

Webbprogrammering

Rapport

921125-0619

A11gusli

Gustaf Linnarsson

Webug

Inledning

I den här rapporten kommer lösningsförslagen att diskuteras och varför dom är gjorda osv.

Domänen som är skapad är hemsidan för ett taxiföretag där kunden skall kunna registrera sig och sedan boka bilar för ett visst datum. Systemet fungerar så att för att kunna boka måste man först logga in, om man inte är medlem redan så måste man därför registrera sig på hemsidan, detta sker igenom ett formulär som finns på registrerings sidan. När man har registrerat sig på sidan så kan man därefter ta sig vidare till bokningssidan. Där skall man välja ett datum och sedan kan man boka en bil. När man bokar en bil så läggs bokningen i kundvagnen. Det finns även en sökfunktion som fungerar så att man kan söka på företagsnamn och se vilken data som resursen har. Efter det så kan användaren logga ut.

Hemsidan består utav 8 olika sidor. Dessa är: Start, Om, Bokning, Medlem, Login, Registrera, Kontakt och Sök.

Start sidan – Index.html

Tanken med start sidan är bara att det skall vara en form av välkomst sida där det står ett meddelande från påstådda VD. Här finns även ett HTML 5 canvas element med en rektangel som rör sig. Då kravet är så här: *”Det skall finnas ett eller flera element som använder HTML5 för rörlig grafik med hjälp av Canvas elementet (handskriven HTML5 kod utan toolkits). Vektorgrafik och HTML5 Bitmaps skall finnas i någon form tillsammans med Transformationer (Exempel är interaktiva kontroller eller rörlig reklam i någon form). Man kan om man vill använda HTML5 video, men detta är inte nödvändigt.”*

Med efter en hel del problem så fick det bli en enkel lösning med en rektangel som åker från ena sidan till den andra. Då tiden var knapp så kunde därför inte en bättre lösning göras. Men en plan för framtiden är att göra ett animerat canvas element som har något mer med domänen att göra. T.ex. en taxibil som åker, eller varför inte en bild på staden Göteborg som företaget befinner sig i.



Så här det ur när man anländer till sidan. Här finns ett litet välkomstmeddelande och då canvas elementet. Den börjar röra på sig när man kommer in på sidan. För tillfället så åker

den bara från ena sidan till den andra men planen är att jobba vidare på sidan och då även få den att åka tillbaka också. Men det mest troliga blir att den byts ut helt mot något mer relevant för sidan. Nedanför kommer scriptet för canvas elementet.

```
window.requestAnimFrame = (function(callback) {  
    return window.requestAnimationFrame ||  
    window.webkitRequestAnimationFrame ||  
    window.mozRequestAnimationFrame ||  
    window.oRequestAnimationFrame ||  
    window.msRequestAnimationFrame ||  
  
    function(callback) {  
        window.setTimeout(callback, 1000 / 60);  
    };  
}) ();
```

I funktionen nedanför så ritas rektangeln ut. Här nedanför väljs t.ex. färg osv.

```
function drawRectangle(myRectangle, context) {  
    context.beginPath();  
    context.rect(myRectangle.x, myRectangle.y,  
myRectangle.width, myRectangle.height);  
    context.fillStyle = '#003366';  
    context.fill();  
    context.lineWidth = myRectangle.borderWidth;  
    context.strokeStyle = 'black';  
    context.stroke();  
}  
  
function animate(myRectangle, canvas, context,  
startTime) {  
    // update  
    var time = (new Date()).getTime() - startTime;  
  
    var linearSpeed = 100;  
    // pixels / second  
    var newX = linearSpeed * time / 1000;
```

```

        if(newX < canvas.width - myRectangle.width -
myRectangle.borderWidth / 2) {
            myRectangle.x = newX;
        }
        // clear
context.clearRect(0, 0, canvas.width, canvas.height);
        drawRectangle(myRectangle, context);
        // request new frame
        requestAnimationFrame(function() {
            animate(myRectangle, canvas, context, startTime);
        });
    }
    var canvas = document.getElementById('myCanvas');
    var context = canvas.getContext('2d');

```

Ger rektangeln x och y koordinater samt en höjd och bredd.

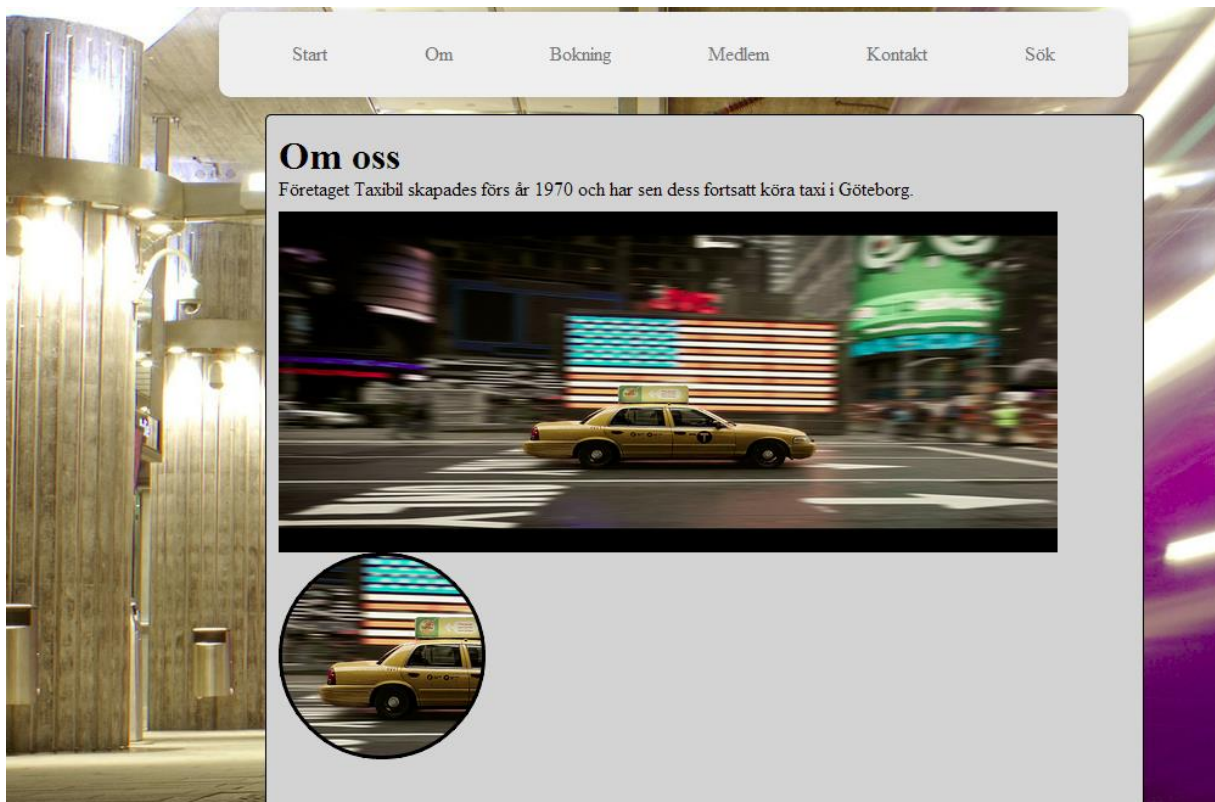
```

var myRectangle = {
    x: 0,
    y: 75,
    width: 100,
    height: 50,
    borderWidth: 5
};
drawRectangle(myRectangle, context);
// wait one second before starting animation
setTimeout(function() {
    var startTime = (new Date()).getTime();
    animate(myRectangle, canvas, context, startTime);
}, 1000);

```

Om sidan-om-html

På om sidan så kan man hitta lite historia om företaget och även en bild med HTML canvas clipping. För tillfället finns inte så mycket information men tanken är att det skall utökas.



Så här ser det ut när man kommer till "om"sidan. Tanken med canvas clipping var att det skulle vara en bild på en taxibil så att kunderna skall kunna se vad det är för typ utav bilar som företaget använder. För att göra sidan lite roligare också så kombinerades bilden med canvas clipping så att man som kund kan zooma in och få en lite bättre titt på bilen.

Dock så existerar det ett problem och det är att inzoomningen inte fungerar helt korrekt. När man rör musen över bilden så skall den zooma in på det område som musen befinner sig. Av någon anledning nu så hamnar detta område nedanför själva bilden. Det fungerar att zooma men positionen på den är helt fel. Detta är något som kommer att fixas till i framtiden. Den finns kvar nu för att den gör sidan lite roligare även att den inte riktigt fungerar som den skall göra.

Här nedanför så ligger script koden för clippningen:

```

var canvas,
    context,
    taxiImage,
    magnification,

    scaleOfSourceImage=4; // Original image is 4 times as large
(2560x1920px) as the displayed size (640x480px)

```

Här skapas canvaselementet och förstoringsglaset samt här laddas bilden på taxin in.

```

window.onload=function(){
    canvas=document.getElementById("a");
    context=canvas.getContext("2d");
    magnification=170;
    taxiImage=document.getElementById("taxiImage");
    taxiImage.addEventListener("mousemove",drawMagnification,false);
    taxiImage.addEventListener("mousedown",setMagnification,false);
    taxiImage.addEventListener('contextmenu',
function(event){event.preventDefault();}, false); //Prevent the context
menu to be displayed when right mouse button is clicked
}

//Sets level of magnification
function setMagnification(event){
    //Check which mouse button is pressed down
    if(event.which==3) //Right mouse button
        magnification+=10;
    else //Left or middle mouse button
        magnification-=10;
    if(magnification>500) magnification=500;
    if(magnification<10) magnification=10;
    drawMagnification(event);
}

```

Ritar ut förstoringsglaset.

```

function drawMagnification(event){
    //Get mouse position
    var positionOfTheGardenImageElement=findPos(event.target);
//event.target is the image
    var x=event.pageX-positionOfTheGardenImageElement.x;
    var y=event.pageY-positionOfTheGardenImageElement.y;

```

```

//Clip the canvas
context.beginPath();

context.arc(canvas.width*0.5, canvas.height*0.5, canvas.width*0.5, 0,
Math.PI*2, false);

context.closePath();

context.fill();

context.clip();


//Determine source x and y positions
var sx=x*scaleOfSourceImage-magnification*0.5;
var sy=y*scaleOfSourceImage-magnification*0.5;
if(sx<0) sx=0;
if(sy<0) sy=0;


//Draw magnification of image in canvas

context.drawImage(taxiImage,sx,sy,magnification,magnification,0,0,canvas.wi
dth,canvas.height);


//Draw circle around magnification
context.lineWidth=4;
context.beginPath();

context.arc(canvas.width*0.5, canvas.height*0.5, canvas.width*0.5-1, 0,
Math.PI*2, false);

context.closePath();

context.stroke();

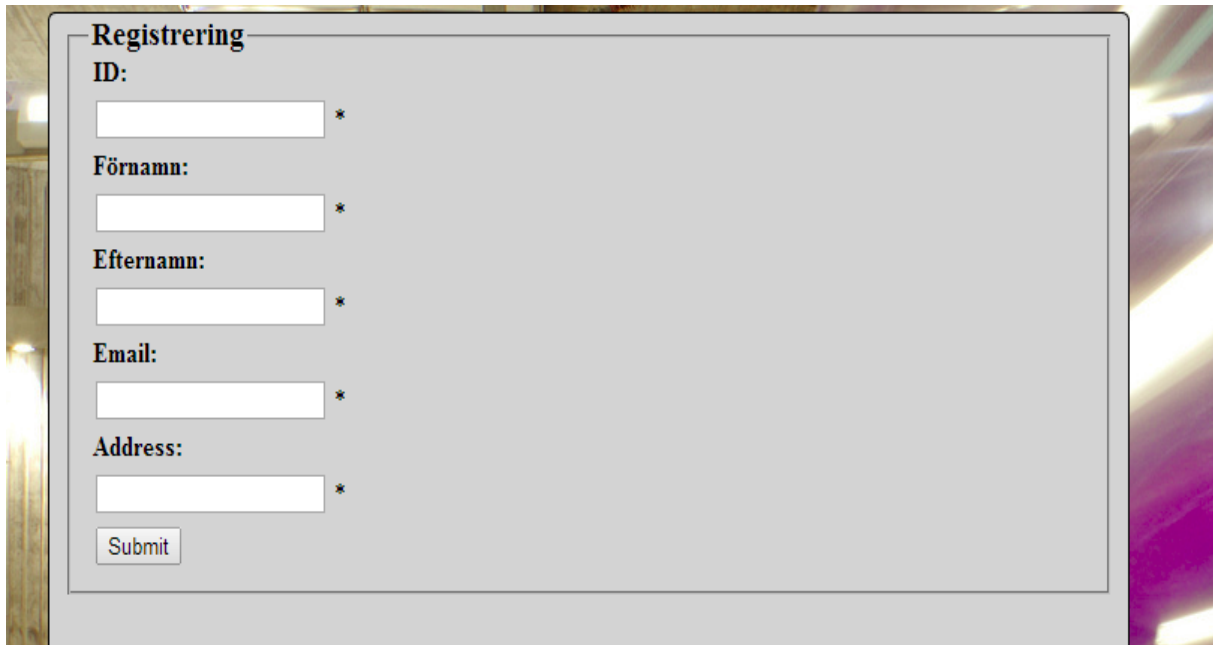

//Place the canvas element centered on the mouse position
var canvasx=event.pageX-canvas.width*0.5;
var canvasy=event.pageY-canvas.height*0.5;

canvas.style.left=canvasx+"px";
canvas.style.top=canvasy+"px";
}

```


Registrera – Registrera.html

För att kunna boka så måste man vara inloggad. För att kunna logga in så krävs det att man är registrerad. För att göra detta så går man in på registrera sidan som ligger under Medlem i navigeringsmenyn.

A screenshot of a web form titled "Registrering". The form is set against a light gray background and contains several input fields, each followed by an asterisk (*) indicating it is required. The fields are labeled "ID:", "Förnamn:", "Efternamn:", "Email:", and "Address:". At the bottom left of the form is a "Submit" button. The form is displayed within a window that has a blurred background showing what appears to be a hallway with lights.

Det här är synen som kunden möter när denne klicka sig in på registrera sidan. Det är ett vanligt formulär som man fyller i så skickas datan sedan in till databasen. Formuläret har validering så om ett av fälten inte är korrekt ifyllda så kommer ett meddelande komma upp som säger vad som är fel. Det kommer inte gå att skicka iväg datan om inte alla fält är ifyllda på rätt sätt.

Två utav fälten har även en extra validerings funktion och dessa är Förnamn och Efternamn. De kollar också längden på det man skriver in. Om man har skrivit in mer mindre än två bokstäver och mer än 16 bokstäver så kommer fältet att lysa rött.

Start Om Bokning Medlem Kontakt Sök

Registrering

ID: 34355 *

Förnamn: G *

Efternamn: Kungen *

Email: Gurrakungen.se *

Address: *

Submit

Detta ser ut så här. Detta används för att det är ett smidigt och snyggt sätt att säga till användaren att man har skrivit inom rätt intervall eller att man har skrivit fel. T.ex. Om användaren skrivit ett för kort namn så vill denne få reda på det direkt så att det går att ändra istället för att få reda på det när hela formuläret skall skickas. I vissa fall så får då användaren skriva om en del saker i formuläret och detta skapar irritation. Att behöva fylla i vissa fält igen bara för ett annat krånglar är slöseri med tid och tid är värdefullt. Får då användaren en indikation medan fältet fylls i att det är något som inte stämmer så får denne en chans att korrigera detta direkt.

Funktionen togs med för att det var enkelt att implementera, en liknande lösning men med ett annat designval hade kunnat vara att man hade en bock och ett kryss i slutet av varje ruta som beroende på om man fyllt i rätt eller inte visade antingen godkänt eller underkänt.

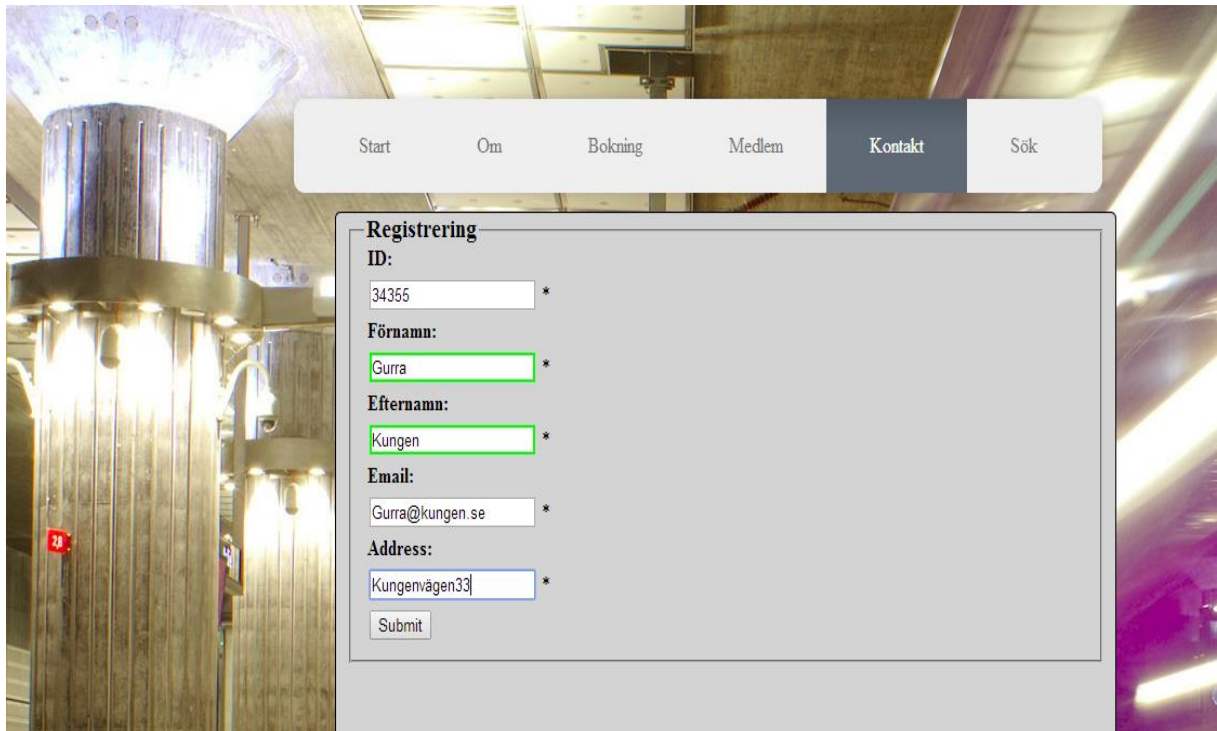
Koden för funktionen ser ut så här:

```
function checkLength(textInput, lowerLimit, upperLimit){
if(textInput.value.length>=lowerLimit &&textInput.value.length<=upperLimit)
textInput.className="isValid";

    else textInput.className="isNotValid";

}
```

Den undersöker helt enkelt om längden på texten är korrekt eller inkorrekt. Om den är det så är den valid och då lyser det grönt, om den inte är valid så kommer den lysa rött.



Start Om Bokning Medlem **Kontakt** Sök

Registrering

ID: 34355 *

Förnamn: Gurra *

Efternamn: Kungen *

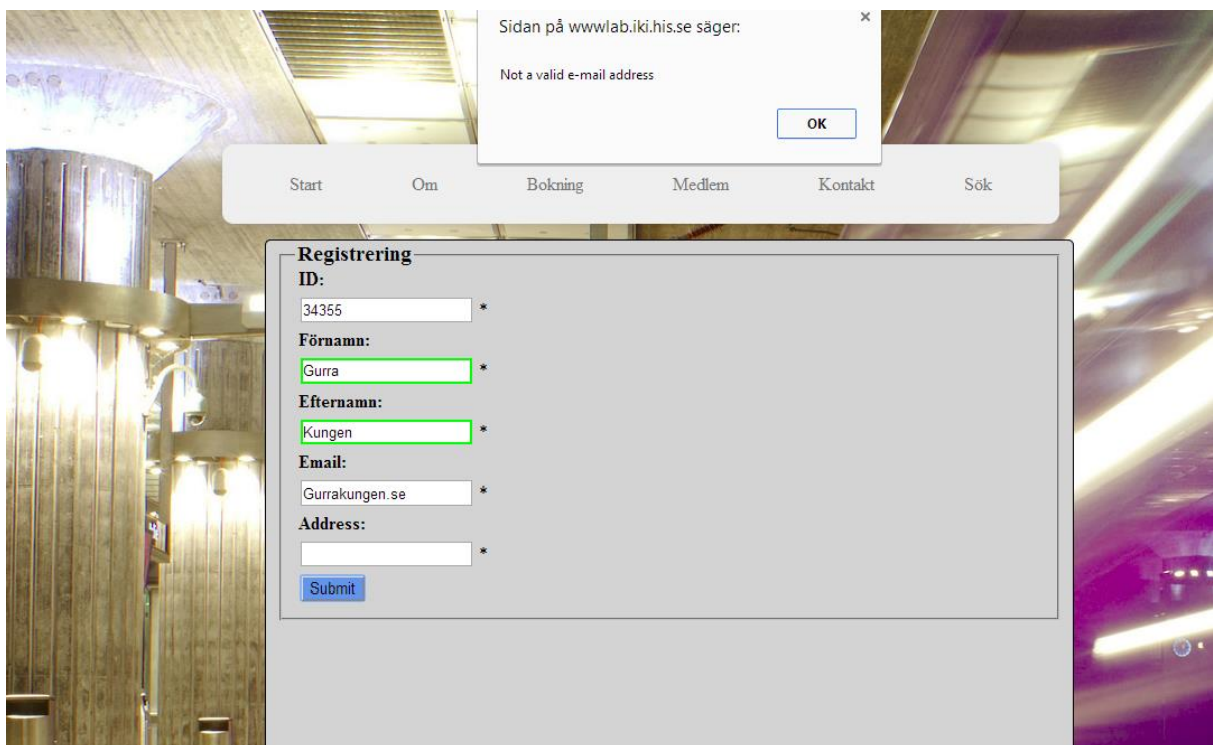
Email: Gurra@kungen.se *

Address: Kungenvägen33 *

Submit

Så här det ser ut när formuläret är korrekt i fyllt och när man trycker på skicka så kommer kunden att läggas in i databasen.

Om ett fält inte skulle vara ifyllt på rätt sätt och man trycker på skicka så kommer ett meddelande att visas som säger vilket fält som det är fel på. Detta ser ut så här:



Sidan på wwwlab.iki.his.se säger:

Not a valid e-mail address

OK

Start Om Bokning Medlem **Kontakt** Sök

Registrering

ID: 34355 *

Förnamn: Gurra *

Efternamn: Kungen *

Email: Gurrakungen.se *

Address: *

Submit

Här är det emailadressen som inte är ifyllt korrekt. Ett liknande meddelande kommer att visas för de andra fälten också.

Hela valideringen är en funktion som heter validateform och ser i princip ser likadan ut för alla fälten. Det fält som skiljer sig är då email fältet. För här kontrolleras om det finns ett snabel a med och om det finns en punkt med. Om inte dessa finns med så är det inte en fungerande emailaddress.

```
var x=document.getElementById("email").value;
    var atpos=x.indexOf("@");
    var dotpos=x.lastIndexOf(".");
    if (atpos<1 || dotpos<atpos+2 || dotpos+2>=x.length)
    {
        alert("Not a valid e-mail address");
        return false;
    }
```

Så här ser koden ut för de andra fälten: Koden ser likadan ut för de andra fälten bara att det är andra värden som tas upp då. Denna funktion kollar om fälten är tomma eller inte.

```
var x = document.getElementById("customerID").value;
    if (x == null || x == "") {
        alert("Ett KundID måste fyllas i");
        return false;
```

När formuläret är korrekt ifyllt och man trycker på skicka så körs två funktioner den ena är validerings funktionen som nämns ovanför och den andra är funktion storeCustomer. Denna skickar in datan i databasen. Koden för den ser ut så här:

```
var customerID=document.getElementById("customerID").value;
var firstname=document.getElementById("firstname").value;
var lastname=document.getElementById("lastname").value;
var email=document.getElementById("email").value;
var address=document.getElementById("address").value;

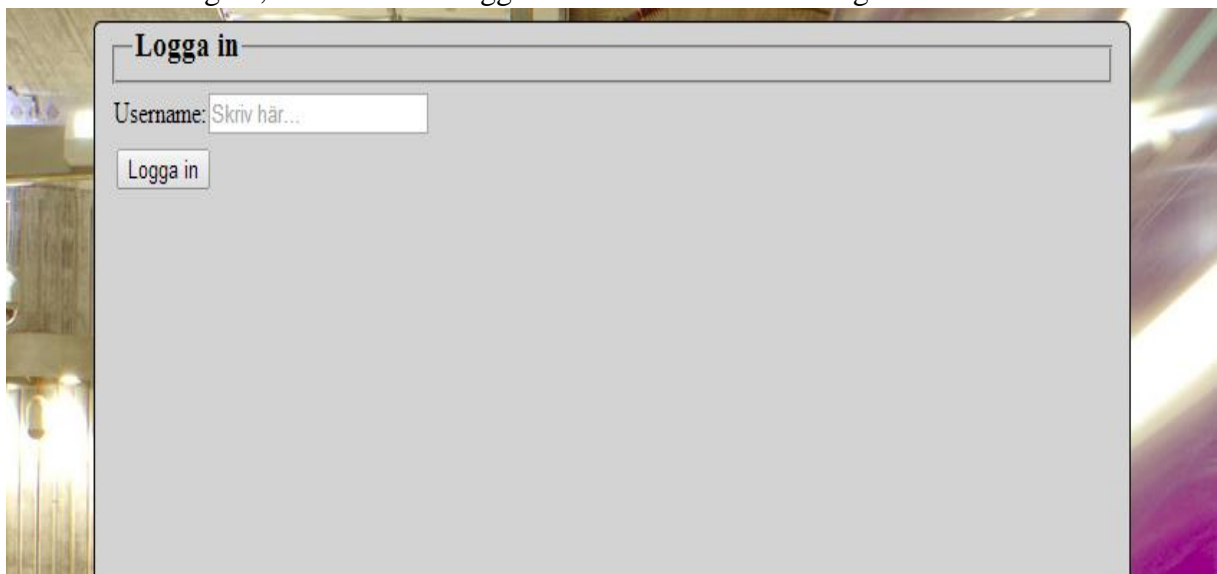
$.ajax({
    type: 'POST',
    url: 'booking/makecustomer_XML.php',
```

```
data: { ID: escape(customerID),  
  firstname: escape(firstname),  
  lastname: escape(lastname),  
  email: escape(email),  
  address: escape(address),  
},  
success: ResultCustomer  
});  
}
```

När datan har blivit korrekt inmatad i databasen så finns det en till funktion som heter ResultCustomer som kommer returnera ett meddelande som säger att en kund har blivit skapad.

Logga in –Login.html

Som nämnt tidigare, så måste man logga in för att kunna boka. Login sidan ser ut så här.



Här skriver man in sitt användarnamn som i databasen heter customerID. Funktionen heter getLoginUser, Vad den gör är att den går in och kollar om customerID finns och om den gör det så hämtar den det från databasen.

```

function getLoginUser(){
if (!!$("#login").val()){
    customerID = $("#login").val();
}else{
    return false;
}

$.ajax({
    type: 'POST',
    url: 'booking/getcustomer_XML.php',
    data: { customerID: escape(customerID) },
    success: ResultCustomerID
});
}

```

Sen jämförs användarnamnet med dom som finns i databasen och om det matchar och man hittar en användare så är man inloggad och man blir skickad till medlemsidan. Hittar systemet inte namnet så kommer ett meddelande att man inte kommer in och man får försöka igen tills man har rätt namn.

```

function ResultCustomerID(returnedData)
{
    var resultset=returnedData.childNodes[0];
    for (i = 0; i < resultset.childNodes.length; i++) {
        if(resultset.childNodes.item(i).nodeName=="customer"){
            var user=resultset.childNodes.item(i);
            username = user.attributes['id'].nodeValue;
            var userfound = true;
        }
    }

    // No user was found, display error.
    if(!userfound) {
        alert("kom ej in");
    }

    // User was found, store session

```

```
else {  
    sessionStorage.setItem("username", username);  
  
    window.location.href = "Medlem.html";  
}  
}
```

När man har skrivit in rätt användarnamn så blir man skickad till medlemssidan direkt. Av den anledningen att det är smidigare att bli direkt skickad istället för att man skall behöva klicka själv. Sen är det logiskt att så fort man har loggat in så skall man komma in till "sin" sida istället för att man skall behöva klicka sig in på den.

[Medlem – Medlem.html](#)

På medlemssidan så finns inte så mycket mer än en knapp för logga ut och ett välkomstmeddelande där det står välkommen och namnet på användaren som har loggat in. Namnet ändras beroende på vem det är som är inloggad. På bilden nedanför så har användaren Kalle loggat in.



Medan här har användaren a1lgusliG loggat in.



Funktionen för detta ser ut så här:

```
function Medlemssida() {  
  
    var user = sessionStorage.getItem("username");  
    if(user != null){  
        $("h1#message").html(' Välkommen ' + user);  
    } else{  
        window.location.href = "Login.html";  
    }  
}
```

Som fungerar så att den hämtar användarnamnet och jämför den med noll. Om den inte är noll så skrivs ett meddelande ut tillsammans med namnet annars så skickas man tillbaka till login sidan.

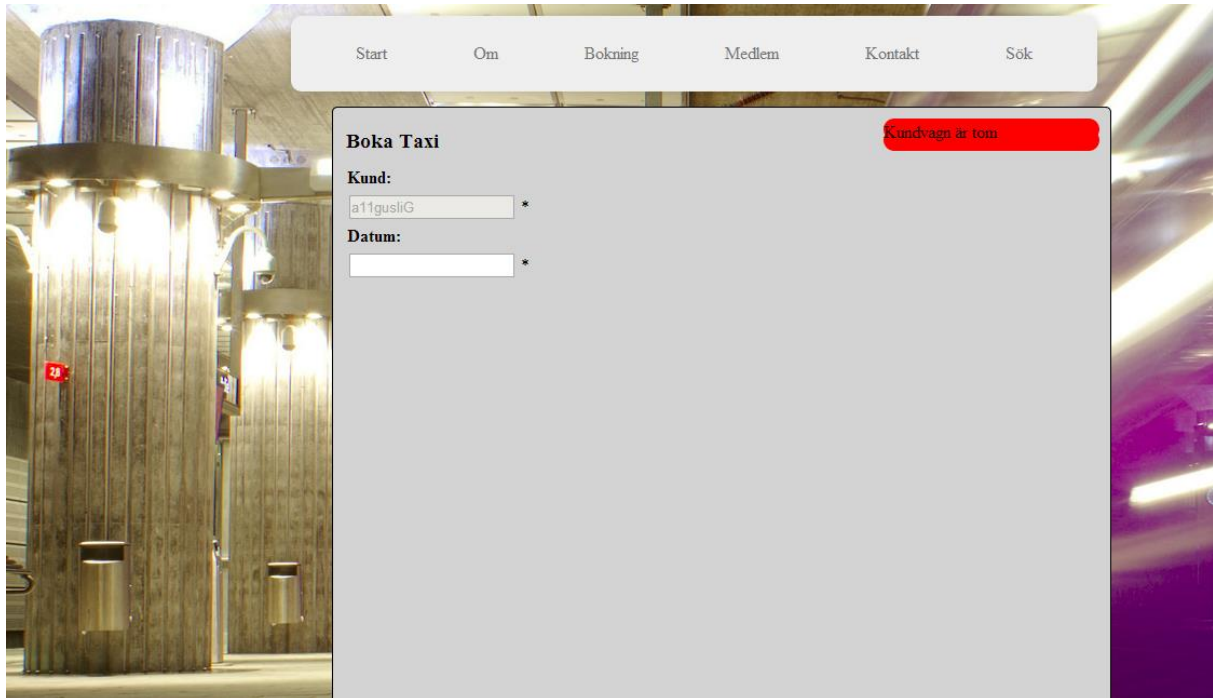
När man loggar ut så blir man tillbaka skickad till startsidan. Anledningen till det är för att då har man verkligen loggat ut och man kanske vill titta runt mer på sidan eller till och med stänga ner den. Det hade gått att bli skickad till Login sidan istället men när man har loggat ut så är man ju på väg ifrån sidan och då vill man inte se login sidan igen. Så därför skickas man till Start istället. Sidan använder sig utav session storage just för att när man sparar ner i en session så kommer all data försvinna när man stänger ner webbläsaren. Den är bara temporär.

```
function Logout() {  
    sessionStorage.clear();  
    window.location.href = "index.html";  
}
```


}

Bokning-Bokning.html

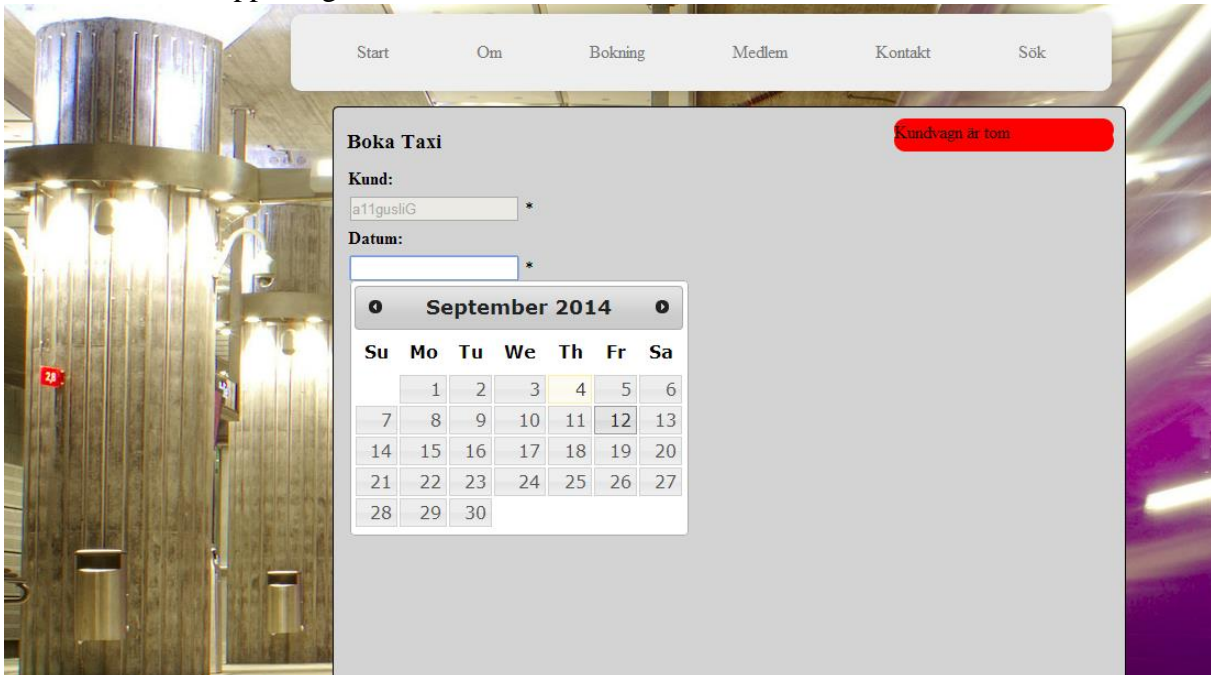
När man först kommer till bokningssidan så möts man av den här vyn:



The screenshot shows a web application interface for booking a taxi. The background is a photograph of a modern interior space with large, cylindrical wooden pillars and warm, ambient lighting. Overlaid on this is a light gray rectangular form titled "Boka Taxi". At the top of the form, there is a navigation bar with links: "Start", "Om", "Bokning", "Medlem", "Kontakt", and "Sök". Below the navigation bar, the form contains two input fields: "Kund:" with the value "a11gusliG" and "Datum:". Both fields have a small asterisk to their right. In the top right corner of the form, there is a red button labeled "Kundvagn är tom".

Som man kan se så är användarens namn redan ifyllt och det enda man kan trycka på här är datum fältet. Man kan inte skriva i kund fältet för att bokningarna sker på den användare som är inloggad. Anledningen till det är för att det är ingen annan än den inloggade kunden som skall boka och så skall denne skriva sitt namn. Om man skriver in ett annat namn så finns risken att det då kan uppstå en massa fel om man så skriver något annat namn där. T.ex. att det inte går in i systemet osv.

När man trycker på datum så kommer en kalender upp som låter användare att trycka på ett datum och då få upp lediga resurser från databasen.



Start Om Bokning Medlem Kontakt Sök

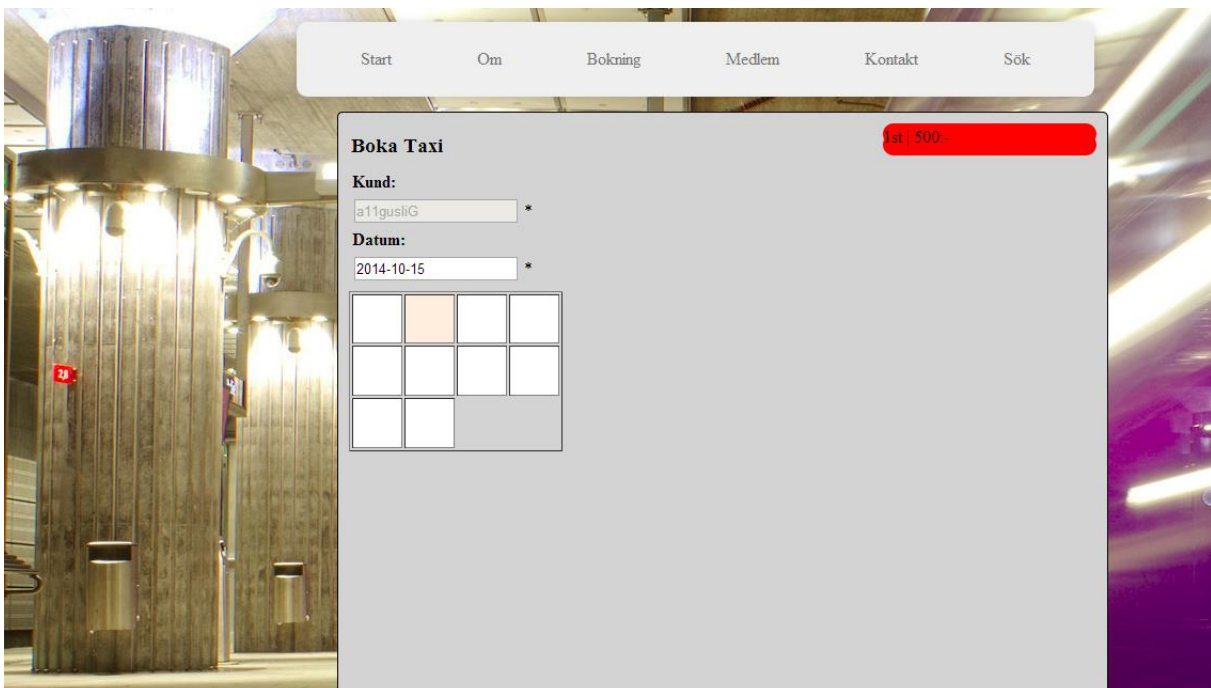
Boka Taxi Kundvagn är tom

Kund: a11gusliG *

Datum: *

September 2014

Su	Mo	Tu	We	Th	Fr	Sa
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30				



Start Om Bokning Medlem Kontakt Sök

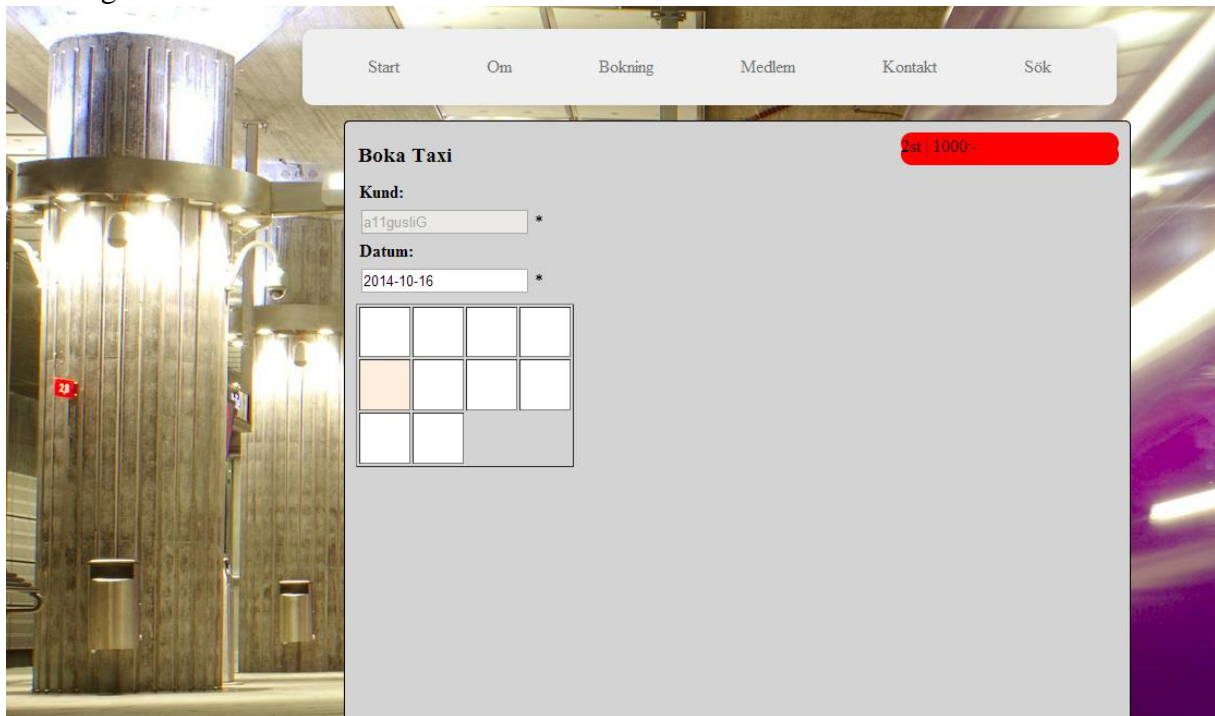
Boka Taxi 1st 500,-

Kund: a11gusliG *

Datum: 2014-10-15 *

När man har tryckt på ett datum så kommer en tabell med olika rutor. När man trycker på en utav dessa så bokar man en resa. När man tryckt på en ruta så blir denna markerad och det går inte att boka på den platsen igen. Varje gång man bokar så läggs en bokning till i

kundvagnen.



När man tryckt på ett datum så körs denna funktion. Den fungerar så att när datumet har valts så hämtas resursens storlek från databasen, alltså hur många platser det finns att boka för kunden. Den hämtar även alla bokningar gjorda på resursen från databasen.

```
function processinputbox()
{
    resource= "Taxibil";
    bookingdate=document.getElementById("date").value;

    $.ajax({
        type: 'POST',
        url: 'booking/getresourcesize_XML.php',
        data: { resourceID: resource},
        success: ResultSize
    });

    $.ajax({
        type: 'POST',
```

```

        url: 'booking/getbookings_XML.php',
        data: {
            resourceID: resource,
            date: bookingdate,
            type: 'allgusli'    // Filter out bookings made from other
applications application.

                                // Most commonly user name of
student

        },
        success: ResultBooking
    });
}

```

När den har gjort detta så sen funktion som heter drawResult. Det är denna funktion som gör att kunden kan välja plats genom att klicka, den målar upp rutnätet alltså. När man sedan trycker på en ruta och bokar så körs två funktioner. Dessa är bookposition och bookingmade.

Funktionen bookposition. Som ser ut så här:

```

function bookPosition(position)

{
    var rebate=0;
    var resource="Taxibil";
    var bookingdate=document.getElementById("date").value;
    var customer=sessionStorage.getItem("username");
    var status = 1;

    $.ajax({
        type: 'POST',
        url: 'booking/makebooking_XML.php',
        data: {
            Name: "allgusli",
            resourceID: resource,
            date: bookingdate,
            customerID: customer,
            rebate: rebate,

```

```

        status: status, // 2 = "Real" booking.
        position: position,
        type: 'allgusli' // Filter out bookings made from other
applications application.

// Most commonly user name of
student

    },
    success: bookingmade,
    error: errormsg
});
}

```

Den fungerar så att när man trycker på ett en plats i rutnätet så gör den bokningen och metoden bookingmade körs. Den uppdaterar kundvagnen och skickar ett meddelande som säger att man har bokat en resurs.

```

function bookingmade(returnedData)
{
    uppdateraKundvagn();
    alert('booked!');
    processinputbox();
    getCustomer();
}

```

Funktionen för att uppdatera kundvagnen ser ut så här:

```

function uppdateraKundvagn(){
    var customer=sessionStorage.getItem("username");
    $.ajax({
        type: 'POST',
        url: 'booking/getcustomerbookings_XML.php',
        data: {
            customerID: escape(customer),
            type: 'allgusli' // Filter out bookings made from
other applications application.
            // Most commonly user name of student
        },
        success: function(returnedData){

```

```

var resultset=returnedData.childNodes[0];
var user = sessionStorage.getItem("username");
var antal=0;
var summa=0;
for (i = 0; i < resultset.childNodes.length; i++) {

    if(resultset.childNodes.item(i).nodeName=="booking"){
        var resource=resultset.childNodes.item(i);
        if(user==resource.attributes['customerID'].nodeValue){ antal++;
        }
        summa += parseInt(resource.attributes['cost'].nodeValue);
    }
}

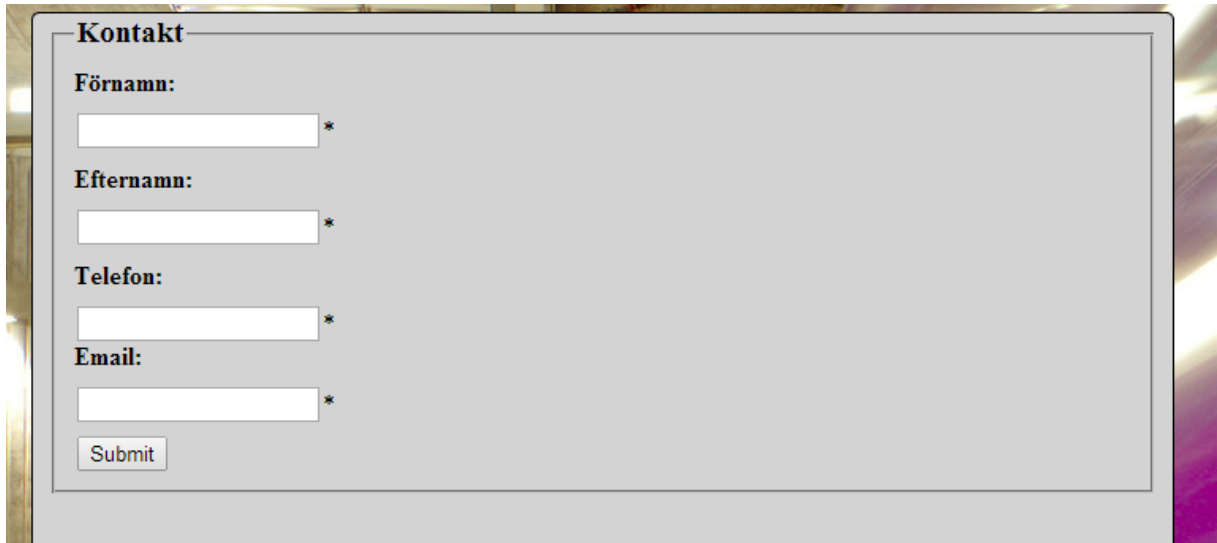
if(summa > 0){
$("span#Kundvagn").html( antal + " resor bokade " + " | " + summa + "kr");
}else{
    $("span#Kundvagn").html("Kundvagnen är tom");
}

```

Den fungerar så att den hämtar alla bokningar gjorda på det användarnamnet. Sedan så skapas två variabler, dessa är antal och summa. Dessa skall representera hur många bokningar man har gjort och hur mycket den sammanlagda kostnaden är för bokningarna. Alltså varje gång en bokning är gjord så läggs det på ett pris och en bokning i kundvagnen.

[Kontakt – Kontakt.html](#)

På kontakt sidan så finns det bara ett formulär och detta ser ut så här :

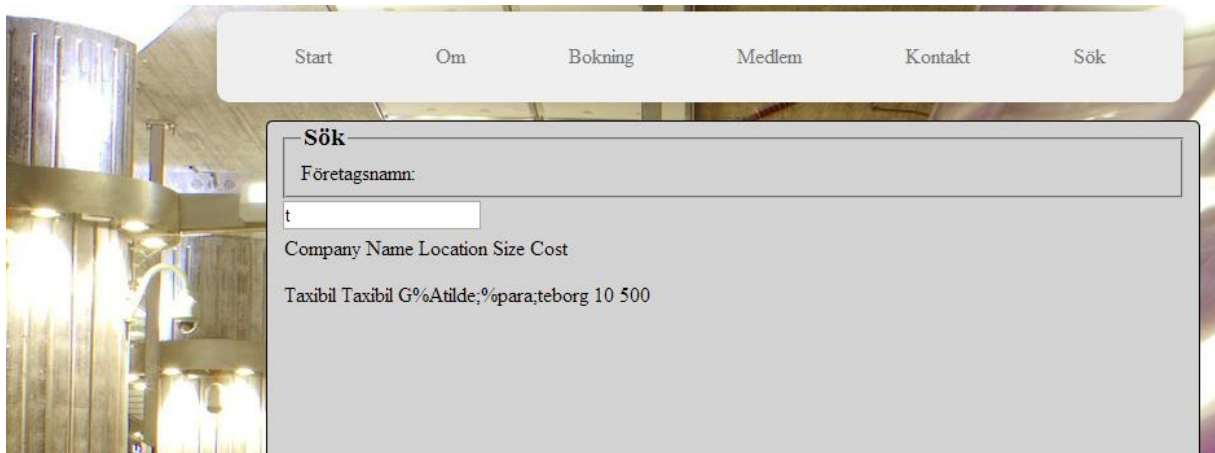
A screenshot of a web form titled "Kontakt". The form is enclosed in a light gray border. It contains four input fields, each with a label to its left and an asterisk to its right, indicating they are required. The labels are "Förnamn:", "Efternamn:", "Telefon:", and "Email:". Below the "Email:" field is a "Submit" button. The form is set against a light gray background.

Den har en validerings funktion som är likadan som registreringsformuläret. Dock skiljer sig en grej här och det är att det är ett telefonnummer med. Den fungerar så att den kollar så att det bara skrivs siffror i fältet. Om det inte gör det så kommer ett meddelande komma upp som säger att det är fel. För det skall inte finnas något annat än siffror i ett telefonnummer.

```
function validatenumber(telefon)
{
    var number = /^[0-9]{10}$/;
    if(telefon.value.match(number))
    else
        if(!telefon.value.match(number))
            alert("Det får bara finnas siffror, max 10 stycken T.ex. 0777-727272");
}
```

Sök- sök.html

På sök sidan så är tanken att man skall kunna söka på data om resursen. Så här ser du ut när man kommer in på sidan.



Company Name	Location	Size	Cost
Taxibil	Taxibil	10	500

Vad den gör är att man söker på företagsnamn och så kommer datan fram. Då det finns ett problem med mina bokningar och API:s interface så blir det konstigt för Göteborg är resursens location och eftersom den har bokstaven ö med så skrivs den inte ut korrekt. Så här ser funktionen ut för sökningen. Sedan finns det en funktion som skriver ut datan i en tabell.

```
function searchResources()
{
    var resname=document.getElementById("resName").value;

    $.ajax({
        type: 'POST',
        url: 'booking/getresources_XML.php',
        data: { name: escape(document.getElementById("resName").value),

                type: 'allgusli' // Filter out bookings made from other
                applications application.

                // Most commonly user name of student

            },
        success: showResources
    });
}
```

Detta var inte den mest optimala lösningen och inte heller den som var med önskvärd. Tanken var för att ha en sök funktion på användarens bokningar. Men eftersom det strular med koden och API så syns inte mina bokningar i systemet och då blir det svårt att försöka hämta något om man inte riktigt vet om det ligger där eller. Då kravet var: *"Det skall finnas en "sök sida" som tillåter sökningar på resursernas data och/eller tillgänglighet på resurs."*

Så fick det bli på resursen data istället. Den mest optimala lösningen hade varit om man kunde söka på användarens bokningar och få ut dom i en tabell. Men det är något som planeras att fixas i framtiden. Då skulle också helst söksidan ligga på bokningssidan istället för en egen sida. Men då det krånglade med systemet så fick det bli en egen sida istället. Helst skulle det också finna en sök knapp men när den sådan implementerades så började koden att sluta fungera så den togs bort. Men planen är att den skall komma tillbaka.

Diskussion

Det har varit en rolig men tuff kurs. Jag har bara mig själv att skylla egentligen då jag inte har jobbat ordentligt och fått en tidspress på mig så därför är det mycket enkla lösningar gjorde på sidan. Så den största lärdomen jag tar med mig från denna kurs är att börja jobba i tid så slipper man sitta in i det sista och nöta för det funkar inte. Det tog även ett tag för mig att förstå hur man skulle koppla sin applikation till APIT och databasen vilket gjorde att jag tyckte det var svårt.

Nu fick vi mycket kod via kodexemplen som gjorde det lättare och det tyckte jag var bra men det svåraste var nog att få till canvasen faktiskt. Satt och försökte väldigt länge innan jag fick till det av någon anledning verkar det som att det buggar när man lägger koden i ett separat javascript dokument så vissa html sidor är långa för att jag då var tvungen att lägga dessa funktioner på html sidan istället för i ett externt script som hade varit det er önskvärda och smidigaste. Men eftersom jag hade pressen på mig så blev inte min canvas som jag ville att den skulle vara. Helst skulle det vara en taxibil som körde men hade inte tiden till att lösa det så fick bli en enkel nödlösning. Så den största lärdomen från denna kurs är att inte börja försent.

Även bokningen blev inte heller som jag tänkt. Helst hade jag velat göra en egen virtuell bokning, alltså istället för att ha rutor som man klickar i när man vill boka något så kommer det en ruta med bilder på taxibilar uppradade så man enkelt kan välja en utav dom och sedan att den läggs i kundvagnen. Detta fungerar dock i dagsläget men jag hade velat ha en egen design på detta med egna bilder osv. Nu hann jag inte fixa det men tanken är att jag skall ta och fixa till sidan ordentligt som jag vill ha den. Kundvagnen kunde också varit annorlunda. För tillfället så går det inte att tömma den något som är ett önskemål att ha. För blir konstigt om det bara plussas på hela tiden så fort man bokar. Till slut kommer det bli jätte dyrt för kunden. Hittade inget i kraven att man var tvungen att kunna tömma vagnen så därför finns det inte sådan funktion, jobbade på att försöka lösa det men lyckades inte. Men

Som jag nämnde tidigare i rapporten så är jag inte helt nöjd heller med hur söksidan blev. Jag Den fungerar och så men jag hade först och främst velat söka på annan data. Jag hade velat ha en funktion att man som kund har en sorts historik där man kan söka på sig själv och så får man upp alla ens tidiga och aktiva bokningar. Denna skulle då vara placerad på bokningssidan. Men då det krånglade med att kunna visa mina bokningar på apit interfacet så blev jag fundersam på om det skulle fungera eller inte så tog det säkra före det osäkra och samtidigt som jag läste kravet så valde jag därför att söka på resurser istället. Nu fungerar inte det klockrent heller då Göteborg skrivs ut fel. Har försökt lösa detta men tror det är databasen som gör att det krånglar. Men planen är att kunna ändra det till bokningar istället och göra som jag vill ha det.

Det jag är mest stolt över i kod väg är faktiskt hur jag lyckades lösa valideringen på telefonnummer. När jag började titta på hur man skulle göra så tog det ändå en stund innan jag fick till det så den är jag mest nöjd över. Men jag hade velat kunna göra mer avancerade grejer nu och ångrar väldigt mycket att jag inte kom igång ordentligt för jag tror och vet att jag hade kunnat göra ett mycket bättre jobb än vad jag har gjort. Så det finns mycket på sidan som jag hade velat förbättra och göra annorlunda. Annars har det varit en rolig kurs, en av de roligaste vi har läste hittills.

Referenser

<http://www.html5canvastutorials.com/advanced/html5-canvas-animation-stage/> - Canvas tutorial