Chapter 3: HTML Forms and iFrames

Learning Outcomes:

- ✓ Create web forms that demonstrate a variety of input types available in HTML
- ✓ Apply CSS to a form to improve presentation
- ✓ Implement an iframe and HTML symbols
- ✓ Identify the composition of a URL address

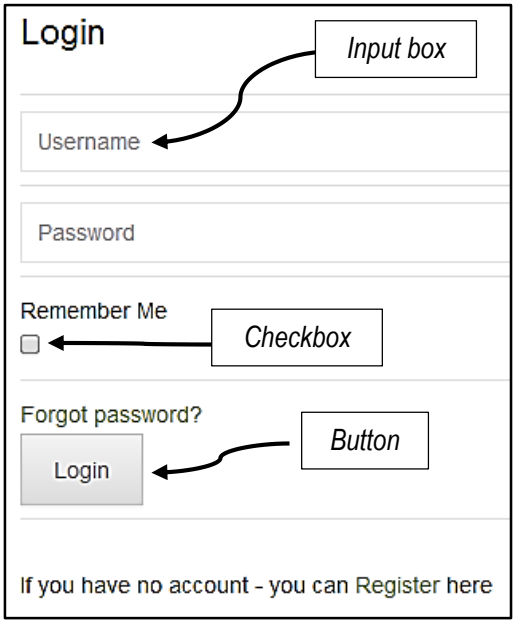
Prerequisite Knowledge:

- ✓ Complete Chapter 2
- ✓ Be able to apply CSS to HTML elements using the `<style>` tag in the document head
- ✓ If working in a Windows environment, have access to the command prompt

Keywords: Class, Pseudo, Property, Source, Scheme, Host, ASCII

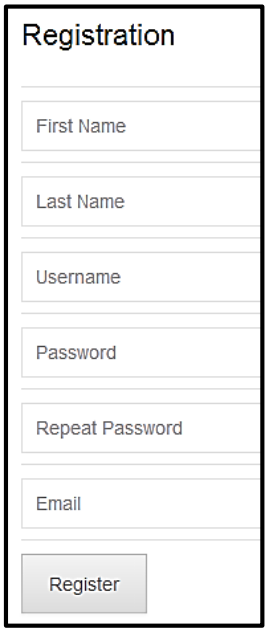
3.1 Theory: HTML Web Forms

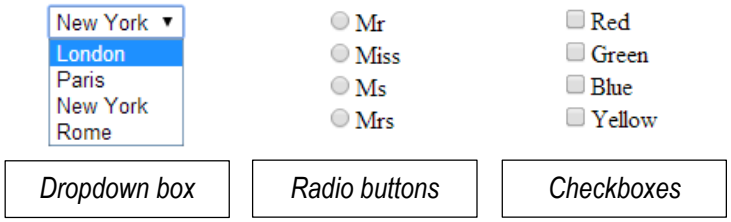
Chapter 3 will examine the role of HTML web forms, as well as demonstrate how to define them in a web page. Forms are an integral part of the modern-day web; for example, they are used when logging into Facebook, when searching for something on Google and even when entering credit card details into Amazon. In other words, they are used to gain some form of input from web users. Once the input from a user has been gained, it is up to the web developer to decide what to do with that information; some ideas might include storing the data into a database, maybe sending the data to an email inbox or possibly even performing a calculation with the data in the web browser itself.



Input Types

The HTML specification provides a wide variety of input elements that can be used to easily gain information from web users. Some of the input elements available in the specification include checkboxes, radio buttons, dropdown boxes, combo boxes, text boxes, text areas and submit/reset buttons.





As stated previously, HTML does not format the presentation of elements, so by default web forms are plain in design. However, the appearance of web forms can by massively improved by applying CSS to them. This chapter provides a generic overview of how CSS can be applied to web forms to improve their appearance; however, it is worth noting that CSS is revisited in subsequent chapters.

3.2 Practical: Defining HTML Web Forms

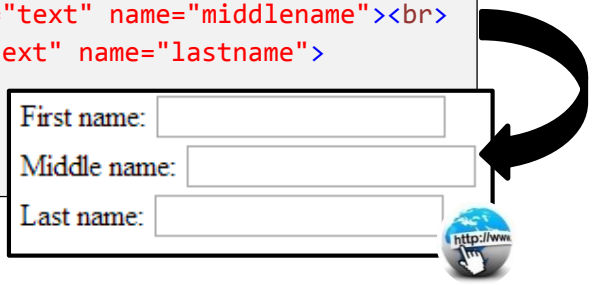
Text Field Input Type

To define the start and end of a web form the opening and closing `<form></form>` tags are used; all input elements (for example, textboxes) are then defined between these two tags. The `<input>` tag is a stand-alone tag used to create an input element. The 'type' of the element is determined using the `type` attribute and can include values such as checkbox, button and radio. In the example below, the `type` attribute is set to 'text' for each of the three input elements; this creates three textboxes to hold the first, last and middle name of the user. The `name` attribute is used to specify the input element's individual name, enabling it to be easily identified once data is submitted by the user. Note that in this example, the textbox labels do not have any special tag, they are written directly between the `<form></form>` tags. The `<br>` tag is then used to move each textbox onto an individual line in the web page.

Activity 3.1

Create a webpage that contains a HTML web form. The web form should have several textboxes to record a web user's address. Experiment with some of the additional attributes available; for example, size, title, value, maxlength and required.

```
<!DOCTYPE html>
<html>
<head>
	<title>My HTML Form</title>
	<meta charset="UTF-8">
</head>
<body>
	<form>
		First name: <input type="text" name="firstname"><br>
		Middle name: <input type="text" name="middlename"><br>
		Last name: <input type="text" name="lastname">
	</form>
</body>
</html>
```



HTML Tips!

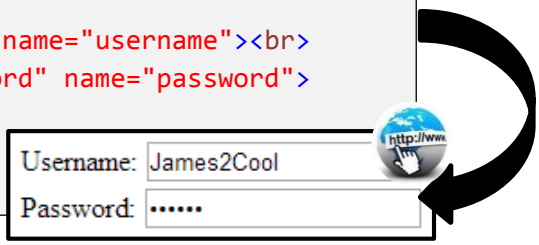
There are several other useful attributes that can be applied to a text field. Below are some examples of them;

```
<input type="text" name="firstname" size="20">
<!-- Sets the size (width) of the textbox in characters-->

<input type="text" name="firstname" value="James">
<!-- Sets the default value of the textbox, in this case to 'James'-->

<input type="text" name="firstname" title="Enter a name">
<!-- Displays the title text when the cursor is hovered over the textbox-->

<input type="text" name="firstname" required>
<!-- Sets the textbox to a mandatory field (it must have a value on submission)-->

<input type="text" name="firstname" maxlength="10">
<!-- Sets the maximum number of characters that can be entered into the textbox-->
```


Password Field Input Type

Another field type available in the HTML specification is 'password'. In appearance, the input type looks the same as a standard textbox; however, when the user begins to type into the textbox the characters are automatically masked from view. The attributes that can be applied to a 'text' type can also be applied the same to the 'password' type.

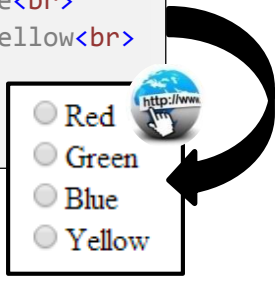
```
<!DOCTYPE html>
<html>
<head>
	<title>My HTML Form</title>
	<meta charset="UTF-8">
</head>
<body>
	<form>
		Username: <input type="text" name="username"><br>
		Password: <input type="password" name="password">
	</form>
</body>
</html>
```



Radio Button Input Type

Radio buttons are a popular input method used to force the user to choose one applicable option out of a series of multiple options; for example, male or female (as a user cannot be both). To create a radio button, the `type` attribute is set to 'radio' and the `name` attribute of each radio button is set to the same value, enabling radio buttons to be grouped together (this is because a form may contain more than one radio button list). The `value` attribute is used to indicate a radio button's value on submission; in other words, defines the value that would be submitted (perhaps to a web server) if that particular radio button is selected. Similar to textboxes, the label text is placed outside of the input tag and the `<br>` tag is used to move each radio button onto an individual line in the web page.

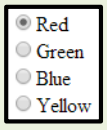
```
<!DOCTYPE html>
<html>
<head>
	<title>My HTML Form</title>
	<meta charset="UTF-8">
</head>
<body>
	<form>
		<input type="radio" name="colours" value="red">Red<br>
		<input type="radio" name="colours" value="green">Green<br>
		<input type="radio" name="colours" value="blue">Blue<br>
		<input type="radio" name="colours" value="yellow">Yellow<br>
	</form>
</body>
</html>
```



HTML Tips!

A radio button list can have a default value set, by using the `checked` attribute. An example of this is shown below:

```
<input type="radio" name="colours" value="red"
checked>Red<br>
```



Checkbox Input Type

The checkbox input type is similar to a radio button, the key difference being that a checkbox list enables multiple criteria to be selected, whereas a radio button list only allows one criterion to be chosen by the user. To create a checkbox, the `type` attribute (of the input element) is set to 'checkbox'. Similar to a radio button list the `name` attribute of each checkbox is set to the same value to group the elements together. On submission, the `value` attribute determines the value of a checkbox if it has been checked. Again (as previously), the label text is placed outside of the input tag. The `checked` attribute can also be added to the input element, to specify which checkboxes are checked by default.

```
<!DOCTYPE html>
<html>
<head>
	<title>My HTML Form</title>
	<meta charset="UTF-8">
</head>
<body>
	<form>
		<input type="checkbox" name="city" value="london">London<br>
		<input type="checkbox" name="city" value="paris">Paris<br>
		<input type="checkbox" name="city" value="rome">Rome<br>
		<input type="checkbox" name="city" value="newyork">New York<br>
	</form>
</body>
</html>
```

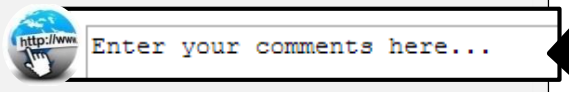
Text Area Input Type

Text areas are large text fields that allow more substantial amounts of information to be entered (and seen); they are particularly useful for allowing web users to add additional comments/notes and/or memo type information. The start and end of a text area is defined using the opening and closing `<textarea></textarea>` tags (as opposed to using the `<input>` element). The `rows` and `cols` attributes are used to specify the size of the text area in characters. As with other input elements, the `name` attribute enables the text area field to be identified on submission of the data. Any text typed between the `<textarea></textarea>` tags will be the default value of the text area when the web page first loads, the format of which is exactly as it appears in the text file (including white space indentation).

Activity 3.2

Create a webpage that contains a HTML web form. The web form can collect any information of choice, but must demonstrate the use of a radio button list and a checkbox list.

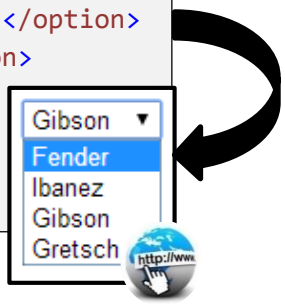
```
<!DOCTYPE html>
<html>
<head>
	<title>My HTML Form</title>
	<meta charset="UTF-8">
</head>
<body>
	<form>
		<textarea rows="15" cols="40" name="comments">
Enter your comments here...
		<textarea>
	</form>
</body>
</html>
```



Dropdown List Input Type

A dropdown list enables a web user to choose a single entry from a dropdown list; in essence, it gathers the same feedback as a radio button list, but uses less space on the web page. To create a dropdown list, the opening and closing `<select></select>` tags are used to define the start and end of the list. The `name` attribute (to identify the source) is also defined inside the opening `<select>` tag; for example, `<select name="guitars">`. Each available item (in the list) is declared between an opening and closing `<option></option>` tag; on submission, the `value` attribute determines the value of the selected list item (for example, the value entered into a database). Finally, the `selected` attribute can be used to set the default selected item in the list box.

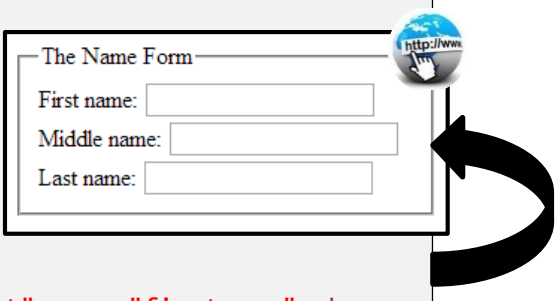
```
<!DOCTYPE html>
<html>
<head>
	<title>My HTML Form</title>
	<meta charset="UTF-8">
</head>
<body>
	<form>
	<select name="guitars">
		<option value="fender">Fender</option>
		<option value="ibanez">Ibanez</option>
		<option value="gibson" selected>Gibson</option>
		<option value="gretsch">Gretsch</option>
	</select>
	</form>
</body>
</html>
```



Field Set and Legend

A field set (groups elements and, by default, applies an outside border) and legend can both be added to a form to improve its appearance. These elements can also be further manipulated using CSS. Below is an example of how the `<fieldset>` and `<legend>` tags can be applied to a form:

```
<!DOCTYPE html>
<html>
<head>
	<title>My HTML Form</title>
	<meta charset="UTF-8">
</head>
<body>
	<form>
	<fieldset>
	<legend>The Name Form</legend>
		First name: <input type="text" name="firstname"><br>
		Middle name: <input type="text" name="middlename"><br>
		Last name: <input type="text" name="lastname">
	</fieldset>
	</form>
</body>
</html>
```



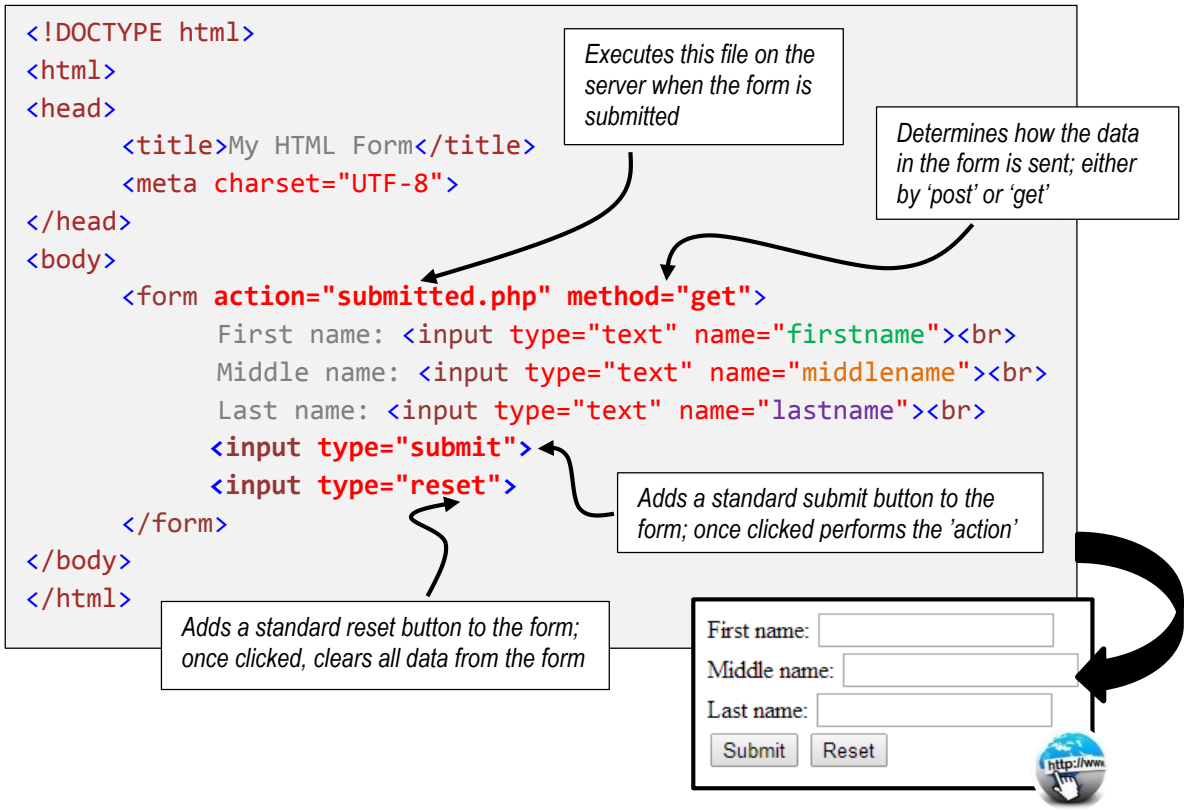
Activity 3.3

Modify the form that was created for activity 3.2 to include a dropdown list, a field set and a legend.

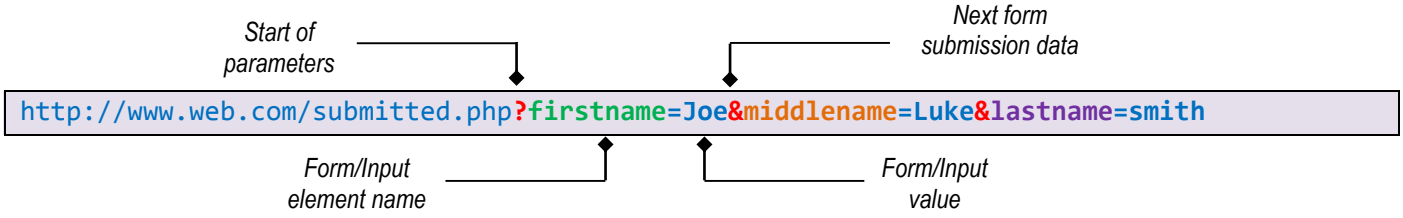
3.3 Theory: Submitting a Form

Post and Get Methods

There is a small problem with all of the forms that were created for activity 3.1, 3.2 and 3.3; the problem, of course, being that none of the forms actually do anything with the captured data (they have no purpose). There is no button to submit the form data either. Solving the submit button issue is straightforward; simply add an input element with the `type` set to 'submit'. A reset button (which will clear the form data) can also be added in the same manner, except changing the `type` to 'reset'. On clicking the submit button, the browser will perform the task stored in the `action` attribute (that is defined in the opening `<form>` tag). In this specific example, find and execute the file called 'submitted.php' on the web server; this file will handle the received data and perform an action with it. In the example below, the form will run the defined action using the 'get' method. The `method` attribute can either be set to 'get' or 'post'.



The `method` attribute simply determines how the information is sent to the web server; either by 'get' or 'post'. There is no method better than the other (they both store data in what is known as an array), but each have their distinct advantages; therefore, the method is normally determined dependent on what the web developer is trying to achieve. The 'get' method sends the form data to the web server via URL parameters; in other words, it attaches it to the end of a web address; for example:



The 'post' method, on the other hand, sends data via the HTTP post method, a method that requests a web server to accept enclosed data sent over the Internet. This method is naturally more secure than the 'get' method, because the values are not clearly displayed and also because the data is stored securely in the body of the HTTP request.

Post vs Get

If both methods can be used, when does a web developer use the 'get' method and when does a web developer use the 'post' method? As stated earlier, data sent by 'get' is visible to all; this method also has a size limitation of about 2,000 characters. Yet, despite these disadvantages, 'get' is useful because the data is part of a URL. This means that form data can be bookmarked, cached (stored in an offline format), distributed/shared with others and stored in the browser history. Therefore, use this method when sending non-sensitive information, such as a search page or blog entry.

The 'post' method, on the other hand, hides the data from web users and has no size limitations; it also supports file uploads to the server. However, as data is hidden, it cannot be bookmarked, shared and/or cached. This method should be used when handling sensitive data; for example, password information and personal data.

The Server (PHP Example)

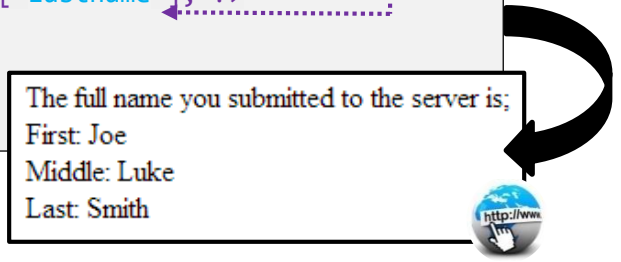
The use of server-side technology is beyond the scope of this text; however, to understand the full role of a HTML web form (and the `action` and `method` attributes) it is worth examining a PHP example. Below demonstrates the syntax that might be found in the 'submitted.php' file (found on the web server when the submit button is clicked); in this case, the PHP code will collect the values (stored in the $_GET global array) from the web form submission and print (echo command) them into a web page for viewing by the user. Note that if the method was set to 'post', then the $_POST global array would be used instead. To collect information from the array, the names of the input elements are used.

form.html

```
...
<form action="submitted.php" method="get">
      First name: <input type="text" name="firstname"><br>
      Middle name: <input type="text" name="middlename"><br>
      Last name: <input type="text" name="lastname"><br>
      <input type="submit">
      <input type="reset">
</form>
...
```

submitted.php

```
<!DOCTYPE html>
<html>
<head>
      <title>My HTML/PHP Response</title>
      <meta charset="UTF-8">
</head>
<body>
      <p>
            The full name you submitted to the server is;<br>
            First: <?php echo $_GET["firstname"]; ?><br>
            Middle:<?php echo $_GET["middlename"]; ?><br>
            Last:  <?php echo $_GET["lastname"]; ?>
      </p>
</body>
</html>
```



3.4 Practical: Mailto Forms and Other Input Types

Mailto Forms

Writing server-side code to handle form submissions is outside the remit of this text; fortunately, however, there is an easy way in HTML to handle form data (without the need of writing a single script). This is achieved by adding the 'MAILTO:' command and a valid email address to the `action` attribute (of the `<form>` element); this way, when a user submits the web form, the browser will seek an application on the client's computer that can be used to email the form data to the given email address. Although this is not the ideal method of handling forms, it is a practical 'work around' that is suitable for beginners who are still learning the web!

HTML Tips!

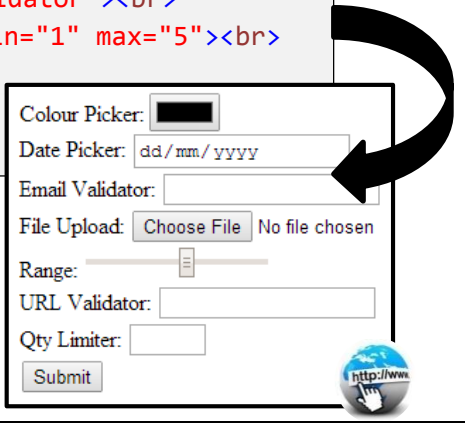
The `enctype` attribute is short for encode type; this is used to state the encoding used on the submitted data. 'Text/plain' means that spaces are converted to '+' symbols, but no special characters are encoded.

```
<!DOCTYPE html>
<html>
<head>
	<title>My HTML Form</title>
	<meta charset="UTF-8">
</head>
<body>
	<form action="MAILTO:person@web.com" method="post" enctype="text/plain">
		First name: <input type="text" name="firstname"><br>
		Middle name: <input type="text" name="middlename"><br>
		Last name: <input type="text" name="lastname"><br>
		<input type="submit">
	</form>
</body>
</html>
```

Other Input Types

HTML5 also provides numerous other input types that are useful when gathering information from web users; this includes colour pickers, calendars, email validators, file uploads, range sliders, URL validators and quantity validators. Below are syntax examples of these additional input types:

```
...
<form action="MAILTO:person@web.com" method="post" enctype="text/plain">
	Colour Picker: <input type="color" name="colour_picker"><br>
	Date Picker: <input type="date" name="date_picker"><br>
	Email Validator: <input type="email" name="email_validator"><br>
	File Upload: <input type="file" name="file_upload"><br>
	Range: <input type="range" name="range" min="1" max="10"><br>
	URL Validator: <input type="url" name="url_validator"><br>
	Qty Limiter: <input type="number" name="qty" min="1" max="5"><br>
	<input type="submit">
</form>
...
```



Activity 3.4

Create a webpage that includes some of the additional input types; the form can be based on any topic. The form should send the form data by email (using the 'MAILTO' feature) to any chosen email address. Test the form to ensure that it works correctly.

3.5 Practical: CSS and Forms

Applying CSS to Forms

CSS can be applied to forms to make them more presentable and attractive. This section will demonstrate, in a step-by-step fashion, how CSS can be applied to a web form. Do **not** be alarmed if the entirety of the CSS syntax is not explained and/or if all of the content is not understood; CSS is explained in-depth in subsequent chapters. The purpose of this section is to give a brief overview of CSS properties and selectors. Below is an example of a simple HTML web form:

```
<!DOCTYPE html>
<html>
<head>
	<title>My HTML Form</title>
	<meta charset="UTF-8">
	<style><!--CSS goes here--></style>
</head>
<body>
	<form class="myform" action="MAILTO:person@web.com" method="post" enctype="text/plain">
		<h1>Contact Form<span>Please complete this form.</span></h1>

		<label><span>Your Name:</span> <input id="name" type="text" name="name" placeholder="Your Full Name"></label>

		<label><span>Your Email:</span> <input id="email" type="email" name="email" placeholder="Your Email Address"></label>

		<label><span>Message:</span> <textarea id="message" name="message" placeholder="Your Message"></textarea></label>

		<label><span>Subject:</span> <select name="selection">
				<option selected>Please choose...</option>
				<option value="">Quote</option>
				<option value="">General Enquiry</option>
				<option value="">Other</option>
		</select></label>

		<label><span> </span> <input type="submit" class="button"></label>
	</form>
</body>
</html>
```

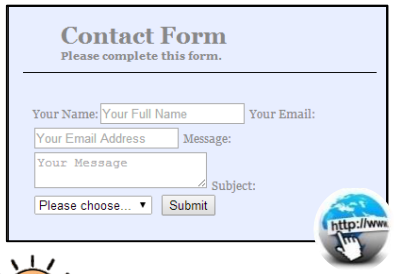
HTML Tips!

There are some new elements in this HTML form. A `<label>` tag defines a label for an input element; in this case, it defines a label for each input element in the form. The `<span>` tag serves a purpose when applying CSS; this will come clearer as the chapter progresses. Finally, the ` ` is used to add a character space in HTML; it is used in the form, because it is important that the span tag has a value for the CSS rules below.

The table below provides a brief overview of CSS syntax used to improve the presentation of the above HTML web form. Apply the CSS code between the opening and closing `<style></style>` tags, found in the head section of the HTML document. The three dots in the table (...) are not syntax, but instead indicate the presence of the previous code; in other words, do not delete any code, just keep adding to the new code as the table progresses. In CSS, a full stop represents a class; a class is syntax that can be applied to any element (unless otherwise stated). For example, the CSS syntax found inside the `.myform` class would be applied to any element labelled with the class name; for example, `<form class="myform">`. Classes can be extended to apply CSS syntax to specific elements too; for example, `.myform h1` would only apply the CSS to `<h1>` elements, found inside any container labelled with the `.myform` class.

<table>
<tr><th>Screen-print</th><th>CSS Code</th><th>Explanation</th></tr>
<tr>
<td> 
</td>
<td>

```
<style>
.myform{
        margin-left:auto;
        margin-right:auto;
        max-width:500px;
        background:#E8EEFF;
        padding:25px;
        font:12px Georgia, Times,  serif;
        color:#888;
        border:1px solid #000;
}
</style>
```

Applies the contained CSS to any element labelled with the '.myform' class. In this example, to the form itself
</td>
<td>The <code>margin-left</code> and <code>margin-right</code> properties are set to auto; this enables the browser to determine the form's left and right margins (the outside area of the form) automatically; this centre aligns the form. The <code>max-width</code> property sets the width to 350 pixels (and cannot exceed this). The <code>background</code> property sets the back colour. The <code>padding</code> property sets the distance between the forms border and the inside content; in this case it is set to 25 pixels. The <code>font</code> property sets the font size and face, while the <code>color</code> property sets the font colour. Finally, the <code>border</code> property applies a one-pixel, solid, black border around the entire form.</td>
</tr>
<tr>
<td> 
</td>
<td>

```
<style>
...
.myform h1{
        padding:0px 0px 10px 40px;
        display:block;
        border-bottom:1px solid #000;
        margin:-10px -15px 30px -10px;
        color:#888;
        font-size:25px;
}
.myform h1>span{
        display:block;
        font-size:11px;
}
</style>
```

top *right* *bottom* *left*

Applies the contained CSS to any span tags placed inside h1 tags that are contained by an element labelled with the '.myform' class
</td>
<td>The <code>.myform h1</code> class is used to apply the CSS code to h1 tags (only) found inside a container labelled with the 'myform' class. The <code>.myform h1>span</code> class extends the rule even further, by applying the CSS syntax to any span tag found inside a h1 container. The <code>display</code> property is used to set the h1 tag as a block element; a full-width element. The <code>border-bottom</code> property adds a solid, black border underneath the h1 element. The <code>margins</code> are set to negative values to position the title outside of the document's natural flow (outside of the web form container). The <code>.myform h1>span</code> class sets any span elements found as a 'block,' with font size '11px.'</td>
</tr>
</table>

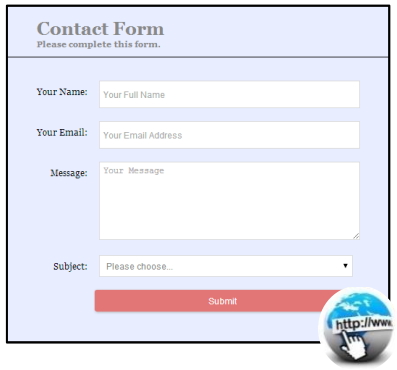
HTML Tips!

If an element is placed inside another element, the containing outside element is often referred to as a 'container'.


```
<style>
...
.myform label{display:block; margin:0px;}
.myform label>span{
        float:left;
        width:20%;
        text-align:right;
        padding-right:10px;
        margin-top:10px;
        color:#000;
}
.myform input, .myform textarea, .myform select{
        border:1px solid #DADADA;
        color:#888;
        height:30px;
        margin:2px 6px 16px;
        padding:3px 3px 3px 5px;
        width:70%;
        font-size:12px;
        line-height:15px;
}
.myform textarea{height:100px;}
</style>
```

Applies the contained CSS code to any input elements, textareas and selects (dropdown boxes) that are in a containing element labelled with the '.myform' class

The `myform label` class is used to apply the CSS code to all label tags found inside any container labelled 'myform'. The `.myform label>span` class extends the rule and applies the CSS code to all span tags found inside any label tags. Many of the properties here have already been explained previously; however, there are some new properties worth looking at. The `float` property positions any span elements to the left. The `width` property sets the width of the span element to 20% of the outside container; in this case, the label element. The `text-align` property sets the alignment of the text to the right; this neatens the alignment of the labels. The `.myform input, .myform textarea, .myform select` class applies the CSS code to all of the input types, textareas and select dropdowns found in a 'myform' container. The `line-height` property sets the height of each line in pixels. The `.myform textarea` class applies the CSS to all textareas found in any '.myform' container. The only property applied to textareas is the `height` property, which is set to 100 pixels.




```
<style>
...
.myform .button{
        display:block;
        background:#E27575;
        border:none;
        color:#FFF;
        box-shadow:1px 1px 5px #B6B6B6;
        border-radius:3px;
        cursor:pointer;
}
.myform .button:hover{background:#CF7A7A}
</style>
```

Rounds the border edges; not supported by all browser versions

The `.myform .button` class is used to apply CSS to any element labelled 'button', found inside any container labelled 'myform'. In this example, there is only one element labelled 'button' and that is the submit input. The `box-shadow` property adds an outside shadow, while the `border-radius` rounds the button's edges. The `cursor` property sets the cursor to a pointer when the button is hovered over. The `.myform .button:hover` class contains what is known as a pseudo-element; in other words, CSS code that is applied only when an element labelled 'button' is hovered over. In this example, the background colour will change when the submit button is hovered over.

Activity 3.5

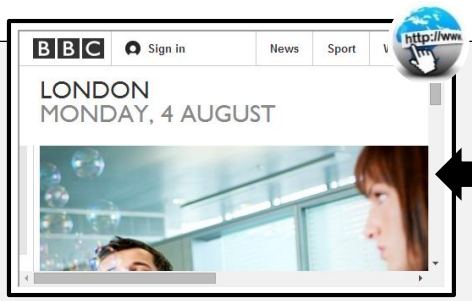
Create a webpage that includes the HTML form example (on page 8) and add the CSS code, from the table above, to the 'head' section of the HTML document. Add the CSS code in stages (each row from the table) and test the webpage, in a browser, after each step. Finally, experiment with the CSS code and make some changes; review the changes in a browser.

3.6 Practical: iFrames and Symbols

Using iFrames

Although not overly common in the modern-day web, the `<iframe>` element can be used to embed a web page inside another web page. Embedded web pages can either be local web pages or web pages from a different website altogether. This element is sometimes useful when comparing information found on one web page against another. To set the source (location) of the embedded web page the `src` attribute is set; this can be set to either a relative source (for example, /mypage.html) or an absolute source (for example, http://www.bbc.co.uk). The `width` and `height` attributes can be used to set the size of the iframe in pixels. Note that a sentence can be added between the opening and closing iframe tags; this sentence will be displayed if the user's browser does not support iframes.

```
<!DOCTYPE html>
<html>
<head>
	<title>My iFrame</title>
	<meta charset="UTF-8">
</head>
<body>
	<iframe src="http://www.bbc.co.uk" width="300" height="100">
		iFrame not supported by your browser.
	</iframe>
</body>
</html>
```



Activity 3.6

Create a webpage that contains an iframe; set an appropriate website as the source.

HTML Symbols

There are numerous HTML symbols available when writing content that is to be displayed in a web page; symbols can include mathematical operators, shapes, currency, technical symbols and arrows (mostly items that are not represented on a standard keyboard). An example of a symbol has been previously used; the character space symbol was used to add a blank space (` `) to the web page. Using HTML symbols can be important, especially if a website is targeting a global audience who may be using different character sets; using a HTML symbol will ensure the specific character is always displayed correctly. For example, as a web designer, it is safer to use this `&euro;` in the content, as opposed to €. Below are some other examples of HTML symbols:

Activity 3.7

Experiment with HTML symbols and apply them to a webpage document. Research three other symbols and share them with the group.

Symbol	Ref. Number	Entity Name	Description
™	&#8482;	&trade;	Trademark
©	&#169;	&copy;	Copyright Symbol
®	&#174;	&reg;	Registered Symbol
€	&#8364;	&euro;	Euro Symbol
←	&#8592;	&larr;	Leftwards Arrow
→	&#8594;	&rarr;	Rightwards Arrow
↓	&#8595;	&darr;	Downwards Arrow
↑	&#8593;	&uarr;	Upwards Arrow
∑	&#8721;	&sum;	Maths Sum
∞	&#8734;	&infin;	Infinity

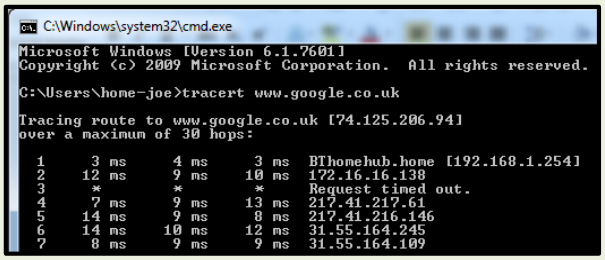
3.7 Theory: HTML URL Encodings

URL Composition

To finalise this chapter, some important HTML theories will be explained. Although HTML theory can sometimes be less interesting to learn (especially compared to practical work), it is important to develop an overall understanding of how the web's architecture functions; doing so will result in a broader and more competent skill set.

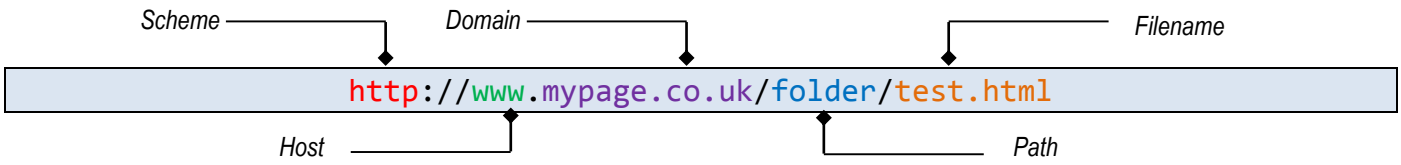
HTML Tips!

Have access to the command prompt (CMD)? If so, type 'tracert www.google.co.uk' into the command prompt and watch the computer find its way to the google website. Try the 'ping' command too!



URL is an acronym for Uniform Resource Locator, but for most people it is simply a web address (such as 'www.google.co.uk'). To a computer, however, a URL address is an Internet Protocol (IP) address, consisting of numbers separated by dots; for example, 74.125.206.94, which if this number was typed into a web address would direct the user to the Google homepage. Basically, an IP address and web address can be used interchangeably, except most people find an address easier to remember, as opposed to a list of numbers.

A URL is used to request a web page or file from a web server. A web URL consists of the following parts:



Feature	Description
Scheme	The scheme determines the type of Internet service that is to be used. The most common is HTTP (Hyper Text Transfer Protocol), which is used to collect standard web pages. However, HTTPS (Hyper Text Transfer Protocol Secure) is another service used to encrypt data when handling sensitive information; this can be seen in the browser's web address bar when completing online banking, etc. FTP (File Transfer Protocol) and FTPS (File Transfer Protocol Secure) are the non-encrypted and encrypted protocols used to transfer files between a web server and the client.
Host	The host determines the domain host to be used; the default host for the HTTP protocol is www. However, some organisations may have their own domain host.
Domain	The domain determines the domain name to be used; for example, mywebsite.com.
Path	The path determines the directory on the web server where the requested web page or file is stored. The forward slash is used to access the next directory. If there is no path specified, this means that the web page or file is stored in the root directory (upmost/first directory) of the website.
Filename	The filename determines the name of the document/resource to be located, by specifying both a name and file type. If no file is specified (for example, www.google.co.uk) the default file is shown; common default names include index.html, index.php and default.asp.

ASCII and URL Encodings

URLs are sent over the internet using ASCII. ASCII is an acronym short for the 'American Standard Code for Information Interchange' and is a character encoding scheme used (by computers) to represent English characters. This enables people to work with letters and symbols easily, including being able to send them to another computer (for example, to request a webpage). ASCII consists of 128 characters; numbers 0 to 9, letters A to Z, letters a to z, basic punctuation and other symbols. Each character has a designated number, for example, number 77 is the letter 'M' and number 209 is a lowercase 'm.' URLs sometimes contain characters that are not represented by the ASCII set and therefore the URL must be converted to a valid ASCII format before it can be sent over the internet. This is achieved by replacing non-ASCII representations with a '%' symbol, followed by two hexadecimal (letters A to F and numbers 0 to 9) digits. For example, the German word 'Rückstöße' ('rebound' in English) contains three non-ASCII characters and thus would need to be converted by the browser to this; 'R%C3%BCckst%C3%B6%C3%9Fe.'

Chapter Summary

- ✓ A form, in HTML, is defined using the opening and closing `<form></form>` tags; form elements are generally defined here.
- ✓ To create a text box this syntax is used: `<input type="text" name="firstname">`. The `name` attribute enables the data to be identified, once it is submitted.
- ✓ To create a radio button this syntax is used: `<input type="radio" name="colours" value="red">`. The `value` attribute represents the data to be submitted if a particular radio button is selected. When using radio buttons, use the `name` attribute to group radio elements together by specifying the same name value.
- ✓ Checkboxes are similar to radio buttons, except they allow multiple inputs to be selected. The syntax for a checkbox is `<input type="checkbox" name="city" value="london">`.
- ✓ Dropdown lists are defined between the opening and closing `<select></select>` tags. Each option in the list is defined using this syntax `<option value="fender">Fender</option>`.
- ✓ Forms are submitted by adding a submit button to the form and setting both the `action` and `method` attributes. A button is added using this syntax `<input type="submit">`. The `action` attribute can be used to specify the file to execute on the server and the `method` attribute specifies how the information is sent; usually either 'get' or 'post'. Both attributes are set in the opening form tag; for example, `<form action="submitted.php" method="get">`.
- ✓ CSS can be applied to forms to improve their presentation greatly. Remember that classes can be defined to apply CSS code to labelled elements only. They can be defined further still by specifying specific elements, in the labelled container, that they apply to; for example, '`.myform h1>span`' would only apply CSS to span tags inside h1 tags, found in a labelled 'myform' container.
- ✓ The opening and closing `<iframe></iframe>` tags can be used to embed a webpage inside another web page. Use the `src` attribute to specify the embedded website.
- ✓ HTML symbols can be used to define non-standard symbols in the browser, to ensure they are displayed correctly all over the world. For example, this '™' could be used to show the trade mark logo ™ globally.
- ✓ A URL (Uniform Resource Locator) is a web address. A web address is made up of a scheme, host, domain, path and filename. Non-ASCII characters are encoded before being sent over the Internet.
- ✓ Computers use IP (Internet Protocol) addresses to locate resources on the Internet.

