Introducing Tables

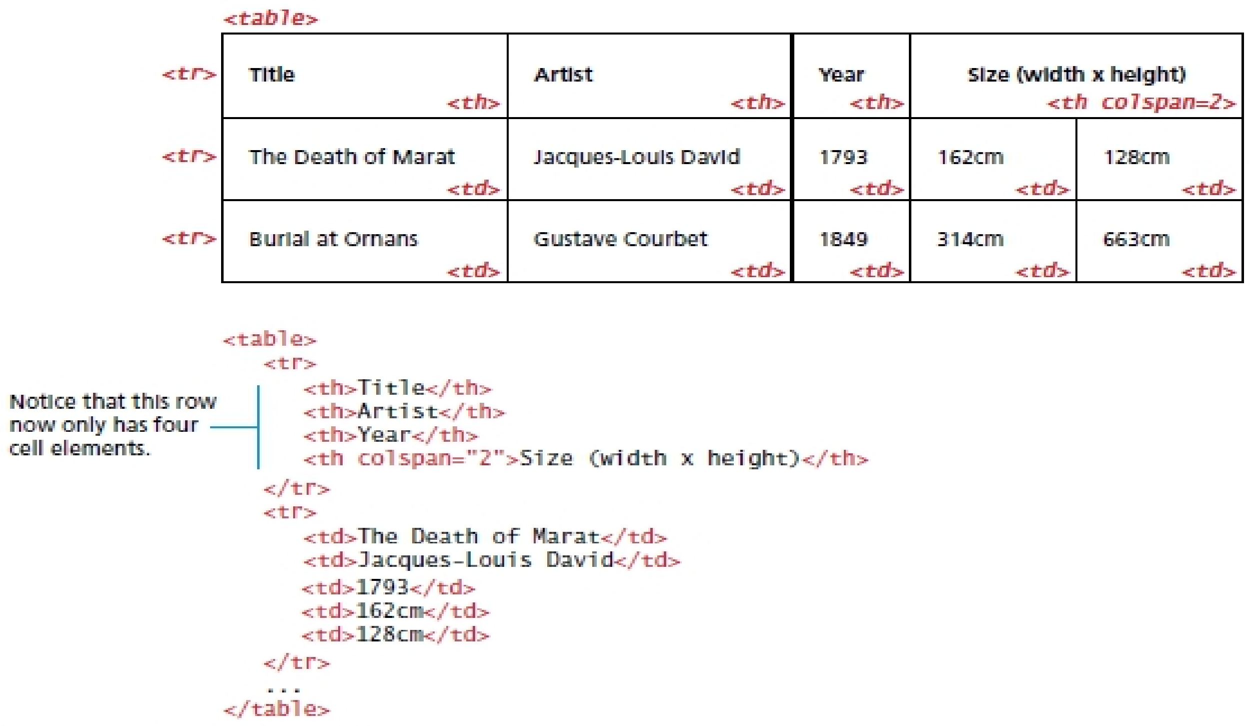
- A **table** in HTML is created using the <table> element and can be used to represent information that exists in a two-dimensional grid.
- Tables can be used to display calendars, financial data, pricing tables, and many other types of data.
- Just like a real-world table, an HTML table can contain any type of data: not just numbers, but text, images, forms, even other tables.

Basic Table Structure

- HTML **<table>** contains any number of rows **<tr>.**
- Each row contains any number of table data cells **<td>**.
- Many tables will contain some type of headings in the first row. In HTML, you indicate header data by using the **<th>** instead of the <td> element, as shown in Figure 4.3.
- Browsers tend to make the content within a <th> element bold, but you could style it anyway you would like via CSS.
- The main reason you should use the <th> element is not, however, due to presentation reasons. Rather, you should also use the <th> element for accessibility reasons and for search engine optimization reasons.
- Some browsers do not by default display borders for the table; however, we can do so via CSS.
- All content must appear within the <td> or <th> container.
- Each row must have the same number of <td> or <th> containers.

Spanning Rows and Columns

If you want a given cell to cover several columns or rows, then you can do so by using the **colspan** or **rowspan** attributes (Figure 4.4).

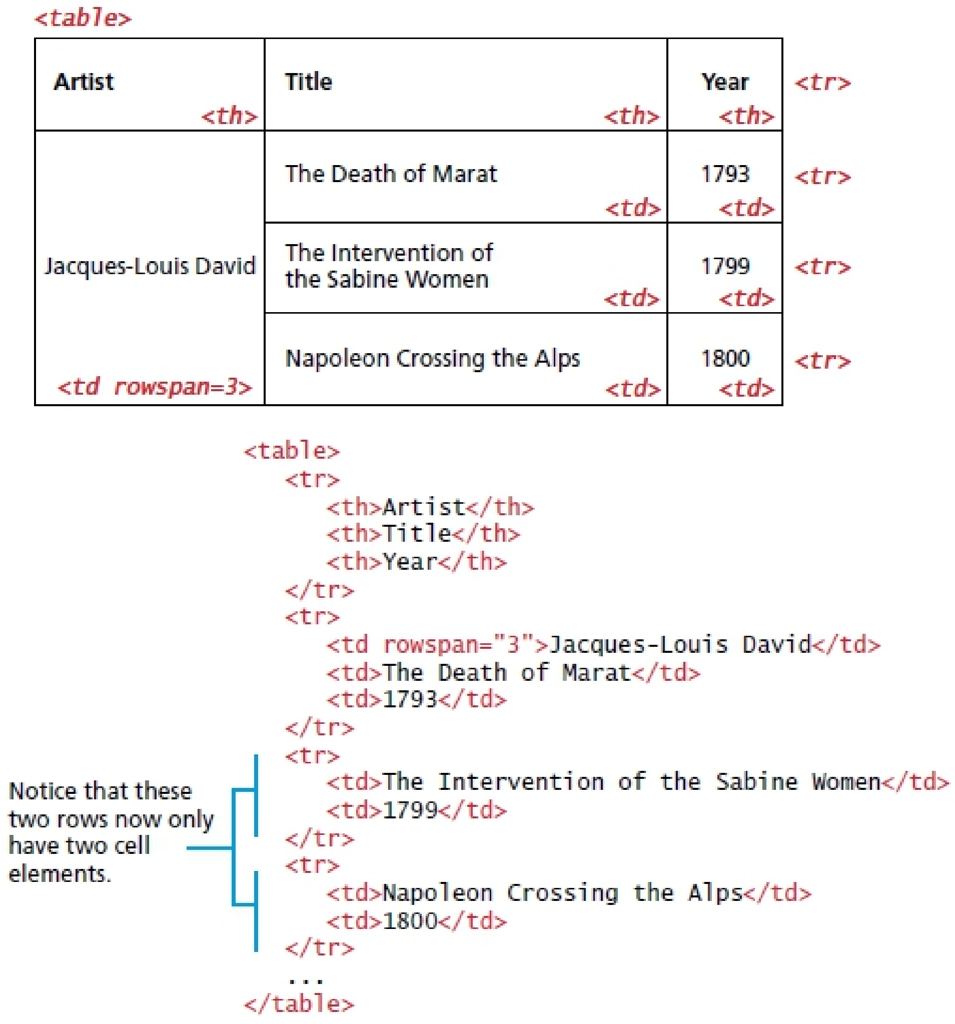


<table>

<tr>	Title <th>	Artist <th>	Year <th>	Size (width x height) <th colspan=2>	
<tr>	The Death of Marat <td>	Jacques-Louis David <td>	1793 <td>	162cm <td>	128cm <td>
<tr>	Burial at Ornans <td>	Gustave Courbet <td>	1849 <td>	314cm <td>	663cm <td>

```
<table>
   <tr>
      <th>Title</th>
      <th>Artist</th>
      <th>Year</th>
      <th colspan="2">Size (width x height)</th>
   </tr>
   <tr>
      <td>The Death of Marat</td>
      <td>Jacques-Louis David</td>
      <td>1793</td>
      <td>162cm</td>
      <td>128cm</td>
   </tr>
   ...
</table>
```

FIGURE 4.4 Spanning columns

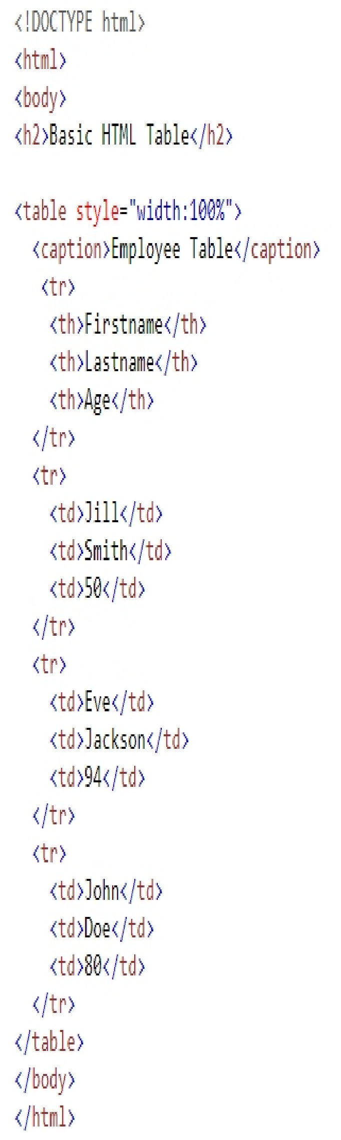

<table>
Artist <th>
Title <th>
Year <th>
<tr>
Jacques-Louis David <td rowspan=3>
The Death of Marat <td>
1793 <td>
<tr>
The Intervention of the Sabine Women <td>
1799 <td>
<tr>
Napoleon Crossing the Alps <td>
1800 <td>
<tr>
Notice that these two rows now only have two cell elements.

```
<table>
   <tr>
      <th>Artist</th>
      <th>Title</th>
      <th>Year</th>
   </tr>
   <tr>
      <td rowspan="3">Jacques-Louis David</td>
      <td>The Death of Marat</td>
      <td>1793</td>
   </tr>
   <tr>
      <td>The Intervention of the Sabine Women</td>
      <td>1799</td>
   </tr>
   <tr>
      <td>Napoleon Crossing the Alps</td>
      <td>1800</td>
   </tr>
   ...
</table>
```

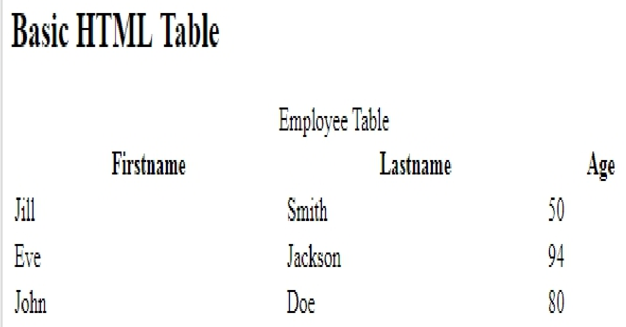
FIGURE 4.5 Spanning rows

Additional Table Elements

- The **<caption>** element is used to provide a brief title or description of the table, which improves the accessibility of the table, and is strongly recommended. You can use the caption-side CSS property to change the position of the caption below the table.
- The **<thead>, <tfoot>, and <tbody>** elements tend in practice to be used quite infrequently. However, they do make some sense for tables with a large number of rows. With CSS, one could set the height and overflow properties of the <tbody> element so that its content scrolls, while the header and footer of the table remain always on screen.
- The **<col> and <colgroup>** elements are also mainly used to aid in the eventual styling of the table. Rather than styling each column, you can style all columns within a <colgroup> with just a single style. Unfortunately, the only properties you can set via these two elements are borders, backgrounds, width, and visibility, and only if they are not overridden in a <td>, <th>, or <tr> element.
- As a consequence, they tend to not be used very often.



```
<!DOCTYPE html>
<html>
<body>
<h2>Basic HTML Table</h2>

<table style="width:100%">
  <caption>Employee Table</caption>
   <tr>
    <th>Firstname</th>
    <th>Lastname</th>
    <th>Age</th>
  </tr>
  <tr>
    <td>Jill</td>
    <td>Smith</td>
    <td>50</td>
  </tr>
  <tr>
    <td>Eve</td>
    <td>Jackson</td>
    <td>94</td>
  </tr>
  <tr>
    <td>John</td>
    <td>Doe</td>
    <td>80</td>
  </tr>
</table>
</body>
</html>
```



Basic HTML Table

Employee Table

Firstname	Lastname	Age
Jill	Smith	50
Eve	Jackson	94
John	Doe	80

Using Tables for Layout

Styling Tables

There are a number of table attributes that can be set:

- **width, height**—for setting the width and height of cells
- **cellspacing**—for adding space between every cell in the table
- **cellpadding**—for adding space between the content of the cell and its border
- **bgcolor**—for changing the background color of any table element
- **background**—for adding a background image to any table element
- **align**—for indicating the alignment of a table in relation to the surrounding container

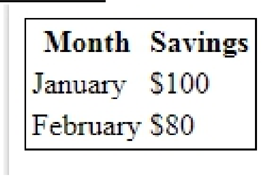
Table Borders

Borders can be assigned to both the <table>, <td> and <th> element. Interestingly, borders cannot be assigned to the <tr>, <thead>, <tfoot>, and <tbody> elements. Notice as well the border-collapse property, where each cell has its own unique borders.

Examples1: **<u>Table Border</u>**

```
<table style="border: 1px solid black">
            <tr>
               <th>Month</th>
               <th>Savings</th>
             </tr>
             <tr>
               <td style="border: 1px solid black">January</td>
               <td style="border: 1px solid black">$100</td>
            </tr>
            <tr>
              <td style="border: 1px solid black">February</td>
              <td style="border: 1px solid black">$80</td>
            </tr>
</table>
```

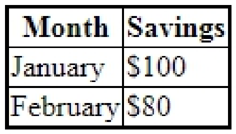
<u>OUTPUT</u>



Month	Savings
January	$100
February	$80

Examples2 : **<u>Table Border ,table header and column with border collapse</u>**

```
<table style="border: 1px solid black; border-collapse: collapse;" >
            <tr>
               <th style="border: 1px solid black">Month</th>
               <th style="border: 1px solid black">Savings</th>
             </tr>
             <tr>
               <td style="border: 1px solid black">January</td>
               <td style="border: 1px solid black">$100</td>
            </tr>
            <tr>
              <td style="border: 1px solid black">February</td>
              <td style="border: 1px solid black">$80</td>
            </tr>
</table>
```

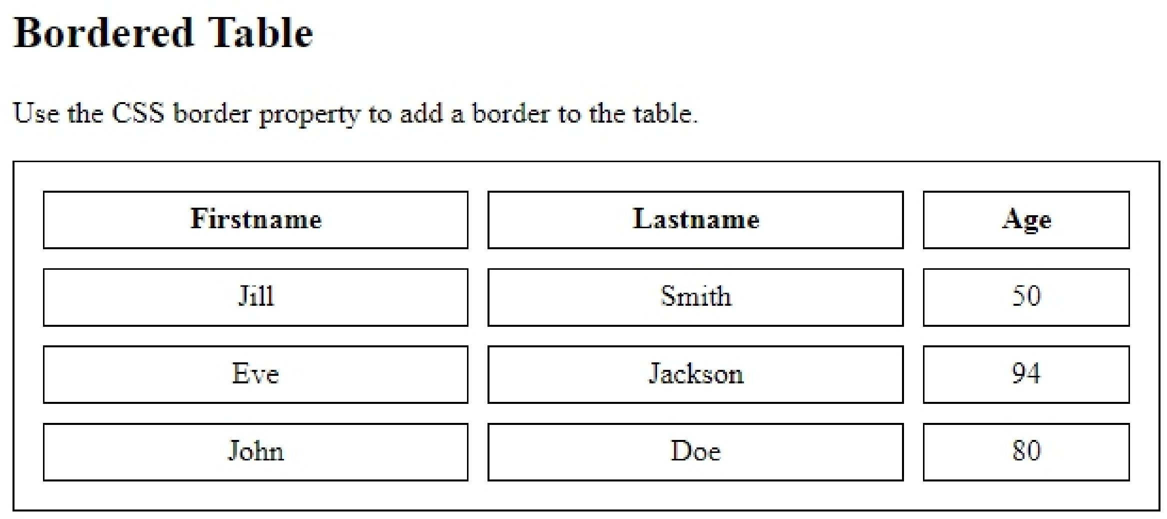


Month	Savings
January	$100
February	$80

<u>Examples 3 : Using <style> tag</u>

```
<!DOCTYPE html>
<html>
<head>
<style>
table,th,td {
        border: solid 1pt black;
        padding: 5px;
        border-spacing: 10px;
         text-align: center;

}
</style>
</head>
......
```

<u>OUTPUT</u>



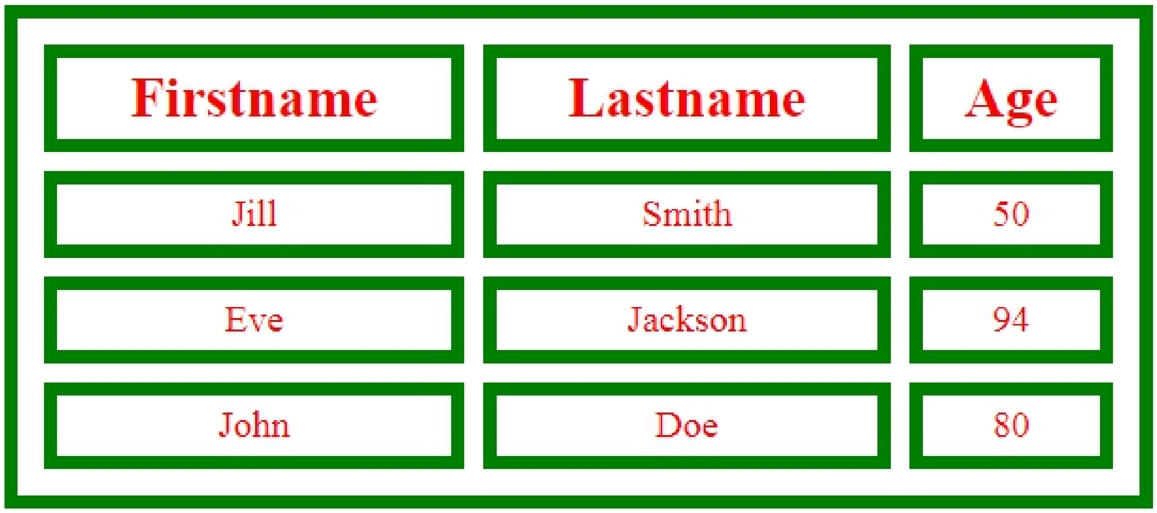
Bordered Table

Use the CSS border property to add a border to the table.

Firstname	Lastname	Age
Jill	Smith	50
Eve	Jackson	94
John	Doe	80

<u>Examples 4: Using <style> tag</u>

```
<style>
table,th,td {
        border: solid 5pt black;
        padding: 5px;
        border-spacing: 10px;
         text-align: center;
         color:red;
         border-color:green;
         font-size:30px
}
td{
font-size:20px;
}
</style>
```

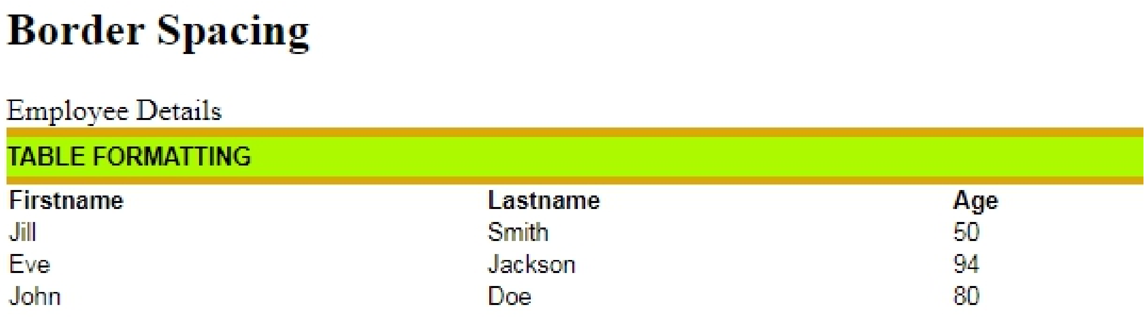


Firstname	Lastname	Age
Jill	Smith	50
Eve	Jackson	94
John	Doe	80

Boxes and Zebras

A box format is one in which background colors and borders in various ways are applied.

```
<style>
table {
        font-size: 0.8em;
        font-family: Arial, Helvetica, sans-serif;
        border-collapse: collapse;
        border-top: 4px solid #DCA806;
        border-bottom: 1px solid white;
        text-align: left;
}
caption {
        font-weight: bold;
        padding: 0.25em 0 0.25em 0;
        text-align: left;
        text-transform: uppercase;
        border-top: 5px solid #DCA806;
        background-color: #ACF900;
}
</style>
```



Border Spacing

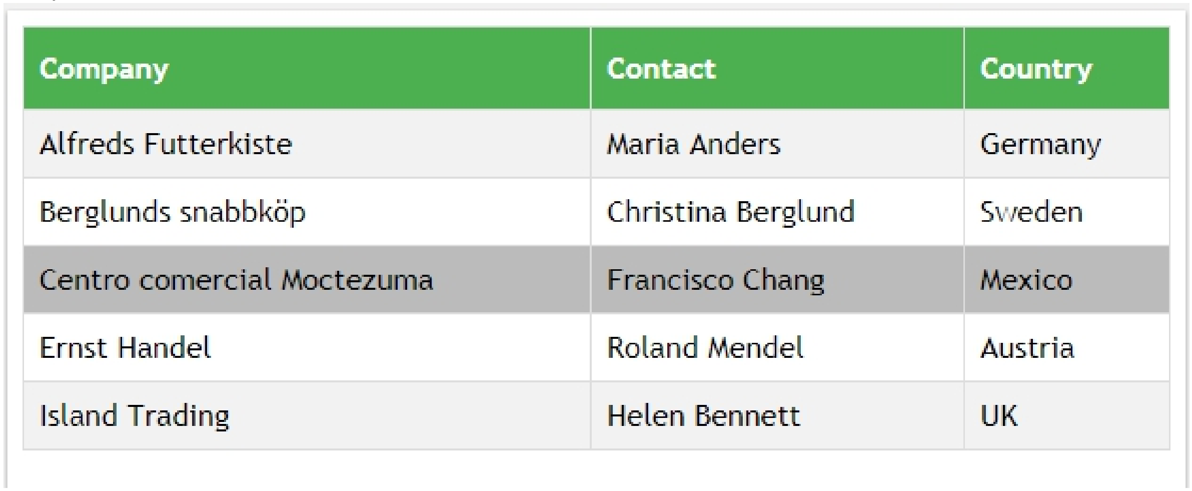
Employee Details

TABLE FORMATTING

Firstname	Lastname	Age
Jill	Smith	50
Eve	Jackson	94
John	Doe	80

We can then add special styling to the :hover pseudo-class of the <tr> element, to highlight a row when the mouse cursor hovers over a cell, as shown in Figure below.

```
<style>
#customers {
            font-family: "Trebuchet MS", Arial, Helvetica, sans-serif;
           border-collapse: collapse;
            width: 100%;
}
#customers td, #customers th {
             border: 1px solid #ddd;
             padding: 8px;
}
#customers tr:nth-child(even){background-color: #f2f2f2;}

#customers tr:hover {background-color: #bbb;}

#customers th {
            padding-top: 12px;
            padding-bottom: 12px;
            text-align: left;
             background-color: #4CAF50;
             color: white;
}
</style>
```



Company	Contact	Country
Alfreds Futterkiste	Maria Anders	Germany
Berglunds snabbköp	Christina Berglund	Sweden
Centro comercial Moctezuma	Francisco Chang	Mexico
Ernst Handel	Roland Mendel	Austria
Island Trading	Helen Bennett	UK

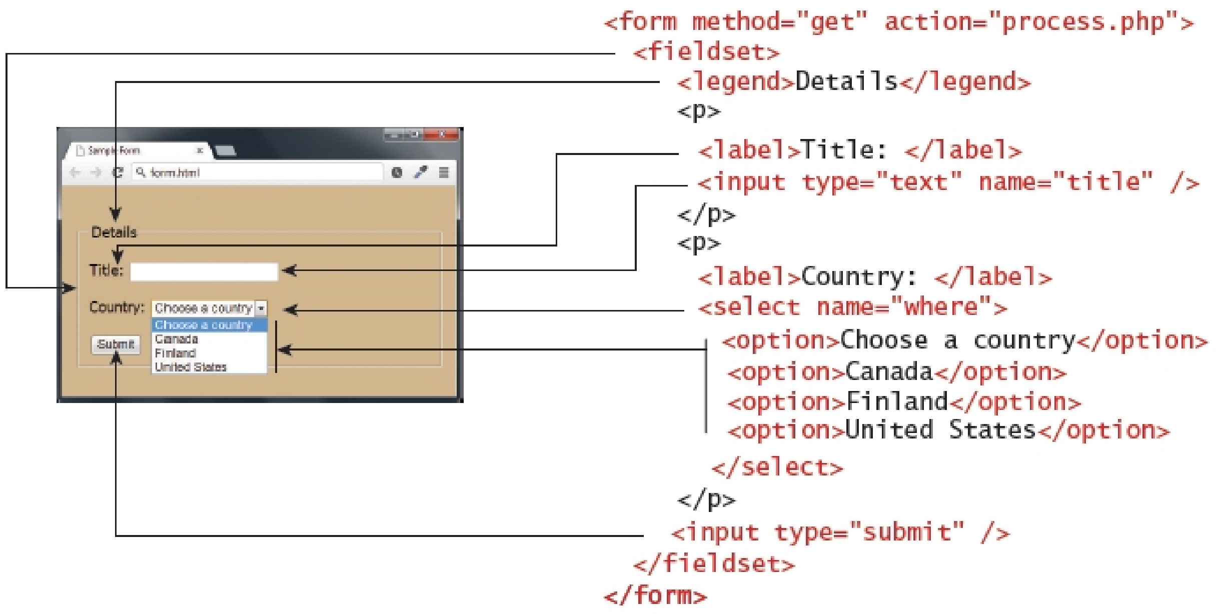
Introducing Forms

Forms provide the user with an alternative way to interact with a web server. Forms provide a much richer mechanism to interact with server. Using a form, the user can enter text, choose items from lists, and click buttons. While a form can contain most other HTML elements, a form cannot contain another <form> element.

Typically, programs running on the server will take the input from HTML forms and do something with it, such as save it in a database, interact with an external web service, or customize subsequent HTML based on that input.

Form Structure

A form is constructed in HTML using special **<form>** HTML elements, which is a container for other elements that represent the various input elements within the form as well as plain text and almost any other HTML element.



```
<form method="get" action="process.php">
  <fieldset>
    <legend>Details</legend>
    <p>
      <label>Title: </label>
      <input type="text" name="title" />
    </p>
    <p>
      <label>Country: </label>
      <select name="where">
        <option>Choose a country</option>
        <option>Canada</option>
        <option>Finland</option>
        <option>United States</option>
      </select>
    </p>
    <input type="submit" />
  </fieldset>
</form>
```

FIGURE 4.11 Sample HTML form

How Forms Work

- The process begins with a request for an HTML page that contains some type of form on it (ex: a user registration form or as simple as a search box).
- After the user fills out the form, the form data needs to be submitted back to the server. This is typically achieved via a submit button, but through JavaScript, it is possible to submit form data using some other type of mechanism.

- Because interaction between the browser and the web server is governed by the HTTP protocol, the form data must be sent to the server via a standard HTTP request. This request is typically some type of server-side program(PHP scripts) that will process the form data in some way; this could include checking it for validity, storing it in a database, or sending it in an email.

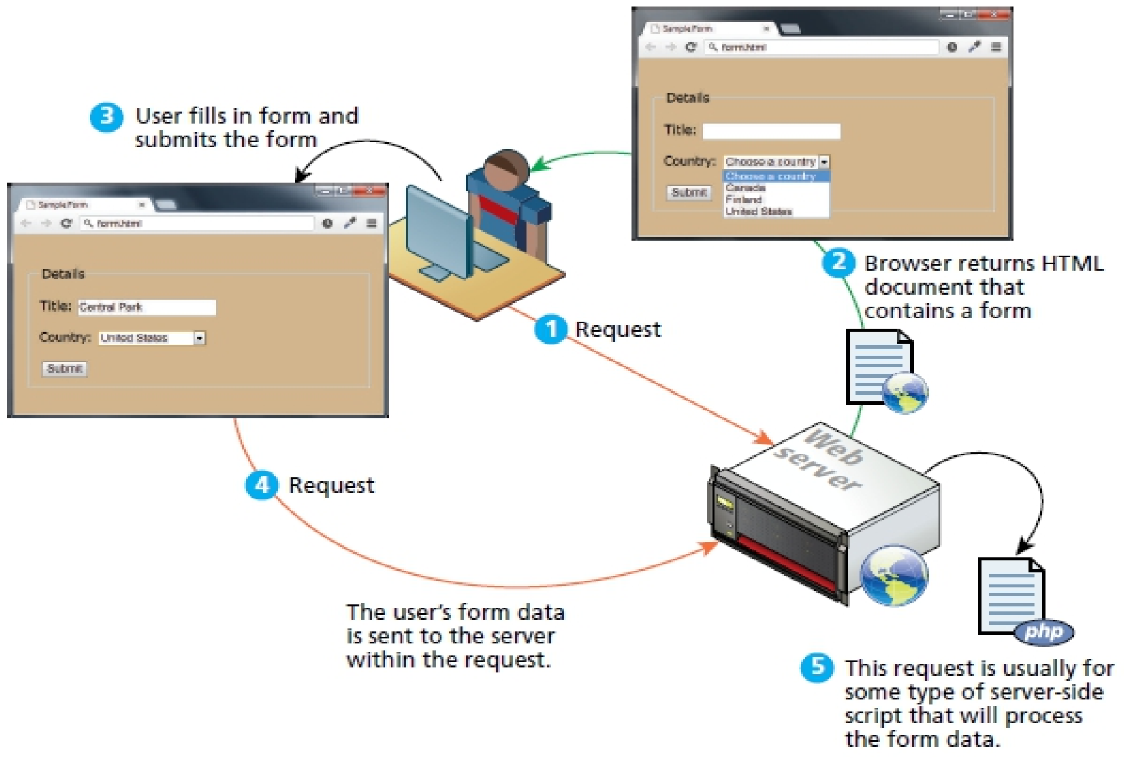
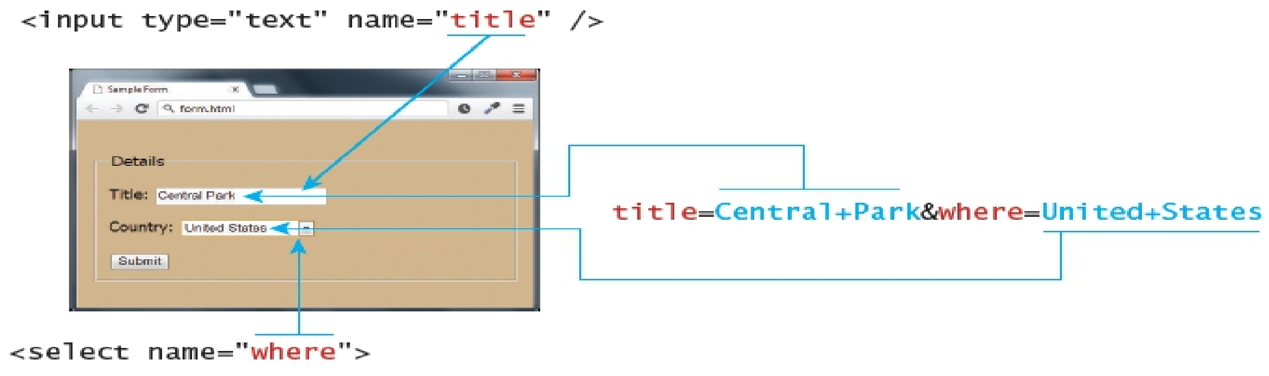

3 User fills in form and submits the form
Details
Title: Central Park
Country: United States
Submit
1 Request
2 Browser returns HTML document that contains a form
Choose a country
Canada
Finland
United States
4 Request
The user's form data is sent to the server within the request.
Web server
php
5 This request is usually for some type of server-side script that will process the form data.

FIGURE 4.12 How forms work

Query Strings

- The browser “sends” the data to the server via an HTTP request. The browser packages the user’s data input into something called a query string.
- A **query string** is a series of name=value pairs separated by ampersands (the **&** character).
- The names in the query string are defined by the HTML form ; each form element contains a name attribute, which is used to define the name for the form data in the query string. The values in the query string are the data entered by the user.



```
<input type="text" name="title" />
```

```
<select name="where">
```

```
title=Central+Park&where=United+States
```

FIGURE 4.13 Query string data and its connection to the form elements

- Query strings have certain rules defined by the HTTP protocol. Certain characters such as spaces, punctuation symbols, and foreign characters cannot be part of a query string. Instead, such special symbols must be **URL encoded** (also called **percent encoded**), as shown in Figure 4.14.

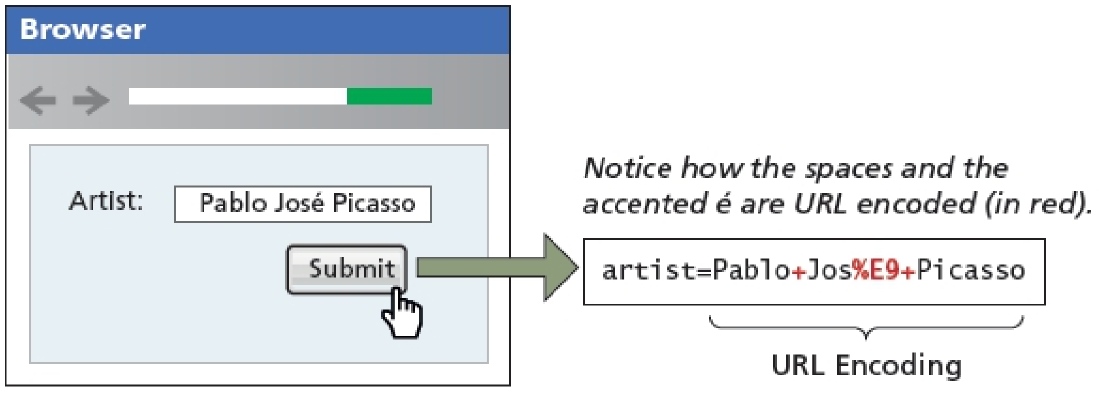

Browser
Artist:
Pablo José Picasso
Submit
Notice how the spaces and the accented é are URL encoded (in red).
artist=Pablo+Jos%E9+Picasso
URL Encoding

FIGURE 4.14 URL encoding

The <form> Element

- The HTML form contains two important attributes, namely the **action** and the **method** attributes.
- The action attribute specifies the URL of the server-side resource that will process the form data. This could be a resource on the same server as the form or a completely different server.
- The method attribute specifies how the query string data will be transmitted from the browser to the server. There are two possibilities**: GET and POST.**

GET Method

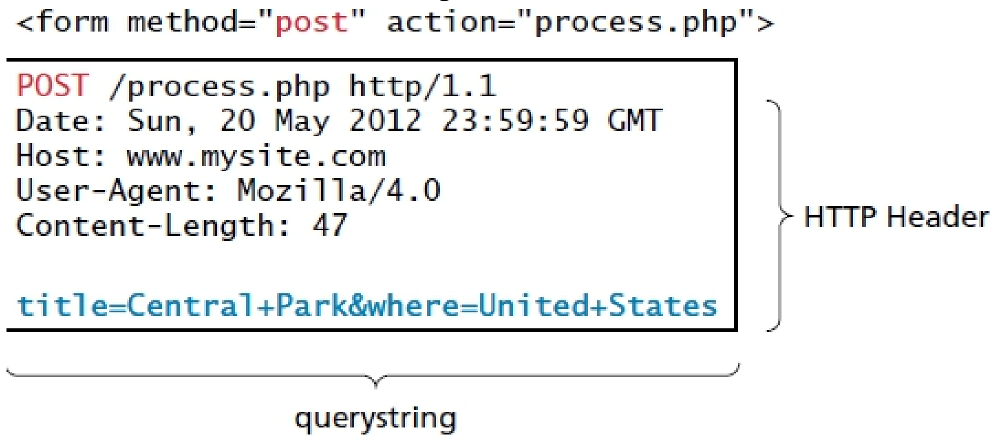
- With **GET**, the browser locates the data in the URL of the request.
- GET is one of the most common HTTP methods.
- GET is used to request data from a specified resource.
- The GET method is useful when you are testing or developing a system, since you can examine the query string directly in the browser's address bar.
- Form data will be saved when the user bookmarks a page, which may be desirable, but is generally a potential security risk for shared use computers.
- GET requests can be cached and remain in the browser history.
- GET requests should never be used when dealing with sensitive data.
- GET requests have length restrictions.
- GET requests is only used to request data (not modify).

  EX:

  /test/demo_form.php?name1=value1&name2=value2

POST Method

- With **POST**, the form data is located in the HTTP header after the HTTP variables.
- Generally, form data is sent using the POST method. Any time passwords are being transmitted, they should be transmitted via the POST method.
- POST is used to send data to a server to create/update a resource.
- POST requests are never cached.
- POST requests do not remain in the browser history.
- POST requests cannot be bookmarked.
- POST requests have no restrictions on data length.



```
<form method="post" action="process.php">
```

```
POST /process.php http/1.1
Date: Sun, 20 May 2012 23:59:59 GMT
Host: www.mysite.com
User-Agent: Mozilla/4.0
Content-Length: 47

title=Central+Park&where=United+States
```

	GET	POST
BACK button/Reload	Harmless	Data will be re-submitted (the browser should alert the user that the data are about to be re-submitted)
Bookmarked	Can be bookmarked	Cannot be bookmarked
Cached	Can be cached	Not cached
Encoding type	application/x-www-form-urlencoded	application/x-www-form-urlencoded or multipart/form-data. Use multipart encoding for binary data
History	Parameters remain in browser history	Parameters are not saved in browser history
Restrictions on data length	Yes, when sending data, the GET method adds the data to the URL; and the length of a URL is limited (maximum URL length is 2048 characters)	No restrictions
Restrictions on data type	Only ASCII characters allowed	No restrictions. Binary data is also allowed
Security	GET is less secure compared to POST because data sent is part of the URL	POST is a little safer than GET because the parameters are not stored in browser history or in web server logs
	Never use GET when sending passwords or other sensitive information!	
Visibility	Data is visible to everyone in the URL	Data is not displayed in the URL

Form Control Elements

Type	Description
`<button>`	Defines a clickable button.
`<datalist>`	An HTML5 element that defines lists of pre-defined values to use with input fields.
`<fieldset>`	Groups related elements in a form together.
`<form>`	Defines the form container.
`<input>`	Defines an input field. HTML5 defines over 20 different types of input.
`<label>`	Defines a label for a form input element.
`<legend>`	Defines the label for a fieldset group.
`<option>`	Defines an option in a multi-item list.
`<optgroup>`	Defines a group of related options in a multi-item list.
`<select>`	Defines a multi-item list.
`<textarea>`	Defines a multiline text entry box.

TABLE 4.2 Form-Related HTML Elements

Text Input Controls

Most forms need to gather text information from the user. Whether it is a search box, or a login form, or a user registration form, some type of text input is usually necessary. Table 4.3 lists the different text input controls.

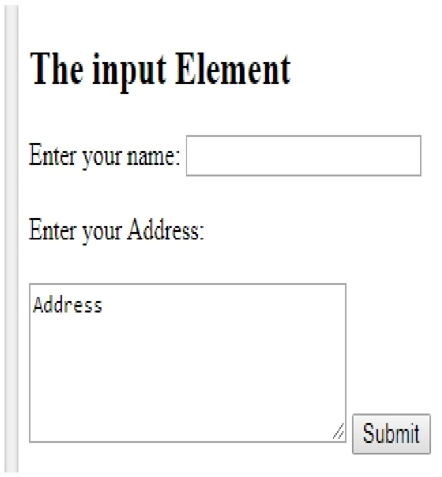
The <input> element: The <input> element can be displayed in several ways, depending on the type attribute.

Example: `<input name="firstname" type="text">`

The <textarea> Element :The <textarea> element defines a multi-line input field (a text area)

Example:`<textarea name="message" rows="10" cols="30">Textarea: multiline </textarea>`

```
<form action="/action_page.php">
  Enter your name:
  <input name="firstname" type="text">
  <br><br>
  Enter your Address:
  <br><br>
   <textarea name="Address" rows="5" cols="30">Address</textarea>
  <input type="submit">
</form>
```



The input Element

Enter your name:

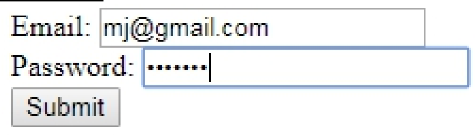
Enter your Address:

Address

The password Element : Creates a single-line text entry box for a password.

```
<form action="/action_page.php">
       Email: <input type="text" name="email"><br>
       Password: <input type="password" name="pwd" maxlength="8"><br>
      <input type="submit">
</form>
```

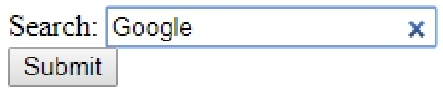
OUTPUT



Search type input: Creates a single-line text entry box suitable for a search string.

```
            Search: <input type="search" name="search"><br>
            <input type="submit" >
```

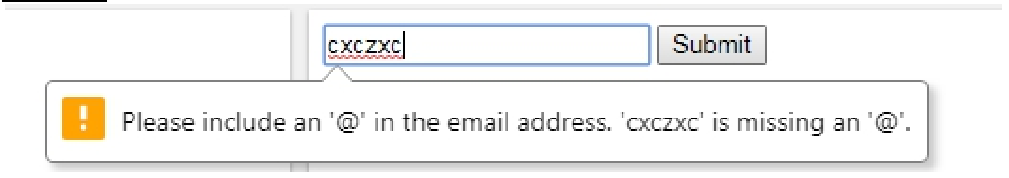
OUTPUT



Email type input: Creates a single-line text entry box suitable for entering an email address.

```
      <input type="email" ... />
```

OUTPUT



Please include an '@' in the email address. 'cxczxc' is missing an '@'.

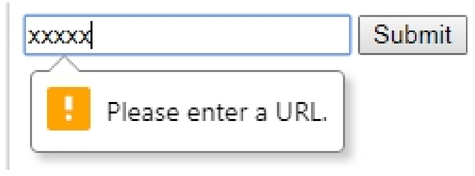
tel type input: Creates a single-line text entry box suitable for entering a telephone.

```
            <input type="tel" ... />
```

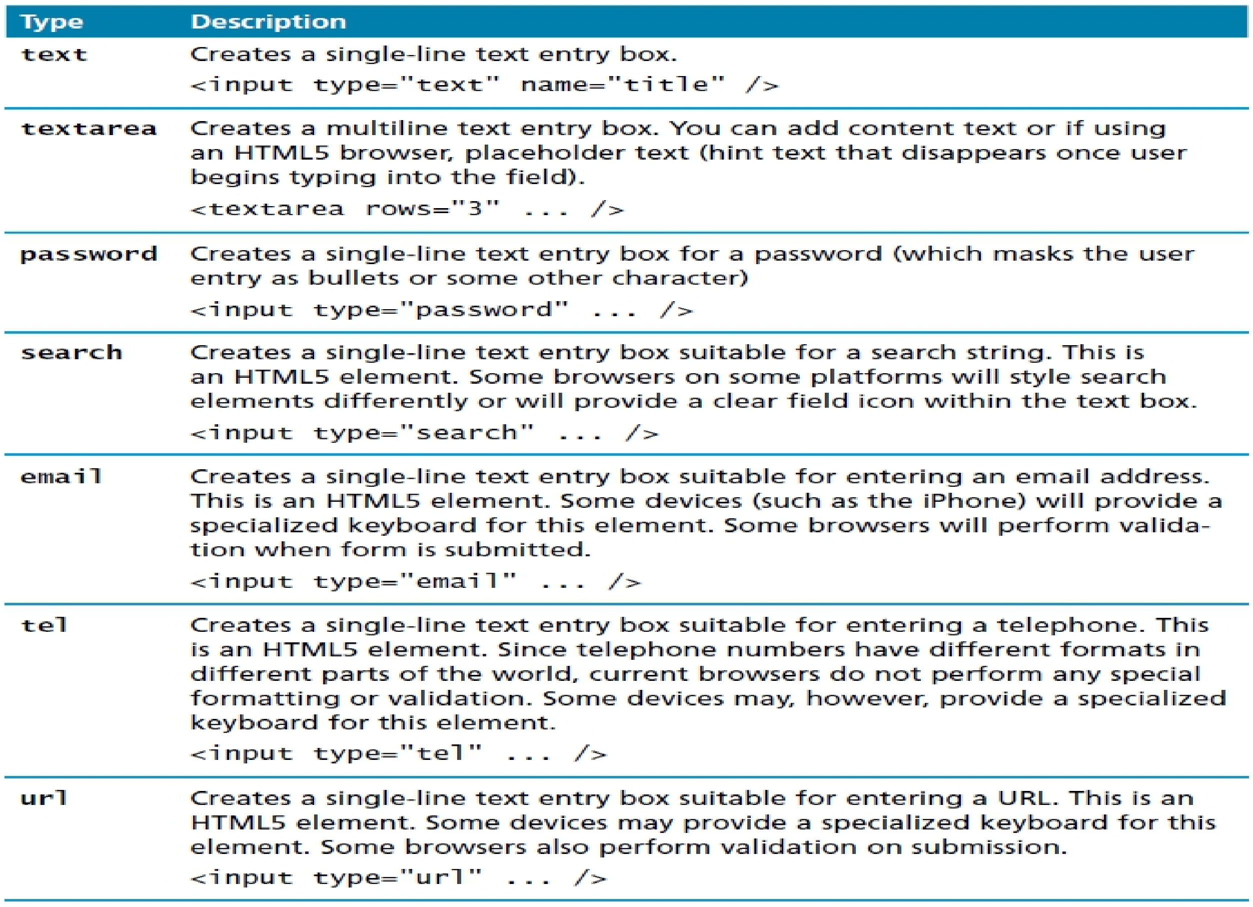
Url type input: Creates a single-line text entry box suitable for entering a URL.

```
      <input type="url" ... />
```

OUTPUT



Please enter a URL.



Type	Description
`text`	Creates a single-line text entry box. `<input type="text" name="title" />`
`textarea`	Creates a multiline text entry box. You can add content text or if using an HTML5 browser, placeholder text (hint text that disappears once user begins typing into the field). `<textarea rows="3" ... />`
`password`	Creates a single-line text entry box for a password (which masks the user entry as bullets or some other character) `<input type="password" ... />`
`search`	Creates a single-line text entry box suitable for a search string. This is an HTML5 element. Some browsers on some platforms will style search elements differently or will provide a clear field icon within the text box. `<input type="search" ... />`
`email`	Creates a single-line text entry box suitable for entering an email address. This is an HTML5 element. Some devices (such as the iPhone) will provide a specialized keyboard for this element. Some browsers will perform validation when form is submitted. `<input type="email" ... />`
`tel`	Creates a single-line text entry box suitable for entering a telephone. This is an HTML5 element. Since telephone numbers have different formats in different parts of the world, current browsers do not perform any special formatting or validation. Some devices may, however, provide a specialized keyboard for this element. `<input type="tel" ... />`
`url`	Creates a single-line text entry box suitable for entering a URL. This is an HTML5 element. Some devices may provide a specialized keyboard for this element. Some browsers also perform validation on submission. `<input type="url" ... />`

TABLE 4.3 Text Input Controls

Choice Controls

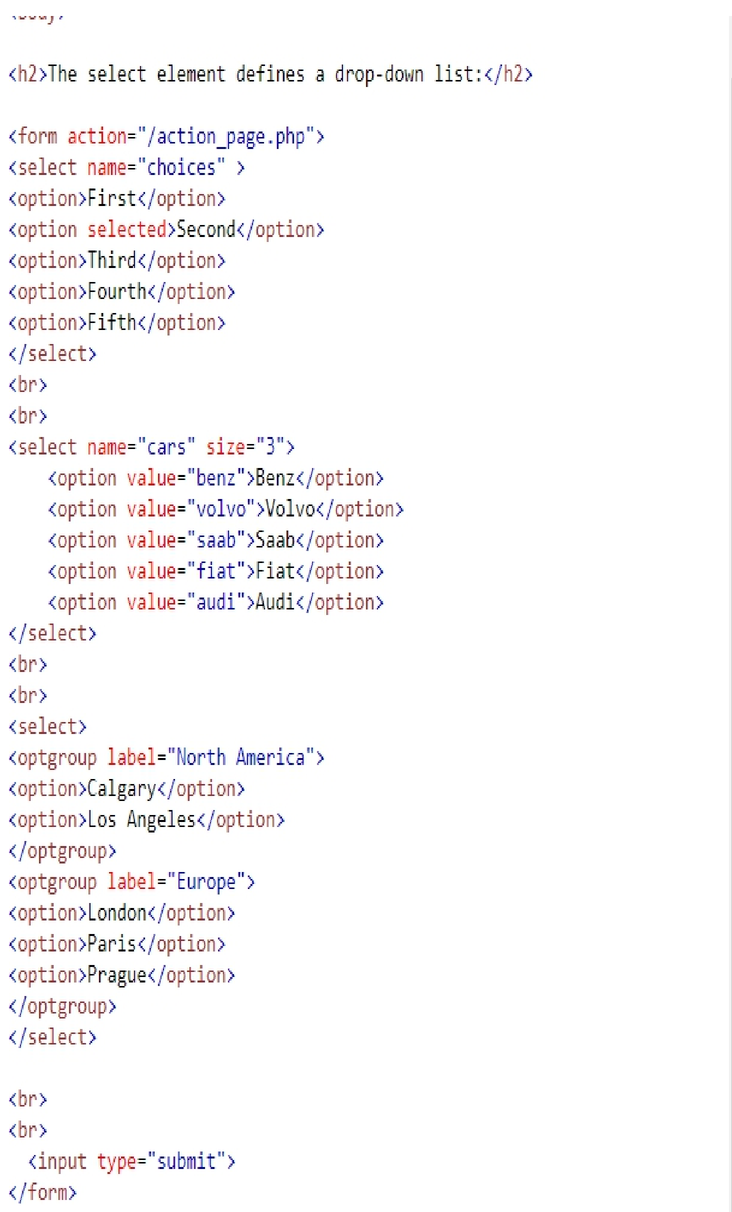
Forms often need the user to select an option from a group of choices. HTML provides several ways to do this.

Select Lists

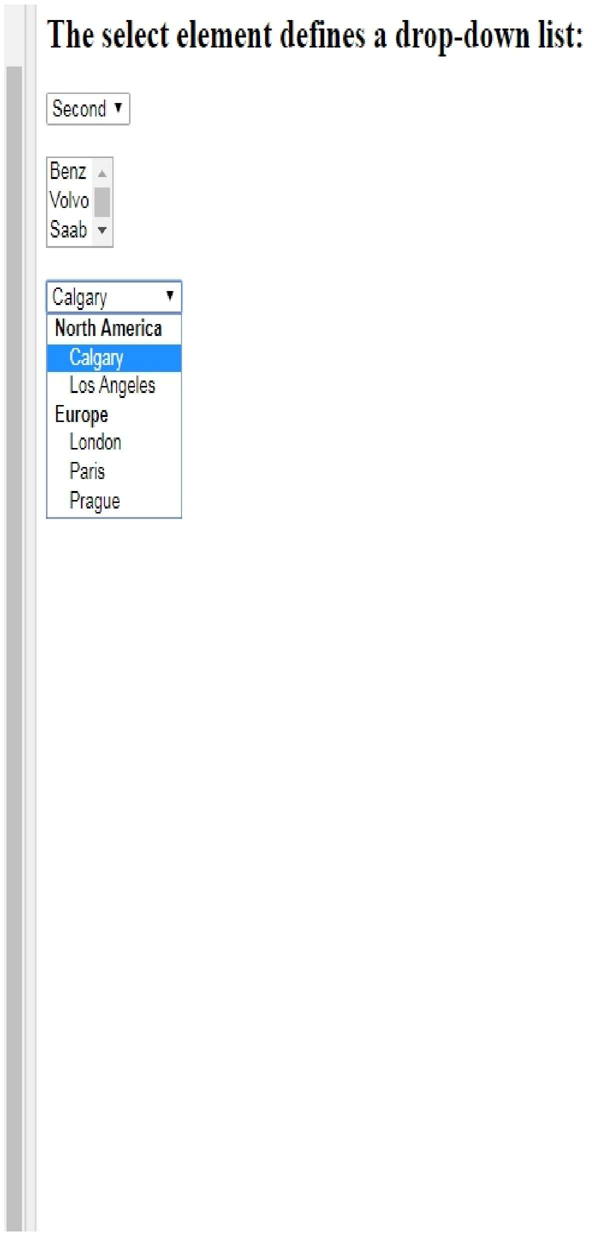
- The **<select> element** is used to create a multiline box for selecting one or more items.
- The options (defined using the **<option> element**) can be hidden in a dropdown list or multiple rows of the list can be visible by specifying size attribute.
- Option items can be grouped together via the **<optgroup> element**.
- The selected attribute in the <option> makes it a default value.
- The **value attribute** of the <option> element is used to specify what value will be sent back to the server in the query string when that option is selected. The value attribute is optional; if it is not specified, then the text within the container is sent instead.

Ex:

```
<select name="choices" >
        <option>First</option>
        <option>Second</option>
        <option>Third</option>
</select>
```



```
<h2>The select element defines a drop-down list:</h2>

<form action="/action_page.php">
<select name="choices" >
<option>First</option>
<option selected>Second</option>
<option>Third</option>
<option>Fourth</option>
<option>Fifth</option>
</select>
<br>
<br>
<select name="cars" size="3">
    <option value="benz">Benz</option>
    <option value="volvo">Volvo</option>
    <option value="saab">Saab</option>
    <option value="fiat">Fiat</option>
    <option value="audi">Audi</option>
</select>
<br>
<br>
<select>
<optgroup label="North America">
<option>Calgary</option>
<option>Los Angeles</option>
</optgroup>
<optgroup label="Europe">
<option>London</option>
<option>Paris</option>
<option>Prague</option>
</optgroup>
</select>

<br>
<br>
  <input type="submit">
</form>
```

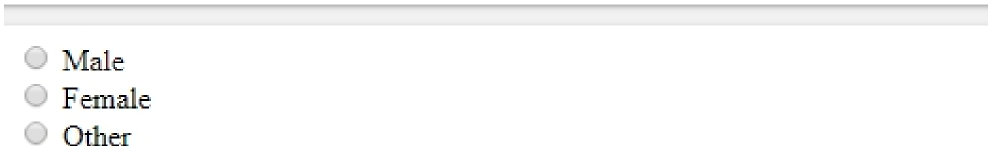


The select element defines a drop-down list:

Radio Buttons

- Radio buttons are useful when you want the user to select a single item from a small list of choices and you want all the choices to be visible.
- Radio buttons are added via the **<input type="radio">** element.
- The buttons are made mutually exclusive (i.e., only one can be chosen) by sharing the same name attribute.
- The checked attribute is used to indicate the default choice, while the value attribute works in the same manner as with the <option> element.

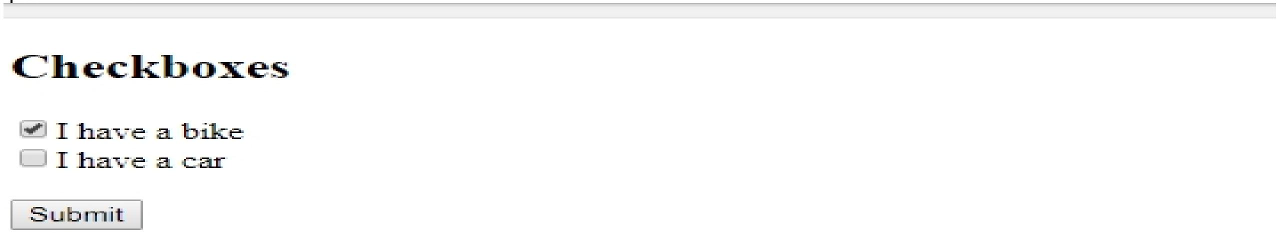
```
<!DOCTYPE html>
<html>
<body>

<form action="">
  <input type="radio" name="gender" value="male"> Male<br>
  <input type="radio" name="gender" value="female"> Female<br>
  <input type="radio" name="gender" value="other"> Other
</form>

</body>
</html>
```



Checkboxes

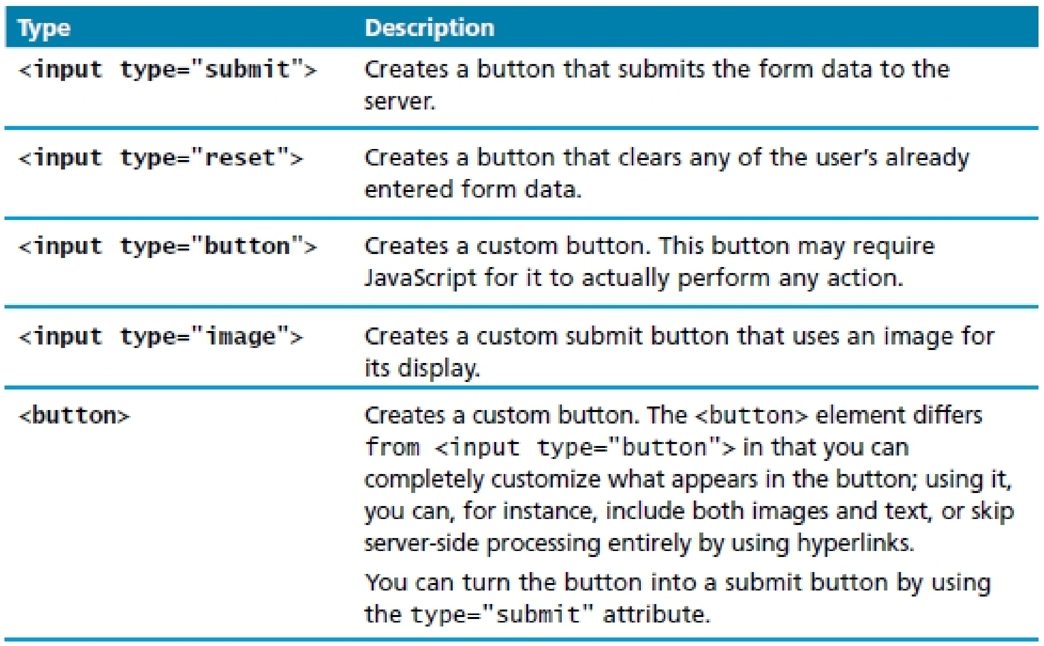
- Checkboxes are used for getting yes/no or on/off responses from the user.
- Checkboxes are added via the <input type="checkbox"> element.
- Also checkboxes can be grouped together by having them share the same name attribute.
- Each checked checkbox will have its value sent to the server.
- Like with radio buttons, the checked attribute can be used to set the default value of a checkbox.

```
<h2>Checkboxes</h2>
<form action="/action_page.php">
<input type="checkbox" name="vehicle1" value="Bike">I have a bike
<br>
<input type="checkbox" name="vehicle2" value="Car">I have a car
<br><br>
<input type="submit">
</form>
```



Button Controls

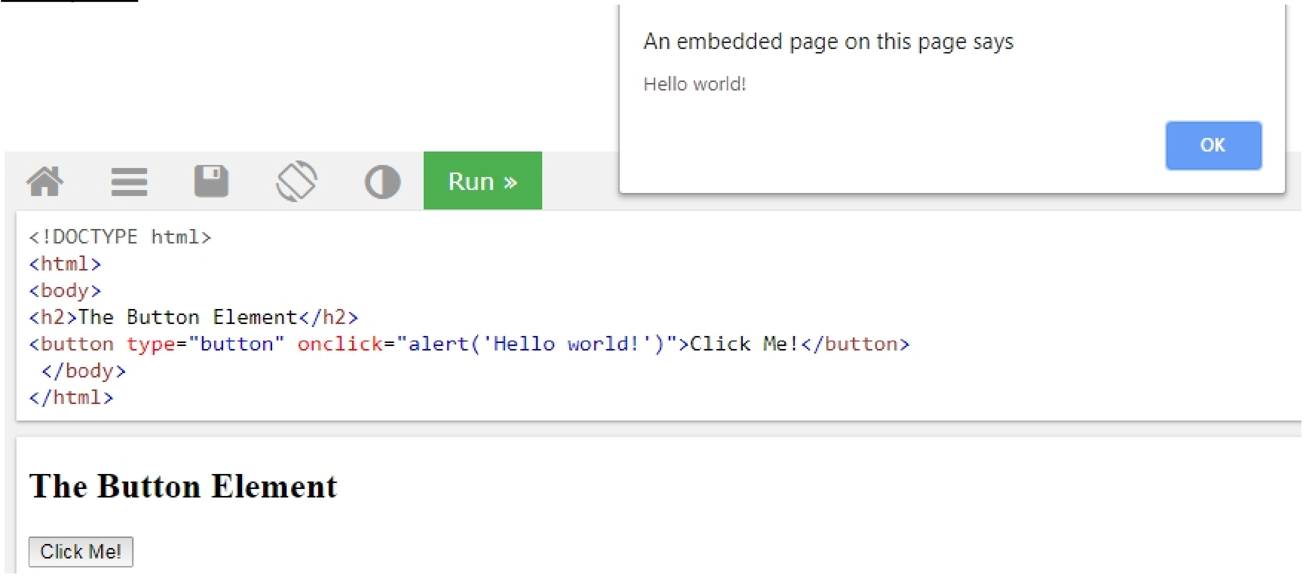
- HTML defines several different types of buttons.
- The <button> element defines a clickable button.
- Inside a <button> element you can put content, like text or images. This is the difference between this element and buttons created with the <input> element.



Type	Description
`<input type="submit">`	Creates a button that submits the form data to the server.
`<input type="reset">`	Creates a button that clears any of the user's already entered form data.
`<input type="button">`	Creates a custom button. This button may require JavaScript for it to actually perform any action.
`<input type="image">`	Creates a custom submit button that uses an image for its display.
`<button>`	Creates a custom button. The `<button>` element differs from `<input type="button">` in that you can completely customize what appears in the button; using it, you can, for instance, include both images and text, or skip server-side processing entirely by using hyperlinks. You can turn the button into a submit button by using the `type="submit"` attribute.

TABLE 4.4 Button Elements

Example 1:



An embedded page on this page says

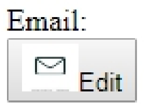
Hello world!

```
<!DOCTYPE html>
<html>
<body>
<h2>The Button Element</h2>
<button type="button" onclick="alert('Hello world!')">Click Me!</button>
 </body>
</html>
```

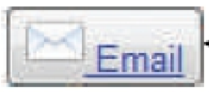
The Button Element

Example 2: Include image in button

```
<button type="submit" >
<img src="download.jpg"  alt="Email" width="30" height="25"/>Edit
</button>
```



Example 3: Include link in button

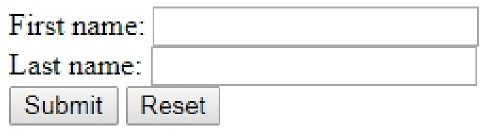


```
<button>
   <a href="email.html">
      <img src="images/email.png" alt=""/>
      Email
   </a>
</button>
```

Example 4:
<input type="submit"> Creates a button that submits the form data to the server.
<input type="reset"> Creates a button that clears any of the user's already entered form data.

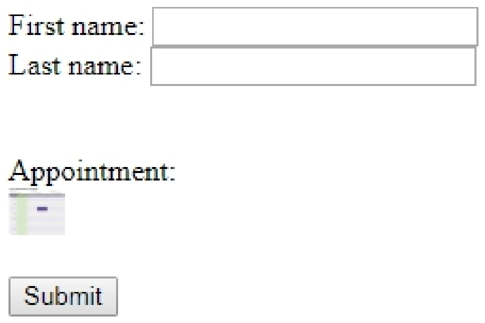
```
<form action="/action_page.php" method="get">
First name: <input type="text" name="fname"><br>
Last name: <input type="text" name="lname"><br>
<button type="submit" value="Submit">Submit</button>
<button type="reset" value="Reset">Reset</button>
</form>
```

Output



Example 5:

<input type="image"> Creates a custom submit button that uses an image for its display.
<input type="image" src="appointment.png" />



Specialized Controls

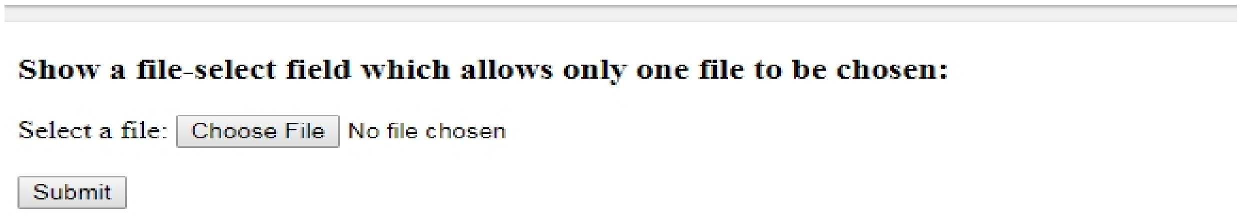
There are two important additional special-purpose form controls that are available in all browsers.

The first of these is the <input type="hidden"> element, defines a hidden input field. A hidden field let web developers include data that cannot be seen or modified by users when a form is submitted.

```
<input type="hidden" id="custId" name="custId" value="3487">
```

The other specialized form control is the <input type="file"> element, which is used to upload a file from the client to the server.

```
<h3>Show a file-select field which allows only one file to be chosen:</h3>
<form action="/action_page.php">
  Select a file: <input type="file" name="myFile"><br><br>
  <input type="submit">
</form>
```



Show a file-select field which allows only one file to be chosen:

Select a file: Choose File No file chosen

Submit

Number and Range

HTML5 introduced two new controls for the input of numeric values. When input via a standard text control, numbers typically require validation to ensure that the user has entered an actual number and, because the range of numbers is infinite, the entered number has to be checked to ensure it is not too small or too large. The number and range controls provide a way to input numeric values that eliminate the need for client-side numeric.

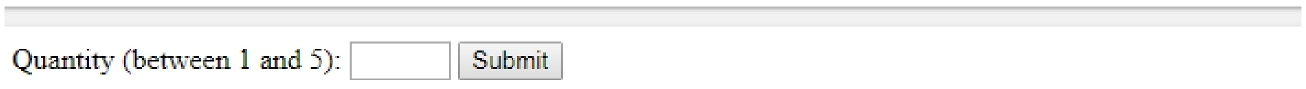
HTML <input type="number">

The <input type="number"> defines a field for entering a number.

Use the following attributes to specify restrictions:

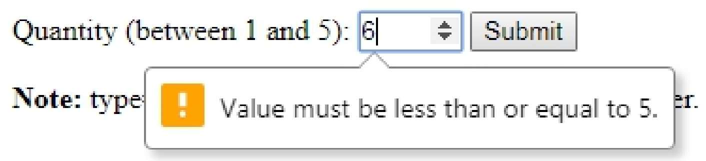
- max - specifies the maximum value allowed
- min - specifies the minimum value allowed
- step - specifies the legal number intervals
- value - Specifies the default value

```
<form action="/action_page.php">
  Quantity (between 1 and 5): <input type="number" name="quantity" min="1" max="5">
  <input type="submit">
</form>
```



Quantity (between 1 and 5): Submit

Output 2


Quantity (between 1 and 5): 6 Submit

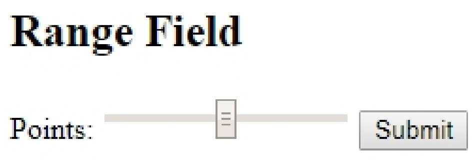
Note: type

Value must be less than or equal to 5.

The <input type="range"> defines a control for entering a number whose exact value is not important (like a slider control). Default range is 0 to 100. However, you can set restrictions on what numbers are accepted with the min, max, and step attributes:

```
<form action="/action_page.php" method="get">
  Points:
 <input type="range" name="points" min="0" max="10" step="2">
  <input type="submit">
</form>
```

Output

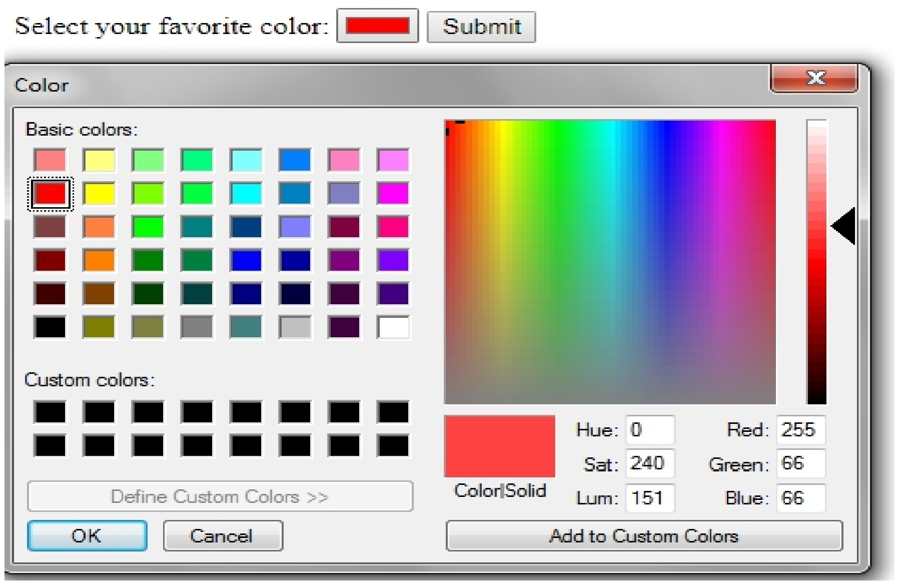

Range Field

Points: Submit

Color

Not every web page needs the ability to get color data from the user, but when it is necessary, the HTML5 color control provides a convenient interface for the user. The **<input type="color">** is used for input fields that should contain color. Depending on browser support, a color picker can show up in input field.

```
<form action="/action_page.php">
  Select your favorite color:
  <input type="color" name="favcolor" value="#ff0000">
  <input type="submit">
</form>
```


Select your favorite color: Submit

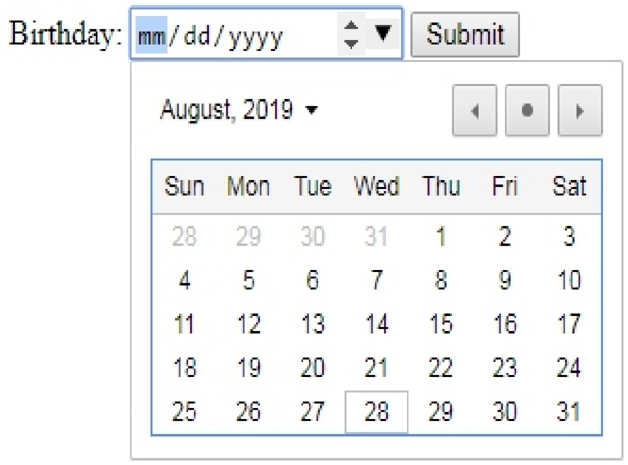
Hue: 0
Sat: 240
Lum: 151
Red: 255
Green: 66
Blue: 66

Date and Time Controls

Asking the user to enter a date or time is a relatively common web development task. Like with numbers, dates and times often need validation when gathering this information from a regular text input control. From a user's perspective, entering dates can be tricky as well: you probably have wondered at some point in time when entering a date into a web form, what format to enter it in, whether the day comes before the month, whether the month should be entered as an abbreviation or a number, and so on. The new date and time controls in HTML try to make it easier for users to input these tricky date and time values.

Example:

```
<form action="/action_page.php">
  Birthday:
  <input type="date" name="bday">
  <input type="submit">
</form>
```

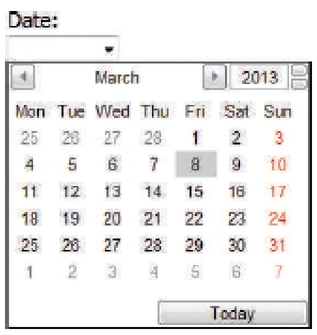


Sun	Mon	Tue	Wed	Thu	Fri	Sat
28	29	30	31	1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

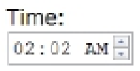
Table 4.5 lists the various HTML5 date and time controls.

Type	Description
`date`	Creates a general date input control. The format for the date is "yyyy-mm-dd."
`time`	Creates a time input control. The format for the time is "HH:MM:SS," for hours:minutes:seconds.
`datetime`	Creates a control in which the user can enter a date and time.
`datetime-local`	Creates a control in which the user can enter a date and time without specifying a time zone.
`month`	Creates a control in which the user can enter a month in a year. The format is "yyyy-mm."
`week`	Creates a control in which the user can specify a week in a year. The format is "yyyy-W##."

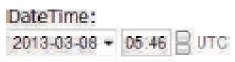
TABLE 4.5 HTML5 Date and Time Controls



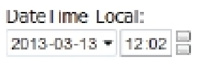
```
<label>Date: <br/>
<input type="date" ... />
```



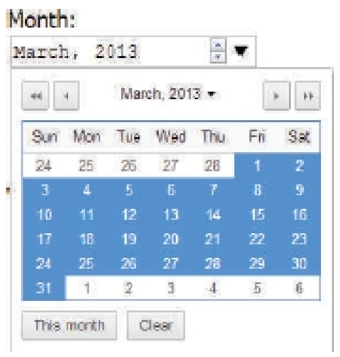
```
<input type="time" ... />
```



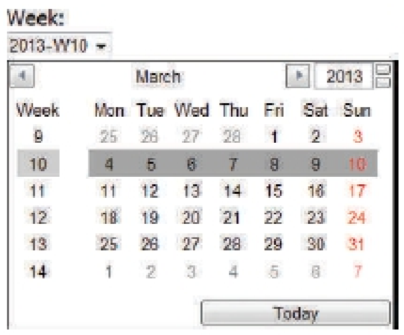
```
<input type="datetime" ... />
```



```
<input type="datetime-local" ... />
```



```
<input type="month" ... />
```



```
<input type="week" ... />
```

FIGURE 4.27 Date and time controls

Table and Form Accessibility

The term **web accessibility** refers to the assistive technologies, various features of HTML that work with those technologies, and different coding and design practices that can make a site more usable for people with visual, mobility, auditory, and cognitive disabilities.

In order to improve the accessibility of websites, the W3C created the **Web Accessibility Initiative (WAI)** in 1997. The WAI produces guidelines and recommendations, as well as organizing different working groups on different accessibility issues. One of its most helpful documents is the Web Content Accessibility Guidelines, which is available at **http://www.w3.org/WAI/intro/wcag.php**.

Perhaps the most important guidelines in that document are:

- *Provide text alternatives for any nontext content so that it can be changed into other forms people need, such as large print, braille, speech, symbols, or simpler language*.
- *Create content that can be presented in different ways (for example simpler layout) without losing information or structure*.
- *Make all functionality available from a keyboard*.
- *Provide ways to help users navigate, find content, and determine where they are*.

Accessible Tables
HTML tables can be quite frustrating from an accessibility standpoint. Users who rely on visual readers can find pages with many tables especially difficult to use. One vital way to improve the situation is to only use tables for tabular data, not for layout. Using the following accessibility features for tables in HTML can also improve the experience for those users:

- Describe the table's content using the <caption> element. This provides the user with the ability to discover what the table is about before having to listen to the content of each and every cell in the table. If you have an especially long description for the table, consider putting the table within a <figure> element and use the <figcaption> element to provide the description.
- Connect the cells with a textual description in the header. While it is easy for a sighted user to quickly see what row or column a given data cell is in, for users relying on visual readers, this is not an easy task. It is quite revealing to listen to reader software recite the contents of a table that has not made these connections. It sounds like this: "row 3, cell 4: 45.56; row 3, cell 5: Canada; row 3, cell 6: 25,000;etc." However, if these connections have been made, it sounds instead like this: "row 3, Average: 45.56; row 3, Country: Canada; row 3, City Count: 25,000; etc.," which is a significant improvement.

Example:
```
<table>
        <caption>Famous Paintings</caption>
        <tr>
        <th scope="col">Title</th>
        <th scope="col">Artist</th>
        <th scope="col">Year</th>
        <th scope="col">Width</th>
        <th scope="col">Height</th>
        </tr>
        <tr>
        <td>The Death of Marat</td>
        <td>Jacques-Louis David</td>
        <td>1793</td>
        <td>162cm</td>
        <td>128cm</td>
        </tr>
        <tr>
        <td>Burial at ornans</td>
        <td>Gustave Courbet</td>
        <td>1849</td>
        <td>314cm</td>
        <td>663cm</td>
        </tr>
</table>
```

Accessible Forms

HTML forms are also potentially problematic from an accessibility standpoint. If you remember the advice from the WAI about providing keyboard alternatives and text alternatives, your forms should be much less of a problem. The forms in this chapter already made use of the <fieldset>, <legend>, and <label> elements, which provide a connection between the input elements in the form and their actual meaning.

In other words, these controls add semantic content to the form. While the browser does provide some unique formatting to the <fieldset> and <legend> elements, their main purpose is to logically group related form input elements together with the <legend> providing a type of caption for those elements.
You can of course use CSS to style (or even remove the default styling) these elements.
The <label> element has no special formatting (though we can use CSS to do so). Each <label> element should be associated with a single input element. You can make this association explicit by using the for attribute, as shown below. Doing so means that if the user clicks on or taps the <label> text, that control will

```
<label for="f-title">Title: </label>
    <input type="text" name="title" id="f-title"/>
<label for="f-country">Country: </label>
    <select name="where" id="f-country">
    <option>Choose a country</option>
    <option>Canada</option>
    <option>Finland</option>
    <option>United States</option>
    </select>
```

Microformats

The web has millions of pages in it. Yet, despite the incredible variety, there is a surprising amount of similar information from site to site. Most sites have some type of Contact Us page, in which addresses and other information are displayed; similarly, many sites contain a calendar of upcoming events or information about products or news. The idea behind **microformats** is that if this type of common information were tagged in similar ways, then automated tools would be able to gather and transform it.
Thus, a **microformat** is a small pattern of HTML markup and attributes to represent common blocks of information such as people, events, and news stories so that the information in them can be extracted and indexed by software agents. Figure 4.29 illustrates this process.

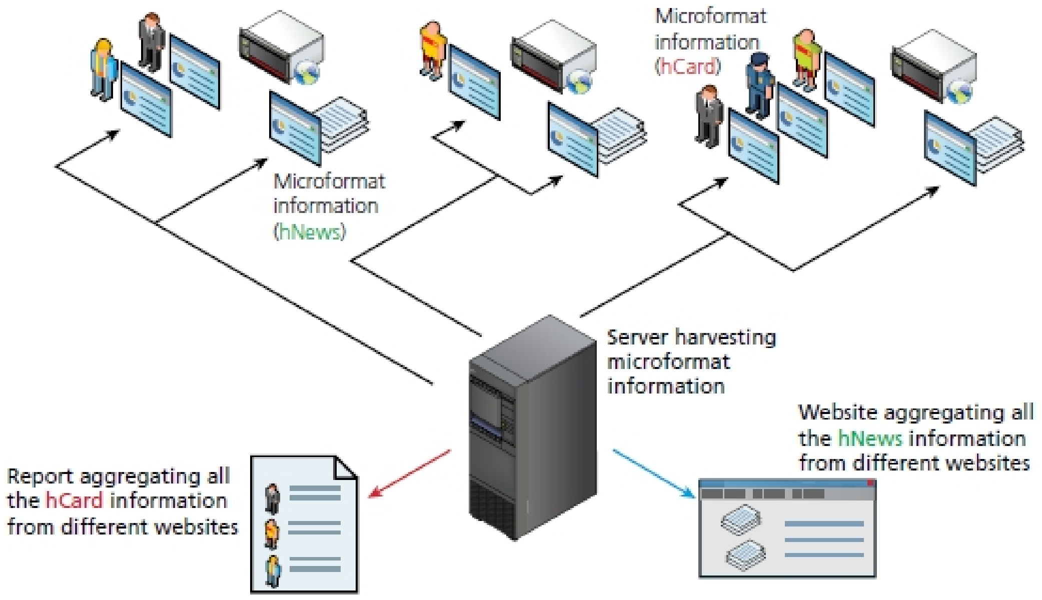

Microformat information (hCard)
Microformat information (hNews)
Server harvesting microformat information
Website aggregating all the hNews information from different websites
Report aggregating all the hCard information from different websites

FIGURE 4.29 Microformats

One of the most common microformats is **hCard**, which is used to semantically mark up contact information for a person. Google Map search results now make use of the hCard microformat so that if you used the appropriate browser extension, you could save the information to your computer or phone's contact list.
Listing 4.2 illustrates the example markup for a person's contact information that uses the hCard microformat.

```
<div class="vcard">
    <span class="fn">Randy Connolly</span>
    <div class="org">Mount Royal University</div>
    <div class="adr">
        <div class="street-address">4825 Mount Royal Gate SW</div>
        <div>
           <span class="locality">Calgary</span>,
           <abbr class="region" title="Alberta">AB</abbr>
           <span class="postal-code">T3E 6K6</span>
        </div>
        <div class="country-name">Canada</div>
    </div>
    <div>Phone: <span class="tel">+1-403-440-6111</span></div>
</div>
```

LISTING 4.2 Example of an hCard