Community Experience Distilled

Thinking in HTML

Learn to write intuitive HTML and build your sites on solid foundations

Aravind Shenoy

[PACKT] PUBLISHING

Thinking in HTML

Learn to write intuitive HTML and build your sites on solid foundations

Aravind Shenoy

[PACKT]
PUBLISHING
BIRMINGHAM - MUMBAI

Thinking in HTML

First published: September 2014

Production reference: 1120914

Published by Packt Publishing Ltd.
Livery Place
35 Livery Street
Birmingham B3 2PB, UK.

www.packtpub.com

Credits

Author
Aravind Shenoy

Reviewer
Anirudh Prabhu

Commissioning Editor
Edward Gordon

Content Development Editor
Neeshma Ramakrishnan

Technical Editor
Dennis John

Copy Editor
Karuna Narayanan

Proofreaders
Simran Bhogal
Maria Gould

Graphics
Adonia Jones

Production Coordinator
Adonia Jones

Cover Work
Adonia Jones

About the Author

Aravind Shenoy is an in-house author at Packt Publishing. An engineering graduate from the Manipal Institute of Technology, his core interests are technical writing, web designing, and software testing. He was born and raised in Mumbai, India, and resides there. He is a music buff and loves listening to Oasis, R.E.M, The Doors, Dire Straits, and Eminem. Rock 'n' Roll and Rap rule his playlists. He is a simple person by nature and believes that we are all here to have a good time, not a long time. After all, the most important thing is to be happy.

About the Reviewer

Anirudh Prabhu is a software engineer at Xoriant Corporation with 4 years' experience in web designing and development. He is responsible for JavaScript development and maintenance in his projects. His areas of expertise are HTML, CSS, JavaScript, and jQuery. When not working, Anirudh loves reading, listening to music, and photography.

He has completed his Master's degree in Information Technology. He has also reviewed some titles related to JavaScript and CSS for Packt Publishing.

Overview

This book provides you with an intermediate knowledge of HTML. Instead of wandering through loads of theory, we will understand HTML practically so that we can understand the markup of a web page. We have used Notepad for the examples in this book. Alternatively, you can also use Notepad++ or any advanced editor. All you need to do is copy the code and paste it into Notepad. Upon execution, you will get the output as depicted in the screenshots. Screenshots are provided for each piece of sample code.

Coding improves with practice. The examples in this book are compatible with almost every modern browser such as Mozilla Firefox or Google Chrome, to name a few. Instead of using the verbatim code, you can modify the code and see the change in the output, thereby understanding the subtle nuances of HTML. By the end of the book, with practice, you can achieve better things as you get to grips with HTML.

Thinking in HTML

HTML originated from a prototype created by Tim Berners-Lee in 1992. Berners-Lee felt that there was a possibility of linking documents together by the use of hypertext, and the concept of HTML evolved. The drawback was that the commercial hypertext packages available at that time, such as Zog and Intermedia, were customized to suit different types of computers and were too ambiguous in nature.

Berners-Lee developed **HyperText Markup Language** (**HTML**), and in conjunction, he developed a protocol to access text from other documents via hyperlinks. The protocol was called **HTTP**, and this paved the way for the future. HTML itself was derived from a markup language called **Standard Generalized Markup Language** (**SGML**).

Standardization being an ongoing process, modifications were constantly made and versions were released accordingly. The various versions of HTML that have been released are as follows:

- HTML 2.0 (November 1995)
- HTML 3.2 (January 1997)
- HTML 4.0 (December 1997)
- HTML 4.01 (December 1999)
- HTML5, which is widely used, but is still in the development stage

Another breakthrough in the field was the introduction of CSS along with HTML 4.0. Prior to the introduction of CSS, web designers and developers used HTML for formatting purposes. Formatting and styling a web page using HTML defeats the purpose of HTML, because HTML elements and attributes must only define the structure of the web page. The purpose of CSS was to separate styling out from structural markup. With the introduction of CSS, we could separate presentation from content. As a result, formatting could be separated from the HTML document and stored in a separate file, which could then be included in the document using a `link` tag.

Instead of wandering through loads of theory, we will understand HTML practically with the help of code examples in this book. We have used Notepad for the examples in this book. Alternatively, you can also use Notepad++ or any advanced editor such as Adobe Dreamweaver. To execute the code, all you need to do is copy the code and paste it into Notepad. Upon execution, you will get the output as depicted in the screenshots. Coding gets better with practice. Instead of using the verbatim code in the book, you can alter the code and see the change in the output to understand HTML better.

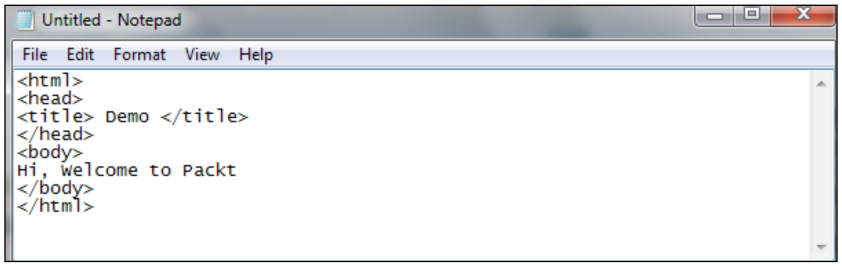
At the time of writing this book, HTML 4.0 is supported by most of the browsers. This means that you can execute the code in major browsers such as Internet Explorer, Google Chrome, and Mozilla Firefox. However, when we move on to HTML5, this might not be the case. HTML5 is still in the development stage and is expected to be released later this year. After HTML5 is accepted as the norm, it will be standardized and supported by all major browsers. Browser support for the forms and attributes in HTML5 is explained in detail at `http://www.wufoo.com/html5/`. This also indicates which browsers are supporting which features and to what degree.

Getting started with HTML

HTML doesn't need any special editor to code. You can use Notepad if you are using Windows or TextEdit if you're using a Mac.

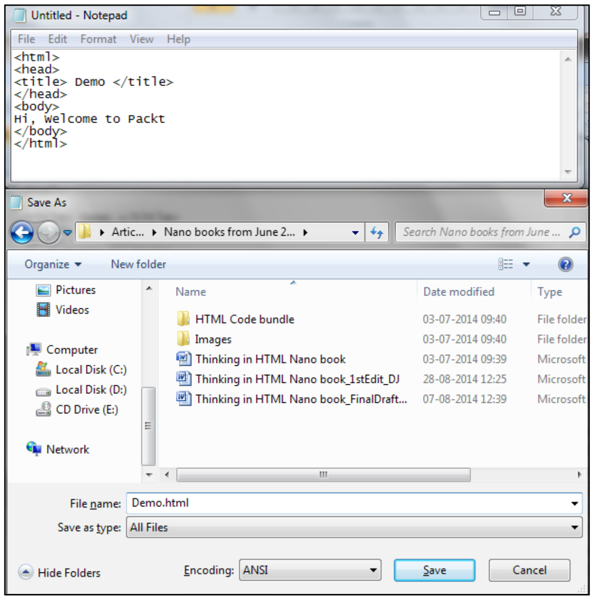
Let's have a look at the following instructions to understand the procedure, which is also depicted in the screenshots accompanying it:

1. Open Notepad on your Windows PC. Write or paste your code in the Notepad file:



```
<html>
<head>
<title> Demo </title>
</head>
<body>
Hi, Welcome to Packt
</body>
</html>
```

2. Save the file in the `(filename).html` format. In this example, we will be using `Demo` as the filename. Make sure that **Save as type** displays **All Files**:



```
<html>
<head>
<title> Demo </title>
</head>
<body>
Hi, welcome to Packt
</body>
</html>
```

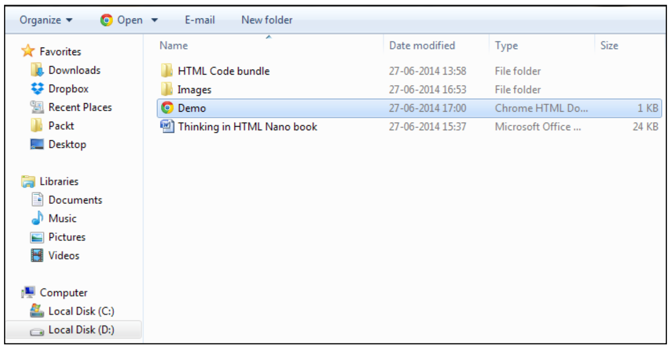
File name: Demo.html

Save as type: All Files

Encoding: ANSI

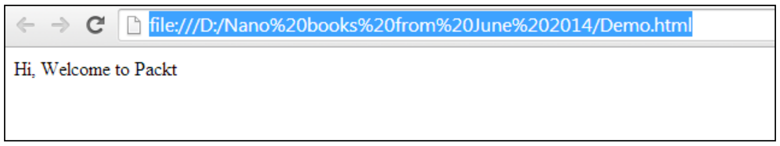
Name	Date modified	Type
HTML Code bundle	03-07-2014 09:40	File folder
Images	03-07-2014 09:40	File folder
Thinking in HTML Nano book	03-07-2014 09:39	Microsoft
Thinking in HTML Nano book_1stEdit_DJ	28-08-2014 12:25	Microsoft
Thinking in HTML Nano book_FinalDraft...	07-08-2014 12:39	Microsoft

3. When you go the location where you saved the file, you will see the `Demo.html` file:



Name	Date modified	Type	Size
HTML Code bundle	27-06-2014 13:58	File folder	
Images	27-06-2014 16:53	File folder	
Demo	27-06-2014 17:00	Chrome HTML Do...	1 KB
Thinking in HTML Nano book	27-06-2014 15:37	Microsoft Office ...	24 KB

4. After opening the `Demo.html` file, you will see the following screen. Voila! You have executed your first piece of HTML code.



file:///D:/Nano%20books%20from%20June%202014/Demo.html

Hi, Welcome to Packt

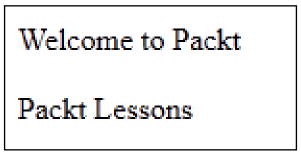
Instead of wasting time on theoretical discussions, let's get straight to coding so that you get to grips with HTML.

> The tutorials on HTML on the Internet usually contain a lot of clutter, where you have redundant tags for formatting and styling. However, the correct procedure is to keep the presentation separate from the content. Therefore, all the presentational HTML elements and attributes were replaced by CSS to provide versatility and better accessibility. Thus, we will gain an understanding of the HTML essentials that are used frequently, keeping in sync with the times. Also, to learn CSS, you can refer to my free Kindle version book, *Thinking on CSS*, *Packt Publishing*, on Amazon at: `http://www.amazon.com/Thinking-CSS-Aravind-Shenoy-ebook/dp/B00JUI6LQU/ref=sr_1_1?ie=UTF8&qid=1405928421&sr=8-1&keywords=thinking+in+css`.

HTML is a markup language and includes a set of markup tags. Basically, an HTML document has HTML tags and content. Let's have a look at the following code and understand its structure:

```
<!DOCTYPE html>
<html>
<body>
<p> Welcome to Packt </p>
<p> Packt Lessons </p>
</body>
</html>
```

In the preceding code sample, `<!Doctype html>` is a declaration and lets the browser know that you are using HTML5. In HTML, **tags** are keywords surrounded by angular brackets (< >). They usually come in pairs. For example, `<html>` is the opening tag and `</html>` is the closing tag. Similarly, `<p>` is the opening tag and `</p>` is the closing tag. The `<html>` tag tells the browser that everything between it and the closing `</html>` tag is an HTML document. The tags between `<html>` and `</html>` are also called **elements**. The tags do not get displayed in the browser; instead, they determine the manner in which content is presented to the user. The text between the `<body>` and `</body>` tags is the content that is presented to the user. The `<p>` tag is used to define paragraphs where you want to break up two streams of information. However, there is an exception to the rule; some tags such as `<br>` and `<hr>` do not have a closing tag. Let's look at the output of the preceding code:


Welcome to Packt

Packt Lessons

We will gradually increase the level of difficulty with each code example so that you understand the concepts in a lucid and practical manner.

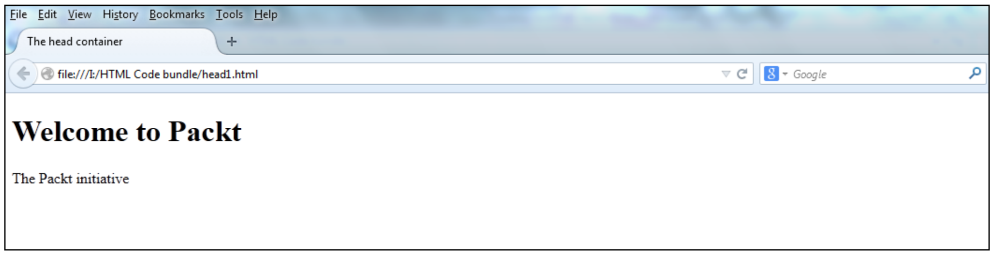
The <head> element

The `<head>` and `</head>` tags appear before the `<body>` element. The `<head>` element gives you information about the web page, and the information doesn't get displayed in the browser. The `<head>` tag is used to include script files responsible for styling and interactivity of the page. It contains tags such as `<title>`, `<style>`, `<meta>`, and `<script>`, to name a few.

Let's look at the following code example to understand the `<head>` element better:

```
<!DOCTYPE html>
<html>
<head>
<title> The head container </title>
<link rel="stylesheet" type="text/css" href="aravind.css" />
<meta name="author" content="Aravind Shenoy" />
</head>
<body>
<h1> Welcome to Packt </h1>
<p> The Packt initiative </p>
</body>
</html>
```

The output of the code will be as follows:


File Edit View History Bookmarks Tools Help
The head container
file:///I:/HTML Code bundle/head1.html
Google

Welcome to Packt

The Packt initiative

If you observe the output, the content between the `<title>` tags (that is `The head container` in this example) gets displayed at the top of the browser window in the tab. The `<title>` tag is an imperative aspect of HTML coding, as search engines will use it as the primary means of cataloging sites. The content between the `<title>` tags is displayed as the link text in search engine results and is also used to bookmark your site. The `<meta>` tag contains information about the document, for example, the keywords in it and a description, which is quite useful for search applications. The `<style>` tag is used to include CSS rules inside your document for styling purposes. The `<script>` tag helps include JavaScript code or a jQuery link for your web page.

Although we have defined the `<style>` element as `aravind.css`, we have not used styling to avoid information overload. Also, the `<script>` element is not included to reduce complexity. We will see some styling in some of the code examples in this book, but we need to stick to the purpose of this book, which is to help you understand HTML better.

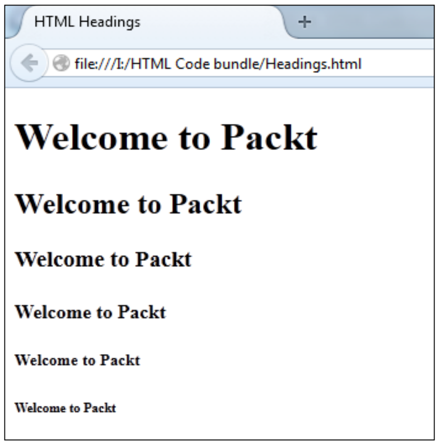
Headings

Heading tags are used to differentiate the heading of a web page from the rest of the content. There is a hierarchy, which means the content in `<h1>` is the most important, and that in `<h6>` is the least important one.

Let's now look at a code example to understand it better:

```
<!DOCTYPE html>
<html>
<head>
<title> HTML Headings </title>
</head>
<body>
<h1>Welcome to Packt</h1>
<h2>Welcome to Packt</h2>
<h3>Welcome to Packt</h3>
<h4>Welcome to Packt</h4>
<h5>Welcome to Packt</h5>
<h6> Welcome to Packt </h6>
</body>
</html>
```

The output of the code on execution will be as follows:



Welcome to Packt

Welcome to Packt

Welcome to Packt

Welcome to Packt

Welcome to Packt

Welcome to Packt

If you observe the output, you can see that the first heading is prominent and the size decreases as you go down in the hierarchy.

Headings are useful for **Search Engine Optimization** (**SEO**) purposes. Heading tags help describe what the web page is all about, and the search engine spiders look for specific information from the heading tags, along with the content. The hierarchy has to be maintained; this means that you should not jump from `<h1>` to `<h3>` bypassing `<h2>`. Heading tags help the website users look for relevant information and should not be used for formatting purposes, as styling should be done only with CSS, barring a few exceptions.

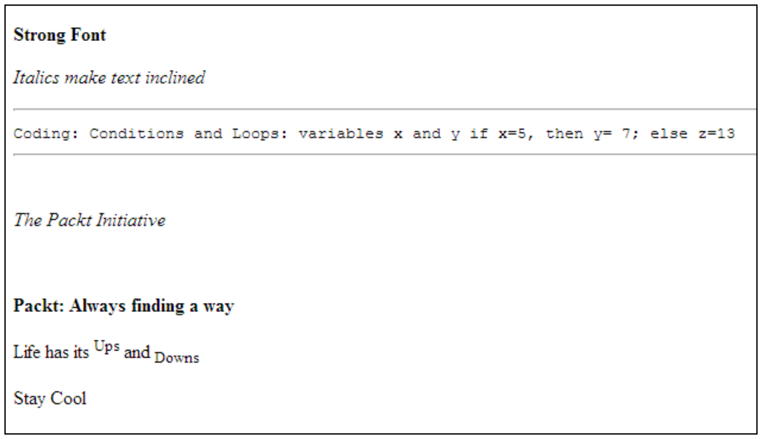
Formatting

The correct procedure for styling is to use CSS; in rare cases, where there is no other alternative, we do use HTML for formatting purposes.

Let's look at the following code to understand this better:

```
<!DOCTYPE html>
<html>
<head>
<title> Formatting in HTML </title>
</head>
<body>
<!-- Comments help aspiring newbies  -->
<p> <strong> Strong Font </strong> </p>
<p> <i>Italics make text inclined </i> </p>
<hr>
<code>
Coding: Conditions and Loops: variables x and y
if x=5, then y= 7; else z=13
</code>
<hr><br>
<p><em>The Packt Initiative </em><br></p>
<br>
<p><b> Packt: Always finding a way </b> </p>
<p>Life has its <sup> Ups </sup> and <sub> Downs </sub> </p>
Stay Cool
</body>
</html>
```

The output of the code will be as follows:



Strong Font

Italics make text inclined

```
Coding: Conditions and Loops: variables x and y if x=5, then y= 7; else z=13
```

The Packt Initiative

Packt: Always finding a way

Life has its $^{\text{Ups}}$ and $_{\text{Downs}}$

Stay Cool

From the output, you can see that `<strong>` is used to denote something important, whereas `<i>` is used for italic fonts. The `<code>` tag defines the computer code text, while `<em>` is used to describe something significant. The `<b>` tag is used for bold text. The `<sup>` and `<sub>` tags are used to define superscripts and subscripts. Superscripts and subscripts are basically those numbers or words that are smaller than the normal line type and appear slightly above and below it. Including comments is a good practice that ensures easy code readability. In HTML, comments are addressed between angular brackets with an exclamation mark, as shown in the following code:

```
<!-- Comments help aspiring newbies  -->
```

Comments are not displayed in the browser.

Attributes

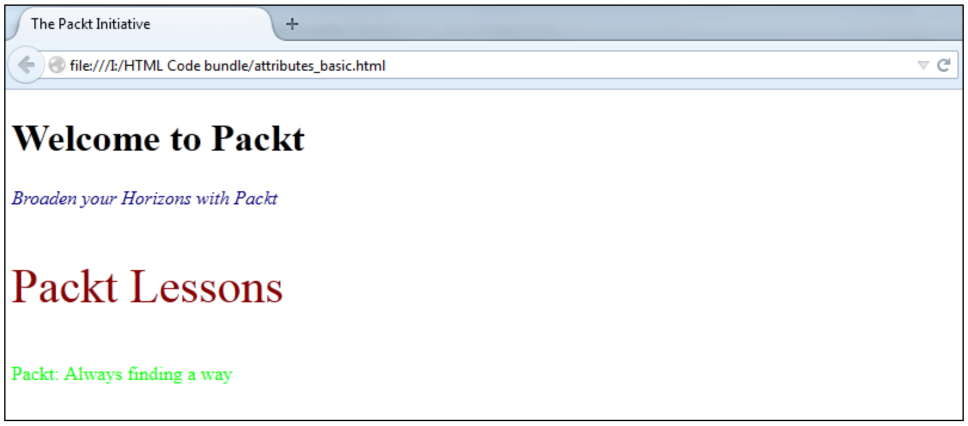
HTML elements can have attributes that provide additional information about an element. They come in `<name: value>` pairs and are always a part of the opening tag. The value of the attribute must be enclosed in quotes for clarity (though it is not always necessary).

Let's look at the following code to understand it better:

```
<!DOCTYPE html>
<html>
<head>
<title> The Packt Initiative </title>
<style>
#packtpub {
      color:navy;
      font-style: italic;
       }

.learn     {   color: maroon;
      font-size: 40px;
       }
</style>
</head>
<body>
<h1> Welcome to Packt </h1>
<p id="packtpub"> Broaden your Horizons with Packt </p>
<p class="learn"> Packt Lessons </p>
<p style="color:lime"> Packt: Always finding a way </p>
</body>
</html>
```

The output of the code on execution will be as follows:



Welcome to Packt

Broaden your Horizons with Packt

Packt Lessons

Packt: Always finding a way

If you observe the code and the output, you can see that we used both inline and internal styles. We defined the first paragraph `<p>` tag with a `"packtpub"` ID and the second paragraph tag with a `"learn"` class. Now, when you check the `<style>` element in the `<head>` section of the code, we designated values to the `"packtpub"` ID and the `"learn"` class. For the third paragraph tag, we used an inline style with the `"lime"` color associated with it. We will get to inline and block elements later on in this book; for now, let's understand the code. The `id` attribute is unique and can be used for a single element, whereas a `class` attribute is used to style multiple elements. Also, an element can have multiple classes. The `id` attribute (`packtpub`) in the preceding code was assigned the `navy` color and `italic` font style as values, whereas the `class` attribute (`learn`) was assigned the `maroon` color and a font size of `40`. Thus, you can define the characteristics of an element with attributes.

Images

Images can be inserted in the document with the help of the `<img>` tag. In the `<img>` tag, we use the `src` attribute that informs the browser about the image source.

Let's look at the following code to understand it better:

```
<!DOCTYPE html>
<html>
<head>
<title> Images in HTML </title>
</head>
<body>
<h1> Image in an HTML doc </h1>
<img src="Image-Packtpub.png" width ="200" height = "210"
alt="PacktLib">
</body>
</html>
```

The output of the code on execution will be as follows:

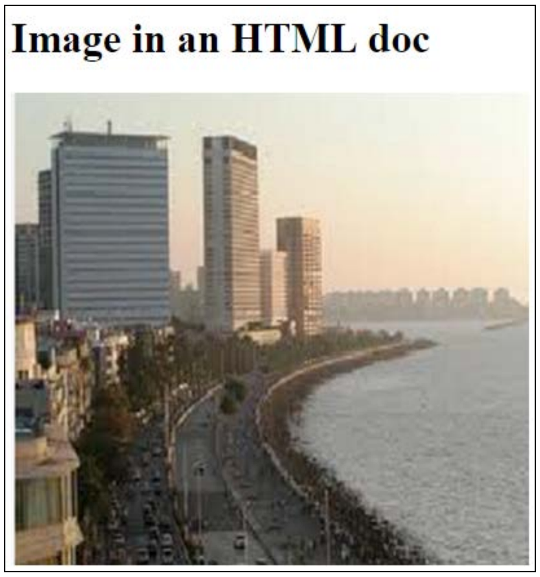


Image in an HTML doc

If you observe the code and the output, you can see that we assigned the width and height of the image as 200 and 210, respectively. Another aspect to consider is that the `<img>` element does not have a closing tag. We also assigned the `"PacktLib"` value to the `alt` attribute. The `alt` attribute specifies the text of the image if the image doesn't get displayed due to reasons such as incorrect syntax or a slow connection.

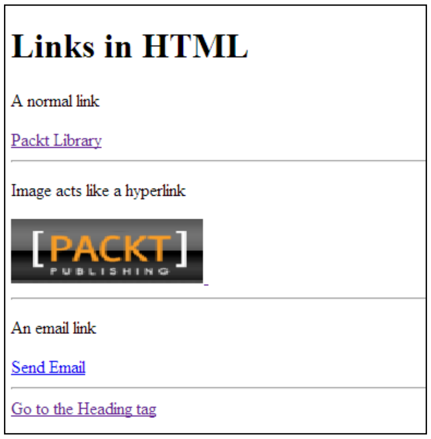
Links

The anchor tag (`<a>`) in HTML is used to define hyperlinks. A **hyperlink** can be an element that will take you to a specific location, be it a website or any other place in the same document or any other document. We use the `href` attribute with the `<a>` tag to specify the link destination.

Let's look at the following code to understand it better:

```
<!DOCTYPE html>
<html>
<head>
<title> Links </title>
</head>
<body>
<h1 id="packt"> Links in HTML </h1>
<p> A normal link </p>
<a href="http://packtlib.packtpub.com">Packt Library</a>
<br><hr>
<p> Image acts like a hyperlink </p>
<a href="http://www.packtpub.com/" >
<img src="Packt-image2.png" alt="Packt" />
</a>
<br><hr>
<p> An email link </p>
<a href="mailto:xyz@packt.com">Send Email</a>
<hr>
<a href="#packt"> Go to the Heading tag </a>
</body>
</html>
```

The output of the code on execution will be as follows:


Links in HTML

A normal link

Packt Library

Image acts like a hyperlink

An email link

Send Email

Go to the Heading tag

If you observe the code, you can see that we assigned the URL of the Packt library as the value of the `href` attribute in the first `<a>` tag. When you click on the **Packt Library** hyperlink, you will be directed to `http://packtlib.packtpub.com/`, as we defined it as the link's destination.

In the second `<a>` tag, we defined an image that can be used as a hyperlink. When you click on the image, you will get directed to `http://www.packtpub.com/`, which is the destination of the link.

In the third `<a>` tag, we defined a `mailto:xyz@packt.com` value for the `href` attribute; this helps your website users send an e-mail. If you add your e-mail to the website, spammers might send copious amount of e-mails using that e-mail address. Thus, to prevent this, you can keep a link that will help your website users e-mail you; however, the users need to have Mozilla Thunderbird, Outlook, or any other e-mail client configured to use this feature. In the previous code example, you can see that we used the **Send Email** link for the same purpose.

As mentioned earlier, you can use links to go to a specific document or any location within the same document. If you observe the code, we have assigned a `"packt"` ID to the `<h1>` tag. In the final `<a>` tag, the `href` attribute is tagged with the `"packt"` ID, which tells the browser to go the location where the ID was defined, that is, the `<h1>` heading tag.

Block and inline elements

HTML elements are usually either block or inline elements. **Block** elements can have a margin and/or padding. They start or end with a new line in the web page. They expand naturally to fill the container in the absence of any specifically mentioned height or width.

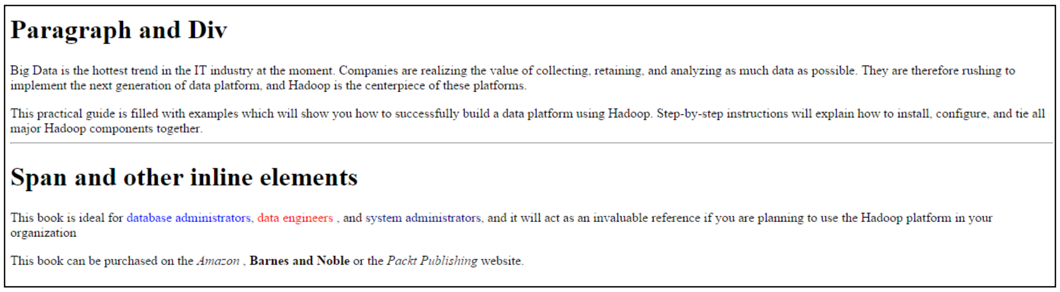
Inline elements do not start or end with a new line, as they flows along with the text. An inline element consumes only as much space as necessary.

Let's look at the following code to understand it better:

```
<!Doctype html>
<html>
<head>
<title> Block and inline elements </title>
</head>
<body>
<h1> Paragraph and Div </h1>
<p>
Big Data is the hottest trend in the IT industry at the moment.
Companies are realizing the value of collecting, retaining, and
analyzing as much data as possible. They are therefore rushing to
implement the next generation of data platform, and Hadoop is the
centerpiece of these platforms.
</p>
```

```
<div>
This practical guide is filled with examples which will show you how
to successfully build a data platform using Hadoop. Step-by-step
instructions will explain how to install, configure, and tie all major
Hadoop components together.
</div>
<br><hr>
<h1> Span and other inline elements </h1>
<p>
This book is ideal for <span style = "color:blue"> database
administrators</span>, <span style=" color: red"> data engineers </
span>, and <span style = "color: navy"> system administrators</span>,
and it will act as an invaluable reference if you are planning to use
the Hadoop platform in your organization
</p>
<p>This book can be purchased on the <em> Amazon </em>, <strong>
Barnes and Noble </strong> or the <i>Packt Publishing</i> website.
</p>
</body>
</html>
```

The output of the code will be as follows:



Paragraph and Div

Big Data is the hottest trend in the IT industry at the moment. Companies are realizing the value of collecting, retaining, and analyzing as much data as possible. They are therefore rushing to implement the next generation of data platform, and Hadoop is the centerpiece of these platforms.

This practical guide is filled with examples which will show you how to successfully build a data platform using Hadoop. Step-by-step instructions will explain how to install, configure, and tie all major Hadoop components together.

Span and other inline elements

This book is ideal for database administrators, data engineers , and system administrators, and it will act as an invaluable reference if you are planning to use the Hadoop platform in your organization

This book can be purchased on the *Amazon* , **Barnes and Noble** or the *Packt Publishing* website.

If you observe the code and the output, you can find various examples of block and inline elements. In the code, `<p>`, `<h1>`, and `<div>` are the block elements, whereas `<span>`, `<em>`, `<strong>`, and `<i>` are the inline elements.

As mentioned earlier, `<p>` is the paragraph element and `<h1>` is the heading element. Coming to `<div>`, it is used to define generic containers within the content of a page. The `<div>` element is a block element used for sectioning, and you can also assign `id`, `class`, and other attributes to give the `<div>` element a semantic meaning. The `<span>` element is an inline element and is used to group elements for styling purposes or because they share attribute values. The `<span>` element is to be used only if there is no other alternative semantic element.

In the code, you can see that we used the `<span>` element for different parts of the text for styling purposes. We also used other inline elements such as `<em>`, `<strong>`, and `<i>` to emphasize text, make it bold, and italicize it, respectively.

Though `<br>` is used to introduce a line break, it is an inline element as it is not displayed on its own, whereas `<hr>` is a block element used to create a horizontal line.

The `<a>`, `<img>`, and `<b>` elements are some of the other inline elements used frequently in HTML.

Lists

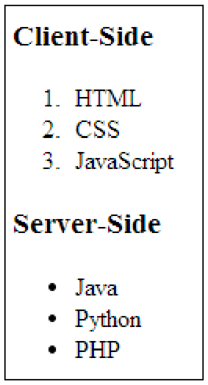
You can list your items and menu in the form of ordered and unordered lists. The unordered lists use bullets for the menu, whereas the ordered lists use numbered bullets for the menu.

Let's look at the following code to understand it better:

```
<!DOCTYPE html>
<html>
<head>
<title> Lists in HTML </title>
</head>
<body>
<h3> Client-Side </h3>
<ol>
  <li> HTML </li>
  <li> CSS </li>
  <li> JavaScript </li>
</ol>

<h3> Server-Side</h3>
<ul>
  <li> Java </li>
  <li> Python </li>
  <li> PHP </li>
</ul>
</body>
</html>
```

The output of the code on execution will be as follows:



Client-Side

1. HTML
2. CSS
3. JavaScript

Server-Side

- Java
- Python
- PHP

If you observe the code and the output, the items listed under `<ol>` related to the **Client-Side** heading have numbered bullets, whereas the items listed under `<ul>` related to the **Server-Side** heading are in normal bullets. Each item is listed between the `<li>` and `</li>` tags.

Forms

Forms are used to collect user input such as name, age, gender, and e-mail address by means of text, radio buttons, and checkboxes, to name a few. The input is then passed for processing on the server side. The most commonly used form attributes are `name`, `action`, `target`, and `enctype` (read as encryption).

Form input types come in various types such as text fields, password, radio buttons, checkboxes, and submit buttons. Between the `<form>` tags, you can specify the `action` attribute that tells the form where its contents will be sent to (mainly, a server-side script for processing). We use the `method` attribute that informs the form how the data is going to be sent; usually, it is **GET** by default, but you can also use **POST** and other methods.

An example of the GET method is the search engine results, where you can see the information you entered in your search box displayed in the web address field. The POST method is used for sensitive data where the form data is not visible to the user, thereby ensuring a high level of security.

Let's look at the following code to understand it better:

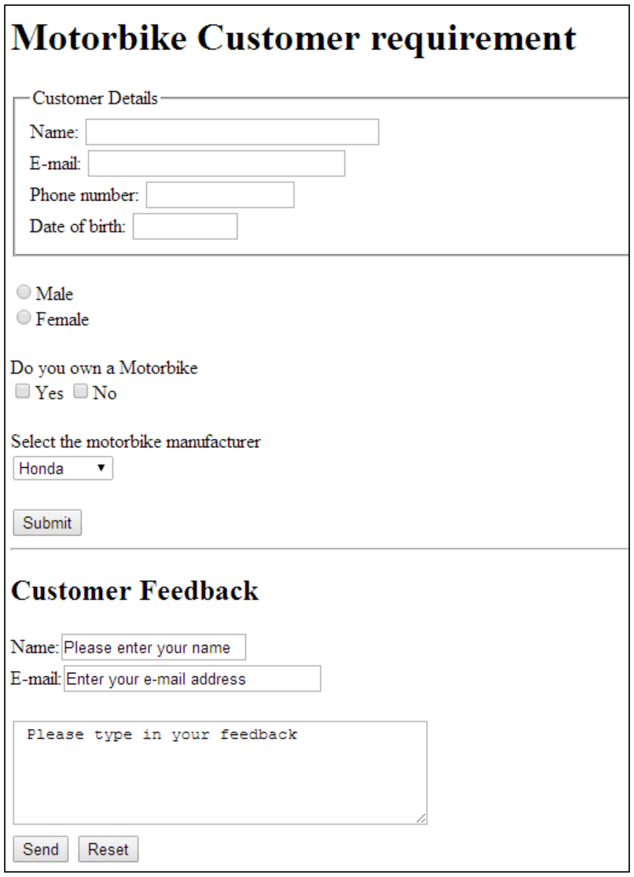
```
<!DOCTYPE html>
<html>
<body>
<h1> Motorbike Customer requirement </h1>
```

```
<form action="backscript">
<fieldset>
<legend> Customer Details </legend>
Name: <input type="text" size="35"><br>
E-mail: <input type="text" size="30"><br>
Phone number: <input type="text" size="15"><br>
Date of birth: <input type="text" size="9">
</fieldset>
<br>
<input type="radio" name="sex" value="male">Male<br>
<input type="radio" name="sex" value="female">Female
<br><br>
Do you own a Motorbike <br>
<input type="checkbox" name="motor" value="bike">Yes
<input type="checkbox" name="motor" value="bike">No
<br><br>
Select the motorbike manufacturer
<br>
<select name="bike">
<option value="honda"> Honda </option>
<option value="ducati"> Ducati </option>
<option value="kawasaki"> Kawasaki </option>
<option value="yamaha"> Yamaha </option>
<option value="aprilia"> Aprilia</option>
</select>
<br><br>
<input type="button" value="Submit">
</form>
<hr>
<h2>Customer Feedback</h2>

<form action="MAILTO:abcxyz@demo.com" method="post" enctype="text/
plain">
Name:<input type="text" name="name" value="Please enter your
name"><br>
E-mail:<input type="text" name="mail" size="30" value="Enter your
e-mail address"><br><br>
<textarea cols="40" rows="5" name="Comments:::"> Please type in your
feedback  </textarea>

<br>
<input type="submit" value="Send">
<input type="reset" value="Reset">
</form>
</body>
</html>
```

The output of the code on execution will be as follows:


Motorbike Customer requirement

Customer Details

Name:

E-mail:

Phone number:

Date of birth:

Male

Female

Do you own a Motorbike

Yes No

Select the motorbike manufacturer

Honda

Submit

Customer Feedback

Name: Please enter your name

E-mail: Enter your e-mail address

Please type in your feedback

Send Reset

If you observe the code, we used the `<fieldset>` and `<legend>` tags to group related form elements. This increases the ease of use for your website visitor, as it organizes related elements in a concise way. Then, we defined the various fields between the `<fieldset>` tags, where we mention that the input type is text. The size for these single text fields is also explained here. Similarly, we defined the radio buttons for **Male** or **Female**, where users can select their gender. You can select only one radio button at a time. Later on, we explained the checkbox input type where we can select multiple checkboxes using this feature. We used the `select` and `option` attributes to create a drop-down menu where you can select the motorbike manufacturer. Finally, we defined the **Submit** button to submit the form.

For this form, we used `action="backscript"` as we do not have a server-side script (such as a PHP script), which is not in the scope of this book.

On the same web page, we defined another form. Let's look at the following line of code from the code sample:

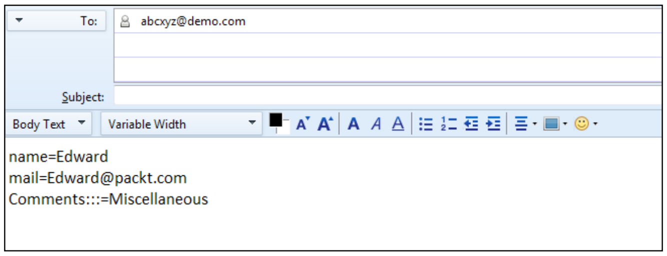
```
<form action="MAILTO:abcxyz@demo.com" method="post" enctype="text/
plain">
```

We defined the action where the feedback will be e-mailed to the concerned person whose e-mail address is `abcxyz@demo.com`. We used the `POST` method and plain text as the encryption type. We proceeded to define the input type as text and assigned a value that will be displayed in the text boxes to help the user know what information needs to be inserted into the boxes.

Later on, we defined a text area for the feedback field as we have a multiline input. We assigned 5 rows and 40 columns for the text area in this example. Thereafter, we created the **Send** and **Reset** buttons. The **Reset** button will clear the information you entered in the fields. Suppose we enter `Edward` as the name, `Edward@packt.com` as the e-mail address, and `Miscellaneous` in the comments field.

[Remember that you need to have an e-mail client such as Microsoft Outlook or Mozilla Thunderbird for this feature. In this example, we used Mozilla Thunderbird as the e-mail client).]

When you click on **Send**, you can see the following output:



```
name=Edward
mail=Edward@packt.com
Comments:::=Miscellaneous
```

HTML5 – the future is here

HTML5 is still in the development stage, and the standardized version is expected to be released later in 2014. HTML5 will soon be accepted as the benchmark. It differs from the previous versions of HTML in various ways. HTML5 works with modern browsers and also offers backward compatibility. It has a lot of new features that will eventually change the approach in designing websites. The document type declaration in HTML5 is very simple. All we need to do is type `<!DOCTYPE html>` so that the browser can recognize that we will be working with HTML5.

If we add `<!DOCTYPE html>` to the code, do you think that it would become HTML5 code? Our answer would be a firm "No". HTML5 is not just some kind of a document type. It has a lot of new elements, attributes, input types, and so on. It is a whole set of rules that enables the developer to define a web page in an impressive manner.

In this book, even in the previous examples, we used `<!DOCTYPE html>`, as it is a good practice that is going to be adapted as a standard in the future of web designing.

Easier and faster syntax

The main benefit of HTML5 is to build a web page using cleaner code. **Sectioning** is an important part of HTML5. Sectioning elements are used in HTML5 to quickly build a semantic web page. An imperative aspect of semantics is to increase accessibility.

The elements used in HTML5 for sectioning are as follows:

- `<header>`
- `<footer>`
- `<nav>`
- `<article>`
- `<aside>`

header and footer

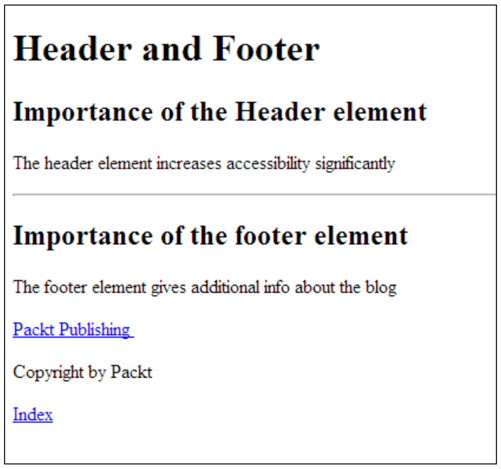
Prior to HTML5, there was no specific way to tag content based on its definition. Earlier, the `<div>` tag was used extensively along with headings to create a header for the web page. However, now we have the `header` tag. A `header` tag might contain headings nested in it, but this is not a necessity. Searching for stuff on the Internet is possible due to search engines such as Google Search and Bing. These search engines, with the help of crawlers, can access a web page content based on the elements that contain them. The `header` element is very important as it provides more accessibility than other elements, such as the `footer` tag. The search engine finds the content in the `header` tag quite easily.

The `footer` element is used to denote aspects such as the author's name, contact information, and copyright patent, to mention a few. The `footer` element allows a programmer to give a general idea about the website. However, footers can be used more than once in a document. An article can have its own independent footer in addition to another `footer` element in the source code for a specific web page. It completes the information in the document and makes it more meaningful.

Let's look at the following code to understand it better:

```
<!DOCTYPE html>
<html>
<body>
<header>
<h1> Header and Footer </h1>
<h2> Importance of the Header element </h2>
</header>
<p> The header element increases accessibility significantly </p>
<hr>
<h2> Importance of the footer element </h2>
<p> The footer element gives additional info about the blog </p>
<footer>
<a href="https://www.packtpub.com"> Packt Publishing </a>
<br>
<p> Copyright by Packt </p>
<a HREF="/index.html">Index</a>
</footer>
</body>
</html>
```

The output of the code on execution will be as follows:



```
Header and Footer

Importance of the Header element

The header element increases accessibility significantly
________________________________________________________
Importance of the footer element

The footer element gives additional info about the blog

Packt Publishing

Copyright by Packt

Index
```

If you observe the output, you can see that we defined the `header` and `footer` elements. Additional information such as the copyright information and the `index.html` link has been defined in the `<footer>` tag.

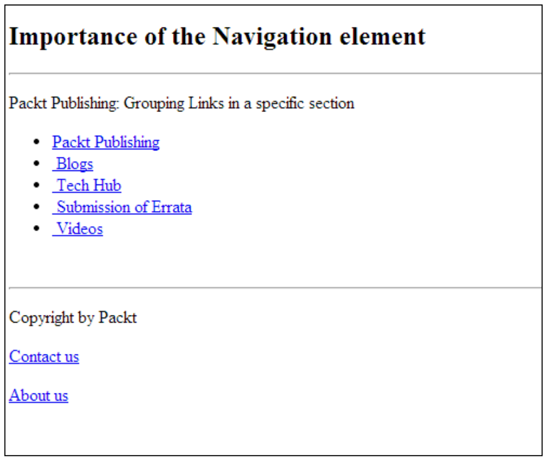
nav

The `<nav>` element is an important part of HTML5 and is used extensively for navigation purposes. It doesn't mean that all links on a page have to be in the `<nav>` tag. For example, the `footer` element has links to varied content and the `<nav>` tag is not mandatory for these links. The `<nav>` element is used when there is a group of navigation links contained in a specific section of the page.

Let's look at the following code to understand it better:

```
<!DOCTYPE html>
<html>
<body>
<header>
<h2> Importance of the Navigation element </h2>
</header>
<hr>
<p> Packt Publishing: Grouping Links in a specific section</p>
<nav>
<ul>
<li> <a href="https://www.packtpub.com/"> Packt Publishing </li>
<li> <a href="https://www.packtpub.com/books/content/blogs"> Blogs </
li>
<li> <a href="https://www.packtpub.com/books/content/tech-hub"> Tech
Hub </li>
<li> <a href="https://www.packtpub.com/books/content/errata">
Submission of Errata</li>
<li> <a href="https://www.packtpub.com/video"> Videos </li> </a>
</nav>
<br><hr>
<footer>
<p> Copyright by Packt </p>
<p> <a href ="https://www.packtpub.com/books/info/packt/contact-us">
Contact us </a>
<p> <a href="https://www.packtpub.com/books/info/packt/about">About us
</a>
</footer>
</body>
</html>
```

The output of the code on execution will be as follows:



If you observe the code, you will find that there are links between the `<nav>` and `</nav>` tags as well as links in the `footer` tags. The content and links between the `<nav>` tags are for major navigation purposes, whereas the links in the `footer` tag are for some additional information.

article

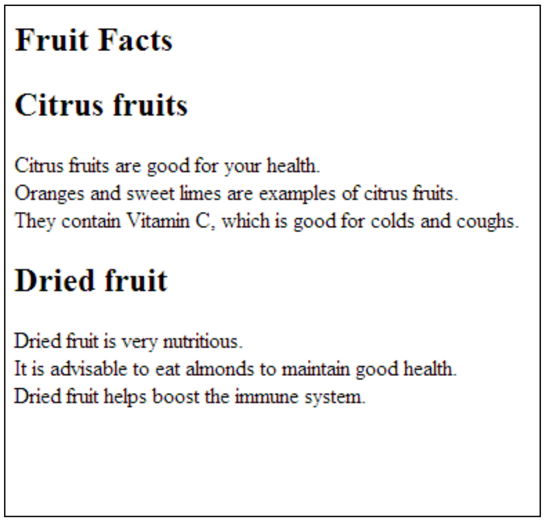
We often come across standalone content on a web page. There are blog entries, forum posts, and user comments, to mention a few. The `article` element is used for these kinds of data where the content is independent of its surroundings. We can also use the `section` element between the `article` tags.

Let's look at the following code to understand it better:

```
<!DOCTYPE html>
<html>
<body>
<article>
<h1> Fruit Facts </h1>
<section>
<h2> Citrus fruits </h2>
Citrus fruits are good for your health.
<br>
Oranges and sweet limes are examples of citrus fruits.
<br>
They contain Vitamin C, which is good for colds and coughs.
</section>
<section>
```

```
<h2> Dried fruit </h2>
Dried fruit is very nutritious.
<br>
It is advisable to eat almonds to maintain good health.
<br>
Dried fruit helps boost the immune system.
</section>
</article>
</body>
</html>
```

The output of the code on execution will be as follows:



Fruit Facts

Citrus fruits

Citrus fruits are good for your health.
Oranges and sweet limes are examples of citrus fruits.
They contain Vitamin C, which is good for colds and coughs.

Dried fruit

Dried fruit is very nutritious.
It is advisable to eat almonds to maintain good health.
Dried fruit helps boost the immune system.

In this code, we included the `section` element within the `article` element. The `article` element might be standalone content, and the `section` tag is used between the `article` tags as it is generic content. Hence, the `article` element is used for independent content such as RSS feeds and news articles, whereas the `section` element is used to denote data of a generic nature.

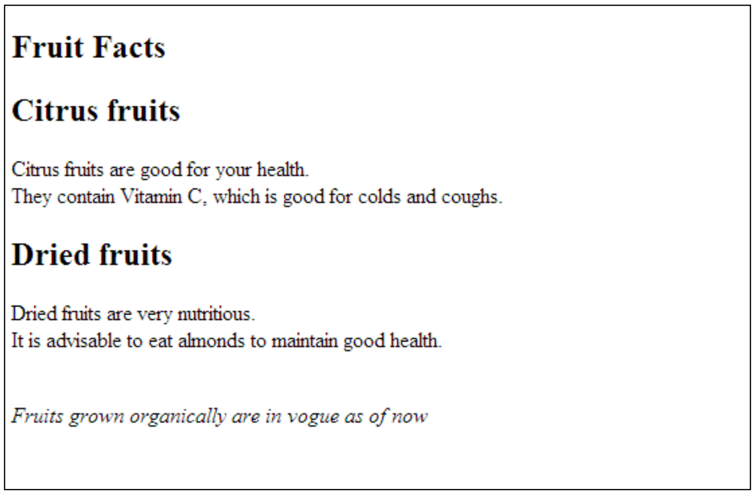
aside

Content in an `aside` element is slightly related to the content outside it. It can be used within an `article` element in cases where the content is relevant to the content in the `article` element. Though the content in the `aside` element is standalone in nature, it can provide additional information about the content around it:

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

```
<article>
<h1> Fruit Facts </h1>
<section>
<h2> Citrus fruits </h2>
Citrus fruits are good for your health.
<br>
They contain Vitamin C, which is good for colds and coughs.
</section>
<section>
<h2> Dried fruits </h2>
Dried fruits are very nutritious.
<br>
It is advisable to eat almonds to maintain good health.
</section>
<br>
<aside>
<p> <em> Fruits grown organically are in vogue as of now </em> <p>
</aside>
</article>
</body>
</html>
```

The output of the code on execution will be as follows:



Fruit Facts

Citrus fruits

Citrus fruits are good for your health.
They contain Vitamin C, which is good for colds and coughs.

Dried fruits

Dried fruits are very nutritious.
It is advisable to eat almonds to maintain good health.

Fruits grown organically are in vogue as of now

If you look at the output of the code, you will see how the `<aside>` tag content regarding organic fruits is slightly related to the main content, which is about "Fruit Facts", though it is standalone and not mainstream.

HTML5 web forms

Prior to HTML5, developers and web designers had to write a lot of code for basic features such as a datepicker. JavaScript was used extensively for basic things, and it was difficult to write such code, given the complexity of the web pages we come across nowadays. HTML5 accounts for cleaner code, and the introduction of new input types makes it easier for developers to design their web pages. In this section, we will discuss the new input types and attributes in HTML5.

New form attributes in HTML5

We will discuss the following form attributes, as they are used quite frequently in web designing in HTML5:

- `placeholder`
- `autofocus`
- `datalist`
- `required`

placeholder and autofocus

The `placeholder` attribute tells the user what data needs to be entered. This information appears in the form of text but with a lighter shade. After the user enters characters in the text field, the text in the lighter shade disappears.

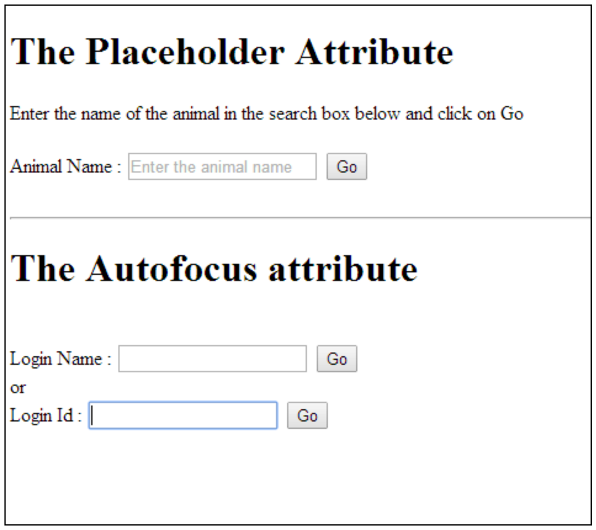
The `autofocus` attribute tells the browser to set the focus on the field where `autofocus` is defined when the page is being loaded.

Let's look at the following code to understand these attributes better:

```
<!DOCTYPE html>
<html>
<head>
<title>
Placeholder and Autofocus
</title>
</head>
<body>
<h1>  The Placeholder Attribute </h1>
<div>
Enter the name of the animal in the search box below and click on Go
<br> <br>
<label for="animal">Animal Name :  </label>
```

```
<input id="animal" placeholder="Enter the animal name">
<input type="submit" value="Go">
<br><br><hr>
<h1> The Autofocus attribute </h1>
<br>
<label for="name">Login Name :</label>
<input id ="name" type="text" />
<input type="submit" value="Go">
<br>
or
<br>
<label for="loginid">Login Id :</label>
<input id ="loginid" type="text" autofocus />
<input type="submit" value="Go">
<br><br>
</div>
</body>
</html>
```

The output of the code on execution will be as follows:



The Placeholder Attribute

Enter the name of the animal in the search box below and click on Go

Animal Name : Enter the animal name Go

The Autofocus attribute

Login Name : Go
or
Login Id : Go

If you observe the output, you can clearly see that the third field, **Login Id**, has the cursor in it as it is set to autofocus. In the same screenshot, we see the **Animal Name** field populated with the **Enter the animal name** placeholder in a lighter shade. After clicking on the textbox, the text disappears. Thus, we saw that `placeholder` is used to help the user fill in the relevant input, whereas `autofocus` will position the cursor in that specific field.

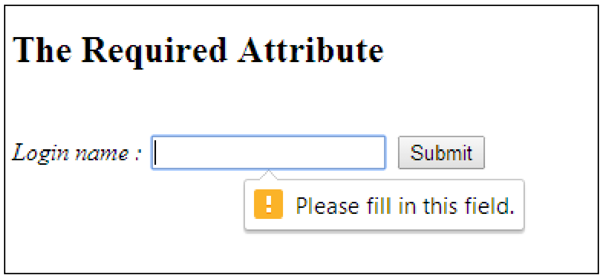
required

The `required` attribute makes it mandatory for a user to enter a value in the field. If the user doesn't enter any data in the field set with the `required` attribute, then it will prompt the user to do so.

Let's look at the following code to understand it better:

```
<!DOCTYPE html>
<html>
<head>
<title> Required attribute </title>
</head>
<body>
<h2> The Required Attribute </h2>
<br>
<form>
<label for="loginid"><em>Login name<em> : </label>
<input name="loginid" type="text" required/>
<input type="submit" value="Submit"/>
</form>
</body>
</html>
```

If you do not enter any value and click on **Submit**, you will get a prompt that asks you to enter data into the field:


The Required Attribute

Login name :

Please fill in this field.

[The message prompt might vary depending on the browser in use, as Google Chrome might render a different prompt compared to Mozilla Firefox, Opera, or Internet Explorer.]

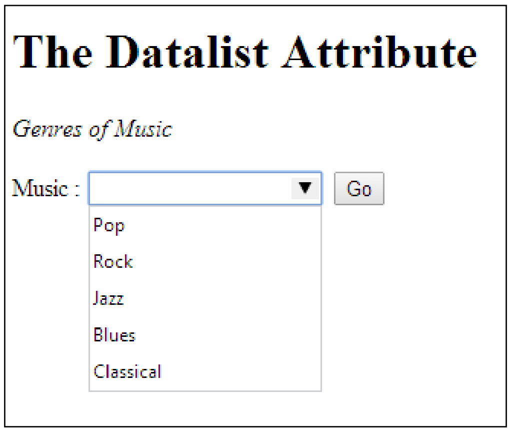
datalist

The `datalist` attribute is used to create a list of predefined items as a drop-down menu. The predefined menu items get displayed when we click on the textbox.

Let's look at the following code to understand it better:

```
<!DOCTYPE html>
<html>
<head>
<title>Datalist : the new form attribute</title>
</head>
<body>
<h1> The Datalist Attribute </h1>
<p> <em> Genres of Music </em> </p>
<div>
<label for="packt"> Music : </label>
<input id="packt" name="packt" type="text" list="music" />
<input type="submit" value ="Go">
<datalist id="music">
   <option value="Pop">
   <option value="Rock">
   <option value="Jazz">
   <option value="Blues">
   <option value="Classical">
</datalist>
</div>
</body>
</html>
```

If you click on the down arrow in the textbox, you will see the drop-down menu that contains different genres of music (predefined in the code), as shown in the following screenshot:



New input types in HTML5

We will now look at the following input types that are frequently used for web designing in HTML5:

- `search`
- `email`
- `url`
- `date`
- `month`

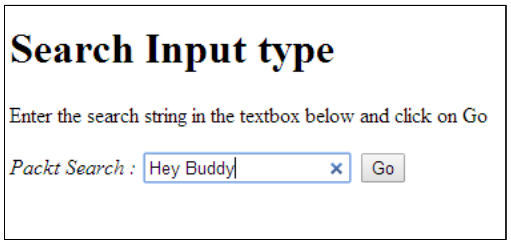
search

With the advent of Google Search, many websites have incorporated a search facility on their web pages. In HTML5, we can create a search box specifying the input type as `search`. Though it looks like a normal textbox, this field is developed to find content within a website or the external web.

Let's look at the following code example to understand it better:

```
<!DOCTYPE html>
<html>
<head>
<title> Search </title>
<h1> Search Input type </h1>
</head>
<p> Enter the search string in the textbox below and click on Go </p>
<div>
<label for="searchid"> <em> Packt Search : </em> </label>
<input id="searchid" type="search" placeholder="Enter Search Item">
<input type="submit" value="Go">
</div>
</html>
```

The output of the executed code will show the following after you enter `Hey Buddy` as the text in the search box. If you click on the blue cross located on the right-hand side of the box, the string will disappear; this means that you can add a new search string in its place.


Search Input type

Enter the search string in the textbox below and click on Go

Packt Search : Hey Buddy

email and url

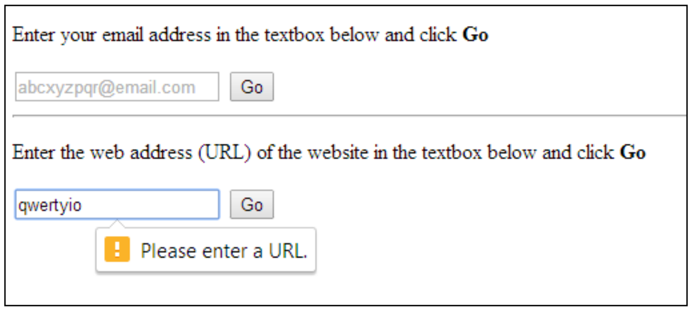
The `email` and `url` input types are the latest additions made specifically for the purpose of e-mail and web addresses respectively. The e-mail address has to be entered with the domain name between an `@` sign and a dot (`.`), for example, `xyz@abc.com`. It accepts the current standard format in which e-mail addresses are written. If you do not write the e-mail address in the correct format, then it prompts you to enter a valid e-mail address.

Most of the URLs are in the `http://www.xyzzzz.com` format. The URL has to be entered in this standard format along with the `http` or `https` protocol. If you do not follow the standardized format, then it prompts you to enter a valid URL.

Let's look at the following code to understand it better:

```
<!DOCTYPE html>
<html>
<head>
<title>Input types: Email and URL </title>
</head>
<body>
<form>
<p> Enter your email address in the textbox below and click
<strong>Go</strong> </p>
<input type="email" placeholder="abcxyzpqr@email.com">
<input type="submit" value="Go">
<br>
<hr>
<p> Enter the web address (URL) of the website in the textbox below
and click <strong>Go </strong> </p>
<input id="website" type="url">
<input type="submit" value="Go">
</form>
</body>
</html>
```

If you do not enter the URL in the standard format, you will see the following output:


Enter your email address in the textbox below and click **Go**

Enter the web address (URL) of the website in the textbox below and click **Go**

Please enter a URL.

Similarly, if you do not enter the e-mail address, it will prompt you to enter it in the standard format.

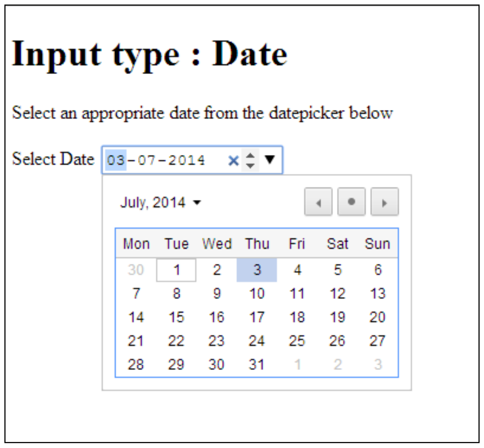
Date and month

In the previous versions of HTML, we had to use JavaScript to develop a datepicker. However, we can create a datepicker with just a few lines of code in HTML5. Apart from date, there are several other related options such as time, week, and month that you can select as per your preference.

Let us look at the following code to understand it better:

```
<!DOCTYPE html>
<html>
<head>
<title> Date Input type </title>
</head>
<body>
<h1> Input type : Date </h1>
<p> Select an appropriate date from the datepicker below </p>
<label for="packt"> Select Date </label>
<input id="packt" type="date" value="2014-07-03">
</body>
</html>
```

The output of the code will be as follows:



Input type : Date

Select an appropriate date from the datepicker below

Select Date 03-07-2014

Mon	Tue	Wed	Thu	Fri	Sat	Sun
30	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	1	2	3

You can choose a date of your choice using this datepicker. As you see, there is no need to use complex JavaScript code or jQuery utilities to develop the datepicker.

HTML5 video

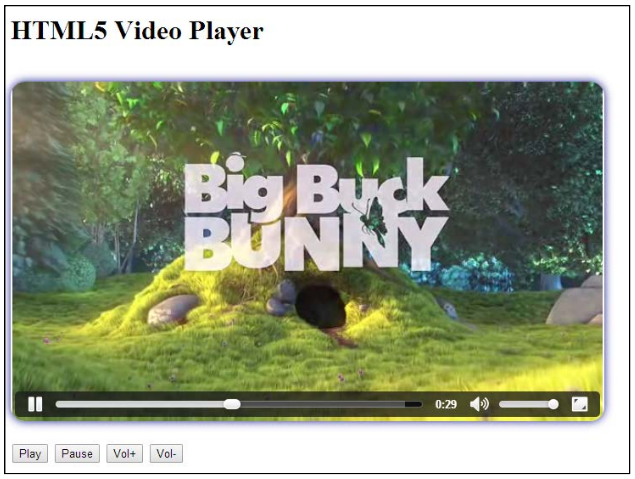
As HTML5 is used extensively for web-based applications, modifications have been made in the present APIs, and new APIs have been introduced. The new APIs support a wide range of features such as Geolocation, Offline data storage, and HTML5 Canvas, to mention a few.

APIs have been developed for media elements such as audio and video. We can embed audio and video, thereby abstracting the need for additional plugins. We can control the `audio` and `video` elements using HTML or JavaScript, whereas the styling will be taken care of by CSS.

We will demonstrate an example of a video player in HTML5. Let's look at the following code to understand this better:

```
<!DOCTYPE html>
<html>
<head>
<title> Video Player in HTML5 </title>
<style>
   video {
      box-shadow:0 0 15px navy;
      border-radius:17px;
   }
</style>
</head>
<h1> HTML5 Video Player </h1>
<br>
<body>
<video width="700" height="400" id= "packt" controls= "controls"
autoplay="autoplay">
<source src ="http://clips.vorwaerts-gmbh.de/big_buck_bunny.ogv" />
<source src ="http://clips.vorwaerts-gmbh.de/big_buck_bunny.mp4" />
</video>
<br><br>
<div>
<button onclick="document.getElementById('packt').play()"> Play </
button>
<button onclick="document.getElementById('packt').pause()"> Pause </
button>
<button onclick="document.getElementById('packt').volume+= 0.1"> Vol+
</button>
<button onclick="document.getElementById('packt').volume-= 0.1"> Vol-
</button>
</div>
</body>
</html>
```

The output of the code upon execution will be as follows:



If you observe the preceding code, you will see that we included the OGV as well as the MP4 formats for the video file. The reason to include both formats is that the MP4 format is not open source and is patented; therefore, there are some constraints due to which some browsers might not support it. The `controls` attribute allows the user to control the video player. For example, we can play and pause the player as and when required. The `autoplay` attribute will play the video automatically without the need to click on the **Play** icon. The `source src` attribute is used for the location of the video file. If you are using your own video, it must be in the same folder as the HTML file, or the entire location or URL has to be specified. Further on, we used the following code snippet:

```
<div>
<button onclick="document.getElementById('packt').play()"> Play </
button>
<button onclick="document.getElementById('packt').pause()"> Pause </
button>
<button onclick="document.getElementById('packt').volume+= 0.1"> Vol+
</button>
<button onclick="document.getElementById('packt').volume-= 0.1"> Vol-
</button>
</div>
```

This code will help you assign manual controls that are not built in with the video player.

HTML is an expansive subject, and there is so much more to learn. With this book, you have just reached the shore of the island. The ocean of knowledge is far beyond. Coding is an art that gets better with practice and effort. By reading this book, you will get to grips with HTML, and you can implement this knowledge to build better websites. HTML is really vast, and the learning curve always shows an upward trend.