Inlämningsuppgift: Kundprojekt

I denna inlämning ska ni utgå från en s.k. "brief" från en kund, och ta fram en webbplats utifrån en given design.

One-pager

Sidan ska vara en s.k. "onepager", dvs. du ska bara scrolla upp och ner på sidan för att komma till sida 2. Du ska *inte* skapa nya HTML-filer för respektive sida.

Brief/bakgrund

Du har precis fått jobb på webbyrån Justin Time AB. Byråns kund Kantarella Löwenskog har gett dig i uppdrag att ta fram en webbplats åt hennes nya företag, och webbplatsens design tas fram av kunden själv. 🤦‍♀️ Kantarella har precis blivit klar med sitt designarbete och tagit semester, liksom byråns projektledare Karin, så det är ju just typiskt. Du får helt enkelt improvisera på de bitar som är otydliga i briefen angående designen.

▶ **Brief - LÄS MIG**

Notera

- Länkarna i menyn ska inte leda någonstans. Du behöver inte kunna stänga menyn när du klickat på en länk (bara via menyknappen).
- Du behöver inte kunna stänga cookie-texten.
- Du behöver inte lägga in något JavaScript på sidan.
- Sidan ska vara en "onepager", du ska inte ha flera sidor. Dvs. första sidan täcker upp hela skärmens höjd på desktop. Scrollar du, dyker textsidan upp.
- När man klickar på "Explore menu" så ska sidan åka ner till menyn.

Bryt ner uppgiften i en arbetsgång, och strukturera upp informationen du har från briefen. Du *måste inte* läsa en grötig text gång på gång, men såhär kan det se ut när man får en överlämning från en kund.

Ibland saknas det information, t.ex. startsidan på mobilen. Titta på resurserna du har, och dra slutsatser av hur den kan tänkas se ut.

😌 Ta ett djupt andetag, gör en checklista & don't panic!

🚨 Du startar inlämningsuppgiften via Classroom-länken.

Introduktion till uppgiften

Intro till inlämningsuppgiften generellt

04:55

Intro till mall-repot och dess filer, samt GitHub Pages

05:44

Intro till att koda

Resurser

- Google Material Icon fonts
- Gimp är ett gratis och kraftfullt bildbehandlingsprogram.
- Tänk på vilka resurser du har! Snacka med varandra i klassen! Hur tänker du? Bilda studiegrupper och kom överens om en tid att t.ex. i 10 minuter prata om hur ni tänker kring olika moment. Kanske går ni igenom "brief-checklistan" och jämför med varandra? Vad har ni missat, vad tänkte någon annan på men inte du?
- Här finns ett color picker-verktyg som app. Du kan även googla på "color picker".

Menyn

För menyn kan du göra på två sätt, antingen med ren HTML/CSS eller JavaScript. Båda är godkända och det spelar ingen roll vilken du använder, och det spelar ingen roll för G/VG.

Alternativ 1: ☞ Skapa en hamburgarmeny i HTML/CSS

▶ **Alternativ 2**

Lärandemål & bedömning

Från kursplanen examineras samtliga moment i denna uppgift. Specifikt kompetens att:

- Använda semantisk HTML
- Validering av HTML och CSS
- Tillgänglighet
- Grundläggande CSS och Sass
- Planera design/layout
- Listor, rubriker, grid/flex, skapa menyer
- Responsivitet, media queries
- Bilder, inkl. tillgänglighet, SEO och best practice
- Länka till annan sektion på sidan
- Text-effekter
- Specialtypsnitt (fonter)
- Ikoner
- Positionering & stacking

- Cookies
- Formulär
- Namngivning, struktur och konsekvens i CSS

Inlämning

Du lämnar in genom att:

1. Ta skärmdumpar i *samtliga* webbläsare och i mobil-, tablet- och desktopvy (om du har Windows så behöver du inte ladda ner Safari).

- Webbläsare att testa i: Chrome, Edge, Firefox, Safari, ev. Brave
- Lägg skärmdumparna i en mapp som heter `screenshots` i ditt repo, och döp respektive till: `webbläsarnamn-upplösning.jpg`, t.ex. `edge-tablet.jpg`

2. Ta en skärmdump på validerad kod, och lägg in i repot i en mapp som heter `validering`.
3. Committa din kod till repot på GitHub.
4. Klistra in en länk till ditt repo på inlämningsuppgiften på itslearning.

Krav för G

- Du har använt semantiska taggar
- Sidan fungerar ungefär
- Det finns en del "designmissar" men designen följer sidan på det stora hela
- Responsiviteten sitter inte som en smäck, men funkar OK
- Du ska använda en extern CSS-fil
- Färger, bilder, typsnitt, etc. ska vara optimerad och inladdade på ett korrekt sätt (vad är en dekorativ bild, vad är en för stor/tung bild, behövs hela typsnittsfamiljen, osv.)
- Det finns lite skavanker här och där, både i utseende och i koden

---

Krav för VG

- Utöver kraven för G...

- Korrekt användning av semantiska taggar
- Sidan validerad utan fel och varningar
- Korrekt indentering
- Korrekta element har använts utifrån tillgänglighetsperspektiv
- Responsiviteten är korrekt implementerad
- Använt Sass
- Ha löst problemen i uppgiften på ett optimalt sätt och utan onödiga "hack".
- Se till att toningar och övergångar fungerar snyggt (du behöver/ska inte använda JavaScript).
- Ha optimerat resurser och laddningstid på sajten.
- Konsekvent kod: namngivning, indentering, struktur.

Vanliga misstag

HTML-uppgiften

Du har glömt bort att kolla listan "Vanliga misstag" från CV-uppgiften. Dessa gäller fortfarande 😉 Du hittar den här.

Inkonsekvent formatering i dokumentet

Tänk på mellanslag, radbrytningar och namngivning. Exempelvis om du skriver alla selektorer såhär:

CSS

```
.main-menu {}

.super-heading {}
```

Dvs. med bindestreck och mellanslag mellan selektornamnet och måsvingen.

Så ska inte en annan selektor se ut såhär (med understreck/"camel case" och utan mellanslag/radbrytning):

```css
.main_menu{}
.superHeading{}
```

Välj en stil och håll dig till den. Samma gäller språk och kolon!

```scss
.some-selector{
  color:red; // inget mellanslag efter kolon
  font-weight: bold; // mellanslag efter kolon
}
.en-selektor {} // svenska helt plötsligt?!
```

Enhet på 0

Du behöver inte sätta en enhet efter ett värde, om dess värde är noll. Dvs. du kan utelämna `px` :

```css
main {
    margin-bottom: 0;
}
```

Citattecken runt `url`

Sätt gärna citattecken runt url:en på bilder för att vara "på den säkra sidan".

```css
body {
    background-image: url('adressen/till/bilden.jpg');
}
```

Versaler

Gör versaler i CSS. Blir "skrikigt" i HTML annars. Dvs. skriv inte `<h1>TAHINI</h1>` utan `<h1 class="uppercase">Tahini</h1>` .

Formatera (indentera) koden automatiskt i VSC

Se instruktioner här .

Import av fonter i CSS

Du ska absolut inte importera fonter i CSS, eller andra CSS-filer för den delen heller. Observera här att det är *skillnad på* import i Sass, och import i CSS! Teknisk förklaring finns här .

Menyknappen saknar textinnehåll

Om din menyknapp ser ut såhär:

```html
<button>
    <span></span>
    <span></span>
    <span></span>
</button>
```

Tänk på att göra den tillgänglig för "assistive devices" genom att sätta ett `aria-label` på den, se info här .

.DS_Store

Om du har Mac, skapa en fil som heter `.gitignore` (med punkten framför filnamnet) och i den filen skriver du:

```gitignore
.DS_Store
```

Så får du inte med denna "dolda systemfil" i versionshanteringen.

Om du redan har fått med `.DS_Store` i versionshantering (dvs. den finns på git), så kan du ta bort den genom att skriva följande i terminalen:

```shell
git rm --cached '*.DS_Store'
git commit -m "Remove .DS_Store files"
git push origin main --force
```

Det som händer ovan är att den tar bort (`rm` = remove) den cachade (sparade) filen som heter .DS_Store, och efter det görs en commit med meddelanden (`-m`) "Remove .DS_Store files". Slutligen pushar du ändringen till servern (i detta fallet GitHub) till branchen `main` (som är default-branchen), och kommandot `--force` gör att versionshistoriken skrivs tvingas fram, om det skulle finnas en konflikt.

Du kan även lägga till `.map` -filen som genereras:

gitignore

```
*.css.map
```

Använd kommentarer

Du kan med fördel använda block-kommentarer i Sass för att dela upp din kod och göra det lättare att "skumma igenom" filen, exempelvis:

scss

```
/*
————————————————————————————————————————
———————————————————— HEADER ————————————————————
————————————————————————————————————————
 */
```

Eller är du riktigt kreativ kan du köra med någon ASCII-version :

scss

```
// ————————————————8←————————————[ HEADER ]————————————————————
```

Använd ej ID som selektor

Även om du har nytta av ID t.ex. för ankarlänkar, använd alltid klasser som selektor för CSS, dvs. Kom ihåg att ID är som ett personnummer, du kan endast använda det en gång. Med klasser skapar du mer återanvändbar kod (och vi gör ju saker av vana, så lägg dig till med rätt vana).

html

```
<div id="section1" class="section1"></div>
```

Font-storlek i px

Du ska aldrig sätta font-storlek i `px`, utan använda `rem` eller `em` för att din sida ska vara tillgänglig och skalbar. Läs teknisk förklaring här

Tips, om du använder Sass och tycker det är svårt att tänka `rem` eller `em`, gör en hjälpfunktion:

```scss
@function rem($size) {
   @return calc($size / 16) * 1rem;
}

h1 {
   font-size: rem(24); // Konverteras via funktionen till (24 / 16) ⇒ 1.
}
```

Sökmotoroptimering

Tänk på hur sidan ser ut utan CSS. Exempelvis "Tough Cookie"-texten bör vara sidans huvudtitel (vilken tagg ska du använda för det?) och hur du radbryter den för designen. Om du gör exempelvis något av följande:

```html
<div>T<br>O<br>U<br>G<br>H<br><br>C<br>O<br>O<br>K<br>I<br>E</div>
<div>t o u g h   c o o k i e</div>
```

Så kommer sökmotor-crawlern att se sidan utan CSS (du kan simulera det i Firefox via View —> Page Style —> No Style). Även om du visuellt kan uppfatta texten, så har en "dum robot" ingen möjlighet att tolka texten. Se exempel i videon nedan:

00:13

Hotlinking av bilder

Vad är hotlinking?

Tänk också på att när du länkar en bild från en annan källa, så har du ingen kontroll över bilder. I värsta fall kan bildinnehållet ersättas med någonting olämpligt.

Därutöver finns en rad andra problem med hotlinking, se t.ex. här .

Compressed i Sass

Glöm inte att ändra tillbaka till "compressed"-formatet i Sass innan slutinlämning.

Google Fonts

Om du kopierar Google Fonts flera gånger, så riskerar du att få `preconnect` -taggarna dubbelt:

```html
<link rel="preconnect" href="https://fonts.googleapis.com" />
<link rel="preconnect" href="https://fonts.gstatic.com" crossorigin />
```

Rensa bort ev. dubbletter i din HTML-fil. Kombinera också gärna Google Fonts-länken så att du har *alla* fonter i en och samma. Så istället för:

```html
<link
      href="https://fonts.googleapis.com/css2?family=Jockey+One&display=s
      rel="stylesheet"
    />
<link
    href="https://fonts.googleapis.com/css2?family=Roboto+Condensed&displ
    rel="stylesheet"
```

ska du ha:

```html
<link href="https://fonts.googleapis.com/css2?family=Jockey+One&family=Ro
```

Detta åstadkommer du genom att välja *alla fonter direkt* som du vill använda, så får du en kombinerad länk av Google, istället för att välja enstaka fonter flera gånger. Om ett typsnitt erbjuder flera tjocklekar, välj helst också bara dom som används så blir sidan mer snabbladdad än om alla tillgängliga tjocklekar laddas in.

Och när alla fonter laddas in gemensamt i en länk, så blir sidan snabbare.

Sidtitel h1

Sidans viktigaste titel (t.ex. ett företagsnamn) bör sättas som en `<h1>` -tagg.

Backup-font

Ha alltid en backup-font när du använder special-fonter, så att om det skulle bli något strul, så ser det *hyfsat* likt ut i alla fall. Bra backup-fonter är de generiska font-klasserna (`sans-serif` , `serif` , `monospace`).

```css
h2 {
  font-family: 'Jockey One', sans-serif;
}
```

Height istället för min-height

Använd gärna `min-height` som måttenhet som "standard" när du behöver sätta höjd på någonting. Om upplösningen är "låg" (på höjden, alltså inte dålig i pixlar) och innehållet inte får plats på skärmen, då stretchar "containern" för att få plats med innehållet. Säg att man t.ex. tar upp inspektorn eller surfar i landskapsläge på en mobil - då kan innehållet flöda över och det blir fult.

Namngivning klasser

När du namnger saker i HTML/CSS, fundera på om designens innehåll kan ändras. Exempelvis kanske "kunden" väljer att byta ut en bild på en tomat till en potatis. Har du då döpt klassen till `.tomato` så blir det mindre logiskt när bilden är en potatis. Då kanske ett bättre klassnamn hade varit `.food-image` eller liknande. Försök att tänka generiskt och "framtidssäkra".

Validera

Glöm inte att validera din HTML . Även om validatorn inte är perfekt så hittar den många fel! Du kan klistra in länken till din live-sida, eller klistra in koden direkt.

Checklista saknas

Du har inte skrivit en checklista till dig själv på saker du ska dubbelkolla i ditt dokument! Viktigast är t.ex.:

- Tillgänglighet och tillgänglighetsanalys
- Validering
- Formatering på koden

srcset

För plus i kanten (och om du kan bildbehandling sedan innan), använd gärna `srcset` på bilder.

Senast uppdaterad: 2024-10-30 06:26

Föregående sida
Inl. 1: CV

Nästa sida
Kundprojekt - Tips & lösningar