Andmehaldus

Eesnimede loetelu

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

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

Nimeharjutused

- Juku
- Kati
- Ants

Funktsiooni lisaargumendid

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

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

Nimeharjutused

- Ants
- Juku
- Sass

Punktide haldus

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

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

Nimeharjutused

Kasutaja	Punktid
Juku	10
Sass	8

Sortimine

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

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

Edasi juba kood tervikuna.

```
<!doctype html>
<html>
  <head>
     <title>Haldus</title>
         <script>
             function Tulemushaldus(kihinimi){
                   this.algus=function(){
                      this.kiht=document.getElementById(kihinimi);
                          this.andmed=new Array();
                   }
                   this.kuva=function(){
                      var t="<table><tr><th>Kasutaja</th><th>Punktid</th></tr>";
                          for(var i=0; i<this.andmed.length; i++){
                             t+="<tr><td>"+this.andmed[i].kasutajanimi+
                                   "</td><td>"+this.andmed[i].punkte+"</td></tr>
";
                          }
                          t+="</table>";
                          this.kiht.innerHTML=t;
                   }
                   this.lisaTulemus=function(kasutajanimi, punkte){
                      this.andmed.push({"kasutajanimi": kasutajanimi, "punkte":
parseInt(punkte)});
                          this.andmed.sort(function(k1, k2){return k1.punkte-
k2.punkte});
                          this.kuva();
                   }
                   this.algus();
                }
                var h1;
                function lehealgus(){
                   h1=new Tulemushaldus("kiht1");
```

```
                        h1.lisaTulemus("Juku", 10);
                        h1.lisaTulemus("Sass", 8);
                }
        </script>
  </head>
  <body onload="lehealgus();">
     <h1>Nimeharjutused</h1>
         <div id="kiht1"></div>
  </body>
</html>
```

Nimeharjutused

Kasutaja	Punktid
Sass	8
Juku	10

Ülesandeid

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

Salvestamine

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

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

Andmete tekstiks ja tagasi muutmiseks sai tehtud kaks abistavat funktsiooni:

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

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

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

Ning edasi juba kood tervikuna

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

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

Ülesandeid

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

Veebiühenduseta rakendus

Veebilehel töötavat rakendust on mugav brauseris ja veebilehelt avada, kuid sugugi pidevalt ei pruugi ühendus olemas olla. Selle mure leevendamiseks on loodud appcache, mis kord avatud lehe andmed brauserisse meelde jätab, nii et lehte saab kasutada ka ühenduse puudumisel. Koos localStorage-ga moodustavad nad hea komplekti, millega teha täiesti töölauarakendustega võrreldavaid lahendusi. Selliseks hoidmiseks tuleb apache-serveri puhul luua eraldi kirjeldusfail, kus CACHE MANIFEST-sektsioonis on failide loetelu, mis palutakse veebilehitsejal meelde jätta. Soovitatavalt on failis ka trellidega välja kommenteeritud real faili muutmisaeg - lahendus töötab nõnda, et html-fail laetakse uuesti alles siis, kui manifesti faili on muudetud.

punktid.appcache

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

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

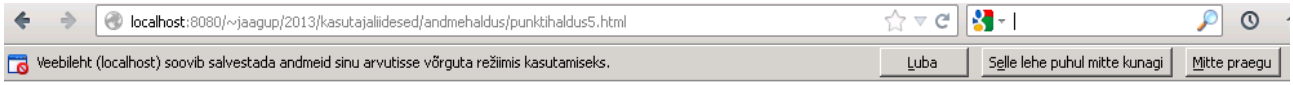
.htaccess

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

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

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

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



Väike läbimäng ka

Avatud leht

Nimeharjutused!

Kasutaja Punktid

Sass 8

Ants 9

Juku 10

kasutajanimi | punktid | Lisa

Salvesta | Loe

Andmete lisamine

Kasutaja Punktid

Sass 8

Ants 9

Juku 10

Mari | 9 | Lisa

Salvesta | Loe

Salvestusnupule vajutus

Kasutaja Punktid

Sass 8

Ants 9

Mari 9

Juku 10

kasutajanimi punktid Lisa

Salvesta Loe

salvesta

Uue kasutaja lisamine

Kasutaja Punktid

Sass 8

Ants 9

Mari 9

Juku 10

Ants 11 Lisa

Salvesta Loe

Kasutaja olemas

Nimeharjutused!

Kasutaja Punktid

Sass 8

Ants 9

Mari 9

Juku 10

Ants 11

kasutajanimi punktid Lisa

Salvesta Loe

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

Ülesandeid

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

XMLHttpRequest

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

teade.txt

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

Kohene päring

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

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

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

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

Ülesandeid

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

Asünkroonne päring

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

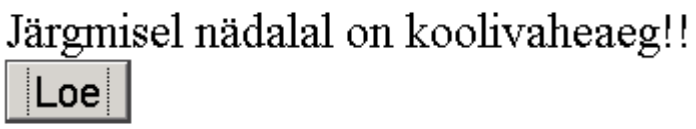
```
<!doctype html>
<html>
  <head>
    <title>Veebiühendus</title>
    <script>
       var xhr=new XMLHttpRequest();
       xhr.onreadystatechange=andmedSaabusid;

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

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

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

Tulemus:



Tervitamine serverist

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

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

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

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

tervitus.php

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

Post-meetodiga saatmine

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

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

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

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

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

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

Mitu parameetrit päringus

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

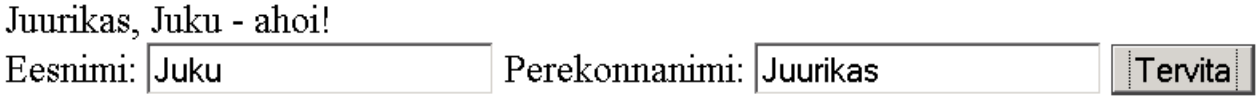
```
<!doctype html>
<html>
  <head>
    <title>Veebiühendus</title>
    <script>
       var xhr=new XMLHttpRequest();
       xhr.onreadystatechange=andmedSaabusid;

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

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

tervitus2.php

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



Ülesandeid

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

JSON

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

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

Väljund:

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

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

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

Väljund

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

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

Väljund Javaskripti

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

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

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

näitekood tervikuna

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

Tulemuseks javaskripti ja HTMLi abil kujundatud tabel

Kasutajanimi	Tulemus
juku	7
kati	8

Ülesandeid

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

Andmed SQL-tabelis

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

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

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

Mõned andmed ka sisse.

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

PHP-fail andmete küsimiseks andmetabelist ning väljastmiseks JSONina

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

  echo json_encode(kysiKasutajad());
```

Tulemus:

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

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

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

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

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

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

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

Nõnda saab pärast andmete uuenduskäsklust serveris olevat uut tulemust vaadata.

Ülesandeid

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

Ajaxi abil lugemine

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

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

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

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

Ülesandeid

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

AJAX ka salvestamiseks

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

edetabel4.php

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

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

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

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

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

kuvamine4.html

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

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

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

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

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

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

Ülesandeid

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

Üldisi ülesandeid

Kordamisküsimused

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

Kolme tasemega ülesanded

Pallide tüübid

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

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

Hulknurgad

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

Jalgratas

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

Lift

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

Mängukaardid

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

Automaatsalvestus veebilehel

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

Kolmnurga joonistamine

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

Graafika salvestus

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

Tähemärkidega kujundus

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

Asukohtadega jututuba

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

Kujundid

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

Hammasrattad

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

Valdade valimine

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

Tähtede püüdmine

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