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

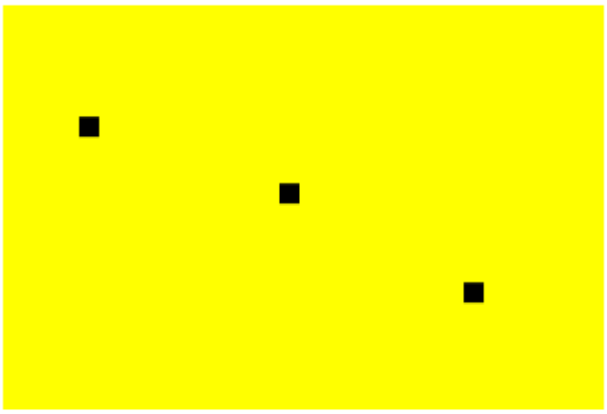
Järgmise näitena püütakse saada hiire vajutuse kohale ristkülik. Selleks on vaja teada hiire koordinaate tahvli suhtes - kättesaadav clientX/clientY annab need aga akna suhtes. Päästjaks on tahvli asukohta küsiv funktsioon getBoundingClientRect(). Suurema lehe puhul lisanduks siia veel arvutus lehe enese 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:

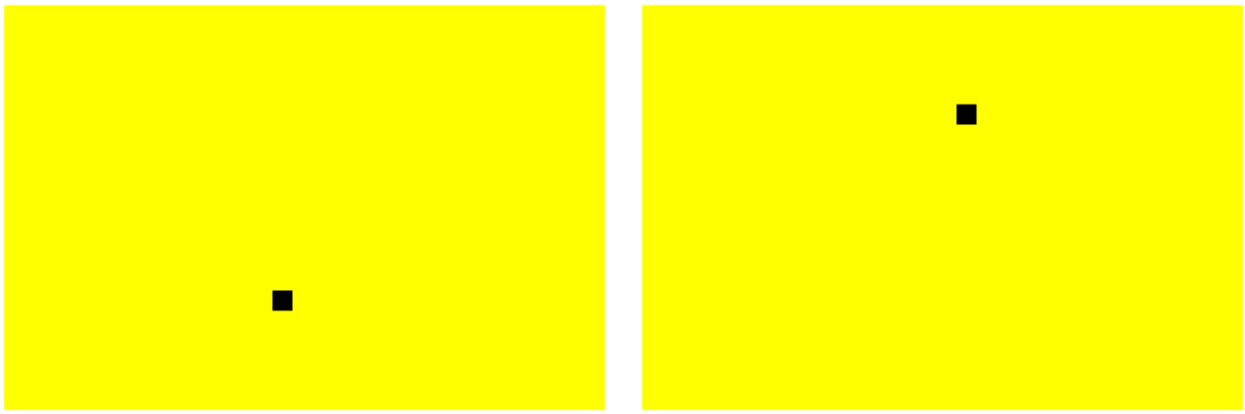
4

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.

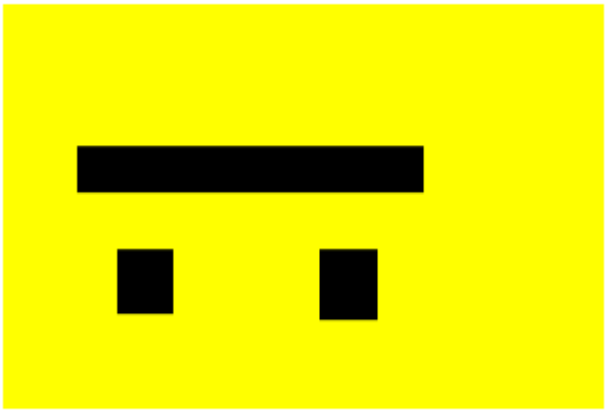
```
<!DOCTYPE html>
<html>
  <head>
```

```
    <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.