# Introduction to HTML Forms

Forms are a part of everyday life. When we use a physical form in real life, we write down information and give it to someone to process. Think of the times you’ve had to fill out information for various applications like a job, or a bank account, or dropped off a completed suggestion card — each instance is a form!

Just like a physical form, an HTML <form> element is responsible for collecting information to send somewhere else. Every time we browse the internet we come into contact with many forms and we might not even realize it. There’s a good chance that if you’re typing into a text field or providing an input, the field that you’re typing into is part of a <form>!

In this lesson, we’ll go over the structure and syntax of a <form> and the many elements that populate it.

**Instructions**

Look over the image, can you think of any other forms you’ve interacted with? Go to the next exercise when you’re ready.

# How a Form Works

We can think of the internet as a network of computers which send and receive information. Computers need an *HTTP request* to know how to communicate. The HTTP request instructs the receiving computer how to handle the incoming information. More information can be found in our article about [HTTP requests](https://www.codecademy.com/articles/http-requests).

The <form> element is a great tool for collecting information, but then we need to send that information somewhere else for processing. We need to supply the <form> element with both the location of where the <form>‘s information goes and what HTTP request to make. Take a look at the sample <form> below:

<form action="/example.html" method="POST">

</form>

In the above example, we’ve created the skeleton for a <form> that will send information to **example.html** as a POST request:

* The action attribute determines where the information is sent.
* The method attribute is assigned a HTTP verb that is included in the HTTP request.

Note: HTTP verbs like POST do not need to be capitalized for the request to work, but it’s done so out of convention. In the example above we could have written method="post" and it would still work.

The <form> element can also contain child elements. For instance, it would be helpful to provide a header so that users know what this <form> is about. We could also add a paragraph to provide even more detail. Let’s see an example of this in code:

<form action="/example.html" method="POST">

<h1>Creating a form</h1>

<p>Looks like you want to learn how to create an HTML form. Well, the best way to learn is to play around with it.</p>

</form>

The example above doesn’t collect any user input, but we’ll do that in the next exercise. For now, let’s practice making the foundation of an HTML <form>!

**1.**

In the <section> element, add a <form> element under the provided comment. Assign the <form> with:

* an action attribute with a value of "/practice.html"
* a method attribute with a value of "POST"

Hint: Make sure you’re writing your code between the opening and closing tags of the <body> element.

To create a <form> provide both an opening and closing <form> tag.

In the opening <form> tag, add the attribute of action and assign to it a value of "/practice.html".

Then, in the same opening <form> tag, add an attribute of method and assign to it a value of "POST".

**2.**

Right now we have a blank <form> on a burger site, let’s add some context.

Add an <h1> inside the <form> element with text related to the site between the opening and closing <h1> tags.

Hint: Make sure that you’re adding the <h1> element inside the <form> element. The <h1> element should render some text to the page.

**3.**

Add some details to the form by inserting a <p> element below the <h1> element. Write a relevant description within the <p> element.

Hint: Add the <p> element underneath the <h1> element. Do NOT nest the <p> element inside the <h1> element. Lastly, make sure that there is text between the opening and closing tag of the <p> element.

Ans:

<!DOCTYPE html>

<html lang="en" dir="ltr">

  <head>

    <meta charset="utf-8">

    <link rel="stylesheet" type="text/css" href="style.css">

    <link href="https://fonts.googleapis.com/css?family=Rubik" rel="stylesheet">

    <title>HTML Forms</title>

  </head>

  <body>

    <section id="overlay">

      <img src="https://s3.amazonaws.com/codecademy-content/courses/web-101/unit-6/htmlcss1-img\_burger-logo.svg" alt="Davie's Burgers Logo" id="logo">

      <hr>

      <!--Add your code below-->

      <form action="/practice.html" method="POST">

      <h1>hi</h1>

      <p>thanks for comming</p>

      </form>

    </section>

  </body>

</html>

**HTML FORMS**

# Text Input

If we want to create an input field in our <form>, we’ll need the help of the <input> element.

The <input> element has a type attribute which determines how it renders on the web page and what kind of data it can accept.

The first value for the type attribute we’re going to explore is "text". When we create an <input> element with type="text", it renders a text field that users can type into. It’s also important that we include a name attribute for the <input> — without the name attribute, information in the <input> won’t be sent when the <form> is submitted. We’ll explain more about submissions and the submit button in a later exercise. For now, let’s examine the following code that produces a text input field:

<form action="/example.html" method="POST">

<input type="text" name="first-text-field">

</form>

Here’s a screen shot of how the rendered form looks like on a web page for the Chrome browser (different browsers have different default rendering). When initially loaded, it will be an empty box:

rendered empty text field from input element type='text'

After users type into the <input> element, the value of the value attribute becomes what is typed into the text field. The value of the value attribute is paired with the value of the name attribute and sent as text when the form is submitted. For instance, if a user typed in “important details” in the text field created by our <input> element:

rendered filled text field which reads 'important details' 

When the form is submitted, the text: "first-text-field=important details" is sent to /example.html because the value of the name attribute is "first-text-field" and the value of value is "important details".

We could also assign a default value for the value attribute so that users have a pre-filled text field when they first see the rendered form like so:

<form action="/example.html" method="POST">

<input type="text" name="first-text-field" value="already pre-filled">

</form>

Which renders:

pre-filled text box due to assigned `value` attribute

Time to put this knowledge into practice!

**Instructions**

**1.**

Let’s start with creating a login form for our users.

Inside the provided <form> element, add an <input> element with a type attribute of "text".

Hint

Make sure you’re adding code in between the opening and closing tags of <form>.

**2.**

Even though we’re not submitting the form, let’s develop some good habits by giving the <input> a name attribute with a value of "username".

Hint

In the opening tag of <input>, add another attribute of name with a value of "username".

**3.**

Let’s see what happens if we add a value attribute with a value of "Davie"

Hint

In the opening <input> tag , add another attribute of value with a value of "Davie".

Ans:

<!DOCTYPE html>

<html lang="en" dir="ltr">

  <head>

    <meta charset="utf-8">

    <link rel="stylesheet" type="text/css" href="style.css">

    <link href="https://fonts.googleapis.com/css?family=Rubik" rel="stylesheet">

    <title>Adding a Text Input</title>

  </head>

  <body>

    <section id="overlay">

      <img src="https://s3.amazonaws.com/codecademy-content/courses/web-101/unit-6/htmlcss1-img\_burger-logo.svg" alt="Davie's Burgers Logo" id="logo">

      <hr>

      <form>

        <h1>Login to start creating a burger!</h1>

        <!--Add your code below-->

        <input type="text" name="username" value="Davie">

      </form>

    </section>

  </body>

</html>

# Adding a Label

In the previous exercise we created an <input> element but we didn’t include anything to explain what the <input> is used for. For a user to properly identify an <input> we use the appropriately named <label> element.

The <label> element has an opening and closing tag and displays text that is written between the opening and closing tags. To associate a <label> and an <input>, the <input> needs an id attribute. We then assign the for attribute of the <label> element with the value of the id attribute of <input>, like so:

<form action="/example.html" method="POST">

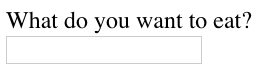
<label for="meal">What do you want to eat?</label>

<br>

<input type="text" name="food" id="meal">

</form>

The code above renders:



Look, now users know what the <input> element is for! Another benefit for using the <label> element is when this element is clicked, the corresponding <input> is highlighted/selected.

Let’s see the <label> element in action!

**Instructions**

**1.**

Add a <label> element that is associated with the included <input> element in **index.html**. (use the for attribute!)

Then add text Username within the <label> element.

After clearing this checkpoint, click on the label to see the <input> field in focus!

Hint

To associate a <label> with a an <input> add a for attribute with the value of the <input> element’s id value. For example:

<form action="/example.html" method="POST">

<label for="meal">What do you want to eat?</label>

<input type="text" name="meal" id="meal">

</form>

Ans:

<!DOCTYPE html>

<html lang="en" dir="ltr">

  <head>

    <meta charset="utf-8">

    <link rel="stylesheet" type="text/css" href="style.css">

    <link href="https://fonts.googleapis.com/css?family=Rubik" rel="stylesheet">

    <title>Labels</title>

  </head>

  <body>

    <section id="overlay">

      <img src="https://s3.amazonaws.com/codecademy-content/courses/web-101/unit-6/htmlcss1-img\_burger-logo.svg" alt="Davie's Burgers Logo" id="logo">

      <hr>

      <form>

        <h1>Login to start creating a burger!</h1>

        <!--Add your code below-->

        <label for="username">Username</label>

        <input type="text" name="username" id="username">

      </form>

    </section>

  </body>

</html>

# Password Input

Think about all those times we have to put sensitive information, like a password or PIN, into a <form>. We wouldn’t want our information to be seen by anyone peeking over our shoulder! Luckily, we have the type="password" attribute for <input>!

An <input type ="password"> element will replace input text with another character like an asterisk (\*) or a dot (•). The code below provides an example of how to create a password field:

<form>

<label for="user-password">Password: </label>

<input type="password" id="user-password" name="user-password">

</form>

After a user types into the field, it would look like:

password field in a form with 6 dots showing text added to the field

Even though the password field obscures the text of the password, when the form is submitted, the value of the text is sent. In other words, if “hunter2” is typed into the password field, “user-password=hunter2” is sent along with the other information on the form.

**Instructions**

**1.**

To complete our login page in **index.html** we need a password field. Add an <input> element under the second <label> element.

* Assign the id to the correct value to associate the second <label> with this new <input>.
* Set the newly created <input> element’s type attribute to "password".
* Set the name attribute to "user-pw".

Hint

Make sure you’re adding the <input> into the <form> element. Assign attributes to the <input> like so:

<input exampleAttribute="value-for-attribute">

To associate the <input> with the created <label> the value of the for attribute must match the value of the id of <input>.

Ans:

<!DOCTYPE html>

<html lang="en" dir="ltr">

  <head>

    <meta charset="utf-8">

    <link rel="stylesheet" type="text/css" href="style.css">

    <link href="https://fonts.googleapis.com/css?family=Rubik" rel="stylesheet">

    <title>Password Input</title>

  </head>

  <body>

    <section id="overlay">

      <img src="https://s3.amazonaws.com/codecademy-content/courses/web-101/unit-6/htmlcss1-img\_burger-logo.svg" alt="Davie's Burgers Logo" id="logo">

      <hr>

      <form>

        <h1>Login to start creating a burger!</h1>

        <label for="username">Username:</label>

        <input type="text" name="username" id="username">

        <br>

        <label for="user-pw">Password:</label>

        <!--Add your code below-->

        <input type="password" id="user-pw" name="user-pw">

      </form>

    </section>

  </body>

</html>

# Number Input

We’ve now gone over two type attributes for <input> related to text. But, we might want our users to type in a number — in which case we can set the type attribute to… (you guessed it)… "number"!

By setting type="number" for an <input> we can restrict what users type into the input field to just numbers (and a few special characters like -, +, and .). We can also provide a step attribute which creates arrows inside the input field to increase or decrease by the value of the step attribute. Below is the code needed to render an input field for numbers:

<form>

<label for="years"> Years of experience: </label>

<input id="years" name="years" type="number" step="1">

</form>

Which renders:

rendered number input field with arrows to the right hand side of the field

Now it’s time to apply this knowledge.

**Instructions**

**1.**

In **index.html** we started a <form> for users to make a custom burger. Right now we have a <label> for patties that needs an associated <input> element.

Since we want users to enter a number, create an <input> and set the attributes:

* Associate the <input> to the first <label> by assigning the correct value to id.
* type="number"
* step="1"
* name to "amount".

Hint

To assign attributes to the <input>, add the attributes inside the opening tag of <input> like so:

<input exampleAttribute="value-for-attribute">

To associate the <input> with the created <label> the value of the for attribute must match the value of the id of <input>.

Ans:

<!DOCTYPE html>

<html lang="en" dir="ltr">

  <head>

    <meta charset="utf-8">

    <link rel="stylesheet" type="text/css" href="style.css">

    <link href="https://fonts.googleapis.com/css?family=Rubik" rel="stylesheet">

    <title>Range Input</title>

  </head>

  <body>

    <section id="overlay">

      <img src="https://s3.amazonaws.com/codecademy-content/courses/web-101/unit-6/htmlcss1-img\_burger-logo.svg" alt="Davie's Burgers Logo" id="logo">

      <hr>

      <form>

        <h1>Create a burger!</h1>

        <section class="protein">

          <label for="patty">What type of protein would you like? </label>

          <input type="text" name="patty" id="patty" value="beef">

        </section>

        <hr>

        <section class="patties">

          <label for="amount">How many patties would you like?</label>

          <!--Add your code below-->

          <input id="amount" type="number" step="1" name="amount">

        </section>

      </form>

    </section>

  </body>

</html>

**Range Input**

Using an <input type="number"> is great if we want to allow users to type in any number of their choosing. But, if we wanted to limit what numbers our users could type we might consider using a different type value. Another option we could use is setting type to "range" which creates a slider.

To set the minimum and maximum values of the slider we assign values to the min and max attribute of the <input>. We could also control how smooth and fluid the slider works by assigning the step attribute a value. Smaller step values will make the slider more fluidly, whereas larger step values will make the slider move more noticeably. Take a look at the code to create a slider:

<form>

<label for="volume"> Volume Control</label>

<input id="volume" name="volume" type="range" min="0" max="100" step="1">

</form>

The code above renders:rendered slider for volume control

In the example above, every time the slider moves by one, the value of the <input>‘s value attribute changes.

**Instructions**

**1.**

Let’s give our users an option for how they want to cook their patties. We can do this by adding a slider to the existing <form>.

In the <section> with class="cooked", add an <input> element. Set the id and name to "doneness". Also, set the type attribute to "range".

Since our <form> is getting long, you might have to scroll down to find the provided <section>.

**2.**

For the newly created <input> set the:

* min attribute to "0".
* max attribute to "5".
* step attribute to "0.5".

Hint

Add the attributes to the opening tag of the <input> element.

Ans:

<!DOCTYPE html>

<html lang="en" dir="ltr">

  <head>

    <meta charset="utf-8">

    <link rel="stylesheet" type="text/css" href="style.css">

    <link href="https://fonts.googleapis.com/css?family=Rubik" rel="stylesheet">

    <title>Range Input</title>

  </head>

  <body>

    <section id="overlay">

      <img src="https://s3.amazonaws.com/codecademy-content/courses/web-101/unit-6/htmlcss1-img\_burger-logo.svg" alt="Davie's Burgers Logo" id="logo">

      <hr>

      <form>

        <h1>Create a burger!</h1>

        <section class="protein">

          <label for="patty">What type of protein would you like?</label>

          <input type="text" name="patty" id="patty">

        </section>

        <hr>

        <section class="patties">

          <label for="amount">How many patties would you like?</label>

          <input type="number" name="amount" id="amount">

        </section>

        <hr>

        <section class="cooked">

          <label for="doneness">How do you want your patty cooked</label>

          <br>

          <span>Rare</span>

           <!--Add your code below-->

          <input id="doneness" name="doneness" class="cooked" type="range" min="0" max="5" step="0.5">

          </input>

          <span>Well-Done</span>

        </section>

      </form>

    </section>

  </body>

</html>

**Checkbox Input**

So far the types of inputs we’ve allowed were all single choices. But, what if we presented multiple options to users and allow them to select any number of options? Sounds like we could use checkboxes! In a <form> we would use the <input> element and set type="checkbox". Examine the code used to create multiple checkboxes:

<form>

<p>Choose your pizza toppings:</p>

<label for="cheese">Extra cheese</label>

<input id="cheese" name="topping" type="checkbox" value="cheese">

<br>

<label for="pepperoni">Pepperoni</label>

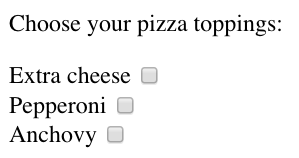
<input id="pepperoni" name="topping" type="checkbox" value="pepperoni">

<br>

<label for="anchovy">Anchovy</label>

<input id="anchovy" name="topping" type="checkbox" value="anchovy">

</form>

Which renders:

Notice in the example provided:

* there are assigned values to the value attribute of the checkboxes. These values are not visible on the form itself, that’s why it is important that we use an associated <label> to identify the checkbox.
* each <input> has the same value for the name attribute. Using the same name for each checkbox groups the <input>s together. However, each <input> has a unique id to pair with a <label>.

Alright, time to use checkboxes in our code!

**Instructions**

**1.**

Time to add some toppings! In the <section> with class="toppings", there are two <label>s but no associated <input> elements. Add an <input> element associated with the first <label>.

The created <input> should have:

* an id set to "lettuce".
* a name attribute with a value of "topping".
* a type set to "checkbox"
* a value of "lettuce".

Hint

In this case, add the <input> before the <label>, so the text appears to the right of the checkbox. Assign the provided attributes inside the opening tag of the <input>.

**2.**

Add another <input> element and associate it with the second <label>.

The <input> element should have:

* an id set to "tomato".
* a type set to "checkbox".
* a name attribute with a value of "topping".
* a value of "tomato".

**3.**

Two choices are good, but it would be better to have even more.

Add another <input type="checkbox"> and <label> pair. Assign the name of the <input> to "topping". You’re free to decide the value and id but make sure that your new <label> and <input> are associated.

Hint

You’ll need to add both a new <input> and an associated <label>.

The type of the <input> must be "checkbox" and the name set to "topping".

You’re free to decide the id, however, the value of the <label>‘s for attribute must match the id of the <input> element. The value of the <input> can be anything you want but it cannot be blank.

Ans:

<!DOCTYPE html>

<html lang="en" dir="ltr">

  <head>

    <meta charset="utf-8">

    <link rel="stylesheet" type="text/css" href="style.css">

    <link href="https://fonts.googleapis.com/css?family=Rubik" rel="stylesheet">

    <title>Checkbox Input</title>

  </head>

  <body>

    <section id="overlay">

      <img src="https://s3.amazonaws.com/codecademy-content/courses/web-101/unit-6/htmlcss1-img\_burger-logo.svg" alt="Davie's Burgers Logo" id="logo">

      <hr>

      <form>

        <h1>Create a burger!</h1>

        <section class="protein">

          <label for="patty">What type of protein would you like?</label>

          <input type="text" name="patty" id="patty">

        </section>

        <hr>

        <section class="patties">

          <label for="amount">How many patties would you like?</label>

          <input type="number" name="amount" id="amount">

        </section>

        <hr>

        <section class="cooked">

          <label for="doneness">How do you want your patty cooked</label>

          <br>

          <span>Rare</span>

          <input type="range" name="doneness" id="doneness" value="3" min="1" max="5">

          <span>Well-Done</span>

        </section>

        <hr>

        <section class="toppings">

          <span>What toppings would you like?</span>

          <br>

          <!--Add your code below for the first checkbox-->

          <input id="lettuce" name="topping" type="checkbox" value="lettuce"> </input>

          <label for="lettuce">Lettuce</label>

          <!--Add your code below for the second checkbox-->

          <input id="tomato" name="topping" type="checkbox" value="tomato"> </input>

          <label for="tomato">Tomato</label>

          <!--Add your code below for the third checkbox-->

          <input id="mashrom" name="topping" type="checkbox" value="mashrom"> </input>

          <label for="mashrom">mashrom</label>

        </section>

      </form>

    </section>

  </body>

</html>

# Radio Button Input

Checkboxes work well if we want to present users with multiple options and let them choose one or more of the options. However, there are cases where we want to present multiple options and only allow for one selection — like asking users if they agree or disagree with the terms and conditions. Let’s look over the code used to create radio buttons:

<form>

<p>What is sum of 1 + 1?</p>

<input type="radio" id="two" name="answer" value="2">

<label for="two">2</label>

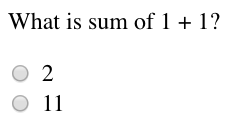
<br>

<input type="radio" id="eleven" name="answer" value="11">

<label for="eleven">11</label>

</form>

Which renders:



Notice from the code snippet, radio buttons (like checkboxes) do not display their value. We have an associated <label> to represent the value of the radio button. To group radio buttons together, we assign them the same name and only one radio button from that group can be selected.

Let’s see this in action by creating our own radio buttons.

**Instructions**

**1.**

We can give our users the option to make the burger into a cheeseburger. Let’s use radio buttons for that.

In <section> element with a class of "cheesy" there are two <label>s that don’t have associated <input> elements. Add an <input> element associated with the first <label>.

The created <input> should have:

* an id set to "yes".
* a type set to "radio"
* a name attribute with a value of "cheese".
* a value of "yes".

Hint

Assign the attributes and values inside the <input> tag.

**2.**

Awesome, now add another <input> element to give users another choice. The created <input> should have:

* an id set to "no".
* a type set to "radio"
* a name attribute with a value of "cheese".
* a value of "no".

<!DOCTYPE html>

<html lang="en" dir="ltr">

  <head>

    <meta charset="utf-8">

    <link rel="stylesheet" type="text/css" href="style.css">

    <link href="https://fonts.googleapis.com/css?family=Rubik" rel="stylesheet">

    <title>Radio Input</title>

  </head>

  <body>

    <section id="overlay">

      <img src="https://s3.amazonaws.com/codecademy-content/courses/web-101/unit-6/htmlcss1-img\_burger-logo.svg" alt="Davie's Burgers Logo" id="logo">

      <hr>

      <form>

        <h1>Create a burger!</h1>

        <section class="protein">

          <label for="patty">What type of protein would you like?</label>

          <input type="text" name="patty" id="patty">

        </section>

        <hr>

        <section class="patties">

          <label for="amount">How many patties would you like?</label>

          <input type="number" name="amount" id="amount">

        </section>

        <hr>

        <section class="cooked">

          <label for="doneness">How do you want your patty cooked</label>

          <br>

          <span>Rare</span>

          <input type="range" name="doneness" id="doneness" value="3" min="1" max="5">

          <span>Well-Done</span>

        </section>

        <hr>

        <section class="toppings">

          <span>What toppings would you like?</span>

          <br>

          <input type="checkbox" name="topping" id="lettuce" value="lettuce">

          <label for="lettuce">Lettuce</label>

          <input type="checkbox" name="topping" id="tomato" value="tomato">

          <label for="tomato">Tomato</label>

          <input type="checkbox" name="topping" id="onion" value="onion">

          <label for="onion">Onion</label>

        </section>

        <hr>

        <section class="cheesy">

          <span>Would you like to add cheese?</span>

          <br>

          <!--Add your first radio button below-->

          <input class="cheesy" id="yes" type="radio" name="cheese" value="yes">

  <label for="yes">Yes</label>

          <!--Add your second radio button below-->

          <input class="cheesy" id="no" type="radio" name="cheese" value="no">

          <label for="no">No</label>

        </section>

      </form>

    </section>

  </body>

</html>

# Dropdown list

Radio buttons are great if we want our users to pick one option out of a few visible options, but imagine if we have a whole list of options! This situation could quickly lead to a lot of radio buttons to keep track of.

An alternative solution is to use a dropdown list to allow our users to choose one option from an organized list. Here’s the code to create a dropdown menu:

<form>

<label for="lunch">What's for lunch?</label>

<select id="lunch" name="lunch">

<option value="pizza">Pizza</option>

<option value="curry">Curry</option>

<option value="salad">Salad</option>

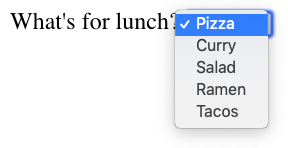
<option value="ramen">Ramen</option>

<option value="tacos">Tacos</option>

</select>

</form>

Which renders:rendered dropdown list with the first option showing

And if we click on the field containing the first option, the list is revealed:

Notice in the code that we’re using the element <select> to create the dropdown list. To populate the dropdown list, we add multiple <option> elements, each with a value attribute. By default, only one of these options can be selected.

The text rendered is the text included between the opening and closing <option> tags. However, it is the value of the value attribute that is used in <form> submission (notice the difference in the text and value capitalization). When the <form> is submitted, the information from this input field will be sent using the name of the <select> and the value of the chosen <option>. For instance, if a user selected Pizza from the dropdown list, the information would be sent as "lunch=pizza".

**Instructions**

**1.**

Let’s now give our users a choice of buns using a dropdown list.

In <section> element with a class of "bun-type" there is a <label> that we can associate a <select> element with.

Add a <select> element with a name of "bun" and an id of "bun".

Hint

Add the <select> element inside the <form> and below the <label>. The <select> element has an opening and closing tag. Add the attributes to the opening tag of <select>.

**2.**

Inside the <select> element, add 3 <option>s.

* The first <option> should have a value of "sesame" and display the text Sesame on the webpage.
* The second <option> should have a value of "potato" and display the text Potato on the webpage.
* The third <option> can be a value that you choose and display text relevant to the value (make sure it’s not empty!)

Hint

To display text on the webpage for an <option>, add the text in between the opening and closing tags of <option>. The value attribute is set in the opening <option> tag.

<!DOCTYPE html>

<html lang="en" dir="ltr">

  <head>

    <meta charset="utf-8">

    <link rel="stylesheet" type="text/css" href="style.css">

    <link href="https://fonts.googleapis.com/css?family=Rubik" rel="stylesheet">

    <title>Dropdown List</title>

  </head>

  <body>

    <section id="overlay">

      <img src="https://s3.amazonaws.com/codecademy-content/courses/web-101/unit-6/htmlcss1-img\_burger-logo.svg" alt="Davie's Burgers Logo" id="logo">

      <hr>

      <form>

        <h1>Create a burger!</h1>

        <section class="protein">

          <label for="patty">What type of protein would you like?</label>

          <input type="text" name="patty" id="patty">

        </section>

        <hr>

  <section class="patties">

          <label for="amount">How many patties would you like?</label>

          <input type="number" name="amount" id="amount">

        </section>

        <hr>

        <section class="cooked">

          <label for="doneness">How do you want your patty cooked</label>

          <br>

          <span>Rare</span>

          <input type="range" name="doneness" id="doneness" value="3" min="1" max="5">

          <span>Well-Done</span>

        </section>

        <hr>

        <section class="toppings">

          <span>What toppings would you like?</span>

          <br>

          <input type="checkbox" name="topping" id="lettuce" value="lettuce">

          <label for="lettuce">Lettuce</label>

          <input type="checkbox" name="topping" id="tomato" value="tomato">

          <label for="tomato">Tomato</label>

          <input type="checkbox" name="topping" id="onion" value="onion">

          <label for="onion">Onion</label>

        </section>

        <hr>

        <section class="cheesy">

          <span>Would you like to add cheese?</span>

          <br>

          <input type="radio" name="cheese" id="yes" value="yes">

          <label for="yes">Yes</label>

          <input type="radio" name="cheese" id="no" value="yes">

          <label for="no">No</label>

        </section>

        <hr>

        <section class="bun-type">

          <label for="bun">What type of bun would you like?</label>

          <!--Add your code below-->

    <select id="bun" name="bun" class="bun-type">

      <option value="sesame">sesame</option>

      <option value="potato">potato</option>

      <option value="salad">Salad</option>

    </select>

        </section>

      </form>

    </section>

  </body>

</html>

# Datalist Input

Even if we have an organized dropdown list, if the list has a lot of options, it could be tedious for users to scroll through the entire list to locate one option. That’s where using the <datalist> element comes in handy.

The <datalist> is used with an <input type="text"> element. The <input> creates a text field that users can type into and filter options from the <datalist>. Let’s go over a concrete example:

<form>

<label for="city">Ideal city to visit?</label>

<input type="text" list="cities" id="city" name="city">

<datalist id="cities">

<option value="New York City"></option>

<option value="Tokyo"></option>

<option value="Barcelona"></option>

<option value="Mexico City"></option>

<option value="Melbourne"></option>

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

</datalist>

</form>

Notice, in the code above, we have an <input> that has a list attribute. The <input> is associated to the <datalist> via the <input>‘s list attribute and the id of the <datalist>.

From the code provided, the following form is rendered:input field with a label 'Ideal city to visit?'

And when field is selected:clicking on the input field reveals a dropdown 
list

While <select> and <datalist> share some similarities, there are some major differences. In the associated <input> element, users can type in the input field to search for a particular option. If none of the <option>s match, the user can still use what they typed in. When the form is submitted, the value of the <input>‘s name and the value of the option selected, or what the user typed in, is sent as a pair.

Now it’s time to make a <datalist> of our own!

**Instructions**

**1.**

Time to add some sauce! Users might get creative with what sauce they choose to put, so let’s use the <datalist> element.

In <section> element with a class of "sauce-selection", we already have the <label> and <input> set up. Add a <datalist> element with an id of "sauces".

**2.**

Inside the <datalist> element, add 3 <option>s.

* The first <option> should have a value of "ketchup".
* The second <option> should have a value of "mayo".
* The third <option> can be a value that you choose and display text relevant to the value (make sure it’s not empty!)

Hint

The <option>s should have opening and closing tags but no text in between. The value attribute is set in the opening <option> tag.

<!DOCTYPE html>

<html lang="en" dir="ltr">

  <head>

    <meta charset="utf-8">

    <link rel="stylesheet" type="text/css" href="style.css">

    <link href="https://fonts.googleapis.com/css?family=Rubik" rel="stylesheet">

    <title>Radio Input</title>

  </head>

  <body>

    <section id="overlay">

      <img src="https://s3.amazonaws.com/codecademy-content/courses/web-101/unit-6/htmlcss1-img\_burger-logo.svg" alt="Davie's Burgers Logo" id="logo">

      <hr>

      <form>

        <h1>Create a burger!</h1>

        <section class="protein">

          <label for="patty">What type of protein would you like?</label>

          <input type="text" name="patty" id="patty" value="beef">

        </section>

        <hr>

        <section class="patties">

          <label for="amount">How many patties would you like?</label>

          <input type="number" name="amount" value="2" id="amount">

        </section>

        <hr>

        <section class="cooked">

          <label for="doneness">How do you want your patty cooked</label>

          <br>

          <span>Rare</span>

          <input type="range" name="doneness" id="doneness" value="3" min="1" max="5">

          <span>Well-Done</span>

        </section>

        <hr>

        <section class="toppings">

          <span>What toppings would you like?</span>

          <br>

          <input type="checkbox" name="topping" id="lettuce" value="lettuce">

          <label for="lettuce">Lettuce</label>

          <input type="checkbox" name="topping" id="tomato" value="tomato">

          <label for="tomato">Tomato</label>

          <input type="checkbox" name="topping" id="onion" value="onion">

          <label for="onion">Onion</label>

        </section>

        <hr>

        <section class="cheesy">

          <span>Would you like to add cheese?</span>

          <br>

          <input type="radio" name="cheese" id="yes" value="yes">

          <label for="yes">Yes</label>

          <input type="radio" name="cheese" id="no" value="yes">

          <label for="no">No</label>

        </section>

        <hr>

        <section class="bun-type">

          <label for="bun">What type of bun would you like?</label>

          <select name="bun" id="bun">

            <option value="sesame">Sesame</option>

            <option value="potato">Potato</option>

            <option value="pretzel">Pretzel</option>

          </select>

        </section>

        <hr>

        <section class="sauce-selection">

          <label for="sauce">What type of sauce would you like?</label>

          <input list="sauces" id="sauce" name="sauce">

          <!--Add your code below-->

        <datalist id="sauces" name="sauces" class="sauce-selection">

            <option value="ketchup"></option>

            <option value="mayo"></option>

            <option value="nil"></option>

        </datalist>

        </section>

# Textarea element

An <input> element with type="text" creates a single row input field for users to type in information. However, there are cases where users need to write in more information, like a blog post. In such cases, instead of using an <input>, we could use <textarea>.

The <textarea> element is used to create a bigger text field for users to write more text. We can add the attributes rows and cols to determine the amount of rows and columns for the <textarea>. Take a look:

<form>

<label for="blog">New Blog Post: </label>

<br>

<textarea id="blog" name="blog" rows="5" cols="30">

</textarea>

</form>

In the code above, an empty <textarea> that is 5 rows by 30 columns is rendered to the page like so:



If we wanted an even bigger text field, we could click and drag on the bottom right corner to expand it.

When we submit the form, the value of <textarea> is the text written inside the box. If we want to add a default value to text to <textarea> we would include it within the opening and closing tags like so:

<textarea>Adding default text</textarea>

This code will render a <textarea> that contains pre-filled text: “Adding default text”.

But don’t just take our word for it, let’s test it out!

**Instructions**

**1.**

We covered a lot of options but users might still have other ideas. Let’s make use of a <textarea> element to give users more freedom.

In <section> element with a class of "extra-info", we have provided a <label> element. Add a <textarea> with the following attributes:

* id and name set to "extra"
* rows set to "3"
* cols set to "40"

Hint

Add the attributes with their values inside the opening <textarea> tag.

**2.**

Now add some default text to the created <textarea>. You may add any text you want, but it cannot be blank!

Hint

To add default text to a <textarea> you would include it between the opening and closing <textarea> tags.

<!DOCTYPE html>

<html lang="en" dir="ltr">

  <head>

    <meta charset="utf-8">

    <link rel="stylesheet" type="text/css" href="style.css">

    <link href="https://fonts.googleapis.com/css?family=Rubik" rel="stylesheet">

    <title>Textarea element</title>

  </head>

  <body>

    <section id="overlay">

      <img src="https://s3.amazonaws.com/codecademy-content/courses/web-101/unit-6/htmlcss1-img\_burger-logo.svg" alt="Davie's Burgers Logo" id="logo">

      <hr>

      <form action="submission.html" method="POST">

        <h1>Create a burger!</h1>

                <section class="protein">

          <label for="patty">What type of protein would you like?</label>

          <input type="text" name="patty" id="patty">

        </section>

        <hr>

        <section class="patties">

          <label for="amount">How many patties would you like?</label>

          <input type="number" name="amount" id="amount">

        </section>

        <hr>

        <section class="cooked">

          <label for="doneness">How do you want your patty cooked</label>

          <br>

          <span>Rare</span>

          <input type="range" name="doneness" id="doneness" value="3" min="1" max="5">

          <span>Well-Done</span>

        </section>

        <hr>

        <section class="toppings">

          <span>What toppings would you like?</span>

          <br>

          <input type="checkbox" name="topping" id="lettuce" value="lettuce">

          <label for="lettuce">Lettuce</label>

          <input type="checkbox" name="topping" id="tomato" value="tomato">

          <label for="tomato">Tomato</label>

          <input type="checkbox" name="topping" id="onion" value="onion">

          <label for="onion">Onion</label>

        </section>

        <hr>

        <section class="cheesy">

          <span>Would you like to add cheese?</span>

          <br>

          <input type="radio" name="cheese" id="yes" value="yes">

          <label for="yes">Yes</label>

          <input type="radio" name="cheese" id="no" value="yes">

          <label for="no">No</label>

        </section>

        <hr>

        <section class="bun-type">

          <label for="bun">What type of bun would you like?</label>

          <select name="bun" id="bun">

            <option value="sesame">Sesame</option>

            <option value="potatoe">Potato</option>

            <option value="pretzel">Pretzel</option>

          </select>

        </section>

        <hr>

        <section class="sauce-selection">

          <label for="sauce">What type of sauce would you like?</label>

          <input list="sauces" id="sauce" name="sauce">

          <datalist id="sauces">

            <option value="ketchup"></option>

            <option value="mayo"></option>

            <option value="mustard"></option>

          </datalist>

        </section>

        <hr>

        <section class="extra-info">

          <label for="extra">Anything else you want to add?</label>

          <br>

          <!--Add your code below-->

          <textarea class="extra-info" name="extra" id="extra" rows="3" cols="40">You may add any text you want, but it cannot be blank!</textarea>

        </section>

      </form>

    </section>

  </body>

</html>

**Submit Form**

Remember, the purpose of a form is to collect information that will be submitted. That’s the role of the submit button — users click on it when they are finished with filling out information in the <form> and they’re ready to send it off. Now that we’ve gone over how to create various input elements, let’s now go over how to create a submit button!

To make a submit button in a <form>, we’re going to use the reliable <input> element and set the type to "submit". For instance:

<form>

<input type="submit" value="Send">

</form>

Which renders:

rendered submit button

Notice in the code snippet that the value assigned to the <input> shows up as text on the submit button. If there isn’t a value attribute, the default text, Submit shows up on the button.

Let’s now add this element to make our <form>s complete!

**Instructions**

**1.**

At the bottom of the <form> inside the element <section class="submission">, add a submit button using the <input> element.

The text inside the submit button should read: Submit.

Hint

To make a submit button, add an <input> element and assign type="submit". To change the text inside the button, set the value attribute to the text you need.

<!DOCTYPE html>

<html lang="en" dir="ltr">

  <head>

    <meta charset="utf-8">

    <link rel="stylesheet" type="text/css" href="style.css">

    <link href="https://fonts.googleapis.com/css?family=Rubik" rel="stylesheet">

    <title>Textarea element</title>

  </head>

  <body>

    <section id="overlay">

      <img src="https://s3.amazonaws.com/codecademy-content/courses/web-101/unit-6/htmlcss1-img\_burger-logo.svg" alt="Davie's Burgers Logo" id="logo">

      <hr>

      <form action="submission.html" method="POST">

        <h1>Create a burger!</h1>

          <section class="protein">

          <label for="patty">What type of protein would you like?</label>

          <input type="text" name="patty" id="patty">

        </section>

        <hr>

        <section class="patties">

          <label for="amount">How many patties would you like?</label>

          <input type="number" name="amount" id="amount">

        </section>

        <hr>

        <section class="cooked">

          <label for="doneness">How do you want your patty cooked</label>

          <br>

          <span>Rare</span>

          <input type="range" name="doneness" id="doneness" value="3" min="1" max="5">

          <span>Well-Done</span>

        </section>

        <hr>

        <section class="toppings">

          <span>What toppings would you like?</span>

          <br>

          <input type="checkbox" name="topping" id="lettuce" value="lettuce">

          <label for="lettuce">Lettuce</label>

          <input type="checkbox" name="topping" id="tomato" value="tomato">

          <label for="tomato">Tomato</label>

          <input type="checkbox" name="topping" id="onion" value="onion">

          <label for="onion">Onion</label>

        </section>

        <hr>

        <section class="cheesy">

          <span>Would you like to add cheese?</span>

          <br>

          <input type="radio" name="cheese" id="yes" value="yes">

          <label for="yes">Yes</label>

          <input type="radio" name="cheese" id="no" value="yes">

          <label for="no">No</label>

        </section>

        <hr>

        <section class="bun-type">

          <label for="bun">What type of bun would you like?</label>

          <select name="bun" id="bun">

            <option value="sesame">Sesame</option>

            <option value="potatoe">Potato</option>

            <option value="pretzel">Pretzel</option>

          </select>

        </section>

        <hr>

        <section class="sauce-selection">

          <label for="sauce">What type of sauce would you like?</label>

          <input list="sauces" id="sauce" name="sauce">

          <datalist id="sauces">

            <option value="ketchup"></option>

            <option value="mayo"></option>

            <option value="mustard"></option>

          </datalist>

        </section>

        <hr>

        <section class="extra-info">

          <label for="extra">Anything else you want to add?</label>

          <br>

          <textarea id="extra" name="extra" rows="3" cols="40"></textarea>

        </section>

        <hr>

        <section class="submission">

          <!--Add your code below-->

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

        </section>

      </form>

    </section>

  </body>

</html>

# Review

Nice work interacting with the extremely common and useful <form> element!

In this lesson we went over:

* The purpose of a <form> is to allow users to input information and send it.
* The <form>‘s action attribute determines where the form’s information goes.
* The <form>‘s method attribute determines how the information is sent and processed.
* To add fields for users to input information we use the <input> element and set the type attribute to a field of our choosing:
  + Setting type to "text" creates a single row field for text input.
  + Setting type to "password" creates a single row field that censors text input.
  + Setting type to "number" creates a single row field for number input.
  + Setting type to "range" creates a slider to select from a range of numbers.
  + Setting type to "checkbox" creates a single checkbox which can be paired with other checkboxes.
  + Setting type to "radio" creates a radio button that can be paired with other radio buttons.
  + Setting type to "list" will pair the <input> with a <datalist> element if the id of both are the same.
  + Setting type to "submit" creates a submit button.
* A <select> element is populated with <option> elements and renders a dropdown list selection.
* A <datalist> element is populated with <option> elements and works with an <input> to search through choices.
* A <textarea> element is a text input field that has a customizable area.
* When a <form> is submitted, the name of the fields that accept input and the value of those fields are sent as name=value pairs.

Using the <form> element in conjunction with the other elements listed above allows us to create sites that take into consideration the wants and needs of our users. Take the opportunity to take what you’ve learned, and apply it!

**Instructions**

If you want to challenge yourself:

* Create a form with multiple fields that accept user input.
* Add two separate sets of radio buttons or checkboxes.
* Search how to add placeholder (hint hint) to an input field. A placeholder shows ups when the field has no value, it disappears after the user types or selects something in the field.

# Introduction to HTML Form Validation

Ever wonder how a login page actually works? Or why the combination of a username and password grants you access to a website? The answers lie in validation. Validation is the concept of checking user provided data against the required data.

There are different types of validation. One type is server-side validation, this happens when data is sent to another machine (typically a server) for validation. An example of this type of validation is the usage of a login page. The form on the login page accepts username and password input, then sends the data to a server that checks that the pair matches up correctly.

On the other hand, we use client-side validation if we want to check the data on the browser (the client). This validation occurs before data is sent to the server. Different browsers implement client-side validation differently, but it leads to the same outcome.

Shared among the different browsers are the benefits of using HTML5’s built-in client-side validation. It saves us time from having to send information to the server and wait for the server to send back confirmation or rejection of the data. This can also help us protect our server from malicious code or data from a malicious user. It also allows us to quickly give feedback to users for specific fields rather than having them fill in a form again if the data they input into the form was rejected.

In this lesson, we’ll learn how to add some validation checks to our <form>s!

# Requiring an Input

Sometimes we have fields in our <form>s which are not optional, i.e. there must be information provided before we can submit it. To enforce this rule, we can add the required attribute to an <input> element.

Take for example:

<form action="/example.html" method="POST">

<label for="allergies">Do you have any dietary restrictions?</label>

<br>

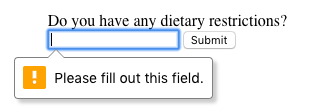
<input id="allergies" name="allergies" type="text" required>

<br>

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

</form>

This renders a text box, and if we try to submit the <form> without filling it out we get this message:



The styling of the message varies from browser to browser, the picture above depicts the message in a Chrome browser. We’ll also continue to show these messages as they appear in Chrome in later exercises.

Let’s see how it shows up in your browser!

**Instructions**

**1.**

Currently, in the provided <form>, the user can submit it without putting any values inside the input field.

Let’s change that by adding a required attribute to the existing <input>.

After you clear this checkpoint, try submitting the <form> without filling out the fields.

Hint

Add an required attribute to the opening <input> tag, like so:

<input type="number" name="guess" id="guess" required>

<!DOCTYPE html>

<html lang="en" dir="ltr">

  <head>

    <meta charset="utf-8">

    <title>Number Guessing</title>

    <link rel="stylesheet" href="style.css" type="text/css">

    <link href="https://fonts.googleapis.com/css?family=Spicy+Rice" rel="stylesheet">

  </head>

  <body>

    <section class="overlay">

      <h1>Guess the right number!</h1>

      <form action="check.html" method="GET">

        <!--Add a required attribute to the input element-->

        <label for="guess">Enter a number between 1-10:</label>

        <br>

        <input type="number" name="guess" id="guess" required>

        <br>

        <input type="submit" id="submission" value="Submit">

      </form>

    </section>

  </body>

</html>

**Set a Minimum and Maximum**

Another built-in validation we can use is to assign a minimum or maximum value for a number field, e.g. <input type="number"> and <input type="range">. To set a minimum acceptable value, we use the min attribute and assign a value. On the flip side, to set a maximum acceptable value, we assign the max attribute a value. Let’s see this in code:

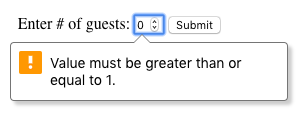
<form action="/example.html" method="POST">

<label for="guests">Enter # of guests:</label>

<input id="guests" name="guests" type="number" min="1" max="4">

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

</form>

If a user tries to submit an input that is less than 1 a warning will appear:

A similar message will appear if a user tries to input a number greater than 4. Let’s now see this action.

**Instructions**

**1.**

Time to enforce the rules of the guessing game.

In the opening <input> tag, set:

* The min attribute to "1"
* The max attribute to "10"

Hint

Make sure you add the attributes to the opening tag, the ordering of the attributes in the opening tag doesn’t matter.

<input type="number" name="guess" id="guess" required min="1" max="10">

# Checking Text Length

In the previous exercise, we were able to use min and max to set acceptable minimum and maximum values in a number field. But what about text fields? There are certainly cases where we wouldn’t want our users typing more than a certain number of characters (think about the character cap for messages on Twitter). We might even want to set a minimum number of characters. Conveniently, there are built-in HTML5 validations for these situations.

To set a minimum number of characters for a text field, we add the minlength attribute and a value to set a minimum value. Similarly, to set the maximum number of characters for a text field, we use the maxlength attribute and set a maximum value. Let’s take a look at these attributes in code:

<form action="/example.html" method="POST">

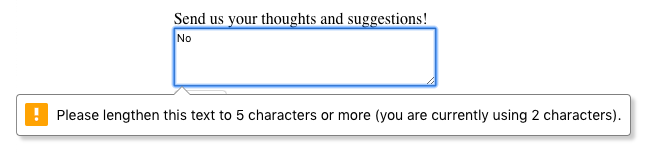
<label for="summary">Summarize your feelings in less than 250 characters</label>

<input id="summary" name="summary" type="text" minlength="5" maxlength="250" required>

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

</form>

If a user tries to submit the <form> with less than the set minimum, this message appears:



And if a user tries to type in more than the maximum allowed number of characters, they don’t get a warning message, but they can’t type it in!

Let’s add this validation to our <form>.

**Instructions**

**1.**

For the login <form>, we want our users to have usernames that are at least 3 characters and at most 15. Let’s add this validation to our current <form>.

To the <input> with an id of "username", add the following attributes and values:

* a minlength of "3"
* a maxlength of "15"

Hint

Set the attributes in the opening <input> tag. Make sure you add the attributes to the right <input> (the one that has an id of "username").

**2.**

We also want passwords to have at least 8 characters and at most 15.

So, in the opening <input> tag that has an id of "pw", add:

* a minlength of "8"
* a maxlength of "15"

<!DOCTYPE html>

<html lang="en" dir="ltr">

  <head>

    <meta charset="utf-8">

    <title>Sign Up Page</title>

    <link rel="stylesheet" href="style.css" type="text/css">

    <link href="https://fonts.googleapis.com/css?family=Fjalla+One" rel="stylesheet">

  </head>

  <body>

    <section class="overlay">

      <h1>Sign Up</h1>

      <p>Create an account:</p>

      <form action="submission.html" method="GET">

        <!--Add the minlength and maxlength attributes to the input fields-->

        <label for="username">Username:</label>

        <br>

        <input id="username" name="username" type="text" required minlength="3" maxlength="15">

        <br>

        <label for="pw">Password:</label>

        <br>

        <input id="pw" name="pw" type="password" required minlength="8" maxlength="15">

        <br>

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

      </form>

    </section>

  </body>

</html>

# Matching a Pattern

In addition to checking the length of a text, we could also add a validation to check how the text was provided. For cases when we want user input to follow specific guidelines, we use the pattern attribute and assign it a regular expression, or regex. Regular expressions are a sequence of characters that make up a search pattern. If the input matches the regex, the form can be submitted.

Let’s say we wanted to check for a valid credit card number (a 14 to 16 digit number). We could use the regex: [0-9]{14,16} which checks that the user provided only numbers and that they entered at least 14 digits and at most 16 digits.

To add this to a form:

<form action="/example.html" method="POST">

<label for="payment">Credit Card Number (no spaces):</label>

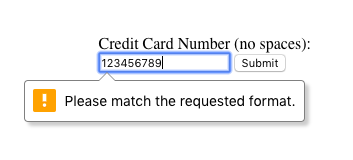
<br>

<input id="payment" name="payment" type="text" required pattern="[0-9]{14,16}">

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

</form>

With the pattern in place, users can’t submit the <form> with a number that doesn’t follow the regex. When they try, they’ll see a validation message like so:



If you want to find out more about Regex, read more at [MDN’s regex article](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Regular_Expressions).

To see it in practice, let’s use the pattern attribute in our HTML!

**Instructions**

**1.**

We might also want to limit usernames to only letters and numbers (and not special characters like ! or @).

To add this validation, add a pattern attribute and set it to: "[a-zA-Z0-9]+" in the first <input> element.

Hint

Make sure you use include the entire regex as provided. It should look like: pattern="[a-zA-Z0-9]+" inside the opening <input> tag. You’re setting this attribute for the <input> that has a type="username".

<input id="username" name="username" type="text" required minlength="3" maxlength="15" pattern="[a-zA-Z0-9]+">

# Review

Awesome job adding client-side validation to <form>s!

Let’s quickly recap:

* Client-side validations happen in the browser before information is sent to a server.
* Adding the required attribute to an input related element will validate that the input field has information in it.
* Assigning a value to the min attribute of a number input element will validate an acceptable minimum value.
* Assigning a value to the max attribute of a number input element will validate an acceptable maximum value.
* Assigning a value to the minlength attribute of a text input element will validate an acceptable minimum number of characters.
* Assigning a value to the maxlength attribute of a text input element will validate an acceptable maximum number of characters.
* Assigning a regex to pattern matches the input to the provided regex.
* If validations on a <form> do not pass, the user gets a message explaining why and the <form> cannot be submitted.

These quick checks help ensure that input data is correct and safe for our servers. It also helps give users immediate feedback on what they need to fix instead of having to wait for a server to send back that information.