**Learn HTML Forms by Building a Registration Form**

You can use HTML forms to collect information from people who visit your webpage.

In this course, you'll learn HTML forms by building a signup page. You'll learn how to control what types of data people can type into your form, and some new CSS tools for styling your page.

Sample Project

Registration Form

Please fill out this form with the required information

Enter Your First Name: 

Enter Your Last Name: 

Enter Your Email:

Create a New Password: 

Personal Account

Business Account

I accept the [terms and conditions](https://www.freecodecamp.org/news/terms-of-service/)

Upload a profile picture:

Input your age (years):

How did you hear about us? 

Provide a bio:

* **



# Step 8

The vh unit stands for viewport height, and is relative to 1% of the height of the viewport.

It is time to spruce the project up with some CSS. Begin by giving the body a width of 100%, and a height of 100vh.

Step 9

Now, get rid of the horizontal scroll-bar, by setting the body default margin added by some browsers to 0.

# Step 10

That is better. Now, make the background easy on the eyes, by changing the body background-color to #1b1b32. Then, to see the text, change the color to #f5f6f7.

CSS for it

body {

  width: 100%;

  height: 100vh;

  margin: 0;

}

# Step 11

As suggested by the title, you are creating a form. So, after the p element, insert a form with an action attribute targeting https://register-demo.freecodecamp.org.

# Step 12

The method attribute specifies how to send form-data to the URL specified in the action attribute. The form-data can be sent via a GET request as URL parameters (with method="get") or via a POST request as data in the request body (with method="post").

Set the method attribute to send your form data via a POST request.

<form action='https://register-demo.freecodecamp.org' method="post"></form>

# Step 13

As the form will have three distinct sections, add three fieldset elements within the form element.

# Step 14

The first fieldset will hold name, email, and password fields. Start by adding four label elements to the first fieldset.

# Step 15

Add the following text to the label elements:

* Enter Your First Name:
* Enter Your Last Name:
* Enter Your Email:
* Create a New Password:
* <fieldset>
* <label>name</label>
* <label>last name</label>
* <label>email</label>
* <label>pasowrd</label>
* </fieldset>

# Step 16

The rem unit stands for root em, and is relative to the font size of the html element.

As label elements are inline by default, they are all displayed side by side on the same line, making their text hard to read. To make them appear on separate lines, add display: block to the label element, and add a margin of 0.5rem 0, to separate them from each other.

# Step 18

Following accessibility best practices, link the input elements and the label elements together using the for attribute.

Use first-name, last-name, email, and new-password as values for the respective id attributes.

<fieldset>

        <label>Enter Your First Name: <input /></label>

        <label>Enter Your Last Name: <input /></label>

        <label>Enter Your Email: <input /></label>

        <label>Create a New Password: <input /></label>

      </fieldset>

    <fieldset>

        <label for="first-name">Enter Your First Name: <input type="first-name" id="first-name" /></label>

        <label for="last-name">Enter Your Last Name: <input type="last-name" id="last-name"/></label>

        <label for="email">Enter Your Email: <input type="email" id="email"/></label>

        <label for="new-password">Create a New Password: <input type="new-password"

        id="new-password" /></label>

      </fieldset>

# Step 19

Specifying the type attribute of a form element is important for the browser to know what kind of data it should expect. If the type is not specified, the browser will default to text.

Give the first two input elements a type attribute of text, the third a type attribute of email, and the fourth a type attribute of password.

The email type only allows emails with a @ and a . in the domain. The password type obscures the input, and warns if the site does not use HTTPS.

# Step 20

The first input element with a type of submit is automatically set to submit its nearest parent form element.

To handle the form submission, after the last fieldset element add an input element with the type attribute set to submit and the value attribute set to Submit.

  <fieldset></fieldset>

      <fieldset></fieldset>

      <input type="submit" value="Submit"/>

# Step 21

At this point, you should be able to submit the form. However, you might notice not much happens.

To make the form more interactive, add the required attribute to the input elements in the first fieldset.

Now, if you try to submit the form without filling in the required fields, you will see an error message.

# Step 22

Certain type attribute values come with built-in form validation. For example, type="email" requires that the value be a valid email address.

Add custom validation to the password input element, by adding a minlength attribute with a value of 8. Doing so prevents inputs of less than 8 characters being submitted.

<label for="new-password">Create a New Password: <input id="new-password" type="password" required minlength="8" />

</label>

</fieldset>

# Step 23

With type="password" you can use the pattern attribute to define a regular expression that the password must match to be considered valid.

Add a pattern attribute to the password input element to require the input match: [a-z0-5]{8,}

The above is a regular expression which matches eight or more lowercase letters or the digits 0 to 5. Then, remove the minlength attribute, and try it out.

<label for="new-password">Create a New Password: <input id="new-password" type="password" required pattern="[a-z0-5]{8,}" /></label>

      </fieldset>

Minlentgh will must remove from it

# Step 24

Let us go to the next part of the registration form. This section will ask for the type of account the user is opening, and will confirm the user has read the terms and conditions.

Start by adding three label elements to the second fieldset.

<fieldset>

      </fieldset>

# Step 25

Users will be allowed to choose either a Personal Account or Business Account.

To do this, within each of the first two label elements, add one input element with type="radio".

<fieldset>

        <label><input type="radio"/></label>

        <label><input type="radio"/></label>

        <label></label>

      </fieldset>

# Step 26

For the terms and conditions, add an input with a type of checkbox to the third label element. Make this input element required because users should not sign up without reading the terms and conditions.

 <label><input type="checkbox" required /></label>

# Step 27

Within each corresponding label element, and immediately after the input element, add a space and add the following text:

Personal Account

Business Account

I accept the terms and conditions

<fieldset>

        <label><input type="radio" /> Personal Account</label>

        <label><input type="radio" /> Business Account</label>

        <label><input type="checkbox" required /> I accept the terms and conditions</label>

      </fieldset>

# Step 28

You only want one radio input to be selectable at a time. However, the form does not know the radio inputs are related.

To relate the radio inputs, give them the same name attribute with a value of account-type. Now, it is not possible to select both radio inputs at the same time.

name="account-type"

this is the solution now the radio button will be selected only one else two

# Step 29

Follow accessibility best practices by linking the input elements and the label elements in the second fieldset.

Use personal-account, business-account, and terms-and-conditions as values for the respective id attributes.

  <fieldset>

        <label for="personal-account" ><input type="radio" name="account-type" id="personal-account"/> Personal Account</label>

        <label for="business-account"><input type="radio" name="account-type" id="business-account" /> Business Account</label>

        <label for="terms-and-conditions"><input type="checkbox" id="terms-and-conditions" required /> I accept the terms and conditions</label>

      </fieldset>

# Step 30

To finish this fieldset off, link the text terms and conditions in the third label to the following location:

https://www.freecodecamp.org/news/terms-of-service/

   <fieldset>

        <label for="personal-account"><input id="personal-account" type="radio" name="account-type" /> Personal Account</label>

        <label for="business-account"><input id="business-account" type="radio" name="account-type" /> Business Account</label>

        <label for="terms-and-conditions"><input id="terms-and-conditions" type="checkbox" required /> I accept the         <a href="https://www.freecodecamp.org/news/terms-of-service/"</a>terms and conditions</label>

      </fieldset>

# Step 31

Moving on to the final fieldset. What if you wanted to allow a user to upload a profile picture?

Well, the input type file allows just that. Add a label with the text Upload a profile picture: , and nest an input accepting a file upload.

<fieldset>

     <label>Upload a profile picture: <input type="file"/></label>

      </fieldset>

# Step 32

Add another label after the first, with the text Input your age (years): . Then, nest an input with the type of number.

Next, add a min attribute to the input with a value of 13 because users under the age of 13 should not register. Also, users probably will not be over the age of 120; add a max attribute with a value of 120.

Now, if someone tries to submit the form with values outside of the range, a warning will appear, and the form will not submit. Give it a try.

      <fieldset>

        <label>Upload a profile picture: <input type="file" /></label>

           <label>Input your age (years): <input type="number" min="13" max="120" /></label>

      </fieldset>

# Step 33

Adding a dropdown to the form is easy with the select element. The select element is a container for a group of option elements, and the option element acts as a label for each dropdown option. Both elements require closing tags.

Start by adding a select element below the two label elements. Then nest 5 option elements within the select element.

<select>

          <option></option>

          <option></option>

          <option></option>

          <option></option>

          <option></option>

        </select>

# Step 34

Nest the select element (with its option elements) within a label element with the text How did you hear about us?. The text should come before the select element.

        <label>

How did you hear about us?

        <select>

          <option>

          </option>

          <option></option>

          <option></option>

          <option></option>

          <option></option>

        </select>

        </label>

# Step 35

The dropdown options are currently empty. To give them content, add the following text to each subsequent option element:

(select one)

freeCodeCamp News

freeCodeCamp YouTube Channel

freeCodeCamp Forum

Other

 <fieldset>

        <label>Upload a profile picture: <input type="file" /></label>

        <label>Input your age (years): <input type="number" min="13" max="120" /></label>

        <label>How did you hear about us?

          <select>

            <option>(select one)</option>

            <option>freeCodeCamp News</option>

            <option>freeCodeCamp YouTube Channel</option>

            <option>freeCodeCamp Forum</option>

            <option>Other</option>

          </select>

        </label>

      </fieldset>

# Step 36

Submitting the form with an option selected would not send a useful value to the server. As such, each option needs to be given a value attribute. Without which, the text content of the option will be submitted to the server.

Give the first option a value of "", and the subsequent option elements value attributes from 1 to 4.

<select>

            <option value="">(select one)</option>

            <option value="1">freeCodeCamp News</option>

            <option value="2">freeCodeCamp YouTube Channel</option>

            <option value="3">freeCodeCamp Forum</option>

            <option value="4">Other</option>

          </select>

# Step 37

The textarea element acts like an input element of type text, but comes with the added benefit of being able to receive multi-line text, and an initial number of text rows and columns.

Users will be able to register with a bio. Add a label with the text Provide a bio: at the end of the fieldset. Add a textarea element inside the label element. Note that the textarea requires a closing tag.

<label>Provide a bio: <textarea></textarea></label>

# Step 38

Link the applicable form elements and their label elements together.

Use profile-picture, age, referrer, and bio as values for the respective id attributes.

<label id="profile-picture", "age", "referrer"> Provide a bio:

          <textarea></textarea>

        </label>

# Step 38

Link the applicable form elements and their label elements together.

Use profile-picture, age, referrer, and bio as values for the respective id attributes.

<fieldset>

        <label for="profile-picture">Upload a profile picture: <input type="file" id="profile-picture"/></label>

        <label for="age">Input your age (years): <input type="number" min="13" max="120" id="age" /></label>

        <label for="referrer">How did you hear about us?

          <select id="referrer">

            <option value="">(select one)</option>

            <option value="1">freeCodeCamp News</option>

            <option value="2">freeCodeCamp YouTube Channel</option>

            <option value="3">freeCodeCamp Forum</option>

            <option value="4">Other</option>

          </select>

        </label>

        <label for="bio"> Provide a bio:

          <textarea id="bio"></textarea>

        </label>

      </fieldset>

# Step 39

The textarea appears too small. To give it an initial size, you can add the rows and cols attributes.

Add an initial size of 3 rows and 30 columns.

 <textarea id="bio" rows="3" cols="30"></textarea>

# Step 40

To give Campers an idea of what to put in their bio, the placeholder attribute is used. The placeholder accepts a text value, which is displayed until the user starts typing.

Give the textarea a placeholder of I like coding on the beach....

  <textarea id="bio" rows="3" cols="30" placeholder="I like coding on the beach...."></textarea>

# Step 41

With form submissions, it is useful, and good practice, to provide each submittable element with a name attribute. This attribute is used to identify the element in the form submission.

Give each submittable element a unique name attribute of your choosing, except for the two radio inputs.

Name=”first-name”

Name=last-name”

# Step 42

The HTML for the registration form is finished. Now, you can spruce it up a bit.

Start by changing the font to Tahoma, and the font size to 16px in the body.

 font-family:Tahoma;

  font-size:16px;

# Step 43

Center the h1 and p elements by giving them a margin of 1em auto. Then, align their text in the center as well.

h1 , p{

  margin:1em auto;

  text-align:center;

}

# Step 44

Center the form element, by giving it a margin of 0 auto. Then, fix its size to a maximum width of 500px, and a minimum width of 300px. In between that range, allow it to have a width of 60vw.

form{

  margin:0 auto;

  max-width:500px;

  min-width:300px;

  width:60vw;

  background:red;

}

# this is very important to undersatand

# Step 45

During development, it is useful to see the fieldset default borders. However, they make the content appear too separated.

Remove the border, and add 2rem of padding only to the top and bottom of each fieldset. Be sure to remove the padding from the left and right.

fieldset{

  border:none;

  background:red;

  padding:2rem 0px;

}

# Step 46

To give the fieldset elements a bit of separation, select them and give them a border-bottom of 3px solid #3b3b4f

border-bottom:3px solid #3b3b4f;

# Step 47

The border of the last fieldset element looks a little out of place. You can select the last element of a specific type using the last-of-type CSS pseudo-class, like this:

p:last-of-type { }

That will select the last p element. Create a new selector that targets the last fieldset element and set its border-bottom to none.

fieldset:last-of-type{

  border-bottom:none;

}

# Step 48

It would be nicer to have the label text appear above the form elements.

Select all input, textarea, and select elements, and make them take up the full width of their parent elements.

Also, add 10px of margin to the top of the selected elements. Set the other margins to 0.

input,textarea,select{

  margin-left:0;

  margin-right:0;

  margin-bottom:0;

  margin-top:10px;

  width:100%;

  background:red;

}

# Step 49

For the second fieldset, you want the input and label text to appear on the same line.

Start, by giving the input elements in the second fieldset a class of inline.

<fieldset class="inline">

        <label for="personal-account"><input id="personal-account" type="radio" name="account-type" class="inline" /> Personal Account</label>

        <label for="business-account"><input id="business-account" type="radio" name="account-type" class="inline" /> Business Account</label>

        <label for="terms-and-conditions">

          <input id="terms-and-conditions" type="checkbox" required name="terms-and-conditions" class="inline" /> I accept the <a href="https://www.freecodecamp.org/news/terms-of-service/">terms and conditions</a>

        </label>

      </fieldset>

# Step 50

Select only the .inline elements, and give them width of unset. This will remove the earlier rule which set all the input elements to width: 100%.

.inline{

  width:unset;

}

# Step 51

Add some space between the .inline elements and the label text, by giving a right margin of 0.5em. Also, set all the other margin to 0.

 margin:0 0.5em 0 0;

# Step 52

If you look close enough, you will notice the .inline elements are too high on the line.

To combat this, set the vertical-align property to middle.

.inline {

  width: unset;

  margin: 0 0.5em 0 0;

  vertical-align:middle; // this is ue for down the entire radio

}

# Step 53

To make the input and textarea elements blend in with the background theme, set their background-color to #0a0a23. Then, give them a 1px, solid border with a color of #0a0a23.

input ,textarea{

  background-color:#0a0a23;

  border:1px solid #0a0a23;

}

# Step 54

Currently, if you type in the input or textarea elements, you will not be able to see the text. Also, their height is too small to be easy to use.

Fix this, by setting the color to #ffffff, and setting their min-height to 2em.

color:#ffffff;

  min-height:2em;

# Step 55

You want the select element to remain with a white background, but now it is not getting the same min-height as the input and textarea elements.

Move the min-height property and value so that all three element types have the same min-height value, and the select element still has a white background.

# Step 56

To style the submit button, you can use an attribute selector, which selects an element based on the given attribute value. Here is an example:

input[name="password"]

The above selects input elements with a name attribute value of password.

Now, use the attribute selector to style the submit button with a display of block, and a width of 60%.

input[type="submit"]{

  display:block;

  width:60%;

}

# Step 57

With a display of block the submit button sits flush against the left edge of its parent.

Use the same technique used to center the form to center the submit button.

input[type="submit"] {

  display: block;

  width: 60%;

  margin:0 auto;

}

# Step 58

To make the submit button look more in line with the rest of the form, give it the same height as the other fields (2em). Also, increase the font-size to 1.1rem.

input[type="submit"] {

  display: block;

  width: 60%;

  margin: 0 auto;

  height:2em;

  font-size:1.1rem;

}

# Step 59

To make the submit button appear more distinct, give it a background-color of #3b3b4f, and a border-color of white.

 background-color:#3b3b4f;

  border-color:white;

# Step 60

Lastly, for the submit button, you want to separate it from the fieldset above, and adjust its width to never be below 300px.

Change the margin property to include 1em on the top and bottom, while leaving the right and left margins set to auto. Then set the width as described above.

  border-color: white;

  min-width:300px;

# Step 61

Most browsers inject their own default CSS properties and values for different elements. If you look closely, you might be able to notice the file input is smaller than the other text input elements. By default, a padding of 1px 2px is given to input elements you can type in.

Using another attribute selector, style the input with a type of file to be the same padding as the other input elements.

# Step 62

Speaking of padding, the submit button is sitting at the bottom of the form element. Add 2em of padding only to the bottom of the form.

form {

  width: 60vw;

  max-width: 500px;

  min-width: 300px;

  margin: 0 auto;

  padding-bottom:2em;

}

# Step 63

Last, but not least, change the text color of the terms and conditions link to #dfdfe2.

Well done! You have completed the final part of the Registration Form practice project.

a{

  color:#dfdfe2;

}