

# Projektrapport - JavaScript 2

Datum: 2019-04-05

Kurs: JavaScript 2

Klass: FEND18

Student: Daniel Hessling

<https://awkes.github.io/Gitarrkungen/index.html>

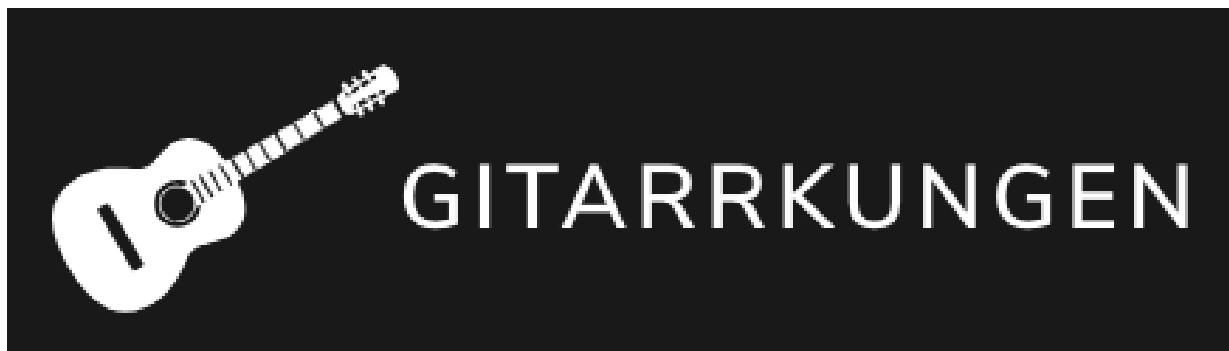
## Introduktion

Projektet var att bygga en webbshop från scratch med HTML, CSS och JavaScript.

Produkter skulle hämtas från en JSON-fil. Kundens beställning och kontaktuppgifter skulle lagras i localStorage för att vi sedan skulle kunna använda informationen, räkna ut summa, lägga till och ta bort produkter och sedan skriva ut den på de andra sidorna (varukorgen och bekräftelsesidan).

## Genomförande

Då vi båda är väldigt intresserade av musik så kom vi snabbt överens om att vi ville ha en webbshop där vi säljer instrument, främst gitarrer. Namnet fick bli "Gitarrkungen".

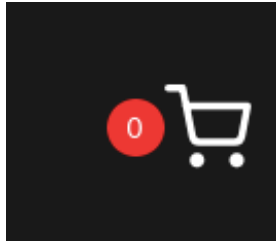


Vi visste att vi skulle behöva tre sidor, en produktkatalog där besökaren kan se alla produkter vi säljer, bilder, priser och en knapp som lägger till produkten i varukorgen.

En sida till varukorgen där kunden kan gå igenom sin beställning, lägga till och ta bort produkter ur varukorgen, eller tömma den helt (även ur localStorage). Här kan kunden också fylla i sina kontaktuppgifter och kunna bekräfta sina beställning. Varukorgen sparas i localStorage som en array i strängformat innehållandes objekt.

Den sista sidan vi skulle göra var bekräftelsesidan där vi skulle visa upp hela beställningen, kundens adress, kontaktuppgifter för att visa upp och se till att kunden har fyllt i allting korrekt. När kunden har bekräftat skulle även localStorage tömmas.

I headern uppe till höger ville vi ha en bild på en varukorg och en siffra som skulle visa om och hur många produkter som finns i varukorgen. Detta använde vi endast på startsidan/produktkatalogen.



## Mockup

Vi började med att göra en mockup på webbshoppen. Den fick en simpel men snygg design. Produkterna visas upp tydligt i stora divar med bilder, produktnamn, beskrivning och priser.

Startsidan / Katalogsidan

# Gitarrkungen

[Varukorg](#)

2

## Elgitarrer

**BILD****Produktnamn**

Lorem, ipsum dolor sit amet consectetur adipisicing elit. Adipisci itaque ad quam illo sit id beatae amet totam lorem ipsum bla. ole. itaque ad quam illo sit id beatae amet totam lorem ipsum bla. ole.

**12900 kr**[Lägg till](#)**BILD****Produktnamn**

Lorem, ipsum dolor sit amet consectetur adipisicing elit. Adipisci itaque ad quam illo sit id beatae amet totam lorem ipsum bla. ole. itaque ad quam illo sit id beatae amet totam lorem ipsum bla. ole.

**12900 kr**[Lägg till](#)

Copyright © Gitarrkungen 2019

JSON-fil

[Varukorg / Checkout](#)

Web 1920 - 2

# Gitarrkungen

Katalog

## Varukorg

BILD

Produktnamn

-

1

+

12900 kr

x

Summa: 32.890 kr

## Skicka beställning

Namn

E-post

Telefonnummer

Adress

Postnummer

Ort

Betalningsalternativ

☐ Postförskott

Bekräfta beställning

Copyright © Gitarrkungen 2019JSON-fil

## Orderbekräftelse-sidan

Web 1920 - 3

# Gitarrkungen

Katalog

## Bekräftelse

Beställare

Förnamn Efternamn

Leveransadress

Gitarrsträngen 6  
174 33 Gitarrstad

E-post

olle@gitarrolle.se

Telefonnummer

070 - 454 45 45

Betalningsalternativ

Postförskott

BILD

Produktnamn

1st

12900 kr

Summa: 32.890 kr

Copyright © Gitarrkungen 2019JSON-fil

Mobile



## BASGITARRER

---



### ROCKSON RSB-110

En bra bas för nybörjare.

Efter att Mockupen var klar så skapade vi en ny repo på Andreas GitHub-konto.  
<https://github.com/Awkes/>

Vi använde oss av 5 brancher som vi sedan merchade in i master branch:  
<https://github.com/Awkes/Gitarrkungen/branches>

**confirm** – till bekräftelse-sidan  
**checkout** – till varukorg-sidan  
**listing** – till produktsidan  
**layout-varukorg** – till varukorgens layout  
**bugfix-checkout** – till att lösa lite buggar

Efter att repon hade blivit skapad så började vi koda sidorna med HTML, CSS och med hjälp av Bootstrap.

Vi hade först tänkt att använda oss av instrument som vi hittat på musikföretags webbshoppar men kom sen på att vi kunde ta våra egna instrument, vilket blev lite roligare. Så vi fotade dessa och la sedan in dem i webbshoppen. (Några av dem nedanför)



Efter detta så skapade vi en JSON-fil med alla våra 10 produkter. Det fick 3 olika kategorier (arrayer) "Basgitarrer", "Gitarrer" och "Leksaker". I varje kategori hade varje objekt ett antal egenskaper: produkt-id, en sökväg till bilden på produkten, produktnamn, en beskrivning samt priset på produkten.

```
{
  "Basgitarrer": [
    {
      "id": 1,
      "url": "images/Rockson-RSB-110.jpg",
      "product": "Rockson RSB-110",
      "description": "En bra bas för nybörjare.",
      "price": 1699
    },
  ],
}
```

## LocalStorage

GITHUB CC PHP JS PS

Network Performance Application >> | : X

Filter

Key	Value
cart	{{"id":1,"qty":1},{ "id":2,"qty":1},{ "id":3,"qty":1}}

```
▼ [{id: 1, qty: 1}, {id: 2, qty: 1}, {id: 3, qty: 1}]
  ▶ 0: {id: 1, qty: 1}
  ▶ 1: {id: 2, qty: 1}
  ▶ 2: {id: 3, qty: 1}
```

På varje sida valde vi att ha en JavaScript-fil med funktioner som skulle användas på alla sidor och sedan fick varje sida en egen script-fil.

**base.js** – Till alla sidor

**catalog.js** – Till produktkatalogen (index)

**checkout.js** – Till checkout-sidan

**confirmation.js** – Till orderbekräftelse-sidan

I basfilen hämtar vi och loopar igenom JSON-filen. Vi har en färdig variabel innehållandes divar med selektorer som hämtar och skriver ut all information på rätt plats.

I bas-script-filen har vi en variabel som kollar om det finns något i varukorgen och det parseas isåfall till ett objekt. annars skapas en tom array.

```
let cart = JSON.parse(localStorage.getItem('cart')) || [];
```



Vi har en funktion för att lägga till produkter i varukorgen (**addProduct()**). När man trycker på lägg till-knappen så loopar funktionen igenom varukorgen, letar efter produktens id och ökar kvantiteten med 1 om den hittas och därefter avslutas loopen. Om produktens id inte hittas i varukorgen så läggs produkten till och sedan avslutas loopen.

En funktion vi har används till att uppdatera siffran i röda cirkeln i headern. En selektor kopplas till siffrans id och funktionen triggas när någon produkt läggs till eller tas bort genom att loopa igenom localStorage och kolla antalet produkter i varukorgen.



Vi har även här en funktion som minskar antalet eller tar bort en produkt i varukorgen (**subProduct()**). Funktionen loopar även här igenom varukorgen och letar efter produktens id. Hittas det så minskas antalet med 1. Finns det endast 1 av denna produkt så tas produkten bort helt med hjälp av splice. Därefter uppdateras localStorage.

En till funktion i varukorgen används om man vill ta bort produkten helt oavsett antal, loopar igenom varukorgen känner av om produktens id matchar

och isåfall tas produkten bort helt i varukorgen (`delProduct()`).

I alla funktionerna `addProduct()`, `subProduct()` och `delProduct()` hittar vi den aktuella knappen kopplat till en specifik produkt genom att man tar attributet `data-id` från knappen som har triggat funktionen och därefter så omvandlar vi värdet till `Number`.

Vi har även en funktion, `updateItem()` som ser till att det bara är den aktuella raden som uppdateras i tabellen på checkout-sidan, den körs från funktionerna `addProduct()`, `subProduct()` och `delProduct()` om `checkout.js` är tillgänglig och bara på checkout-sidan.

I script-filen på katalog/varukorg-sidan har vi ett script som hämtar alla produkter från JSON-Filen, loopar igenom hela filen, formaterar och kategoriserar med kategorinamn samt en variabel innehållandes färdigstrukturerad html, divar där våra produkter visas upp. Det görs genom att en selektor till ett id på vårt main-element där vi använder jQuerys `.html-` metod för att skicka dit datan.

```
let html = '';
$.each(data, (cat, prods) => {
  html += `<h2>${cat}</h2><hr>`;
  $.each(prods, (i, prod) => {
```

```
    html += `
      <div class="card mb-3">
        <div class="row no-gutters">
          <div class="col-md-3 p-3 align-self-center">
            
          </div>
          <div class="col-md-9">
            <div class="card-body d-flex flex-column" style="height: 100%;">
              <h3 class="card-title">${prod.product}</h3>
              <p class="card-text">
                ${prod.description}
              </p>
              <div class="d-flex align-items-center justify-content-between row no-gutters mt-auto">
                <p class="card-text font-weight-bold text-xl-left col-md-8 mb-0">${prod.price} kr</p>
                <button data-id="${prod.id}" class="btn btn-primary col-md-4">Lägg i kundvagn</button>
              </div>
            </div>
          </div>
        </div>
      `;
    });
  });
  // 3. Skriv ut produkterna på sidan
  $products.html(html);
```



På vår "Checkout"-sida så valde vi att lista kundens valda produkter där denne även har möjlighet att ändra antalet produkter denne vill köpa. Under det listade produkterna satte vi ett beställningsformulär där kunden kan fylla i sina kontaktuppgifter och adress dit beställningen ska skickas.

## SKICKA BESTÄLLNING

Namn	E-post
<input type="text" value="Förnamn Efternamn"/>	<input type="text" value="namn@doman.se"/>
Adress	Postnummer
<input type="text" value="Vägnamn 13"/>	<input type="text" value="12345"/>
Stad	Telefonnummer
<input type="text" value="Stad"/>	<input type="text" value="0701234567"/>

SKICKA BESTÄLLNING

I denna sidas script-fil har vi massa selektorer och eventlisteners kopplade till funktionerna i bas-script-filen bland annat. En till funktion här som uppdaterar varukorgen genom att loopa igenom den och kontrollera värden och uppdaterar vår html-variabel(tabell). Funktionen räknar också ut totalsumma och ändrar den beroende på vilka produkter som kunden lägger till och tar bort. Finnes inga produkter i varukorgen så läggs endast en tabellrad till med texten "Din varukorg är tom".

I den här filen finns också ett gäng selektorer till alla input-fält i beställningsformuläret. Varje fält har fått en egen valideringsfunktion. Med hjälp av jQuerys "keyup"- och "change"-event så triggas funktionerna och varje funktion har egna valideringar, bla egenmodifierade RegExp-objekt där vi jämför kundens input med våra egna RegExp-tecken.

Exempel:

```
const re = /^[a-öA-Ö¥-] {2, 50} (?:¥s ([a-öA-Ö¥s¥-]) {2, 50}) +$ /;
```

Varje fält har även ett varningsmeddelande(div) som visas om det är någonting som inte stämmer och som hjälper kunden att se till att formuläret fylls i korrekt. Så fort fältet är korrekt ifyllt så försvinner diven.

Namn	E-po
Danie	
Adress	Post
Maltesholmsvägen 141	
Stad	Telef
Hässelby	

**Namn**

- Minst fem tecken, minst ett mellanrum, minst två bokstäver per namn.
- Du får endast använda "A-Ö", "a-ö", "-"" och mellanrum.


När man trycker på "Skicka beställning" så triggas funktionen `sendOrder()` som kontrollerar att alla fält är validerade.

Är detta fallet så skapas ett objekt i `localStorage` med kundens information och skickar kunden vidare till bekräftelsesidan utan att "submitta" formuläret med hjälp av `preventDefault()`.

På bekräftelsesidan visas kundens orderbekräftelse med all information som denne har fyllt i beställningsformuläret, produkterna samt totalsumman som beställningen landade på.

## ORDERBEKRÄFTELSE

Beställare:	E-post:
Daniel Hessling	daniel.hessling@gmail.com
Leveransadress:	Telefonnummer:
Maltesholmsvägen 141 16566 Hässelby	070707070

PRODUKT	ANTAL	PRIS
 Rockson RSB-110	1	1699 kr

Summa: 1699 kr

I bekräftelsesidans script-fil så kontrolleras det så att varukorgen inte är tom och att beställningsformuläret är korrekt ifyllt. Är det inte det så visas ett felmeddelande upp i en div ("Du har kommit till den här sidan av misstag, gå tillbaka till butiken och försök igen". Med en länk som länkar tillbaka till startsidan/katalogsidan.

## ORDERBEKRÄFTELSE

---

Du har kommit till den här sidan av misstag, gå tillbaka till butiken och försök igen.

Är det korrekt ifyllt så hämtas all information från kunden i "kund"-objektet samt alla produkter som finns i varukorgen och visar upp detta på sidan genom att lägga in all information i en tabell som ligger i en "html"-variabel.

Felmeddelandet och bekräftelsen har som standard "display: none" och visas upp beroende på resultatet. Vid bekräftelsen tömts även localStorage på både varukorg och kund-objektet.

## Slutsatser

Nu i efterhand när vi är klara med projektet så tycker jag att det har vart kul att jobba med det. Jag skulle egentligen inte ha velat använda Bootstrap då jag tycker det är roligare att göra all CSS själv. Vi fick med en del CSS själva iaf. I och med att vi inte hade jättemycket tid på oss så var det nog ändå bra iallafall. Nu var ju dock syftet med projektet att fokusera på JavaScript-delen. Jag tycker att resultatet blev väldigt bra och jag är jättenöjd.

I övrigt har det varit väldigt givande och det har varit kul att lära sig att använda localStorage på det sätt som vi gjort. Jag tror också att kommande studenter kommer tycka att detta projekt är kul.