Forms

The contents and slides of this topic are used with permission from:

- Jennifer Robbins, Learning Web Design, O'Reilly, 5th edition, May 2018, ISBN 978-1-491-96020-2.
- Paul S. Wang, Dynamic Web programming and HTML5, Routledge, 1 edition, 2012, ISBN 1439871825.

1

Forms

- How forms work
- The `form` element
- Text entry controls
- Buttons
- Menus
- Specialized inputs
- Form accessibility
- Form design tips

2

How Forms Work

Web forms have two components:

- The **form on the page** that collects input
- An **application on the server** that processes the collected information

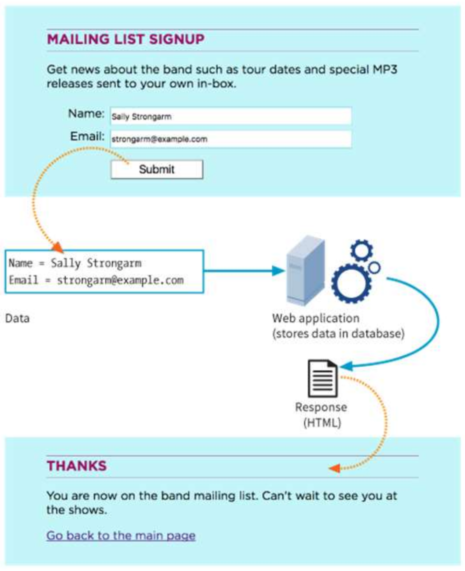
3

Web Form Transaction

1. Browser renders the form inputs as indicated in the markup.

2. User enters information in the form and hits Submit.

3. The browser encodes the information entered and sends it to the server.


MAILING LIST SIGNUP
Get news about the band such as tour dates and special MP3 releases sent to your own in-box.
Name: Sally Strongarm
Email: strongarm@example.com
Submit
Name = Sally Strongarm
Email = strongarm@example.com
Data
Web application (stores data in database)
Response (HTML)
THANKS
You are now on the band mailing list. Can't wait to see you at the shows.
Go back to the main page

4. The application processes the information.

5. The application returns a response (for example, a thank you message or reloading the page).

4

Web Processing Applications

Web forms may be processed by any of the following technologies:

- PHP (*.php*)
- Microsoft ASP (*.asp*)
- Microsoft ASP.net (*.aspx*)
- Ruby on Rails
- JavaServer Pages (*.jsp*)
- Python

5

The form Element

```
<form action="URL" method="POST or GET">
  <!-- Form content and inputs here -->
</form>
```

- The **`form`** element is a container for all the content in the form and its input controls.
- The **`action`** and **`method`** attributes are necessary for interacting with the processing program.

6

The action Attribute

```
<form action="mailinglist.php" method="POST">
```

The **`action`** attribute provides the location of the script or application that will process the collected form data.

7

The method Attribute

```
<form action="mailinglist.php" method="POST">
```

The **`method`** attribute specifies how the encoded information should be sent to the server (GET or POST):

- **`GET`**: The encoded data is tacked onto the URL sent to the server:

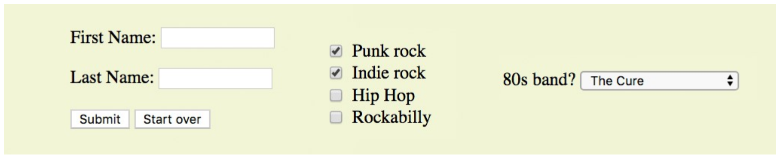
```
 get
http://www.bandname.com/mailinglist.php?name=Sally%20Stron
garm&email=strongarm%40example.com
```

- **`POST`**: Data is send in a separate transaction and can be encrypted with HTTPS.

NOTE: POST is the most common method.

8

Form Control Elements



Form control elements (also called **widgets**) collect data from the user. A few examples:

```
<input type="text">
<input type="submit">
<input type="checkbox">
<select>
```

9

Form Control Elements (cont’d)

Form controls collect data in **variable/value pairs.**

Examples:

variable = "email"
value = jen@example.com

variable = "color"
value = green

10

Variables (the name Attribute)

- A **variable** is a bit of information collected by a form control (example: the user's last name).
- The required **`name`** attribute in the control element provides the name of the variable for that control:

```
<input name="lastname">
```

NOTE: The variable name is also programmed into the web processing script or app, so the name in the markup must match the name in the processor.

11

Values

- The data entered or selected by the user in the form control is the **value** that gets sent to the server. It is paired with the variable for that control.
- You can provide a default value with the **`value`** attribute:

```
Name: <input name="lastname" value="Unknown">
```

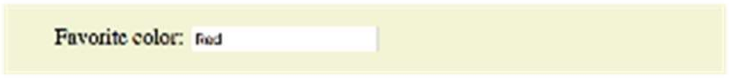
In this example, if the text input is left blank, the value "Unknown" would be sent to the server for the variable "lastname".

12

FORM CONTROL ELEMENTS

Text Entry Input

`<input type="text">`



```
<input type="text" name="color" value="Red" maxlength="24">
```

`type`: Type of input control, in this case a single-line text field

`name`: Required variable name

`value`: Default text that appears in the field and is sent to server if the field is left blank

`maxlength`, **`minlength`**: Sets a character limit for the field

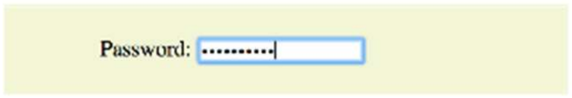
`size`: The length of the field in number of characters (CSS is more common for sizing)

13

FORM CONTROL ELEMENTS

Password Field

`<input type="password">`



```
<input type="password" name="pswd" maxlength="10">
```

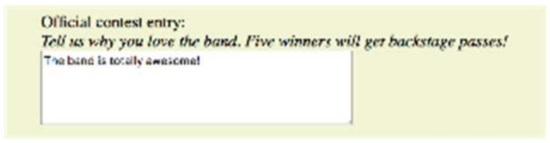
- Like a text entry field, except the characters are obscured from view
- The data entered is *not* encrypted on the way to the server (unless it uses HTTPS, a secure web protocol).

14

FORM CONTROL ELEMENTS

Multi-line Text Entry

`<textarea> </textarea>`


Official contest entry:
Tell us why you love the band. Five winners will get backstage passes!
The band is totally awesome!

```
<textarea name="entry" rows="6" cols="64">This band is totally
awesome!</textarea>
```

The content of the `textarea` element is the default value.

`rows`: The number of rows tall the field is initially drawn (users can write more)

`cols`: Width of initial text field, in number of characters

`maxlength`, `minlength`: Limits the number of characters that can be entered

15

FORM CONTROL ELEMENTS

Specialized Text Entry Fields

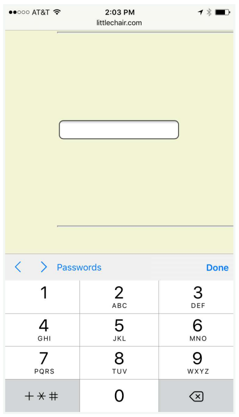
```
<input type="search">
<input type="email">
<input type="tel">
<input type="url">
```

- These input types are more semantically rich than a default text field.
- Browsers may provide keyboards tailored to the input type.
- Browsers may validate entries on the fly without using the server application.

16

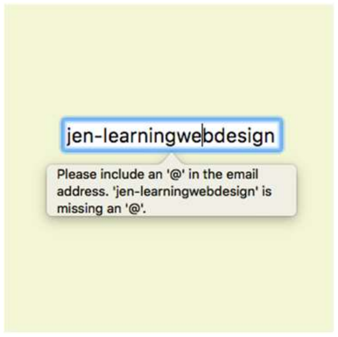
Specialized Text Entries (cont'd)

`<input type="tel" name="">`



Numerical keyboard provided on iOS

`<input type="email" name="">`



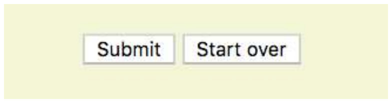
Please include an '@' in the email address. 'jen-learningwebdesign' is missing an '@'.

Opera looks for email address structure

17

FORM CONTROL ELEMENTS

Submit and Reset Buttons



`<input type="submit">`

- **Submits** the collected form data to the server. Does not require a variable name (`name` attribute):

`<input type="reset" value="Start over">`

- **Resets** the form to its defaults
- Less common with the rise of JavaScript for form handling
- Change the button text with the `value` attribute.

18

FORM CONTROL ELEMENTS

Custom Buttons

`<button> </button>`

The **`button`** element is used for creating custom buttons with JavaScript.

`<input type="button">`

Creates a custom button that has no predefined function and can be customized with JavaScript

`<input type="image" alt="">`

Allows an image to be used as a button to replace the Submit button. It requires a descriptive **`alt`** attribute value.

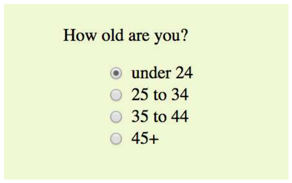
19

FORM CONTROL ELEMENTS

Radio Buttons

`<input type="radio">`

Only one radio button may be selected at a time.



How old are you?

- ● under 24
- ○ 25 to 34
- ○ 35 to 44
- ○ 45+

```
<p>How old are you?</p>
<ol>
 <li><input type="radio" name="age" value="under24" checked> under 24</li>
 <li><input type="radio" name="age" value="25-34"> 25 to 34</li>
 <li><input type="radio" name="age" value="35-44"> 35 to 44</li>
 <li><input type="radio" name="age" value="over45"> 45+</li>
</ol>
```

NOTE: You can't belong to more than one age group, so radio buttons are the right choice for this list.

20

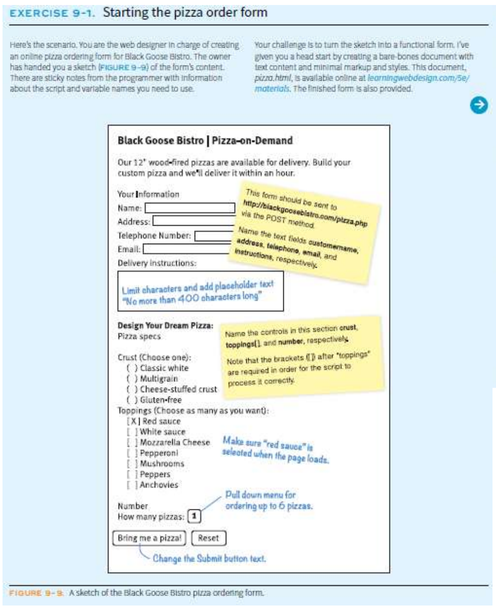
Radio Buttons (cont'd.)

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

- Applying the same variable name to input elements binds them together as a mutually exclusive set of options.
- The value for each button must be provided with the **`value`** attribute.
- The **`checked`** attribute causes an option to be selected when the page loads. Only one input may be checked.

21

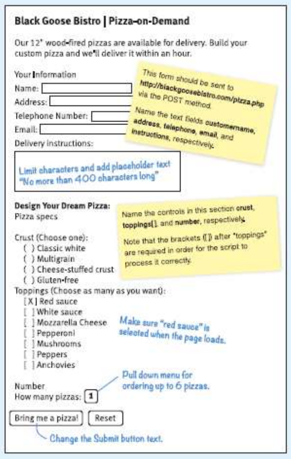
Example



EXERCISE 9-1. Starting the pizza order form

Here's the scenario. You are the web designer in charge of creating an online pizza ordering form for Black Goose Bistro. The owner has handed you a sketch (FIGURE 9-9) of the form's content. There are sticky notes from the programmer with information about the script and variable names you need to use.

Your challenge is to turn the sketch into a functional form. I've given you a head start by creating a bare-bones document with text content and minimal markup and styles. This document, *pizza.html*, is available online at *learningwebdesign.com/5e/materials*. The finished form is also provided.



Black Goose Bistro | Pizza-on-Demand

Our 12" wood-fired pizzas are available for delivery. Build your custom pizza and we'll deliver it within an hour.

Your Information
Name:
Address:
Telephone Number:
Email:
Delivery instructions:

Limit characters and add placeholder text "No more than 400 characters long"

This form should be sent to **http://blackgoosebistro.com/pizza.php** via the POST method.

Name the text fields **customername**, **address**, **telephone**, **email**, and **instructions**, respectively.

Design Your Dream Pizza:
Pizza specs

Name the controls in this section **crust**, **toppings[]**, and **number**, respectively.

Note that the brackets ([]) after "toppings" are required in order for the script to process it correctly.

Crust (Choose one):
( ) Classic white
( ) Multigrain
( ) Cheese-stuffed crust
( ) Gluten-free
Toppings (Choose as many as you want):
[X] Red sauce
[ ] White sauce
[ ] Mozzarella Cheese
[ ] Pepperoni
[ ] Mushrooms
[ ] Peppers
[ ] Anchovies

Make sure "red sauce" is selected when the page loads.

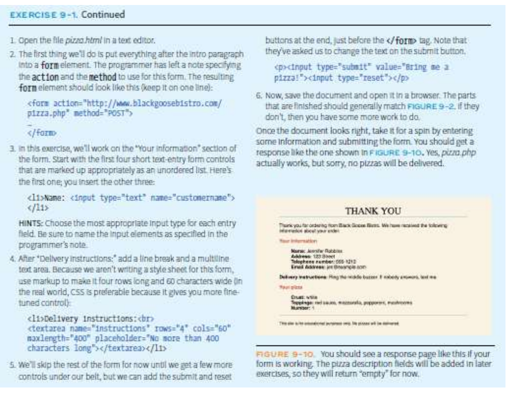
Number
How many pizzas: 1

Pull down menu for ordering up to 6 pizzas.

Bring me a pizza! | Reset

Change the Submit button text.

FIGURE 9-9. A sketch of the Black Goose Bistro pizza ordering form.



EXERCISE 9-1. Continued

1. Open the file *pizza.html* in a text editor.
2. The first thing we'll do is put everything after the intro paragraph into a `form` element. The programmer has left a note specifying the `action` and the `method` to use for this form. The resulting `form` element should look like this (keep it on one line):

```
<form action="http://www.blackgoosebistro.com/
pizza.php" method="POST">
…
</form>
```

3. In this exercise, we'll work on the "Your Information" section of the form. Start with the first four short text-entry form controls that are marked up appropriately as an unordered list. Here's the first one; you insert the other three:

```
<li>Name: <input type="text" name="customername">
</li>
```

HINTS: Choose the most appropriate input type for each entry field. Be sure to name the input elements as specified in the programmer's note.

4. After "Delivery instructions:" add a line break and a multiline text area. Because we aren't writing a style sheet for this form, use markup to make it four rows long and 60 characters wide (in the real world, CSS is preferable because it gives you more fine-tuned control):

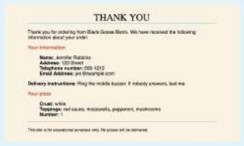
```
<li>Delivery instructions:<br>
<textarea name="instructions" rows="4" cols="60"
maxlength="400" placeholder="No more than 400
characters long"></textarea></li>
```

5. We'll skip the rest of the form for now until we get a few more controls under our belt, but we can add the submit and reset buttons at the end, just before the `</form>` tag. Note that they've asked us to change the text on the submit button.

```
<p><input type="submit" value="Bring me a
pizza!"><input type="reset"></p>
```

6. Now, save the document and open it in a browser. The parts that are finished should generally match FIGURE 9-2. If they don't, then you have some more work to do.

Once the document looks right, take it for a spin by entering some information and submitting the form. You should get a response like the one shown in FIGURE 9-10. Yes, *pizza.php* actually works, but sorry, no pizzas will be delivered.



THANK YOU

Thank you for ordering from Black Goose Bistro. We have received the following information about your order:

Your Information

Name: Jennifer Robbins
Address: 123 Street
Telephone number: 555-1212
Email Address: jen@example.com

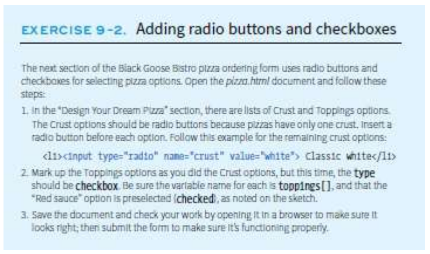
Delivery instructions: Ring the middle buzzer. If nobody answers, text me.

Your pizza

Crust: white
Toppings: red sauce, mozzarella, pepperoni, mushrooms
Number: 1

This site is for educational purposes only. No pizzas will be delivered.

FIGURE 9-10. You should see a response page like this if your form is working. The pizza description fields will be added in later exercises, so they will return "empty" for now.



EXERCISE 9-2. Adding radio buttons and checkboxes

The next section of the Black Goose Bistro pizza ordering form uses radio buttons and checkboxes for selecting pizza options. Open the *pizza.html* document and follow these steps:

1. In the "Design Your Dream Pizza" section, there are lists of Crust and Toppings options. The Crust options should be radio buttons because pizzas have only one crust. Insert a radio button before each option. Follow this example for the remaining crust options:

```
<li><input type="radio" name="crust" value="white"> Classic white</li>
```

2. Mark up the Toppings options as you did the Crust options, but this time, the `type` should be `checkbox`. Be sure the variable name for each is `toppings[]`, and that the "Red sauce" option is preselected (`checked`), as noted on the sketch.
3. Save the document and check your work by opening it in a browser to make sure it looks right; then submit the form to make sure it's functioning properly.

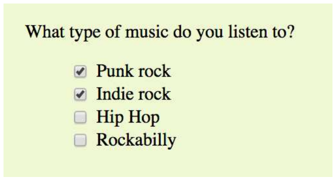
22

FORM CONTROL ELEMENTS

Checkbox Buttons

`<input type="checkbox">`

Multiple checkbox buttons may be selected.


What type of music do you listen to?

- [x] Punk rock
- [x] Indie rock
- [ ] Hip Hop
- [ ] Rockabilly

```
<p>What type of music do you listen to?</p>
<ul>
  <li><input type="checkbox" name="genre" value="punk" checked> Punk rock</li>
  <li><input type="checkbox" name="genre" value="indie" checked> Indie rock</li>
  <li><input type="checkbox" name="genre" value="hiphop"> Hip Hop</li>
  <li><input type="checkbox" name="genre" value="rockabilly"> Rockabilly</li>
</ul>
```

NOTE: You can like more than one type of music, so checkbox buttons are the right choice for this list.

23

Checkbox Buttons (cont’d)

`<input type="checkbox" value="">`

- Applying the same variable name to input elements binds them together as a group.
- The value for each button must be provided with the **`value`** attribute.
- The **`checked`** attribute causes an option to be selected when the page loads. Multiple buttons in a group may be checked.

24

FORM CONTROL ELEMENTS

Drop-down Menus

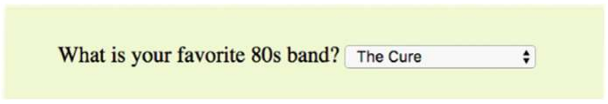
```
<select> </select>
<option> </option>
<optgroup> </optgroup>
```

- The **`select`** element creates a drop-down menu when there is no **`size`** attribute (or if `size="1"`).
- The **`select`** element contains some number of **`option`** elements.
- The content of the **`option`** element is the value sent to the server (or one can be provided with the **`value`** attribute).

25

Drop-down Menus (cont'd.)

The select menu drops down to reveal options when the user clicks on it.

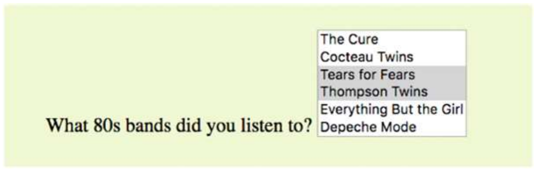

What is your favorite 80s band? The Cure

```
<p>What is your favorite 80s band?
<select name="EightiesFave">
   <option>The Cure</option>
   <option>Cocteau Twins</option>
   <option>Tears for Fears</option>
   <option>Thompson Twins</option>
   <option value="EBTG">Everything But the Girl</option>
   <option>Depeche Mode</option>
   <option>The Smiths</option>
   <option>New Order</option>
</select>
</p>
```

26

FORM CONTROL ELEMENTS

Scrolling Menus



```
<p>What is your favorite 80s band?
<select name="EightiesFave" size="6" multiple>
   <option>The Cure</option>
   ...
</select>
</p>
```

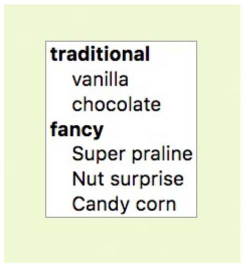
- The same markup as drop-down menus, but the **`size`** attribute specifies how many lines to display.
- The **`multiple`** attribute allows more than one option to be selected.

27

Scrolling Menus (cont'd)

Use the **`optgroup`** element to create a conceptual group of options.

The **`label`** attribute provides the the heading for the group:



```
<select name="icecream" size="7" multiple>
  <optgroup label="traditional">
    <option>vanilla</option>
    <option>chocolate</option>
  </optgroup>
  <optgroup label="fancy">
    <option>Super praline</option>
    <option>Nut surprise</option>
    <option>Candy corn</option>
  </optgroup>
</select>
```

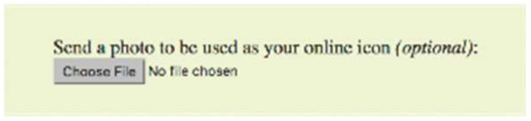
28

FORM CONTROL ELEMENTS

File Upload Control

`<input type="file">`

File input (on Chrome browser)


Send a photo to be used as your online icon *(optional)*:

```
<form action="/client.php" method="POST" enctype="multipart/form-data">
  <label>Send a photo to be used as your online icon <em>(optional)</em><br>
  <input type="file" name="photo"></label>
</form>
```

- The file input type allows a user to select a document from their hard drive to be submitted with the form data.
- The method must be set to POST, and the encoding type must be included.

29

FORM CONTROL ELEMENTS

Hidden Control

`<input type="hidden">`

```
<input type="hidden" name="success-link"
       value="http://www.example.com/thankyou.html">
```

- Sometimes it is necessary to feed values to the processing script/app that don’t come from the user.
- Hidden controls don’t display in the browser.

30

FORM CONTROL ELEMENTS

Date and Time Controls

```
<input type="date">
<input type="time">
<input type="datetime-local">
<input type="month">
<input type="week">
```

```
<input type="date" name="birthday" value="2017-01-14">
```

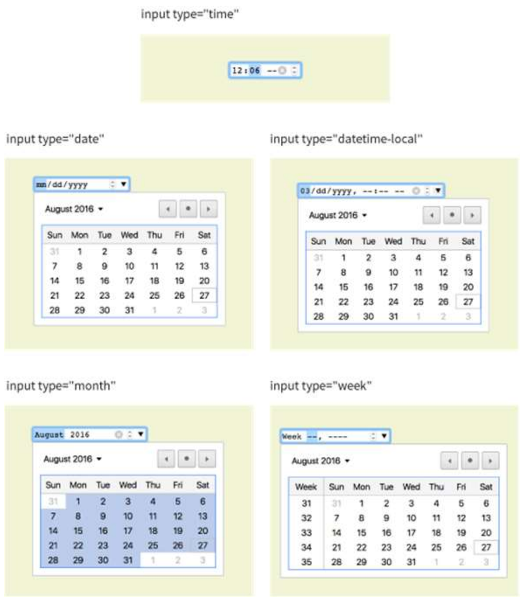
A starting value may be provided in standard date-time format.

31

Date and Time Controls (cont'd)

Browsers may display date and time selection widgets (not universally supported).

On non-supporting browsers, date and time inputs display as usable text-entry fields.



32

FORM CONTROL ELEMENTS

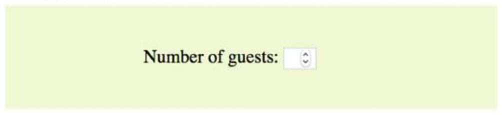
Numerical Controls

```
<input type="number">
<input type="range">
```

Number and range controls collect numerical data. Browsers may render counter or slider widgets.

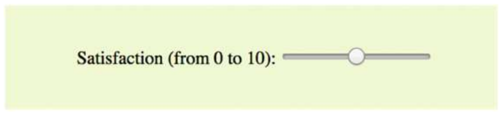
Both types accept **`min`** and **`max`** attributes for setting limits on the allowed values.

input type="number"



Number of guests:

input type="range"



Satisfaction (from 0 to 10):

33

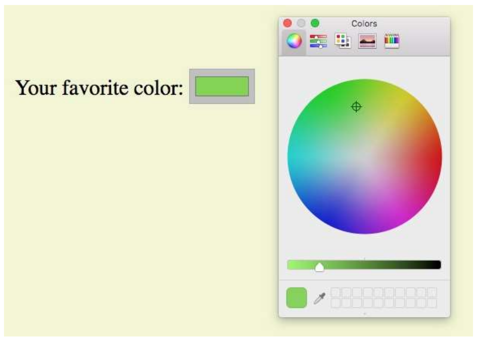
FORM CONTROL ELEMENTS

Color Selector

```
<input type="color">
```

The color input type is intended to provide a pop-up color picker.

It is not well supported. Non-supporting browsers display a text-entry field.



Your favorite color:

34

Form Accessibility

- Users may not be able to see the form. They may be listening to it with a screen reader.
- Whereas sighted users can see at a glance how elements are organized, form accessibility features create semantic connections between form components.

35

FORM ACCESSIBILITY

Labels

`<label> </label>`

The `label` element associates a descriptive label with a form field.

Implicit association
The label text and form control are both contained within the `label` element:

```
<li><label>Red <input type="radio" name="color"
value="red"></label></li>
```

Explicit association
Matches the label with the control's ID reference using the `for` attribute:

```
<li><label for="form-colors-red">Red</label>  <input type="radio"
name="color" value="red id="form-colors-red"></li>
```

36

FORM ACCESSIBILITY

Fieldsets and Legends

`<fieldset> </fieldset>`

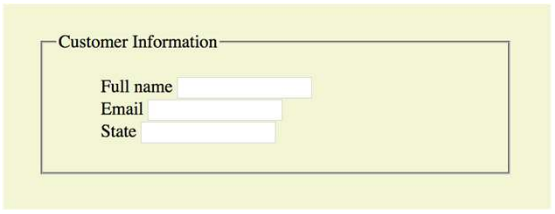
`<legend> </legend>`

`fieldset`
Indicates a logical grouping of controls (examples: credit card name, number, date, etc.). By default, rendered with a box around the set of controls.

`legend`
Provides a caption for the enclosed fields. By default, it's displayed along the top edge of the fieldset.

37

Fieldsets and Legends (cont'd)



```
<fieldset>
  <legend>Customer Information</legend>
  <ul>
    <li><label>Full name: <input type="text" name="fullname"></label></li>
    <li><label>Email: <input type="text" name="email"></label></li>
    <li><label>State: <input type="text" name="state"></label></li>
  </ul>
</fieldset>
```

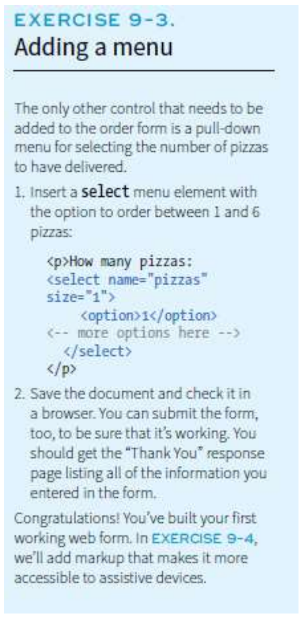
38

Form Design Tips

- Avoid unnecessary questions.
- Consider the impact of label placement. Labels above fields tend to lead to faster completion.
- Choose input types carefully.
- Group related inputs.
- Primary actions (e.g., "Buy") should be visually dominant to secondary actions (e.g., "Back").

39

Exercise



EXERCISE 9-3. Adding a menu

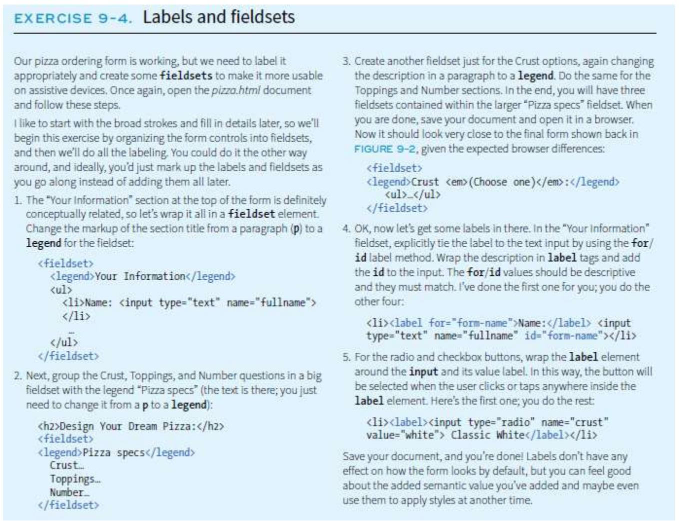
The only other control that needs to be added to the order form is a pull-down menu for selecting the number of pizzas to have delivered.

1. Insert a **select** menu element with the option to order between 1 and 6 pizzas:

```
<p>How many pizzas:
<select name="pizzas"
size="1">
      <option>1</option>
<-- more options here -->
   </select>
</p>
```

2. Save the document and check it in a browser. You can submit the form, too, to be sure that it's working. You should get the "Thank You" response page listing all of the information you entered in the form.

Congratulations! You've built your first working web form. In EXERCISE 9-4, we'll add markup that makes it more accessible to assistive devices.



EXERCISE 9-4. Labels and fieldsets

Our pizza ordering form is working, but we need to label it appropriately and create some **fieldsets** to make it more usable on assistive devices. Once again, open the *pizza.html* document and follow these steps.

I like to start with the broad strokes and fill in details later, so we'll begin this exercise by organizing the form controls into fieldsets, and then we'll do all the labeling. You could do it the other way around, and ideally, you'd just mark up the labels and fieldsets as you go along instead of adding them all later.

1. The "Your Information" section at the top of the form is definitely conceptually related, so let's wrap it all in a **fieldset** element. Change the markup of the section title from a paragraph (**p**) to a **legend** for the fieldset:

```
<fieldset>
  <legend>Your Information</legend>
  <ul>
    <li>Name: <input type="text" name="fullname">
    </li>
    …
  </ul>
</fieldset>
```

2. Next, group the Crust, Toppings, and Number questions in a big fieldset with the legend "Pizza specs" (the text is there; you just need to change it from a **p** to a **legend**):

```
<h2>Design Your Dream Pizza:</h2>
<fieldset>
<legend>Pizza specs</legend>
  Crust…
  Toppings…
  Number…
</fieldset>
```

3. Create another fieldset just for the Crust options, again changing the description in a paragraph to a **legend**. Do the same for the Toppings and Number sections. In the end, you will have three fieldsets contained within the larger "Pizza specs" fieldset. When you are done, save your document and open it in a browser. Now it should look very close to the final form shown back in FIGURE 9-2, given the expected browser differences:

```
<fieldset>
<legend>Crust <em>(Choose one)</em>:</legend>
   <ul>…</ul>
</fieldset>
```

4. OK, now let's get some labels in there. In the "Your Information" fieldset, explicitly tie the label to the text input by using the **for**/**id** label method. Wrap the description in **label** tags and add the **id** to the input. The **for**/**id** values should be descriptive and they must match. I've done the first one for you; you do the other four:

```
<li><label for="form-name">Name:</label> <input
type="text" name="fullname" id="form-name"></li>
```

5. For the radio and checkbox buttons, wrap the **label** element around the **input** and its value label. In this way, the button will be selected when the user clicks or taps anywhere inside the **label** element. Here's the first one; you do the rest:

```
<li><label><input type="radio" name="crust"
value="white"> Classic White</label></li>
```

Save your document, and you're done! Labels don't have any effect on how the form looks by default, but you can feel good about the added semantic value you've added and maybe even use them to apply styles at another time.

40