What is HTML Form

HTML Forms are required to collect different kinds of user inputs, such as contact details like name, email address, phone numbers, or details like credit card information, etc.

Forms contain special elements called controls like inputbox, checkboxes, radio-buttons, submit buttons, etc. Users generally complete a form by modifying its controls e.g. entering text, selecting items, etc. and submitting this form to a web server for further processing.

The `<form>` tag is used to create an HTML form. Here's a simple example of a login form:

Example

```
<form>
    <label>Username: <input type="text"></label>
    <label>Password: <input type="password"></label>
    <input type="submit" value="Submit">
</form>
```

The following section describes different types of controls that you can use in your form.

Input Element

This is the most commonly used element within HTML forms.

It allows you to specify various types of user input fields, depending on the `type` attribute. An input element can be of type *text field*, *password field*, *checkbox*, *radio button*, *submit button*, *reset button*, *file select box*, as well as several new input types introduced in HTML5.

The most frequently used input types are described below.

Text Fields

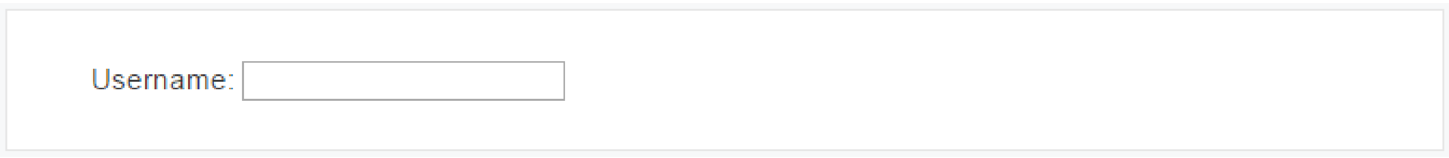
Text fields are one line areas that allow the user to input text.

Single-line text input controls are created using an `<input>` element, whose `type` attribute has a value of `text`. Here's an example of a single-line text input used to take username:

Example

```
<form>
    <label for="username">Username:</label>
    <input type="text" name="username" id="username">
</form>
```

— The output of the above example will look something like this:


Username:

Note: The `<label>` tag is used to define the labels for `<input>` elements. If you want your user to enter several lines you should use a `<textarea>` instead.

Password Field

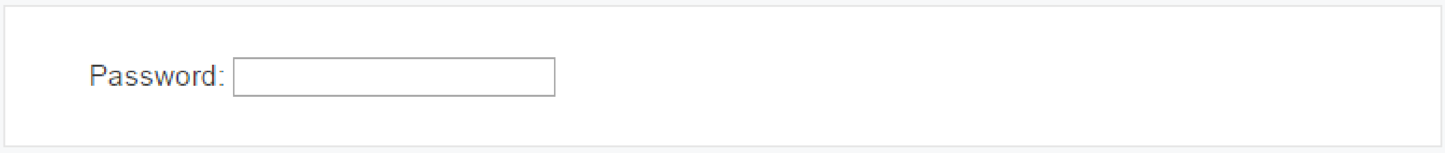
Password fields are similar to text fields. The only difference is; characters in a password field are masked, i.e. they are shown as asterisks or dots. This is to prevent someone else from reading the password on the screen. This is also a single-line text input controls created using an `<input>` element whose `type` attribute has a value of `password`.

Here's an example of a single-line password input used to take user password:

Example

```
<form>
    <label for="user-pwd">Password:</label>
    <input type="password" name="user-password" id="user-pwd">
</form>
```

— The output of the above example will look something like this:



Radio Buttons

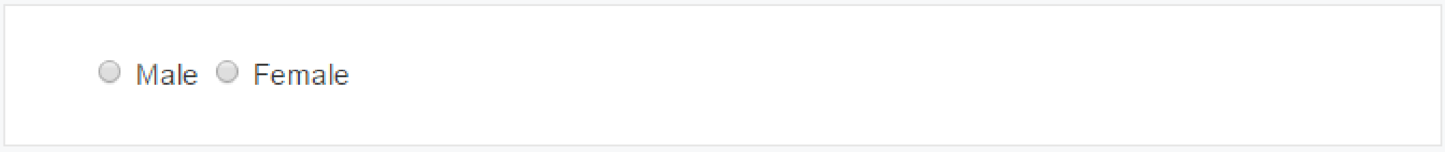
Radio buttons are used to let the user select exactly one option from a pre-defined set of options. It is created using an `<input>` element whose `type` attribute has a value of `radio`.

Here's an example of radio buttons that can be used to collect user's gender information:

Example

```
<form>
    <input type="radio" name="gender" id="male">
    <label for="male">Male</label>
    <input type="radio" name="gender" id="female">
    <label for="female">Female</label>
</form>
```

— The output of the above example will look something like this:



Checkboxes

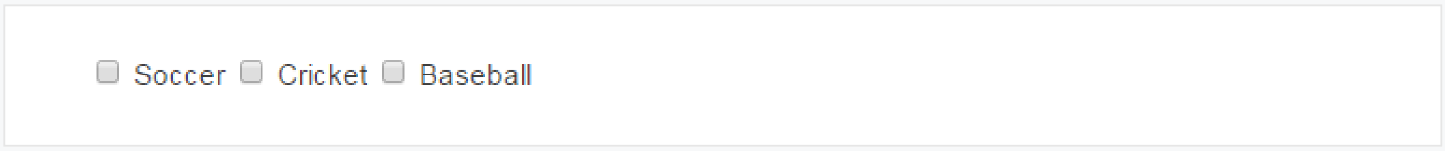
Checkboxes allows the user to select one or more option from a pre-defined set of options. It is created using an `<input>` element whose `type` attribute has a value of `checkbox`.

Here's an example of checkboxes that can be used to collect information about user's hobbies:

Example

```
<form>
    <input type="checkbox" name="sports" id="soccer">
    <label for="soccer">Soccer</label>
    <input type="checkbox" name="sports" id="cricket">
    <label for="cricket">Cricket</label>
    <input type="checkbox" name="sports" id="baseball">
    <label for="baseball">Baseball</label>
</form>
```

— The output of the above example will look something like this:


☐ Soccer ☐ Cricket ☐ Baseball

Note: If you want to make a radio button or checkbox selected by default, you can add the attribute checked to the input element, like `<input type="checkbox" checked>`.

File Select box

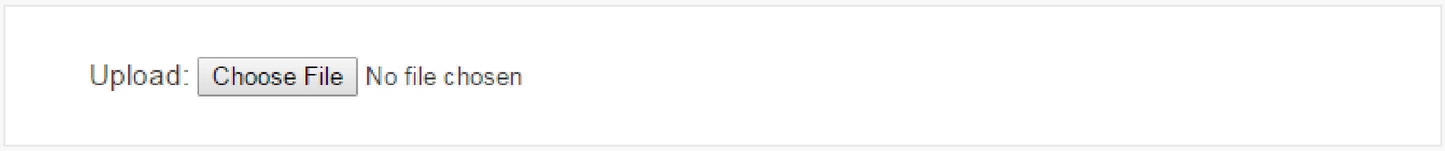
The file fields allow a user to browse for a local file and send it as an attachment with the form data. Web browsers such as Google Chrome and Firefox render a file select input field with a Browse button that enables the user to navigate the local hard drive and select a file.

This is also created using an `<input>` element, whose `type` attribute value is set to `file`.

Example

```
<form>
    <label for="file-select">Upload:</label>
    <input type="file" name="upload" id="file-select">
</form>
```

— The output of the above example will look something like this:


Upload: Choose File No file chosen

Tip: There are several other input types. Please check out the chapter on HTML5 new input types to learn more about the newly introduced input types.

Textarea

Textarea is a multiple-line text input control that allows a user to enter more than one line of text. Multi-line text input controls are created using an `<textarea>` element.

Example

```
<form>
    <label for="address">Address:</label>
    <textarea rows="3" cols="30" name="address" id="address"></textarea>
</form>
```

— The output of the above example will look something like this:

Select Boxes

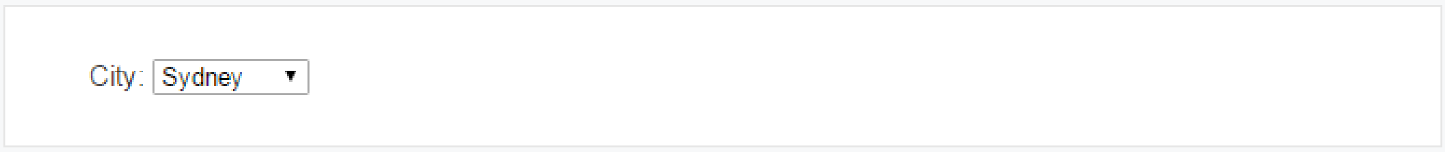
A select box is a dropdown list of options that allows user to select one or more option from a pull-down list of options. Select box is created using the `<select>` element and `<option>` element.

The `<option>` elements within the `<select>` element define each list item.

Example

```
<form>
    <label for="city">City:</label>
    <select name="city" id="city">
        <option value="sydney">Sydney</option>
        <option value="melbourne">Melbourne</option>
        <option value="cromwell">Cromwell</option>
    </select>
</form>
```

— The output of the above example will look something like this:



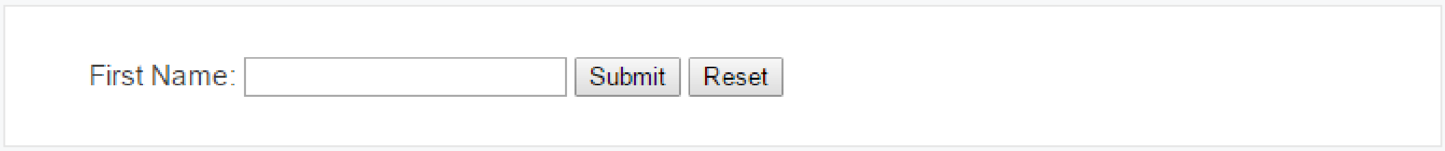
Submit and Reset Buttons

A submit button is used to send the form data to a web server. When submit button is clicked the form data is sent to the file specified in the form's `action` attribute to process the submitted data.

A reset button resets all the forms control to default values. Try out the following example by typing your name in the text field, and click on submit button to see it in action.

Example

```
<form action="action.php" method="post">
    <label for="first-name">First Name:</label>
    <input type="text" name="first-name" id="first-name">
    <input type="submit" value="Submit">
    <input type="reset" value="Reset">
</form>
```



Type your name in the text field above, and click on submit button to see it in action.

Note: You can also create buttons using the `<button>` element. Buttons created with the `<button>` element function just like buttons created with the input element, but they offer richer rendering possibilities by allowing the embedding of other HTML elements.

Grouping Form Controls

You also group logically related controls and labels within a web form using the `<legend>` element. Grouping form controls into categories makes it easier for users to locate a control which makes the form more user-friendly. Let's try out the following example to see how it works:

Example

```
<form>
    <fieldset>
        <legend>Contact Details</legend>
        <label>Email Address: <input type="email" name="email"></label>
        <label>Phone Number: <input type="text" name="phone"></label>
    </fieldset>
</form>
```

Frequently Used Form Attributes

The following table lists the most frequently used form element's attributes:

Attribute	Description
`name`	Specifies the name of the form.
`action`	Specifies the URL of the program or script on the web server that will be used for processing the information submitted via form.
`method`	Specifies the HTTP method used for sending the data to the web server by the browser. The value can be either `get` (the default) and `post` .
`target`	Specifies where to display the response that is received after submitting the form. Possible values are `_blank` , `_self` , `_parent` and `_top` .
`enctype`	Specifies how the form data should be encoded when submitting the form to the server. Applicable only when the value of the `method` attribute is `post` .

There are several other attributes, to know about them please see the `<form>` tag reference.

Note: The `name` attribute represents the form's name within the forms collection. Its value must be unique among the forms in a document, and must not be an empty string.

Tip: All the data sent via `get` method is visible in the browser's address bar. But, the data sent via `post` is not visible to the user. Please check out the tutorial on GET vs. POST to learn about the difference between these two HTTP methods in detail.