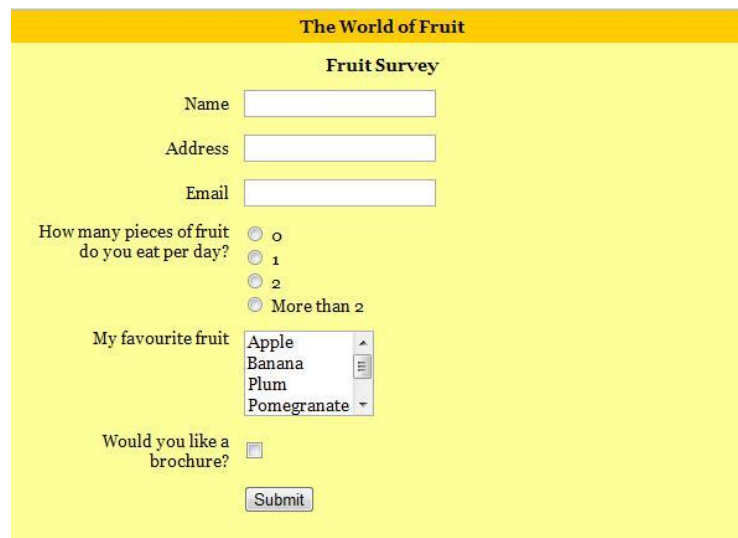


HTML Forms

1. What is the Form?

The web is shift from reading a network of pages to a place where getting information done. For example email address, credit card, signing petitions, searching a site, and posting a tweet. These interactions are handled by forms.

HTML provides form in order to supply a way for the user to interact with a Web server. The most widely used method to process the data submitted through a form is to send it to server-side software typically written in a scripting language.



The World of Fruit

Fruit Survey

Name

Address

Email

How many pieces of fruit do you eat per day?

☐ 0

☐ 1

☐ 2

☐ More than 2

My favourite fruit

Apple

Banana

Plum

Pomegranate

Would you like a brochure? ☐

Submit

Form example

There are two parts to a working form. The **first part** is the form that is seen on the page itself that is created using HTML markup. Forms are made up of buttons, input fields, checkboxes and drop-down menus (collectively known as form controls) used to collect information from the user. Forms may also contain text and other elements. **The other component** of a web form is an application or script on the server that processes the information collected by the form and returns an appropriate response. It's what makes the form work. In other words, posting an

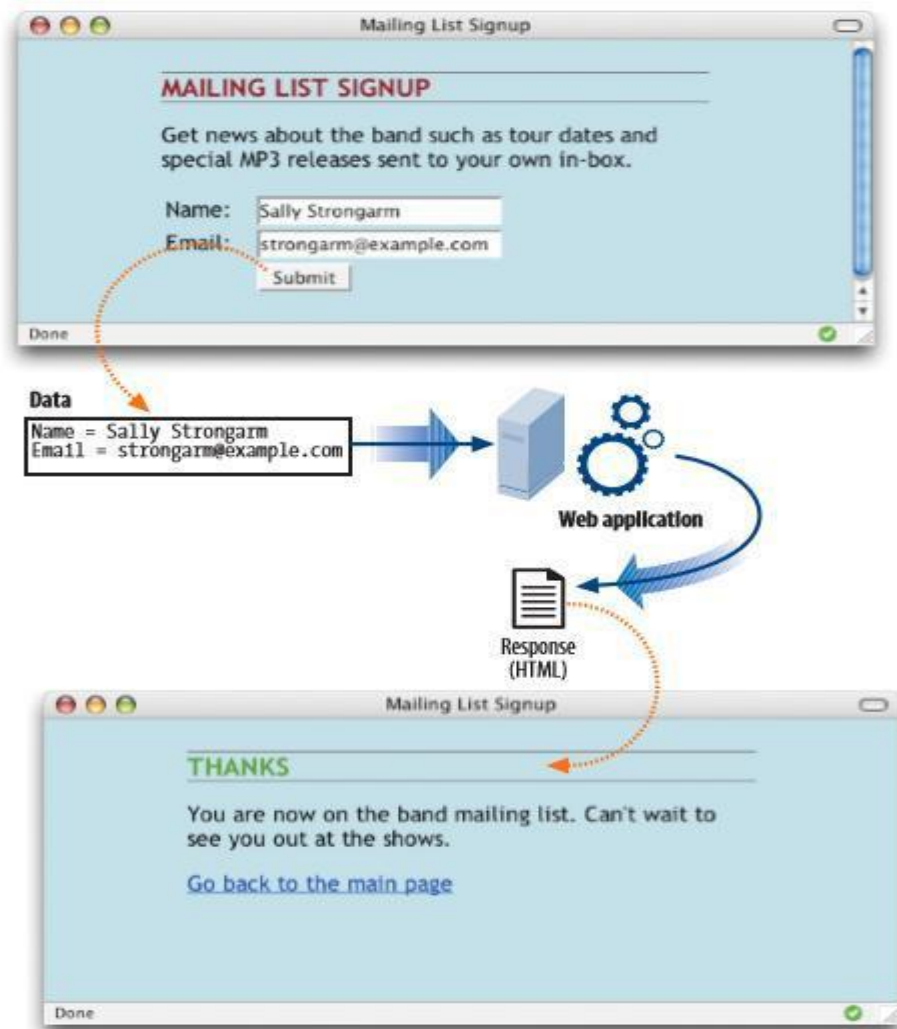
HTML document with form elements isn't enough. Web applications and scripts require programming know how that is beyond the scope.

2. How Forms Work

To understand what is happening behind the scenes when creating web forms, a simple form is used as an example to trace the steps of a transaction that gathers names and email addresses for a mailing list. It is typical of the process for many forms.

1. The visitor opens the page with a web form in the browser window. The browser sees the form control elements in the markup and renders them with the appropriate form controls on the page, including two text entry fields and a submit button as shown in the Figure below.
2. The visitor would like to sign up for this mailing list, the name and email addresses are entered into the fields and submit the form by hitting the Submit button.
3. The browser **collects the entered information**, **encodes it**, and **sends it to the web application on the server**.
4. The web application accepts the information and processes it (that is, does whatever it is programmed to do with it). In this example, the name and email address are added to a database.
5. The web application also returns a response. The kind of response sent back depends on the content and purpose of the form. Here, the response is a web page that contains thanks for signing up for the mailing list. Other applications might respond by reloading the HTML form page with updated information, by moving the user on to another related form page, or by issuing an error message if the form is not filled out correctly.

6. The server sends the web application's response back to the browser where it is displayed.



3. The Form Element

The HTML **<form>** element is used to create interactive forms. The form element is a container for all the content of the form, it has following syntax:

<form>

..

Form elements

..

</form>

An HTML form contains **form elements** and the form elements are different types of input elements, like text fields, checkboxes, radio buttons and submit buttons. It may also contain block elements such as H1, p, and lists.

In addition to being a container for form control elements, the form element has some attributes that are necessary for interacting with the form-processing program on the server.

4. The Form Attribute

4.1 Action Attribute

The action attribute provides the location (URL) of the application or script that will be used to process the form. The action attribute in this example sends the data to a script called mailinglist. php.

```
<form action="/mailinglist.php" >...</form>
```

The .php suffix indicates that this form is processed by a script written in the PHP scripting language, but web forms may be processed using one of the following technologies:

- ☐ PHP (.php) is an open source scripting language most commonly used with the Apache web server.
- ☐ Active Server Pages (ASP.NET) is a programming environment for the Microsoft Internet Information Server (IIS).
- ☐ Java Server Pages (JSP) is a Java based technology similar to ASP.

4.2. Method Attribute

The method attribute specifies how the encoded information should be sent to the server. There are only two methods for sending encoded data to the server: **POST** or **GET**, indicated using the method attribute in the form element. The method is optional and will default to GET if omitted.

```
<form action="/cgi-bin/maillinglist.pl" method="POST">...</form>
```

□ GET Method

The GET method sends the encoded user information appended to the page request (URL) to the server. The ULR and the encoded information are separated by the question mark character. For example:

```
http://www.test.com/index.htm?name1=value1&name2=value2
```

The GET method is appropriate when the results of a form submission are returned to the users such as a list of search results. Because the content of the form is in plain format, GET is not appropriate for forms with private information. In addition, because there is a 256 character limit on what can be appended to URL, GET may not be used for sending a lot of data or when the form is used to upload a file. There are some notes about GET requests:

- GET requests remain in the browser history.
- GET requests can be bookmarked.
- GET requests have length restrictions.
- GET requests should never be used when dealing with sensitive data.
- GET requests should be used only to retrieve data.

□ **POST Method**

The POST method transfers encoded information via HTTP headers. This means that data sent through the POST method will not be visible in the URL, as parameters are not sent along with the URI.

Only the server sees the content of this request, thus it is the best method for sending secure information such as credit card or other personal information. The POST method is also preferable for sending a lot of data, such as a lengthy text entry, because there is no character limit as there is for GET. There are some notes about POST requests:

- POST requests do not remain in the browser history.
- POST requests cannot be bookmarked.
- POST requests have no restrictions on data length.

5. Form Controls

Web forms use a variety of controls that allow users to enter information or choose options. Control types include various text entry fields, buttons, menus, and a few controls with special functions. As a web designer, it is important to be familiar with control options to make forms easy to use. It is also useful to have an idea of what form controls are doing behind the scenes.

The HTML elements that add form controls to the page are:

5.1 Label Element

The label is used to label a control, so that users know what kind of data they should enter.

5.2 Input Element

The **input** element is the most important form element which is used for entering a single word or single-line text field. The input element has different attributes for creating text field as follow:

Attribute	Description
type	Indicates the type of input control and for text input control it will be set to text.
name	Used to give a name to the control which is sent to the server to be recognized and get the value (the variable name for the control).
value	Used to provide an initial value inside the control.
size	Used to specify the width of the text-input control in terms of characters.
maxlength	Used to specify the maximum number of characters a user can enter into the text box.

The <input> element can be displayed in several ways depending on the type attribute. Here are some examples:

	Attribute Type of input element
Attribute Value	Description
type="text"	Defines a one-line text input field
type="password"	Defines a password button for entering a password
type=" button"	Defines a button to trigger a client-side script when the user clicks button.
type="submit"	Defines a submit button for submitting the form
type="reset"	Defines a reset button for reset the form
type="radio"	Defines a radio button for selecting one of many choices
type="checkbox"	Define a checkbox button for selecting more than one choice
type="file"	Allow a user to browse a local file and send it as an attachment with the form data.

□ Text value

Text value is used for items that require only one line of user input, such as search boxes or names.

Example: single-line text input used to take first name and last name:

```
<!DOCTYPE html>
<html>
<head>
<title>Text Input Control</title>
</head>
<body>
<form >
First name: <input type="text" name="first name" />
<br>
Last name: <input type="text" name="last name" />
</form>
</body>
</html>
```

□ Password value

This is also a single-line text input but it masks the character as soon as a user enters it, using asterisk (*) or bullet (•) characters, or another character determined by the browser. It is also created using `<input>` tag but type attribute is set to password. It's important to note that although the characters entered in the password field are not visible to user, the form does not encrypt the information, so it should not be considered a real security measure.

□ Button value

There are various ways in HTML to create clickable buttons. Input element can be used to create a clickable button by setting its type attribute to button. This creates a button that is used to trigger a client-side script when the user clicks that button.

□ Submit value

There are different kinds of buttons that can be added to web forms. The most fundamental is the submit button. When clicked or tapped, the submit button immediately sends the collected form data to the server for processing.

□ **Reset value**

A reset button returns the form controls to the state they were in when the form initially loaded. In other words, resetting the form doesn't simply clear all the fields.

Both submit and reset buttons are added using the input element and type attribute. As mentioned earlier, because these buttons have specific functions that do not include the entry of data, they are the only form control elements that do not require the name attribute. Submit and reset buttons are straightforward to use. Just place them in the appropriate place in the form, which in most cases is at the very end. By default, the submit button displays with the label “Submit” or “Submit Query” and the reset button is labeled “Reset.”

Example: Created a form to display: first, last name text input and the submit buttons.

```
<html>
<head>
  <title> Form Example </title>
</head>
<form>
  <label for="yourname">Enter your First name:</label><br>
  <input type="text" name="firstname"><br>
  <label for="lastname">Enter your Last name:</label><br>
  <input type="text" name="lastname"><br>
  <br><br>
  <input type="submit" value="Submit">
</form>
</html>
```

Enter your First name:

Enter your Last name:

Submit

□ Radio value

Radio buttons are added to a form using the input element with the type attribute set to radio. Here is the syntax for a minimal radio button:

```
<input type="radio" name="variable" value="value">
```

The name attribute is required and plays an important role in binding multiple radio inputs into set. When you give a number of radio button inputs the same name value (age in the following example), they create a group of mutually exclusive options. In this example, radio buttons are used as an interface for users to enter their age group (a person can't belong to more than one age group, so radio buttons are the right choice).

```
<input type="radio" name="age" value="under 25"> under 25<br>
<input type="radio" name="age" value="26-35"> 26-35<br>
<input type="radio" name="age" value="36-45"> 36-45<br>
<input type="radio" name="age" value="over 46"> over 46
```

Notice that all of the input elements have the same variable name (“age”), but their values are different. Because these are radio buttons, **only one button can be checked at a time**, and therefore, **only one value will be sent to the server for processing when the form is submitted**.

A form control made up of a collection of radio buttons is appropriate when only one option from the group is permitted, in other words, when the selections are mutually exclusive (such as Yes or No, or True or False). When one radio button is "on" all of the others must be "off".

□ Checkbox value

Checkboxes are added using the input element with its type set to checkbox. As with radio buttons, groups of checkboxes can be created by assigning them the same

name value. The difference, is that more than one checkbox may be checked at a time. This makes them the right choice for lists in which more than one selection is okay.

The value of every checked button will be sent to the server when the form is submitted. Here is an example of a group of checkbox buttons used to indicate musical interests.

```
<input type="Checkbox" name="genre" value="Arabic"> arabic<br>
<input type="Checkbox" name="genre" value="india"> india<br>
<input type="Checkbox" name="genre" value="hip hop"> hiphop<br>
<input type="Checkbox" name="genre" value="rockabilly"> rockabilly<br>
```

□ File value

The file value allows 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 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 like this:

Upload: No file chosen

5.3 Multiple-Line Text Input Element

The textarea element is used when the user is required to give details that may be longer than a single sentence. Multi-line input controls are created using HTML `<textarea>` tag. The textarea element has the following attributes:

Attribute	Description
name	Used to give a name which is sent to the server to be recognized and get the value.
rows	Indicates the number of rows of text area box.
cols	Indicates the number of columns of text area box
maxlength	Used to specify the maximum number of characters a user can enter into the textarea.

Example: a multi-line text input used to take item description:

```
<!DOCTYPE html>
<html>
<head>
<title>Multiple-Line Input Control</title>
</head>
<body>
<form>
Description : <br />
<textarea rows="5" cols="50" name="description">
Enter description here...
</textarea>
</form>
</body>
</html>
```

5.4 Select Element

Radio buttons allow the user to choose only one item from a range of options, which is useful. But what would happen if the number of options is 20 or more option. A better option is to use the select element (also called drop down menu). This element lets users choose from a range of items, but takes up less space on web

page; initially taking up a single line, where users view the options by clicking on the drop-down arrow (using a mouse or keyboard).

Example:

```
<html>
  <form>
    <label for="role"> Select the country</label>
    <select name="role" >
      <option>Iraq</option>
      <option>Syria</option>
      <option>Egypt</option>
      <option>Other</option>
    </select>
  </form>
</html>
```

A select element provides option to list down various options in the form of drop down list, from where a user can select one or more options by using multiple attribute. If multiple set to "multiple" then allows a user to select multiple items from the menu.

Note: Selecting multiple options vary in different operating systems and browsers. For windows the user need to hold down the control (ctrl) button to select multiple options.

Example:

```
<!DOCTYPE html>
<html>
<head>
<title>Select Box Control</title>
</head>
<body>
<form>
<select name="dropdown" multiple>
<option value="Maths" selected>Maths</option>
```

```

<option value="Physics">Physics</option>
<option value="chemistry">chemistry</option>
<option value="biology">biology</option>
</select>
<input type="submit">
</form>
</body>
</html>

```

5.5 Fieldset Element

The fieldset element (open tag `<fieldset>` and close tag `</fieldset>`) indicates a logical group of form controls. A fieldset may also include a legend element that provides a caption for the enclosed fields.

Example: displays the logical group of customer information:

```

<form>

  <fieldset>
    <legend> Customer Information</legend>
    <label for="Full Name"> Full Name</label><br>
    <input type="text" name="Full name" > <br>
    <label for="E mail"> Email</label><br>
    <input type="text" name="Email"><br>
    <label for="State"> State </label><br>
    <input type="text" name="state"> <br>

  </fieldset>

</form>

```

Customer Information	
Full Name	<input type="text"/>
E_mail	<input type="text"/>
State	<input type="text"/>

6. Form Design

There are many points about designing a form. These points make a form usable and easily accessible to the users, which are:

- ❑ **Avoid unnecessary questions.**

Help the users get through the form easily as possible by avoiding questions that are not necessary to the task at hand. Extra questions, in addition to slowing things down, may make a user wary of the motivations for asking.

- ❑ **Consider impact of label placement.**

The position of the label relative to the input affects the time it takes to fill out the form. Putting the labels above their respective fields creates a single alignment for faster scans when asking for familiar information (username, address). Top-positioned labels can also accommodate labels of varying lengths and work best on narrow, small-screen devices.

- ❑ **Choose input types carefully.**

There are different input types to choose from, and sometimes it's not easy to decide which one to use. For example, a list of options could be presented as a pull-down menu or a number of choices with checkboxes.

- ❑ **Group related inputs.**

It is easier to parse the many fields, menus, and buttons in a form if they are visually grouped by related topic. For example, a user's contact information could be presented in a compact group so that five or six inputs are perceived as one unit.

- ❑ **Clarify primary and secondary actions.**

The primary action at the end of the form is usually some form of Submit button that signals the completion of the form and moving forward. While secondary actions take the user a step back, such as clearing or resetting the form. The primary

action is styled to look different and more important than the secondary action. It is a good idea to provide an undo action.