientY. Vastuskihil on algul lihtsalt kihi asukohta tähistav v-täht, kuid pärast hiirevajutust kuvatakse sinna hiire Y-koordinaat.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Joonis</title>
    <script type="text/javascript">
      function vajutus(e){
```

```
          document.getElementById("vastus").innerHTML=e.clientY;
        }
      </script>
    </head>
    <body>
      <h1>Joonis</h1>
      <canvas id="tahvel" width="300" height="200"  style="background-
color:yellow" onmousedown="vajutus(event)"></canvas>
      <div id="vastus">
        v
      </div>
    </body>
</html>
```

Siin korral vajutati nõnda, et y-i koordinaadiks tuli 169.

169

Ülesandeid

* Tutvu hiire abil y-koordinaadi näitamise võimalusega

* Lisa ekraanile nähtavaks ka x-koordinaat
* Asenda onmousedown-sündmus onmousemove'ga,
  nii et koordinaatide muutumine oleks
  reaalajas näha.

Ristkülik hiire kohale

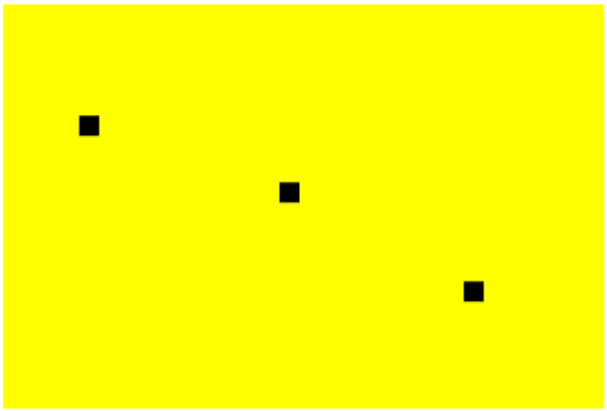
Järgmise näitena püütakse saada hiire vajutuse kohale ristkülik. Selleks on vaja teada hiire koordinaate tahvli suhtes - kättesaadav clientX/clientY annab need aga akna suhtes. Päästjaks on tahvli asukohta küsiv funktsioon getBoundingClientRect(). Suurema lehe puhul lisanduks siia veel arvutus lehe enese kerimise kohta. Kui aga leht nõnda väike et seda kerima ei hakata, siis saab veidi lihtsamalt läbi. Arvutus

```
          var hx=e.clientX-tahvlikoht.left;
```

annab hiire asukoha tahvli suhtes. Selle järgi saab hiljem ristküliku või muu kujundi ka soovitud kohta joonistada.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Joonis</title>
    <script type="text/javascript">
      function vajutus(e){
         var tahvlikoht=document.getElementById("tahvel").
                                          getBoundingClientRect();
         var g=document.getElementById("tahvel").getContext("2d");
         var hx=e.clientX-tahvlikoht.left;
         var hy=e.clientY-tahvlikoht.top;
         g.fillRect(hx, hy, 10, 10);
      }
    </script>
  </head>
  <body>
    <h1>Joonis</h1>
    <canvas id="tahvel" width="300" height="200"
         style="background-color:yellow" onmousedown="vajutus(event)"></canvas>
  </body>
</html>
```

Pärast vajutusi võib imetleda nende kohtadele tekkinud ristkülikuid.



Ülesandeid

* Muuda ruudu joonistamise näidet nõnda, et ruut tekib onmousemove-sündmuse puhul, st. koos hiire liikumisega tekivad ekraanile ruudud.

  Otsi varasematest näidetest tahvli puhastamise näide

```
g.clearRect(0, 0, 300, 200);
```

* Hoolitse, et ruut liiguks ekraanil hiirega kaasa. St. iga liikumissündmuse peale puhastatakse tahvel ning joonistatakse ruut uues kohas.

Juhuarv

Arvutite eeliseks peetakse võimet korduvalt samade lähteandmete juures samadele tulemustele jõuda. Tavaolukorras see äratab usaldust ning ka näiteks vigu on parem leida, kui on teada, et alati samal moel valesti arvutatakse. Mõnikord aga tuleb vaheldus kasuks. Olgu siis enese treenimiseks uute ülesannetega, mängule uue algseisu välja mõtlemisel või lihtsalt ekraanisäästjal uute kujundite välja mõtlemiseks. Kõik need vaheldused saab programmikoodi luua juhuarvude abiga. Javaskriptis saavad nad alguse käsust `Math.random()` mis loob arvu nulli ja ühe vahelt, kuid mis on alati väiksem kui 1. Järgmine väike näide tekitab sellise arvu ekraanile.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Joonis</title>
    <script type="text/javascript">
      function algus(){
         document.getElementById("vastus").innerHTML=Math.random();
      }
    </script>
  </head>
  <body onload="algus()">
    <div id="vastus">
      v
    </div>
  </body>
</html>
```

Tulemuseks sai sel korral imetleda arvu

0.282310351980924

Hulgaliselt komakohti võib küll olla ilus vaadata, aga kujundeid ekraanile paigutades või loetelust sobivat anektdooti otsides läheb vaja arve enamasti suuremas vahemikus kui nulli ja ühe vahel ning mõnigikord võiksid nad olla täisarvud. Siin näites siis korrutatakse random-funktsioonist tulnud arv kümnega ning käsu parseInt abil võetakse sellest täisosa. Tulemuseks on kümnest väiksem täisarv

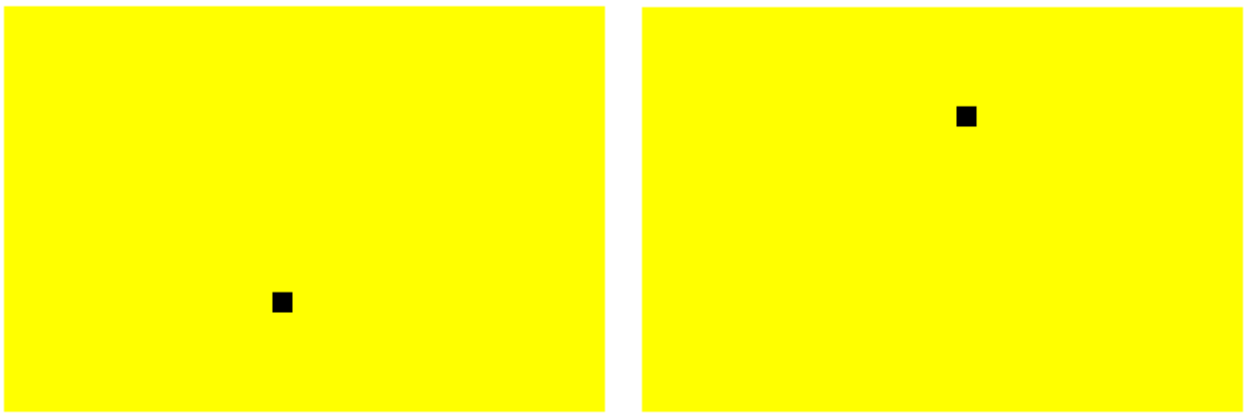
```
<!DOCTYPE html>
<html>
  <head>
    <title>Joonis</title>
    <script type="text/javascript">
      function algus(){
              //arv 0..9
         document.getElementById("vastus").innerHTML=parseInt(10*Math.random());
      }
    </script>
  </head>
  <body onload="algus()">
    <div id="vastus">
      v
    </div>
  </body>
</html>
```

Praegusel korral juhtus siis tulema neli:

Loodud arve võib edasi juba igal pool omal valikul kasutada. Siin näites arvutatakse juhuarvud selliselt, et nende järgi paigutatud ruut leiab omale koha tahvli peal. Varemgi tuttavaks saanud `getBoundingClientRect()`-funktsioon annab tagasi objekti, mille käest võimalik küsida tahvli servade koordinaadid - vastavalt siis `tahvlikoht.left`, `tahvlikoht.right`, `tahvlikoht.top`, `tahvlikoht.down`. Arvutus `tahvlikoht.right-tahvlikoht.left` annab tahvli laiuse ning `tahvlikoht.bottom-tahvlikoht.top` tahvli kõrguse. Nende järgi saab siis paigutatavale ruudule sobiva asukoha leida.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Joonis</title>
    <script type="text/javascript">
      function algus(){
             var tahvlikoht=document.getElementById("tahvel").
                                        getBoundingClientRect();
         var g=document.getElementById("tahvel").getContext("2d");
             var tahvlilaius=tahvlikoht.right-tahvlikoht.left;
                 var tahvlikorgus=tahvlikoht.bottom-tahvlikoht.top;
         rx=parseInt(tahvlilaius*Math.random());
                 ry=parseInt(tahvlikorgus*Math.random());
         g.fillRect(rx, ry, 10, 10);
      }
    </script>
  </head>
  <body onload="algus()">
    <h1>Joonis</h1>
    <canvas id="tahvel" width="300" height="200"
      style="background-color:yellow" ></canvas>
  </body>
</html>
```

Igal lehe avamisel võiks ruut tahvlil siis uude kohta tekkida.



Ülesandeid

* Pane tahvlil päike tekkima ülaservas juhuslikku kohta

* Lisa joontest pargipink. Pingi pikkus sõltub juhuslikust arvust.

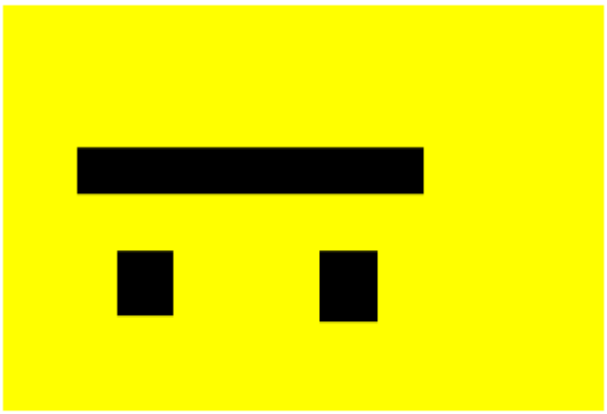
Hiirega kujundi suuruse määramine

Vaid allavajutustele reageerides võib küll määrata joonistamise või muu jaoks asukohti, kuid puudu jäävad suurus ja/või suund. Eks mitme vajutuse peale neid andmeid kokku korjates ole võimalik ka nii tulemust sättida, aga üheks mugavaks lahenduseks on hiire alla- ja ülesliikumise andmed mõlemad kokku koguda ning siis kokku kahe punkti koordinaatide abil arvutama asuda.

Et allavajutuspunkti koordinaadid meeles püsiksid, selleks sobib nad väjaspool funktsioone deklareerida ehk nende olemasolust teada anda. Funktsioonis hiirAlla arvutatakse nende väärtused ja jäetakse meelde. Hiljem hiirYles juures on need kaks arvu juba mälust võtta. Tuleb lihtsalt edasi mõelda, mida kokku käepärast oleva nelja arvuga ette võtta.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Joonis</title>
    <script type="text/javascript">
          var algusx, algusy;

      function hiirAlla(e){
         var tahvlikoht=document.getElementById("tahvel").
                                        getBoundingClientRect();
         algusx=e.clientX-tahvlikoht.left;
         algusy=e.clientY-tahvlikoht.top;
      }

      function hiirYles(e){
         var tahvlikoht=document.getElementById("tahvel").
                                          getBoundingClientRect();
         var g=document.getElementById("tahvel").getContext("2d");
         var loppx=e.clientX-tahvlikoht.left;
         var loppy=e.clientY-tahvlikoht.top;
         var laius=loppx-algusx;
                 var korgus=loppy-algusy;
         g.fillRect(algusx, algusy, laius, korgus);
      }
    </script>
  </head>
  <body>
    <h1>Joonis</h1>
    <canvas id="tahvel" width="300" height="200"
          style="background-color:yellow"
          onmouseup="hiirYles(event)" onmousedown="hiirAlla(event)" ></canvas>
  </body>
</html>
```

Siin näites tõmmatakse nende põhjal ekraanile ristkülik - hiirega vedades saab neid kergelt ka mitu tükki luua.



Ülesandeid

* Tutvu hiire abil ristküliku loomise näitega
* Võimalda sarnaselt paigutada ning suurust määrata kolmest ristkülikust koosnevalt tornil.
* Hiirega saab ekraanile joonistada ringist ja joonest koosnevaid puid.

Asukoha meelespidamine

Siiani võtsime joonistamisel koordinaadid otse hiire käest või lahtritesse kirjutatud väärtustest. Vähegi suurema süsteemi puhul aga on igasugu kujundite kohad ja andmed pigem programmi sees meeles. Nende andmete järgi arvutatakse objektide asendeid ja kokkupuutumisi. Ning samuti joonistatakse nende põhjal pärast ekraanile pilt.

Siin näites tehakse selle suhtes algust ühe lihtsa ruuduga. Üleval rida

```
var x=20, samm=5;
```

teatab, et hoitakse meeles ruudu asukohta x-koordinaat, algväärtuseks tal 20. Ning et ruut liigub iga sammu puhul viie ekraanipunkti jagu soovitud suunas.

Edasi juba eraldi funktsioonid ehk alamprogrammid liikumiste tarbeks, kus siis kõigepealt arvutatakse välja uus asukoht ning edasi palutakse tulemus ekraanile joonistada.

```
function vasakule(){
  x=x-samm;
  joonista();
}
```

Joonistamiseks omaette funktsioon. Kõigepealt küsitakse juurdepääs joonistustahvlile ja sealtkaudu sinna joonistamiseks vajalikule graafilisele kontekstile.

```
function joonista(){
   var t=document.getElementById("tahvel");
   var g=t.getContext("2d");
   g.clearRect(0, 0, t.width, t.height);
   g.fillRect(x, 20, 50, 50); //x, y, laius, kõrgus
}
```

Muutujas nimega t seetõttu andmed tahvli kohta: t.width näitab tahvli laiust ning t.height tahvli kõrgust.

```
g.clearRect(0, 0, t.width, t.height);
```

tühjendab tahvli taustavärviga kogu ulatuses. Alates siis vasakust ülanurgast (0,0) kuni parema alanurgani (t.width, t.height). Edasi joonistatakse ristkülik juba soovitud kohta

```
g.fillRect(x, 20, 50, 50); //x, y, laius, kõrgus
```

Y-koordinaat ja mõõtmed esiotsa koodi sisse otse kirjutatud, x-i asukoht loetakse vastavast muutujast, mille väärtust siis vasakule/paremale käskude abil muudetakse.

Skripti lõpuosas olev

```
window.onload=joonista;
```

tähendab sama, kui ennist kirjutasime HTMLi sisse

```
<body onload="joonista()">
```

Ehk siis antakse teada, et pärast lehe täielikku laadimist pannakse joonistuskäsklus käima.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Joonis</title>
    <script type="text/javascript">
      var x=20, samm=5;
      function joonista(){
         var t=document.getElementById("tahvel");
         var g=t.getContext("2d");
         g.clearRect(0, 0, t.width, t.height);
         g.fillRect(x, 20, 50, 50); //x, y, laius, kõrgus
      }
      function vasakule(){
        x=x-samm;
        joonista();
      }
      function paremale(){
        x=x+samm;
        joonista();
      }
      window.onload=joonista;
    </script>
  </head>
  <body>
    <h1>Joonis</h1>
    <canvas id="tahvel" width="300" height="200"
          style="background-color:yellow" ></canvas><br />
    <input type="button" value="v" onclick="vasakule()" />
    <input type="button" value="p" onclick="paremale()" />
  </body>
</html>
```

Joonis

v p

Joonis

v p

Ülesandeid

* Lisa nupud ruudu liigutamiseks üles ja alla
* Lisa nupud ruudu suuruse muutmiseks
* Jäta ära lehe vahepealse tühjendamise käsklus. Nii saab nuppude abil liigutatava ristküliku abil joonistada.

* Paiguta uuele lehele kaks ristkülikut. Kumbagi saab nuppudega liigutada.

Liikumine

Eelnevas näites liigutasime nupuvajutusel ühe sammu kaupa. Samuti sai varem panna kujundi hiire järgi liikuma. Kui aga panna sammud iseenesest ja piisava sagedusega korduma, siis kasutajale jääb mulje liikumisest ilma, et selle jaoks peaks ta midagi ise pidevalt tegema.

Korduvalt käivituv käsk

Tehniliselt saab käsu korduvalt käivitumise panna kirja kujul

```
window.onload=setInterval("paremale()", 500);
```

Lahtiseletatult pannakse funktsioon `paremale()` tööle iga viiesaja millisekundi tagant ehk praegusel juhul kaks korda sekundis. Nõnda siis liigubki ruut me silmade ees vaiksete hüpetega edasi. Ooteaega vähendades või sammu pikkust suurendades saab liikumist sujuvamaks või kiiremaks muuta. Samas seab masina tehniline võimsus piltide joonistamise sagedusele piirid. Enamasti ei tasu veidigi suurema joonistustahvli korral loota ülejoonistusajale alla 100 millisekundi.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Joonis</title>
    <script type="text/javascript">
      var x=20, samm=5;
      function joonista(){
         var t=document.getElementById("tahvel");
         var g=t.getContext("2d");
         g.clearRect(0, 0, t.width, t.height);
         g.fillRect(x, 20, 50, 50); //x, y, laius, kõrgus
      }
      function vasakule(){
        x=x-samm;
        joonista();
      }
      function paremale(){
        x=x+samm;
        joonista();
      }
      window.onload=setInterval("paremale()", 500);
    </script>
  </head>
  <body>
    <h1>Joonis</h1>
    <canvas id="tahvel" width="300" height="200"
          style="background-color:yellow"></canvas><br />
    <input type="button" value="v" onclick="vasakule()" />
    <input type="button" value="p" onclick="paremale()" />
  </body>
</html>
```

Joonis

Ülesandeid

* Ruut liigub vasakult paremale, klahvidega saab ruutu samal ajal liigutada alla ja üles
* Korraga liigub kaks ruutu - üks paremale, teine vasakule
* Vasakule liikuva ruudu suurus liikumise ajal väheneb
* Sammu pikkus ja sellega koos liikumise kiirus valitakse juhuslikult

Liikumise suuna määramine

Siiani liikus ruut ühes suunas, suuna määras muutuja nimega samm. Kui sellele sammule anda negatiivne väärtus, siis liigub ruut vasakule. Liikumissuuna muutmiseks rakenduse töö ajal tuleb ka sammu väärtust muuta. Eraldi lisati muutuja nimega kiirus, mille abil võimalik määrata, kui ruttu ruut liigub. Suuna määramiseks tuleb kiiruse väärtus lihtsalt sammule üle kanda - olgu siis pluss- või miinusmärgiga. Ja kui tahta ruut seisma jätta, piisab, kui sammu pikkuseks sättida null.Alla siis vastavad nupud, et kasutaja võiks vastavaid käsklusi käivitada.

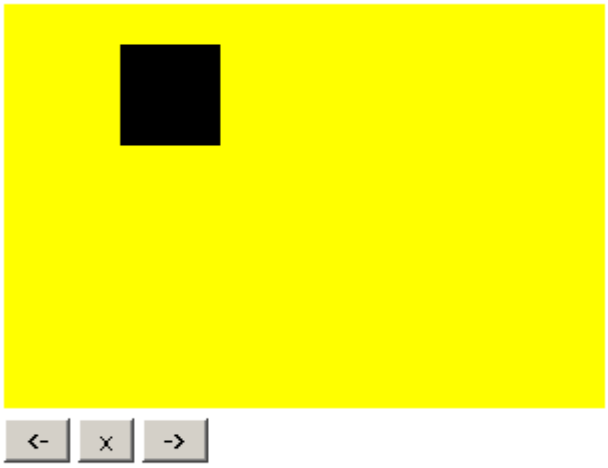
Nuppude väärtuseks olevad `<-` ja `->` võivad tunduda raskelt mõistetavad.

```
<input type="button" value="&lt;-" onclick="vasakule()" />
<input type="button" value="x" onclick="seis()" />
<input type="button" value="-&gt;" onclick="paremale()" />
```

Tegelikult aga tegemist erisümbolitega, mille abil saab sinna noole kujutise kuvada. `<` tähendab "less than" ehk "väiksem kui" ehk siis sümbol "<". Teistpidine `>` on "greater than" ehk "suurem kui" ehk ">". Kuna need märgid niisamuti kirjutatuna tähistavad HTML koodis erisümboleid, siis ei või neid otse nuputeksti kohale kirjutada, et saada noole kuju <- tekitada. Asenduse abil aga õnnestub ning nupu silt täiesti loetav.

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

```
            var x=20, kiirus=2, samm=kiirus;
            function joonista(){
              var t=document.getElementById("tahvel");
              var g=t.getContext("2d");
              g.clearRect(0, 0, t.width, t.height);
              g.fillRect(x, 20, 50, 50);
            }
            function liigu(){
              x=x+samm;
              joonista();
            }
            function vasakule(){
              samm=-kiirus;
            }
            function seis(){
              samm=0;
            }
            function paremale(){
              samm=kiirus;
            }
        </script>
  </head>
  <body onload="setInterval('liigu()', 100)">
      <canvas id="tahvel" width="300" height="200"
                  style="background-color:yellow"></canvas><br />
      <input type="button" value="&lt;-" onclick="vasakule()" />
      <input type="button" value="x" onclick="seis()" />
      <input type="button" value="-&gt;" onclick="paremale()" />
  </body>
</html>
```



Ülesandeid

* Pane näide käima, uuri. Muuda liikumise kiirust
* Võimalda ruudul liikuda vasakule-paremale suuna asemel üles-alla
* Loo nupud, et saaks määrata liikumise kõigi nelja ilmakaare suunas.

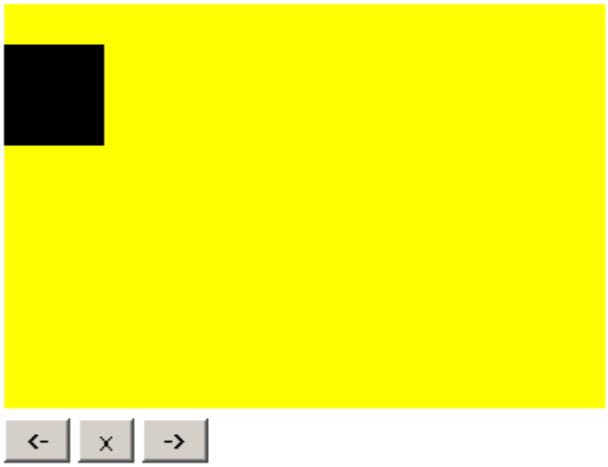
Seiskumine vasakus servas

Nõnda järjest liikudes on ruut küllalt varsti ekraanilt läinud. Ise pidevalt juhtides saab usinasti liikumist sättida. Kui aga tegemist hiljem rakendusega, kus korraga enam kujundeid liikumas, siis ei saa kõigil pidevalt käsitsi silma peal pidada ja kontrollida, et kuidas keegi käitub. Kontrollimiseks sobib tingimuslause nimega if. Praegusel juhul, kui ruut liigub liiga vasakule ehk x läheb negatiivseks, siis paras aeg öelda, et ta seisma jääks. Ning kasulik on ruudule märkida ka, et ta serva juurde tagasi tuleks (x-i väärtuseks arv 0).

```
            if(x<0){seis(); x=0;}
```

Siis pääseb ruut näiteks paremale suunava nupu vajutamisel taas liikuma. Muidu tekiks imelik olukord, kus pole võimalik ka üle ääre sattununa sealt tagasi minna, sest nullist väiksema iksi puhul antakse kohe seiskumiskäsklus.

```
<!doctype html>
<html>
  <head>
    <title>Liikumine</title>
        <script>
           var x=100, kiirus=2, samm=-kiirus;
           function joonista(){
             var t=document.getElementById("tahvel");
             var g=t.getContext("2d");
             g.clearRect(0, 0, t.width, t.height);
             g.fillRect(x, 20, 50, 50);
           }
           function liigu(){
             if(x<0){seis(); x=0;}
             x=x+samm;
             joonista();
           }
           function vasakule(){
             samm=-kiirus;
           }
           function seis(){
             samm=0;
           }
           function paremale(){
             samm=kiirus;
           }
        </script>
  </head>
  <body onload="setInterval('liigu()', 100)">
      <canvas id="tahvel" width="300" height="200"
                  style="background-color:yellow"></canvas><br />
      <input type="button" value="&lt;-" onclick="vasakule()" />
      <input type="button" value="x" onclick="seis()" />
      <input type="button" value="-&gt;" onclick="paremale()" />
  </body>
</html>
```



Ülesandeid

* Tee näide läbi
* Pane ruut liikuma alt üles. Ruut seiskub ülesse serva jõudmisel
* Ruutu saab liigutada iga ilmakaare suunas. Seiskub nii vasakusse kui ka ülemisse serva jõudmise puhul.

Seespüsimise kontroll

Vasaku ja ülemise serva tabamise kontroll on suhteliselt lihtne vähemasti ruudu puhul. Kuna ruut joonistatakse vasaku ülemise nurga koordinaatide järgi, siis juhul, kui need on nullist väiksemad, siis järelikult ollakse servast üle läinud. Parema ja alumise poolega on veidi rohkem arvutamist. Parema serva koordinaadi saab kätte liites vasaku serva koordinaadile ruudu laiuse. Kui nüüd selle parema serva koordinaadi väärtus ületab tahvli laiust ehk tahvli parema serva koordinaadi väärtust, sellisel juhul on ruut sealtpoolt üle läinud.

Iseenesest on võimalik nõnda tingimusi sättides ruudu seiskamist täiesti kontrollida. Arvestades aga tulevikku ning võimalust, et kontrollitavaid kujundeid tuleb rohkem, siis mitmekülgsema kontrolli tarbeks on hea teha omaette funktsioon. Näiteks selline:

```
function kasSees(uusX){
  if(uusX<0){return false;}
  if(uusX+laius>t.width){return false;}
  return true;
}
```

Funktsiooni nimeks on kasSees - nagu nimest aimata võib, siis sealt vastatakse kas jah või ei. Ehk siis true või false. Selle järgi siis võimalik hiljem otsustada kuidas käituda - kui ruudu uuritav asukoht on tahvli sees, võib sinna julgelt asuda. Kui mitte, ei tasu sinna ronida. Funktsiooni ümarsulgude sees olev `uusX` tähendab, et funktsioonile etteantav väärtus nimetatakse funktsiooni sees kasutamiseks `uusX`-i nime alla ning sealtkaudu saab tema sisu pruukida. Edasi juba järgnevad kontrollid ja vastused.

```
  if(uusX<0){return false;}
```

teatab, et kui etteantud parameetri (`uusX`) väärtus on väiksem kui null, siis funktsioon tagastab

(`return`) false, ehk siis ruut pole tervikuna tahvli sees. Sama lugu parema serva kontrolliga. Muutujas t meil eelnevalt tahvli andmed olemas. Seega siis avaldis

```
uusX+laius>t.width
```

kontrollib, et kas ruudu parema serva koordinaadi väärtus (`uusX+laius`) ületab tahvli laiust (`t.width`).

Kui mõlemad kontrollid edukalt läbitud, siis järelikult ruut x-telge pidi alas sees ning funktsioon võib tagastada väärtuse, et on sees küll (`return true`).

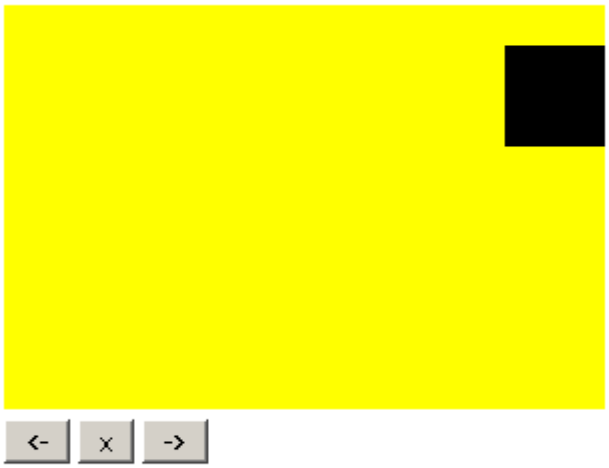
Liikumisfunktsioonis saab eelneva kontrolli tulemust kasutada.

```
if(kasSees(x+samm)){
  x=x+samm;
} else {
  seis();
}
```

Esimene lause siis vaatab, et kui ruudu eeldatav uus asukoht (`x+samm`) on tahvli sees, siis minnakse sinna uude kohta kohale (`x=x+samm`). Muul juhul jäädakse seima (`seis()`).

```
<!doctype html>
<html>
  <head>
    <title>Liikumine</title>
        <script>
           var x=100, laius=50;
           var kiirus=2, samm=-kiirus;
           var t, g; //tahvel, graafiline kontekst

           function algus(){
             t=document.getElementById("tahvel");
             g=t.getContext("2d");
             setInterval('liigu()', 100);
           }
           function joonista(){
             g.clearRect(0, 0, t.width, t.height);
             g.fillRect(x, 20, laius, 50);
           }
           function kasSees(uusX){
             if(uusX<0){return false;}
             if(uusX+laius>t.width){return false;}
             return true;
           }
           function liigu(){
             if(kasSees(x+samm)){
               x=x+samm;
             } else {
               seis();
             }
             joonista();
           }
           function vasakule(){
             samm=-kiirus;
           }
           function seis(){
             samm=0;
           }
```

```
            function paremale(){
              samm=kiirus;
            }
        </script>
  </head>
  <body onload="algus()">
      <canvas id="tahvel" width="300" height="200"
                  style="background-color:yellow"></canvas><br />
      <input type="button" value="&lt;-" onclick="vasakule()" />
      <input type="button" value="x" onclick="seis()" />
      <input type="button" value="-&gt;" onclick="paremale()" />
  </body>
</html>
```



Ülesandeid

* Pane näide tööle. Muuda ruudu suurust ning veendu, et näide töötab endiselt.
* Pane ruut liikuma ülalt alla ning selles suunas mõlemast servast seisma
* Ruutu saab liigutada iga ilmakaare suhtes. Ruut seiskub, kui jõuab serva.

Ruudu kutsumine hiirega

Nuppudega saab lehel toimetada küll. Mõnikord aga animatsioonide ja mängude juures mõistlik, kui õnnestub otse lehel olles kujundeid juhatada. Siinjuures aitavad samamoodi hiirevajutused nagu õpikus eespool kirjeldatud. Väike hoiatus: kui lehte vaadata mobiiliekraanil, siis seal tavalised hiiresündmused ei toimi, sest mobiil püüab korraga mitut näppu jälgida ning selle tarbeks on loodud touch-event. Aga sellest edasipidi.

Tahvli küljes olevale `onmousedown`-sündmuse külge sidusime funktsiooni `hiirAlla`, mille parameetrina tulnud e-ks nimetatud väärtuse kaudu saab kätte hiire andmed. Arvutus `e.clientX-tahvlikoht.left` annab hiire x-koordinaadi väärtuse tahvli asukoha suhtes. See püüti praegu muutujasse nimega `hx` (hiire x).

Praegusel juhul otsustatakse hiire järgi, et kas ruut peaks liikuma vasakule või paremale. Kui hiire

x-koordinaadi väärtus on suurem, kui ruudu x-koordinaadi väärtus, siis asub hiir ruudust paremal ning järelikult peaks ruut liikuma paremale, kui soovime, et ta hiire poole tuleks. Ehk siis ruudu liikumise sammuks saab koodi algul märgitud kiirus (mis hetkel positiivse väärtusega). Kui hiir sattus ruudust vasakule, siis sammuks saab kiiruse vastandarv, ehk siis ruut hakkab liikuma vasakule.

```
            function hiirAlla(e){
              var tahvlikoht=t.getBoundingClientRect();
              var hx=e.clientX-tahvlikoht.left;
              if(hx>x){samm=kiirus;}
              else{samm=-kiirus;}
            }
```

Edasi kood tervikuna.

```
<!doctype html>
<html>
  <head>
    <title>Liikumine</title>
        <script>
           var x=100, laius=50;
           var kiirus=2, samm=-kiirus;
           var t, g; //tahvel, graafiline kontekst

           function algus(){
             t=document.getElementById("tahvel");
             g=t.getContext("2d");
             setInterval('liigu()', 100);
           }
           function joonista(){
                 g.clearRect(0, 0, t.width, t.height);
                 g.fillRect(x, 20, laius, 50);
           }
           function kasSees(uusX){
             if(uusX<0){return false;}
             if(uusX+laius>t.width){return false;}
             return true;
           }
           function liigu(){
             if(kasSees(x+samm)){
               x=x+samm;
             } else {
               seis();
             }
             joonista();
           }
           function hiirAlla(e){
             var tahvlikoht=t.getBoundingClientRect();
             var hx=e.clientX-tahvlikoht.left;
             if(hx>x){samm=kiirus;}
             else{samm=-kiirus;}
           }
           function seis(){
             samm=0;
           }

        </script>
  </head>
  <body onload="algus()">
      <canvas id="tahvel" width="300" height="200"
                  style="background-color:yellow"
                          onmousedown="hiirAlla(event)"></canvas><br />
```

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

Ülesandeid

* Pane näide käima.
* Pane ruut liikuma ülalt-alla suunas (vertikaalselt). Võimalda selles sihis ruutu enese poole kutsuda.
* Muuda koodi nõnda, et hiirevajutus lükkaks ruutu pigem hiirest eemale.
* Ruut saab liikuda kõigi nelja ilmakaare suhtes. Võimalda igal pool ruutu hiirest eemale liikuma lükata (puhuda).

Liikumiskiiruse määramine hiirega

Eelmises näites lihtsalt vaadati, kuidas ruutu hiire poole meelitada. Arvestades aga kaugust hiirest, saab nõnda määrata ruudu liikumise kiiruse. Olgu siis nii, et lähemale vajutus tõmbab/lükkab tugevamini, või kehtib sama kaugema vajutuse kohta. Üheks mooduseks oleks öelda, et uueks ruudu liikumise sammuks saab kaugus hiire ja ruudu vahel (`samm=hx-x`). Sellisel puhul aga oleks ruut esimese sammuga juba hiire juures, teise sammuga sama palju möödas. Ning vähegi suurema sammude tiheduse ehk kaadrisageduse ja pikema sammu korral juba ekraanilt väljas. Sammu võib veidi väiksemas võtta seda nulli ja ühe vahel oleva kiiruskoefitsiendiga korrutades. Näiteks, kui see kordaja on 0,1 (arvutikoodis vaja koma asemel punkt kirjutada), siis kui hiir vajutatakse ruudust 30 punkti kaugusele, saab tegelikuks sammu pikkuseks kolm punkti, mida on silmaga juba suhteliselt sujuv vaadata.

```
            samm=(hx-x)*kiiruskoef;
```

Muu jääb üllatuslikult samaks, aga kiiruse muutmise paindlikkus on märgatavalt suurem.

```
<!doctype html>
<html>
  <head>
    <title>Liikumine</title>
        <script>
           var x=100, laius=50;
           var kiirus=2, samm=-kiirus;
           var kiiruskoef=0.1;
           var t, g; //tahvel, graafiline kontekst

           function algus(){
             t=document.getElementById("tahvel");
                 g=t.getContext("2d");
                 setInterval('liigu()', 100);
           }
           function joonista(){
                 g.clearRect(0, 0, t.width, t.height);
                 g.fillRect(x, 20, laius, 50);
           }
           function kasSees(uusX){
             if(uusX<0){return false;}
```

```
            if(uusX+laius>t.width){return false;}
            return true;
          }
          function liigu(){
            if(kasSees(x+samm)){
              x=x+samm;
            } else {
              seis();
            }
            joonista();
          }
          function hiirAlla(e){
            var tahvlikoht=t.getBoundingClientRect();
            var hx=e.clientX-tahvlikoht.left;
            samm=(hx-x)*kiiruskoef;
          }
          function seis(){
            samm=0;
          }

      </script>
  </head>
  <body onload="algus()">
      <canvas id="tahvel" width="300" height="200"
                  style="background-color:yellow"
                          onmousedown="hiirAlla(event)"></canvas><br />

  </body>
</html>
```

Ülesandeid

* Võimalda liikumise kiirust määrata ülalt-alla liikuva ruudu puhul.
* Ruut saab liikuda iga ilmakaare suunas. Samuti saab hiire abil määrata liikumise kiirusi.

* Pane ristkülik ekraanil laiustpidi suurenema
* Pane ristkülik ekraanil suurenema hiirevajutuse peale
* Üks hiirevajutus paneb ristküliku suurenema, teine vähenema
* Vähenemisel jääb ristküliku parem külg paigale. Nõnda on võimalik kordamööda hiirevajutuste abil jäljendada ussikese liikumist edasi.
* Ussikene pole enam ristkülik, vaid on ovaal
* Ussikene koosneb mitmest üksteise sappa ühendatud ovaalist.

Kukkumine

Tegelikus elus toimuvaid liikumisi jäljendades tuleb mõnigikord ka kukkumist ehk vaba langemist järgi teha. Järgnevalt mõned näited, seletused ja soovitused selle kohta.

Ring ekraanil

Kukkumine kipub millegipärast vahel kergesti palliga seostuma. Pall aga teadaolevalt enamasti ringikujuline. Seetõttu tuletame meelde, kuidas sai ringi joonistada ekraanile. Ristküliku puhul piisas joonistamiseks ühest käsust. Ringi ja muude kujundite puhul aga tuleb joonistamist alustada

käsuga `beginPath()`. Seejärel luua vajalikud kujundid. Ning lõpuks öelda `stroke()` või `fill()` vastavalt sellele, kas soovitakse näha vaid piirjooni või siis kogu kujund seest ära täita. Ning ringi joonistamiseks sobib kaare tegemise käsklus (`arc`). Määratakse, keskpunkt, raadius, kaare algnurk ja kaarepikkus radiaanides ning joonistamise suund. Ringi puhul võib alustada algusest (ehk nullkraadist), täisringiks on 2 piid ehk `2*Math.PI` ning joonistamise suund pole tähtis, siin näites jätsime selle `true`-ks.

```
<!doctype html>
<html>
  <head>
    <title>Kukkumine</title>
        <script>
           var x=20, y=20, r=10, samm=0;
           function joonista(){
             var t=document.getElementById("tahvel");
                 var g=t.getContext("2d");
                 g.clearRect(0, 0, t.width, t.height);
                 g.strokeStyle="red";
                 g.beginPath()
                 g.arc(x, y, r, 0, 2*Math.PI, true);
                 g.stroke()
           }

        </script>
  </head>
  <body onload="joonista()">
      <canvas id="tahvel" width="300" height="200"
                  style="background-color:yellow"></canvas><br />

  </body>
</html>
```

Ühtlane kukkumine

Ülalt alla liikumine toimub sarnaste arvutuste tulemusena nagu ennist vaadatud vasakult paremale liikumine - vaid muuta tuleb y-koordinaati. Algusfunktsioonis küsitakse ligipääs tahvlile ja temale joonistamiseks vajalikule graafilisele kontekstile (muutujad `t` ja `g`). Käsklus

```
setInterval('liigu()', 100);
```

teatab, et funktsioon `liigu()` tuleb käivitada iga 100 millisekundi tagant, ehk siis 10 korda

sekundis. Seda ooteaega saab katsetada ja muuta, aga 100 on osutunud suhteliselt kuldseks keskteeks. Pikem ooteaeg näib juba silmale viivitusena. Lühema aja puhul aga ei pruugi seade suuta enne pilti korralikult valmis joonistada.

```
<!doctype html>
<html>
  <head>
    <title>Kukkumine</title>
        <script>
           var x=20, y=20, r=10, ysamm=1;
           var t, g; //tahvel, graafiline kontekst

           function algus(){
             t=document.getElementById("tahvel");
             g=t.getContext("2d");
             setInterval('liigu()', 100);
           }

           function joonista(){
             g.clearRect(0, 0, t.width, t.height);
             g.strokeStyle="red";
             g.beginPath()
             g.arc(x, y, r, 0, 2*Math.PI, true);
             g.stroke()
           }

           function liigu(){
             y=y+ysamm;
             joonista();
           }

        </script>
  </head>
  <body onload="algus()">
      <canvas id="tahvel" width="300" height="200"
                  style="background-color:yellow"></canvas><br />

  </body>
</html>
```

Ülesandeid

* Tee näide läbi. Muuda ringi suurust ja kukkumise kiirust

* Pane kukkuma mitme joonistuskäsuga loodud kujund - nt pall, millele paar täppi peale joonistatud.
* Kujundi suurust ja kukkumise kiirust saab all olevas tekstiväljas muuta.

Kiirenev kukkumine

Tavaelu jälgides saab kõrgemalt kukkudes kätte suurema kiiruse. Ehk siis kiirus kukkumise ajal kasvab. Mingist suurusest hakkab kiirust piirama õhutakistus, aga lihtsamate arvutuste juures ka füüsikaõpikus jäetakse õhutakistus välja. Väliolukorda üsna täpselt saaks jäljendada, kui arvestada, mitu ekraanipunkti vastab ühele meetrile looduses ning siis vastavalt esemed ja raskuskiirendus paika ajada. Lihtsamal juhul aga võib kukkumissammule igal ringil veidi otsa liita, et sammu pikkus suureneks. Ning sobivat tulemust saab juba silma järgi hinnata. Siin näites said kõigepealt üles muutujad sammu ja kiirenduse kohta

```
var ysamm=1, ykiirendus=0.4;
```

Liikumisfunktsioonis kõigepealt suurendatakse sammu kiirenduse jagu ning siis liigutakse igrekiga sammu jagu allapoole.

```
ysamm=ysamm+ykiirendus;
y=y+ysamm;
```

Kood tervikuna

```
<!doctype html>
<html>
  <head>
    <title>Kukkumine</title>
        <script>
           var x=20, y=20, r=10;
           var ysamm=1, ykiirendus=0.4;
           var t, g; //tahvel, graafiline kontekst

           function algus(){
             t=document.getElementById("tahvel");
                g=t.getContext("2d");
                setInterval('liigu()', 100);
           }

           function joonista(){
                 g.clearRect(0, 0, t.width, t.height);
                 g.strokeStyle="red";
                 g.beginPath()
                 g.arc(x, y, r, 0, 2*Math.PI, true);
                 g.stroke()
           }

           function liigu(){
             ysamm=ysamm+ykiirendus;
             y=y+ysamm;
             joonista();
           }

        </script>
  </head>
  <body onload="algus()">
      <canvas id="tahvel" width="300" height="200"
```

```
                style="background-color:yellow"></canvas><br />

  </body>
</html>
```

Ülesandeid

* Pane näide käima, katseta erinevaid algseid sammu pikkusi ja kiirenduse väärtusi
* Katseta liikumist negatiivse sammu korral
* Määra algul kiirendus ja samm nulliks. Käivita kukkumine nupuvajutuse peale.
* Kiirendust ning palli suurust saab tekstiväljas määrata.

Kukkumiskoha määramine hiirega

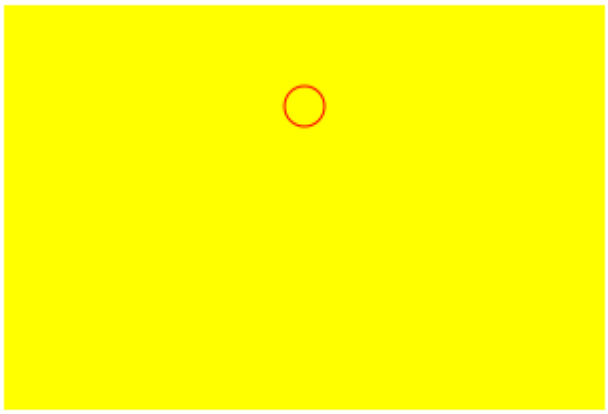
Liikumist jällegi mugavam ekraanil juhatada hiirega. Siin näites leitakse kõigepealt hiire asukoht tahvli suhtes (muutujad `hx` ja `hy`) ning pärast paigutatakse ring vastavale kohale. Vajalik on samuti ysamm'u ehk allakukkumiskiiruse määramine nulliks, sest muul juhul jätkaks pall hiirevajutuse kohast endise kiirusega kukkumist ning varsti poleks teda ekraanil võimalik enam kuigivõrd jälgida.

```
function hiirAlla(e){
  var tahvlikoht=t.getBoundingClientRect();
  var hx=e.clientX-tahvlikoht.left;
  var hy=e.clientY-tahvlikoht.top;
  x=hx;
  y=hy;
  ysamm=0;
}
```

Kui nüüd nõndamoodi toimetada, siis igal hiire vajutamise korral hüppab sinna pall ja asub sealt vaikselt alla kukkuma. Kood tervikuna.

```
<!doctype html>
<html>
  <head>
    <title>Kukkumine</title>
        <script>
           var x=20, y=20, r=10;
           var ysamm=1, ykiirendus=0.4;
           var t, g; //tahvel, graafiline kontekst

           function algus(){
                 t=document.getElementById("tahvel");
                 g=t.getContext("2d");
                 setInterval('liigu()', 100);
           }

           function joonista(){
                 g.clearRect(0, 0, t.width, t.height);
                 g.strokeStyle="red";
                 g.beginPath()
                 g.arc(x, y, r, 0, 2*Math.PI, true);
```

```
                g.stroke()
            }

            function liigu(){
              ysamm=ysamm+ykiirendus;
              y=y+ysamm;
              joonista();
            }

            function hiirAlla(e){
              var tahvlikoht=t.getBoundingClientRect();
              var hx=e.clientX-tahvlikoht.left;
              var hy=e.clientY-tahvlikoht.top;
              x=hx;
              y=hy;
              ysamm=0;
            }
        </script>
  </head>
  <body onload="algus()">
      <canvas id="tahvel" width="300" height="200"
                  style="background-color:yellow"
                          onmousedown="hiirAlla(event)"></canvas><br />

  </body>
</html>
```



Ülesandeid

* Pane näide tööle, muuda palli värvi ja suurust.
* Iga järgmise hiirevajutuse peale läheb pall veidi väiksemaks.
* Joonista tahvli alumine pool siniseks. Kui pall on jõudnud allapoole veepiiri, siis hakkab ta vaikselt väiksemaks minema.
* Veepiirist allpool jaguneb pall kaheks ning kumbki tükk hakkab kukkudes vaikselt vähenema.
* Vee all palli kiirus on aeglasem ja ei muutu.

Peatumine servas

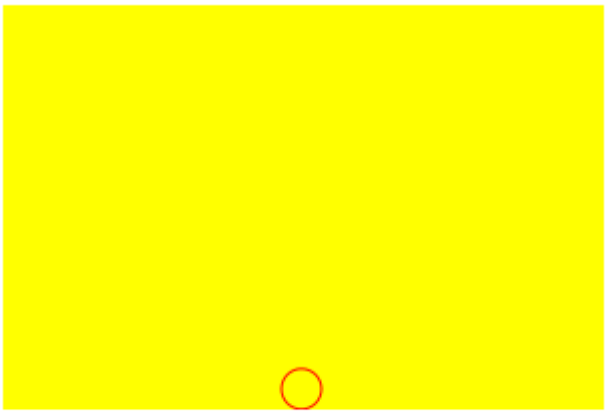
Nagu niisama liikumise juures, nii ka kukkumise puhul on vahel hea, kui kokkupandud kujundid meie tahtmata silmapiirilt ei kao. Taas tuleb kontrollida, et soovitav järgmine asukoht oleks lubatud ala sees, vaid siis liigutakse edasi. Ning tulevikule mõeldes on lisatud muutuja ka x-suunalise liikumise tarbeks. Lihtsalt kuna xsamm on parajasti 0, siis kukutakse otse alla. Kui juhtub, et uus leitud asukoht satuks tahvlilt välja, siis pannakse x ja y-sammud nulliks.

```
function liigu(){
     ysamm=ysamm+ykiirendus;
     if(kasSees(x+xsamm, y+ysamm)){
        x=x+xsamm;
        y=y+ysamm;
      } else {
        xsamm=0; ysamm=0;
      }
      joonista();
}
```

Ringi peatamise võimega kood tervikuna:

```
<!doctype html>
<html>
  <head>
    <title>Kukkumine</title>
        <script>
           var x=20, y=20, r=10;
           var ysamm=1, xsamm=0, ykiirendus=0.4;
           var t, g; //tahvel, graafiline kontekst

           function algus(){
             t=document.getElementById("tahvel");
                g=t.getContext("2d");
                setInterval('liigu()', 100);
           }

           function joonista(){
                 g.clearRect(0, 0, t.width, t.height);
                 g.strokeStyle="red";
                 g.beginPath()
                 g.arc(x, y, r, 0, 2*Math.PI, true);
                 g.stroke()
           }

           function liigu(){
                ysamm=ysamm+ykiirendus;
                if(kasSees(x+xsamm, y+ysamm)){
                   x=x+xsamm;
                   y=y+ysamm;
                 } else {
                   xsamm=0; ysamm=0;
                 }
                 joonista();
           }

           function kasSees(uusX, uusY){
             if(uusX-r<0){return false;}
                 if(uusX+r>t.width){return false;}
             if(uusY-r<0){return false;}
                 if(uusY+r>t.height){return false;}
                 return true;
           }
```

```
            function hiirAlla(e){
              var tahvlikoht=t.getBoundingClientRect();
              var hx=e.clientX-tahvlikoht.left;
              var hy=e.clientY-tahvlikoht.top;
              x=hx;
              y=hy;
              ysamm=0;
            }
        </script>
  </head>
  <body onload="algus()">
      <canvas id="tahvel" width="300" height="200"
                  style="background-color:yellow"
                          onmousedown="hiirAlla(event)"></canvas><br />

  </body>
</html>
```



Ülesandeid

* Katseta näidet, muuda palli värvi, joonista seest täis.
* Tõmba ekraanile maapinda tähistav joon, jäta pall selle juures seisma.
* Maapinnani jõudnud pall hakkab uuesti ülevalt kukkuma.
* Ülevalt uue kukkumise asukoht valitakse juhuslikult.

Palli loopimine

Nüüdseks on juba piisavalt oskusi kogunenud, et midagi täiesti elavat ja usutavat kokku panna. Näitena palli heitmise rakendus, kus pall hakkab liikuma hiire poole. Mida kaugemal hiir on, seda kiiremini. Servade juures kontrollitakse, et pall sealt välja ei läheks. Kui serv ette tuleb, siis võetakse hoog maha. Kui võimalust, hakkab ta sellest kohast uuesti allapoole kukkuma. Kui alla jõudnud, siis peab lihtsalt jääma ootama, kuni hiirega uuesti vajutatakse ja pall liikuma meelitatakse.

Tehniliselt midagi väga eripärast võrreldes eelnenuga polegi. Mõlema koordinaadi puhul pannakse pall sammuma hiire suunas. Sammu pikkus võrdeline kaugusega hiirest.

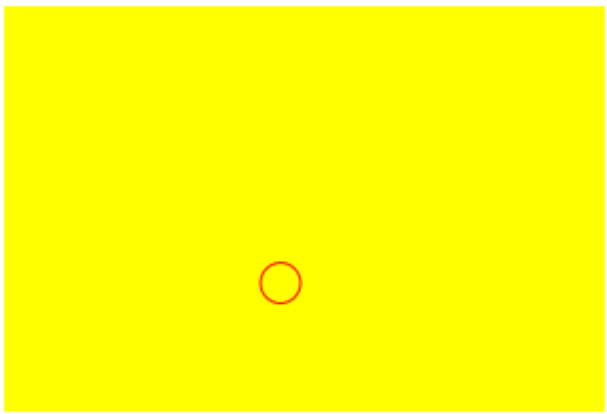
```
            xsamm=(hx-x)*kiiruskoef;
            ysamm=(hy-y)*kiiruskoef;
```

Tulemusena aga valmib täiesti mitmekülgne palli heitmise rakendus, mida saab edaspidi mitmesuguste liikuvate mudelite juures kasutada.

```
<!doctype html>
<html>
  <head>
    <title>Kukkumine</title>
        <script>
           var x=20, y=20, r=10;
           var ysamm=1, xsamm=0, ykiirendus=0.4, kiiruskoef=0.1;
           var t, g; //tahvel, graafiline kontekst

           function algus(){
             t=document.getElementById("tahvel");
             g=t.getContext("2d");
             setInterval('liigu()', 100);
           }

           function joonista(){
                 g.clearRect(0, 0, t.width, t.height);
                 g.strokeStyle="red";
                 g.beginPath()
                 g.arc(x, y, r, 0, 2*Math.PI, true);
                 g.stroke()
           }

           function liigu(){
              ysamm=ysamm+ykiirendus;
              if(kasSees(x+xsamm, y+ysamm)){
                   x=x+xsamm;
                   y=y+ysamm;
              } else {
                   xsamm=0; ysamm=0;
              }
               joonista();
           }
           function kasSees(uusX, uusY){
             if(uusX-r<0){return false;}
             if(uusX+r>t.width){return false;}
             if(uusY-r<0){return false;}
             if(uusY+r>t.height){return false;}
             return true;
           }

           function hiirAlla(e){
             var tahvlikoht=t.getBoundingClientRect();
             var hx=e.clientX-tahvlikoht.left;
             var hy=e.clientY-tahvlikoht.top;
             xsamm=(hx-x)*kiiruskoef;
             ysamm=(hy-y)*kiiruskoef;
           }
        </script>
  </head>
  <body onload="algus()">
      <canvas id="tahvel" width="300" height="200"
                  style="background-color:yellow"
                          onmousedown="hiirAlla(event)"></canvas><br />

  </body>
```

```
</html>
```

Nagu piltidelt näha, saab nõnda panna palli oma asukohta muutma.



Ülesandeid

* Pane näide tööle. Muuda muuda kiiruskoefitsienti, jälgi, kuidas see mõjutab palli liikumist.
* Muuda y-suunalist kiirendust ja jälgi, kuidas see mõjutab palli kukkumist.

Täpsusviskamine

* Võta aluseks palli loopimise näide. Paiguta pall tahvli ühele poolele ning kast tahvli teisele poolele maha
* Kui pall õnnestub kasti sisse visata, siis muudab kast värvi
* Eraldi arvuga näidatakse, mitu korda on kasti tabatud. Pärast tabamist hüppab pall algkohta tagasi
* Kasutajad viskavad palli kordamööda. Loetakse, mitu korda on kummalgi pall kasti pihta läinud.

Hiirt jälitav pall

* Tutvu palli loopimise näitega. Eemalda raskuskiirendus
* Sammu arvutamine käivitatakse mitte hiire vajutamise, vaid hiire liigutamise peale (sündmus onmousemove). Pall võiks hakata hiire poole liikuma.
* Lisa süsteemi teine pall teiste algkoordinaatidega. Mõlemad pallid püüavad sarnase valemiga liikuda hiire poole.
* Teine pall ei püüa liikuda mitte hiire, vaid esimese palli poole. Nii võiks hiire liigutamise tulemusena tekkida hiire järgi pallidest ahel.

Andmed ja otsing

Massiiv

Tavalises muutujas saame hoida ühte väärtust. Kui aga lehel on sarnaseid andmeid palju, siis igaüht omaette muutujas hoida on tülikas. Neid tuleb palju ning sobiva leidmine sealt ikkagi keerukas. Sarnaseid andmeid aitab koos hoida massiiv. Eesnimede loetelu saab näiteks teha nõnda:

```
            var eesnimed=new Array("Juku", "Kati", "Katrin", "Madis");
```

Selle peale luuakse loetelu, kus Juku poole saab pöörduda kujul `eesnimed[0]` ning Madise poole `eesnimed[3]`. Ehk siis lugemist alustatakse nullist ning viimase järjekorranumbriks on elementide arvust ühe jagu väiksem arv. Katsetamiseks juurde ka töötav koodinäide.

```
<!doctype html>
<html>
  <head>
     <title>Otsing</title>
         <meta charset="utf-8" />
         <script>
            var eesnimed=new Array("Juku", "Kati", "Katrin", "Madis");
                function algus(){
                   document.getElementById("vastus").innerHTML=
                     "Loetelu alguses on "+eesnimed[0]+" ja "+eesnimed[1];
                }
         </script>
  </head>
  <body onload="algus()">
      <div id="vastus">

       </div>
  </body>
</html>
```

Näite tulemuseks:

Loetelu alguses on Juku ja Kati

Elementide arv

Javaskript annab õnneks mugavasti kätte massiivi elementide arvu - selleks omadus .length massiivimuutuja nime küljes. Sealtkaudu võimalik siis kohe arvutuse teel kätte saada massiivi viimane element. Eesnimede massiivis siis

```
eesnimed[eesnimed.length-1]
```

Näide tervikuna

```
<!doctype html>
<html>
  <head>
     <title>Otsing</title>
         <meta charset="utf-8" />
         <script>
            var eesnimed=new Array("Juku", "Kati", "Katrin", "Madis");
                function algus(){
                   document.getElementById("vastus").innerHTML=
                     "Eesnimesid kokku on "+eesnimed.length+". Viimane neist "+
                            eesnimed[eesnimed.length-1]+".";
                }
         </script>
  </head>
```

```
  <body onload="algus()">
      <div id="vastus">

          </div>
  </body>
</html>
```

Ja tulemus

Eesnimesid kokku on 4. Viimane neist Madis.

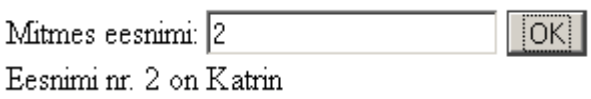
Ülesandeid

* Koosta massiiv inimeste pikkustega. Trüki välja esimene ja viimane pikkus.
* Trüki välja teine ja eelviimane pikkus.

Järjekorranumbri järgi küsimine massiivist

Nagu kõiksugu muid väärtusi, nii ka massiivi elementide järjekorranumbreid saab küsida kasutaja käest. Siin siis võetakse eesnime number sisestuskastist ning kuvatakse vastav eesnimi.

```
<!doctype html>
<html>
  <head>
     <title>Otsing</title>
         <meta charset="utf-8" />
         <script>
            var eesnimed=new Array("Juku", "Kati", "Katrin", "Madis");
                function otsi(){
                   var nimenr=parseInt(document.getElementById("kast1").value);
                   document.getElementById("vastus").innerHTML=
                     "Eesnimi nr. "+nimenr+" on "+eesnimed[nimenr];
                }
         </script>
  </head>
  <body>
      Mitmes eesnimi:
          <input type="text" id="kast1" />
          <input type="button" value="OK" onclick="otsi()" />
      <div id="vastus">

      </div>
  </body>
</html>
```



Juhusliku järjekorranumbriga element

Ka juhuarv toimib massiivi elemendi järjekorranumbri leidmise juures. Lihtsalt on kasulik hoolitseda, et järjekorranumbriks tuleks täisarv - parseInt pakub sellise võimaluse.

```
<!doctype html>
<html>
  <head>
     <title>Otsing</title>
         <meta charset="utf-8" />
         <script>
            var eesnimed=new Array("Juku", "Kati", "Katrin", "Madis");
                function otsi(){
                   var juhuarv=Math.random(); //0 .. 0.99
                   var nimenr=parseInt(eesnimed.length*juhuarv);
                   document.getElementById("vastus").innerHTML=
                     "Eesnimi nr. "+nimenr+" on "+eesnimed[nimenr]+
                         ".<br /> Juhuarv oli "+juhuarv+".";
                }
         </script>
  </head>
  <body>
          <input type="button" value="Otsi juhuslik" onclick="otsi()" />
      <div id="vastus">

          </div>
  </body>
</html>
```

Ülesandeid

* Koosta massiiv ütlustega. Lehe avanemisel ilmub ekraanile neist üks juhuslik ütlus.

* Erinevalt eelmisest teatatakse kasutajale, mitme ütluse vahel tal on valida. Kasutaja sisestab soovitud ütluse järjekorranumbri ning näeb vastaval kohal olevat ütlust.

* Koosta üks massiiv poiste eesnimedega ja teine tüdrukute eesnimedega. Lehe avanemisel valitakse juhuslik poiss ja juhuslik tüdruk ning teatatakse, et nemad lähevad esimestena tantsima.

Kordused

Massiivi andmete otsimiseks, muutmiseks ja töötlemiseks on kasulikuks abiliseks kordused. Kõigepealt väike näide korduse töötamise kohta

```
for(var i=0; i<3; i++){
  document.getElementById("vastus").innerHTML+=i+" ";
}
```

Tuumikuks siis for-tsükkel, kus iga sammuga liidetakse vastuse sisule (innerHTML) juurde tsükliloenduri i väärtus ning sinna lisatakse tühik, et arvud üksteisega kokku ei kasvaks. Tehtemärk += tähendab olemasolevale väärtusele millegi juurde lisamist, ilma, et peaks seda kohta korduvalt välja trükkima.

```
<!doctype html>
<html>
  <head>
     <title>Otsing</title>
         <meta charset="utf-8" />
         <script>
                function algus(){
                 for(var i=0; i<3; i++){
                   document.getElementById("vastus").innerHTML+=i+" ";
                 }
                }
         </script>
  </head>
  <body onload="algus()">
      <div id="vastus">

          </div>
  </body>
</html>
```

Väljund

0 1 2

Ülesandeid

```
* Tutvu tsükli näitega. Trüki arvude ruudud ühest kahekümneni
  (suurimaks siis 20*20 ehk 400).
* Luba kasutajal sisestada, mitmeni arvude ruute soovitakse
```

Tsükkel ja massiiv

Sarnaselt tsükli abil saab välja trükkida või muul moel ette võtta ka kõik samas massiivis olevad eesnimed. Lihtsalt väljatrüki või muul moel kasutuse koha peal pole enam tsükliringi loendur, vaid küsitakse massiivist vastava järjekorranumbriga element. Praegusel juhul siis `eesnimed[i]`.

```
<!doctype html>
<html>
  <head>
     <title>Otsing</title>
         <meta charset="utf-8" />
         <script>
                var eesnimed=new Array("Juku", "Kati", "Katrin", "Madis");
                function algus(){
                 for(var i=0; i<eesnimed.length; i++){
                   document.getElementById("vastus").innerHTML+=eesnimed[i]+" ";
                 }
                }
         </script>
  </head>
  <body onload="algus()">
      <div id="vastus">

          </div>
  </body>
</html>
```

Väljundiks:

Juku Kati Katrin Madis

Eelmises näites olid eesnimede vahel tühikud. Nõnda tekstina saab aga ka kõiki HTMLi korraldusi anda ja nendega funktsiooni töö tulemust mõjutada. Kui eesnimede vahele kirjutada reavahetusmärgend `<br />` , siis kasutaja näeb iga nime omaette real.

```
<!doctype html>
<html>
  <head>
     <title>Otsing</title>
         <meta charset="utf-8" />
         <script>
                var eesnimed=new Array("Juku", "Kati", "Katrin", "Madis");
                function algus(){
                 for(var i=0; i<eesnimed.length; i++){
                   document.getElementById("vastus").innerHTML+=
                       eesnimed[i]+"<br />";
                 }
                }
         </script>
  </head>
  <body onload="algus()">
      <div id="vastus">

          </div>
  </body>
</html>
```

Kasutaja näeb tulemust nõnda:

Juku
Kati
Katrin

Madis

Tegelikult on kokkupandud HTML-kood selline:

```
Juku<br />Kati<br />Katrin<br />Madis<br />
```

Veebilehitseja aga teab, et `<br />` käskluse peale tuleb teksti edasi näitama hakata järgmiselt realt ning nõnda näemegi ilusti üksteise all olevaid eesnimesid.

Veidi põhjalikumaks loetelu looomiseks on HTMLis nummerdamata loetelu (unordered list), mille loomise käsuks märgend `<ul>` ning iga elemendi alguse teadustamiseks `<li>`. Lõpud siis vastavalt `</li>` ja `</ul>`. Nimede loetelu nummerdamata loetelus HTMLis näeb välja

```
<ul>
  <li>Juku</li>
  <li>Kati</li>
  <li>Katrin</li>
  <li>Madis</li>
</ul>
```

Programmikoodiga sellist loetelu tekitades tuleb üksikud lõigud kokku koguda. Siinses näiteks on selleks muutuja `t` (nagu tulemus).

Algul antakse `t`-le väärtus `<ul>`. Jutumärgid seetõttu ümber, et tegemist tekstiga.

```
var t="<ul>";
```

Iga tsükliringi juures lisatakse vastava järjekorranumbriga eesnimi, mille ees `<li>` ja taga `</li>`

```
  t+="<li>"+eesnimed[i]+"</li>";
```

Ning lõpuks loetelu lõpp `</ul>`.

```
t+="</ul>";
```

Sellisena saabki tulemuse lehel kuvamiskõlblikuks.

```
<!doctype html>
<html>
  <head>
     <title>Otsing</title>
         <meta charset="utf-8" />
         <script>
                var eesnimed=new Array("Juku", "Kati", "Katrin", "Madis");
                function algus(){
                 var t="<ul>";
                 for(var i=0; i<eesnimed.length; i++){
                   t+="<li>"+eesnimed[i]+"</li>";
                 }
                 t+="</ul>";
                 document.getElementById("vastus").innerHTML=t;
                }
         </script>
  </head>
  <body onload="algus()">
      <div id="vastus">
```

```
      </div>
  </body>
</html>
```

- Juku
- Kati
- Katrin
- Madis

Ülesandeid

* Tutvu massiivi andmete trükkimise näidetega, tee näited läbi.
* Kirjuta iga eesnime välja kaks korda
* Trüki eesnimed ekraanile tagurpidises järjekorras
* Asenda `<ul>` `<ol>`'ga ning vaata, kuidas mummud nimede ees asenduvad järjekorranumbritega

Teksti otsimise käsk

Javaskripti levinud tööks veebilehel on andmete seast sobivate välja otsimine - kui nimekiri on pikk, siis aitab programmikood sisestatud lõigu järgi sobivad välja valida. Selle jaoks leidub tekstifunktsioon nimega `indexOf` - mis teatab, kas ja kust otsitav tekst leiti. Tekstis on programmeerimiskeelte tarvis tähed nummerdatud, kusjuures Javaskripti puhul hakkab nummerdamine nullist. Ehk siis eesnimes `"Mart"` on täht `"M"` järjekorranumbriga 0 ja täht `"a"` järjekorranumbriga 1. Ning kui küsin et `"Mart".indexOf("a")`, siis saan vastuseks 1. Kui aga küsin mõnd olematut tähte, näiteks `"Mart".indexOf("u")`, siis saan vastuseks -1, sest tähte `"u"` Mardi nimes pole. Järgnevalt väike demo, kuidas seda käsklust veebilehel kasutada.

Ühte tekstikasti sisestab kasutaja uuritava lause, teise kasti otsitava teksti. Käsklus

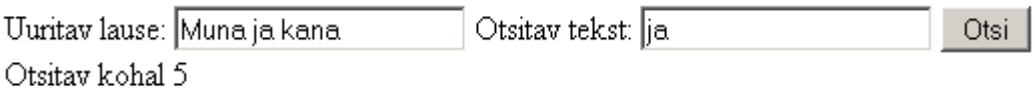
```
var koht=lause.indexOf(otsitav);
```

püüab muutujasse nimega koht `indexOf`-käskluse tulemuse, kus otsiti lausest otsitavat. Kui lõik leiti, nädatakse tähe järjekorranumbrit mitmendast lõik hakkab. Kui mitte, siis vastavaks koodiks olevat arvu -1.

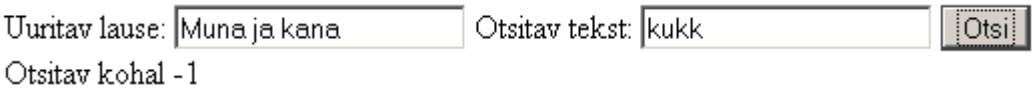
```
<!doctype html>
<html>
  <head>
     <title>Otsing</title>
         <meta charset="utf-8" />
         <script>
                function otsi(){
                 var lause=document.getElementById("kast1").value;
                 var otsitav=document.getElementById("kast2").value;
                 var koht=lause.indexOf(otsitav);
                 document.getElementById("vastus").innerHTML=
```

```
                  "Otsitav kohal "+koht;
               }
        </script>
  </head>
  <body>
    Uuritav lause:
      <input type="text" id="kast1" value="Muna ja kana" />
        Otsitav tekst:
      <input type="text" id="kast2" value="ja" />
          <input type="button" value="Otsi" onclick="otsi()" />
      <div id="vastus">

      </div>
  </body>
</html>
```

Kui lauseks "Muna ja kana" ning tekstiks "ja", siis vastuseks 5, sest tähe "j" järjekorranumber on 5 ("M"-tähe number on 0).



Samas lauses aga kukk puudub ning vastuseks on miinus üks.

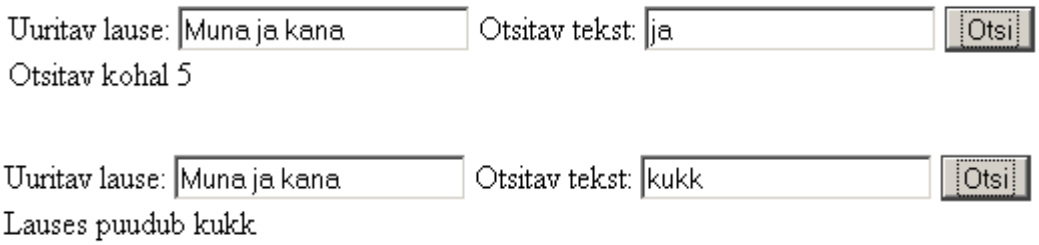


Saadud arve on võimalik kasutajale koos selgitustega näidata. Tingimuslause abil tasub kindlaks teha, et kui leitud koha väärtus on -1, siis võib kasutajale selgesõnaliselt teada anda, et lauses puudub otsitav lõik. Muul juhul aga öeldakse, kust otsitav leiti.

```
<!doctype html>
<html>
  <head>
     <title>Otsing</title>
         <meta charset="utf-8" />
         <script>
                function otsi(){
                 var lause=document.getElementById("kast1").value;
                 var otsitav=document.getElementById("kast2").value;
                 var koht=lause.indexOf(otsitav);
                 if(koht==-1){
                   document.getElementById("vastus").innerHTML=
                     "Lauses puudub "+otsitav
                 } else {
                   document.getElementById("vastus").innerHTML=
                     "Otsitav kohal "+koht;
                 }
                }
         </script>
  </head>
  <body>
    Uuritav lause:
      <input type="text" id="kast1" value="Muna ja kana" />
        Otsitav tekst:
```

```
      <input type="text" id="kast2" value="ja" />
          <input type="button" value="Otsi" onclick="otsi()" />
      <div id="vastus">

      </div>
  </body>
</html>
```

Samade lausete puhul siis saab nüüd koos selgitustega vasted.



Otsing massiivist

Edasi piisab lõigu otsimise näide kokku panna varasema massiivi elementide näitamise näitega. Iga eesnime puhul kontrollitakse, et kas otsitav lõik selles eesnimes leidub. Kui jah, siis lisatakse vastav eesnimi näidatavate loetellu. Kui ei leidu, siis ei lisata ning seda ekraanil ei näe. Vastav lõik eraldi välja tooduna:

```
if(eesnimed[i].indexOf(otsitav)!=-1){
  t+="<li>"+eesnimed[i]+"</li>";
}
```

Kui muidu koostati loetelu nõnda, et pandi kõik massiivis olevad eesnimed loetellu

```
var t="<ul>";
for(var i=0; i<eesnimed.length; i++){
  t+="<li>"+eesnimed[i]+"</li>";
}
t+="</ul>";
```

siis nüüd lisandus `for`-tsükli sisse `if`-tingimuslause, mille abil vaid tingimusele vastavad eesnimed näidatakse:

```
var t="<ul>";
for(var i=0; i<eesnimed.length; i++){
  if(eesnimed[i].indexOf(otsitav)!=-1){
    t+="<li>"+eesnimed[i]+"</li>";
  }
}
t+="</ul>";
```

Töötav näide tervikuna:

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

```
    <title>Otsing</title>
        <meta charset="utf-8" />
        <script>
               var eesnimed=new Array("Juku", "Kati", "Katrin", "Madis");
               function otsi(){
                var otsitav=document.getElementById("kast2").value;
                var t="<ul>";
                for(var i=0; i<eesnimed.length; i++){
                  if(eesnimed[i].indexOf(otsitav)!=-1){
                    t+="<li>"+eesnimed[i]+"</li>";
                  }
                }
                t+="</ul>";
                document.getElementById("vastus").innerHTML=t;
               }
        </script>
  </head>
  <body>

       Otsitav tekst:
     <input type="text" id="kast2" />
         <input type="button" value="Otsi" onclick="otsi()" />
     <div id="vastus"></div>
  </body>
</html>
```

Suur- ja väiketähed

Javaskript eristab suur- ja väiketähti. Ehk siis K ning k on tema jaoks sama erinevad tähed kui a ja b. Kasutaja aga otsimise ajal ei pruugi jälgida ega eristada, kumma suurusega tähe järgi ta parajasti otsida soovib. Kasutajat häirivate üllatuste vältimiseks saab otsingulehe panna käituma nõnda, et sobivad eesnimed leitakse nii suurte kui väikseste tähtede järgi otsides. Üheks mooduseks on nii kasutaja sisestatud tekst kui ka võrreldav nimi muuta võrdlemise ajaks väiketähtedeks. Sellisel juhul pole vahet, milline oli tähekõrgus enne - võrreldakse ikkagi lõpuks ühesuguseid sümboleid. Kasutaja sisestatav tekst tehakse väikseks kohe andmete sisselugemisel.

```
                var otsitav=
                   document.getElementById("kast2").value.toLowerCase();
```

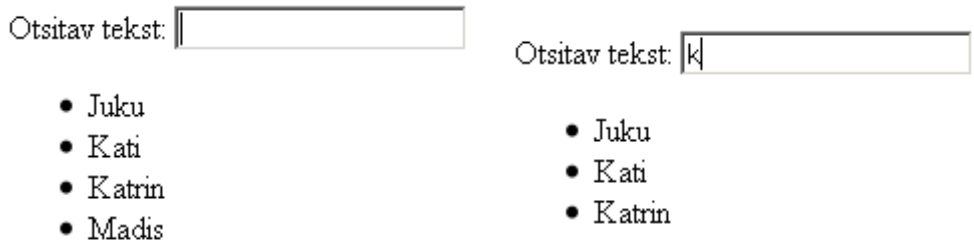
Massiivist eesnimesid võttes aga tehakse nad väikseks ükshaaval iga võrdlemise tarbeks.

```
                  if(eesnimed[i].toLowerCase().indexOf(otsitav)!=-1){
                    t+="<li>"+eesnimed[i]+"</li>";
                  }
```

Õigemini massiivis olev nimi ise jääb endiselt nõnda nagu ta on, aga võrdlemiseks tehakse koopia mille toLowerCase() väljastab.

Otsingurakendus tekstina:

```
<!doctype html>
<html>
  <head>
     <title>Otsing</title>
         <meta charset="utf-8" />
         <script>
                var eesnimed=new Array("Juku", "Kati", "Katrin", "Madis");
                function otsi(){
                 var otsitav=
                    document.getElementById("kast2").value.toLowerCase();
                 var t="<ul>";
                 for(var i=0; i<eesnimed.length; i++){
                   if(eesnimed[i].toLowerCase().indexOf(otsitav)!=-1){
                     t+="<li>"+eesnimed[i]+"</li>";
                   }
                 }
                 t+="</ul>";
                 document.getElementById("vastus").innerHTML=t;
                }
         </script>
  </head>
  <body onload="otsi()">
        Otsitav tekst:
      <input type="text" id="kast2" onkeyup="otsi()" />
      <div id="vastus"></div>
  </body>
</html>
```



Ülesandeid

* Tutvu kasutaja otsitavate nimede näitamisega
* Näita vaid neid nimesid, mis algavad otsitava tekstiga
  (indexOf(otsitav)==0)
* Näita kõik leitud nimed välja väiketähtedena

Otsing kirjetest

Päriselus ette tulevate andmete juures kipub sageli olema rohkem kui üks tunnus. Raamatul on pealkiri ja autor ja ilmumisaeg. Pildil mõõtmed, tegemisaeg, koht ja vahel kirjeldus. Ning mõnikord neid tunnuseid ehk välju võib andmetel päris palju olla. Kui otsitakse korvpallureid, siis pigem kahesaja ringis olev vastav väärtus pikkuse juures võib sobiva vihje anda. Kui aga kehakaal samale arvule läheneb, siis pole vastaval tegelasel ehk suuremate võistluste juures kuigi palju peale hakata. Õnneks võimaldab ka Javaskript tunnuseid nõnda eraldi jagada ja pärast vaadata, et millist

neist just parajasti kätte saada tahetakse. Järgnevalt näide, kus laste andmeteks on nende eesnimed ja sünniajad. Otsitakse eesnime järgi, aga näidatakse mõlemat tunnust. Eelnevalt väike seletus andmestiku kirjapaneku kohta.

Luuakse uus massiiv nimega lapsed. Tühi massiiv näeks välja

```
var lapsed=[];
```

Siia aga on võimalik kohe väärtused ka sisse panna. Ühe lapse andmed kannatab kirja panna kujul

```
{"eesnimi":"Juku", "synniaeg": "13.06"}
```

Igal lapsel siis kaks tunnust: eesnimi ja synniaeg. Teispool koolonit nende väärtused konkreetse lapse puhul. Kui laste andmeid mitu järjest ja komadega eraldatult on, siis tulebki tulemuseks laste massiiv.

```
var lapsed=[
   {"eesnimi":"Juku", "synniaeg": "13.06"},
   {"eesnimi":"Kati", "synniaeg": "11.06"},
   {"eesnimi":"Mati", "synniaeg": "25.08"},
   {"eesnimi":"Madis","synniaeg": "09.02"}
];
```

Sellist andmete kirjapanekut nimetatakse JSON-iks (JavaScript Object Notation). Sellisel kujul programmile etteantud andmeid saab mugavasti koodis kasutada. Muutuja `lapsed` on täiesti tavaline massiiv, `lapsed[0]` näiteks massiivi esimene laps. Kui aga soovitakse tema eesnime, siis tuleb küsida `lapsed[0].eesnimi`. Soovides tsükli ehk korduslause abil järgemööda kõikide laste poole pöörduda, tuleb kantsulgudes oleva arvu asemele panna tsüklimuutuja. Nõnda siis kontrollitakse i-nda lapse puhul, et kas tema väiketähtedena vaadatud eesnimes on otsitav tekst.

```
if(lapsed[i].eesnimi.toLowerCase().indexOf(otsitav)!=-1){}
```

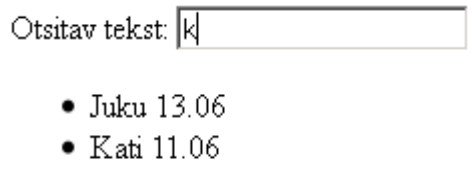
Loetellu näitamiseks aga lisatakse eesnimi koos sünniajaga.

```
t+="<li>"+lapsed[i].eesnimi+ " "+
      lapsed[i].synniaeg+"</li>";
```

Edasi juba otsingu kood tervikuna.

```
<!doctype html>
<html>
  <head>
     <title>Otsing</title>
         <meta charset="utf-8" />
         <script>
                var lapsed=[
                   {"eesnimi":"Juku", "synniaeg": "13.06"},
                   {"eesnimi":"Kati", "synniaeg": "11.06"},
                   {"eesnimi":"Mati", "synniaeg": "25.08"},
                   {"eesnimi":"Madis","synniaeg": "09.02"}
                ];
                function otsi(){
                 var otsitav=
                    document.getElementById("kast2").value.toLowerCase();
                 var t="<ul>";
                 for(var i=0; i<lapsed.length; i++){
```

```
                    if(lapsed[i].eesnimi.toLowerCase().indexOf(otsitav)!=-1){
                      t+="<li>"+lapsed[i].eesnimi+ " "+
                            lapsed[i].synniaeg+"</li>";
                    }
                  }
                  t+="</ul>";
                  document.getElementById("vastus").innerHTML=t;
                }
          </script>
  </head>
  <body onload="otsi()">
        Otsitav tekst:
      <input type="text" id="kast2" onkeyup="otsi()" />
      <div id="vastus"></div>
  </body>
</html>
```



Ülesandeid

* Tutvu eesnimede ja sünnipäevade vaatamise näitega.
* Kuva sünnipäev eespool, eesnimi tagapool

* Koostage väike sõnaraamat, kus näidatakse sisestatud sõna vasteid.

Sõnaraamat.

Sõnu koos vastetega vähemasti ühe õpiku õppetüki jagu.
Kasutaja sisestatud tähtede peale näidatakse välja sobivad
sõnad ja vasted, leht on mõnevõrra kujundatud
(näiteks vasted üle ühe erinevat värvi)
Viisakas on näiteks kasutada mõnd olemasolevat mobiililehe malli (template)

Pikkuste joonis.

Massiivis on inimeste nimed ja pikkused.
Joonisele kuvatakse vaid nende inimeste nimed, kes on etteantud pikkuste
vahemikus. Nime kõrval näidatakse pikkus vastava kõrgusega tulbana.

Asukohad parklas.

Massiivis on kirjas autode numbrid ning nende
x- ja y-koordinaadid parklas. Sisestades autonumbri või selle osa
näeb otsingule vastavaid autosid parklas (koos õige asukohaga).

Pallide mäng

Nüüdseks on üksikuid programmeerimisoskusi juba päris palju tekkinud. Järgnevalt paneme kokku näite kus neid üheskoos pruukida saab. Ning selle näite pealt saab loodetavasti juba mitmesuguseid keerukamaid lahendusi kombineerida, kus korraga mitu kujundit ekraanil liiguvad ning kasutaja neid mõjutada saab. Olgu siis tegemist ringi sõitvate autodega, pallimänguga või mõne aparaadi jäljendamisega.

Liikuvad pallid

Varem tehtud näites liikus meil ekraani peal üks pall. Tema koordinaadid olid kirjas lehel muutujatena. Liikumise tarvis muudeti iga natukese aja tagant nende väärtusi ning siis muutujate väärtuste järgi joonistati pall lehele sobivasse kohta. Kui on tarvis panna liikuma paar-kolm palli, siis saab vajadusel seda näidet lihtsalt laiendada - igale pallile oma x-i ja y-i komplekt, mille väärtusi muudetakse ning mille järgi joonistatakse kujundid pärast ekraanil sobivasse asukohta. Iga palli lisandumisega läheb nõnda küll kood mõnevõrra pikemaks. Aga kolm-neli lisanduvat koodirida palli kohta pole veel hullu, kui pallide arv on ette teada ja piiratud.

Kui aga palle võib lehel olla kümneid - või polegi nende lõplik arv teada, sellisel juhul igale pallile omi eraldi muutujaid teha pole mõistlik. Välja aitab eelmisest peatükist tuttavaks saanud massiiv.

```
var pallid=[
  {"x":10 , "y":50, "r":10, "dx":1, "dy":0},
  {"x":100, "y":30, "r":10, "dx":0, "dy":1}
];
```

Nõnda saab iga palli kohta kirja panna tema asukoha ja liikumise tarbeks vajalikud andmed. Nagu siin näha, siis iga palli jaoks on kirja tema x ja y-koordinaat ja raadius. Samuti sammu pikkus liikumisel: `"dx":1` tähendab, et iga kaadri ehk liikumissammu puhul suurendatakse x-i väärtust ühe võrra, `"dy":0` ütleb, et palli y-koordinaati ei muudeta, see tähendab, et ta üles- ega allapoole ei liigu.

Funktsioonid iseenesest on eelnevatest liikumisnäidetest tuttavad. Ikka tuleb iga sammu jaoks uued koordinaadid arvutada (funktsioon `liigu`) ning nende põhjal pilt välja kuvada (funktsioon `joonista`). Funktsiooni sisu lihtsalt mõnevõrra teistsugune.

```
function joonista(){
      g.clearRect(0, 0, t.width, t.height);
      for(var i=0; i<pallid.length; i++){
        g.beginPath();
        g.arc(pallid[i].x, pallid[i].y, pallid[i].r,
              0, 2*Math.PI, true);
        g.fill();
      }
}
```

Joonistamise puhul tuleb tahvel ikka puhtaks teha alguskoordinaatidest (0, 0) kuni tahvli laiuse ja kõrguseni. Lihtsalt joonistuskäsklus vaja iga palli jaoks eraldi käivitada. Tsüklimuutuja i näitab, et mitmenda palliga tegu on. Joonistuskäsu juures `pallid[i].x` ja `pallid[i].y` annavad teada vastava palli asukoha. Kui iga palli juures eraldi käsud `g.beginPath()` ja `g.fill()`, siis pole karta, et ühe kujundi juures tõmmatud jooned võiksid järgmise omi segama hakata.

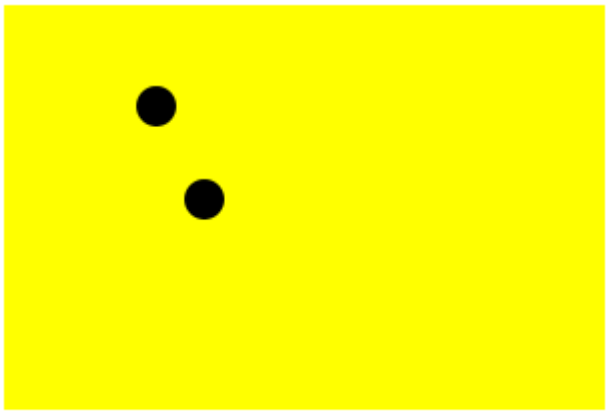
Liikumise funktsioon peab samuti arvestama, et pallide arv sõltub massiivi pikkusest.

```
            function liigu(){
              for(var i=0; i<pallid.length; i++){
                    pallid[i].x+=pallid[i].dx;
                    pallid[i].y+=pallid[i].dy;
              }
              joonista();
            }
```

Õnneks on iga palli juures kirjas, et kui palju ja kuhu poole ta liikuma peab. Käsk += lisab iga palli koordinaadile juurde liikumiseks vajaliku sammu pikkuse. Ning pärast uute asukohtade arvutamist tuleb välja kutsuda joonista-funktsioon, et ka ekraanil tulemust näha oleks. Edasi kood tervikuna.

```
<!doctype html>
<html>
  <head>
    <title>Pallid</title>
        <script>
           var pallid=[
             {"x":10 , "y":50, "r":10, "dx":1, "dy":0},
             {"x":100, "y":30, "r":10, "dx":0, "dy":1}
           ];


           var t, g; //tahvel, graafiline kontekst

           function algus(){
             t=document.getElementById("tahvel");
             g=t.getContext("2d");
             setInterval('liigu()', 100);
           }

           function joonista(){
                 g.clearRect(0, 0, t.width, t.height);
                 for(var i=0; i<pallid.length; i++){
                   g.beginPath();
                   g.arc(pallid[i].x, pallid[i].y, pallid[i].r,
                          0, 2*Math.PI, true);
                   g.fill();
                 }
           }

           function liigu(){
             for(var i=0; i<pallid.length; i++){
                   pallid[i].x+=pallid[i].dx;
                   pallid[i].y+=pallid[i].dy;
             }
             joonista();
           }

        </script>
  </head>
  <body onload="algus()">
      <canvas id="tahvel" width="300" height="200"
                  style="background-color:yellow" ></canvas><br />

  </body>
</html>
```

Ning ekraanil võib imetleda, kuidas pallid kumbki oma suunas vaikselt liiguvad.



Ülesandeid

- Pane näide tööle
- Lisa mõned pallid
- Lisa pallile omaduseks tema värv. Tulemusena liiguvad mitut värvi pallid mööda ekraani

- Koosta kujunditena kriipsujukud. Pane nad mööda ekraani liikuma.

Põrkavad pallid

Seinast või takistuselt tagasipöördumine kuulub ikka mitmesuguste lahenduste juurde. Kui kujundeid on hulgem, tasub siingi massiivi ja tsükli võimalusi arvestada, sest suurema kujundite hulga korral ei jõua kuidagi kõike eraldi koodiridadega kontrollida. Sarnaselt varasemaga tuleb kontrollida, kas kujund ikka on lubatud ala sees et hiljem tema edasise liikumissuuna üle otsustada. Siin antakse funktsioonile kaasa terve palli objekt korraga - erinevalt enne tehtust, kus tulid eraldi parameetritena x- ja y-koordinaat. Eriti keerukamate kujundite abil on nii hea kõik andmed kätte saada ilma muretsemata, kui palju neid ühe kujundi kohta on. Kujundi sees on juba sel enda teada, mis omadused temas peituvad. Ümmarguse palli puhul saab piiridesse sobivust mugavalt arvutada. Palli vasaku serva x-koordinaadi leiab, kui keskpunktist lahutada palli raadius, palli parema serva koordinaadi leidmiseks tuleb see liita. Üles-alla suunas tuleb ülemise y-koordinaadi leidmiseks see keskpunkti omast lahutada, sest y-telg liigub arvuti joonistuskäskude puhul ülalt alla. Alumise punkti leidmiseks siis jällegi liita.

```
function kasSees(pall){
  if(pall.x-pall.r<0){return false;}
  if(pall.x+pall.r>t.width){return false;}
  if(pall.y-pall.r<0){return false;}
  if(pall.y+pall.r>t.height){return false;}
  return true;
}
```

Põrkamise puhul peab vaatama, et pall taas soovitud suunas liikuma hakkaks. Mõndapidi oleks lihtne lihtsalt liikumissammu väärtusel märk ära vahetada nii et plussist saaks miinus ja vastupidi. Sellega kipub aga vahel tekkima eriolukord, kus pall jääb ühe serva juurde "värisema", sest ei saa aru, kas on juba põrganud või mitte ja proovib uuesti suunda vahetada. Kindlam on märk nõnda sättida, et pall taas platsi keskosa poole liikuma hakkaks. Ehk kui mindi üle vasaku serva `(pall.x-pall.r<0)`, siis tuleb hoolitseda, et liikumissammuks oleks kindlasti positiivne arv `pall.dx=Math.abs(pall.dx)`. Absoluutväärtuse võtmise käsk Math.abs võtab arvult miinusmärgi

eest ära - juhul, kui see peaks olema sinna sattunud.

```
function p6rka(pall){ //Põrkab üle läinud servast tagasi
  if(pall.x-pall.r<0){pall.dx=Math.abs(pall.dx);}
  if(pall.x+pall.r>t.width){pall.dx=-Math.abs(pall.dx);}
  if(pall.y-pall.r<0){pall.dy=Math.abs(pall.dy);}
  if(pall.y+pall.r>t.height){pall.dy=-Math.abs(pall.dy);}
}
```

Liikumise juures tuleb siis kõigepealt kontrollida, kas liigutatav pall on alas sees. Juhul kui mitte, ehk käsu ees hüüumärk, mis tulemuse vastupidiseks keerab, siis vaja lasta pallil põrgata, vajalikku suunda muuta.

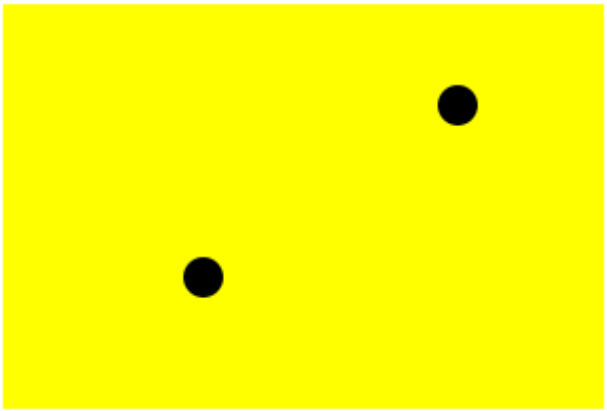
```
 if(!kasSees(pallid[i])){
    p6rka(pallid[i]);
 }
```

Nüüd jällegi peaks kood tervikuna juba mõistetav olema.

```
<!doctype html>
<html>
  <head>
    <title>Pallid</title>
        <script>
           var pallid=[
             {"x":10 , "y":50, "r":10, "dx":1, "dy":0},
             {"x":100, "y":30, "r":10, "dx":0, "dy":1}
           ];


           var t, g; //tahvel, graafiline kontekst

           function algus(){
             t=document.getElementById("tahvel");
                 g=t.getContext("2d");
                 setInterval('liigu()', 100);
           }

           function joonista(){
                 g.clearRect(0, 0, t.width, t.height);
                 for(var i=0; i<pallid.length; i++){
                   g.beginPath();
                   g.arc(pallid[i].x, pallid[i].y, pallid[i].r,
                          0, 2*Math.PI, true);
                   g.fill();
                 }
           }

           function liigu(){
             for(var i=0; i<pallid.length; i++){
               if(!kasSees(pallid[i])){
                 p6rka(pallid[i]);
               }
               pallid[i].x+=pallid[i].dx;
               pallid[i].y+=pallid[i].dy;
             }
             joonista();
           }

           function kasSees(pall){
             if(pall.x-pall.r<0){return false;}
                 if(pall.x+pall.r>t.width){return false;}
```

```
            if(pall.y-pall.r<0){return false;}
            if(pall.y+pall.r>t.height){return false;}
            return true;
          }

          function p6rka(pall){ //Põrkab üle läinud servast tagasi
            if(pall.x-pall.r<0){pall.dx=Math.abs(pall.dx);}
            if(pall.x+pall.r>t.width){pall.dx=-Math.abs(pall.dx);}
            if(pall.y-pall.r<0){pall.dy=Math.abs(pall.dy);}
            if(pall.y+pall.r>t.height){pall.dy=-Math.abs(pall.dy);}
          }
        </script>
  </head>
  <body onload="algus()">
      <canvas id="tahvel" width="300" height="200"
                  style="background-color:yellow" ></canvas><br />

  </body>
</html>
```

Pildi pealt palju ei näe, aga töötava rakenduse juures paistab, et kui pall jõuab servani, siis ta vahetab suunda. Kumbki omasoodu.



Ülesandeid

- Pane näide tööle
- Muuda pallide raadiused erinevateks ning veendu, et igaüks arvestab põrkamisel omi mõõtmeid
- Pane servadest põrkama liikuvad kriipsujukud. Kriipsujukude suurust määravad raadiuse asemel eraldi pikkust ja laiust tähistavad väärtused. Hoolitse, et ka põrkamisel arvestataks kriipsujukude pikkust ja laiust. Vihje: mõtle, milline on "kuum punkt", mida meeles peetakse ning mille suhtes kriipsujuku koordinaate arvestatakse.

Raskuskiirendus ja põrkamine

Reaalsete liikumiste ja hüpete juures tuleb arvestada ka kukkumisi ja kiirenemist kukkumisel. Kui liikuvaid kujundeid on palju, siis kehtib see neist igaühe puhul. Selle arvutamisel aitab y-suunaline kiirendus ehk sammu pikkuse kasv iga kaadri juures. Lihtsamal juhul võib selle väärtuse määrata katseliselt, et tulemus võimalikult reaalne välja paistaks.

```
var ykiirendus=0.1;
```

Kiirenduse mõjumiseks saab selle liikumisfunktsiooni juures igal korral otsa liita olemasolevale dy-väärtusele. Juhul kui dy on positiivne ehk pall liigub alla, siis allasuunaline kiirus kasvab. Kui dy on negatiivne ja pall liigub üles, siis kiirenduse liitmine viib sammu pikkust nulli suunas ehk ülesliikumise kiirus väheneb. Kuni lõpuks palli ülesliikumine peatub ning asutakse taas allapoole tulema.

```
function liigu(){
  for(var i=0; i<pallid.length; i++){
    if(!kasSees(pallid[i])){
      p6rka(pallid[i]);
    }
    pallid[i].dy+=ykiirendus;
    pallid[i].x+=pallid[i].dx;
    pallid[i].y+=pallid[i].dy;
  }
  joonista();
}
```

Põrkamise juures kipub kiirenduse lisamisel tekkima anomaalia: kui pall liigub piisavalt aeglaselt ning raskuskiirendus on piisavalt suur, siis mõnikord ületab raskuskiirendusest tekkinud allaliikumise samm põrkamise poolt antud ülespoole liikuva sammu suurust ning tulemuseks on, et pall hakkab väikeste jõnksudena läbi maapinna allapoole liikuma, kuni lõpuks kaob altpoolt silmast. Väljapääsuna aitab, et pärast allasuunalist põrget tõstetakse pall igal juhul maapinnale tagasi, ehk siis y-koordinaadiks saab maapinna asukohast ehk tahvli kõrgusest palli raadiuse jagu väiksem väärtus.

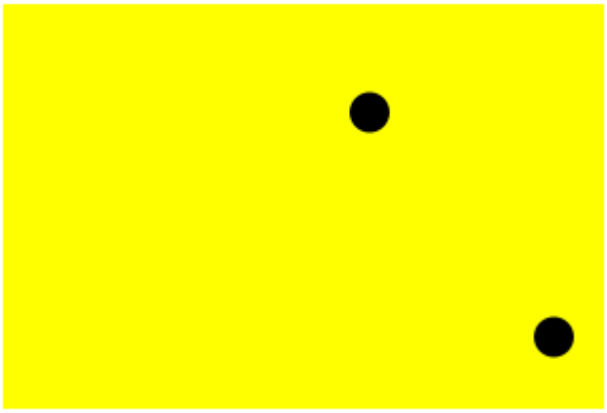
```
function p6rka(pall){
      if(pall.x-pall.r<0){pall.dx=Math.abs(pall.dx);}
      if(pall.x+pall.r>t.width){pall.dx=-Math.abs(pall.dx);}
      if(pall.y-pall.r<0){pall.dy=Math.abs(pall.dy);}
      if(pall.y+pall.r>t.height){
        pall.dy=-Math.abs(pall.dy);
        pall.y=t.height-pall.r;
      }
}
```

Töötav näide tervikuna.

```
<!doctype html>
<html>
  <head>
    <title>Pallid</title>
        <script>
           var pallid=[
             {"x":10 , "y":50, "r":10, "dx":1, "dy":-3},
             {"x":100, "y":30, "r":10, "dx":-1, "dy":1}
           ];


           var t, g; //tahvel, graafiline kontekst
           var ykiirendus=0.1;

           function algus(){
             t=document.getElementById("tahvel");
             g=t.getContext("2d");
```

```
            setInterval('liigu()', 50);
          }

          function joonista(){
                g.clearRect(0, 0, t.width, t.height);
                for(var i=0; i<pallid.length; i++){
                  g.beginPath();
                  g.arc(pallid[i].x, pallid[i].y, pallid[i].r,
                         0, 2*Math.PI, true);
                  g.fill();
                }
          }

          function liigu(){
            for(var i=0; i<pallid.length; i++){
              if(!kasSees(pallid[i])){
                p6rka(pallid[i]);
              }
              pallid[i].dy+=ykiirendus;
              pallid[i].x+=pallid[i].dx;
              pallid[i].y+=pallid[i].dy;
            }
            joonista();
          }

       function kasSees(pall){
             if(pall.x-pall.r<0){return false;}
             if(pall.x+pall.r>t.width){return false;}
             if(pall.y-pall.r<0){return false;}
             if(pall.y+pall.r>t.height){return false;}
             return true;
          }

       function p6rka(pall){
             if(pall.x-pall.r<0){pall.dx=Math.abs(pall.dx);}
             if(pall.x+pall.r>t.width){pall.dx=-Math.abs(pall.dx);}
             if(pall.y-pall.r<0){pall.dy=Math.abs(pall.dy);}
             if(pall.y+pall.r>t.height){
               pall.dy=-Math.abs(pall.dy);
               pall.y=t.height-pall.r;
             }
       }
        </script>
  </head>
  <body onload="algus()">
      <canvas id="tahvel" width="300" height="200"
                 style="background-color:yellow" ></canvas><br />

  </body>
</html>
```

Pilt püsib paigal, aga rakendust jälgides näeb, kuidas pallid kukkudes hoogu saavad ja servadest põrkavad.

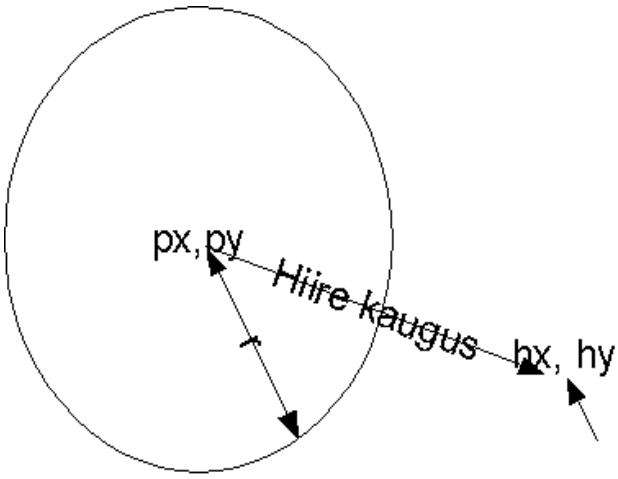


Ülesandeid

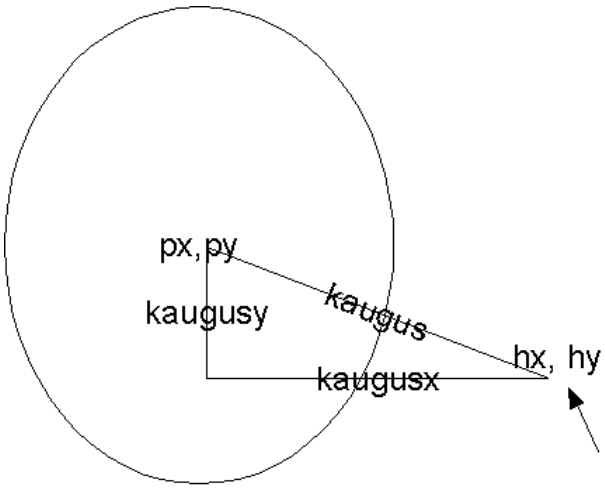
- Pane näide tööle.
- Lisa palle koos mitmesuguste algkohtade ja kiirustega.
- Muuda raskuskiirendust ning jälgi selle mõju rakenduse tööle.
- Tõmba eraldi joon maapinnaks ning lase pallidel sellelt põrgata
- Jäta pind ühest otsast lõpetamata. Tekkinud august saavad pallid alla kukkuda (põrkamisel kontrollitakse tingimust, et põrge tehakse vaid siis, kui pall on joone kohal).

Hiirevajutus ühel pallidest

Ümmarguse palli puhul piisab tabamise kindlaks tegemiseks kontrollida, kas hiire asukoht on palli keskpaigast vähem kui raadiuse kaugusel. Meil on kasutada viis arvu: palli x ja y-koordinaat (px, py), hiire x ja y-koordinaat (hx, hy) ning palli raadius (r). Silma järgi on hea vaadata, kas hiir tabas palli või mitte. Arvuti peab aga arvudega hakkama saama. Õnneks on nende andmete puhul ülesanne täiesti lahenduv.



Hiire kauguse ringi keskpunktist saab leida kahe punkti vahelise kaugusena tasandil. Selle juures omakorda aitab täisnurkse kolmnurga pikimat külge arvutada lubav Pythagorase teoreem.



Kolmnurga kaatetite pikkused saab koordinaatide lahutamise teel: hx-px on hiire asukoha ning ringi keskpunkti x-i suunaline kaugus, hy-py y-suunaline kaugus. Kas tema väärtus on pluss- või miinusmärgiga - see sõltub juba, kus suunas hiir ringi keskpunktist on. Aga õnneks arvu ruutu võtmisel arvu märk ei muuda tulemust - nii saab kauguse sarnaselt kätte ka siis, kui hiir juhtub pallist vasakul pool olema.

Koodina vormistatuna näeb arvutus välja allpool järgmine:

```
function hiirAlla(e){
  var tahvlikoht=t.getBoundingClientRect();
  var hx=e.clientX-tahvlikoht.left;
  var hy=e.clientY-tahvlikoht.top;
  for(var i=0; i<pallid.length; i++){
     var kaugusx=hx-pallid[i].x;
     var kaugusy=hy-pallid[i].y;
     var kaugus=Math.sqrt(kaugusx*kaugusx+kaugusy*kaugusy);
     if(kaugus<pallid[i].r){pallid[i].r=pallid[i].r*0.8;}
  }
  joonista();
}
```

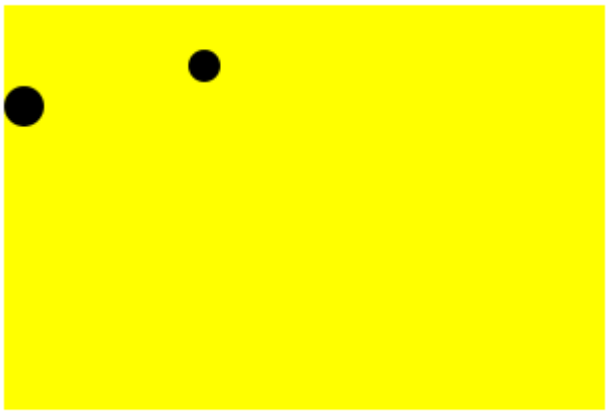
Muutujad `kaugusx` ja `kaugusy` arvutatakse kõigepealt välja, et hiljem poleks ruutu võtmise ajal vaja valemit mitu korda sisse kirjutada - `hx-pallid[i].x` annab punktide `x`-i suunalise kauguse. Math.sqrt arvutab ruutjuure ning edasi võib arvutatud kaugust juba võrrelda konkreetse palli raadiusega. Kui kaugus on piisavalt väike, siis järelikult hiir tabas palli ning palliga võib midagi ette võtta. Praegusel juhul tehakse pall veidi väiksemaks, ehk tema raadius korrutatakse 0.8ga. Nii võetakse suurest pallist rohkem maha ja väikesest pallist vähem. Vajutuse järgi tuleb eraldi välja kutsuda ka joonistuskäsk, et arvutuse tulemust saaks ka ekraanil näha.

Edasi juba kood tervikuna.

```
<!doctype html>
<html>
  <head>
    <title>Pallid</title>
        <script>
           var pallid=[
             {"x":10 , "y":50, "r":10, "dx":1, "dy":0},
             {"x":100, "y":30, "r":10, "dx":0, "dy":1}
           ];
```

```
            var t, g; //tahvel, graafiline kontekst

            function algus(){
              t=document.getElementById("tahvel");
              g=t.getContext("2d");
              joonista();
            }

            function joonista(){
                  g.clearRect(0, 0, t.width, t.height);
                  for(var i=0; i<pallid.length; i++){
                    g.beginPath();
                    g.arc(pallid[i].x, pallid[i].y, pallid[i].r,
                           0, 2*Math.PI, true);
                    g.fill();
                  }
            }

            function hiirAlla(e){
              var tahvlikoht=t.getBoundingClientRect();
              var hx=e.clientX-tahvlikoht.left;
              var hy=e.clientY-tahvlikoht.top;
              for(var i=0; i<pallid.length; i++){
                 var kaugusx=hx-pallid[i].x;
                 var kaugusy=hy-pallid[i].y;
                 var kaugus=Math.sqrt(kaugusx*kaugusx+kaugusy*kaugusy);
                 if(kaugus<pallid[i].r){pallid[i].r=pallid[i].r*0.8;}
              }
              joonista();
            }
        </script>
  </head>
  <body onload="algus()">
      <canvas id="tahvel" width="300" height="200"
                  style="background-color:yellow"
               onmousedown="hiirAlla(event)" ></canvas><br />

  </body>
</html>
```

Pildil näha, kuidas üks pall on hiirevajutuse tõttu väiksemaks jäänud.



Ülesandeid

- Tee näide läbi.
- Lisa mitmesuguse asukoha ja suurusega palle ning veendu, et lahendus töötab.
- Pane pallid muutuma hiirevajutuse puhul suuremaks.
- Pane pall hüppama hiirevajutuse puhul paremale
- Pane pall hüppama hiirevajutuse puhul hiirest eemale. St, et hiirega palli serva juurde vajutades valib pall uue asukoha, nagu oleks teda sealt serva juurest lükatud. Vihje: endiselt saab kätte palli keskpunkti ning hiire vahelise x-i ja y-i suunalise kauguse ning nende põhjal saab juba arvutada, kui palju ja millises suunas on mõistlik palli asukohta muuta.
- Joonista ekraanile lisaks pallidele üks suur ringjoon. Määra pallidele algul juhuslikud asukohad. Nõnda hiirega lükates tuleb pallid suure ringi sisse saada. Programm teatab, kui kõik pallid on suure ringi sees.

Liikumine ja hiirevajutus üheskoos

Eelnevad näited saab omavahel kokku tõsta. Igasugu ühendamiste puhul peab sageli vaatama, et osad omavahel tülli ei läheks. Ka siin on üheks ohuks näiteks, et samal ajal, kui setInterval-käsu kaudu tööle pandud liikumine ning kasutajalt tulnud hiirevajutuse teada püüavad samal ajal samade pallide andmeid muuta, siis võib sellega muresid tekkida. Õnneks praegusel juhul Javaskript hoolitseb, et käsud korraga tööle ei läheks ning hiire andmeid asutakse töötlema alles pärast seda, kui järjekordne liikumise samm tehtud. Tuleb lihtsalt funktsioonid sobivatele kohtadele paigutada. Ning kuna praegu kasutasid mõlemad näited samade nimedega muutujaid, siis õnneliku juhuse tõttu hakkabki rakendus tööle. Muul juhul niisama katsetades võib ühendamisel vaja olla näiteks jälgida ja sättida, et millise muutuja nime alt millised väärtused kätte saadakse.

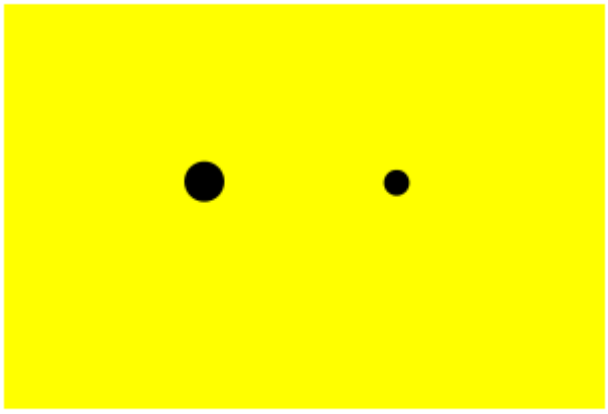
Funktsioonide järjekord Javaskriptis pole üldiselt tähtis - seetõttu võib neid lehe päiseossa vabalt sobivasse kohta paigutada lihtsalt selle järgi, kuidas neid mugavam leida on.

Näide tervikuna.

```
<!doctype html>
<html>
  <head>
    <title>Pallid</title>
        <script>
           var pallid=[
             {"x":10 , "y":50, "r":10, "dx":1, "dy":0},
             {"x":100, "y":30, "r":10, "dx":0, "dy":1}
           ];


           var t, g; //tahvel, graafiline kontekst
           var ykiirendus=0.1;

           function algus(){
             t=document.getElementById("tahvel");
             g=t.getContext("2d");
             setInterval('liigu()', 50);
           }

           function joonista(){
                 g.clearRect(0, 0, t.width, t.height);
                 for(var i=0; i<pallid.length; i++){
                   g.beginPath();
                   g.arc(pallid[i].x, pallid[i].y, pallid[i].r,
```

```
                    0, 2*Math.PI, true);
            g.fill();
          }
        }

        function hiirAlla(e){
          var tahvlikoht=t.getBoundingClientRect();
          var hx=e.clientX-tahvlikoht.left;
          var hy=e.clientY-tahvlikoht.top;
          for(var i=0; i<pallid.length; i++){
            var kaugusx=hx-pallid[i].x;
            var kaugusy=hy-pallid[i].y;
            var kaugus=Math.sqrt(kaugusx*kaugusx+kaugusy*kaugusy);
            if(kaugus<pallid[i].r){pallid[i].r=pallid[i].r*0.8;}
          }
          joonista();
        }

        function liigu(){
          for(var i=0; i<pallid.length; i++){
            if(!kasSees(pallid[i])){
              p6rka(pallid[i]);
            }
            pallid[i].dy+=ykiirendus;
            pallid[i].x+=pallid[i].dx;
            pallid[i].y+=pallid[i].dy;
          }
          joonista();
        }

        function kasSees(pall){
          if(pall.x-pall.r<0){return false;}
          if(pall.x+pall.r>t.width){return false;}
          if(pall.y-pall.r<0){return false;}
          if(pall.y+pall.r>t.height){return false;}
          return true;
        }

        function p6rka(pall){
          if(pall.x-pall.r<0){pall.dx=Math.abs(pall.dx);}
          if(pall.x+pall.r>t.width){pall.dx=-Math.abs(pall.dx);}
          if(pall.y-pall.r<0){pall.dy=Math.abs(pall.dy);}
          if(pall.y+pall.r>t.height){
            pall.dy=-Math.abs(pall.dy);
            pall.y=t.height-pall.r;
          }
        }

      </script>
  </head>
  <body onload="algus()">
      <canvas id="tahvel" width="300" height="200"
                  style="background-color:yellow"
               onmousedown="hiirAlla(event)" ></canvas><br />

  </body>
</html>
```

Tulemusena siis pallid liiguvad, põrkavad ning neile vajutamisel palli suurus väheneb.



Ülesandeid

- Pane näide tööle
- Palli juures on meeles tema värv. Hiirevajutusega saab palle kordamööda punaseks ja mustaks muuta.
- Mustadele pallidele mõjub suurem kiirendus kui punastele.
- Arvuti teatab juhusliku arvu, mitu palli peaks mustad, mitu punased olema (kokku teevad need ikka algse pallide arvu). Kasutaja saab vajutustega pallide värve muuta. Arvuti teatab, kui sobiv arv musti ja punaseid on saadud.

Kellaaeg

Ajast on arvutiprogrammide juures kasu mitut moodi. Vahel on lihtsalt hea kuupäeva ja kellaaega näha. Vahel näidatakse sündmusi, mis mingi aja jooksul toimunud või tulemas. Siis jälle saab mõõta aega, et kui palju mingist hetkest kulunud on. Javaskripti juures aitab nende toimetuste juures sisseehitatud objekt tüübist Date. Kõige lihtsamal juhul väljastatakse hetke kuupäev ja kellaaeg.

```
<!doctype html>
<html>
  <head>
    <title>Aeg</title>
    <script>
      function algus(){
         document.getElementById("vastus").innerHTML=new Date();
      }
    </script>
  </head>
  <body onload="algus()">
    <div id="vastus">

    </div>
  </body>
</html>
```

Väljund:

Thu Oct 18 2012 21:09:53 GMT+0300 (FLE Daylight Time)

Ajavahemik

Aja arvutamise juures tuleb kasuks Date-objekti käsklus getTime(). Käsklus tagastab millisekundite arvu alates 1. jaanuarist 1970, mida peetakse mitmegi programmeerimiskeele puhul "aegade alguseks". Ehkki käsklus väljastab vaid ühe arvu, saab seda programmide sees mitmel moel kasulikult pruukida. Kulunud aja arvestamiseks tasub meelde jätta algusaeg. Edasi saab juba iga funktsioonikäivituse juures leida uus kaugus aegade algusest millisekundites ning arvestada, kui palju vahepeal aega on kulunud. Kui millisekundites kaugus käes, siis edasi saab seda arvutuste teel juba sobivatesse ühikutesse teisendada.

```
<!doctype html>
<html>
  <head>
    <title>Aeg</title>
    <script>
      var algusaeg=new Date().getTime();
      function algus(){
         setInterval('liigu()', 1000);
      }
      function liigu(){
         document.getElementById("vastus1").innerHTML=new Date();
         document.getElementById("vastus2").innerHTML=
           new Date().getTime() + " millisekundit 1. jaanuarist 1970";
         document.getElementById("vastus3").innerHTML=
           "Kulunud "+parseInt((new Date().getTime()-algusaeg)/1000)+" sek ";
      }
    </script>
  </head>
  <body onload="algus()">
    <div id="vastus1"></div>
    <div id="vastus2"></div>
    <div id="vastus3"></div>
  </body>
</html>
```

Väljund:

Thu Oct 18 2012 21:11:07 GMT+0300 (FLE Daylight Time)
1350583867598 millisekundit 1. jaanuarist 1970
Kulunud 49 sek

Ülesandeid

- Pane ajaga seotud näited käima
- Näita, mitu minutit ja mitu sekundit on kulunud lehe avamise algusest. Täisosa leidmiseks sobib käsklus parseInt.
- Kasutajale näidatakse juhuslikku arvu, mitme millisekundi möödumist ta peaks püüdma tabada. Kui kasutaja arvates on sobiv aeg, siis ta vajutab nuppu. Arvuti näitab, kui pikka ajavahemikku inimeselt küsiti, kui palju kulus tegelikult, kui palju oli nende aegade vahe ning mitu protsenti eksiti.
- Püüa Internetist leida näide, kuidas töötab Javaskripti Date-objekti käsklus getHours(). Pane ekraanile tiksuma kell, kus näha tunnid, minutid ja sekundid.

Liikumine, hiir ja aeg

Ka ajaarvestuse saab eelnevale rakendusele küllalt viisakasti lisada. Algusesse muude muutujate kõrvale tuleb algusaeg, pärast hea sellega võrreldes arvestada, et kui palju on lehe töö algusest aega kulunud. Muutuja pihtasloendur aitab kokku lugeda, et mitu korda mõnd palli on tabatud.

Võimalusel on hea keele sõnakujudega arvestada. Hädapärast ju saab teadetega hakkama kujul "avatud on 1 leht(e)" - aga jäägu sellised sõnaväänamised olukordadesse, kus tuleb koodi muutmata tõlkida võõrkeelset rakendust ning pole võimalik vastavalt arvule sõna sättida. Viisakamal juhul aga tasub tingimuslausega vaadata, et ikka tulemusele vastav sõna kirja saaks. Üheks mooduseks on anda algul muutujale üks väärtus ning siis edasi tingimuslause abil kontrollida, kas seda äkki asendada on vaja. Hiljem lauses saab siis muutuja kaudu sobival kujul sõna välja trükkida.

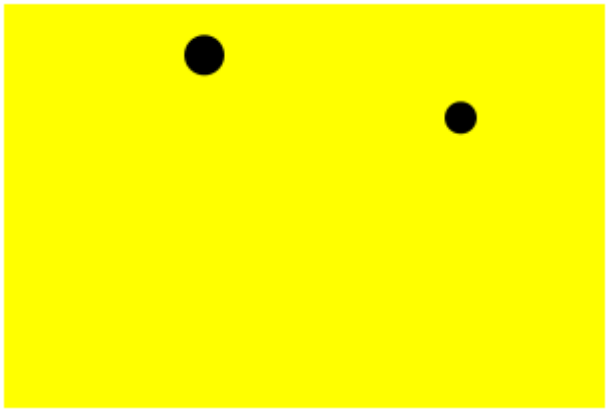
```
var lopusona="pall";
if(pihtasloendur>1){
  lopusona="palli";
}
document.getElementById("vastus2").innerHTML=
   "Tabatud "+pihtasloendur+" "+lopusona;
}
```

Kood tervikuna, mille abil saab hiirega tabada liikuvaid palle ning loetakse kokku, kui kaua aega läinud ning mitu palli pihta saadud.

```
<!doctype html>
<html>
  <head>
    <title>Pallid</title>
        <script>
           var pallid=[
             {"x":10 , "y":50, "r":10, "dx":1, "dy":0},
             {"x":100, "y":30, "r":10, "dx":0, "dy":1}
           ];


           var t, g; //tahvel, graafiline kontekst
           var ykiirendus=0.1;
           var algusaeg=new Date().getTime();
           var pihtasloendur=0;

           function algus(){
             t=document.getElementById("tahvel");
                 g=t.getContext("2d");
                 setInterval('liigu()', 50);
           }

           function joonista(){
                 g.clearRect(0, 0, t.width, t.height);
                 for(var i=0; i<pallid.length; i++){
                   g.beginPath();
                   g.arc(pallid[i].x, pallid[i].y, pallid[i].r,
                          0, 2*Math.PI, true);
                   g.fill();

                 }
           }

           function hiirAlla(e){
             var tahvlikoht=t.getBoundingClientRect();
```

```
            var hx=e.clientX-tahvlikoht.left;
            var hy=e.clientY-tahvlikoht.top;
            for(var i=0; i<pallid.length; i++){
              var kaugusx=hx-pallid[i].x;
              var kaugusy=hy-pallid[i].y;
              var kaugus=Math.sqrt(kaugusx*kaugusx+kaugusy*kaugusy);
              if(kaugus<pallid[i].r){
                pallid[i].r=pallid[i].r*0.8;
                pihtasloendur++;
                var lopusona="pall";
                if(pihtasloendur>1){
                  lopusona="palli";
                }
                document.getElementById("vastus2").innerHTML=
                   "Tabatud "+pihtasloendur+" "+lopusona;
                }
            }
            joonista();
          }

          function liigu(){
            for(var i=0; i<pallid.length; i++){
              if(!kasSees(pallid[i])){
                p6rka(pallid[i]);
              }
              pallid[i].dy+=ykiirendus;
              pallid[i].x+=pallid[i].dx;
              pallid[i].y+=pallid[i].dy;
            }
            document.getElementById("vastus1").innerHTML="Kulunud "+
                    parseInt((new Date().getTime()-algusaeg)/1000)+" sek ";
            joonista();
          }

          function kasSees(pall){
              if(pall.x-pall.r<0){return false;}
              if(pall.x+pall.r>t.width){return false;}
              if(pall.y-pall.r<0){return false;}
              if(pall.y+pall.r>t.height){return false;}
              return true;
          }

          function p6rka(pall){
            if(pall.x-pall.r<0){pall.dx=Math.abs(pall.dx);}
            if(pall.x+pall.r>t.width){pall.dx=-Math.abs(pall.dx);}
            if(pall.y-pall.r<0){pall.dy=Math.abs(pall.dy);}
            if(pall.y+pall.r>t.height){
            pall.dy=-Math.abs(pall.dy);
            pall.y=t.height-pall.r;
          }
       }

        </script>
  </head>
  <body onload="algus()">
      <canvas id="tahvel" width="300" height="200"
                  style="background-color:yellow"
             onmousedown="hiirAlla(event)" ></canvas><br />
      <div id="vastus1"></div>
      <div id="vastus2"></div>
  </body>
</html>
```

Rakenduse ekraanikuva:



Kulunud 11 sek
Tabatud 1 pall

Ülesandeid

- Pane näide tööle
- Korralda nõnda, et iga viie sekundi tagant lähevad pallid uuesti ühesuurusteks tagasi. Jäetakse meelde suurim tabamuste arv, mida ühe sellise viiesekundise ajaga õnnestunud saada.
- Igal järgmisel korral ühesuuruseks minnes on pallid eelmisest korrast natuke väiksemad ning tabamiseks mõeldud aeg kasvab.

Tulemuste loetelu

Edetabelid ja tulemuste loetelud kipuvad mängude ja võistluste juurde kuuluma - ehkki nendega põhjust ka mujal kokku puutuda. Eelnevalt otsingu juures koostasime mõne, aga siin väike meeldetuletus, kuidas mälus olevaid andmeid veebilehel esitada kannatab. Kõigepealt koostatakse näitamise tarbeks muutuja ning for-tsükli abil HTML-kujul loetelu ning edasi näidatakse selle väärtus kihi innerHTML-omaduse kaudu lehele. Kuna võrrelduna eelneva <ul> - loetelu asemel (unordered list) kasutatakse <ol> - loetelu (ordered list), siis kuvamisel näidatakse automaatselt välja osaliste järjekorranumbrid.

```
<!doctype html>
<html>
  <head>
    <title>Aeg</title>
    <script>
      var tulemused=[
        {"eesnimi":"Juku", "punkte": 7},
        {"eesnimi":"Kati", "punkte": 6},
        {"eesnimi":"Mati", "punkte": 5}
      ];
      function algus(){
         var t="<ol>";
         for(var i=0; i<tulemused.length; i++){
           t+="<li>"+tulemused[i].eesnimi+": "+
             tulemused[i].punkte+" punkti</li>";
```

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

    </div>
  </body>
</html>
```

1. Juku: 7 punkti
2. Kati: 6 punkti
3. Mati: 5 punkti

Loetellu lisamine

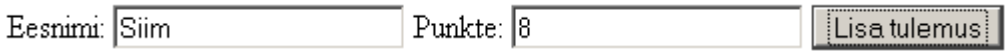
Eelnevalt vaid näidati massiivis olevaid andmeid. Töötava rakenduse käigus aga andmed täienevad ja muutuvad ning ka neid on põhjust sobival ajal välja näidata.
Tulemusi saab massiivi lisada push-käsklusega. Kuna siin iga massiivi element koosneb osaleja eesnimest ja tema punktide arvust, siis tasub järjepidevuse hoidmiseks ka uue osaleja andmed samal kujul lisada. Lihtsuse mõttes võtame esialgu andmed tekstikastidest. Looksulgude vahel luuakse üks osaleja kirje mil väljadeks "eesnimi" ja "punkte". Kogu see uus kirje läheb push-käsu ümarsulgude vahele, mille tulemusena ta massiivi lõppu jõuab. Pärast lisamist on põhjust tulemused uuesti kuvada, et ikka näeks mis lisatud.

```
function lisaTulemus(){
  tulemused.push(
    {
      "eesnimi":document.getElementById("kast1").value,
      "punkte": parseInt(document.getElementById("kast2").value)
    }
  );
  kuvaTulemused();
}
```

Ning näite kood tervikuna.

```
<!doctype html>
<html>
  <head>
    <title>Aeg</title>
    <script>
      var tulemused=[
        {"eesnimi":"Juku", "punkte": 7},
        {"eesnimi":"Kati", "punkte": 6},
        {"eesnimi":"Mati", "punkte": 5}
      ];
      function algus(){
```

```
        kuvaTulemused();
      }
      function kuvaTulemused(){
         var t="<ol>";
         for(var i=0; i<tulemused.length; i++){
           t+="<li>"+tulemused[i].eesnimi+": "+
             tulemused[i].punkte+" punkti</li>";
         }
         t+="</ol>";
         document.getElementById("vastus").innerHTML=t;
      }
      function lisaTulemus(){
        tulemused.push(
          {
            "eesnimi":document.getElementById("kast1").value,
            "punkte": parseInt(document.getElementById("kast2").value)
          }
        );
        kuvaTulemused();
      }
    </script>
  </head>
  <body onload="algus()">
    Eesnimi: <input type="text" id="kast1" />
    Punkte: <input type="text" id="kast2" />
    <input type="button" value="Lisa tulemus" onclick="lisaTulemus()" />
    <div id="vastus">

    </div>
  </body>
</html>
```



1. Juku: 7 punkti
2. Kati: 6 punkti
3. Mati: 5 punkti
4. Siim: 8 punkti

Ülesandeid

- Pane näide tööle, muuda andmeid.
- Trüki välja vaid need osalejad, kel on rohkem kui viis punkti
- Koosta massiiv, kus on kirjas vaid punktide arvud, kuva arvude loetelu lehele. Vihjena sobib vaadata nimede loetelu otsingu peatükis. Arvudele pole jutumärke ümber vaja.
- Võimalda veebilehe kaudu arve loetellu lisada. Koos lisamisega kuva loetelu olemasolevatest arvudest.
- Püüa leida loetelust suurim punktide arv. Vihjeks aitab abimuutuja, mille väärtuseks kõigepealt esimene väärtus. Siis iga järgmise väärtuse puhul kontrollitakse, kas leitud väärtus on abimuutuja omast suurem. Kui jah, siis omistatakse leitud väärtus abimuutujasse. Pärast kõikide väärtuste võrdlemist on nõnda suurim käes ja võib välja trükkida.

Tulemuste järjestamine

Paljude kasutajatega edetabelis tuntakse sageli huvi oma koha vastu. Õnneks mõistab Javaskript väärtusi ka mõne tunnuse alusel ritta sättida ning sealtkaudu järjestatud loetelu kokku panna. Tavalise massiivi puhul võib öelda lihtsalt sort. Ehk siis eesnimede järjestamiseks

```
var eesnimed=new Array("Siim", "Anu", "Sass");
eesnimed.sort();
```

ning tulemuseks ongi Anu eespool ja poisid tagapool. Kui aga igal kirjel mitu tunnust - praegusel juhul "eesnimi" ja "punkte", siis ei tea Javaskript, mille järgi järjestada soovitakse, see tuleb eraldi teada anda. Teada andmiseks tuleb sorteerimisfunktsioonil aidata võrrelda korraga kahte massiivielementi omavahel - et kumb neist peaks pärast sortimist järjekorras eespool olema. Võrdluse vastusena tagastatakse arv: nullist väiksem vastus tähendab, et eespool peaks olema esimene võrreldav element, nullist suurem vastus et teine võrreldav element. Ning kui vastava tunnuse järgi järjestamisel kahe võrreldava elemendi väärtustel vahet pole, siis tuleb funktsioonil tagastada arv 0. Õnneks saab sellise võrdluse lihtsasti teha lahutustehte abil. Arvutus `t2.punkte-t1.punkte` teeb punktide arvutamisel just seda mida vaja. Kuna soovitakse suurema punktisummaga osalejad panna loetelus ettepoole, siis t1 peaks jääma loetelus eespoole vaid siis, kui seal on rohkem punkte kui t2 juures. Sellisel juhul peaks väljastatav tulemus olema alla nulli ning nii see tulebki, kui suurema punktiväärtusega t1 on miinusmärgi taga.

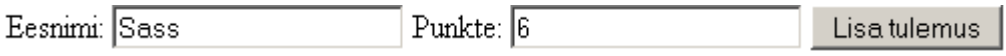
```
function sordiTulemusedPunktideJ2rgi(){
  tulemused.sort(function(t1, t2){return t2.punkte-t1.punkte});
}
```

Edasi taas töötav rakendus.

```
<!doctype html>
<html>
  <head>
    <title>Aeg</title>
    <script>
      var tulemused=[
        {"eesnimi":"Juku", "punkte": 7},
        {"eesnimi":"Kati", "punkte": 8},
        {"eesnimi":"Mati", "punkte": 5}
      ];
      function algus(){
        kuvaTulemused();
      }
      function sordiTulemusedPunktideJ2rgi(){
        tulemused.sort(function(t1, t2){return t2.punkte-t1.punkte});
      }
      function kuvaTulemused(){
         sordiTulemusedPunktideJ2rgi();
         var t="<ol>";
         for(var i=0; i<tulemused.length; i++){
           t+="<li>"+tulemused[i].eesnimi+": "+
             tulemused[i].punkte+" punkti</li>";
         }
         t+="</ol>";
         document.getElementById("vastus").innerHTML=t;
```

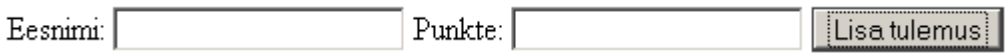
```
      }
      function lisaTulemus(){
        if(document.getElementById("kast1").value.length==0){return;}
        tulemused.push(
          {
            "eesnimi":document.getElementById("kast1").value,
            "punkte": parseInt(document.getElementById("kast2").value)
          }
        );
        document.getElementById("kast1").value="";
        document.getElementById("kast2").value="";
        kuvaTulemused();
      }
    </script>
  </head>
  <body onload="algus()">
    Eesnimi: <input type="text" id="kast1" />
    Punkte: <input type="text" id="kast2" />
    <input type="button" value="Lisa tulemus" onclick="lisaTulemus()" />

    <div id="vastus">

    </div>
  </body>
</html>
```

Kõigepealt näha algul massiivi pandud tegelased.



1. Kati: 8 punkti
2. Juku: 7 punkti
3. Mati: 5 punkti

Neile lisatakse Sass kuune punktiga. Ning omaloodud abifunktsiooniga sort-käsklus suudab Sassi punktide järgi just õigele kohale paigutada.



1. Kati: 8 punkti
2. Juku: 7 punkti
3. Sass: 6 punkti
4. Mati: 5 punkti

Ülesandeid

- Tee näide läbi
- Sorteeri sisestatud andmed nõnda, et vähem punkte saanud on eespool
- Sorteeri sisestatud andmed eesnimede pikkuste järgi kasvavasse järjekorda. Teksti pikkuse saab kätte käsuga length, näiteks inimene1.eesnimi.length

- Sorteeri sisestatud andmed eesnimede järgi tähestikulisse järjekorda. Sel puhul aitab võrdlemiseks if-lause ning sobivas suuruses arvu tagastamine. Näiteks
  `if(inimene1.eesnimi<inimene2.eesnimi){return -1;}`

Valmis mäng

Eraldi tükid nüüdseks mitut moodi läbi proovitud. Edasi juba tasub tulemus kasutatavaks mänguks viimistleda ning sobiv kasutajaskond leida. Või siis tehtut pruukida toimiva alusena, mille peale omi mitmesuguseid lahendusi looma hakata. Väidetakse, et pärast seda, kui lahendus kuidagi tööle saadud, kulub vähemalt veerand kui mitte pool tööd selle tarbeks, et kasutajatel sellega ka mugav toimetada oleks. Püüame siingi mõned täiendused teha, et lahendus rohkem "päris" mängu moodi välja paistaks.

Edetabeli lisandumisega on hea mäng konkreetseteks etappideks jagada. Ehk siis igale mängija saab omale piiratud aja, mille jooksul võimalikult rohkem palle tabada. Tema tulemus jäetakse meelde ning salvestatakse edetabelisse. Punktide/tabamuste arvu ei küsita enam tekstiaknast, vaid need saab kätte konkreetse mängu tulemuste kaudu. Andmete lisamiseks eraldi funktsioon lisaTulemus. Parameetrina on põhjust ette anda vaid kasutaja eesnimi, punktide arv tuleb muutujast pihtasloendur. Looksulgude vahel luuakse uus kirje ning lisatakse tulemuste loendisse.

```
function lisaTulemus(enimi){
  tulemused.push(
    {
      "eesnimi":enimi,
      "punkte": pihtasloendur
    }
  );
  kuvaTulemused();
}
```

Eraldi peetakse arvet mängu seisundi üle. Muutuja `m2ngK2ib` on `true` või `false` vastavalt sellele, kas kasutaja saab parajasti hiirega palle väiksemaks klõpsida või mitte. Kõigepealt ei saa. Kui aga kasutaja vajutab alustamise nupule, siis sellest hetkest alates saab. Kui mängu kestuse jagu aega mööda tiksunud, siis taas ei saa. Ning kui kasutaja on oma eesnime sisestanud ja punktid salvestunud, siis jääb mäng uue mängija algusnupuvajutust ootama. Selle toimumisel taastatakse algseaded ning too võib taas algusest peale pallikesi püüdma asuda. Algseadete taastamisel siis nullitakse pihtasloendur, võetakse uus algusaeg, peidetakse alustusnupp ning muudetakse kõikide pallide raadius algseks. Samuti määratakse seisinud m2ngKäib true-ks, et rakendus taas hiirevajutustele oleks valmis reageerima.

```
function m2nguAlgus(){
   pihtasloendur=0;
   algusaeg=new Date().getTime();
   document.getElementById("nupp1").style.visibility="hidden";
   for(var i=0; i<pallid.length; i++){
     pallid[i].r=algraadius;
   }
   m2ngK2ib=true;
}
```

Edasi võib juba rahus töötavat rakendust imetleda ning mõelda, et mis temaga peale võiks hakata.

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

```
  <script>
var pallid=[
       {"x":10 , "y":50, "r":10, "dx":1, "dy":0},
       {"x":100, "y":30, "r":10, "dx":0, "dy":1}
];

var tulemused=[
  {"eesnimi":"Juku", "punkte": 7},
  {"eesnimi":"Kati", "punkte": 8},
  {"eesnimi":"Mati", "punkte": 5}
];

var t, g; //tahvel, graafiline kontekst
var ykiirendus=0.1;
var algusaeg=new Date().getTime();
var pihtasloendur=0, m2ngK2ib=false;
var m2nguKestus=10000; //millisekundit
var algraadius=10;

function algus(){
       t=document.getElementById("tahvel");
       g=t.getContext("2d");
       setInterval('liigu()', 50);
}

function m2nguAlgus(){
   pihtasloendur=0;
   algusaeg=new Date().getTime();
   document.getElementById("nupp1").style.visibility="hidden";
   for(var i=0; i<pallid.length; i++){
     pallid[i].r=algraadius;
   }
   m2ngK2ib=true;
}

function joonista(){
    g.clearRect(0, 0, t.width, t.height);
    for(var i=0; i<pallid.length; i++){
      g.beginPath();
      g.arc(pallid[i].x, pallid[i].y, pallid[i].r,
                0, 2*Math.PI, true);
      g.fill();
    }
}

function hiirAlla(e){
   if(!m2ngK2ib){return;}
   var tahvlikoht=t.getBoundingClientRect();
   var hx=e.clientX-tahvlikoht.left;
   var hy=e.clientY-tahvlikoht.top;
   for(var i=0; i<pallid.length; i++){
      var kaugusx=hx-pallid[i].x;
      var kaugusy=hy-pallid[i].y;
      var kaugus=Math.sqrt(kaugusx*kaugusx+kaugusy*kaugusy);
      if(kaugus<pallid[i].r){
        pallid[i].r=pallid[i].r*0.8;
        pihtasloendur++;
        var lopusona="pall";
        if(pihtasloendur>1){
          lopusona="palli";
        }
        document.getElementById("vastus2").innerHTML=
          "Tabatud "+pihtasloendur+" "+lopusona;
      }
```

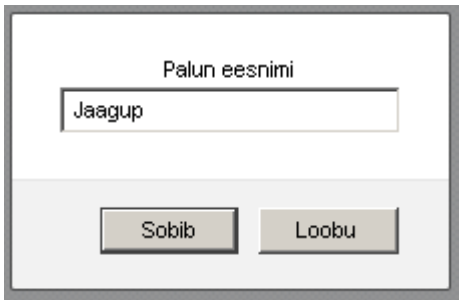
```
    }
    joonista();
}

function liigu(){
    if(!m2ngK2ib){return;}
    for(var i=0; i<pallid.length; i++){
      if(!kasSees(pallid[i])){
        p6rka(pallid[i]);
      }
      pallid[i].dy+=ykiirendus;
      pallid[i].x+=pallid[i].dx;
      pallid[i].y+=pallid[i].dy;
    }
    document.getElementById("vastus1").innerHTML=
      "Kulunud "+parseInt((new Date().getTime()-algusaeg)/1000)+" sek ";

    joonista();
    if(new Date().getTime()-algusaeg>m2nguKestus){
      m2ngK2ib=false;
      lisaTulemus(prompt("Palun eesnimi", "Niptiri"));
      document.getElementById("nupp1").style.visibility="visible";
    }
  }

  function kasSees(pall){
        if(pall.x-pall.r<0){return false;}
        if(pall.x+pall.r>t.width){return false;}
        if(pall.y-pall.r<0){return false;}
        if(pall.y+pall.r>t.height){return false;}
        return true;
  }

  function p6rka(pall){ //Põrkab üle läinud servast tagasi
        if(pall.x-pall.r<0){pall.dx=Math.abs(pall.dx);}
        if(pall.x+pall.r>t.width){pall.dx=-Math.abs(pall.dx);}
        if(pall.y-pall.r<0){pall.dy=Math.abs(pall.dy);}
        if(pall.y+pall.r>t.height){
          pall.dy=-Math.abs(pall.dy);
          pall.y=t.height-pall.r;
        }
  }


 function sordiTulemusedPunktideJ2rgi(){
      tulemused.sort(function(t1, t2){return t2.punkte-t1.punkte});
 }
 function kuvaTulemused(){
    sordiTulemusedPunktideJ2rgi();
    var t="<ol>";
    for(var i=0; i<tulemused.length; i++){
      t+="<li>"+tulemused[i].eesnimi+": "+
        tulemused[i].punkte+" punkti</li>";
    }
    t+="</ol>";
    document.getElementById("vastus3").innerHTML=t;
 }
 function lisaTulemus(enimi){
   tulemused.push(
     {
       "eesnimi":enimi,
       "punkte": pihtasloendur
     }
   );
```

```
        kuvaTulemused();
      }

        </script>
  </head>
  <body onload="algus()">
      <h1>Vajuta hiirega pallile</h1>
      <canvas id="tahvel" width="300" height="200"
              style="background-color:yellow"
              onmousedown="hiirAlla(event)" ></canvas><br />
      <div id="vastus1"></div>
      <div id="vastus2"></div>
      <div id="vastus3"></div>
      <input type="button" id="nupp1" value="Alusta" onclick="m2nguAlgus()" />
  </body>
</html>
```

Mängu algus tervitab meid tühja ekraaniga. Kasutaja saab ise valida, millal mängu alustada ja aeg käima panna. Alustusnupp kaob ning võimalik asuda palle püüdma.

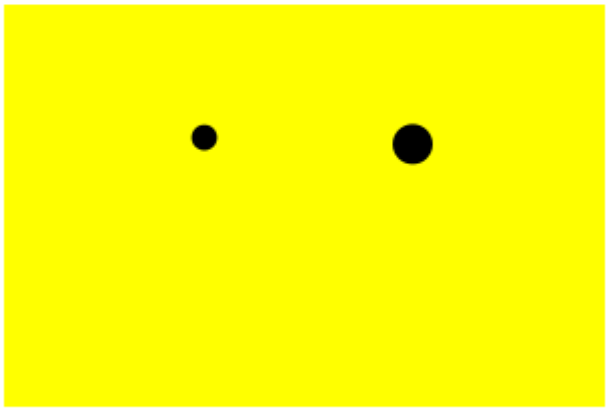
Hiirevajutuse tulemusena läheb pihtasaadud pall väiksemaks. Näidatakse sellel kasutajajal kulunud aega ning tabatud pallide arvu.

Kui kümme sekundit täis, siis küsitakse mängija eesnimi.



Sisestatud nimi koos punktidega läheb edetabelisse sobivale kohale ning võib oma tulemust teiste omadega võrrelda. Alla tekib nupp, millest saab uus mängija taas pallid liikuma lükata.

Vajuta hiirega pallile



Kulunud 10 sek
Tabatud 2 palli

1. Kati: 8 punkti
2. Juku: 7 punkti
3. Mati: 5 punkti
4. Jaagup: 2 punkti

Alusta

Ülesandeid

Pane näide tööle
Kui tulemuse salvestamisel juba olemasoleva nimega mängija saab parema tulemuse kui tal enne oli, siis salvestatakse tema nime alla uus tulemus. Kui aga sellist mängijat veel pole, siis lisatakse ta koos punktidega uuena.
Iga mängija juures loetakse lisaks saadud punktidele ka, et mitu mängu ta ühes avanenud aknas mänginud on. Vormista tulemused tabelina.

Palliga pallide tabamine

- Hulk palle põrkab ekraanil omasoodu. Ühe teist värvi palli liikumist saab kasutaja hiirega eraldi määrata.
- Iga iseliikuva palli peal on arv, mitu korda on ta juhitava pallis vastu põrganud. Kui juhitav pall puutub vastu ise liikuvat palli, siis vastav number suureneb ning tabatud pall hüppab juhuslikku kohta.
- Iseliikuvaid palle on rohelisi ja punaseid, mõlemil tabamisarvud peal. Rohelisi tuleb punktide saamiseks tabada, punastest hoiduda. Loetakse kokku, kui palju kumbagi tüüpi palle tabatud on.

Pallivise üle seina

- Tahvli keskel on ligikaudu poole tahvli kõrguseni ulatuv püstine sein (joon). Liikuv ja põrkav pall jääb seina puudutades pidama.

- Pall tekib algul vasakusse alanurka. Kasutaja saab palli suunda ja kiirust hiirega määrata. Palli saab loopida üle seina, aga seina tabamisel jääb pall kinni.
- Platsil on kaks seina, üks kasvab välja põrandast teine laest. Mängija ülesandeks on anda pallile selline paras hoog, et pall lendaks esimesest seinast üle ja teise alt läbi vajadusel maast põrgates.

Lahendusi ja täiendusi

Olgu ise katsetades või teisi juhendades on vahel hea vaadata ettevõetud ülesannete võimalikke lahendusi. Siia on toodud mitmete eelpool antud ülesannete lahendused mida siis hea kasutada oma lahendusega tagantjärele võrdlemiseks või vihjete otsimiseks olukorras, kus omalt poolt on mitut moodi proovitud, aga kuidagi ei taha tulemus õnnestuda.

Algus

Tervitamine ees- ja perekonnanimega

Evolutsioonis pidi tähtis roll olema kahekordistumisel. Ka programmide kirjutamise juures tasub vaadata, et kuidas lõike nõnda kopeerida ja muuta, et tulemus kasutuskõlbulikuks jääb. Tekstivälja koopia saab julgelt algse välja kõrvale lisada. Tervitamiseks piisab aga endiselt ühest nupust. Ja nupu küljes olevast ühest tervitusfunktsioonist. Lihtsalt funktsiooni sisse tuleb üksteise järele kirjutada kust andmed võetakse ning siis tulemused plussmärkidega üksteise taha siduda.

```
<!doctype html>
<html>
  <head>
    <title>Suhtlus arvutiga</title>
        <script>
           function tervita(){
              document.getElementById("vastus").innerHTML="Tere, "+
                     document.getElementById("eesnimi").value+" "+
                     document.getElementById("perekonnanimi").value;
           }
        </script>
  </head>
  <body>
     <h1>Tervitamine</h1>
         <div>
            Palun sisesta oma eesnimi:
             <input type="text" id="eesnimi" /> <br/>
            Palun sisesta oma perekonnanimi:
             <input type="text" id="perekonnanimi" /> <br/>
                 <input type="button" value="Sisesta" onclick="tervita()" />
         </div>
         <div id="vastus">

         </div>
  </body>
</html>
```

Tervitamine

Palun sisesta oma eesnimi: Juku
Palun sisesta oma perekonnanimi: Juurikas
Sisesta
Tere, Juku Juurikas

Käibemaksu arvutamine

```
<!doctype html>
<html>
  <head>
    <title>Arvutamine</title>
        <meta charset="utf-8" />
        <script>
           function arvuta(){
              document.getElementById("vastus").innerHTML=
                    document.getElementById("poesumma").value/120*20;
           }
        </script>
  </head>
  <body>
     <h1>Käibemaksu arvutamine</h1>
         <div>
           Hinnasilt:
           <input type="text" id="poesumma" />
           <input type="button" value="Sisesta" onclick="arvuta()" />
         </div>
         <div id="vastus">

         </div>
  </body>
</html>
```

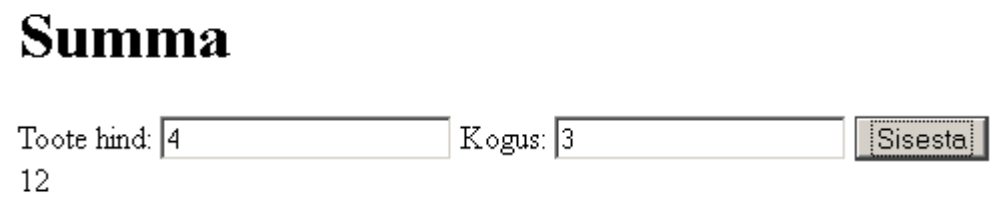
Käibemaksu arvutamine

Hinnasilt: 0.60 Sisesta
0.1

Toodete hindade summa

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

```
    <title>Arvutamine</title>
        <meta charset="utf-8" />
        <script>
           function arvuta(){
              document.getElementById("vastus").innerHTML=
                    document.getElementById("hind").value*
                        document.getElementById("kogus").value;
           }
        </script>
  </head>
  <body>
     <h1>Summa</h1>
         <div>
         Toote hind:
           <input type="text" id="hind" />
         Kogus:
           <input type="text" id="kogus" />
           <input type="button" value="Sisesta" onclick="arvuta()" />
         </div>
         <div id="vastus">

         </div>
  </body>
</html>
```



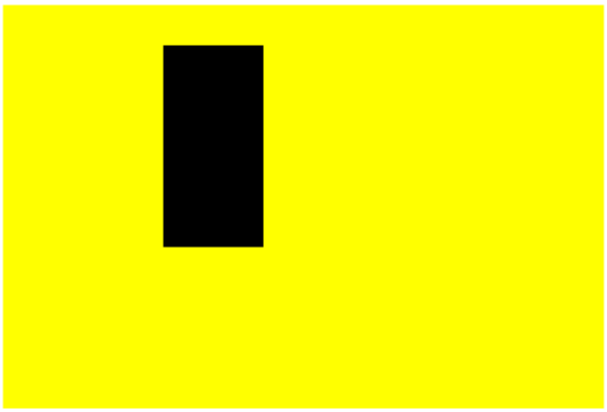
Joonistamine

Ristküliku asukoha ja suuruse määramine koodis

```
<!DOCTYPE html>
<html>
  <head>
    <title>Joonis</title>
    <script type="text/javascript">
      function joonista(){
         var g=document.getElementById("tahvel").getContext("2d");
         g.fillRect(80, 20, 50, 100); //x, y, laius, kõrgus
      }
    </script>
  </head>
  <body>
    <h1>Joonis</h1>

    <canvas id="tahvel" width="300" height="200"
            style="background-color:yellow"></canvas><br />
    <input type="button" value="Tee ristkülik" onclick="joonista()" />
```

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

Joonis

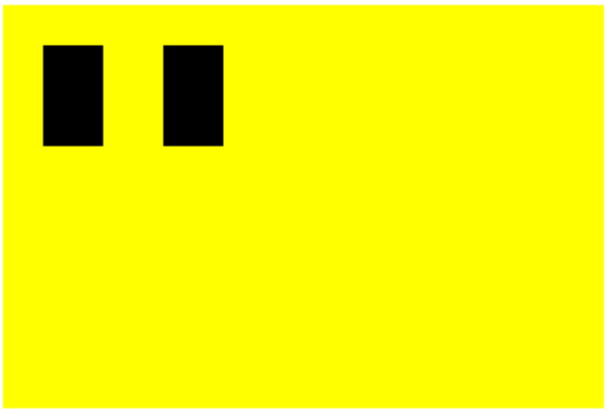


Tee ristkülik

Kaks ristkülikut

```
<!DOCTYPE html>
<html>
  <head>
    <title>Joonis</title>
    <script type="text/javascript">
      function joonista(){
         var g=document.getElementById("tahvel").getContext("2d");
         g.fillRect(20, 20, 30, 50); //x, y, laius, kõrgus
         g.fillRect(80, 20, 30, 50); //x, y, laius, kõrgus
      }
    </script>
  </head>
  <body>
    <h1>Joonis</h1>

    <canvas id="tahvel" width="300" height="200"
            style="background-color:yellow"></canvas><br />
    <input type="button" value="Tee kaks ristkülikut" onclick="joonista()" />
  </body>
</html>
```

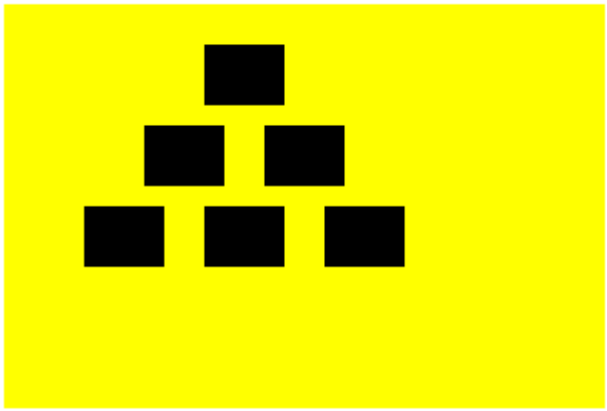
Joonis



Torn

```
<!DOCTYPE html>
<html>
  <head>
    <title>Joonis</title>
    <script type="text/javascript">
      function joonista(){
         var g=document.getElementById("tahvel").getContext("2d");
         g.fillRect(100, 20, 40, 30);

         g.fillRect(70 , 60, 40, 30);
         g.fillRect(130, 60, 40, 30);

                 g.fillRect( 40,100, 40, 30);
         g.fillRect(100,100, 40, 30);
         g.fillRect(160,100, 40, 30);

          }
    </script>
  </head>
  <body>
    <h1>Joonis</h1>

    <canvas id="tahvel" width="300" height="200"
            style="background-color:yellow"></canvas><br />
    <input type="button" value="Tee torn" onclick="joonista()" />
  </body>
</html>
```

Joonis



Tee torn

Ristküliku asukoha ja suuruse sisestamine kasutajalt

```
<!DOCTYPE html>
<html>
  <head>
    <title>Joonis</title>
    <script type="text/javascript">
      function joonista(){
         var g=document.getElementById("tahvel").getContext("2d");
         var x=parseInt(document.getElementById("xkast").value);
         var y=parseInt(document.getElementById("ykast").value);
         var laius=parseInt(document.getElementById("laiuskast").value);
         var korgus=parseInt(document.getElementById("korguskast").value);
                 g.clearRect(0, 0, 300, 200); //Tühjendab tahvli
         g.fillRect(x, y, laius, korgus); //x, y, laius, kõrgus
      }
    </script>
  </head>
  <body>
    <h1>Joonis</h1>
    <canvas id="tahvel" width="300" height="200"
           style="background-color:yellow"></canvas><br />
    x=<input type="text" id="xkast"  value="10" />
    y=<input type="text" id="ykast"  value="20" /><br />
    laius=<input type="text" id="laiuskast"  value="40" />
    kõrgus=<input type="text" id="korguskast" value="60" />
    <input type="button" value="Joonista" onclick="joonista()" />
  </body>
</html>
```

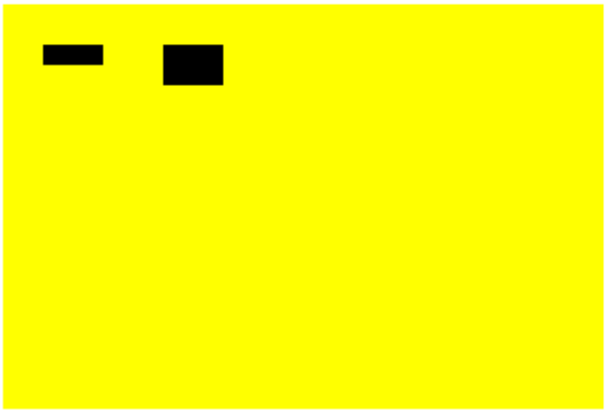
Kaks muudetava laiusega ristkülikut

```
<!DOCTYPE html>
<html>
  <head>
    <title>Joonis</title>
    <script type="text/javascript">
      function joonista(){
         var g=document.getElementById("tahvel").getContext("2d");
         var arv1=parseInt(document.getElementById("arv1kast").value);
         var arv2=parseInt(document.getElementById("arv2kast").value);
                 var vasakserv=20;
                 var korgus=30;
                 g.clearRect(0, 0, 300, 200);
         g.fillRect(vasakserv, 30, arv1, korgus);
         g.fillRect(vasakserv, 80, arv2, korgus);
      }
    </script>
  </head>
  <body>
    <h1>Joonis</h1>
    <canvas id="tahvel" width="300" height="200"  style="background-
color:yellow"></canvas><br />
        Palun kaks arvu:
    <input type="text" id="arv1kast"  value="10" />
    <input type="text" id="arv2kast"  value="20" />
    <input type="button" value="Joonista" onclick="joonista()" />
  </body>
</html>
```

Palun kaks arvu: 10 20 Joonista

Muudetava kõrgusega ristkülikud

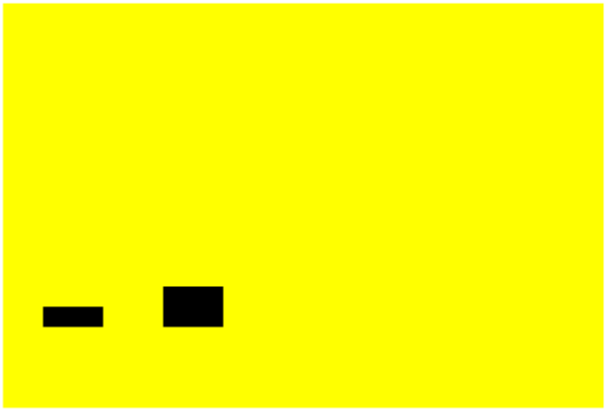
```
<!DOCTYPE html>
<html>
  <head>
    <title>Joonis</title>
    <script type="text/javascript">
      function joonista(){
         var g=document.getElementById("tahvel").getContext("2d");
         var arv1=parseInt(document.getElementById("arv1kast").value);
         var arv2=parseInt(document.getElementById("arv2kast").value);
                 var ylaserv=20;
                 var laius=30;
                 g.clearRect(0, 0, 300, 200);
         g.fillRect(20, ylaserv, laius, arv1);
         g.fillRect(80, ylaserv, laius, arv2);
      }
    </script>
  </head>
  <body>
    <h1>Joonis</h1>
    <canvas id="tahvel" width="300" height="200"  style="background-
color:yellow"></canvas><br />
        Palun kaks arvu:
    <input type="text" id="arv1kast"  value="10" />
    <input type="text" id="arv2kast"  value="20" />
    <input type="button" value="Joonista" onclick="joonista()" />
  </body>
</html>
```



Palun kaks arvu: 10 20 Joonista

Alt üles kasvavad ristkülikud

```
<!DOCTYPE html>
<html>
  <head>
    <title>Joonis</title>
    <script type="text/javascript">
      function joonista(){
         var g=document.getElementById("tahvel").getContext("2d");
         var arv1=parseInt(document.getElementById("arv1kast").value);
         var arv2=parseInt(document.getElementById("arv2kast").value);
                 var alaserv=160;
                 var laius=30;
                 g.clearRect(0, 0, 300, 200);
         g.fillRect(20, alaserv-arv1, laius, arv1);
         g.fillRect(80, alaserv-arv2, laius, arv2);
      }
    </script>
  </head>
  <body>
    <h1>Joonis</h1>
    <canvas id="tahvel" width="300" height="200"  style="background-
color:yellow"></canvas><br />
        Palun kaks arvu:
    <input type="text" id="arv1kast"  value="10" />
    <input type="text" id="arv2kast"  value="20" />
    <input type="button" value="Joonista" onclick="joonista()" />
  </body>
</html>
```

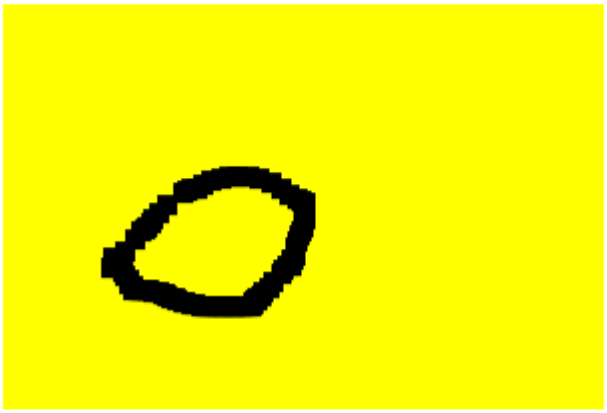


Palun kaks arvu: 10 20 Joonista

Hiir

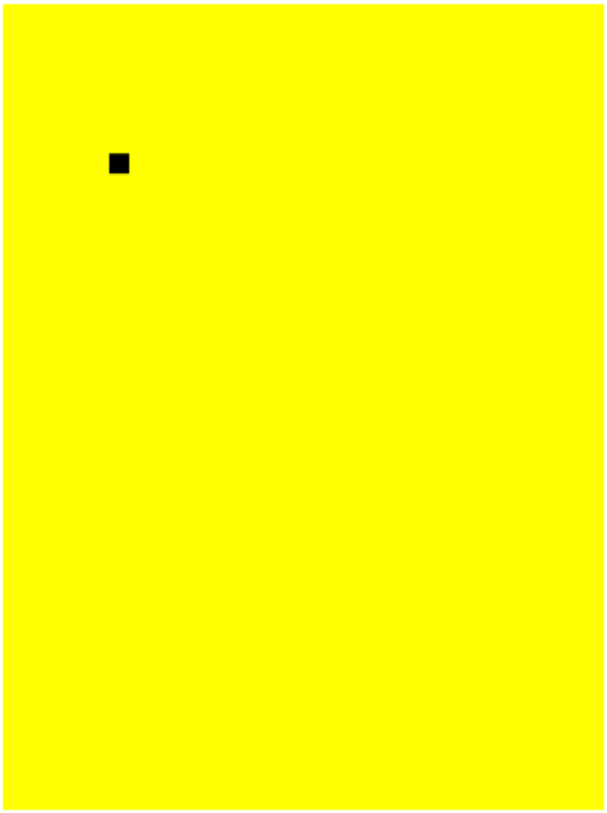
Hiire liikumise järgi ruutude joonistamine

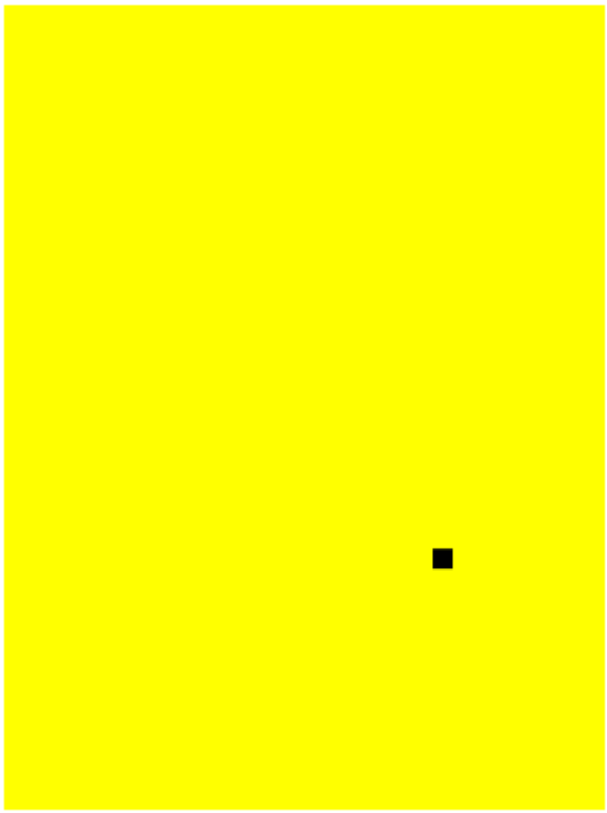
```
<!DOCTYPE html>
<html>
  <head>
    <title>Joonis</title>
    <script type="text/javascript">
      function liigub(e){
         var tahvlikoht=document.getElementById("tahvel").
                                          getBoundingClientRect();
         var g=document.getElementById("tahvel").getContext("2d");
         var hx=e.clientX-tahvlikoht.left;
         var hy=e.clientY-tahvlikoht.top;
         g.fillRect(hx, hy, 10, 10);
      }
    </script>
  </head>
  <body>
    <h1>Joonis</h1>
    <canvas id="tahvel" width="300" height="200"  style="background-
color:yellow" onmousemove="liigub(event)"></canvas>
  </body>
</html>
```



Hiirega kaasa liikuv ruut.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Joonis</title>
    <script type="text/javascript">
      function vajutus(e){
         var tahvlikoht=document.getElementById("tahvel").
                                          getBoundingClientRect();
         var g=document.getElementById("tahvel").getContext("2d");
         var hx=e.clientX-tahvlikoht.left;
         var hy=e.clientY-tahvlikoht.top;
                 g.clearRect(0, 0, tahvlikoht.right-tahvlikoht.left,
                                   tahvlikoht.bottom-tahvlikoht.top);
         g.fillRect(hx, hy, 10, 10);
      }
    </script>
  </head>
  <body>
    <h1>Joonis</h1>
    <canvas id="tahvel" width="300" height="400"  style="background-
color:yellow"
          onmousedown="vajutus(event)" onmousemove="vajutus(event)"></canvas>
  </body>
</html>
```





Ringi joonistamine tahvlile hiire järgi

Muu hulgas tasub märkida hiire asukoha arvutamise valemit. Võrreldes eelnevaga arvestatakse ka, et veebileht võib olla pikem ja keritud enne, kui tahvli peal midagi tegema hakatakse. Selleks puhukus küsitakse ligipääs dokumendi sisuosale (juur). Ning koordinaatide arvutamisel lahutatakse hiire koordinaatidest nii tahvli ülanurga koordinaadid kui ka lisaks lehe enese kerimisnihe.

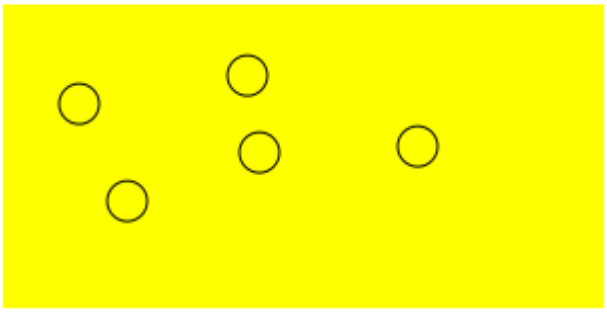
```
var juur=document.documentElement;
var hx=hiireAndmed.clientX-joonistusAla.left-juur.scrollLeft;
var hy=hiireAndmed.clientY-joonistusAla.top-juur.scrollTop;
```

Edasi kood tervikuna

```
<!DOCTYPE html>
<html>
  <head>
    <title>Joonis</title>
    <script type="text/javascript">
      var ringX=100, ringY=70, raadius=10
      function joonista(){
         var g=document.getElementById("tahvel").getContext("2d");
         g.beginPath();
         g.arc(ringX, ringY, raadius, 0, 2*Math.PI, true);
         g.stroke();
      }
      function hiirAlla(hiireAndmed){
         var joonistusAla=document.getElementById("tahvel").
```

```
                    getBoundingClientRect();
          var juur=document.documentElement;
          var hx=hiireAndmed.clientX-joonistusAla.left-juur.scrollLeft;
          var hy=hiireAndmed.clientY-joonistusAla.top-juur.scrollTop;
          ringX=hx;
          ringY=hy;
          joonista();
        }
        function algus(){
          document.getElementById("tahvel").
                    addEventListener("mousedown", hiirAlla, false);
        }
    </script>
  </head>
  <body onload="algus()">
    <h1>Joonis</h1>
    <canvas id="tahvel" style="width:400;height:300;background-
color:yellow"></canvas><br />
    Vajuta hiirega tahvlile!
  </body>
</html>
```

Joonis



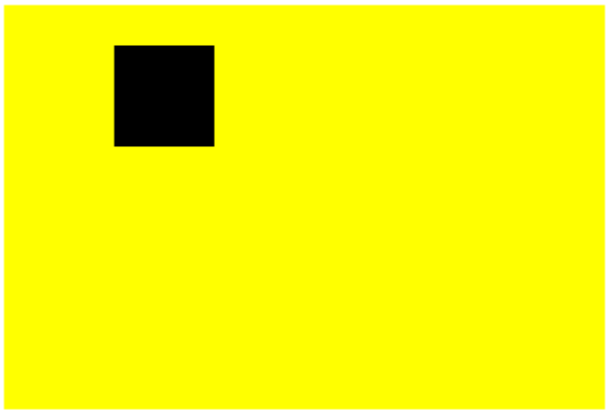
Vajuta hiirega tahvlile!

Liikumine

Nupuvajutusel samm paremale

```
<!doctype html>
<html>
  <head>
    <title>Liikumine</title>
        <script>
           var x=20, samm=5;
           function joonista(){
             var t=document.getElementById("tahvel");
                 var g=t.getContext("2d");
                 g.clearRect(0, 0, t.width, t.height);
                 g.fillRect(x, 20, 50, 50);
           }
           function paremale(){
```

```
            x=x+samm;
                joonista();
            }
        </script>
  </head>
  <body onload="joonista()">
      <canvas id="tahvel" width="300" height="200"
                style="background-color:yellow"></canvas><br />
      <input type="button" value="p" onclick="paremale()" />
  </body>
</html>
```

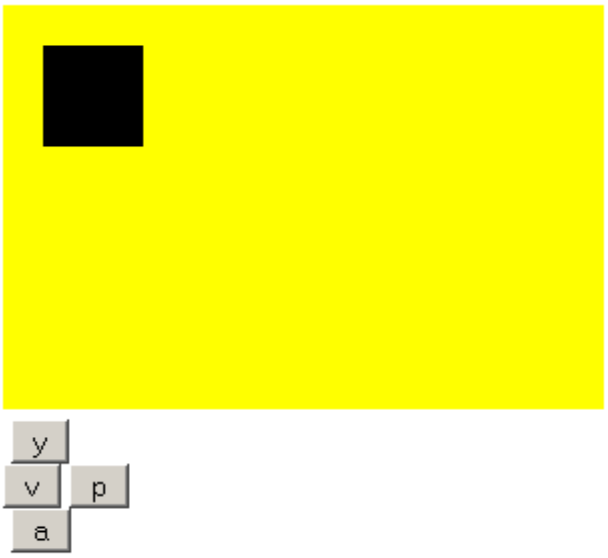


p

Nuppudega iga ilmakaare suunas

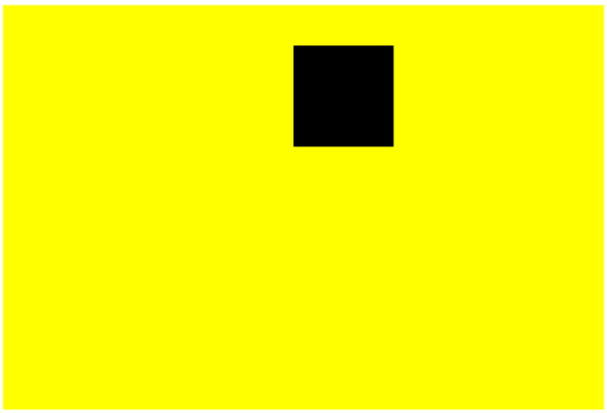
```
<!doctype html>
<html>
  <head>
    <title>Liikumine</title>
        <script>
           var x=20, y=20, samm=5;
           function joonista(){
             var t=document.getElementById("tahvel");
                 var g=t.getContext("2d");
                 g.clearRect(0, 0, t.width, t.height);
                 g.fillRect(x, y, 50, 50);
           }
           function paremale(){
             x=x+samm;
                 joonista();
           }
           function vasakule(){
             x=x-samm;
                 joonista();
           }
           function yles(){
             y=y-samm;
                 joonista();
           }
           function alla(){
             y=y+samm;
                 joonista();
           }
```

```
        </script>
  </head>
  <body onload="joonista()">
      <canvas id="tahvel" width="300" height="200"
                  style="background-color:yellow"></canvas><br />
       <input type="button" value="y" onclick="yles()" /><br />
      <input type="button" value="v" onclick="vasakule()" />
      <input type="button" value="p" onclick="paremale()" /><br />
       <input type="button" value="a" onclick="alla()" />
  </body>
</html>
```



Ise liikuvalt paremale

```
<!doctype html>
<html>
  <head>
    <title>Liikumine</title>
        <script>
           var x=20, samm=5;
           function joonista(){
             var t=document.getElementById("tahvel");
                 var g=t.getContext("2d");
                 g.clearRect(0, 0, t.width, t.height);
                 g.fillRect(x, 20, 50, 50);
           }
           function paremale(){
             x=x+samm;
                 joonista();
           }
        </script>
  </head>
  <body onload="setInterval('paremale()', 1000)">
      <canvas id="tahvel" width="300" height="200"
                  style="background-color:yellow"></canvas><br />
  </body>
</html>
```



Massiiv

Juhusliku seose kuvamine

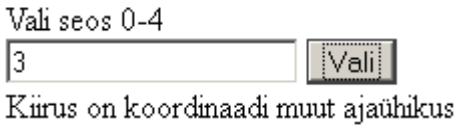
```
<!doctype html>
<html>
  <head>
     <title>Otsing</title>
         <meta charset="utf-8" />
         <script>
            var seosed=new Array(
                   "Koordinaatide ruutude summa võrdub hüpotenuusi ruuduga",
                   "Teepikkuse ja aja suhet nimetatakse kiiruseks",
                   "Kiirendus on kiiruse muutumise kiirus",
                   "Kiirus on koordinaadi muut ajaühikus",
                   "Kineetiline energia on võrdeline kiiruse ruuduga");
                function alusta(){
                   var juhuarv=Math.random(); //0 .. 0.99
                   var seosenr=parseInt(seosed.length*juhuarv);
                   document.getElementById("vastus").innerHTML=
                     seosed[seosenr];
                }
         </script>
  </head>
  <body onload="alusta()">
      <div id="vastus">

          </div>
  </body>
</html>
```

Väljund:

Kiirendus on kiiruse muutumise kiirus

Kasutaja valitud seos

```
<!doctype html>
<html>
  <head>
     <title>Otsing</title>
         <meta charset="utf-8" />
         <script>
            var seosed=new Array(
                   "Koordinaatide ruutude summa võrdub hüpotenuusi ruuduga",
                   "Teepikkuse ja aja suhet nimetatakse kiiruseks",
                   "Kiirendus on kiiruse muutumise kiirus",
                   "Kiirus on koordinaadi muut ajaühikus",
                   "Kineetiline energia on võrdeline kiiruse ruuduga");
                function alusta(){
                   document.getElementById("vastus").innerHTML=
                     "Vali seos 0-"+(seosed.length-1);
                }
                function otsi(){
                   var seosenr=parseInt(document.getElementById("kast1").value);
                   document.getElementById("vastus2").innerHTML=seosed[seosenr];
                }
         </script>
  </head>
  <body onload="alusta()">
      <div id="vastus"></div>
          <input type="text" id="kast1" value="0" />
          <input type="button" value="Vali" onclick="otsi()" />
          <div id="vastus2"></div>
  </body>
</html>
```



Juhusliku tantsupaari leidmine

```
<!doctype html>
<html>
  <head>
     <title>Otsing</title>
         <meta charset="utf-8" />
         <script>
            var poisid=new Array("Juku", "Madis", "Sass", "Mait", "Mart");
            var tydrukud=new Array("Kati", "Katrin", "Mari", "Minna");
                function alusta(){
                   var poisinr=parseInt(poisid.length*Math.random());
                   var tydrukunr=parseInt(tydrukud.length*Math.random());
                   document.getElementById("vastus").innerHTML=
```

```
                    "Tantsivad "+poisid[poisinr]+" ja "+tydrukud[tydrukunr];
                }
        </script>
  </head>
  <body onload="alusta()">
      <div id="vastus">

      </div>
  </body>
</html>
```

Väljund:

Tantsivad Madis ja Minna

Arvude ruudud ühest kahekümneni

```
<!doctype html>
<html>
  <head>
     <title>Otsing</title>
         <meta charset="utf-8" />
         <script>
                function algus(){
                 for(var i=1; i<=20; i++){
                   document.getElementById("vastus").innerHTML+=(i*i)+" ";
                 }
                }
         </script>
  </head>
  <body onload="algus()">
      <div id="vastus">

      </div>
  </body>
</html>
```

Väljund:

1 4 9 16 25 36 49 64 81 100 121 144 169 196 225 256 289 324 361 400

Kasutaja soovitud arv ruute

```
<!doctype html>
<html>
  <head>
     <title>Otsing</title>
         <meta charset="utf-8" />
         <script>
                function kirjutaRuudud(){
                 var kogus=parseInt(document.getElementById("kast1").value);
                                document.getElementById("vastus").innerHTML="";
                 for(var i=1; i<=kogus; i++){
```

```
                  document.getElementById("vastus").innerHTML+=(i*i)+" ";
                 }
                }
        </script>
  </head>
  <body>
      Ruutude arv:
          <input type="text" id="kast1" />
          <input type="button" value="Kirjuta" onclick="kirjutaRuudud()" />
      <div id="vastus">

      </div>
  </body>
</html>
```

Ruutude arv: 7 Kirjuta

1 4 9 16 25 36 49

Eesnimi kaks korda kõrvuti

```
<!doctype html>
<html>
  <head>
     <title>Otsing</title>
         <meta charset="utf-8" />
         <script>
                var eesnimed=new Array("Juku", "Kati", "Katrin", "Madis");
                function algus(){
                 var t="<ul>";
                 for(var i=0; i<eesnimed.length; i++){
                   t+="<li>"+eesnimed[i]+" "+eesnimed[i]+"</li>";
                 }
                 t+="</ul>";
                 document.getElementById("vastus").innerHTML=t;
                }
         </script>
  </head>
  <body onload="algus()">
      <div id="vastus">

      </div>
  </body>
</html>
```

- Juku Juku
- Kati Kati
- Katrin Katrin
- Madis Madis

Tagurpidi loetelu

```
<!doctype html>
<html>
  <head>
     <title>Otsing</title>
         <meta charset="utf-8" />
         <script>
                var eesnimed=new Array("Juku", "Kati", "Katrin", "Madis");
                function algus(){
                 var t="<ul>";
                 for(var i=eesnimed.length-1; i>=0; i--){
                   t+="<li>"+eesnimed[i]+"</li>";
                 }
                 t+="</ul>";
                 document.getElementById("vastus").innerHTML=t;
                }
         </script>
  </head>
  <body onload="algus()">
      <div id="vastus">

      </div>
  </body>
</html>
```

- Madis
- Katrin
- Kati
- Juku

Nummerdatud loetelu

```
<!doctype html>
<html>
  <head>
     <title>Otsing</title>
         <meta charset="utf-8" />
         <script>
                var eesnimed=new Array("Juku", "Kati", "Katrin", "Madis");
                function algus(){
                 var t="<ol>";
                 for(var i=0; i<eesnimed.length; i++){
                   t+="<li>"+eesnimed[i]+"</li>";
                 }
                 t+="</ol>";
                 document.getElementById("vastus").innerHTML=t;
                }
         </script>
  </head>
  <body onload="algus()">
      <div id="vastus">
```

```
      </div>
  </body>
</html>
```

1. Juku
2. Kati
3. Katrin
4. Madis

Sõna alguse otsing. Väiketähtedega tekst

```
<!doctype html>
<html>
  <head>
     <title>Otsing</title>
         <meta charset="utf-8" />
         <script>
                var eesnimed=new Array("Juku", "Kati", "Katrin", "Madis");
                function otsi(){
                 var otsitav=
                    document.getElementById("kast2").value.toLowerCase();
                 var t="<ul>";
                 for(var i=0; i<eesnimed.length; i++){
                   if(eesnimed[i].toLowerCase().indexOf(otsitav)==0){
                     t+="<li>"+eesnimed[i].toLowerCase()+"</li>";
                   }
                 }
                 t+="</ul>";
                 document.getElementById("vastus").innerHTML=t;
                }
         </script>
  </head>
  <body onload="otsi()">
        Otsitav tekst:
      <input type="text" id="kast2" onkeyup="otsi()" />
      <div id="vastus"></div>
  </body>
</html>
```

Otsitav tekst: k

- kati
- katrin

Elementide vahetatud järjekord

```
<!doctype html>
<html>
  <head>
     <title>Otsing</title>
         <meta charset="utf-8" />
         <script>
                var lapsed=[
                   {"eesnimi":"Juku", "synniaeg": "13.06"},
                   {"eesnimi":"Kati", "synniaeg": "11.06"},
                   {"eesnimi":"Mati", "synniaeg": "25.08"},
                   {"eesnimi":"Madis","synniaeg": "09.02"}
                ];
                function otsi(){
                 var otsitav=
                      document.getElementById("kast2").value.toLowerCase();
                 var t="<ul>";
                 for(var i=0; i<lapsed.length; i++){
                   if(lapsed[i].eesnimi.toLowerCase().indexOf(otsitav)!=-1){
                     t+="<li>"+lapsed[i].synniaeg+ " "+
                           lapsed[i].eesnimi+"</li>";
                   }
                 }
                 t+="</ul>";
                 document.getElementById("vastus").innerHTML=t;
                }
         </script>
  </head>
  <body onload="otsi()">
        Otsitav tekst:
      <input type="text" id="kast2" onkeyup="otsi()" />
      <div id="vastus"></div>
  </body>
</html>
```

Otsitav tekst: k

- 13.06 Juku
- 11.06 Kati