

RAPPORT PROJEKTARBETE

Datum: 2020-02-12

Kurs: JavaScript 2

Klass: FEND19

Student: ANNELIE_FOGEL

Introduktion

Uppgiften är att skapa en webbsida med ca 10 produkter. Dessa produkter ska kunna läggas till i en varukorg. Det ska gå att ändra antal på produkterna i varukorgen samt ta bort en produkt helt och hållet. Användaren ska även kunna tömma varukorgen helt. När användaren klickar på knappen "Beställ nu" visas en orderbekräftelse i ett nytt webbfönster.

Genomförande

Allra först granskade vi uppgiftens krav och den funktionalitet som krävdes för de olika betygsnivåerna, för att tydliggöra hur koden skulle struktureras upp. Vi valde också ett ämne för vilka produkter vi skulle visa på sidan.


Till att börja med gjordes ett enkelt skelett i html med endast de nödvändiga elementen (header, main, div för produkterna samt footer. Resterande element ansvarar Javascript för att skapa utifrån vår JSON-fil.

Under projektets gång har vi varvat gemensamt arbete där vi sitter tillsammans, med att dela upp olika uppgifter mellan varandra och sitta för oss själva. Vi har använt oss av kursboken, lärarens presentationer och exempel, samt googlat och sökt information på olika relevanta webbsidor, till exempel W3Schools, MDN och Stack overflow.













Webbsidans funktionalitet och strukturen i Javascript

Webbsidan består initialt av en tabell där produkterna visas, se bild nedan.

Kryddexperten - din krydda i tillvaron

 Visa/dölj varukorg

Kryddor

	Namn	Ursprungsland	Pris/st	Antal	
	Basilika	Egypten	20 SEK	<input type="text"/>	 Lägg i varukorg
	Dragon	Frankrike	25 SEK	<input type="text"/>	 Lägg i varukorg
	Ingefära	Nigeria	10 SEK	<input type="text"/>	 Lägg i varukorg
	Kanel	Indonesien	25 SEK	<input type="text"/>	 Lägg i varukorg
	Koriander	Marocko	30 SEK	<input type="text"/>	 Lägg i varukorg
	Oregano	Chile	15 SEK	<input type="text"/>	 Lägg i varukorg

Vår varukorgsarray deklareras initialt som tom och först när produkter börjar läggas till skapas html-elementen för varukorgen högst upp på sidan. Produkter som lagts till presenteras i en tabell med färre egenskaper än i produktlistan. I resten av rapporten kommer jag att benämna varukorgsarrayen med dess variabelnamn, \$cartArray.

Enligt samma princip skapar alltså vår javascript först upp produktlistan utifrån JSON-filen. Produktsidan (eller rättare sagt produktlistan) består av en statisk mängd element som är desamma under hela shopping-upplevelsen (produkterna samt tillhörande knappar). I samband med att "Lägg i varukorg"-knapparna skapas sätts deras id till samma id (siffra mellan 1-10) som själva produkten har. Detta så att rätt knapp kopplas till rätt produkt när man söker på knapp-id.

Varukorgen är å andra sidan av naturliga skäl bestående av en mängd föränderliga element som förändras dynamiskt beroende på användarens interaktion. Nedan en bild där produkter lagts till i varukorgen.

Kryddexperten - din krydda i tillvaron

 Visa/dölj varukorg

Din varukorg

Namn	Antal	Pris/st	
Kanel	<input type="button" value="-"/> 3 <input data-bbox="576 562 609 598" type="button" value="+"/>	25 SEK	<input data-bbox="1096 562 1209 598" type="button" value="Ta bort"/>
Dragon	<input type="button" value="-"/> 2 <input data-bbox="576 640 609 676" type="button" value="+"/>	25 SEK	<input data-bbox="1096 640 1209 676" type="button" value="Ta bort"/>

Totalsumma: 125 kr

 Skicka beställning

 Töm varukorgen

Kryddor

	Namn	Ursprungsland	Pris/st	Antal	
	Basilika	Egypten	20 SEK	<input type="text" value=""/>	<input data-bbox="1209 1039 1388 1113" type="button" value="Lägg i varukorg"/>
	Dragon	Frankrike	25 SEK	<input type="text" value=""/>	<input data-bbox="1209 1165 1388 1239" type="button" value="Lägg i varukorg"/>

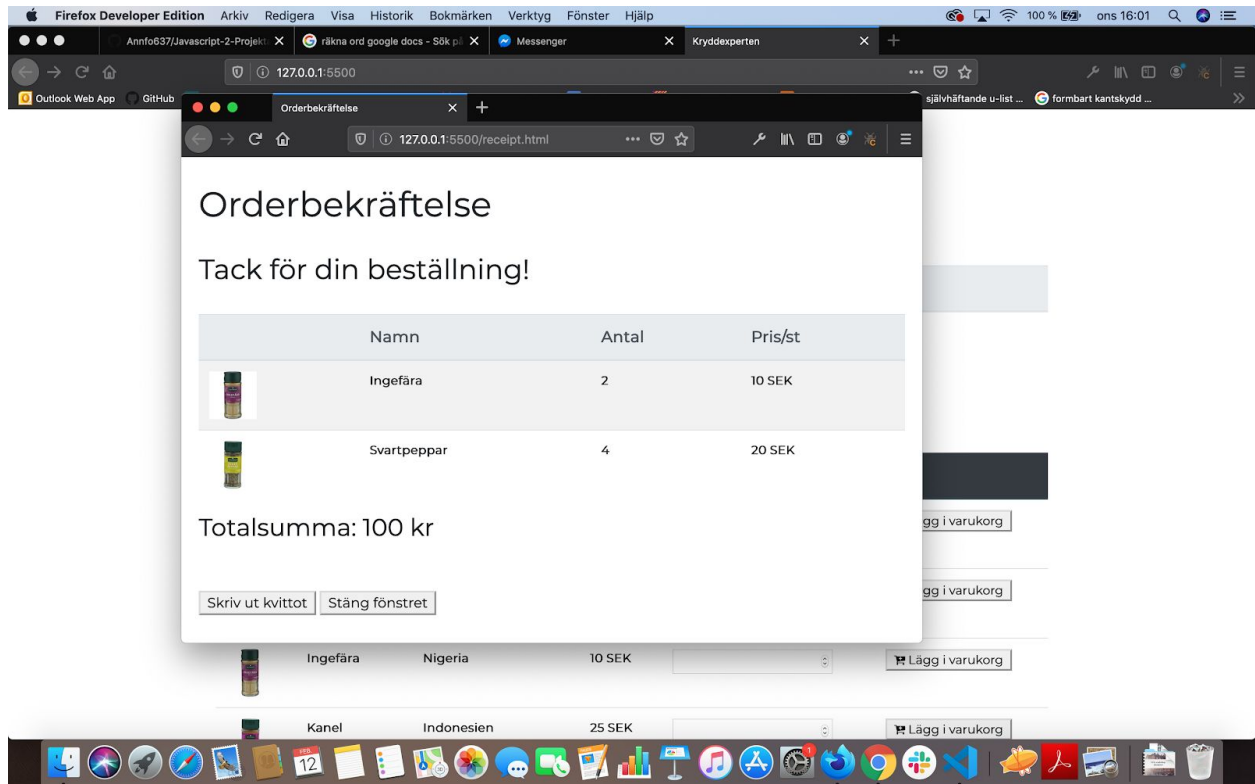
Hjärtat i main.js är funktionen drawCart, som anropas vid samtliga typer av förändring av varukorgen. Den hämtar den aktuella \$cartArray och ritat ut elementen i DOM:en utifrån den uppdaterade statusen. För att hålla en så hög Separation Of Concerns som möjligt så begränsas drawCart så mycket som möjligt till själva skapandet av html-element, dvs presentationen av sidan. Inuti drawCart behöver vi även koppla lyssnare till de knappar som skapats i funktionen så att de direkt knyts till rätt element.

All interaktion som sker utifrån events som triggas vid användarens input hanteras i mindre separata funktioner, som samlats under punkt 3 i main.js. Jag kommer nedan kort beskriva varje funktion i en punktlista.

- 3.1 addQty, anropas när produktlistans fält för antal ändras. Denna funktion ändrar inte \$cartArray utan uppdaterar endast värdet i det aktuella objektets egenskap qty.
- 3.2 addtoCart, anropas vid klick på "Lägg till i varukorgen" och får in knappens id som argument (knappens id är som tidigare nämnts = produktens id). Sedan hämtas motsvarande produkt-objekt från vår JSON-fil, observera att vi för att hämta rätt objekt behöver minska knapp-id med minus 1, eftersom index i JSON-arrayen går mellan 0-9 och inte mellan 1-10 som egenskapen id gör. Med hjälp av \$cartArray.includes görs sedan en kontroll ifall produkten redan finns i varukorgen, i så fall aktiveras en alert och användares ombeds att istället ändra antal på produkten i varukorgen. Annars loopar funktionen igenom vår JSON-array och jämför knapp-id med produkt-id och lägger till den matchande produkten i \$cartArray.
- 3.3 removeFromCart, anropas vid klick på "Ta bort"-knapp och får in knappens id som argument. En variabel för en filtrerad lista deklarerar och via \$cartArray.filter returnerar funktionen endast de produkter som har ett id skilt från det id som den klickade knappen har.
- 3.4 changeQty, anropas vid klick på plus- eller minusknappar och får in den klickade knappen som argument. Orsaken att vi skickar in hela knappen som argument är att vi längre ner i funktionen vill komma åt klasser på knappen, vilket inte går om vi endast får in knappens id som argument. Funktionen loopar för att matcha knappens id med produktens id, går sedan vidare till att antingen öka qty med 1 om knappen har klassen plusOne eller att minska med 1 om knappen har klassen minusOne. Sista else med en alert att något är fel borde aldrig köras, men lade dit den bara som en "fallback" vid testning av kod. Bör kunna tas bort.
- 3.5 emptyCart, anropas vid klick på "Töm varukorgen" samt sist i funktionen sendOrder (se under punkt 3.7). Denna sätter helt enkelt \$cartArray till en tom array.

Samtliga funktioner som på något sätt manipulerar eller ändrar \$cartArray, nummer 3.2 till 3.5, avslutas med att uppdatera localStorage och anropa drawCart för att rita ut uppdaterad \$cartArray. Orsaken att vi uppdaterar localStorage med vår aktuella \$cartArray är att vi sedan vill kunna ha access till den i den nya sidan där kvittot skapas.

- 3.6 totalPrice anropas inuti drawCart i samband med att elementet <h3>Totalsumma: </h3> skapas och tar in aktuell \$cartArray som argument. Den utför egentligen bara en enkel matematisk operation genom att loopa över arrayen och summera ihop qty*price för varje produkt. För att kunna utföra detta behövs dock dessa värden först göras om till "integers" högst upp i funktionen innan loopen startar.
- 3.7 sendOrder anropas vid klick på "Skicka beställning"-knappen. Den visar först en alert med "Din order skickas", sedan anropas openReceiptWindow och slutligen anropas emptyCart.
- 3.8 openReceiptWindow skapar en orderbekräftelse/kvitto i nytt fönster. Se bild nedan.



Kvittot skapas i en separat html-fil, samt har en egen JS-film, receipt.js, där endast kvittot hanteras.

Principen för receipt.js är densamma som för skapandet av produktlistan och varukorgen i main.js. Den deklarerar först en variabel \$receiptArray som hämtar varukorgen från localStorage och ritar sedan ut kvittosidan. För att komma åt varukorgen på olika html-sidor behövde vi arbeta med att först lagring i localStorage.

Längst ner i kvittot lade vi även till knappar för att kunna skriva ut kvittot samt stänga fönstret och komma tillbaka till startsidan.

Slutsatser

Vårt tillvägagångssätt med att varva gemensamt och enskilt arbete tycker jag har fungerat väldigt bra, då det finns fördelar och nackdelar med båda arbetssätten. En stor fördel med att jobba i team på ett projekt är att kunna "byta" uppgifter med varandra, antingen ifall man kör fast eller helt enkelt för att variera arbetet. Det kan ibland vara ganska lätt att bli blind för sin egen kod, och ofta kan ett par nya ögon hitta lösningar som den andre inte tänkt på. Ibland kan det dock vara mer tidseffektivt att sitta på egen hand. Som sagt, båda sätten har sina respektive plus och minus.

Överlag är det nyttigt att vara flera utifrån att man då på ett mer naturligt sätt får bolla och utvärdera olika infallsvinklar och synsätt på genomförande, presentation och struktur av kod.

Vi har försökt använda JQUERY så mycket som möjligt, men jag är medveten om att koden inte är helt enhetlig utan att vi ibland skriver vanilla JS.

Vi har medvetet valt att jobba med enkel CSS, då detta inte är fokus i denna kurs.

Förbättringsförslag, saker som kan göras annorlunda

- I nuläget kan man lägga till en produkt utan att ha fyllt i antal, bör förhindras med hjälp av villkor
- I nuläget kan man klicka minus i varukorgen och få negativt antal, bör förhindras med hjälp av villkor
- Dela upp Main.js i mindre script-filer för bättre överskådlighet.
- Skapa key/value för id och qty i javascript istället för direkt i JSON-filen.
- Även i html kunnat flytta fler objekt till Javascript?
- Presenterat produkterna som "produktkort" i t ex flexbox eller grid istället för i en tabell.