Web Design Course

LECTURE Nine
Introduction to HTML and XHTML

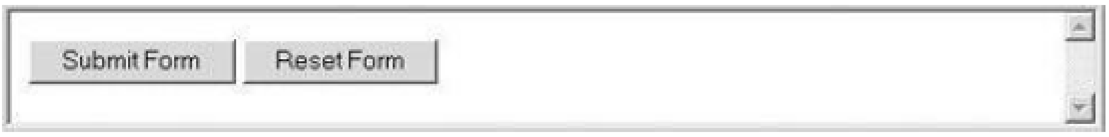
Computer Science
Collage of Education
AL_Mustansyria University
Forth Year

2018-2019

7.5 The Action Buttons

The Reset button clears all of the controls in the form to their initial states. The Submit button has two actions: First, the form data is encoded and sent to the server; second, the server is requested to execute the server-resident program specified in the action attribute of the <form> tag. The purpose of such a server-resident program is to process the form data and return some response to the user. Every form requires a Submit button. The Submit and Reset buttons are created with the <input> tag, as shown in the following example:

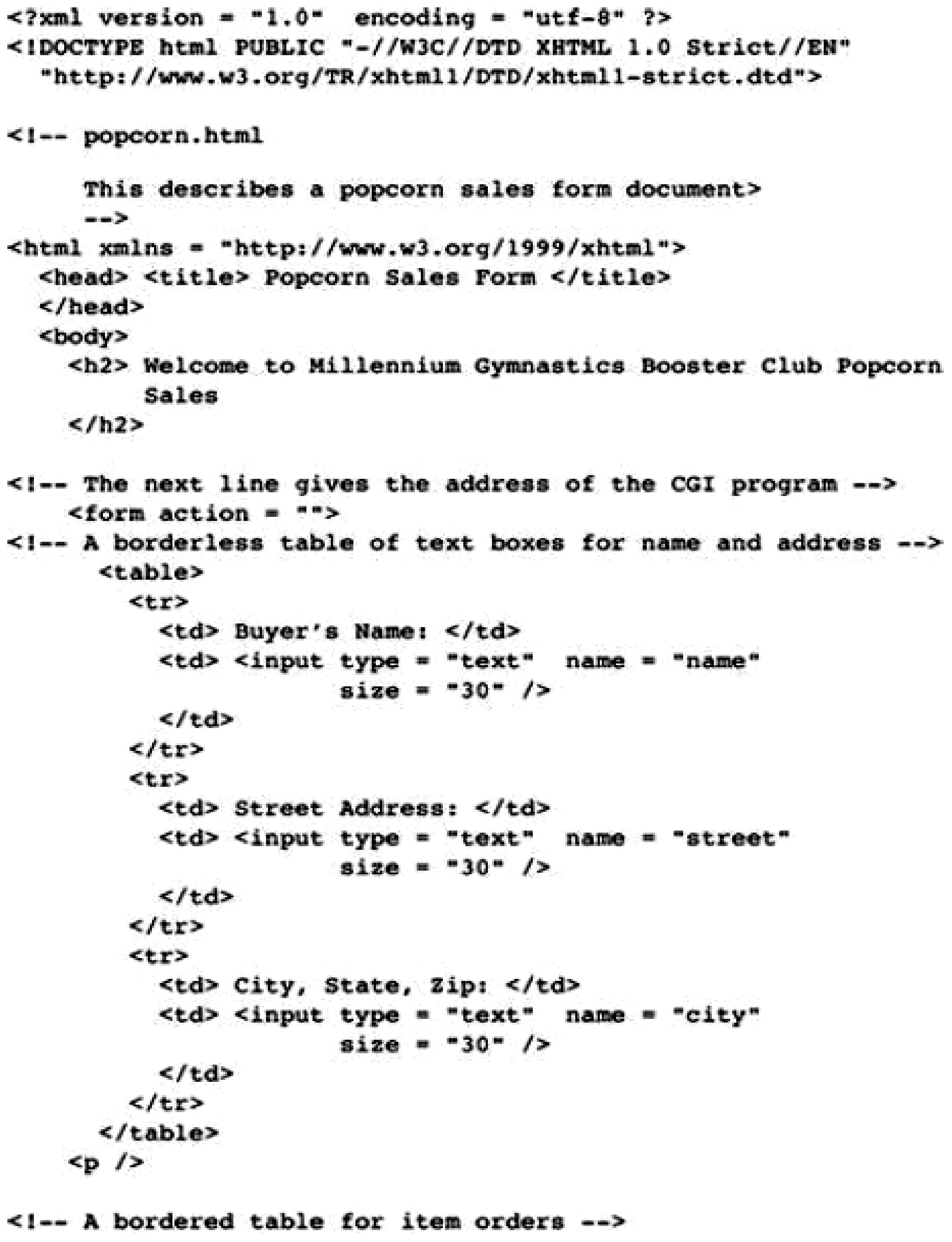
```
<form action = ““>
<p>
<input type = “submit” value = “Submit Form” />
<input type = “reset” value = “Reset Form” />
</p>
</form>
```



A plain button has the type button. Plain buttons are used to choose an action.

7.6 Example of a Complete Form

The document that follows describes a form for taking sales orders for popcorn. Three text boxes are used at the top of the form to collect the buyer’s name and address. These boxes are placed in a borderless table to force them to align vertically. A second table is used to collect the actual order. Each row of this table names a product with the content of a <td> tag, displays the price with another <td> tag, and uses a text box with size set to 2 to collect the quantity ordered. The payment method is input by the user through one of four radio buttons. Notice that none of the input controls in this document are embedded in label elements. This is because table elements cannot be labeled, except by using the row and column labels. Tables present special problems for the visually impaired. The best solution is to use style sheets instead of tables to lay out tabular information.

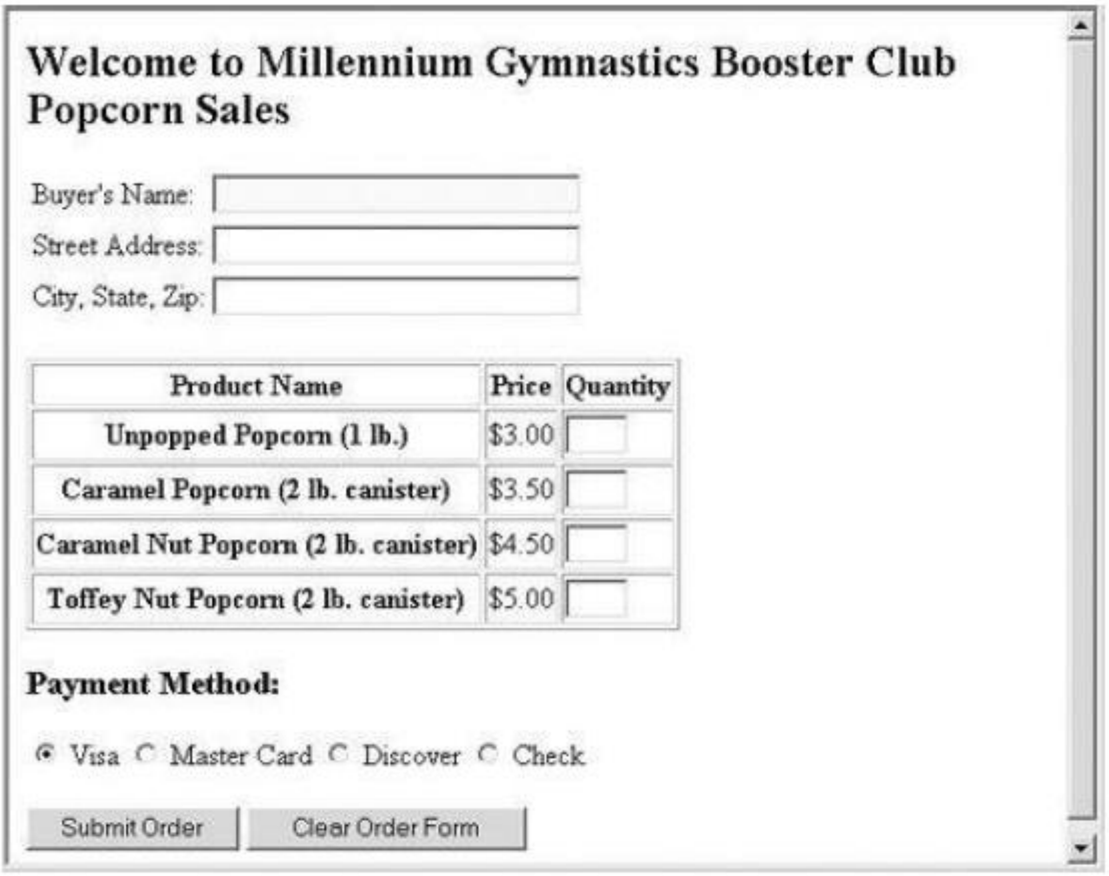


```
<?xml version = "1.0"  encoding = "utf-8" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<!-- popcorn.html

     This describes a popcorn sales form document>
     -->
<html xmlns = "http://www.w3.org/1999/xhtml">
  <head> <title> Popcorn Sales Form </title>
  </head>
  <body>
    <h2> Welcome to Millennium Gymnastics Booster Club Popcorn
         Sales
    </h2>

<!-- The next line gives the address of the CGI program -->
    <form action = "">
<!-- A borderless table of text boxes for name and address -->
      <table>
        <tr>
          <td> Buyer's Name: </td>
          <td> <input type = "text"  name = "name"
                      size = "30" />
          </td>
        </tr>
        <tr>
          <td> Street Address: </td>
          <td> <input type = "text"  name = "street"
                      size = "30" />
          </td>
        </tr>
        <tr>
          <td> City, State, Zip: </td>
          <td> <input type = "text"  name = "city"
                      size = "30" />
          </td>
        </tr>
      </table>
    <p />

<!-- A bordered table for item orders -->
```

```
<!-- First, the column headings -->
        <tr>
          <th> Product Name </th>
          <th> Price </th>
          <th> Quantity </th>
        </tr>

<!-- Now, the table data entries -->
        <tr>
          <td> Unpopped Popcorn (1 lb.) </td>
          <td> $3.00 </td>
          <td> <input type = "text"  name = "unpop"
                      size = "2" />
          </td>
        </tr>
        <tr>
          <td> Caramel Popcorn (2 lb. canister) </td>
          <td> $3.50 </td>
          <td> <input type = "text"  name = "caramel"
                      size = "2" />
          </td>
        </tr>
        <tr>
          <td> Caramel Nut Popcorn (2 lb. canister) </td>
          <td> $4.50 </td>
          <td> <input type = "text"  name = "caramelnut"
                      size = "2" />
          </td>
        </tr>
        <tr>
          <td> Toffey Nut Popcorn (2 lb. canister) </td>
          <td> $5.00 </td>
          <td> <input type = "text"  name = "toffeynut"
                      size = "2" />
          </td>
        </tr>

      </table>
      <p />
```

```
<!-- The radio buttons for the payment method -->
      <h3> Payment Method: </h3>
      <p>
        <label> <input type = "radio"  name = "payment"
                       value = "visa"  checked = "checked" />
                       Visa
        </label>
        <br />
        <label> <input type = "radio"  name = "payment"
                       value = "mc" /> Master Card
        </label>
        <br />
        <label> <input type = "radio"  name = "payment"
                       value = "discover" /> Discover
        </label>
        <br />
        <label> <input type = "radio"  name = "payment"
                       value = "check" /> Check
        </label>
        <br />
      </p>

<!-- The submit and reset buttons -->
      <p>
        <input type = "submit"  value = "Submit Order" />
        <input type = "reset"  value = "Clear Order Form" />
      </p>
    </form>
  </body>
</html>
```

Figure 29 shows a browser display of popcorn.html.



Welcome to Millennium Gymnastics Booster Club Popcorn Sales

Buyer's Name:	
Street Address:	
City, State, Zip:	

Product Name	Price	Quantity
Unpopped Popcorn (1 lb.)	$3.00	
Caramel Popcorn (2 lb. canister)	$3.50	
Caramel Nut Popcorn (2 lb. canister)	$4.50	
Toffey Nut Popcorn (2 lb. canister)	$5.00	

Payment Method:

[✓ Visa] Master Card Discover Check

Figure 29 Display of popcorn.html

Exercises

1- Create, test, and validate an XHTML document for yourself, including your name, address, and e-mail address. If you are a student, you must include your major and your grade level. If you work, you must include your employer, your employer's address, and your job title. This document must use several headings and <em>, <strong>, <hr />, <p>, and <br /> tags.

2- Add pictures of yourself and at least one other image (of your friend, spouse, or pet) to the document created for Exercise 1.

3- Add a second document to the document created for Exercise 1 that describes part of your background, using background as the link content. This document should have a few paragraphs of your personal or professional history.

4- Create, test, and validate an XHTML document that describes an unordered list equivalent to your typical grocery shopping list. (If you've never written a grocery list, use your imagination.)

5- Create, test, and validate an XHTML document that describes an unordered list of at least four states. Each element of the list must have a nested list of at least three cities in the state.
6- Create, test, and validate an XHTML document that describes an ordered list of your five favorite movies.

7- Modify the list of Exercise 6 to add nested, unordered lists of at least two actors and/or actresses in your favorite movies.
8- Create, test, and validate an XHTML document that describes an ordered list with the following contents: The highest level should be the names of your parents, with your mother first. Under each parent, you must have a nested, ordered list of the brothers and sisters of your parents, in order by age, eldest first. Each of the nested lists in turn must have nested lists that list the children of your uncles and aunts (your cousins)—under the proper parents, of course. Regardless of how many aunts, uncles, and cousins you actually have, there must be at least three list items in each sublist below each of your parents and below each of your aunts and uncles.

9- Create, test, and validate an XHTML document that describes a table with the following contents: The columns of the table must have the headings "Pine", "Maple", "Oak", and "Fir". The rows must have the labels "Average Height", "Average Width", "Typical Life Span", and "Leaf Type". You can make up the data cell values.

10- Modify, test, and validate an XHTML document from Exercise 9 that adds a second-level column label, "Tree", and a second-level row label, "Characteristics".

11- Create, test, and validate an XHTML document that defines a table with columns for state, state bird, state flower, and state tree. There must be at least five rows for states in the table. You must include attribute specifications for cellpadding and cellspacing.

12- Create, test, and validate an XHTML document that defines a table with two levels of column labels: an overall label, "Meals", and three secondary labels, "Breakfast", "Lunch", and "Dinner". There must be two levels of row labels: an overall label, "Foods", and four secondary labels, "Bread", "Main Course", "Vegetable", and "Dessert". The cells of the table must contain a number of grams for each of the food categories.

13- Create, test, and validate an XHTML document that is the home page of a business, Tree

Branches, Unlimited, that sells tree branches. This document must include images and descriptions of at least three different kinds of tree branches. There must be at least one unordered list, one ordered list, and one table. Detailed descriptions of the different branches must be stored in separate documents that are accessible through links from the home document. You must invent several practical uses for tree branches and include sales pitches for them.

14- Create, test, and validate an XHTML document that has a form with the following controls:
a. A text box to collect the user's name
b. Four checkboxes, one each for the following items:
  i. Four 100-watt light bulbs for $2.39
  ii. Eight 100-watt light bulbs for $4.29
  iii. Four 100-watt, long-life light bulbs for $3.95
  iv. Eight 100-watt, long-life light bulbs for $7.49
c. A collection of three radio buttons that are labeled as follows:
  i. Visa
  i. Visa
  ii. Master Card
  iii. Discover