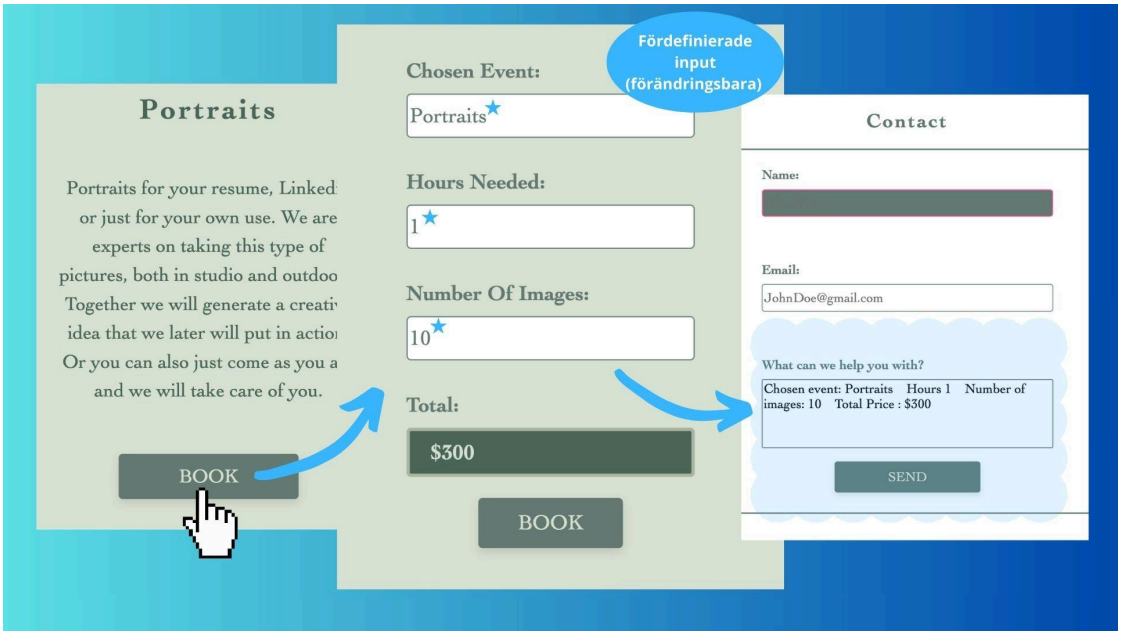
Freya Photos

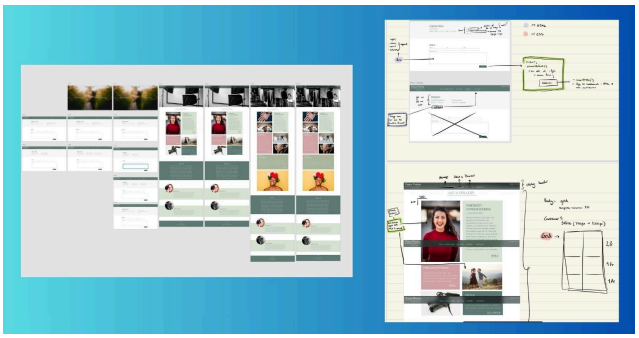
Malin Jakob

Detta gesällprov är en hemsida för den fiktiva fotografstudion Freya Photos. Huvudsakligen har hemsidan två viktiga funktioner, det första är bilderna som förmedlar deras verksamhet och den andra är hur kunden, med lätta och effektiva knapptryck, själv kan få fram priset för önskad tjänst. Genom fördefinierade priser samt interaktiva inputs kan kunden lätt ta sig från idé till köp av tjänst på några knapptryck, denna översikten är nyckeln.



Framtagande

1. Händelseförloppet för denna webbplats började för ca 5 år sedan med en Gränssnitt Kurs där jag arbetade fram en layout för denna webbplats. För att inte behöva uppfinna hjulet en gång till tog jag ideen från arbetet och började skissa om den med ny kunskap samt ur ett perspektiv med HTML, CSS och JavaScripts funktionalitet i fokus.



2. Efter att ha spånat hur jag ville att själva sidan skulle se ut och vilken funktionalitet jag skulle implementera började jag att arbeta fram HTML strukturen. Jag strukturerade fram alla sidor som skulle finnas på webbplatsen och därefter strukturen på headern samt footern, eftersom de kommer att vara konstanta över hela webbplatsen. Efter att ha skapat de konstanta bitarna började jag arbeta sida för sida, med HTML parallellt med CSS. All design utgår från en mobil vy med media queries för större skärmar.

3. Eftersom det är en webbplats som ska framhäva sin tjänst kring fotografin så arbetade jag en del i **GIMP** för att justera format, sammanflätning och bitdjup på bilderna innan jag tillämpade dem i projektet.

4. När skalet på hemsidan var på plats började jag arbeta med funktionaliteten på webbplatsen. Fokuset låg först på att skapa funktionalitet för att använda förladdade bilder i det skiftande fotogalleriet som man bemöter på förstasidan. Denna funktionalitet ansåg jag vara viktig för att inte få några fördröjningar. I home.html strukturerade jag upp en container som innehåller ett **img element** med ett **ID** som jag senare i mitt JavaScript program hämtar för att skapa ett fotogalleri med skiftande bilder. Jag skapade en funktion som går igenom listan med förladdade bilder och satte ett intervall på 2700 millisekunder så att bilden skiftas till nästa index i listan. När den har gått igenom alla bilder så börjar den om, så att bilderna ändras hela tiden utan något stopp. Denna funktionalitet bygger på en tidigare uppgift i kursen och gick smidigt att implementera då strukturen för funktionaliteten hade jag redan arbetat fram i ett tidigare skede.

```html
<div class="homeImg">
  <img src="images/gallery/gallery1.jpg" id="currentImage" /
</div>
```

```javascript
const images = [
  'images/gallery/gallery1.jpg',
  'images/gallery/gallery2.jpg',
  'images/gallery/gallery3.jpg',
  'images/gallery/gallery4.jpg',
  'images/gallery/gallery5.jpg',
  'images/gallery/gallery6.jpg',
  'images/gallery/gallery7.jpg',
  'images/gallery/gallery8.jpg',
  'images/gallery/gallery9.jpg',
  'images/gallery/gallery10.jpg'
]
let imgIndex = 0

function changeImage () {
  imgIndex++
  let currentImage
  currentImage = imgIndex % images.length
  $('#currentImage').attr('src', images[currentImage])
}

function imageTimer () {
  let changeImageTimer
  changeImageTimer = setInterval(changeImage, 2700)
}

$(document).ready(function () {
  preloadImages()
  imageTimer()
})
```

5. Jag arbetade vidare med den andra och en av de viktigaste funktionaliteter för sidan vilket är *pris genereringen*. Detta tog en hel del tid att få ihop då jag ville att användaren på ett simpelt sätt skulle kunna ta sig från förstasidan till pris-sidan och sedan till kontaktsidan med all information de behövde. Jag skulle därmed skapa en funktionalitet som kunde bidra till att med endast med några knapptryck skulle användaren kunna bestämma vilken tjänst de ville ha, hur det vill ha det (eller ta förvalda inställningar) och sedan kunna se totalpris för utvalda värden för att tillsist kunna lägga till en förfrågan för denna beställning. Jag började med att forma själva "uträkningsfunktionen". Till att börja med tillämpade jag några knappar (priceBox) med de olika kategorierna som rubrik. Detta gjorde jag för att skapa ett simpelt sätt för användaren att få en översikt över kategorierna och samtidigt för att få ut de fördefinierade värden tillsammans med den totala kostnaden för den valda tjänsten. Det är denna knapp som får uträkningsfunktionen i JavaScript att rulla. I HTML strukturen skapade jag presentationen av värdena med hjälp av ett **formulär,** vilket även skapade dimensionen där kunden själv kan ändra värdena och samtidigt se hur den totala kostnaden ändras dynamiskt. Alla värdena går att förändra, alla förutom den totala kostnaden som jag skapade genom att göra en **div** med ett **span** element med ett **ID** som jag senare kunde använda för att manipulera in totalen i det området utifrån värdet beräknat av funktionen.

```
<div id="total">Total: <span id="totalAmount">0 €</span> </div>
```

6. För att kunna utvinna information från knapparna tillämpade jag **data** till event boxarna i price.html för att kunna utvinna denna information senare i JavaScript funktionerna dynamiskt. Jag genererade fram ett första koncept av funktionaliteten där jag tog in värden från boxarna (**getElementById** & **querySelector**) och la till en **eventListener** som väntade på ett 'klick' på knapparna för att kalla på funktionen som räknar

```
+ function initPriceBoxes () {
+   const eventInput = document.getElementById('event')
+   const hoursInput = document.getElementById('hours')
+   const imagesInput = document.getElementById('numberOfImages')
+   const totalAmount = document.getElementById('totalAmount')
+
+   let basePrice = 0
+
+   function getTotal () {
+     const hours = parseInt(hoursInput.value) || 0
+     const images = parseInt(imagesInput.value) || 0
+     const total = basePrice + hours * 100 + images * 5
+     totalAmount.value = `$${total}`
+   }
+
+   document.querySelectorAll('#priceBox .box').forEach(box => {
+     box.addEventListener('click', () => {
+       const eventName = box.getAttribute('data-event')
+       const defaultHours = box.getAttribute('data-hours')
+       const defaultImages = box.getAttribute('data-images')
+       basePrice = parseInt(box.getAttribute('data-price'))
+
+       eventInput.value = eventName
+       hoursInput.value = defaultHours
+       imagesInput.value = defaultImages
+
+       getTotal()
+     })
+   })
+   hoursInput.addEventListener('input', getTotal)
+   imagesInput.addEventListener('input', getTotal)
```

ihop den totala kostnaden. Om användaren sedan ändrade input själv satte jag en **eventListener** för att fånga upp det för att räkna om totalen med användarens input istället. I denna iterationen fick jag inte till att det även skulle kunna gå att få informationen från första sidan och även att information skulle följa med till kontakt-sidan utan fram tills nu kunde man bara se funktionaliteten när man ändrade event på pris-sidan.

7. Nästa iteration var att få till funktionaliteten av att manipulera de olika värdena till HTML på kontaktsidan. Jag gjorde detta genom att läsa in URL:en och sedan söka efter **query-strings** som skickas med när man trycker på “boka”-knappen på pris-sidan. Genom att söka efter dessa värden kunde jag sedan lägga till en funktionaliteten som gör att när man kommer till kontakt-sidan, på grund av användning av **innerText,** kommer värdena att visas som vanlig text tillsammans med en fördefinierad text i textrutan. Kunden kan även ändra alla dessa parametrar på kontaktsidan eller gå tillbaka till pris-sidan och börja om på grund av att strukturen på HTML-sidan är ett **formulär**. Här stötte jag på ett problem, vilket var att den totala summa inte skickades med till kontaktsidan utan visade värdet **null**. Jag insåg att problemet var att elementet för den totala kostnaden inte var i rätt format. Problemet löstes genom att göra **elementet** för totala kostnaden till ett **input field** istället för en div och lägga till **readonly** i HTML-strukturen så att användaren inte själv kunde ändra totalen. På detta sätt kunde jag få med den totala kostnaden i query-stringen som jag sedan kunde fånga upp för att visa den information på kontakt-sidan.

```
function getQueryString () {
  const queryString = window.location.search
  const searchQuery = new URLSearchParams(queryString)
  const eventName = searchQuery.get('event')
  const hours = searchQuery.get('hours')
  const numImg =  searchQuery.get('numberOfImages')
  const total =  searchQuery.get('totalAmount')

  const message = document.getElementById('message')
  if (!eventName) {
    message.innerText = ''
  }
  else{
    message.innerText = `Chosen event: ${eventName}
    Hours ${hours}
    Number of images: ${numImg}
    Total Price : ${total}  `
  }
}

$(document).ready(function () {
  getQueryString()
})
```

```
<label for="totalAmount">Total: </label>
<input readonly id="totalAmount" name="totalAmount" />
```

8. Sista iterationen blev att lägga till funktionaliteten av att användaren kunde från kategorierna på förstasidan ta sig, via “boka”- knappen, till att boka med givna parametrar med endast några knapptryck. Nu var det mycket som behövdes ändras och “snyggas” till. Jag började med att lägga till värden på HTML-sidan för de olika knapparna (kopplat till den kategorin) som kunde skickas med som **query-strings.** På detta sätt kunde jag sedan ta upp dem i min JavaScript-funktion för att veta vilket event användaren hade valt. Jag tillämpade en sökning av event namnet och med den returnerade informationen kunde jag nu koppla samman de tidigare skapta funktionerna. Flödet blev att vi nu kan få fram de värden kopplade till de olika kategorierna och skicka med dessa till vår funktion som räknar ut totalen för den valda tjänsten. Jag hade även glömt att ta in värdet för baspriset för de olika eventen i mina tidigare iterationer så jag tillämpade även det i funktionen. Efter många iterationer senare kan man nu ta sig från A till B till C på bara några knapptryck.

```
function initPriceBoxes () {
  const eventInput = document.getElementById('event')
  const hoursInput = document.getElementById('hours')
  const imagesInput = document.getElementById('numberOfImages')
  const totalAmount = document.getElementById('totalAmount')

  document.querySelectorAll('#priceBox .box').forEach(box => {
    box.addEventListener('click', () =>
      selectBox(box, eventInput, hoursInput, imagesInput, totalAmount)
    )
  })
  hoursInput.addEventListener('input', getTotal)
  imagesInput.addEventListener('input', getTotal)
}

function getTotal (hoursInput, imagesInput, basePrice) {
  const hours = parseInt(hoursInput.value) || 0
  const images = parseInt(imagesInput.value) || 0
  const total = basePrice + hours * 100 + images * 5
  return total
}

function selectBox (box, eventInput, hoursInput, imagesInput, totalAmount) {
  const eventName = box.getAttribute('data-event')
  const defaultHours = box.getAttribute('data-hours')
  const defaultImages = box.getAttribute('data-images')
  const basePrice = parseInt(box.getAttribute('data-price'))

  eventInput.value = eventName
  hoursInput.value = defaultHours
  imagesInput.value = defaultImages

  const total = getTotal(hoursInput, imagesInput, basePrice)
  totalAmount.value = `$${total}`
}
```

```
function initSelectedBox () {
  const eventName = getEventName()

  if (!eventName) {
    return
  }

  const box = document.getElementById(eventName)
  if (box) {
    const eventInput = document.getElementById('event')
    const hoursInput = document.getElementById('hours')
    const imagesInput = document.getElementById('numberOfImages')
    const totalAmount = document.getElementById('totalAmount')
    selectBox(box, eventInput, hoursInput, imagesInput, totalAmount)
  }
}

function getEventName () {
  const queryString = window.location.search
  const searchQuery = new URLSearchParams(queryString)
  const eventName = searchQuery.get('event')
  return eventName
}

$(document).ready(function () {
  initPriceBoxes()
  initSelectedBox()
})
```

9. Efter att funktionaliteten kom på plats började jag arbeta med **responsiv design**. Jag valde att lägga mina brytpunkter vid 780 px samt 1024px. Allt utgår från mobil vy, sedan blir text och rubriker successivt större, angivet i procentform. Gjorde även så att bilderna får ta mindre yta av skärmen då de annars blev för “inzoomat”. Kolumnerna i designen

ökar med en för varje brytpunkt och text där det inte gick att dela upp i fler kolumner (för att det blev oläsbart) minskade jag bredden i relation till sidan för att användaren inte ska behöva vrida huvudet för att kunna läsa texten när de läser på större skärmar. Arbetade även fram att de olika vyerna skulle ta olika stor del av skärmen, där vi började från 100% till att gå ner mot 90,80,70% av skärmen yta. Detta gör så att delarna i innehållet håller sig relativt till skärmstorleken och inte stretchas ut för mycket vilket gör användarupplevelsen lite mer koncentrerad till mitten av skärmen.

10. Till sist la jag till en till sida där jag ville skicka användaren till för att ge feedback om att deras förfrågan av tjänst är mottagen. Eftersom jag inte har något verkligt ställe denna förfrågan går så gjorde jag så att när man trycker på “skicka’”- knappen blir man skickad till en ny sida/tab där ett meddelande visas. På den nya sidan förmedlas det att mottagning av förfrågan var framgångsrik.

Form

contact.html & contact.js:

Programmet för **contact.html** sidan ligger i filen **contact.js** och dess huvuduppgift är att ta in query-strings och presentera informationen på ett läsbart sätt i HTML:en. Detta görs genom att först läsa in informationen från URL:en, formatera informationen och till sist manipulera HTML:en.

Pseudokod:

```
FUNCTION getQueryString ()
    GET event,hours, numberOfImages, totalAmount from query string
    IF message-element does not exist THEN
        do nothing
    ELSE
        format information
```

set message innertext to formatted information

END FUNCTION

when document is ready, call getQueryString function

---

prices.html & prices.js:

Bilder på centrala kod-avsnitt

```
/>
<label for="totalAmount">Total: </label>
<input readonly id="totalAmount" name="totalAmount" />
<input
  class="bookButton"
  value="BOOK"
  type="submit"
  name="submit"
  id="submitButton"
```

```
<h2>Price List</h2>
<div id="priceBox">
  <div
    id="portrait"
    class="box"
    data-event="Portraits"
    data-hours="1"
    data-images="10"
    data-price="150"
  >
```

```
function initPriceBoxes () {
  const eventInput = document.getElementById('event')
  const hoursInput = document.getElementById('hours')
  const imagesInput = document.getElementById('numberOfImages')
  const totalAmount = document.getElementById('totalAmount')

  document.querySelectorAll('#priceBox .box').forEach(box => {
    box.addEventListener('click', () =>
      selectBox(box, eventInput, hoursInput, imagesInput, totalAmount)
    )
  })
  hoursInput.addEventListener('input', getTotal)
  imagesInput.addEventListener('input', getTotal)
}

function getTotal (hoursInput, imagesInput, basePrice) {
  const hours = parseInt(hoursInput.value) || 0
  const images = parseInt(imagesInput.value) || 0
  const total = basePrice + hours * 100 + images * 5
  return total
}

function selectBox (box, eventInput, hoursInput, imagesInput, totalAmount) {
  const eventName = box.getAttribute('data-event')
  const defaultHours = box.getAttribute('data-hours')
  const defaultImages = box.getAttribute('data-images')
  const basePrice = parseInt(box.getAttribute('data-price'))

  eventInput.value = eventName
  hoursInput.value = defaultHours
  imagesInput.value = defaultImages

  const total = getTotal(hoursInput, imagesInput, basePrice)
  totalAmount.value = `$${total}`
}
```

```
function initSelectedBox () {
  const eventName = getEventName()

  if (!eventName) {
    return
  }

  const box = document.getElementById(eventName)
  if (box) {
    const eventInput = document.getElementById('event')
    const hoursInput = document.getElementById('hours')
    const imagesInput = document.getElementById('numberOfImages')
    const totalAmount = document.getElementById('totalAmount')
    selectBox(box, eventInput, hoursInput, imagesInput, totalAmount)
  }
}

function getEventName () {
  const queryString = window.location.search
  const searchQuery = new URLSearchParams(queryString)
  const eventName = searchQuery.get('event')
  return eventName
}

$(document).ready(function () {
  initPriceBoxes()
  initSelectedBox()
})
```

Programmet för **prices.html** sidan ligger i filen **prices.js** och dess huvuduppgift är att hantera och beräkna kostnader baserat på användarens val i formuläret. Detta görs genom att först registrera användarens val av event och antalet timmar samt bilder, för att sedan beräkna den

totala kostnaden. Programmet manipulerar också HTML-element för att visa den beräknade kostnaden på ett tydligt och lättförståeligt sätt

---

Pseudokod:

```
FUNCTION initPriceBoxes ()
    GET eventInput, hoursInput, imagesInput, totalAmount from DOM elements
    FOR each box in priceBox
        ADD event listener for click, to call selectBox with current box and inputs
    ADD event listener for input on hoursInput to call getTotal with required variables
    ADD event listener for input on imagesInput to call getTotal with required variables
    END FUNCTION

FUNCTION getTotal (hoursInput, imagesInput, totalAmount, basePrice)
    SET hours to integer value of hoursInput or 0
    SET images to integer value of imagesInput or 0
    CALCULATE total as basePrice + (hours * 100) + (images * 5)
    UPDATE previousBasePrice to basePrice
    SET totalAmount to formatted total string
END FUNCTION

FUNCTION selectBox (box, eventInput, hoursInput, imagesInput, totalAmount)
    GET eventName, defaultHours, defaultImages, basePrice from data attributes of box
    PARSE basePrice to integer
    SET eventInput, hoursInput, imagesInput to corresponding variables
    CALL getTotal(hoursInput, imagesInput, totalAmount, basePrice)
END FUNCTION
FUNCTION initSelectedBox ()
    SET eventName to getEventName()
    IF eventName does not exist THEN
```

```
            RETURN
        END IF
        SET box to DOM element with ID equal to eventName
        IF box exists THEN
            GET eventInput, hoursInput, imagesInput, totalAmount from DOM elements
            CALL selectBox(box, inputs)
        END IF
END FUNCTION

FUNCTION getEventName ()
        GET query string from window location
        SET searchQuery to parsed URLSearchParams from query string
        RETURN event name from searchQuery
END FUNCTION

when document is ready, call initPriceBoxes and initSelectedBox
```

---

home.html & home.js:

Programmet för **home.html** sidan ligger i filen **home.js** och dess huvuduppgift är att förladda bilder och visa en bild som byts ut var tredje sekund. Detta görs genom att först ladda bilderna i minnet i browsern, för att sedan byta mellan dem utifrån en lista.

Pseudokod:

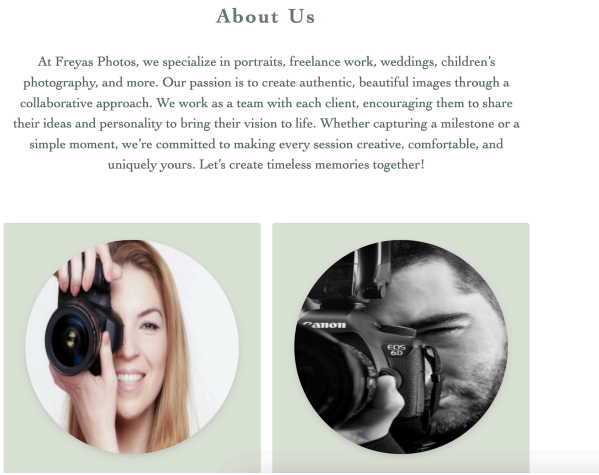
```
FUNCTION preloadImages ()
        IF document contains images THEN
            DECLARE image variables for each gallery image
            SET source for each image
        END FUNCTION

DECLARE array of image sources
```

```
SET imgIndex to 0

FUNCTION changeImage ()
        INCREMENT imgIndex
        SET currentImage to imgIndex modulo length of images array
        UPDATE current imagesource in HTML to currentImage
END FUNCTION

FUNCTION imageTimer ()
        SET changeImageTimer to call changeImage every 2700 milliseconds
END FUNCTION

when document is ready, call preloadImages and imageTimer functions
```

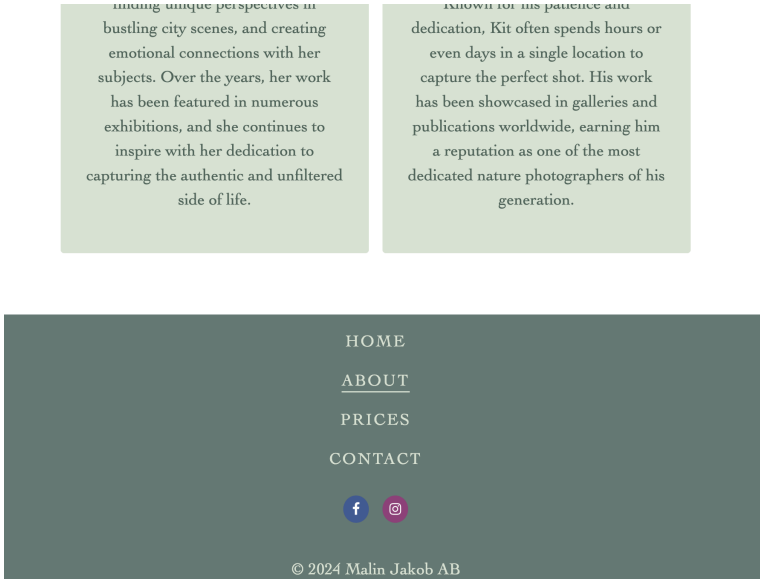
Funktion

När allt är i rullning är detta det första man ser. Det är en förstasida med bilder som ändras var tredje sekund med en konstant navigation bar högst upp på sidan. Bilderna i galleriet som visas

sker utan några fördröjningar eftersom tillsammans med JavaScript förladdas bilderna och vilket ger ett sömfritt bild skifte i galleriet. Detta är en viktig funktionalitet för att ge kunden ett bra första intryck av själva produkten webbplatsen vill sälja men även ge förtroende för webbplatsen i sin helhet. I navvigationsbaren kan användaren lätt se vilken sida den befinner sig på samt om man tar muspekaren över de olika rubrikerna kan man se att de är klickbara. Om användaren trycker på de olika länkarna skickas de vidare till den relativa länken. Om exempelvis en användaren trycker på “About Us” skickas dem dit och möts av denna vy;

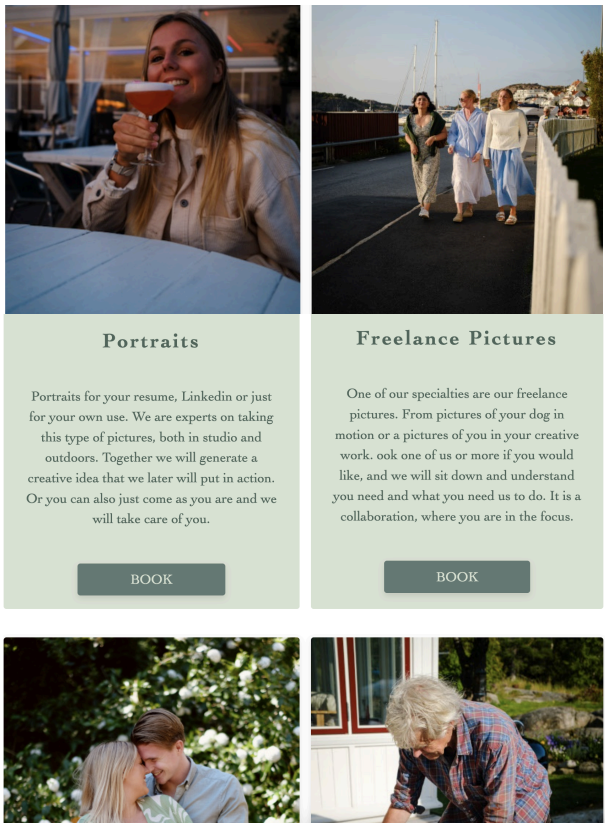


Här bemöter kunden information om företaget samt kring de anställda som jobbar där. Om man scrollar vidare bemöts man av text kring personalen som är inramade i olika behållare tillsammans med en bild för skapa en större helhet kring personen som beskrivs. När beskrivningen är över , eller generellt på alla sidor där huvudinnehållet (main) tar slut, möts

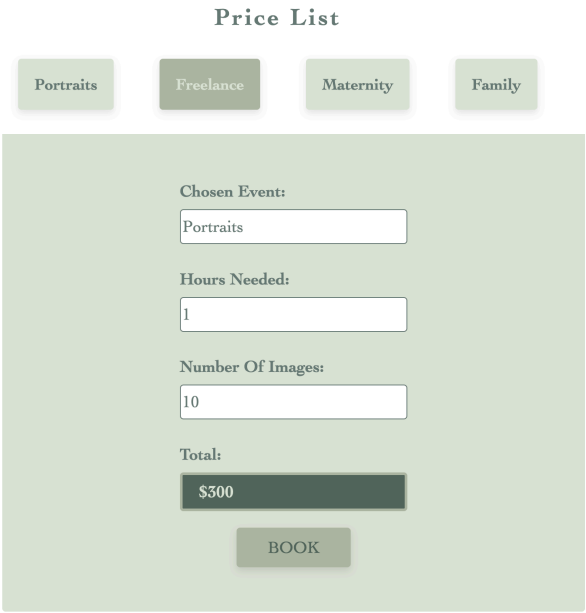


användaren av en footer som är konstant på hela webbplatsen. Här kan användaren lätt navigera sig vidare till andra sidor på webbplatsen samt se vilken sida de befinner sig på i nuläget. De kan även trycka på sociala media ikoner och skickas då vidare till företagets FaceBook-konto och Instagram-konto. Till sist möts man av en copy right som

indikerar vem som har skapat webbplatsen och vilket år.



Eftersom hemsidans syfte är att sälja fotografierna, eller konceptet av fotografierna, bemöter vi fler bilder längre ner kopplat till en kategori. Kategorierna skapar en överblick av de möjliga tjänsterna som går att boka. Samtidigt skapar uppdelningen möjligheten till funktionaliteten som gör att kunden lätt kan beställa den tjänst man vill ha med möjlighet av egna modifikationer, idéer eller med förutbestämda val.
Genom att enkelt trycka på “boka”-knappen på den tjänst man vill ha kommer man direkt vidare till nästa steg, pris-sidan.



När kunden kommer till pris-sidan visas värdena för de fördefinierade valen i fomruläret. Det visas även knappar ovanför formuläret med olika kategorier som man kan trycka på. Om användaren exempelvis klickar på “Freelance”-knappen (som muspekaren är över i denna bild) byts dynamiskt värdena i rutorna "Chosen Event”, “Hours Needed” och “Number Of Images”. Detta ger kunden ett lätt sätt att byta till en annan kategori utan att lämna sidan. Efter att ha gett kunden en överblick kring de förvalda värdena samt den totala kostnaden kan kunden även lätt ändra valen i de olika fälten i

formuläret. När en ändring sker tar programmet direkt upp detta och ändrar totalsumman dynamiskt utifrån värdena som ges. På detta sätt ger vi kunden alla möjligheter att labba med hur de vill ha sin tjänst samt ger dem en enkel överblick över hur kostnaden ändras.

Det slutar inte här för att förenkla för kunden utan med ett knapptryck på “boka”-knappen kommer de val kunden själv har angett eller förbestämda valen att hamna som förskrivet meddelande på kontaktsidan. Detta innebär att kunden kan (i teorin i alla fall) trycka på tre knappar, samt lite tillägg som namn och email, för att få iväg en förfrågan för önskad tjänst. Snabbt och enkelt! På kontaktsidan måste användaren skriva in namn och email för att få skicka iväg förfrågan samt visas tydliga visuella meddelanden över vad som krävs av användaren. Utöver detta finns det även indikationer över vart de befinner sig i formuläret.

För att knyta ihop säcken avslutas hela denna processen med att kunden trycker på “skicka”-knappen och där de skickas iväg till en ny sida och möts av ett “ det lyckades att skickas” meddelande för att lugna kunden med feedback om att allt har gått som det ska och nu är det bara invänta svar från företaget. Webbplatsen lyckas tillsammans med bilder och ytterst få knapptryck leverera en plattform med väl anpassad funktionalitet för tjänsten företagets vill förmedla, vilket ger värde för både kund och Freyas Photos.