

- » **With the `rgb()` function:** You can select the color using decimal values from 0 to 255 for the red, green, and blue intensities. To specify the same color using the `rgb()` method, you'd use the following:

```
p {color: rgb(255, 165, 0);}
```

If you're not picky about the shade of red you want, the first method will work just fine. But odds are, you'll want to be more precise in your color selection (for example, matching the shade of red to the red in your company's logo), so you'll want to use one of the other two methods. Which of the other two methods you use is a matter of personal preference.

The updated CSS3 standard provides four new ways of working with colors in your web pages:

- » **RGBA:** Adds an opacity value to the standard RGB settings
- » **HSL:** Defines the color as a hue, saturation, and lightness percentage
- » **HSLA:** Defines the color as an HSL value, plus adds an opacity value
- » **Opacity:** Defines a transparency value to make the element more opaque

The main addition to the CSS3 color scheme is the opacity feature. The opacity feature provides the ability to make elements transparent, or faded. The opacity value ranges from 0.0 (fully transparent) to 1.0 (no transparency, also called opaque).

Here's an example to demonstrate just how changing colors in elements works:

1. **Open your favorite text editor, program editor, or integrated development environment (IDE) package.**
2. **Enter the following code into the editor window:**

```
<!DOCTYPE html>
<html>
<head>
<title>Testing colors in CSS3</title>
<style>
p {
  font-family: Arial, Helvetica, sans-serif;
  color: #ff0000;
  background-color: cyan;
}
```

```
h1 {
    color: rgb(255, 165, 0);
    background-color: green;
}
</style>
</head>
<body>
<h1>Testing the color scheme</h1>
<p>
The quick brown fox jumps over the lazy dog.
</p>
<h1>This is the end of the color test</h1>
</body>
</html>
```

3. Save the program as `colortest.html` in the DocumentRoot folder of your web server.

If you're using XAMPP, it's `c:\xampp\htdocs` for Windows or `/Applications/XAMPP/htdocs` for macOS.

4. Start the web server.

If you're using XAMPP, launch the XAMPP Control Panel and then click the Start button for the Apache web server.

5. Open your browser and go to the URL for the new file:

```
http://localhost:8080/colortest.html
```

Note: You may need to change the port in the URL to what your web server uses.

6. Stop the web server and close the browser.

You should see in the output from your web page that the browser uses different colors for the `h1` elements and the `p` elements. However, notice that there's some whitespace between the elements, as shown in Figure 2-2.

You didn't define any space between the `p` and `h1` elements in the HTML5 code, so why is that there? You may be thinking that something has gone wrong with the browser, but actually, it's a feature of CSS that I cover next.

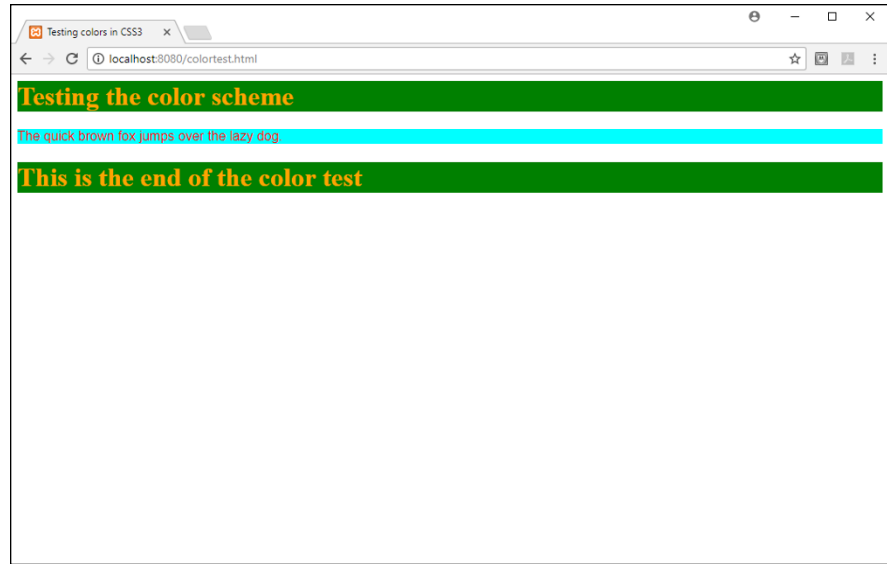


FIGURE 2-2:
Displaying
elements with
different colors
in CSS3

Working with the Box Model

CSS3 handles all elements on the web page using the *box model*, which defines the area inside and around the element and provides a way for you to alter the style of those features. Figure 2-3 shows the box model defined in CSS3.

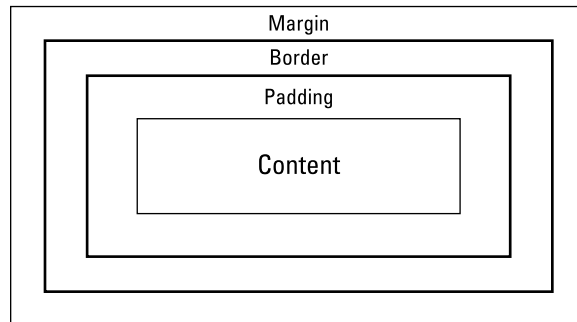


FIGURE 2-3:
The CSS3 box
model.

The box model defines four different sections in the element. Working from the inside out, they are as follows:

- » **Content:** The text or image the element contains
- » **Padding:** The space around the content

- » **Border:** An area, usually visible, that goes around the content and padding
- » **Margin:** The space outside of the element border, between elements

With CSS3, you can alter the padding, margin, and border around an element to help make it stand out in the web page. You do that using the padding, margin, and border style properties.

Let's correct the `colortest.html` code to remove the margin around the elements and add some extra padding to see how that changes things:

1. **Open the `colortest.html` file you created in the “Playing with color” section in your favorite text editor, program editor, or IDE package.**
2. **Modify the `p` and `h1` element styles to set the element margins to `0px` and add `10px` of padding.**

The styles should now look like this:

```
<style>
p {
  font-family: Arial, Helvetica, sans-serif;
  color: #ff0000;
  background-color: cyan;
  margin: 0px;
  padding: 10px;
}

h1 {
  color: rgb(255, 165, 0);
  background-color: green;
  margin: 0px;
  padding: 10px;
}
</style>
```

3. **Save the updated `colortest.html` file.**
4. **Start the web server.**
If you're using XAMPP, launch the XAMPP Control Panel and then click the Start button for the Apache web server.
5. **Open your browser and go to the URL for the new file:**

```
http://localhost:8080/colortest.html
```

Note: You may need to change the port in the URL to what your web server uses.

6. Stop the web server and close the browser.

Figure 2-4 shows the web page this code produces.

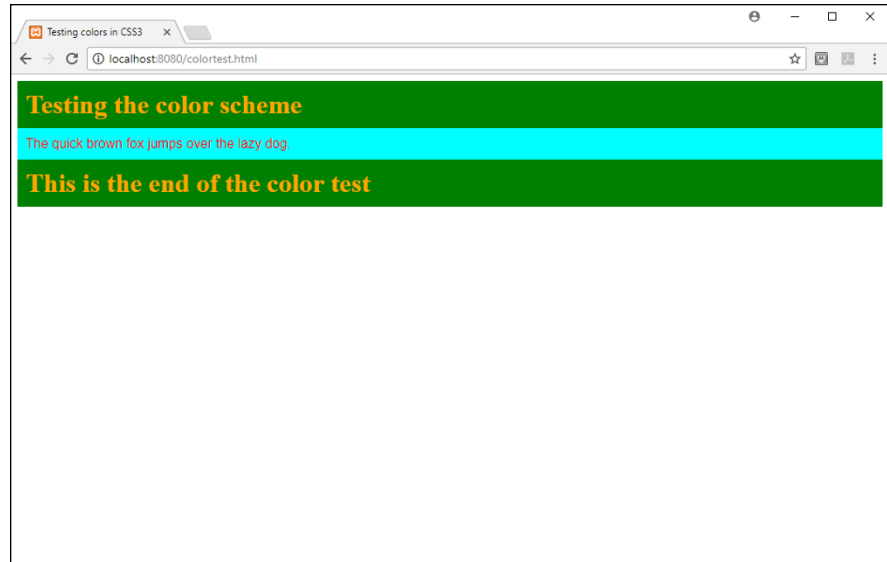


FIGURE 2-4:
The updated
colortest.html
file output.

Notice that the white space is gone and the background space around the text in the headings and paragraph is larger. Feel free to play around with the margin and padding numbers in the HTML5 code and watch how it changes the display results.

Styling Tables

The previous chapter explains how to create tables using HTML5. Older versions of HTML defined attributes in the table element to help add some features, such as creating borders around the table cells and sizing the table cells. However, HTML5 removed all those attributes, so it's up to CSS to provide those features.

Table borders

When you're presenting data in tables, you may want to create borders around the table and around the individual cells in the table. You do that with the CSS `border` property:

```
table {border: 1px solid black;}
```

The first value in the `border` property (`1px`) is the width of the border. The second value (`solid`) is the type of border; you can specify `dashed`, `dotted`, `double`, or `solid` for the border type. The third value (`black`) specifies the color of the border.

You can add borders around any of the table family of elements — `table`, `th`, `tr`, or `td`. However, if you specify the `border` property for all of them, you'll see double borders around the individual cells. To prevent that from happening, add the `border-collapse` property to the rule, and set its value to `collapse`.



TIP

If you only want to show horizontal lines between the table rows, you can use the `border-bottom` property for the `tr` element. This only creates borders at the bottom of each row.

Follow these steps to add borders around a table:

1. **Open the `mytable.html` file that you created in the preceding chapter in your favorite text editor, program editor, or IDE package.**

If you haven't yet read Chapter 1 of this minibook, you'll have to turn back and at least work through the section on tables before proceeding with these steps. I'll wait for you!

2. **Add a style element to the head section of the document to define the table styling rule:**

```
<style>
  table tr td {
    border: 1px solid black;
    border-collapse: collapse;
  }
</style>
```



REMEMBER

I included the `border-collapse` property to prevent double borders from appearing.

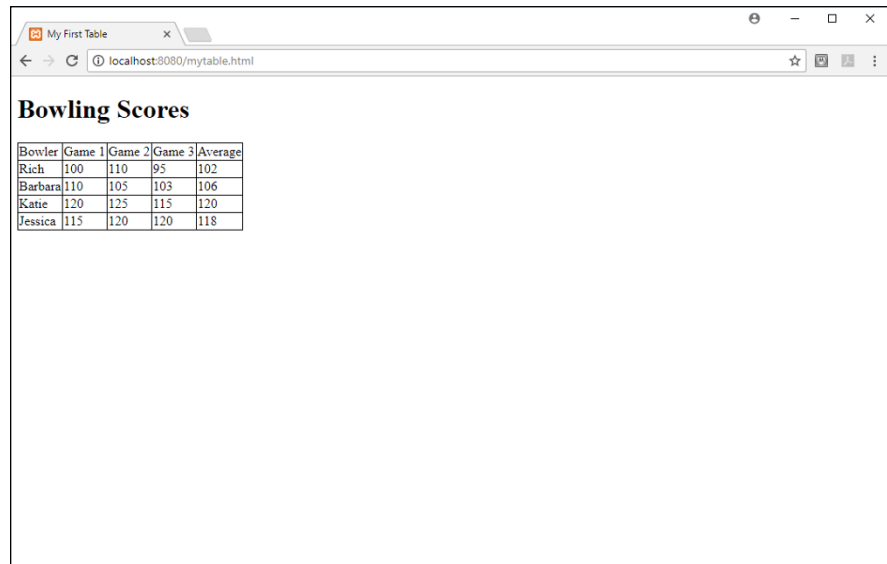
3. **Save the file.**

4. Start your web server software, open your browser, and go to the following URL:

```
http://localhost:8080/mytable.html
```

5. Close the browser and stop your web server software.

With the added stylings, you should see a single border line around each table cell and a single border line around the entire table, as shown in Figure 2-5.



The screenshot shows a web browser window with the title "My First Table" and the address bar displaying "localhost:8080/mytable.html". The main content of the page is a table titled "Bowling Scores". The table has five columns: "Bowler", "Game 1", "Game 2", "Game 3", and "Average". The data rows are: Rich (100, 110, 95, 102), Barbara (110, 105, 103, 106), Katie (120, 125, 115, 120), and Jessica (115, 120, 120, 118). The table is styled with a single border line around each cell and a single border line around the entire table.

Bowler	Game 1	Game 2	Game 3	Average
Rich	100	110	95	102
Barbara	110	105	103	106
Katie	120	125	115	120
Jessica	115	120	120	118

FIGURE 2-5:
Adding a border
to the table.

Now that you have borders around each cell, it may seem a bit more obvious how cramped the data inside the table looks. You can do some more playing around with sizing and positioning the text inside each cell. I cover that in the next section.

Table data

As you can see in Figure 2-5, by default, the browser creates the table cells just large enough to contain the largest data value in the cells. That can make for a somewhat cramped table. Fortunately, you can add a little more space around the data in the table cells using some additional CSS properties.

Padding the cells

A padded cell sounds somewhat ominous, but adding the padding property to your table cells can make a huge difference in the appearance of the table data:

```
table tr td {
    border: 1px solid black;
    border-collapse: collapse;
    padding: 10px;
}
```

When you provide some additional space inside the table cells, you have some more options on where the data appears within the table.

Aligning text in the cells

You can align the data to the left side, center, or right side of the cell with the `text-align` property:

```
table th {
    border: 1px solid black;
    border-collapse: collapse;
    padding: 10px;
    text-align: center;
}
```

This definition centers the text in the table header (th) elements. If you also want to move the text upward inside the cell, use the `vertical-align` property.

Coloring tables

Just using the default black-and-white tables can quickly put your site visitors to sleep! Add the `color` and `background-color` properties to your table to make it stand out. You can apply the colors to the entire table, individual rows, or even individual cells.

To simulate the old mainframe printer report style using alternating row colors in the table, use the `nth-child` pseudo-class to style every other row in the table as a different color, like this:

```
tr: nth-child(even) {
    background-color: lightgreen;
}
```

If you're old enough to remember the mainframe computer report days, this should bring back memories!

Another feature that comes in handy is to use the `hover` pseudo-class to change the background color of an individual cell as your site visitor hovers the mouse pointer over it:

```
td: hover {  
    background-color: yellow;  
}
```

Now things are really starting to get fancy!

Positioning Elements

By default, browsers place elements in the window following a set order. As the web page defines each element, the browser places it in the window starting at the upper-left corner of the window, proceeding from left to right, and top to bottom.

To demonstrate this, let's run a quick test. You'll create a web page that contains five sections:

- » A header to display at the top of the web page
- » A footer to display at the bottom of the web page
- » A navigation section to display on the left side of the middle section
- » An aside section to display on the right side of the middle section
- » A main content section to display in the middle of the middle section

This is a pretty standard web page layout structure, which I'm sure you've seen lots of times as you've browsed the web.

Follow these steps to run the test:

- 1. Open your favorite text editor, program editor, or IDE package, and enter the following code:**

```
<!DOCTYPE html>  
<html>  
<head>  
<title>Positioning Test</title>  
<style>
```

```
header {
    background-color: red;
    margin: 0px;
    padding: 10px;
    height: 25px;
    width: 600px;
}

nav {
    background-color: blue;
    margin: 0px;
    padding: 10px;
    height: 125px;
    width: 200px;
}

section {
    background-color: green;
    margin: 0px;
    padding: 10px;
    height: 125px;
    width: 200px;
}

aside {
    background-color: yellow;
    margin: 0px;
    padding: 10px;
    height: 125px;
    width: 200px;
}

footer {
    background-color: orange;
    margin: 0px;
    padding: 10px;
    height: 25px;
    width: 600px;
}
</style>
</head>
<body>
<header><p>This is the header</p></header>
<nav><p>Navigation</p></nav>
```

```
<section><p>Section</p></section>
<aside><p>Aside</p></aside>
<footer><p>This is the footer</p></footer>
</body>
</html>
```

2. **Save the file as** `positiontest.html` **in the** `DocumentRoot` **folder of your web server.**
3. **Start the web server, open your browser, and go to the following URL:**

```
http://localhost:8080/positiontest.html
```

4. **Close the browser and stop the web server.**

This test creates a web page that contains a header section, a navigation section, a main content section, an aside section, and a footer section. It uses the `height` and `width` style properties to define how large each section should be and sets a different background color for each section so you can tell them apart on the web page. However, when you display the web page, you'll probably be a bit disappointed with the results, which are shown in Figure 2-6.

The browser positioned each of the different sections in the order you defined them, each on top of the other. Ouch! That's not what we wanted at all!

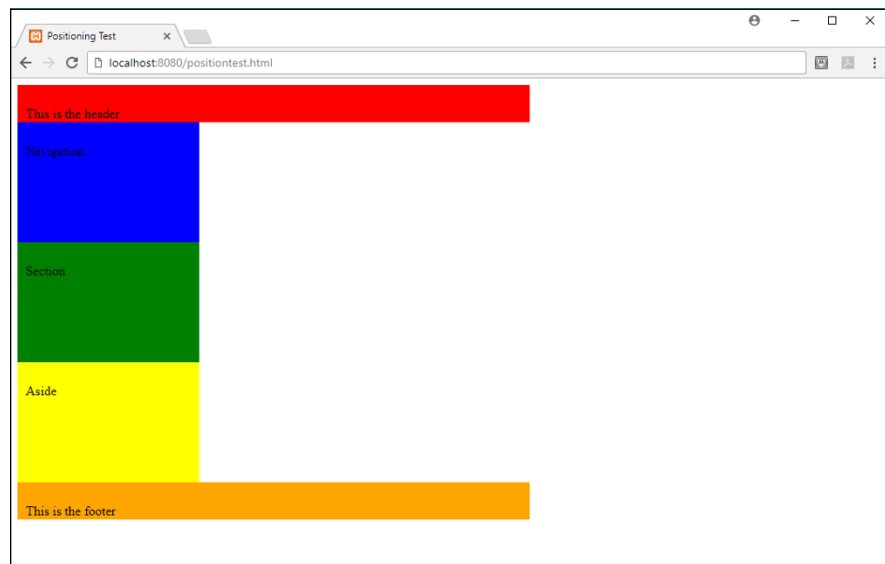


FIGURE 2-6: Displaying the web page with no positioning.

To get the browser to place the different web page sections the way we want, we'll need to use some of the positioning properties available in CSS. The next sections walk you through how to do that.

Putting elements in a specific place

Placing elements in specific locations on the web page requires using the *positioning properties* available in CSS. There are three main positioning properties that are normally used:

- » `position`: Sets the position method the browser should use to place the element
- » `top`: Defines the location for the top of the element
- » `left`: Defines the location for the left side of the element

The `position` property defines what method the browser uses to place the element in the web page. There are four different positioning methods:

- » `absolute`: Changes the element's position relative to the nearest positioned element that precedes it.
- » `fixed`: Places the element in a fixed location in the browser window. If the site visitor scrolls the window, the element stays in the same spot.
- » `relative`: Changes the element's position relative to the default position.
- » `static`: Places the element in its normal location in the web page following the default placement rules.

To use the `absolute`, `fixed`, and `relative` positioning methods, you need to define the location in the browser window where the element will be placed. You do that using the `top` and `left` properties.

Let's change the `positiontest.html` test file to use absolute positioning to place the sections. Just follow these steps:

1. **Open the `positiontest.html` file in your favorite text editor, program editor, or IDE package.**

2. Modify the styles defined so they look like this:

```
<style>

header {
    background-color: red;
    margin: 0px;
    padding: 10px;
    height: 25px;
    width: 600px;
    position: absolute;
    top: 0px;
    left: 0px;
}

nav {
    background-color: blue;
    margin: 0px;
    padding: 10px;
    height: 125px;
    width: 200px;
    position: absolute;
    top: 46px;
    left: 0px;
}

section {
    background-color: green;
    margin: 0px;
    padding: 10px;
    height: 125px;
    width: 200px;
    position: absolute;
    top: 46px;
    left: 201px;
}

aside {
    background-color: yellow;
    margin: 0px;
    padding: 10px;
    height: 125px;
    width: 200px;
}
```

```
    position: absolute;
    top: 46px;
    left: 402px;
}

footer {
    background-color: orange;
    margin: 0px;
    padding: 10px;
    height: 25px;
    width: 600px;
    position: absolute;
    top: 192px;
    left: 0px;
}
</style>
```

3. Save the updated `positiontest.html` file as `positiontest2.html`.
4. Start your web server, open a browser, and go to the following URL:

```
http://localhost:8080/positiontest2.html
```

5. Close the browser and stop the web server.

The additional code sets the positioning method for the browser to use for each section to `absolute`, which means it will place the sections at exactly the place in the browser window you define using the `top` and `left` properties. When you display the web page, you should see the result as shown in Figure 2-7.

Now things are starting to look like a real web page!

Floating elements

Absolute positioning has made a huge difference in how we can lay out elements in our web pages, but it doesn't solve all problems. You've probably already realized that trying to figure out the exact location for each element in a complicated web page would be somewhat difficult. Also, you'll notice as you resize the browser window that the sections stay in a fixed location and size — they don't expand or shrink with the browser window. Fortunately, there's a way you can avoid these problems.



FIGURE 2-7:
Using absolute
positioning to
place sections in
the web page.

CSS uses a feature called the `float` property to aid in positioning elements in the web page using a more dynamic method. The `float` property allows you to take an element out of the normal positioning flow in the web page and position it within the right or left edge of its parent container element. You don't need to calculate the exact position for the elements within the parent.

The format of the float property is pretty simple:

```
float: position
```

The position value can be `none`, `left`, or `right`.

The `float` property is most often used to create columns in a web page layout. Instead of using absolute positioning for the columns, you define a parent container element, and then just float the column elements inside the parent.

Let's give that a try with our `positiontest.html` example. You'll add a `div` element to use as the container for the middle three sections (`nav`, `section`, and `aside`) in the web page document. Follow these steps:

- 1. Open the original `positiontest.html` file in your favorite text editor, program editor, or IDE package.**

2. Modify the styles defined so they look like this:

```
<style>

header {
    background-color: red;
    margin: 0px;
    padding: 10px;
    height: 25px;
    width: 100%
}

nav {
    background-color: blue;
    margin: 0px;
    padding: 10px;
    height: 125px;
    width: 20%;
    float: left;
}

section {
    background-color: green;
    margin: 0px;
    padding: 10px;
    height: 125px;
    width: 55%;
    float: left;
}

aside {
    background-color: yellow;
    margin: 0px;
    padding: 10px;
    height: 125px;
    width: 20%;
    float: right;
}

footer {
    clear: both;
    background-color: orange;
    margin: 0px;
    padding: 10px;
}
```



```

height: 25px;
width: 100%;
}
</style>

```

3. **Modify the HTML code to add a div parent element around the nav, section, and aside elements.**

That code should look like this:

```

<body>
<header><p>This is the header</p></header>
<div id="container">
<nav><p>Navigation</p></nav>
<section><p>Section</p></section>
<aside><p>Aside</p></aside>
</div>
<footer><p>This is the footer</p></footer>
</body>

```

4. **Save the updated positiontest.html file as positiontest3.html.**
5. **Start your web server, open your browser, and go to the following URL:**

```
http://localhost:8080/positiontest2.html
```

6. **Close the browser and stop the web server.**

When you view the resulting web page, it should look similar to Figure 2-8.



FIGURE 2-8:
Using float
positioning to
place sections in
the web page.

The `float` property in the `nav`, `section`, and `aside` elements causes them to float within the parent `div` element. I gave the parent element an `id` attribute value of `container` to help me remember its purpose. It's not necessary for it to have an `id` attribute defined because it isn't styled by itself.

Each of the inner sections appears side by side, as long as there's enough space for them in the browser window. By using a percentage value for the width, this creates what's called a *liquid layout*. With a liquid layout, if you resize the browser window, the individual section elements resize as well. If you resize the browser window too small, the browser automatically repositions the elements so that they all appear in the window.

- » Creating a data form
- » Examining the form fields
- » Looking at additions to HTML5
- » Validating forms

Chapter 3

HTML5 Forms

Quite possibly one of the most common ways that PHP programming helps is by processing data entered into an HTML5 form. There are plenty of applications that require data entry — from keeping track of your bowling team to filling out online job applications. HTML forms have been around for a long time, and with HTML5 it's sure to stick around for years to come. This chapter shows you how to create forms for your web applications using the HTML5 form features.

Understanding HTML5 Forms

A dynamic web application requires some type of interaction with the site visitors who use it. That interaction is usually done with a form. Forms allow you to ask your site visitor for information using many of the same input interfaces that are commonly found in Windows and macOS systems, such as text boxes, drop-down lists, and radio buttons.

Before you can create a form for your web application, you need to do some house-keeping for HTML5. You need to define the form and how the browser should handle the data the site visitor enters into it. This section explains just how to do that.

Defining a form

It's probably not too surprising that the HTML element you use to create a form is the *form element*. The form element has a simple enough format:

```
<form attributes>  
  form elements  
</form>
```

The `<form>` tag defines the start of the form area, which contains all the elements that create the form fields. The `</form>` tag defines the end of the form area.

The form element has lots of attributes that define just how the browser handles the data in the form. Table 3-1 shows all the attributes available.

TABLE 3-1 The Form Element Attributes

Attribute	Description
accept-charset	Specifies the character used in the form if it's different from the web page
action	Defines the URL where the browser should send the form data
autocomplete	Specifies whether the browser is allowed to use the autocomplete feature
enctype	Specifies the encoding the browser uses to submit the form data
method	Specifies the transfer method the browser should use to send the data
name	Defines a name assigned to the form
novalidate	Specifies that the browser shouldn't validate the data
target	Specifies the target window for the action URL

Often, when you create a form, you don't need to worry about setting all the attributes shown in Table 3-1; you can use the standard default values. Here are the attributes you'll probably work with the most:

- » **action:** You'll need to define the URL of the web page that will accept and process the form data. Usually, this is a page that contains server-side programming, such as PHP code.
- » **enctype:** If your form contains binary data (such as an upload file), you'll need to set the encoding type so the server knows there's binary data involved with the form data.

» method: You'll need to define how the browser sends the data to the server, using either the HTTP GET method or the HTTP PUT method.

- GET: The HTTP GET method sends the form data as part of the URL to the server. It embeds the form field names and data values together in the URL. Often, if you fill out a form on the Internet and click the Submit button, you'll see a URL that looks something like this:

```
http://myhost.com/index.php?content=store&id=100
```

This means the form used the GET method to send two form fields back to the server. Because the server needs to identify each value, the GET method associates the form field name with each value:

```
content=store
id=100
```

These values indicate that a form field named `content` is set to a value of `store` and a form field name `id` is set to a value of `100`.

This method is a great way to quickly send small pieces of form data to the server, but it isn't recommended for larger forms. For forms that send lots of data, you're better off using the HTML PUT method.

- PUT: The PUT method sends the data behind the scenes in the HTTP request packets instead of using the URL. The data isn't seen in the address bar of the browser; instead, it's processed by the client browser and server as part of the HTTP communication behind the scenes.

Just because the data isn't easily seen doesn't mean it's secure. The data sent by the PUT method is still sent in plain text in the HTTP request message. Any person with a network sniffer can still read that data. The only secure method of sending data is with an encrypted HTTPS session.



WARNING

After you define the form and how it will send the form data, you're ready to start adding some form fields.

Working with form fields

The original version of HTML didn't specify all that many form field elements for us to use. The list of form field elements that were available are shown in Table 3-2.

TABLE 3-2

HTML Basic Form Field Elements

Field	Description
button	A clickable area on the web page that triggers an action
input	Provides a single interface for one data value
select	A list of multiple objects in a drop-down list
submit	Signals to the browser to send the form data to the action URL
textarea	A larger multiline box for entering larger amounts of text

HTML5 adds a couple more form field elements to the list:

- » data list: Provides a list of predefined options
- » keygen: Creates a public/private key pair for authentication
- » output: Creates an area to display results from a process

The following sections walk you through how to use each of these elements in your web forms.

Using Input Fields

The *input element* is the most versatile of the form field elements. It provides for a few different types of interfaces to input data. You define the type of input field element to use by adding the `type` attribute to the tag:

```
<input type="type" attributes>
```

The HTML standard defines a handful of different input field types. If you've ever interacted with a Windows or macOS workstation (and who hasn't these days?), you're familiar with all these input types. The following sections explain how to use each one.

Text boxes

The *text box* is the workhorse of the form. How many times have you filled out an online form that asked for your name, age, address, and so on? All these single-line form fields use the text box input type.