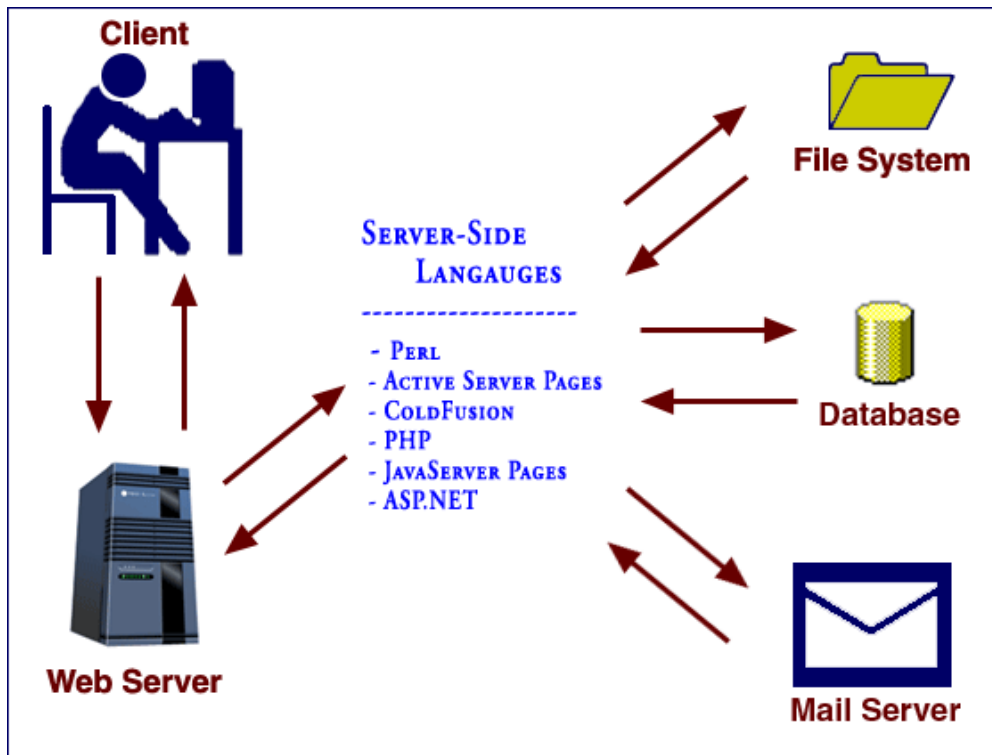


Introduction to HTML Training

HTML Forms

Lesson 1, Activity 2: How HTML Forms Work

HTML forms are used for submitting data back to a script on the server for data processing. When a form is submitted, the data in the form fields is passed to the server in the form of name-value pairs. Server-side scripts, which can be written in several different languages, are used to process the incoming data and return a new HTML page to the browser. The page returned to the browser could be anything from a "Thank you for registering" message or a list of search results generated from a database query.



The process is as follows:

1. The user submits a form. The data is sent to the web server.
2. A script on the web server processes the form, possibly interacting with the file system, one or more databases, a mail server, or any number of other applications.
3. The script generates an HTML page, which the server returns to the client for display.

Lesson 1, Activity 3: The <form> Tag

HTML forms are created using the <form> tag, which takes two main attributes: `action` and `method`.

The `action` specifies the URL of the page that processes the form. It is required. The `method` attribute has two possible values: `post` and `get`. A basic form tag is shown below:

```
<form method="post" action="Register.cfm">
  <!--form fields go here-->
</form>
```

Get vs. Post

The value of the `method` attribute is used to determine how the form data will be passed to the server.

get

If `get` is used the data will be appended to the URL as part of a *querystring*. For example, if the form is filled out as follows:

The screenshot shows a Mozilla Firefox browser window titled "Ice cream Order Form - Mozilla Firefox". The address bar shows "http://localhost/". The form itself is titled "Ice Cream Order Form" and contains the following fields and controls:

- Email:
- Password:
- Flavor:
- Toppings: ☐ Sprinkles ☐ Nuts ☐ Whipped Cream
- Cup or Cone? ☒ Cup ☐ Plain cone ☐ Sugar cone ☐ Waffle cone
- Special Requests:
- Buttons: "Place Order" and "Reset Form"

The status bar at the bottom shows "Done" with a green checkmark.

When the user submitted the form, the URL for the new page would read (without the line breaks):

```
Register.cfm?email=ndunn@webucator.com&pw=foobar&flavor=hardChoc&
sprinkles=on&whip=on&container=wafflecone&
requests=I+want+a+really+big+cone%21&discount=20%2
```

You'll notice the file name is followed by a question mark, which is followed by several name-value pairs (e.g, `container=wafflecone`) separated by ampersands (&).

The `get` method is commonly used by search engines, because it allows the resulting page to be bookmarked.

post

When `post` is used, the name-value pairs are not sent as part of the querystring. Instead they are sent behind the scenes. This has the advantage of keeping the values hidden from anyone looking over the user's shoulder. Two other advantages of the `post` method are:

1. It allows for much more data to be submitted (i.e, larger forms).
2. It allows for files to be uploaded to the server.

As a general rule, you should use `post` unless you want the user to be able to bookmark or share (e.g, via email) the resulting web page.

Files can be uploaded to the server via the file input type. The tag syntax is: `<input type="file" name="filename">`

Lesson 1, Activity 4: Form Elements

This section describes the different form elements that can be used to input data into a form. As you will see, many of these elements, but not all, are created with the `input` tag.

Note that in XHTML strict and HTML 4 **strict**, `input`, other form controls, and `label` elements may not be direct children of the `form` element. They must be nested in other block-level elements.

id and Name Attributes

Form fields (also called controls) take both the `name` attribute and the `id` attribute. They are used for different purposes.

- The `name` attribute is used to hold the value of the field. It is passed to the server as a variable.
- The `id` attribute is used by CSS and JavaScript to identify a specific element.

Labels

Form element labels should be put in `<label>` tags. Labels can be associated with form elements using two methods:

1. Using the `for` attribute of the `<label>` to point to the `id` element of the form element.
2. Wrapping the form element in the `<label>` tag.

We'll show both methods in the first example, but only use the first method in the rest of the examples.

Text Fields

Text fields are created with the `input` tag with the `type` attribute set to `text`. They are used for single lines of text.

Email:

Method 1:

```
<label for="email">Email:</label>
<input type="text" name="email" id="email" value="" size="20" maxlength="40">
```

Method 2:

```
<label>Email: <input type="text" name="email" id="email" value="" size="20" maxlength="40"></label>
```

`<input type="text">` Attributes

Attribute	Description
<code>type</code>	must be set to "text"
<code>name</code>	variable name
<code>value</code>	initial value in the text field
<code>size</code>	width of the text field

maxlength	maximum number of characters that can be entered
-----------	--

Password Fields

Password fields are similar to text fields. The only difference is that the value entered is disguised so that onlookers cannot see it.

Password:

```
<label for="pw">Password:</label>
<input type="password" name="pw" id="pw" size="10" maxlength="12">
```

<input type="password"> Attributes

Attribute	Description
type	must be set to "password"
name	variable name
size	width of the password field
maxlength	maximum number of characters that can be entered

Submit and Reset Buttons

Submit and reset buttons are both created with the `<input>` tag.

Submit Button

A form must have a submit button to be "submittable".

Firefox and Internet Explorer both use "Submit Query" as the default text, but this can be changed with the `value` attribute.

```
<input type="submit" name="submitbutton" id="submitbutton" value="Register">
```

When a form has a submit button, it can be submitted either by clicking on the button or pressing the Enter key when an `input` element has focus.

When a submit button is clicked, the name and value of that button are sent to the server (as a name-value pair). This can be useful in the event that a form has multiple submit buttons as the processing page could behave differently depending on which button was clicked.

Reset Button

A reset button is used to set all the form fields back to their original values.

Firefox and Internet Explorer both use "Reset" as the default text, but this can be changed with the `value` attribute.

```
<input type="reset" name="resetbutton" id="resetbutton" value="Clear">
```

Hidden Fields

```
<input type="hidden" name="discount" id="discount" value="20%">
```

Hidden fields are created with the `input` tag with the `type` attribute set to `hidden`. They are used to pass name-value pairs to the server without displaying them to the user. They are sometimes used in multi-page forms to keep track of variables from a form on a previous page.

`<input type="hidden">` Attributes

Attribute	Description
<code>type</code>	must be set to "hidden"
<code>name</code>	variable name
<code>value</code>	initial value in the hidden field

Although the user can't change the value, it could be changed programatically with JavaScript. For example, the discount might be calculated based on other options the user selected in the form.

Checkboxes

Checkboxes are created with the `input` tag with the `type` attribute set to `checkbox`. The default value for a checkbox is "on". Although the value of a checkbox can be changed with the `value` attribute, there is usually no reason to do so, as the name-value pair only gets sent to the server if the checkbox is checked. In other words, the script on the server only needs to check for the existence of the variable name to see if the checkbox was checked or not.

Toppings: ☒ Sprinkles ☐ Nuts ☒ Whipped Cream

Toppings:

```
<input type="checkbox" name="sprinkles" id="sprinkles"> <label for="sprinkles">Sprinkles</label>
<input type="checkbox" name="nuts" id="nuts"> <label for="nuts">Nuts</label>
<input type="checkbox" name="whip" id="whip"> <label for="whip">Whipped Cream</label>
```

`<input type="checkbox">` Attributes

Attribute	Description
<code>type</code>	must be set to "checkbox"
<code>name</code>	variable name

value	value of the checkbox
checked	indicates that checkbox is pre-checked

Radio Buttons

Radio buttons are created with the `input` tag with the `type` attribute set to `radio`. Radio buttons generally come in groups, in which each radio button has the same name. Only one radio button in the group can be checked at any given time. Each radio button in the group should have a unique value - the value to be sent to the server if that radio button is selected. There is often no need to provide ids for radio buttons as they are not usually styled with CSS and any script code would most likely try to access the whole group and treat it as an array, rather than try to access a single radio button.

Cup or Cone? ☐ Cup ☐ Plain cone ☐ Sugar cone ☒ Waffle cone

```
Cup or Cone?
<label><input type="radio" name="container" value="cup"> Cup</label>
<label><input type="radio" name="container" value="plaincone"> Plain cone</label>
<label><input type="radio" name="container" value="sugarcone"> Sugar cone</label>
<label><input type="radio" name="container" value="wafflecone"> Waffle cone</label>
```

You will notice that we used `label` differently this time. Instead of using the `for` attribute, we wrapped the radio button in the `label` tag. This is because the radio buttons don't include `id` attributes.

<input type="radio"> Attributes

Attribute	Description
type	must be set to "radio"
name	variable name
value	value of the radio button
checked	indicates that radio button is pre-checked

Id Attribute for Radio Buttons

You may have noticed that form elements such as text fields and checkboxes, the `id` is usually the same as the `name`. They aren't actually related at all, so they don't have to be the same. As stated above:

- The `name` attribute is used to hold the value of the field. It is passed to the server as a variable.
- The `id` attribute is used by CSS and JavaScript to identify a specific element.

Nonetheless, it's usually easier just to give them the same name, so you can refer to the elements using the same name in all other files (CSS, JavaScript, etc...).

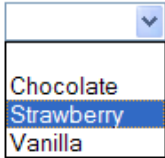
For radio buttons, this isn't possible though, because radio buttons in the same set must all have the same name, but different ids.

When the form is submitted, the value of the selected button gets sent to the server. So, for example, if we had a radio button asking to select the gender ("m" or "f"), selecting "m" would mean the form will send "gender=m" to the server. However, if we wanted to use JavaScript to autoselect the "m" radio button, we would need to be able to distinguish it

from the "f" radio button - this is where IDs would come in. Essentially, the radio buttons in the same set need to have different `id` values.

Select Menus

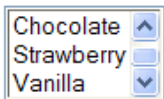
Select menus are created with the `select` tag. The `size` attribute determines how many options are shown at once.



By default, only one option can be selected; however, this can be changed by adding the `multiple` attribute to the `select` tag. The following tag would create a scrollable combo box that showed three options at a time and allowed multiple options to be selected:

```
<label for="flavor">Flavor:</label>
<select name="flavor" id="flavor" size="3" multiple="multiple">
  <option value="0" selected="selected"></option>
  <option value="choc">Chocolate</option>
  <option value="straw">Strawberry</option>
  <option value="van">Vanilla</option>
</select>
```

The result would look like this:



The `select` element must contain one or more `option` elements. The text between the open and close `option` tags appears in the select menu. The value of the `value` attribute is what gets passed to the server if that `option` is selected. The `selected` attribute is used to preselect an option.

<select> Attributes

Attribute	Description
name	variable name
size	number of options to appear at once
multiple	indicates that multiple options can be selected. value must be "multiple"

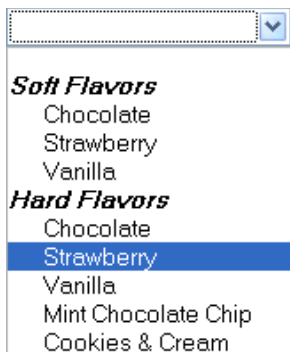
<option> Attributes

Attribute	Description
value	value to send to server if option is selected
selected	indicates that option is pre-selected. value must be "selected"

Option Groups

Options can be arranged in groups using the `<optgroup>` tag. The `label` attribute is used to set the option group heading.

```
<label for="flavor">Flavor:</label>
<select name="flavor" id="flavor">
  <option value="0" selected="selected"></option>
  <optgroup label="Soft Flavors">
    <option value="softChoc">Chocolate</option>
    <option value="softStraw">Strawberry</option>
    <option value="softVan">Vanilla</option>
  </optgroup>
  <optgroup label="Hard Flavors">
    <option value="hardChoc">Chocolate</option>
    <option value="hardStraw">Strawberry</option>
    <option value="hardVan">Vanilla</option>
    <option value="hardMint">Mint Chocolate Chip</option>
    <option value="hardCC">Cookies & Cream</option>
  </optgroup>
</select>
```



Textareas

Textareas are created with the `textarea` tag. The `cols` and `rows` attributes indicate the number of columns and rows (in characters) that the textarea should span.

Special Requests:

An initial value can be entered into the textarea by adding text between the open and close `textarea` tags. For example,

```
<textarea name="requests" cols="40" rows="6">
  Initial value goes here.
```

```
</textarea>
```

<textarea> Attributes

Attribute	Description
name	variable name
cols	number of columns to span in character width
rows	number of rows to span in character height

Lesson 1, Activity 6: Creating a Registration Form

Duration: 15 to 25 minutes.

In this exercise, you will begin to create a registration form. The form will submit to a page on <http://www.remotecourse.com> which will simply "dump" the values entered into the form back to you.

The form should appear as follows:

1. Open [Forms/Exercises/RegistrationForm.html](#) for editing.
2. Add a form to the page.
 - The action page is <http://www.remotecourse.com/Classes/htm101/process.cfm>.
 - The method should be "post".
3. Add the following input elements:
 - text: name and id should be "username"
 - password: name and id should be "pw"
 - hidden: name and id should be "emailMe"; value should be your email.
 - submit button: value should be "Register"
 - reset button
4. When you are done, open the page in a browser, and fill out and submit the form. If successful, the processing page will display the values you entered. You will need internet access to see the resulting page.

Solution:

[Forms/Solutions/RegistrationForm1.html](#)

```
<!DOCTYPE HTML>
<html>
<head>
<meta charset="UTF-8">
<title>Registration Form</title>
</head>
<body>
<h1>Registration Form</h1>
<form method="post" action="http://www.remotecourse.com/Classes/htm101/process.cfm">
<div>
  <input type="hidden" name="emailMe" id="emailMe" value="you@you.com">
```

```
<label for="username">User name:</label>
<input type="text" name="username" id="username" size="15">
</div>
<div>
  <label for="pw">Password:</label>
  <input type="password" name="pw" id="pw" size="15">
</div>
<div>
  <input type="submit" name="submitbutton" id="submitbutton" value="Register">
  <input type="reset" name="resetbutton" id="resetbutton">
</div>
</form>
</body>
</html>
```

Lesson 1, Activity 9: Adding Checkboxes and Radio Buttons

Duration: 10 to 15 minutes.

In this exercise, you will add a checkbox and radio buttons to the registration form. At completion, the form should look like this:

The screenshot shows a Mozilla Firefox browser window with the title 'Registration Form - Mozilla Fi...'. The address bar shows 'http://'. The form content is as follows:

Registration Form

User name:

Password:

Gender: ☐ Male ☐ Female

Click to accept our terms: ☐

Done

1. Open [Forms/Exercises/RegistrationForm.html](#) for editing if you don't have it open already.
2. Add the following input elements:
 - radio buttons: name gender and values should be "male" and "female"
 - checkbox: name and id should be "terms"
3. When you are done, open the page in a browser, and fill out and submit the form. You will need internet access to see the resulting page.

Solution:

[Forms/Solutions/RegistrationForm2.html](#)

```
<!DOCTYPE HTML>
<html>
<head>
<meta charset="UTF-8">
<title>Registration Form</title>
</head>
<body>
<h1>Registration Form</h1>
<form method="post" action="http://www.remotecourse.com/Classes/html01/process.cfm">
<div>
  <input type="hidden" name="emailMe" id="emailMe" value="you@you.com">

  <label for="username">User name:</label>
  <input type="text" name="username" id="username" size="15">
</div>
<div>
  <label for="pw">Password:</label>
  <input type="password" name="pw" id="pw" size="15">
</div>
```

```
<div>
  Gender:
  <input type="radio" name="gender" id="genderm" value="m"><label for="genderm">Male</label>
  <input type="radio" name="gender" id="genderf" value="f"><label for="genderf">Female</label>
</div>
<div>
  <label for="terms">Click to accept our terms:</label>
  <input type="checkbox" name="terms" id="terms">
</div>
<div>
  <input type="submit" name="submitbutton" id="submitbutton" value="Register">
  <input type="reset" name="resetbutton" id="resetbutton">
</div>
</form>
</body>
</html>
```

Lesson 1, Activity 10: Adding a Select Menu and a Textarea

Duration: 15 to 25 minutes.

In this exercise, you will add a select menu and textarea to the registration form. At completion, the form should look like this:

The screenshot shows a Mozilla Firefox browser window with the title 'Registration Form - Mozilla Firefox'. The address bar shows 'http://'. The form contains the following elements:

- Registration Form** (Title)
- User name:
- Password:
- Where did you hear about us?
- Gender: ☐ Male ☐ Female
- Comments:
- Click to accept our terms: ☐
- Register Reset

The status bar at the bottom shows 'Done' and a green checkmark.

1. Open [Forms/Exercises/RegistrationForm.html](#) for editing if you don't have it open already.
2. Add the following input elements:
 - select: name and id should be "referral". Options should be:
 - "--Please choose--" with value of "0"
 - "Word of Mouth" with value of "wom"
 - "Other" with value of "other"
 - Option group with label of "Search Engine" and the following options:
 - "Google" with value of "Google"
 - "Yahoo!" with value of "Yahoo"
 - "MSN" with value of "MSN"
 - textarea: name and id should be "comments". Be sure to include `cols` and `rows` attributes.
3. When you are done, open the page in a browser, and fill out and submit the form. You will need internet access to see the resulting page.

Solution:

[Forms/Solutions/RegistrationForm3.html](#)

```
---- C O D E   O M I T T E D ----
<label for="pw">Password:</label>
<input type="password" name="pw" id="pw" size="15">
```



```

</div>
<div>
  <label for="referral">Where did you hear about us?</label>
  <select name="referral" id="referral">
    <option value="0">--Please choose--</option>
    <optgroup label="Search Engine">
      <option value="Google">Google</option>
      <option value="Yahoo">Yahoo!</option>
      <option value="MSN">MSN</option>
      <option value="OtherSearchEngine">Other</option>
    </optgroup>
    <option value="wom">Word of Mouth</option>
    <option value="other">Other</option>
  </select>
</div>
<div>
  Gender:
  <input type="radio" name="gender" id="genderm" value="m"><label for="genderm">Male</label>
  <input type="radio" name="gender" id="genderf" value="f"><label for="genderf">Female</label>
</div>
<div>
  <label for="comments">Comments:</label><br>
  <textarea name="comments" id="comments" cols="40" rows="4"></textarea>
</div>
<div>
  <label for="terms">Click to accept our terms:</label>
  <input type="checkbox" name="terms" id="terms">
</div>
<div>
  <input type="submit" name="submitbutton" id="submitbutton" value="Register">
  <input type="reset" name="resetbutton" id="resetbutton">
</div>
</form>
</body>
</html>

```