Forms

Table of Contents

[User Experience](#_uswqzoalgz7f)

[The Form Tag](#_510pzaxh4y4k)

[Fieldsets, Legends, and Common Form Layouts](#_oljqs8ci8ju3)

[Labels and Inputs](#_u8h9j23bskmx)

[HTML5 Input Types](#_k0473vyer4my)

[Checkboxes and Radiobuttons](#_5z78yjsk5l17)

[Buttons](#_2h4g5sit0oq3)

[Styling Forms](#_y8j6rz2exfql)

## 

## User Experience

Have you ever been frustrated filling out a form online? Of course. We all have. A lot of money has been lost in e-commerce from poor form design. Planning a well laid out form, with HTML5 inputs and labels, is essential to a successful user experience.

## The Form Tag



The form tag has two very important attributes: *method* and *action*.

The **method attribute** defines *how* the data is sent to the server when a form is submitted. *Post* is the method that is used most commonly, but *get* is used too. This will be covered much more indepthly next semester, but if you want to read more, [check out this article](https://www.diffen.com/difference/GET-vs-POST-HTTP-Requests).

The **action attribute** defines *where* the data is sent when a form is submitted. When you submit a form usually you’re taken to some kind of thank you or invoice type page. Again, that’s a topic for next semester, but it’s important to understand this even at a basic level. In case, if the action **attribute** is not defined the action is set to the current page.

## Fieldsets, Legends, and Common Form Layouts

With the form tag come other tags that are intrinsic to a form.

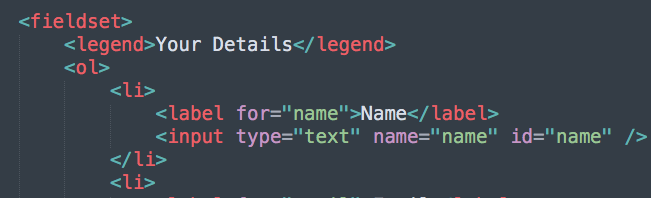


The <fieldset> tag defines a form area where common text fields can be grouped. The <legend> tag provides a heading for that fieldset.



Forms can contain many text fields and headings, most of which have a display of *inline* by default. This can make it challenging to layout a form.

Similar to navigations, using list tags (like <ul> or <ol>) is a common practice for forms to allow for easy layouts.



## Labels and Inputs

<input>, an empty tag (like <img>), is the most commonly used tag within a form. From its attributes we can determine what type of input it will be. The <label> tag works in tandem with the <input> so that when clicked, focus can be sent to the input.





The ***for* attribute** in the <label> is used to link the input and label together. The value in the *for* attribute references the *id* attribute of the input. This is what links them.

The *type* attribute determines what type of input will be used. “text” is the default type for an input. More on these below.

The *name* attribute is used by the server when the form is submitted. It’s used to differentiate and assign keys to the values that are passed to the server.

In situations where the user is allowed to write multiple lines or paragraphs of text, the <textarea> tag is used instead of input tag:



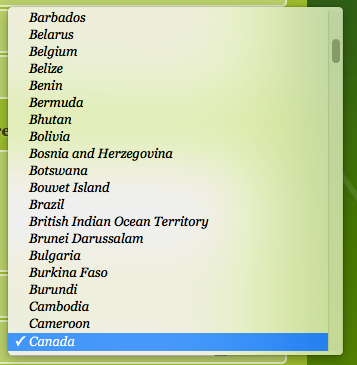


For lists, the <select> tag can be used, with <option> tags within it for each item in the list. Notice the “selected” attribute in the option tag with the Canada as a content, below.





When expanded:



## HTML5 Input Types

There are many input types:

* Text
* Button
* Checkbox
* Radio
* Password
* File

And more. Here’s a full list of [input types](https://www.w3schools.com/tags/att_input_type.asp). That signifies that these types are relatively new and are [HTML5 Input Types](https://www.webfx.com/blog/images/assets/cdn.sixrevisions.com/demos/0345-new_html5_form_input_types/new-html5-form-input-types.html).

Have you ever filled out a form on a mobile device and noticed the keyboard changed? That’s done by using an appropriate input type. For example, if an input type of *number* was used, the mobile keyboard would change to this:

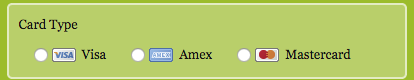


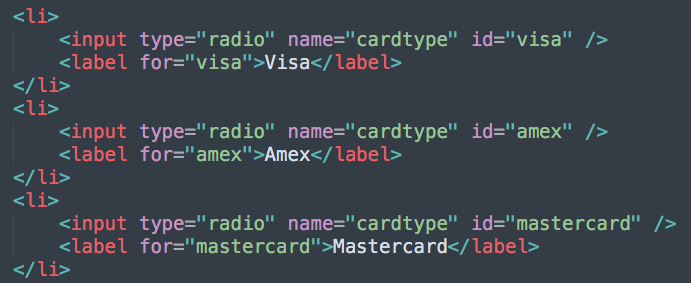
This works with email, tel, and a few other types. There’s no reason not to use these. In fact, they *should* be used. USER EXPERIENCE!

## Checkboxes and Radiobuttons

Data choices are handled in forms using checkboxes and radiobuttons. Checkboxes should be used when multiple choices can be made, and radio buttons should be used when only one choice can be accepted.

Below is an example of using radio buttons for a credit card type selection:





**Notice that all three inputs have the same value for their *name* attribute.** This is what groups them! If they had different values then multiple radio buttons could be selected, which would defeat the purpose of them.

## Buttons

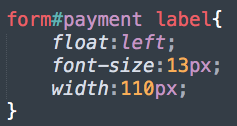
Buttons can be created by either using the <input type=”button”> or by using the <button> tag. Using the button tag makes it a lot easier to style the button, as most <input> tags will have similar styles. The button type is quite a bit different than a regular input, so some CSS styling will have to be done to prevent those styles cascading to the button. Button tags can have tags like img, strong, b and i etc. This is not possible with the input tag. Always, specify the type attribute for the button element to tell the browser what type of button it is.



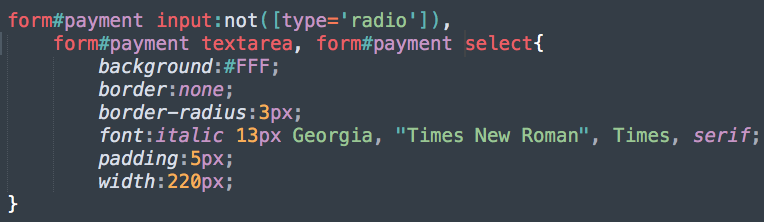


## Styling Forms

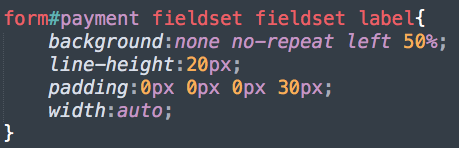
With <input> and <label> being inline it can be challenging to line up form inputs to be visually pleasing. Assigning a common width to the labels will aid in lining up inputs.



All inputs, apart from the radio buttons, have the same styles. When these are created, we have to ensure that these styles **don’t** apply to the radio buttons. To do this, the not() pseudo-class is used, where we pass an attribute and value to define what inputs should not be targeted with this style.



If you remember the radio buttons above, you might wonder how those images were inserted. It was all done in CSS:



First, a selector was created to target labels that were inside a fieldset, that was also inside a fieldset. Remember to read selectors right to left!

The background was removed for all of them, but when one was applied, it would be aligned to the left and given a size of 50%.

Then the individual radio buttons were targeted:



By using [for=”credit card type”], the labels were targeted using an attribute and its value. This is a handy way to target tags.