Form a Story

Forms are great for collecting information on users, like job applications or insightful surveys. However, we can also stretch our creative muscles and have a little fun with forms. For this project, we’ll use our knowledge of the HTML <form> and grab user input to put a spin on a classic story!

The logic for parsing and inserting of user inputs is already taken care of in main.js using JavaScript . We’ve also added some styling to the page in style.css. You can find these files by click on the folder icon located at the top of the code editor and selecting the files you’re interested in.

What you have to do is now make a <form> capable of collecting the correct information so that the newly crafted story makes sense!

Note: Save your code as you follow the steps to see your progress!

If you get stuck during this project, check out the project walkthrough video which can be found in the help menu.

Adding The Form

1.

We’ll be collecting information from our users using a <form> so, first, we have to add a <form> under the <hr> element inside the <body> of index.html.

2.

Assign the <form> an action of "story.html" and a method of "GET".

We’ll be sending information from our form to story.html using a GET request.

Stuck? Get a hint

3.

In the <form>, create a submit button by adding an <input> with a type of "submit". Assign the value a value of "Form My Story!". Save your code to see the button rendered.

This might seem counterintuitive, but by creating this submit button and submitting the form as you add more code, you can see how the final result of the story evolves and debug in smaller chunks. Check the hint for more debugging tips!

Stuck? Get a hint

Adding Text Input

4.

Now we can populate our <form> with <input> elements that allow users to type in their responses. We’ll also want to have associated <label>s with these <input> elements so users know what they’re filling in.

Add a <label> element and assign the for attribute a value of "animal-1". Have the <label> render the text Animal: on the webpage.

Write your code so that the submit button always shows up at the bottom of the <form>. As you add more code, the submit button should be kept at the bottom.

Stuck? Get a hint

5.

Now we can create an <input> to associate our <label> element with.

Set the id of the <input> to "animal-1" and the type to "text". Assign the name to "animal-1". Remember, the name attribute is needed for information from this <input> to be sent with the form during submission.

Speaking of submission, since we want our users to put in some value, add the required attribute to the <input>.

Stuck? Get a hint

6.

We’re going to be adding more <label>s and <input>s so we should add some spacing.

Below the <input> element, add a line break using <br>.

With the first input field and working submit button, type some text into the field and submit it! Remember, in order for you to see the new code rendered on the browser, you’re going to need to save your code.

Stuck? Get a hint

7.

Our story has quite a few blanks, so we’re going to need a lot more <label>s and <input>s.

Add another <label> with a for attribute of "animal-2" that renders Another Animal: to the webpage.

Underneath the <label>, create a new <input> with the attributes:

id and name set to "animal-2".

type to "text".

required

Add another <br> for a line break.

Stuck? Get a hint

8.

There’s another animal in our story, so let’s add another <label> with a for attribute of "animal-3" that renders One More Animal: to the webpage.

Then, add a new <input> with the attributes:

id and name set to "animal-3".

type to "text".

required

Remember to add another <br> for a line break.

9.

Let’s have our users provide an adjective.

Add another <label> with a for attribute of "adj-1" that renders the text Adjective (ends in -ed): to the webpage.

Then, add a new <input> with the attributes:

id and name set to "adj-1".

type to "text".

required

Add another line break.

10.

Let’s get a verb.

Create a <label> with has a for attribute set to "verb-1" that renders the text Verb (ends in -ing):.

Follow the <label> with an <input> with the attributes:

id and name set to "verb-1".

type to "text".

required

Also, add a line break.

Other Inputs

11.

Great, we have some <input>s set up that accept text, but we can use some <input> with different types in our <form>.

Let’s add a field that will accept a number. First add <label> with a for attribute of "num-1"that renders the text Number: to the webpage.

After the <label> element, add an <input> that has:

id and name set to "num-1".

a type attribute of "number".

a required attribute.

Also, add a line break.

Stuck? Get a hint

12.

One of our blanks requires a “yes” or “no” answer— sounds like we can use some radio buttons for that.

Before we can add the radio buttons, add a <span> element that has the text Yes or No:.

Stuck? Get a hint

13.

For radio buttons, we want to add the <input> before the <label> to render the radio button on the left of the text.

Add an <input> element with the following attributes:

an id set to "yes".

a type with a value of "radio".

a name of "answer".

a value of "yes".

a required attribute.

Under the <input>, add a <label> with a for attribute assigned a value of "yes" that renders the text Yes on the webpage.

Stuck? Get a hint

14.

We should now add a radio button that represents the “no” choice.

Add another <input> element that has the following attributes:

an id set to "no".

a type with a value of "radio".

a name of "answer".

a value of "no".

Under the just added <input>, add another <label> with a for attribute assigned a value of "no" that renders the text No on the webpage.

This time, add a line break!

Stuck? Get a hint

15.

The story that we’re creating this <form> for involves some sort of speed, so let’s give our users a dropdown list of speed options.

Create a <label> and set the for attribute to "speed". The <label> should render the text: Relative speed (ends in -er):

Then add a <select> element with an id and name of "speed". Add the required attribute to make this field mandatory.

Insert a <br> after the closing <select> tag. The <select> element will be empty for now.

Stuck? Get a hint

16.

Inside the <select> add a few <option>s for users to choose from.

One example of an <option> you could include is:

<option value="faster">Faster</option>

Add as many or as few as you’d like!

Remember to assign a value and include text between the opening and closing <option> tags.

Stuck? Get a hint

17.

One of our story’s blank requires a quote. We could provide our users with a few options but also give them the choice of adding their own custom quote. To do this, we can use the <datalist> element.

To set up the <datalist> we need an accompanying <label> and <input>.

Under the last <br>, add a <label> with a for attribute of "quote". Have the <label> render the text Motivational Quote:.

After the <label>, add an <input> with the follow attributes:

an id and name assigned a value of "quote".

the type set to "text".

a required attribute.

a list attribute of "quote-choices".

Stuck? Get a hint

18.

Awesome, we should add the <datalist> now under the <input> element. Assign the <datalist> an id of "quote-choices".

Add a <br> after the closing <datalist> tag.

Stuck? Get a hint

19.

Let’s add a few <option>s with values within the <datalist> element. For example:

<option value="winner gets ice cream!"></option>

Stuck? Get a hint

20.

Every good story has a key takeaway, so let’s finish off this <form> by having our users provide this message!

Add a <label> that has a for attribute of "message". Have the <label> render Meaningful Message: on the web page.

Under the <label> add a <textarea> that has an id and name of "message". Make the <textarea> a required field. The <textarea> should have 8 rows and 40 columns. (Check the hint for a syntax reminder).

Then add a line break after the <textarea> element.

Stuck? Get a hint

Project Extras

21.

Fantastic job creating a <form> to fill in a story!

If you want to challenge yourself:

Add pre-selected values for each input field.

Add placeholder text that contains examples for users.

Add some extra validations like min, minlength, or pattern to the elements that accept user input.