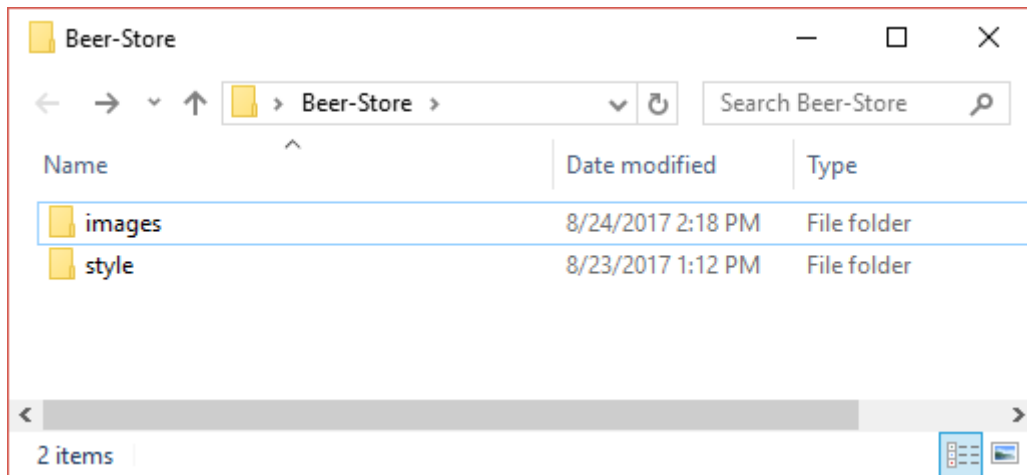


Exercise: Web Fundamentals Introduction – Beer Store

I. Structure Overview

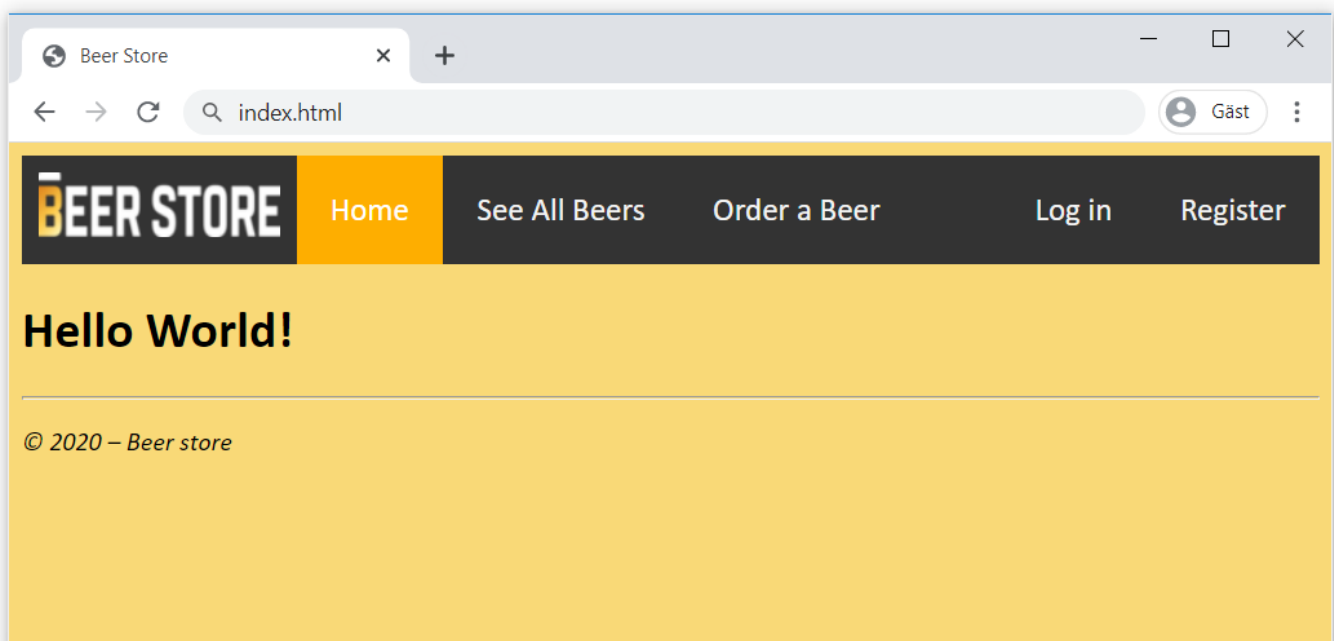
Your task is to create the design for a beer store. With this exercise you are provided with 2 folders.



All the styling will be in a single file – **style.css**, which you will put in the **/style** folder. In the provided resources file you will find several photos. You should find them in the **/images** folder.

II. Designing a Reusable Template

In this chapter, you will design a template, or in other words, a base HTML structure which you will use for the rest of the pages. If you follow every step correctly, you should end up with something that looks a bit like this.



Create a file called **template.html**. In it, write the code from the picture below.

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <link rel="stylesheet" href="style/style.css" />
6   <title>Beer Store</title>
7 </head>
8 <body>
9 <nav>
10   <!-- Here you will code in the navigation bar. -->
11 </nav>
12
13 <main>
14   <!-- Here will be all the content between the navigation bar and the footer -->
15 </main>
16
17 <footer>
18   <!-- Here is where you will code in your footer.
19   content shown on the bottom of the page. -->
20 </footer>
21 </body>
22 </html>
```

Let's start with the footer, since it's the easiest to code. As you can see in the first screenshot, there is a straight line right across the bottom. You can make such a line with the **<hr/>** tag.

```
<footer>
  <hr />
  <p><em>&copy; 2020 &ndash; Beer Store</em></p>
</footer>
```

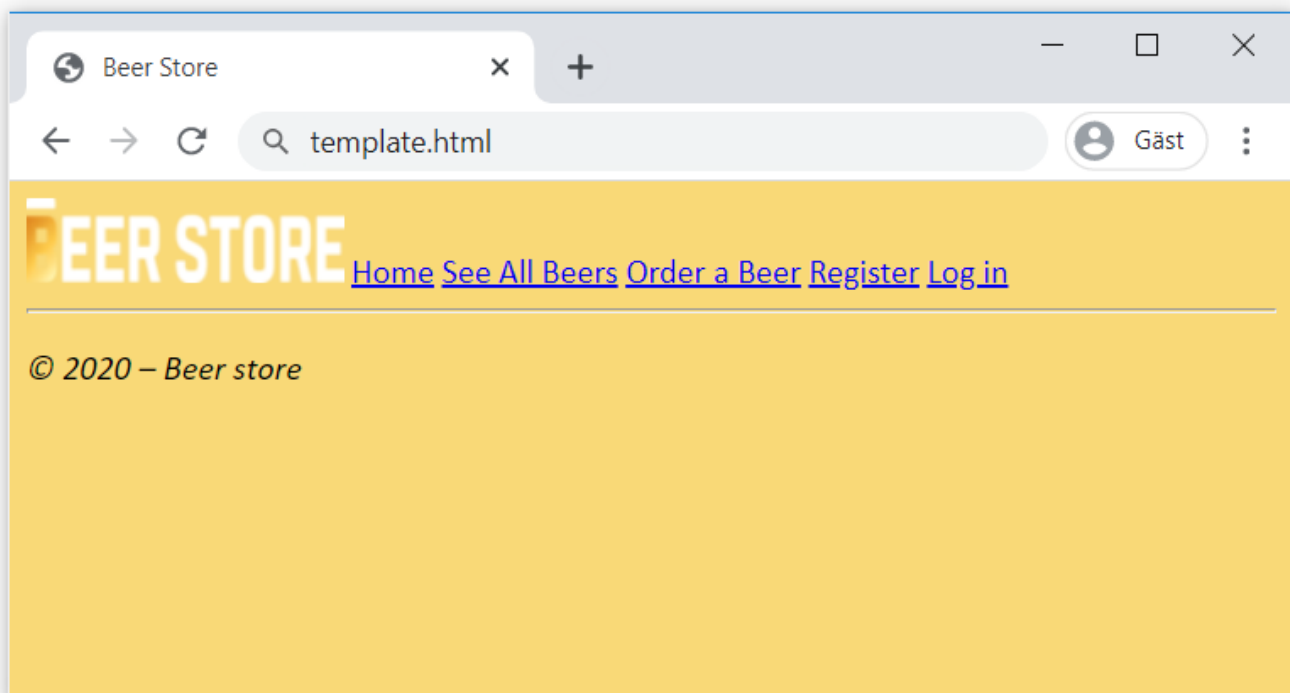
Now it's time to code the hardest part – the navigation bar. We will use a **<div>** to store the links in it and we will style it with CSS. To manipulate it with our **style.css** file, we need to give it an appropriate class name. In our case, we will name the class **topnav**. If you look in your **/images** folder you will see a photo **logo.png**. We need to turn this page into a clickable link. To do that, we put the **** inside of the **<a>** tag like in the photo below.

```
<nav>
  <div class="topnav">
    <a href="index.html"></a>
    <a href="index.html">Home</a>
    <a href="all-beers.html">See All Beers</a>
    <a href="order.html">Order a Beer</a>
    <a href="register.html">Register</a>
    <a href="login.html">Log in</a>
  </div>
</nav>
```

That's enough HTML for now. Time to go to our CSS file. The first thing we need to do is change the font. Let's face it, Times New Roman looks ugly in websites. So, we will use Calibri. Also, we need to change the background color so that it isn't just plain white. Open the **style.css** file and, in it, write the following code:

```
body {  
    background-color: #F9D977;  
    font-family: 'Calibri';  
}
```

Now, your page should look something a bit like this:



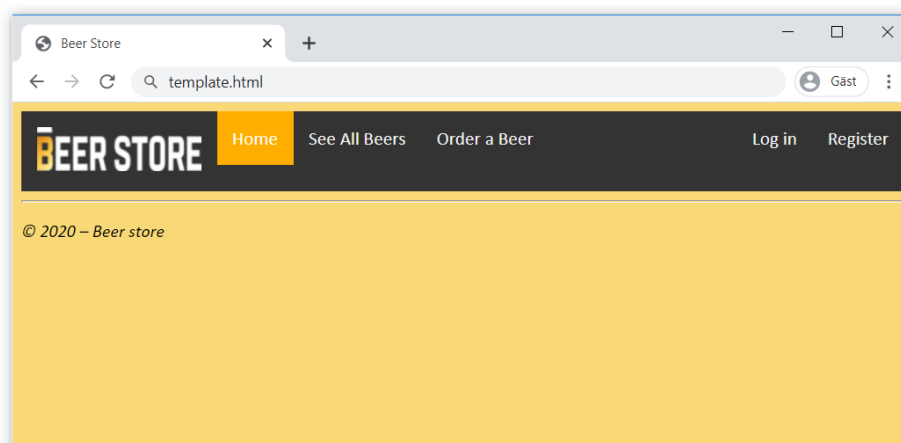
But the navigation bar still looks far from complete – we must style it a bit more and we should change the alignment of some of the links. We added the **topnav** class to the bar some time ago. Now it's time to put it to use. Add the following code to our styling file:

```
/* Add a black background color to the top navigation */  
.topnav {  
    background-color: #333;  
    overflow: hidden;  
}
```

We also need to style the links, but only the ones that are within an element with the **topnav** class. The syntax of styling elements nested within other elements is: **“.class_name element_name”**.

```
/*Style the links inside the navigation bar */
.topnav a {
  float: left;
  display: block;
  color: #f2f2f2;
  text-align: center;
  padding: 21px;
  text-decoration: none;
  font-size: 20px;
}
```

When it comes to links, the **text-decoration** property tells whether the link should be underlined or not. Until now, your template should look something like this:

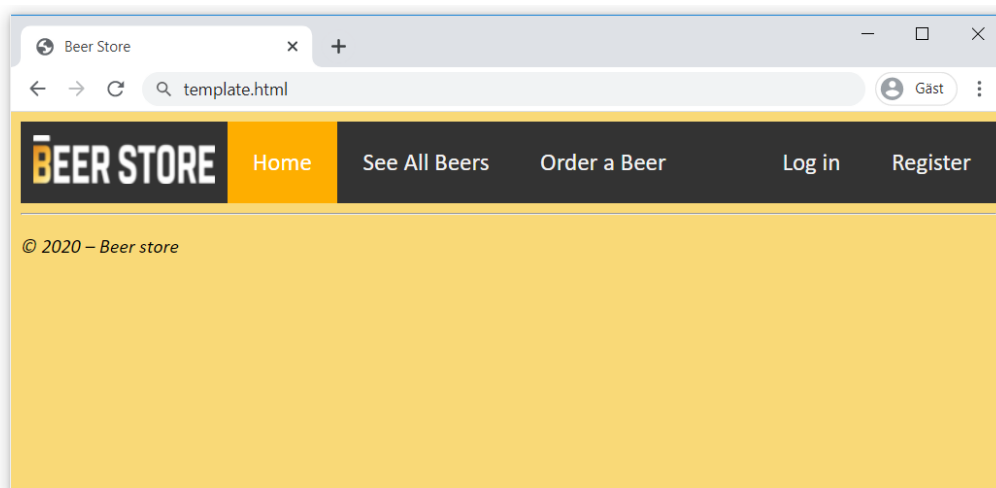


Here we see our first problem – the logo pushes the navigation bar down and messes up the alignment. There are two ways out. The first one is to mess around with the sizes of the image, but that will distort it and it won't look good. The other way around is to remove the padding around the logo and give it a tiny padding on the top so that it is centered:

```
.topnav a.logo {
  padding: 10px;
  margin-top: 1px;
}
```

```
<nav>
  <div class="topnav">
    <a class="logo" href="index.html"></a>
    <a href="index.html">Home</a>
```

Now, if everything is all right, your navigation bar should look a bit nicer:



If you look at the image from the very beginning of the chapter, you will see that the “Register” and “Log in” buttons are aligned to the right. Element alignment is done with the **float** CSS attribute.

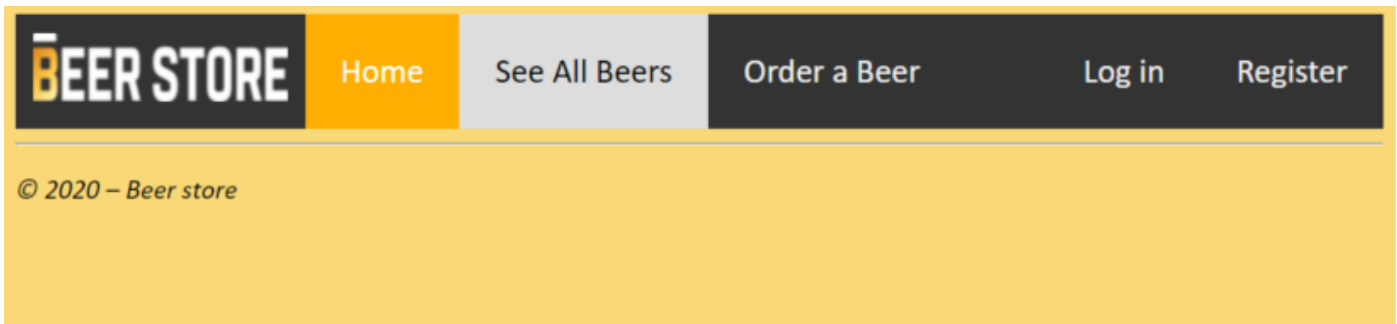
```
.topnav a.right {  
    float: right;  
}
```

```
<a class="right" href="register.html">Register</a>  
<a class="right" href="login.html">Log in</a>
```

If you try and hover with the mouse over any of the links, you will see that nothing happens. This gives no feedback to the user and makes the navigation bar feel dull. We are going to change it. In CSS, you can add styling to elements based on what happens with them (on click, on hover with mouse, etc.) These are called **pseudo-class selectors**. The syntax of them is: “**element_name:pseudoclass_selector**”.

```
.topnav a:hover {  
    background-color: #ddd;  
    color: black;  
}
```

Now if you load up the template, you will see the two links are properly aligned and when you place your mouse over a link, it changes its color:

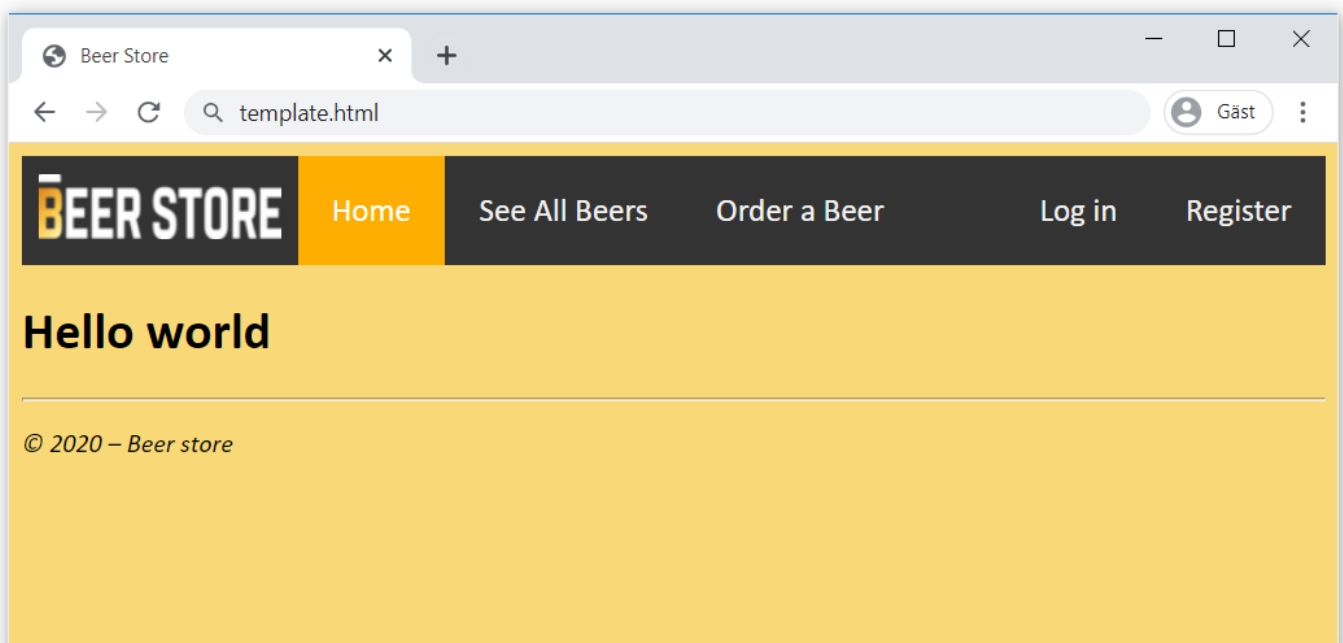


Finally, if you take one last look at the first photo, you will see that the page you are currently on has a different color than the rest, so that the user can know on which page he is. We can do this with one last CSS class.

```
.topnav a.active {  
  background-color: #FEAE00;  
  color: white;  
}
```

```
<a class="active" href="index.html">Home</a>
```

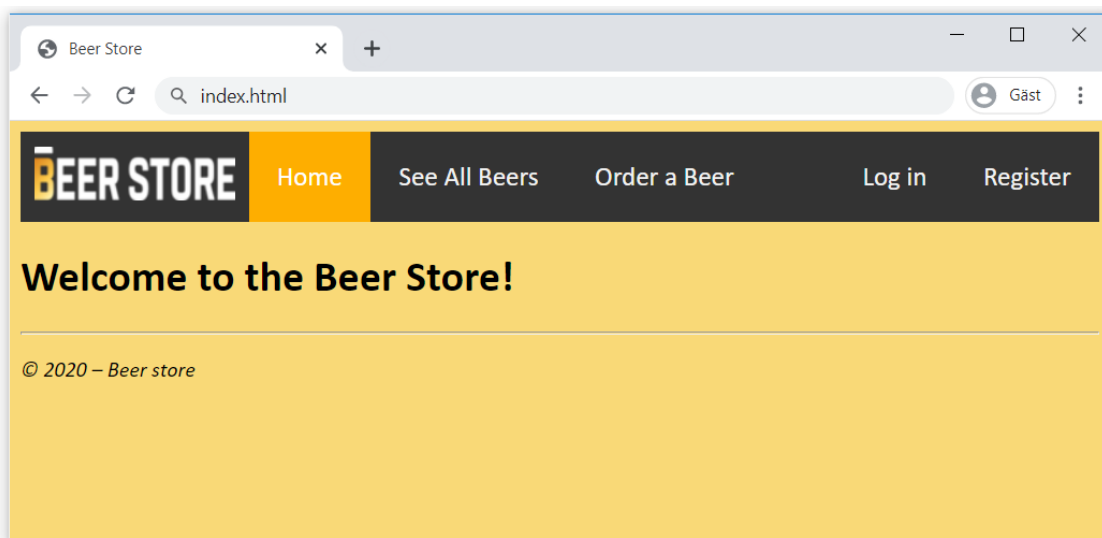
Save everything, add some test text inside of the `<main>` tag and enjoy your template! 😊



For the remaining tasks, you will copy and paste the template you just wrote.

III. Index Page

The index page won't be anything too complicated. Just one heading:



Hint: Use `` for the “Beers” segment of the heading. Also, make the heading bigger:

```
main h1 {  
  font-size: 48px;  
}
```

IV. Register Page

Your next task will be to make a register page that looks something like this:

If you look in the provided resources file you will find a CSS file called **form.css**. It contains the styling for the form we will code. Add it to the **/style** folder and reference it in the **register.html** file.

We need to change the active link on the navigation bar. One element can have more than one class attached to it. The syntax is: **"class1 class2"**.

```
<div class="topnav">
  <a class="logo" href="index.html"></a>
  <a href="index.html">Home</a>
  <a href="all-beers.html">See All Beers</a>
  <a href="order.html">Order a Beer</a>
  <a class="right active" href="register.html">Register</a>
  <a class="right" href="login.html">Log in</a>
</div>
```

You will notice that everything is centered. That's because everything is put inside a **<center>** tag:

```
<main>
  <center>
    <h1>Register</h1>
    <br/>
    <form>
```

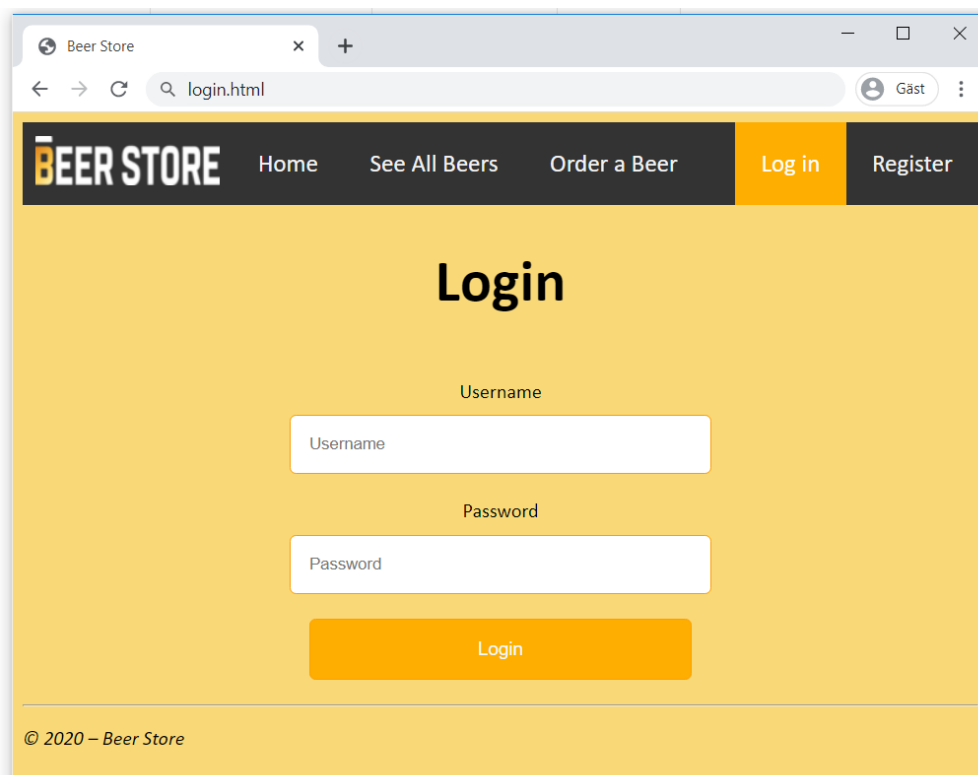

Now it's time to code the form itself. As you can see, there is descriptive text above each of the input fields. No that's not a paragraph. That's a label. Labels are used only within forms and are defined with the `<label>` tag.

[illegible]

Hint: Use `<input>` tags with appropriate **type** attributes and **<label>**.

V. Login Page

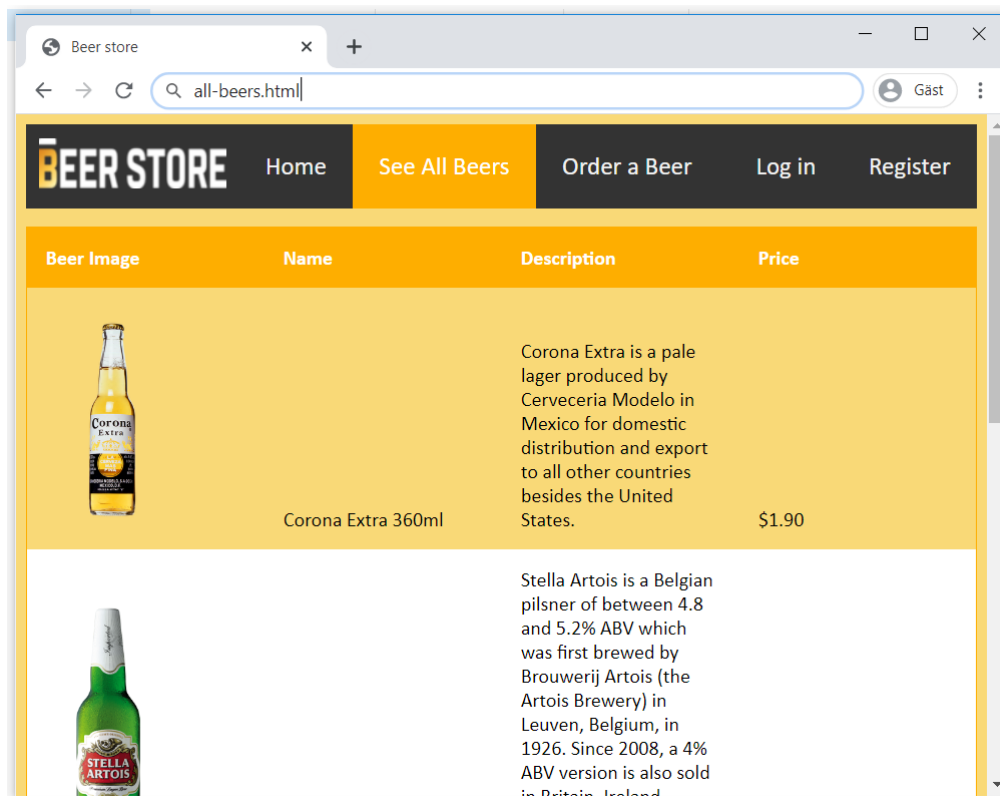
Next up, you need to style a login page.



The process is the same as with the register page – reference the **form.css** file, change the active page in the navigation bar, center everything, make a form with all the appropriate labels and input types.

VI. All Beers List

We need to make a page where the user can see all the available beers:



We are going to use, you guessed it, a table. It will contain 4 columns and 5 rows. The first row will be a header and the other 4 rows will be for data. You should be able to do this without additional help from images.

We first need to change the active link. You should probably know how to do it by now. We need to span the table across the entire page, give it a border and, most importantly, remove the default 1-pixel border around the table elements.

```
main table {  
  width: 100%;  
  border: 1px solid #FEAE00;  
  border-spacing: 0px;  
}
```

We also need to style the header cells and the data cells themselves. Since we have 4 columns, each column should take a quarter of the table. Since the header and data cells share some of the CSS, we can put it in a different block to avoid repetition:

```
main th, td {  
  width: 25%;  
  padding: 15px;  
  text-align: left;  
}
```

By default, table data is centred. We want it to be aligned to the bottom of the cell. To do that, we use the **vertical-align** property:

```
main td {  
    vertical-align: bottom;  
}
```

As for the header, all we need to do is a simple recolor:

```
main th {  
    background-color: #FEAE00;  
    color: white;  
}
```

Now it's time to seed our data in the table. In the first row (the header) add 4 `<th>` tags and write the following:

```
<th>Beer Image</th>
<th>Name</th>
<th>Description</th>
<th>Price</th>
```

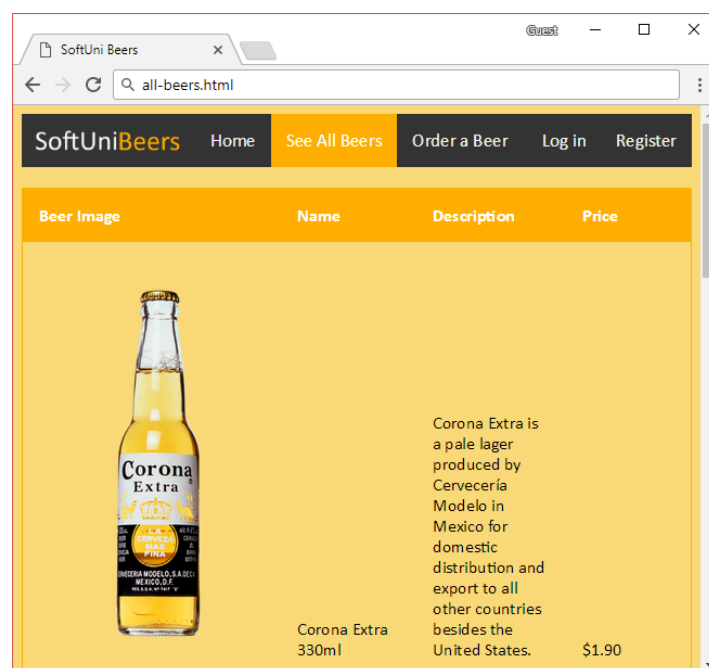
The remaining 4 rows will have the following information inside them:

1. An image of the beer itself
2. The brand of the beer and its quantity
3. A short description (backstory, facts, etc.)
4. The price of the beer

The only thing we've provided is the images themselves. As for the rest – open Wikipedia or just use your imagination. The general concept is:

```
<tr>
  <td></td>
  <td>Corona Extra 330ml</td>
  <td>Corona Extra is a pale lager produced by Cervecería Modelo in Mexico
    for domestic distribution and export to all other countries besides the
    United States.</td>
  <td>$1.90</td>
</tr>
```

Now, we come across a big (literally) problem. The provided images have quite big sizes:



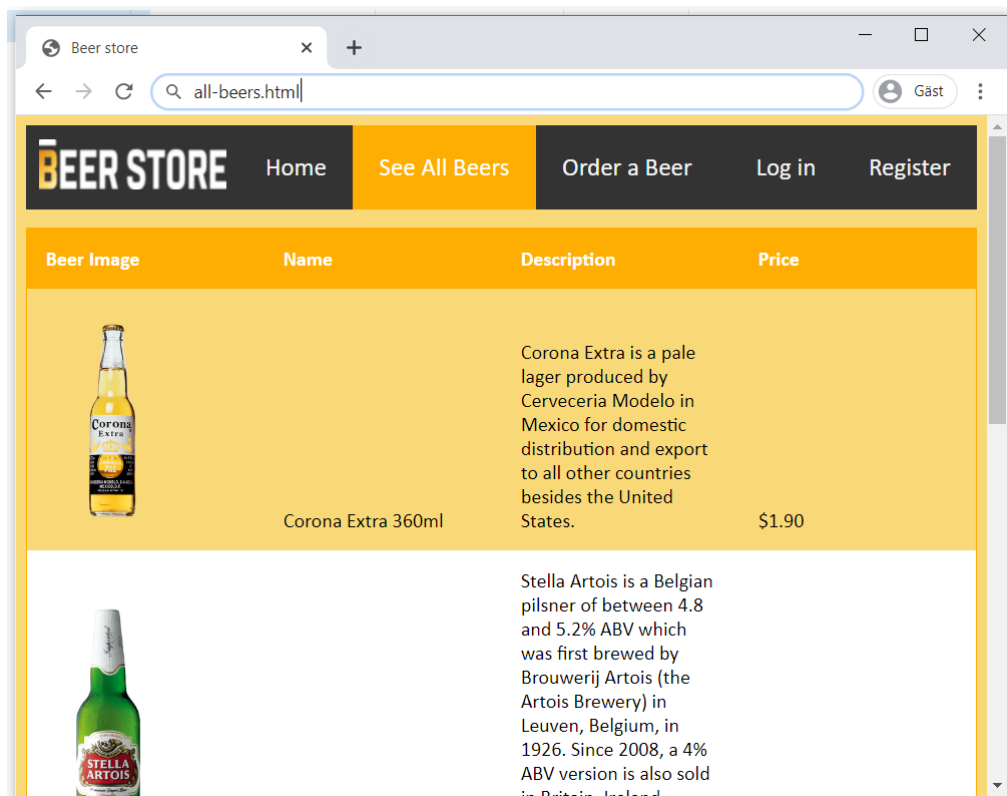
We need to fix this. One clever way is to set a constant width to the image and let the height change proportionally to the width:

```
main table img {
  width: 100px;
  height: auto;
}
```

And, lastly, if you look at the photo from the beginning of the chapter, you will see that there is a “zebra” pattern on the table. Doing this with a single CSS class and copy-pasting it over and over gets hideous and, if there are new entries to the table, you must add the class manually. Remember those pseudo-class selectors we talked about earlier? They can be used to help us for this. The “**:nth-child(even/odd)**” is the selector we need:

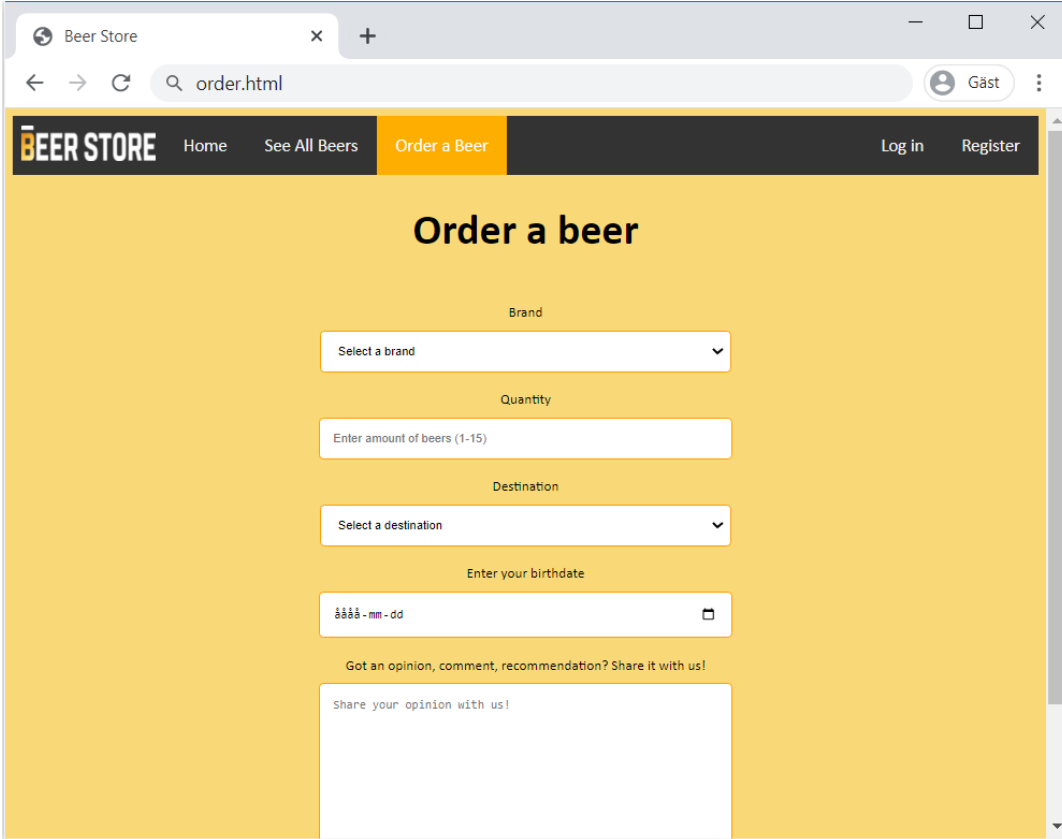
```
main tr:nth-child(odd) {
  background-color: white;
}
```

Now everything should be set and this page should be complete:



VII. Order Page

Finally, we need to style a page from which the user can order his beers:



If this looks like what we have done before, that's because we will use the same **form.css** file we used for the login and register pages.

The process is pretty much the same – we will use a **<center>** tag, a heading on the top, and, of course, the form itself.

For the brand selection and the destination, we will use a dropdown list (**<select>** and **<option>** tags). This should be pretty straight-forward, but we also need to add a placeholder option which is just for description and should not be selected. Luckily, HTML has the solution we are looking for (we won't need to write CSS). We can tell an option to be selected, yet disabled. This means that that option will be displayed to the user, but once a different selection is made, you cannot get back to it.

```

<form>
  <label>Brand</label>
  <select>
    <option selected disabled>Select a brand</option>
    <option>Corona Extra 330ml</option>
    <option>Stella Artois 500ml</option>
    <option>Carlsberg 330ml</option>
  </select>

```

For the quantity, we'll use a number input type. However, the input type doesn't have default restrictions for when the user to stop increasing the number. This means that the user can order a thousand beers from our store. The **min** and **max** attributes can help us for this.

```

<label>Quantity</label>
<input type="number" min="1" max="15"
placeholder="Enter amount of beers (1-15)">

```

Repeat the same process for a while ago for the destination dropdown list. For the birth date we will use, you guessed it, a date input type. No special styling needed. Same thing about the text area. Finally, add a submit input type on the bottom and you should be all set with your beer store template!