Chapter 4

A crash course in HTML5 and CSS3

Objectives

Applied

1. Code the HTML that defines the content and structure of a web page using any of the elements and attributes presented in this chapter.
2. Code the CSS that provides the formatting of the web pages using any of the techniques presented in this chapter.
3. Code HTML forms that use text boxes, check boxes, radio buttons, combo boxes, list boxes, and buttons to send data to a servlet.

Objectives (continued)

Knowledge

1. Describe the use of HTML's head, title, and body elements.
2. Describe the use of HTML's h1, h2, p, img, form, a, input, label, and br elements.
3. Described the use of HTML5's header, nav, aside, and footer elements.
4. Describe the use of HTML's div and span tags.
5. Describe the use of HTML's table, tr, th, and td tags.
6. Explain why it's a best practice to use an external style sheet.

Objectives (continued)

Knowledge

7. Name and describe three types of CSS selectors.
8. Name and describe the components of a CSS rule set.
9. Name and describe the components of a CSS rule.
10. Describe how to use HTML to display text boxes, check boxes, radio buttons, combo boxes, list boxes, and buttons.
11. Explain how to use an HTML form to send data to a servlet.

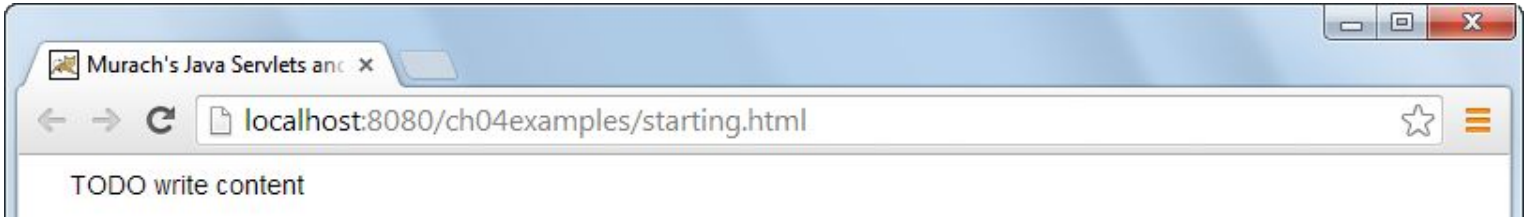
The generated HTML for a new page

```
<!DOCTYPE html>
<html>
    <head>
        <title>TODO supply a title</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width">
    </head>
    <body>
        <div>TODO write content</div>
    </body>
</html>
```

The HTML after it has been modified

```
<!DOCTYPE html>
<html>
    <head>
        <title>Murach's Java Servlets and JSP</title>
        <link rel="stylesheet" href="styles/main.css"
              type="text/css"/>
        <meta charset="UTF-8">
    </head>
    <body>
        <div>TODO write content</div>
    </body>
</html>
```

The title displayed in the browser's tab



How to start a web page

- The DOCTYPE element specifies the document uses HTML5.
- The title element specifies the name the browser shows in its title bar or tab.
- The link element references the external style sheet that contains the CSS for the page.

Common HTML elements

Element	Type	Defines
`h1`	Block	A level-1 heading with content in bold at 200% of the base font size.
`h2`	Block	A level-2 heading with content in bold at 150% of the base font size.
`p`	Block	A paragraph at 100% of the base font size.
`img`	Block	An image.
`form`	Block	A form that can be submitted to the web server for processing.
`a`	Inline	A link that goes to another page or a location on the current page when clicked.
`input`	Inline	A control on a form like a text box or button.
`label`	Inline	A label that identifies a control on a form.
`br`		A line break that starts a new line.

How to code HTML elements

Two block elements with opening and closing tags

```
<h1>Email List application</h1>
<p>Here is a list of links:</p>
```

Two self-closing tags

```
<br>
<img src="logo.gif" alt="Murach Logo">
```

How to code the attributes for HTML elements

How to code an opening tag with attributes

```
<a href="contact.html" title="Click to Contact Us">
```

How to code a Boolean attribute

```
<input type="checkbox" name="mailList" checked>
```

How to code an HTML comment

```
<!-- The text in a comment is ignored -->
```

How to code a character entity for a space

```
<td> </td>
```

An introduction to HTML

- An *HTML document* contains *HTML elements* that specify the content of a web page.
- By default, *block elements* are displayed on new lines, but *inline elements* flow to the right of the elements that precede it.
- An *attribute* consists of an attribute name, an equals sign, and a value in quotation marks. But to show that a Boolean attribute is on, code just the name of the attribute.
- *Comments* can be used to describe or *comment out* portions of HTML code.
- *Character entities* provide for special characters, like a non-breaking space ( )

The primary HTML5 semantic elements

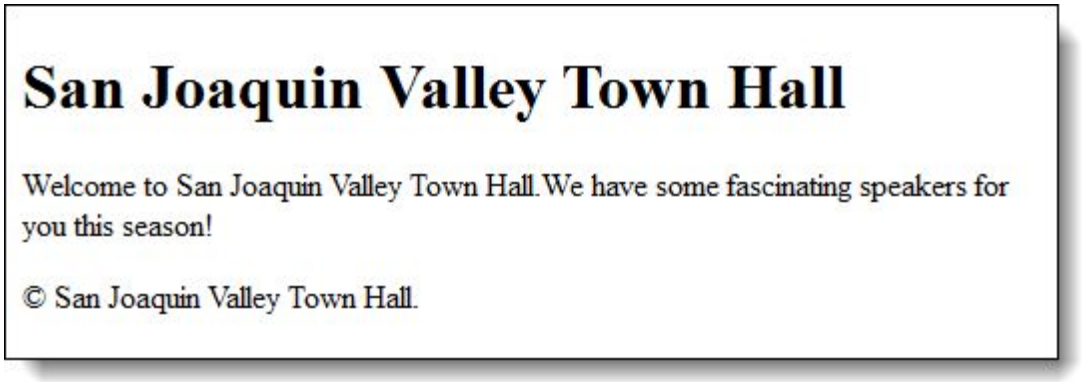
Element	Contents
`header`	The header for a page.
`section`	A generic section of a document that doesn't indicate the type of content.
`article`	A composition like an article in the paper.
`nav`	A section of a page that contains links to other pages or placeholders.
`aside`	A section of a page like a sidebar that is related to the content that's near it.
`figure`	An image, table, or other component that's treated as a figure.
`footer`	The footer for a page.

http://www.w3schools.com/html/html5_semantic_elements.asp

A page with header, section, and footer elements

```
<body>
    <header>
        <h1>San Joaquin Valley Town Hall</h1>
    </header>
    <section>
        <p>Welcome to San Joaquin Valley Town Hall. We have some
           fascinating speakers for you this season!</p>
    </section>
    <footer>
        <p>&copy; San Joaquin Valley Town Hall.</p>
    </footer>
</body>
```

The page displayed in a web browser



How to use the HTML5 semantic elements

- HTML5 provides new *semantic elements* that you should use to structure the contents of a web page. Using these elements can be referred to as *HTML5 semantics*.
- Two benefits that you get from using the semantic elements are (1) simplified HTML and CSS, and (2) improved *search engine optimization* (*SEO*).

The div and span elements

Element	Description
`div`	A block element that provides a container for other elements.
`span`	An inline element that lets you identify text that can be formatted with CSS.

The way div elements were used before HTML5

```
<div id="header">
    <h1>San Joaquin Valley Town Hall</h1>
</div>
<div id="contents">
     <p>Welcome to San Joaquin Valley Town Hall. We have some
        fascinating speakers for you this season!</p>
</div>
<div id="footer">
    <p>&copy; San Joaquin Valley Town Hall.</p>
</div>
```

Span elements in HTML for JavaScript

```
<label>Email Address:</label>
<input type="text" name="email_address1">
<span id="email_address1_error">*</span><br>

<label>Re-enter Email Address:</label>
<input type="text" name="email_address2">
<span id="email_address2_error">*</span><br>
```

The div and span elements with HTML5

- Before HTML5, div elements were used to organize the content within the body of a document. Then, the ids for these div elements were used to apply CSS formatting to the elements.
- Today, HTML5 semantic elements should replace most div elements. That makes the structure of a page more apparent.
- Before HTML5, span elements were used to identify portions of text that you could apply formatting to.
- Today, a better practice is to use elements that identify the contents.

The current browsers and their HTML5 ratings (perfect score is 500)

Browser	Release	HTML5 Test Rating
Google Chrome	33	505
Opera	20	496
Mozilla Firefox	29	467
Apple Safari	7	397
Internet Explorer	11	376

The website for these ratings

http://www.html5test.com

Guidelines for cross-browser compatibility

- Test web pages on all major browsers, including all older versions of browsers that are still commonly used.
- Use HTML5 features supported by all modern browsers, especially HTML5 semantic elements. Use two workarounds that follow so these applications run on older browsers too.

Two workarounds for HTML5 semantic elements

The JavaScript shiv

```
<script src="http://html5shiv.googlecode.com/svn/trunk/html5.js"></script>
```

The CSS rule set

```
article, aside, figure, figcaption, footer, header, nav, section {
    display: block;
}
```

How to ensure cross-browser compatibility

- Today, there are still differences in the way that different browsers handle HTML and CSS, and especially HTML5 and CSS3.
- As a developer, though, you want your web pages to work on as many different web browsers as possible. This is referred to as *cross-browser compatibility*.
- To provide for cross-browser compatibility, you need to test your applications on all of the browsers that your users might use.
- In general, Internet Explorer gives web developers the most problems because it is the least standard and hasn't provided for automatic updates.
- Eventually, all browsers will support HTML5 and CSS3 so the workarounds won't be necessary.

Two links viewed in a browser

Examples of links

Relative to the current directory

```
<a href="join.html">The Email List application 1</a><br>
<a href="email/join.html">The Email List application 2</a><br>
<a href="../">Go back one directory level</a><br>
<a href="../../">Go back two directory levels</a><br>
```

Relative to the webapps directory

```
<a href="/">Root directory of the web server</a><br>
<a href="/musicStore">Root directory of the musicStore app</a>
```

Absolute URLs

```
<a href="http://www.murach.com/email">An Internet address</a>
<a href="http://64.71.179.86/email">An IP address</a>
```

Note: you can pass information via an anchor by encoding the url with get notation:
`http://www.site.com?name=john&id=5`

The a (anchor) element

Element	Description
`a`	Defines a link to another URL. When the user clicks on the text that's displayed by the tag, the browser requests the page that is identified by the href attribute of the tag.

One attribute of the anchor element

Attribute	Description
`href`	Specifies the URL for the link.

Relative and absolute URLs

- When you code a *relative URL* in the href attribute, the URL can be relative to the current directory, which is the one for the current HTML page, or the URL can be relative to the web server's directory for web applications.
- When you code an *absolute URL*, you code the complete URL. To do that, code the name of the host or the IP address for the host.

HTML code that includes an image

```
<img src="images/murachlogo.png" alt="Murach Logo" width="100">
```

The image displayed in a browser

The image element

Element	Description
`img`	Specifies how to place a PNG, GIF, or JPEG image.

Common attributes of the Image tag

Attribute	Description
`src`	Specifies the relative or absolute URL for the GIF or JPEG file.
`alt`	Specifies the text that's displayed when the image can't be displayed.
`height`	Specifies the height of the image in pixels.
`width`	Specifies the width of the image in pixels.

Other examples

```
<img src="java.png" alt="Java Logo" width="175">

<img src="../../images/murachlogo.png" width="100">

<img src="http://www.murach.com/images/murachlogo.png"
     alt="Murach Logo" width="100">
```

Three types of image formats

- Portable Network Graphics (PNG)
- Graphic Interchange Format (GIF)
- Joint Photographic Experts Group (JPEG or JPG).

Typical uses

- JPEG or JPG for photographs and scans.
- PNG and GIF files for other types of images such as logos.

The HTML code for a table

```
<table>
    <tr>
        <th>Code</th>
        <th>Description</th>
        <th class="align_right">Price</th>
    </tr>
    <tr>
        <td>pf01</td>
        <td>Paddlefoot - The first CD</td>
        <td class="align_right">12.95</td>
    </tr>
    <tr>
        <td>jr01</td>
        <td>Joe Rut - Genuine Wood Grained Finish</td>
        <td class="align_right">14.95</td>
    </tr>
</table>
```

The table displayed in a browser

Code	Description	Price
pf01	Paddlefoot - The first CD	12.95
jr01	Joe Rut - Genuine Wood Grained Finish	14.95

The elements for working with tables

Element	Description
`table`	Defines a table.
`tr`	Defines a row.
`th`	Defines a header cell within a row.
`td`	Defines a data cell within a row.

Attributes of the td element

Attribute	Description
`colspan`	Specifies the number of columns that the cell spans.
`rowspan`	Specifies the number of rows that the cell spans.

Tables, rows, columns, and cells

- A *table* consists of *rows* and *columns*.
- The intersection of a row and column creates a *cell* that can hold data.

Three ways to provide styles

External style sheet - Code a link in the head section

```
<link rel="stylesheet" href="styles/main.css">
```

Embedded styles - Code the styles in the head section

```
<style>
    body {
        font-family: Arial, Helvetica, sans-serif;
        font-size: 87.5%;
    }
    h1 {
        font-size: 250%;
    }
</style>
```

Inline styles – Use the style attribute of an element

```
<span style="color: red; font-size: 14pt;">Warning!</span>
```

The sequence in which styles are applied

- Styles from an external style sheet
- Embedded styles
- Inline styles

Two external style sheets

```
<head>
    <title>Fresh Corn Records</title>
    <link rel="stylesheet" href="main.css">
    <link rel="stylesheet" href="order.css">
</head>
```

The sequence in which styles are applied

From the first external style sheet to the last

External, embedded, and inline style sheets

- It's a best practice to use *external style sheets* because that leads to better separation of concerns. Specifically, you separate the content for a page (HTML) from its formatting (CSS).
- Using external style sheets also makes it easy to use the same styles for two or more pages. In contrast, if you use *embedded styles* or *inline styles*, you have to copy the styles to other documents before you can use them again.
- If more than one rule for the same property is applied to the same element, the last rule overrides the earlier rules.

HTML with element types, classes, and ids

```
<body>
    <section>
        <h1 id="first_heading">The Speaker Lineup</h1>
        <p class="blue">October 19: Jeffrey Toobin</p>
        <p class="blue">November 16: Andrew Ross Sorkin</p>
    </section>
    <footer>
        <p class="blue right">Copyright SJV Town Hall</p>
    </footer>
</body>
```

Three CSS rule sets with type selectors

```
body {
    font-family: Arial, Helvetica, sans-serif;
    width: 400px;
    margin: 1em auto; }
section {
    border: 2px solid black;
    padding: 1em; }
p { margin: .25em 0 .25em 3em; }
```

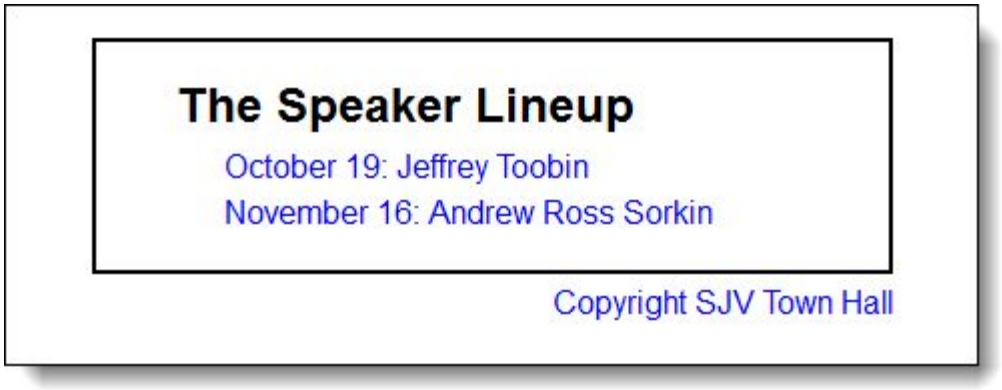
Two CSS rule sets with class selectors

```
.blue { color: blue; }
.right { text-align: right; }
```

One rule set with an id selector

```
#first_heading { margin: 0 1em .25em; }
```

The elements displayed in a browser



Element, class, and id selectors

- You code a selector for all elements of a specific type by naming the element. This is referred to as a *type* or *element selector*.
- You code a selector for an element with a class attribute by coding a period followed by the class name. Then, the rule set applies to all elements with that class name. This is known as a *class selector*.
- You code an id selector for an element with an id attribute by coding a pound sign (#) followed by the id value. This is known as an id selector.

An external style sheet

```
/* The styles for the elements */
body {
    font-family: Arial, Helvetica, sans-serif;
    font-size: 85%;
    margin-left: 2em;
    margin-right: 2em;
    width: 400px;
}
h1 {
    font-size: 140%;
    color: teal;
    margin-bottom: .5em;
}
label {
    float: left;
    width: 7em;
    margin-bottom: 0.5em;
    font-weight: bold;
}
```

An external style sheet (continued)

```
input[type="text"], input[type="email"] {  /* Attribute selectors */
    width: 15em;
    margin-left: 0.5em;
    margin-bottom: 0.5em;
}
span {
    margin-left: 0.5em;
    margin-bottom: 0.5em;
}
br {
    clear: both;
}

/* The styles for the classes */
.pad_top {
    padding-top: 0.25em;
}
.margin_left {
    margin-left: 0.5em;
}
```

Rule sets, selectors, rules, and comments

- A CSS *rule set* consists of a selector and one or more rules within braces.
- A CSS *selector* consists of the identifiers that are coded at the beginning of the rule set. If more than one selector is coded for a rule set, the selectors are separated by commas.
- A CSS *rule* consists of a *property*, a colon, a *value*, and a semicolon. Although the semicolon for the last declaration in a block is optional, it's a best practice to code it.
- To make your code easier to read, use spaces, indentation, and blank lines within a rule set.
- CSS *comments* begin with /* and end with */. A CSS comment can be coded on a single line, or it can span multiple lines.

Common properties for formatting tables

Property	Description
`border-collapse`	A keyword that determines whether space exists between the borders of adjacent cells. Possible values are “separate” and “collapse”. The default is “separate”.
`padding`	The space between the cell contents and the outer edge of the cell.
`text-align`	The horizontal alignment of text.
`vertical-align`	The vertical alignment of text.

The CSS for a table

```
table {
    border: 1px solid black;
    border-collapse: collapse;
}
th, td {
    border: 1px solid black;
    text-align: left;
    padding: .5em;
}
.align_right {
    text-align: right;
}
```

Note

- With HTML5, you should use CSS, not HTML, to format tables.

A table with collapsed borders

Code	Description	Price
pf01	Paddlefoot - The first CD	12.95
jr01	Joe Rut - Genuine Wood Grained Finish	14.95

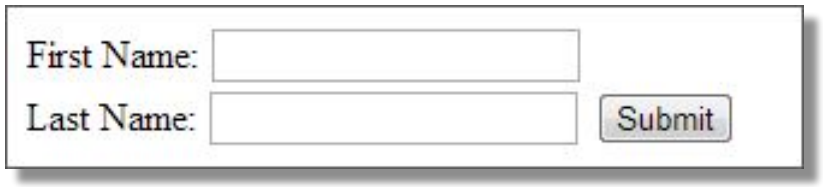
A table without collapsed borders

Code	Description	Price
pf01	Paddlefoot - The first CD	12.95
jr01	Joe Rut - Genuine Wood Grained Finish	14.95

The HTML code for a form

```
<form action="contactList" method="post">
    <label>First Name:</label>
    <input type="text" name="firstName"><br>
    <label>Last Name:</label>
    <input type="text" name="lastName">
    <input type="submit" value="Submit">
</form>
```

The form displayed in a browser



Terms

- A *form* contains one or more *controls* such as text boxes, buttons, check boxes, and list boxes.

Elements for working with a simple form

Element	Description
`form`	Defines the start and end of the form.
`input`	Defines the input type.

Attributes of the form element

Attribute	Description
`action`	Specifies the URL of the servlet or JSP that's called when the user clicks on the submit button.
`method`	Specifies the HTTP method that the browser uses for the HTTP request. The default method is the GET method, but the POST method is also commonly used, especially when the request includes data that's saved on the server.

Common control attributes

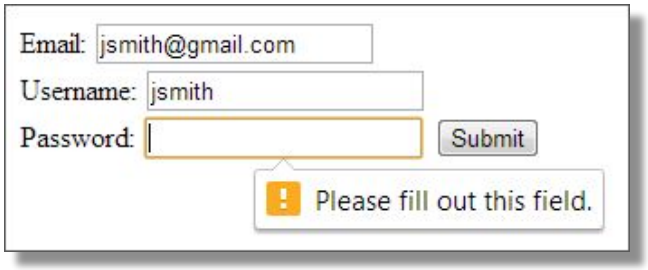
Attribute	Description
`name`	Specifies the name of the control. When writing Java code, use this attribute to refer to the control.
`value`	The default value of the control. This varies depending on the type of control. For a text box, this attribute sets the text that's displayed in the box. For a button, this attribute sets the text that's displayed on the button.

The code for four types of text controls

```
Email:    <input type="email" name="email" required><br>
Username: <input type="text" name="username" required><br>
Password: <input type="password" name="password" required>
          <input type="hidden" name="action" value="register">
```

The text controls displayed in a browser…

When a required field hasn't been entered…



Attributes of these text controls

Attribute	Description
`type`	Specifies the type of input control. Traditional values are "text", "password", and "hidden". HTML5 introduced some new types like "email" for a text box that should receive an email entry.
`name`	Specifies the name of the control. This is the name that is used to refer to the data in the control from a servlet or JSP.
`value`	Specifies the value of data in the control.
`maxlength`	Specifies the maximum number of characters that can be entered into the text box.
`required`	Specifies that the user must enter a value for a text box. If the user submits the form and the field is empty, the browser displays its default error message.

Text boxes

- Use the type attribute to specify the type of *text box*.
- A value of "text" creates a *standard text box*.
- A value of "email" creates a special type of text box that's designed for working with email addresses. This type of text box was introduced with HTML5.
- A value of "password" creates a *password box* that displays asterisks instead of text.
- A value of "hidden" creates a *hidden field* that stores a name and value that's sent to the server but isn't shown by the browser.

The code for three types of buttons

```
<input type="submit" value="Submit">
<input type="reset" value="Reset">
<input type="button" value="Validate">
```

The buttons displayed in a browser

The code for two submit buttons

```
<form action="quickOrder" method="get">
    <input type="submit" name="continue" value="Continue Shopping">
</form>
<form action="cart" method="get">
    <input type="submit" name="checkout" value="Checkout">
</form>
```

The buttons displayed in a browser

Buttons

- The type attribute identifies the type of *button* to be used.
- A type attribute of "submit" creates a *submit button* that activates the action attribute of the form when it's clicked.
- A type attribute of "reset" creates a *reset button* that resets all controls on the form to their default values when it's clicked.
- A type attribute of "button" creates a generic button that can be used to trigger JavaScript actions.

Four check boxes and three radio buttons

```
<p><input type="checkbox" name="addEmail" checked>
Yes, add me to your mailing list.</p>

<p>Contact me by:
<input type="radio" name="contactVia" value="Email">Email
<input type="radio" name="contactVia" value="Postal Mail">Postal
mail
<input type="radio" name="contactVia" value="Both" checked>Both</p>

<p>I'm interested in these types of music:<br>
<input type="checkbox" name="rock">Rock<br>
<input type="checkbox" name="country">Country<br>
<input type="checkbox" name="bluegrass">Bluegrass</p>
```

The check boxes and radio buttons in a browser

Attributes of these controls

Attribute	Description
`type`	Specifies the type of control. A value of "checkbox" creates a check box while a value of "radio" creates a radio button.
`checked`	Selects the control. When several radio buttons share the same name, only one radio button can be selected at a time.

Check boxes and radio buttons

- Use *check boxes* to allow the user to supply a true/false value.
- Use *radio buttons* to allow a user to select one option from a group of options. To create a group of radio buttons, use the same name for all of the radio buttons.
- If you don't group radio buttons, more than one can be on at the same time.

The code for a combo box

```
Select a country:<br>
<select name="country">
    <option value="USA" selected>United States</option>
    <option value="CAN">Canada</option>
    <option value="MEX">Mexico</option>
</select>
```

The combo box displayed in a browser

The code for a list box with 3 options displayed

```
<select name="country" size="3" multiple>
```

The list box displayed in a browser

Attributes of the select element

Attribute	Description
`size`	The number of items to display in the control. If the value is 1, the control will be a combo box. If more than 1, the control will be a list box.
`multiple`	If coded, the user can select more than one option. This is only valid with a list box.

Attributes of the option element

Attribute	Description
`selected`	Selects the option.

Combo and list boxes

- A *combo box* provides a drop-down list that lets the user select a single option.
- A *list box* provides a list of options and lets the user select more than one option if the multiple attribute is coded.
- To select more than one option from a list box, the user can hold down the Ctrl key on a Windows system or the Command key on a Mac and then click on the options.