# <HTML> OS

*Ett operativsystem på webben*




**Marcus Kryle**

# Innehållsförteckning

# Mål & syfte

**Mål**: Att skapa ett fungerande operativsystem-,
med olika appar & inställningar,
i en "webb-app" som kan köras i webbläsaren.

**Syfte**: Skapa en kul applikation, men framförallt
vidareutvecklas inom javascript, lära mig
mer om hur olika "moduler" fungerar ihop med
andra och hur man får dem att samarbeta i
ett större projekt.

# Introduktion

Jag valde att göra ett operativsystem för det var något
jag försökte med innan utbildningen med html och css.
Vid sidan av utbildningen försökte jag även greja små
funktioner i detta projekt då vi lärde oss javascript,
vilket även var mitt största mål att bemästra.
Jag fick även mer information av läraren på fritiden,
och alltid bra återkoppling-, eller vart jag kunde
finna information. Det kändes som att jag hade lärt mig
alla byggstenar jag behövde för att få rull på denna app.
Jag installerade "vue-client" och detta är första gången
jag ska skriva ihop ett större projekt på egen hand.
Planen är ju att börja ifrån noll och skriva allt i
javascript med vue.js.

**Kravlista**

Detta ska vara med:

-olika appar såsom bild, musik och anteckningar

-de ska kunna öppnas och stängas

-alla appar ska köras i en större app vilken ska fungera som ett "OS"

-spara & läsa in filer

-en inställningspanel vilken ska utföra olika förändringar
t.ex byta bakgrund osv.

Om tid räcker till:

-stänga av & starta, samt logga in & logga ut

-multipla filer för inläsning & sparning

-kunna flytta "fönster" i OS'et

-ändra utseende på menyer & paneler osv

# Metod

Jag har skrivit webbappen i vue-cli, och har byggt upp den på ett sätt som är "snäll" för vidareutveckling av projektet.
Jag började med att sätta upp vue-cli och bygga upp första huvud-delen av appen "skrivbordet". Okej, olika views, nice!
Jag gör en start vue lägger en dator i bakgrunden så man lättare förstår att här är ett operativsystem. Lägger in en skrivbordsbild och med css så ser det riktigt bra ut. Jag skapar en meny i överkant och en i underkant, det blir en "dock" inspererad av mac os.
Jag använder komponenter för apparna, vilket blev väldigt fördelaktigt då varje app finns i sin helhet, för sig själv, i en komponent och importerar dessa i viewen. Det blir blir både lätt att tillverka och att ändra var och en app för sig, även möjlighet att lägga till och ta bort appar på ett enkelt sätt. Jag använt mig av router för olika routes vilket blir ännu trevligare och utökar möjligheterna mer. Allt blev struktuerat på ett fint sätt och det känns som en utomstående men ändå kunnig på vue skulle kunna navigera, göra ändringar och lägga till eller ta bort saker utan problem.
Backendet skrev jag i Node.js, vilket levererar och ändrar en textfil i html OS'et. Hade mer tid funnits skulle jag nog vidareutvecklat det för flera filer och appars funktionalitet.

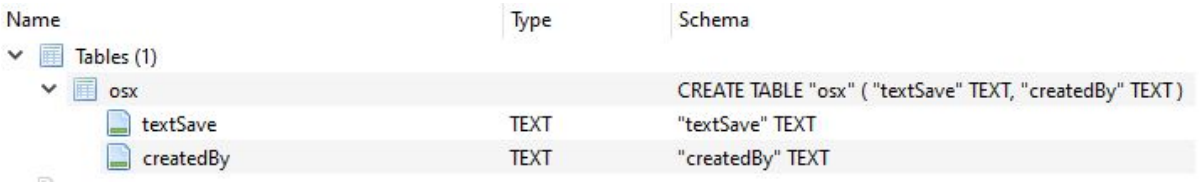
Design
Detta blev rätt lätt då jag hade något att utgå ifrån, eller allt ska ju ligga i en stor "div" och inte utanför. Sen tog det mycket tid att skriva all css på egen hand då alla under-apppar skulle ha sitt eget utseende osv.

Resultat

Det är ett fullt fungerande operativsystem som körs i webbläsaren.
Det tog timmar att googla runt och finna lösningar på saker jag inte hade en aning om hur man fick ihop. Bollade även med olika vänner om tillvägagångssätt, och fick ytterligare perspektiv på hur man förstår sig på olika funktioner m.m. i javascript.
Såhär i efterhand känns det självklart och jag har minst sagt vidareutvecklas-, och jag kan nu bestämt säga att, jag kan javascript.

# Teknisk dokumentation

**Databasens struktur:**



| Name | Type | Schema |
|---|---|---|
| Tables (1) | | |
| osx | | CREATE TABLE "osx" ( "textSave" TEXT, "createdBy" TEXT ) |
| textSave | TEXT | "textSave" TEXT |
| createdBy | TEXT | "createdBy" TEXT |

**Backendet:**

```
app.put('/change/:name', (request, response) => {
    //undersöker om användaren är marcus och det kommer ifrån frontendets IP
    if (request.params.name === "marcus" && request.body.ip === "185.76.65.38") {

    database.run('UPDATE osx SET textSave=? WHERE createdBy=?;',
        [request.body.textSave, request.params.name]
    ).then(() => {
        database.all('SELECT * FROM osx').then(texten => {
            response.send(texten)
        })
        response.status(201)
    })
}
})
```

Jag valde att köra en if-sats för att se om användaren är marcus och om frontendets ip stämmer.

**Frontendet:**

**Views'en:**


Frontendet startar med "Start.vue" i vilken jag undersöker skärmstorleken hos användaren och skickar om användaren till mobil eller pad version, om den verkar liten.

```
if (window.matchMedia('(max-width: 767px)').matches) {

  this.$router.push('/startmobile')

} else if (window.matchMedia('(max-width: 999px)').matches) {

  this.$router.push('/startpad')
}
}
```

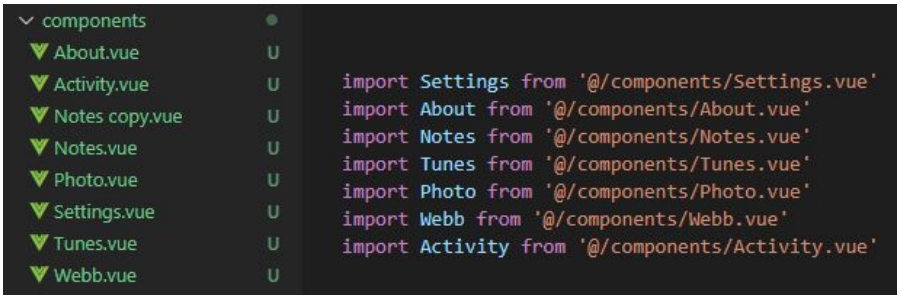
Om användaren kör desktop så utförs ingen redirect och användaren får själv lista ut hur man startar OS'et vilket förenklas med en "click to start" text.


Efter användaren klickat här öppnas "/boot" vilken mest är en "meme" med en fördröjning på tre sekunder för att skapa mer känsla i projektet då jag tyckte den laddas in för fort.
Efter de tre sekunder redirectas man in till huvud applikationen "/running".

**Ruinning Viewen:**



Här importeras alla komponenter som vilka är en enskild app var för sig.



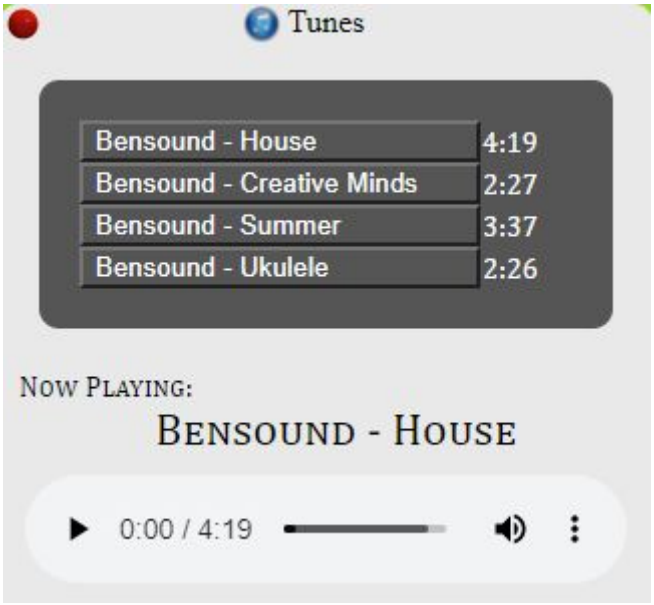

Jag valde att sätta en v-if på varje komponent och en data attribut med true eller false vilket gör det möjligt att visa komponenten eller inte beroende på om attributet är sant eller falskt. Jag tar även här emot emits ifrån komponenterna för att “gömma” dem på högre ort. Och emits ifrån “settings” för att tysta applikationen och byta bakgrund.

```
<Settings @settings-hide="Hide('settings')" @bg-number="background" @opentab="settab" v-if="settingsVisible"></Settings>
<Notes @notes-hide="Hide('notes')" v-if="notesVisible"></Notes>
<About @about-hide="Hide('about')" v-if="aboutVisible"></About>
<Tunes @tunes-hide="Hide('tunes')" v-if="tunesVisible"></Tunes>
<Photo @photo-hide="Hide('photo')" v-if="photoVisible"></Photo>
<Webb @webb-hide="Hide('webb')" v-if="webbVisible"></Webb>
<Activity @activity-hide="Hide('activity')" v-if="activityVisible"></Activity>
```

“tunes” och “settings” komponenterna (apparna) i synligt läge:

**Komponenterna**

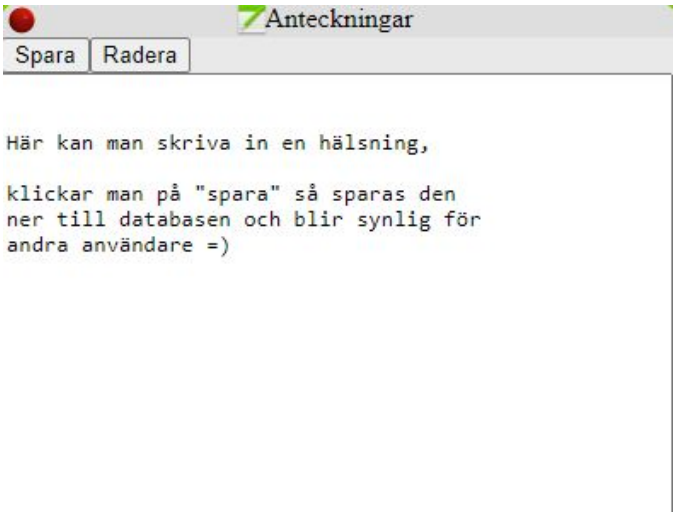
**Tunes:**




Jag använder här ett <audio> element och kör v-model och v-bind ifrån ”buttons” vilka jag sen kör en funktion som sätter “source” till diverse filnamn.
Man kan spela fyra olika låtar ifrån en playlist

```
playTune(tuneName){
  this.musicSrc = 'http://kryle.se/data/bensound-'+ tuneName + '.mp3'
  let player = document.getElementById('tuneplayer')
  this.nowPlays = tuneName
  player.play()
},
```

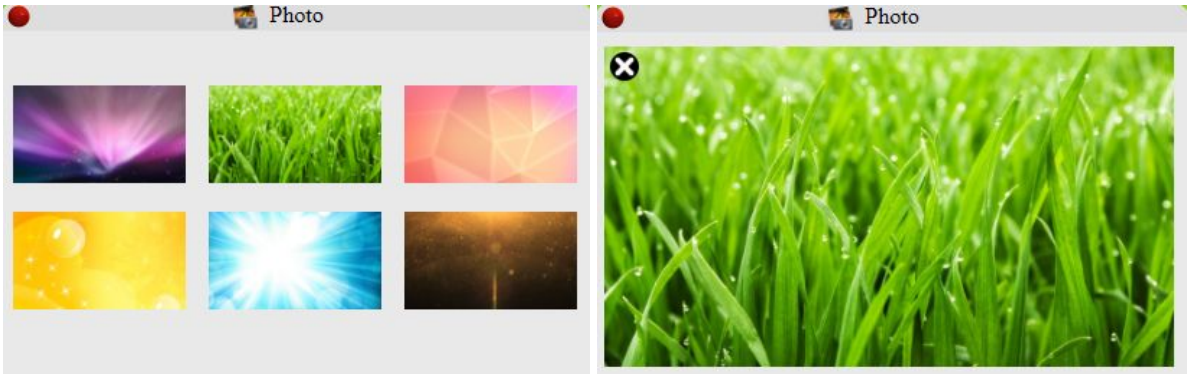
**Notes:**




Här använder jag “fetch()” i en “created()” för att läsa in själva textfilen ifrån Backendet, sen har jag en “radera” knapp vilket bara sätter texten till “=null” och en “spara” knapp med vilken jag för en fetch med en “put” i, till Backendet för att spara ner texten ifrån textfältet.
Jag skickar även med Frontendets IP och mitt namn för godkännande på Backendet.

```
saveText() {
  let stringText = JSON.stringify(this.textSave[0].textSave)
  let frontendIP = JSON.stringify(location.host)
  fetch('http://95.216.198.12/api/change/marcus', {
    body: '{  "ip": '+ frontendIP + '"textSave":  ' + stringText + ', "createdBy": "marcus"}',
    headers: {
      'Content-Type': 'application/json'
    },
    method: 'PUT'
  }).then(function (response) {
    console.log('Saved!')
    alert("Sparat !!")
    return response.json()
  })
},
```

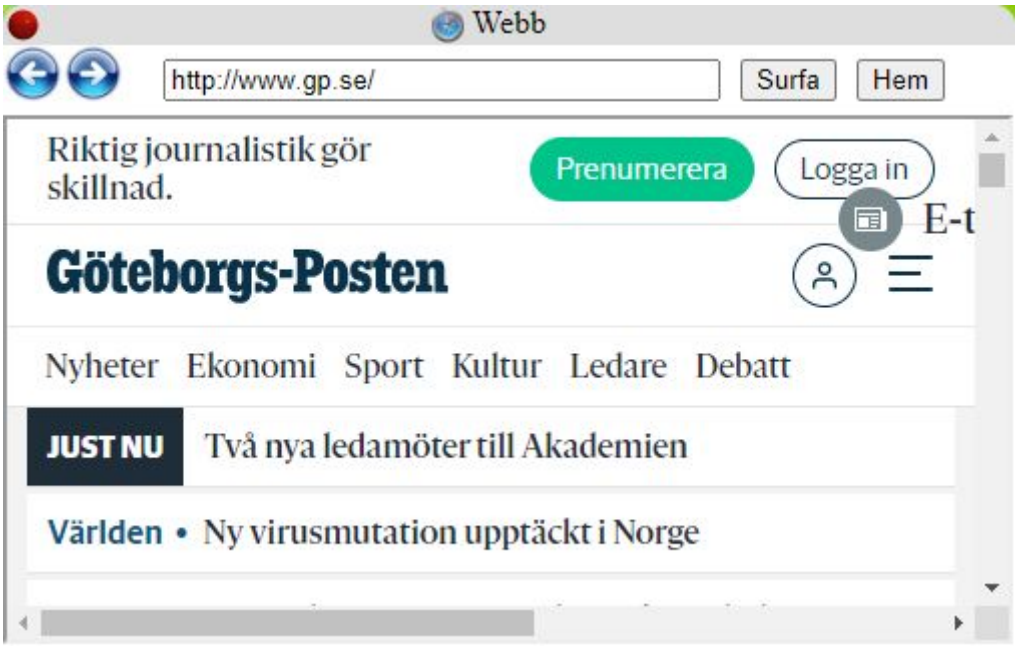
**Photo:**




Här använder jag en funktion som nätt och jämt “stylar”, visar och gömmer den bild man klickar på med “css” kommandon. Beroende på vilket summer som skickas med in, byter den bild till just det filnamnet.

```
<a href="#" @click="changebg('1')"><img src="@/assets/images/back.jpg"></a>
<a href="#" @click="changebg('2')"><img src="@/assets/images/back2.jpg"></a>
<a href="#" @click="changebg('3')"><img src="@/assets/images/back3.jpg"></a>
<a href="#" @click="changebg('4')"><img src="@/assets/images/back4.jpg"></a>
<a href="#" @click="changebg('5')"><img src="@/assets/images/back5.jpg"></a>
<a href="#" @click="changebg('6')"><img src="@/assets/images/back6.jpg"></a>


changebg(number){
  let image = document.getElementById('ontop')
  image.style.visibility = 'visible'
   let bg = document.getElementById('ontop')
bg.style.backgroundImage = "url('http://kryle.se/data/images/back" + number + ".jpg')"
}
```

**Webb:**



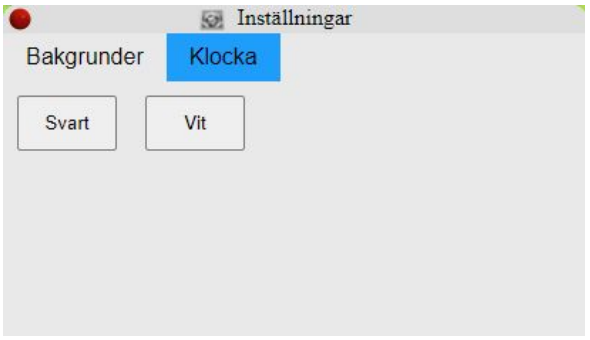

En webbläsare i webbläsaren? -Japp! Det är precis vad det är!
Här hade jag kommit såpass långt i min javascripts orientering så jag var säker på att det var möjligt, men det tog tid att knäcka nöten.
Framåt och bakåt i historiken var lättare än trott då javascript hade inbyggda funktioner för det: “history.back()” och “history.forward()”.
Sen blev det att köra en “form” med text input och knappar till funktioner som sätter en datavariabel med v-model baserat på adressen, som användaren angivit, som i sin tur läses in av en “iframe” när man klickar på “surfa” knappen. Hem-knappen ställer bara in iframe sourcen till sin ursprungliga adress.

```
<a href="#" @click="back()">
<a href="#" @click="forward()">
<form class="src-bar">
<input type="text" v-model="newSrc" class="textinput" autofocus>
<input name="button2" type="button" class="submit" title="Surfa" v-on:click="surf()" value="Surfa">
<input name="button4" type="button" class="submit" title="Hem" v-on:click="home()" value="Hem">
</form>

<iframe :src="browseSrc" class="web-src" scrolling="yes">

    surf() {
     this.browseSrc = this.newSrc
     return
     },
    home() {
    this.browseSrc = "http://www.kryle.se/"
    },
    back() {
     history.back()
     return false
    },
    forward() {
     history.forward()
     return false
```

**Settings:**

flik 1 - Bakgrunder


flik 2 - Klocka




Flik 1 - Bakgrunder:

Här använder jag samma princip som i “Photo” appen. Men det gick ju inte att ändra “parent” komponentens bakgrund härifrån!? Här var också ett ställe där jag fastnade, efter mycket googlande kom svaret-, “To get “child” components to speak with “parent” components one can use “emits” (“this.$emit()”)”.
Okej, djupdykning i emits alltså! Det här sker i fyra steg.

```
Steg 1
<a href="#" @click="background('1')"><img src="@/assets/images/back.jpg"></a>
<a href="#" @click="background('2')"><img src="@/assets/images/back2.jpg"></a>
<a href="#" @click="background('3')"><img src="@/assets/images/back3.jpg"></a>
<a href="#" @click="background('4')"><img src="@/assets/images/back4.jpg"></a>
<a href="#" @click="background('5')"><img src="@/assets/images/back5.jpg"></a>
<a href="#" @click="background('6')"><img src="@/assets/images/back6.jpg"></a>

Steg 2
background(number) {
  this.$emit('bg-number', number)
 }

Steg 3
<Settings @bg-number="background" ></Settings>

Steg 4
background(number) {
let bg = document.getElementById('desk')
bg.style.backgroundImage = "url('http://kryle.se/data/images/back" + number + ".jpg')"
```

Steg 1 (child-component): kör funktionen background() och skicka med ett nummer för respektive bild.

Steg 2 (child-component): skicka iväg en “emit” med “bg-number” och siffran som en “payload”.

Steg 3 (parent-component): lyssna efter en “emit” med “bg-number” och kör funktionen “background()” om en sådan ropas på.

Steg 4 (parent-component): backround() funktionen med ett nummer vilket sänds med som en “payload”, det trevliga med payloads är att man i princip kan skicka med vad man vill, t.ex. en sträng med valfritt innehåll eller liknande, i detta fallet ett nummer i en sträng.
sen sätter jag med javascript “background-image” i css’en till bildfilnamn baserat på nummer 1-6, som sen skickas med i payloaden, hmm javascript är ju både kul och nice!

Flik 2 - Klockan:

Flikarna löste jag på liknande sätt, men skickade med strängen “background”, för att visa fliken bakgrunder, och om något annat skickas med visas klock-fliken.
Klockan löste jag även den på samma sätt, men inställning här har ingen större vikt i “desktop” mode, då klockan där ligger i en grå meny. Denna inställning la jag till först när jag gjorde projektet responsivt, då klockan visas direkt på bakgrunden, och väljer användaren en mörk eller ljus bakgrund, så kanske inte klockan syns tydligt nog, då finns det möjlighet att ändra färgen på den till svart eller vit.


**Aktivitetskontroll:**


Denna app är i en lite mindre skala, jag la till den för det kändes som det skulle vara mer “känsla” i projektet med fler appar i det.
Den kör samma fetch() funktion som “Notes”-appen ifrån Backendet, och om den får något svar så ändras texten från “Offline” till den som visas i bilden “Runnig”.

```
fetch('http://95.216.198.12/api/')
.then(response => response.json())
.then(result => {
  if (result !== undefined) {
    this.backendStatus = true
    this.backendON = "Running"
  }
})
```

Efter det gör jag en fetch() till en api som drivs av “ipify.org” vilken undersöker användarens IP-nummer och skickar tillbaka den i ett json svar.
Nederst har vi en inbyggd funktion i javascript “navigator.appVersion.indexOf()” som undersöker om användaren surfat in, med windows, mac eller linux osv.

**About:**


Det är väl klart man måste ha information om den här datorn, denna innehåller inga script utan är i princip bara ett visitkort med info.

**Övrigt:**

Komponenterna ville jag naturligtvis kunna flytta omkring på skärmen, så att det imiterar “fönster” i ett riktigt OS.
Detta var mycket komplicerat och jag fick följa en guide för att få ordning på det, men det var det värt! Här är bild på “jquery” koden som jag utgick ifrån och sedan resultatet i “javascript”.

JQuery:

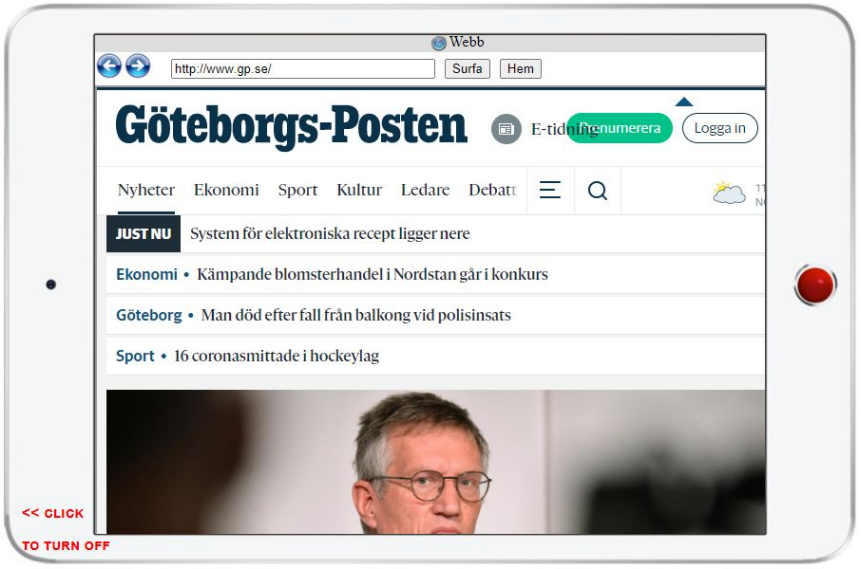
```
var element_position = 0
var iCnt = 0
this.$(document).ready(function () {
  this.$(function () {
    this.$('#about,#settings,#notes,#tunes,#photo,#webb,#activity').draggable()
  })
  this.$(function () {
    this.$("#divResize").draggable().resizable()
  })
  this.$('#MoveMyDiv').click(function () {
    var dynamic_div = this.$(document.createElement('div')).css({
      border: '1px dashed',
      position: 'absolute',
      left: element_position,
      top: this.$('#MyDiv').height() + 20,
      width: '120',
      height: '120',
      padding: '3',
      margin: '0'
    })
    element_position = element_position + this.$('#MyDiv').width() + 20
    this.$(dynamic_div).append('Move it!')
    this.$(dynamic_div).appendTo('#MyDiv').draggable()
    iCnt = iCnt + 1
  })
})
```

JavaScript:

```
  dragMyDiv(event) {
    event.preventDefault()
    this.positions.clientX = event.clientX
    this.positions.clientY = event.clientY
    document.onmousemove = this.divDrag
    document.onmouseup = this.stopDivDrag
  },

  divDrag(event) {
    event.preventDefault()
    this.positions.movementX = this.positions.clientX - event.clientX
    this.positions.movementY = this.positions.clientY - event.clientY
    this.positions.clientX = event.clientX
    this.positions.clientY = event.clientY
    this.$refs.moveMydiv.style.top = (this.$refs.moveMydiv.offsetTop - this.positions.movementY) + 'px'
    this.$refs.moveMydiv.style.left = (this.$refs.moveMydiv.offsetLeft - this.positions.movementX) + 'px'
  },
  stopDivDrag() {
    document.onmouseup = null
    document.onmousemove = null
  },
```

**Pad-versionen:**


Här tog jag “responsivt” till nästa nivå och gjorde helt enkelt pad- och mobil-versioner av projektet, det blev både kul och intressant.
I apparna på pad-versionen flyttade jag “stäng” knappen med css och la den på “Hemknappen” visuellt sett. Exemplet med “Webb” appen öppen.



**Mobil-versionen:**

Running (“ /mobile “):


Avstängd ( “/startmobile” )


Boot ( “/mobileboot” )

Den blev på samma sätt som pad-versionen, ( fast i en mobil :)
och med en app öppen, “Tunes” i detta fallet.

# Sammanfattning

Jag tycker det var ett roligt projekt, och jag känner att jag utvecklades enormt inom ämnet genom att göra det här. Vissa saker gick vi igenom under utbildningen, men ett par av dem saknade jag djupare förståelse för, såsom “emit”s, men efter detta slutarbete så blev jag både mer kunnig och djupare insatt och införstådd med hur saker hänger ihop i JavaScript-världen.
Jag är nöjd över projektet, mest att jag fick ihop det som en helhet, men hade det funnits mer tid hade jag gärna lagt till fler appar och funktionalitet i det. Vem vet, kanske bygger jag vidare på det i framtiden som ett kul hobby-projekt. :)
Jag ser verkligen fram emot att få använda mina kunskaper som webbutvecklare i framtiden.