Inlämningsuppgift: Webbshop (individuell uppgift)

🍩 Introduktion till uppgiften

Gottfrid Jästson är en kreativ och företagsam person i Snaskköping, som i många år har sålt munkar i butik. Tyvärr har han märkt av sjunkande försäljningssiffror på grund av allsköns "kurirer" som far runt på livsfarliga elsparkcyklar i staden. Det verkar som att ungdomarna inte längre orkar gå utanför dörren för att inhandla mat?!

Nåväl, Gottfrid misströstar inte mer än nödvändigt; en liten webbshop för munkar* vore på sin plats - och just ditt företag, `Webbyrån Justin Time AB` , har blivit utvald som leverantör för att bygga den! Det vilar nu på dina axlar att Gottfrid inte går i konkurs.

Oj, oj, oj! En webbshops-prototyp på bara fyra veckor?! Hur ska detta gå? 😮🏎️

Bra såklart!

* Du får gärna ta ut de kreativa svängarna och kränga en annan produkt också, såklart (strumpor, säkerhetsdörrar, kottar) - så länge funktionsbeskrivningarna följs.

Se Referensbilder längst ner i dokumentet.

> **Tips**
>
> Uppgiften innehåller (som vanligt) mycket text. Vi kommer att gå igenom delar av inlämningsuppgiften som lektionsövningar och demos, så ni kommer att få hjälp på vägen (lösningsförslag). Därmed dock inte sagt att ni får hela lösningen, viss ansträngning krävs 😉
>
> Skriv pseudokod och lägg tid på att tänka innan ni börjar koda, så slipper ni gå omkring med panik (se "Förslag på upplägg" nedan).

Planera och bryta ner projekt med hjälp av GitHub Issues

13:41

- Länk till min gitignore

Funktions-/kravbeskrivning

Generellt

- Beställningssidan ska vara `EN` sida; du ska inte växla mellan kundkorg och produktöversikt t.ex. Det räcker alltså med `EN` HTML-fil och tillhörande JavaScript-kod.
- Webbshoppen ska vara responsiv. Hur design/utseende ser ut, är upp till er, men det ska fungera på mobil, tablet & desktop.
- Gränssnittet/sidans utseende ska vara intuitivt.
- Det ska gå att utföra hela beställningsförfarandet med hjälp av enbart tangentbordet (tillgängligt).
- Bredvid/i anslutning till varje munk ska det finnas knappar för att öka och minska antalet beställda munkar. Tänk på att dessa ska fungera även med tangentbordet.
- Munkarna är hårdkodade i filen (de behöver inte komma från ett API, en databas eller JSON-fil).

- När man tryckt på beställ-knappen så ska en bekräftelse-ruta visas med information om beställningen och leveranstid.
- Produkterna ska gå att sortera utifrån namn, pris, kategori och rating

▶ 🤿 **Extra (frivilligt, påverkar ej betyget)**

Beställningssammanfattning

- Totalsumman ska uppdateras baserat på ändringar som sker i antal beställda munkar i realtid
- Det ska finnas en varukorgssammanställning som visar endast de munkar som har beställts. Denna ska alltså vara skild från själva beställningsformuläret. Se referensbilder längre ner.

Munkarna (produkterna)

- Varje munk ska ha följande egenskaper:
  - Ett namn
  - Ett pris
  - En rating
  - En kategori
- Det ska finnas minst 10 munkar i webbshopen.

▶ 🤿 **Extra (frivilligt, påverkar ej betyget)**

Gottfrids specialregler

- På måndagar innan kl. 10 ges 10 % rabatt på hela beställningssumman. Detta visas i varukorgssammanställningen som en rad med texten "Måndagsrabatt: 10 % på hela beställningen".
- På fredagar efter kl. 15 och fram till natten mellan söndag och måndag kl. 03.00 tillkommer ett helgpåslag på 15 % på alla munkar. Detta ska inte framgå för kunden att munkarna är dyrare, utan priset ska bara vara högre i "utskriften" av munkarna.
- Om kunden har beställt för totalt mer än 800 kr ska det inte gå att välja faktura som betalsätt.

- Om kunden har beställt minst 10 munkar av samma sort, ska munkpriset för just denna munksort rabatteras med 10 %
- Om kunden beställer totalt mer än 15 munkar så blir frakten gratis. I annat fall är fraktsumman 25 kr plus 10% av totalbeloppet i varukorgen.
- Om kunden inte har lagt beställningen inom 15 minuter så ska beställningsformuläret tömmas/rensas och kunden ska meddelas att denne är för långsam.

▶ 🤿 **Extra (frivilligt, påverkar ej betyget)**

Formulär för kunduppgifter

Formuläret där köparen fyller i sina uppgifter skall ha:

- Fält för:
  - Förnamn
  - Efternamn
  - Adress (gata)
  - Postnummer
  - Postort
  - Ev. portkod
  - Telefon (mobil)
  - E-postadress
  - Val för betalsätt: kort eller faktura
    - Om faktura valts som betalsätt ska ett formulärfält för svenskt personnummer visas. Även detta fält ska valideras innan formuläret går att skicka iväg, dvs. att man fyllt i korrekt personnummer.
    - Om kort väljs som betalsätt, visas fält för kortnummer, datum/år och CVC. Dessa behöver inte valideras!
  - Checkbox för godkännande av behandling av personuppgifter
  - Checkbox för beställning av nyhetsbrev (ska vara iklickad som default)
- Samtliga formulärfält ska valideras och formuläret/beställningen ska inte gå att skicka om det finns några fel
- Felen ska markeras och kommuniceras tydligt (t.ex. ej enbart med röd färg, tag i beaktande a11y)
- När formuläret är korrekt ifyllt ska Skicka-/Beställ-knappen aktiveras, innan det är den utgråad

- Det ska finnas en "Rensa beställning"-knapp som återställer samtliga formulärfält liksom eventuella beställda munkar/produkter (alltså antalet återställs till 0)
- Det ska finnas ett fält för att mata in en rabattkod.

Övrigt

- Det ska ges någon from av visuell feedback på när varukorgens totalsumma uppdateras. Med detta menas exempelvis någon visuell förändring, såsom en färg-skiftning, storleksskiftning, eller motsv.

Tekniska anmärkningar

- CSS:en ska byggas med hjälp av Sass (alt. Tailwind)
- Ni ska endast använda "Vanilla JavaScript", dvs. ni FÅR inte göra detta med hjälp av ett ramverk.
- HTML kan skrivas i HTML-dokumentet; noder behöver INTE skapas med JavaScript mer än om det är absolut nödvändigt.

Förslag på upplägg/arbetsgång

Vecka 1

- Skriv pseudokod för alla funktioner/krav INNAN DU BÖRJAR KODA. Du kan gott lägga 2-3 dagar på att bara skriva pseudokod, det vinner du på i tid senare, för att du har tänkt igenom hur du ska göra. Sällan blir det bra av att börja koda direkt så att fingrarna brinner. Rita flödesscheman för logik, gör wireframes, skriv upp funktionsnamn - ja, planera!
- Sätta upp kodbas/repo med nödvändiga filer.
- Se till att sidan publiceras online när ni gör uppdateringar.
- Ta fram design på sidan, minst wireframe men gärna en grundläggande design.
- Layouta upp HTML-strukturen & sätt upp CSS-strukturen.
- Diskutera programmets arkitektur och vilka metoder (funktioner) och variabler ni behöver för programmet.
- Skriv Issues i GitHub-repot som är tillräckligt små för att man ska kunna plocka en uppgift och tackla den.

Vecka 2

- Implementera logiken för beställningsformuläret (produktdelen)
- Implementera logiken för att beställa (kunduppgifter)

Vecka 3

- Animationer och effekter
- Strukturera om kod och flöde

Vecka 4

- Finlir och tid för att fixa de sista felen

Examinerade områden

Använd följande lista för att förstå vilka saker ni ska använda för att lösa uppgiften. Det behöver alltså inte vara svårare än det som gås igenom under kursens fyra första veckor.

- Logik & programflöde
- Kommentarer och självdokumenterande kod
- Hög kodkvalitet, konventioner
- Conditionals (if-satser)
- Event
- DOM-manipulation
- Funktioner
- Variabler
- Aritmetik
- Objekt
- Arrayer
- Timers
- Loopar
- Datum

Bedömningsmall

För IG

- Namngivningen av variabler, funktioner, attribut är namngivna som vilda västern - ingen konsekvens
- HTML, CSS och JavaScript är blandat utan någon ordentlig struktur och logik. T.ex. används `onclick` -event i HTML-koden, det finns `style` -attribut inskrivna direkt i HTML-koden, istället för att använda CSS-klasser.
- Sidan är otillgänglig; `<div>` har använts istället för `<button>` exempelvis.
- Koden har inte validerats.
- Det har skrivits lite eller ingen JavaScript. Det är uppenbart att kunskaperna inte sitter.
- Alla standarder kring CSS, HTML och tillgänglighet har ignorerats och glömts bort.
- Koden saknar indentering och struktur.
- Koden saknar kommentarer.
- Det finns ingen ansträngning att göra sig av med koduppprepning.
- Spaghettikod: det innebär att t.ex. funktioner ligger i funktioner som ligger i funktioner. Kan även appliceras på if-satser; har man 4+ if-satser nästlade i varandra, så bör man se över sin kodstruktur.
- Ni har lagt loopar i loopar och skapar t.ex. event eller funktioner flera gånger (oftast ett indenteringsproblem).
- Koden är full med anonyma funktioner.

För G

- Cirka 80 % av kravlistan är gjord
- Logiken i programmet/på sidan är logisk
- Sidan är responsiv och fungerar på olika enheter på det stora hela, någon enstaka miss här och var är OK
- Sidan är publicerad live
- Det finns en README med skrämdumpar på sidan som beskriver projektet och visar upp slutresultatet (så att ni har ett portfolio case), samt namn på personerna som bidragit till projektet. Tips - kolla readme.so
- HTML:en är validerad

- CSS:en är validerad
- CSS:en har inte lika stor vikt i detta projekt, så sitt inte för länge med designen

För VG

Utöver kraven för G...

- Samtliga punkter på kravlistan är implementerade, testade och validerat att det fungerar. OBS! Ej automatiskt testade, men ni ska ha klickat er igenom programmet och säkerställt att logiken fungerar som det ska.
- Koden är väldokumenterad och/eller självdokumenterande; korrekt namngivning, kommentarer, struktur i dokumenten
- Koden är strukturerad på ett logiskt sätt
- Koden följer de konventioner ni har satt upp för formatering (inga indenteringsmissar)
- Best practice följs i den mån det går/är känt
- Sidan är responsiv och fungerar på olika enheter felfritt. Anpassningar har gjorts för mörkt tema i någon mån (CSS räcker).
- HTML:en är semantisk och strukturen är logisk i förhållande till koden.
- När CSS/HTML/JS läses för sig är det intuitivt vad som är vad; namngivning och struktur går att "gissa lätt"
- Sidan är tillgänglig.
- Ni håller er till tidsramarna och gör prioriteringar utifrån tidsåtgång och viktighet
- Arbetet är självständigt (det betyder absolut inte att man inte ber om hjälp, men du ska också själv ha TÄNKT TILL och kan föra ett resonemang kring kod)
- Det är ingen swenglish i koden; variabler, attribut, funktioner, etc. är konsekvent namngivna

I README-filen har ni:

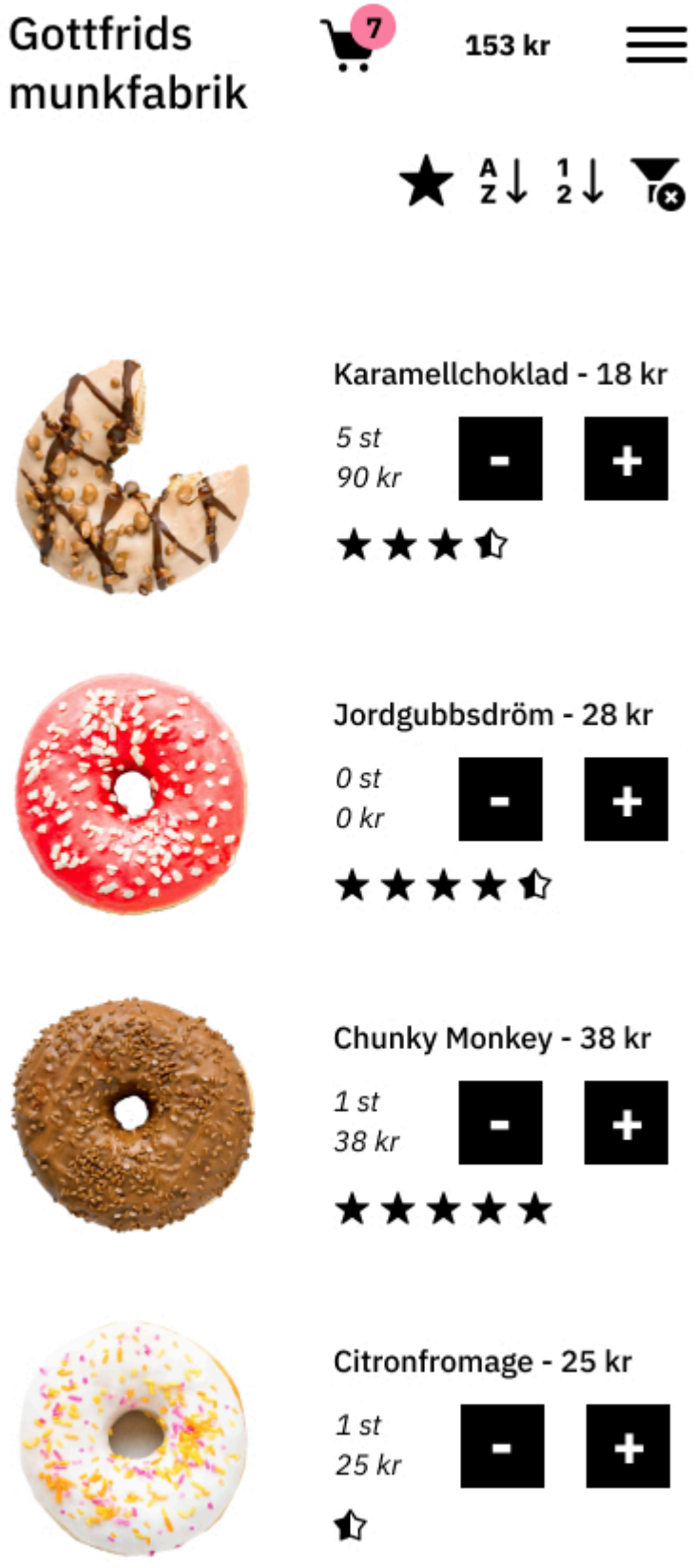
- Inkluderat bilder på valideringsrapporter
- Inkluderat bilder på Lighthouse-analys (eller motsv.)
- Länkat till live-versionen av projektet
- Lista teknikstack

Se readme.so - bra verktyg.

Referensbilder

OBS! Nedan skisser innehåller inte allt som finns i kravlistan, men för att ni ska få ett hum om vad som ska göras.

Beställningsknappar


Gottfrids
munkfabrik
7
153 kr
Karamellchoklad - 18 kr
5 st
90 kr
-
+
Jordgubbsdröm - 28 kr
0 st
0 kr
-
+
Chunky Monkey - 38 kr
1 st
38 kr
-
+
Citronfromage - 25 kr
1 st
25 kr
-
+

Munkshop

Varukorgssammanställning

Vanliga misstag

Glömt lärdomar från HTML-kursen

Kolla gärna på listan med Vanliga misstag från HTML-uppgiften (CV) och Vanliga misstag från CSS-uppgiften.

- Gör gärna en egen checklista på sådant du behöver kolla/komma ihåg! Så en "launch checklist" att stämma av mot. Jämför t.ex. denna och gör en egen version till dig själv.

Tillgänglighet

Glöm inte aria labels och liknande för att hålla koll på tillgängligheten, t.ex. om ni lägger ut information i HTML-dokumentet som bara är synligt "visuellt".

Exempelvis denna kod är inte tillgänglig för att visa ratings:

```html
<span class="fa fa-star checked"></span>
<span class="fa fa-star checked"></span>
<span class="fa fa-star checked"></span>

```

```
  <span class="fa fa-star-o"></span>
  <span class="fa fa-star-o"></span>
```

```html
<div aria-description="Rated 3 out of 5">
  <span class="fa fa-star checked"></span>
  <span class="fa fa-star checked"></span>
  <span class="fa fa-star checked"></span>
  <span class="fa fa-star-o"></span>
  <span class="fa fa-star-o"></span>
</div>
```

☞ Se också Death to icon fonts

.DS_Store (Mac)

Lägg in en fil som heter `.gitignore` och lägg in `.DS_Store` i den.

Komprimera CSS-filen

Glöm inte att sätta "compressed" som stil på CSS:en som genereras från Sass.

Gör en `.gitignore`

Lägg in följande i en fil som heter `.gitignore` :

```shell
.DS_Store
*.css.map
```

Undvik ääö samt mellanslag

Såväl i filnamn som variabelnamn som klassnamn.

Undvik också mellanslag i filnamn, t.ex. `my image.jpg` heter bättre om den heter `my_image.jpg` . Vissa webbläsare/servrar klarar inte av att visa bilder som innehåller mellanslag i filnamnet.

Datum

Tänk på var någonstans du skapar datumet för if-checkarna. Om du t.ex. högst upp i ditt program skapar en variabel såhär:

```js
const today = new Date();
```

Och sedan i en funktion gör såhär:

```js
function checkDiscount() {
  if (today.getDay() === 1) {
    // annan kod
  }
}
```

Om användaren låter webbläsarfönstret vara uppe t.ex. över ett dygn (så att fredag övergår till lördag exempelvis), så kommer man ändå att få fredagens rabatt då datumet inte genreras på nytt i funktionen, utan är samma som när webbsidan öppnades.

Du kan lösa det genom att generera ett nytt datum:

```js
function checkDiscount() {
  const today = new Date(); // tidpunkten för när funktionen anropas
  if (today.getDay() === 1) {
    // annan kod
  }
}
```

Rätt datatyp

När du skapar objekt, tänk på att använda rätt datatyp för att underlätta för dig själv framöver. T.ex. om du har ett pris, skriv hellre enbart siffran, då du förmodligen behöver göra någon ändring för själva siffran vid ett senare tillfälle (t.ex. rabatt). Jämför nedan:

```javascript
const myDonut = {
  price: '18 kr'
};
```

```
const myOtherDonut = {
  price: 18,
};

// 50 % discount
console.log(myDonut.price * 0.5); // ⇒ NaN
console.log(myDonut.price * 0.5); // ⇒ 9
```

Style:a ID istället för klass

Fortsätt använda klasser i stylingen och undvik att använda id:n. Kombinera hellre, exempelvis:

html

```
<div class="cart" id="cart"></div>
```

css

```
.cart {
  /* styling här */
}
```

script-taggen längst ner

Det är praxis att ha script-taggen precis innan avslutande body-tagg:

html

```
<script src="main.js"></script>
</body>
```

Då laddas scriptet in sist och sidan är tillgänglig/synlig för användaren. Det är också vanligt att script-taggen ligger precis innan body, vilket då gör det snabbt för en annan utvecklare att se var någonstans en inlänkning sker. Bra att följa praxis, alltså.

Om du av någon anledning inte kan ha det, lägg då till attributet `defer` så att scriptet inte blockerar inladdningen av resten av sidan:

html

```
<script src="main.js" defer></script>
```

Vi vill ha så snabba och responsiva sidor som möjligt.

Senast uppdaterad: 2024-11-13 18:39

Föregående sida
Git-fördjupning

Nästa sida
Det funkar inte...