








Lesson 07 - HTML Forms


We're nearing the end! You've learned how to display text, images, group things together, make lists, and even link to other resources!



We've been pretty focused on displaying information to the user, but how would we *take in* input from the user?





Introducing: Forms!


 What's happening?
      Tweet


Create Post ×





 

 Photo/Video  Tag Friends  Feeling/Activ... 

☒  News Feed

 Public ▼

☐  Your Story

 Friends ▼

Post



Sign in

Continue to Gmail

Email or phone

|

[Forgot email?](#)

Not your computer? Use Guest mode to sign in privately.

[Learn more](#)

[Create account](#)

Next

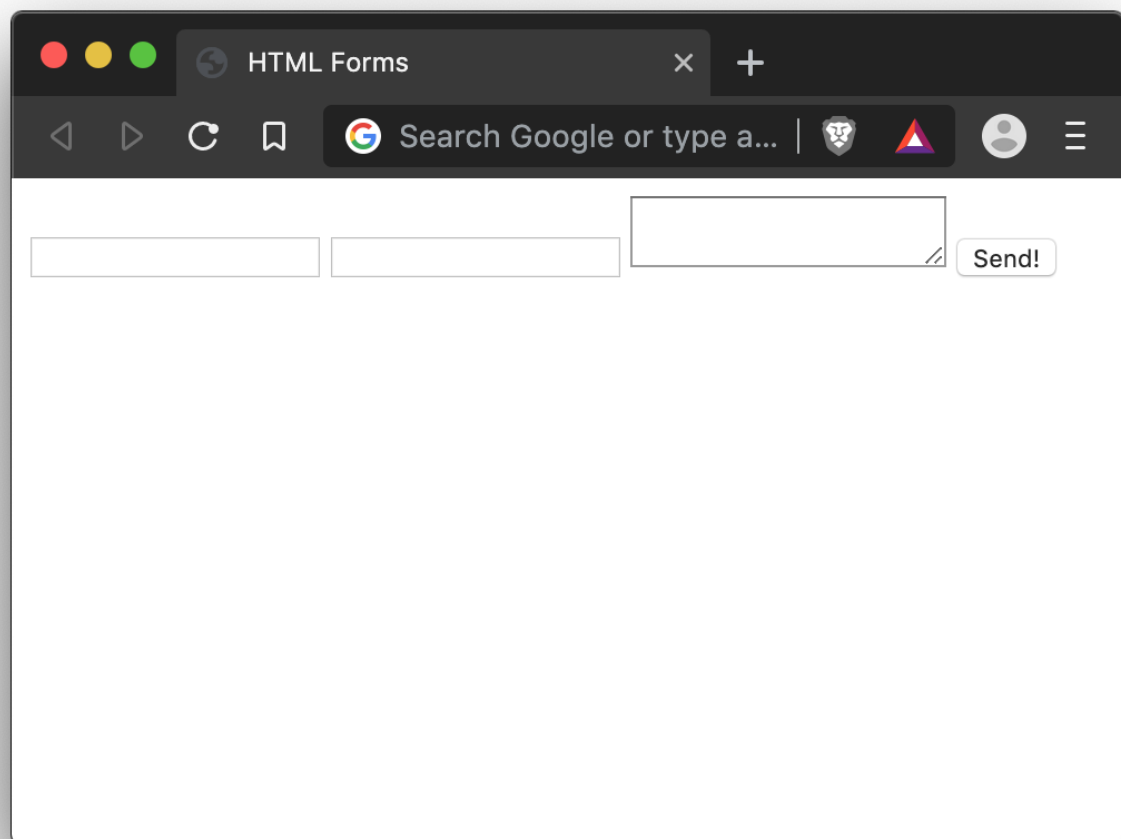
A screenshot of a web-based email composer window titled "New Message". The window has a dark header bar with standard window controls (minimize, maximize, close) on the right. Below the header, there are two input fields: "To" and "Subject". The "To" field has "Cc Bcc" links to its right. The "Subject" field is empty. The main body of the window is a large, empty text area for composing the message. At the bottom, there is a rich text editor toolbar with various icons for undo, redo, font face (set to "Sans Serif"), font size, bold, italic, underline, text color, background color, bulleted list, numbered list, link, unlink, quote, insert link, and insert image. Below the toolbar is a row of icons for text alignment, link, unlink, emoji, insert image, and insert video. A blue "Send" button is on the left, and a trash icon is on the right.

Above are some examples of forms across various popular services. Composing a tweet? You're using a form. Logging in? Yep it's a form. Composing an email? You guessed it. Form.

Let's see what a simple email composer would look like:

```
<form action="send-email.php" method="post">
  <input type="email" name="email" />
  <input type="text" name="subject" />
  <textarea name="body"></textarea>
  <input type="submit" value="Send!" />
</form>
```

This produces the following form



It's pretty plain and hard to see what's going on. There's a few things we immediately notice:

- Form inputs are displayed **inline**
- We can't tell what the inputs are for
- One `<input />` produces a button, not a text input

Let's go over what goes into HTML forms and see how we can improve this example

`<form>`

`<form>` is the root container for HTML forms. A `<form>` element contains one or many input's as children. When the form is submitted it gathers all child inputs that contain a **name** attribute and sends it to the url specified in the form's **action** attribute.

Attributes

- **action**
 - The uri of the server that will process the request
- **method**
 - The HTTP method that the browser uses to submit the form
 - Can be either **GET** or **POST**

- get: corresponds to the HTTP GET method; form data are appended to the action attribute URI with a '?' as separator, and the resulting URI is sent to the server
- post: corresponds to the HTTP POST method; form data are included in the body of the form and sent to the server
- **enctype**
 - when the value of the method attribute is **POST**, enctype is the MIME type of content that is used to submit the form to the server. Possible values:
 - Default: **application/x-www-form-urlencoded**
 - To send file content: **multipart/form-data**
 - Just text: **text/plain**
 - If this attribute is omitted, **application/x-www-form-urlencoded** will be used

<input />

The **<input />** tag is quite versatile. It could vary from a text input, to a button, to a color picker. What it does depends on the **type** attribute provided.

Attributes

- **name**
 - When an input is sent to a server, it is sent in a **key: value** style format so that the server knows what text belongs to what input
 - The **name** attribute acts as the key in this case, and whatever is entered in the input is the value (or whatever the **value** attribute is set to)
- **disabled**
 - A Boolean attribute which is present if the input should be disabled
- **required**
 - Ensures the input to be non-empty before the form is submitted
- **value**
 - The inputs current value
- **placeholder**
 - Shows as a lighter text inside the input. Automatically disappears once text is entered
- **type**
 - Specifies what type of input should be accepted
 - If no **type** attribute is provided, **text** is used
 - Values can be (from [MDN](#)):
 - **text** - A single-line text field. Line-breaks are automatically removed from the input value.
 - **button** - A push button with no default behavior.
 - **checkbox** - A check box allowing single values to be selected/deselected.
 - **color** - A control for specifying a color. A color picker's UI has no required features other than accepting simple colors as text.
 - **date** - A control for entering a date (year, month, and day, with no time).
 - **datetime** - A control for entering a date and time, with no time zone.
 - **email** - A field for editing an e-mail address.
 - **file** - A control that lets the user select a file. Use the accept attribute to define the types of files that the control can select.
 - **hidden** - A control that is not displayed but whose value is submitted to the server.

- **image** - A graphical submit button. You must use the src attribute to define the source of the image and the alt attribute to define alternative text. You can use the height and width attributes to define the size of the image in pixels.
- **month** - A control for entering a month and year, with no time zone.
- **number** - A control for entering a number.
- **password** - A single-line text field whose value is obscured. Use the maxlength and minlength attributes to specify the maximum length of the value that can be entered.
- **radio** - A radio button, allowing a single value to be selected out of multiple choices.
- **range** - A control for entering a number whose exact value is not important.
- **reset** - A button that resets the contents of the form to default values.
- **search** - A single-line text field for entering search strings. Line-breaks are automatically removed from the input value.
- **submit** - A button that submits the form.
- **tel** - A control for entering a telephone number.
- **time** - A control for entering a time value with no time zone.
- **url** - A field for entering a URL.
- **week** - A control for entering a date consisting of a week-year number and a week number with no time zone.

Common Input uses

Checkboxes

- The input elements of type checkbox are rendered by default as square boxes that are checked (ticked) when activated
- They allow you to select single values for submission in a form (or not).
- Radio buttons are similar to checkboxes, but with an important distinction:

radio buttons are grouped into a set in which only one radio button can be selected at a time, whereas checkboxes allow you to turn single values on and off. Where multiple controls exist, radio buttons allow one to be selected out of them all, whereas checkboxes allow multiple values to be selected.

- To select multiple values we need to use the same name input
- Use the value attribute to set each input selected value
- The checked attribute and value works the same way as radio buttons
- [MDN input type checkbox doc](#)

```
<form>
  Select your favourite Harry Potter wizzards
  <input type="checkbox" name="wizards" value="harry potter" checked />
  Harry Potter
  <input type="checkbox" name="wizards" value="hermione granger" checked />
  Hermione Granger
  <input type="checkbox" name="wizards" value="ron weasley" />
  Ron Weasley
  <input type="checkbox" name="wizards" value="aberforth dumbledore" />
```

Aberforth Dumbledore

```
</form>
```

Radio Buttons

- The input type radio elements are generally used in radio groups—collections of radio buttons describing a set of related options
- Only one radio button in a given group can be selected at the same time
- Radio buttons are typically rendered as small circles, which are filled or highlighted when selected
- To define a radio group each of the radio buttons in the group must have same name attribute
- The user will see a legend next to each radio button by using a label element or just plain text
- [MDN input type radio doc](#)

```
<form>
  Are you a Jets Fan?
  <input type="radio" name="jetsfan" value="true" /> Oh Yeahhhhhh, Go Jets!!!
  <input type="radio" name="jetsfan" value="false" /> No
</form>
```

Submit Button

- The input type submit is just a button that submits the form
- This type of element's value attribute contains the button's label
- If you don't specify a value, the button will have a default label, chosen by the user agent
- [MDN input submit doc](#)

```
<form>
  <input type="submit" value="Submit this form" />
</form>
```

<textarea>

The textarea element represents a multi-line plain-text editing control. This element has an opening and close tag. Unlike the `<input />` element, `<textarea>` has its value set based on what's inside its opening and closing tags.

```
<textarea name="body">This would be what is inside the textarea</textarea>
```

<label>

The label is an element that we can use in conjunction with an input to provide context as to what the input is for. Labels have an `inline` layout by default.

```
<form>
  <label>First Name:</label>
  <input type="text" name="first_name" />
</form>
```

Now you might be thinking, can't we just use a `<p>` tag to say what the input is for? Well, you could but there are a few reasons not to:

1. `<label>` is more *semantically* correct when it comes to accessibility in that it can inform the user that this label is for a certain input
2. Related to the above, we can directly tell a label that it is for a certain input using the `for=""` attribute on the label. This will also make it so when you click on the label it focuses the input (places your cursor inside of the input).

```
<form>
  <label for="first_name">Name:</label>
  <input type="text" name="first_name" id="first_name" />
</form>
```

In order for the label to know which input to focus, we need to have the label's `for` attribute match the `id` attribute set on the input.

`<button>`

If you need a button but do not want it to submit the form, instead of using `<input type="submit" value="Submit!" />` we could use a `<button>` instead.

The button element represents a clickable button, which can be used in forms, or anywhere in a document that needs simple, standard button functionality

```
<button>Click me!</button>
```

`<fieldset>` & `<legend>`

The fieldset element is used to group several controls as well as labels within a web form.

The legend element represents a caption for the content of its parent fieldset.

```
<form>
  <fieldset>
    <legend>User Form</legend>
    <label>First Name:</label>
    <input type="text" name="first_name" />
    <label>Last Name:</label>
    <input type="text" name="last_name" />
  </fieldset>
</form>
```



```
</fieldset>
</form>
```

Exercise Instructions

- Create two files, `index.html` and `create-user.html`

index.html

- Add a Document Title: Create User
- Add a Main Title: Create User
- Add an Horizontal Rule
- Create a form element with the following attributes:
 - action: create-user.html
 - method: get
- Inside the form create the following inputs all one above the other, not inline with eachother:
 - Create an email input with the following attributes:
 - name: email
 - placeholder: email
 - Add a label for this input
 - Make the input required
 - Create a text input with the following attributes:
 - name: username
 - placeholder: username
 - size: 30
 - maxlength: 12
 - Add a label for this input
 - Create a password input with the following attributes:
 - name: password
 - size: 30
 - maxlength: 8
 - placeholder: password
 - Add a label for this input
 - Make the input required
 - Create a textarea input with the following attributes:
 - name: bio
 - placeholder: give a brief description of this user
 - Add a label for this input ABOVE the textarea, not on the same line
 - Create a radio button group with the following values:
 - Description: Select your favourite superhero
 - Inputs values and description: Superman, Batman, Wonder Woman, Flash, Thor, Hulk
 - Default selected superhero Wonder Woman
 - Add a label for this input
 - We should only be able to select one of these at a time
 - Create a Checkbox group with the following values:

- Description: Select your favourite video game
- Inputs values and description: Fifa, NHL, Resident Evil, Uncharted
- Default selected video games should be Fifa and Uncharted
- Inputs name should be videogames
- Add a label for this input
- Place a single `<fieldset>` around the checkbox AND radio buttons. Add a `<legend>` "Interests"
- Create a hidden input with the following attributes:
 - name: userid
 - value: 2300
- Clicking on any of the above labels should focus the proper inputs
- Create a submit button with the value "Create User"

create-user.html

- Add a Document Title: Created User
- Add a Main Title: Created User
- Add an Horizontal Rule
- Add a paragraph with the following text: Now you can login to our site
- Create a fake link (`href="#"`) around the login word from the previous paragraph

Exercise Result

Create User

Email:

Username:

Password:

Bio:

Interests

Select your favourite superhero ☐ Superman ☐ Batman ☒ Wonder Woman ☐ Flash ☐ Thor ☐ Hulk

Select your favourite video game ☒ Fifa ☐ NHL ☐ Resident Evil ☒ Uncharted

Created User

Now you can [login](#) to our site

Further Reading

- [MDN form element](#)
- [MDN input element](#)
- [MDN textarea element](#)
- [MDN label element](#)
- [MDN button element](#)