Overview and Objectives Up to this pont we have seen only very limited interactivity between the user and any of our web pages. In fact, all a user coud do was type n the URL of a page and press Enter to dispay that page, or click a link on one page to browse to another page. That is, there has so far been no way for the user to supply any additona informa tion to a website, through one of its web pages, have the site process the information in some way, and then return some new information to the user based on the data sup plied. This kind of twoway communication scenario is extremely usefu in a business environment, since it permits a user to make product choices, pay for them online, and suppy a shippng address, for example. This chapter marks the beginning of our discussion of how such twoway communi cation is accompished, and the first thing we need to study is how a web form can be paced on a web page in preparation for accepting information from the user on the cient side. n subsequent chapters you wil see how that information can be transmtted to the server and how your website can process, and respond to, the data received from its visitors. In this chapter we wil discuss the folowing: The genera idea of a web form The form element (though we postpone a dscussion of the GET and POST values of its action attribute, since these ndicate how we wish to submit the form's data, and in this chapter we are only creating forms, not submitting their data to a server) The input element The select and option elements The textarea eement The submit and reset button elements The fieldset and legend elements The label element and its for attribute Setting up a form that alows a user to compute his or her Body Mass Index (BMI) Setting up a form that allows a user to submit feedback to our business Getting ready to submit form data (but not actualy submitting it) Aso in ths chapter our examples wi focus entrey on form design and we wil not return to an update of our fu website example until Chapter 7, after we have discussed forms (this chapter), form data validation via JavaScript (Chapter 6), and dropdow 5.2 The form Element165 menus (also mplemented using JavaScript) as an exampe of nteraction with the DOM (Document Object Model) and interactively changing our document's Cascading Style Sheets (CSS) styes (Chapter 7). 5.1 Web Forms Collect User Input Data in a Browser for Transfer to a Server for Processing We are all familiar, perhaps too familiar, with the forms we have to fill out in our everyday lives, such as employment applications, medical reports, student loan applications, income tax forms, and so on. Online forms contain the same kinds of blank spaces or "textboxes" to be filled in, checkboxes to be checked, and so on, and we need to learn how to set these things up for data entry on our website. Even though a user can only communicate with a website in two basic ways—by clicking on various parts of a web page using a pointing device such as a mouse, or by typing text using a keyboard—such a web form allows the information supplied in this way to be quite sophisticated. Once a web form is filled out, it "looks and feels" much like its paper counterpart. Submit ting a web form, however, is accomplished in a different manner. Usually there is a button, often labeled Submit, that the user clicks, and there may also be one labeled Reset, to permit the user to start over and reenter form data. It is what happens after the user clicks Submit that we will discuss in later chapters. We are going to illustrate simple form design through two examples for our business: 1.The first one will be a web page containing a form that is essentially a tool to help users find out if their weight is reasonable by calculating a quantity known as the BMI. 2.The second one will be a page containing a standard user-feedback form of the type used by many websites to gather user opinions on everything from the usability of the site to the overall customer experience if it is an e-commerce site. 5.2 The form Element Placing a form on a web page is done using an HyperText Markup Language (HTML) form element. This element acts like a “container” for the usual kinds of things you see on a form, in the sense that between the and tags you place the other HTML tags that create the visual “widgets” that allow data entry by the user when the web page is displayed. Let us begin with a web page that will eventually serve as your BMI calculator. Look at the HTML markup from the file bmi1.html shown in FIGURE 5.1 to see how we introduce a form element into a web document. The relevant HTML code is shown in lines 12 and 13 of that file: The displayed web page itself does not look any different (see FIGURE 5.2) from any of the previous web pages you have seen so far. The page has no CSS styling and therefore uses the default browser fonts and font sizes and the page will grow and shrink according to any resizing of the browser window. Note that the mere introduction of the (empty) form element has no effect on the display. However, it helps us study the tag without worrying about the rest of the details. We have identified our form element as bmiForm by giving this value to its (optional) id attribute. This is generally a good idea, since we may have more than one form and in any case it allows us to identify the form for styling or access by JavaScript. We should also say something about the action attribute of the form element, which is not shown here. The value of the action attribute is generally the name of the serverside script or other program that will process the data from the form when that data is submitted to the server. Since you are not submitting at the moment, you don’t need that attribute. However, this is one of the few differences between XHTML and HTML5 that can lead to some confusion. For XHTML markup to validate, the action attribute of the form tag had to be present and have a value, but if the form was not being submitted that value could be the empty string. What is potentially confusing is that in HTML5 the action attribute is optional, but if present the value cannot be the empty string. That is why we simply omit the action attribute, but if you are looking at the markup of others you may well see action="" as a form attribute. In later chapters, when you submit your form data, you will have Personal Home Page (PHP) scripts that run on your server and process that data, and the value of the action attribute will then contain the name of one of those scripts. Obviously, having just an empty form element on a web page is not very interesting. The following sections will show you how to put different form elements on your BMI calculator page. These other HTML elements that are placed within the form element itself and determine the interactive behavior of the form are also called form controls, or, more informally, just “widgets”. 5.2.1 A Brief Aside, a Reminder, and Some Good Advice Just before we move on, let’s discuss a brief aside. Note that the HTML file in Figure 5.1 contains an HTML element you have not seen before, a center element (see line 10). The effect of this element should be obvious: It centers its content, in this case the h1 element, and this is certainly a very easy way to achieve that effect. Unfortunately, the HTML center tag is now deprecated, which means that even though it still works it should no longer be used in new markup and may eventually not work at all. This begs the question: Why was it deprecated? Well, because “centering” something is a “presentational” aspect of that thing, and therefore rightly should be handled by CSS, which you will do in the next example. We use the center element here to remind you once again that you should try to avoid the easy way out when developing new web pages. One way of staying the course is to make sure you continually validate your web pages. If you attempt to validate the file bmi1.html, for example, you will find that in fact it does not validate as HTML5, precisely because of the presence of the center element. 5.2.2 How Will We Deal with CSS from Now On? From now on, we will handle our CSS in the following way. We will have, for each chapter, either a single CSS file called default.css, (or two files called desktop.css and tablet.css if we want to make use of responsive design, which we do not in this chapter). Any such files we are using in a given chapter will be placed in a subdirectory called css. If we have a default.css, it will contain all CSS styles used for the general sample files in that chapter. Not every style will necessarily be used for every sample file in the chapter, but that does not matter. In fact, that is the point. We can put all our styles in one place (one file) and link any HTML document file that needs to use one or more styles in that style file. Of course, for a large and complex website you may well want to have more than one style file, but the point remains. Subsequent files in this chapter will be making use of the version of this file corresponding to this chapter. We will only discuss the default.css file if it contains something of particular interest or a new CSS feature of some kind. If a later chapter also contains an updated version of our complete website, we will put the corresponding files in a subdirectory whose name starts with nature and which has its own css subdirectory containing its own default.css (or desktop.css and tablet.css, as the case may be). We actually began this convention in the last chapter, where we had four such subdirec tories, nature1 to nature4, containing four different versions of our website, but we will not see a nature subdirectory again until Chapter 7. 5.3 The input Element One of the most versatile HTML form controls for collecting data from users is the input element. It allows you to create form fields (places where the user can supply data of some sort) of various types. The kind of data that an input form field will accept is determined by the value of its type attribute. Here are some possible values for this attribute: text creates a one-line text box whose visible width, in characters, is determined by the value of the size attribute, and for which the maxlength attribute specifies the maxi mum number of characters allowed as input. radio creates a “radio button” that usually appears as a small circle in a group of small circles, only one of which may be selected. checkbox creates a small box, generally square, that may be “checked” or “unchecked” and also often appears in a group that (unlike the radio button) permits multiple boxes to be checked. We use the center element here to remind you once again that you should try to avoid the easy way out when developing new web pages. One way of staying the course is to make sure you continually validate your web pages. If you attempt to validate the file bmi1.html, for example, you will find that in fact it does not validate as HTML5, precisely because of the presence of the center element. 5.2.2 How Will We Deal with CSS from Now On? From now on, we will handle our CSS in the following way. We will have, for each chapter, either a single CSS file called default.css, (or two files called desktop.css and tablet.css if we want to make use of responsive design, which we do not in this chapter). Any such files we are using in a given chapter will be placed in a subdirectory called css. If we have a default.css, it will contain all CSS styles used for the general sample files in that chapter. Not every style will necessarily be used for every sample file in the chapter, but that does not matter. In fact, that is the point. We can put all our styles in one place (one file) and link any HTML document file that needs to use one or more styles in that style file. Of course, for a large and complex website you may well want to have more than one style file, but the point remains. Subsequent

files in this chapter will be making use of the version of this file corresponding to this chapter. We will only discuss the default.css file if it contains something of particular interest or a new CSS feature of some kind. If a later chapter also contains an updated version of our complete website, we will put the corresponding files in a subdirectory whose name starts with nature and which has its own css subdirectory containing its own default.css (or desktop.css and tablet.css, as the case may be). We actually began this convention in the last chapter, where we had four such subdirec tories, nature1 to nature4, containing four different versions of our website, but we will not see a nature subdirectory again until Chapter 7. 5.3 The input Element One of the most versatile HTML form controls for collecting data from users is the input element. It allows you to create form fields (places where the user can supply data of some sort) of various types. The kind of data that an input form field will accept is determined by the value of its type attribute. Here are some possible values for this attribute: text creates a one-line text box whose visible width, in characters, is determined by the value of the size attribute, and for which the maxlength attribute specifies the maxi mum number of characters allowed as input. radio creates a "radio button" that usually appears as a small circle in a group of small circles, only one of which may be selected. checkbox creates a small box, generally square, that may be "checked" or "unchecked" and also often appears in a group that (unlike the radio button) permits multiple boxes to be checked. submit creates a button on which you can place a label (often just the word Submit) and, when clicked on with the mouse, causes the data that has been entered in the form to be acted upon. reset also creates a button on which you can place a label (often just the word Reset) and allows the values in the form to be cleared, or set to their default values, thus permit ting the user to start over. So, any desired control is obtained by supplying the appropriate value from the above list as the value of the type attribute of the input tag. For example, in FIGURE 5.3 you see illustrated input controls of type text (lines 18 and 26), radio (lines 20–21 and 28–31), and checkbox. (line 36). FIGURE 5.4 displays our first "real" form, since it contains some actual "fields", form controls for entering or indicating data input. Each of the controls in the form displayed in Figure 5.4 is created by using an input element. These controls include the things that we call "textboxes", "radio buttons", and a "checkbox", and which are produced by the corresponding HTML markup referred to above and seen in the file bmi2.html of Figure 5.3. Note, first of all, as promised in the previous section, that we are no longer using the (depre cated) HTML center element for centering our page title, but instead a class called Centered, which is defined in this chapter's default.css1 file and used with the h4 element. Second, note that for this form we are using a table layout with three rows and four columns to contain the controls used to collect information from the user. We can argue for a table layout here as the appropriate choice, since a table makes sense whether we are displaying data, or, as in the current case, collecting data. Note that although we are using a table, it is not immediately obvious since we do not display any table borders. And now for the details . . . 5.3.1 Textboxes (input Elements of Type text) The empty textbox in the first row of the table (line 18 of Figure 5.3) allows a user to enter his or her height. This is the markup that creates this textbox: As you can see, the input element is an empty element, and this one has three attributes. The type attribute tells the web browser what kind of input control this one is (textbox, checkbox, radio button, and so on). In this case, the value text indicates we have a textbox. The name attribute is used to identify this particular HTML element and distinguish it from other elements of the same kind. This will be useful (necessary, in fact) when you write programs to process the information in this form control that will be “submitted” via the form, although we do not make use of it here. The size attribute is used for textboxes to indicate how many characters will fit into the text box “window” as displayed on the screen. Users can type more characters, but only the first size characters will be displayed (and seen by the user). We have chosen a value of 7 as a reasonable one for the size of this particular textbox. There is also an attribute called maxlength (not seen here) whose value can be used to restrict the maximum number of characters that may be entered. In other words, there are actually two sizes: a “visible size” and a “maximum size”. Although we have not used it here since it did not make sense to do so, the value attribute may be used to place a “default value” into the box. In any case, it is the value of the value attri bute that will be used when this data is eventually processed as part of the form submission, since whatever is entered into the textbox by the user becomes the value of the value attribute. It is important to note that, just because we have given the name attribute of the form field that will receive the user’s height the value height, does not make it obvious to users that they are supposed to enter their height into that particular textbox. That is why we have inserted the text Height:into the table column on the left. You can (and should) also use the label element to group the text that prompts the user for a height value, and the textbox that actually receives that height, into a logical unit. For simplicity we will omit this feature temporarily and come back to it at the end of the chapter in section 5.9. 5.3.2 Radio Buttons (input Elements of Type radio) To the right of the height field we have two radio buttons, which are created by the markup in lines 2021 of Figure 5.3. The user will click one of these to specify the unit for height as either inches or centimeters. We again use the input element to create each of these radio buttons, and this time we need the value radio for the type attribute. When used for radio buttons, the behavior of the input element is a little more complicated, because radio buttons are normally used in a group. Radio buttons in the same group are identified by having the same value for the name attribute. We distinguish them based on the values of their value attributes. In our example, the first group of radio buttons contains two buttons and is found to the right of the height field. The common name for this group of radio buttons is heightUnit. The value of the value attribute for the first radio button in the group is in, corresponding to “inches”. The second radio button has the value cm for the value attribute, corresponding to “centimeters”. Using the same name for both radio buttons ensures that only one of them can be selected at a time by the user. You may want to verify this fact by opening the file bmi2.html in a browser and clicking on each of these radio buttons in turn. The thing to note is that if one button is “selected”, that is, has been clicked and shows a “bullet” in the center of its circle, and then you click the other button, the first one is “deselected” as the second one becomes “selected”. The next row is essentially identical to the previous one in structure and format. It is used to obtain the user’s weight and choice of weight unit. Again, we have three input elements. The first input element, of type text, is named weight. The next two input elements give us radio but tons grouped under the name weightUnit, with values lb (for pounds) and kg (for kilograms) Checkboxes (input Elements of Type checkbox) The third type of input element lets us create checkboxes. An example of a checkbox also appears in bmi2.html, in the table row just below the row for weight input (see line 36 of Figure 5.3).

Checkboxes are similar to radio buttons in many ways. Multiple checkboxes can also be grouped together under the same name and distinguished by value just like radio buttons, even though our example only uses a single checkbox. Our checkbox has the name details and the value yes. You will note that in the display of Figure 5.4 the checkbox is not checked by default. This is the typical case, and the user is asked to check the textbox if he or she wishes to take the option represented by that particular checkbox. There may, however, be times when you want the check box checked by default. To accomplish this, the input element also has another attribute with the name checked. This attribute need only be present to cause its intended effect. It does not need a value. The point of including the attribute is to have the checkbox checked by default when the page is displayed, should you wish to do that. The attribute checked can also be used (and is, in fact, more often used) for radio buttons, but in that case you should make sure that only one of the radio buttons in the group is checked (the one the user is most likely to choose, presumably). It should be noted that data from input controls of type radio or checkbox is actually “submitted” by the submit button only if the corresponding checked attribute is present. 5.4 The select and option Elements for Dropdown List-boxes In the previous section we illustrated how radio buttons can be used to allow a user to specify a choice of units for weight and height. Another method for letting the user pick an option is through a dropdown list-box (also called a dropdown menu, but we call this particular version a dropdown list-box to distinguish it from the dropdown menus we introduce in Chapter 7). Drop down list-boxes are especially useful when the number of options is large, because they use much less display space on a web page. The web page bmi3.html displayed in FIGURE 5.5 shows a dropdown list-box for units of height and weight. The corresponding form element markup is shown in FIGURE 5.6. We will only focus on the code between the ... pairs of tags (the select ele ments) in bmi3.html (lines 20–23 and lines 29–32). The rest of the file shows you nothing you have not already seen in bmi2.html. Any select element used to create a web page dropdown list-box should be provided with a value for its name attribute for the same reasons you would give a name to an input element. Each text item corresponding to an option in the dropdown list-box is specified by the text content of an ... tag pair. By default the option that appears first in the markup will appear in the little window of the dropdown list-box, so you should leave that option empty if you want the box to appear empty. In our example, the select element for the weight unit has attribute name="weightUnit" and the select element for the height has attribute unit name="heightUnit". Both of these dropdown listboxes have only two options. We have not included name attributes for the option elements, but when submitting a form you may want to do this as well. 5.5 What Is Missing from the BMI Calculator Web Page? If you look at the display in Figure 5.5 for a moment, it should be clear that there is something missing. This web page allows the user to enter some information that will be needed to perform a certain kind of calculation, but does not provide any visible way to cause the calculation to take place. In other words, it is missing a “button” of some kind that the user can click on, and after which the BMI calculation will be performed and the results displayed to the user. In this chapter we are concerned only with displaying the button, not activating it. As we have said earlier, we will deal with the calculations in the next chapter. But bear with us for a bit while we also postpone the introduction of buttons onto our BMI calculator page. In the next couple of sections we will start our second example, the feedback form, and use it to introduce both the textarea element (for extended text entry) and the submit and reset buttons for data submission and form reset. Then we will come back to our BMI calculator and extend it with buttons and some other enhancements. 5.6 The textarea Element Providing a mechanism for getting feedback from its visitors should be one of the essential fea tures of any ecommerce website. If the right kind of information is obtained and used properly, the user experience can be enhanced and sales inc It is a good idea to separate the name into two fields, one for the first name and one for the last name. This will help avoid any confusion between first and last names, and also allow you to personalize any response to the user by using his or her first name, or a salutation followed by the last name, depending on the context. The textarea element allows you to provide a large text area for the user to enter a message, since you can specify its size by supplying the desired number of rows and columns as values of its rows and cols attributes. Some users may just want to have a oneway communication. Others may wish to receive a reply. Therefore, we give the user a choice of receiving a reply using a checkbox at the bottom. We do need to point out a couple of things about the HTML markup of the form in feedback1.html (see Figure 5.7). First, since the salutation is obtained from a dropdown listbox, we have again used a select element to contain the options. However, since we did not want to presume any particular salutation, we have left the content of the first option element empty (see line 17), causing the dropdown listbox to appear empty in Figure 5.8. Actually, if you look closely you will see that it is not quite empty. Its content is the HTML entity , a useful entity whenever you want to ensure that you have an actual space at some location. In fact it's a “nonbreaking” space, so a new line will not be created at that space, which is not a feature you need at this point. Leaving all content out of the option element would work fine as far as display ing the page is concerned, but if you are using an editor to format your code you need to be careful that it does not remove empty elements from your markup. This may be something you want most f the time, but not in this situation, so putting this HTML entity in as the content avoids that problem and has the same effect as an empty option element. There is nothing unusual about the following five fields that are used to collect the first name, last name, email address, phone number, and subject. They are all textboxes created by using input elements with their type attributes set to text. The second new thing of interest in this form is, of course, the textarea element in line 46. This particular textarea element has no content between its and tag pair, but if you wished to have some default text displayed when the page is rendered, you could supply that text as the element content. Just like other form fields, the textarea tag has an attribute called name, which we have set to the value message. Two additional attributes for the textarea tag are rows and cols, and they allow us to specify the height (number of rows) and width (number of columns, or characters) of the text area, respectively. We have chosen to have 6 lines (rows), each one allowing for 30 characters (columns). By default, a textarea element is “scrollable”. That means a user can enter more than six lines of text, and the textarea control will then show a vertical scroll bar on the right. The option that determines whether a user receives a reply uses an input element with the type attribute set to a value of checkbox. The name of the field is reply. We have set the value to be yes, in case the user does check the checkbox and data from this control is submitted. There is no checked attribute because we want the checkbox to be unchecked by default when the page displays. 5.7 The submit and reset Button Elements In general, every form needs to “submit” the data entered by the user to be processed in some way. This capability is usually provided by a “submit button”. Another useful button to have on a form is a “reset button”, which can be used to clear any information that may have been entered and give the user a chance to start over. FIGURE 5.9 shows our complete feedback form that includes buttons for submitting and resetting. Since the rest of the form is exactly the same as the one in feedback1.html, we show in FIGURE 5.10 only the HTML markup that creates these buttons. We use the versatile input element to create both the submit and reset buttons. The type attribute must be set to submit to get a submit button. The value attribute for our submit button is set to Send Feedback, which is the text label that appears on the button in the display of the form. A value of reset for the type attribute of the input element creates a reset button. Its value attribute is set to the text Reset Form, which appears as the visible label on the reset but ton when the web page containing the form is displayed. The reset button is immediately active, in the sense that if you enter some data and click on the reset button, the data will disappear from the form. However, we have not yet activated the submit button in the sense of connecting it to a script (program) that will process the data, and will do so only in a later chapter when we have some programming code (a script) to respond when a user clicks on this button and data is submitted. 5.8 Organizing Form Controls with the fieldset and legend Elements Usually, the highlevel view of a web form consists of several logical partsrequired input, optional input, processing buttons, and so on. Each logical part, in turn, consists of a group of fields to col lect or deal with in some way a particular category of information, and we now return to our BMI example to discuss how we can emphasize these logical groupings in a web page display. We have a more refined version of our BMI calculator form shown in the display of ch05/bmi4.html in FIGURE 5.11. From the figure you can see how the three “logical chunks”