My First Website for Students

Build Your First Website from Design to Code with Ease

Age 8+

SHIRISH CHAVAN

bpb

My First Website for Students

Build Your First Website from Design to Code with Ease

Shirish Chavan

www.bpbonline.com

First published: 2023

Published by BPB Online

WeWork

119 Marylebone Road

London NW1 5PU

UK | UAE | INDIA | SINGAPORE

ISBN 978-93-55511-645

www.bpbonline.com

Dedicated to

*Honorable **Sharadchandra Pawar ji***

The Philanthropist and

Former Defense and Agriculture Minister of India

About the Author

Shirish Chavan has been writing code for the past 30 years. In these years, he has written hundreds of computer programs, delivered hundreds of lectures, written a few books on computers, and participated in dozens of seminars. Currently, he is running a small company Webilog that creates websites. He writes in C, C++, C#, VB.NET, HTML, JavaScript, Java, and Python.

He has written the following books:

- C Recipes: A Problem-Solution Approach, Apress, New York, USA.
- Rapidex Computer Course, Windows Vista Marathi Edition, PUSTAK MAHAL, New Delhi.
- Rapidex Computer Course, Windows Vista English Edition, PUSTAK MAHAL, New Delhi.
- Java for Beginners, Second Edition, Shroff Publishers and Distributors, Mumbai.
- Visual Basic .NET, Pearson Education, New Delhi.
- Rapidex DTP Course, Unicorn Books, Division of PUSTAK MAHAL, New Delhi.
- Rapidex Computer Course, Windows 95/98 English Edn, PUSTAK MAHAL, New Delhi.
- Rapidex Complete and Comprehensive: Microsoft Word 2000, Unicorn Books, Division of PUSTAK MAHAL, New Delhi.
- Structured Programming: Go To Controversy to OOP, BPB Publications, New Delhi.

About the Reviewers

Deepti Kuthari, Software Developer interested in exploring new technologies and passionate about developing creative applications, having 6+ years of experience in software development. Completed MCA in 2015, since then working with different companies. Native to Dehradun and working as a technical reviewer in this book for BPB Publications.

Bajram, a Product Designer, and Developer with more than 5 years of experience in creating and converting rich UI / UX into fully-functional code with top-notch Front-End Technologies. Passionate about designing modern, responsive, and dynamic web applications. Specialized in web optimization which speeds up the performance of applications.

He is a good mentor, who has assisted more than 100 students in learning more about web design and applications. He loves to share his knowledge via blogs, video tutorials, courses, etc.

Interested in creating mobile applications with React Native and especially for iOS devices.

Acknowledgement

I want to thank my family members and my friend circle. I cannot do anything without friends.

,

, (Friends, you have done lot of favor to me, I am indebted to you for the same).

My gratitude also goes to the team at BPB Publications for their patience and support in writing this book.

Preface

Congratulations on your prudent decision of buying this book. You have taken the right step in your mission of learning the basics of website creation. In the market, there is no dearth of books on the subject of website creation. But this one certainly stands apart from the crowd.

Firstly, this book is self-contained. While reading this book, you are not required to refer to any other book. Everything that is needed is included in this book. Secondly, this book judiciously covers HTML and CSS in a certain depth so that you can create web pages with confidence. Thirdly, this book covers the essential features of a website, like the inclusion of graphics, audio, video, tables, forms, hyperlinks, and menu bars in the web pages so that you can create the websites like a professional. Fourthly, this is a small book, not a 500-page tome; you can finish it quickly. Finally, this book includes a website project in which you will create a multi-page website and upload it to the server using cPanel. This project ensures that your knowledge is not only theoretical but also practical.

This book is divided into nine chapters. The contents of these chapters are listed below:

Chapter 1 introduces you to the world of websites and covers the rich history of the Internet, emphasizing the history of the World Wide Web. It also takes a quick survey of the latest technologies which are currently used in making the websites.

Chapter 2 illustrates how to create a simple but working website using HTML. It also deals with various types of text formatting elements, grouping elements, core attributes, HTML editors, and validation of web pages. Finally, it explains how to design a website using sitemaps and wireframes.

Chapter 3 describes how to include images, audio, videos, hyperlinks, email links, in-page links, and other web pages in your web page. It also explains how to deal with directories, including subdirectory, parent directory, root directory, and current directory.

Chapter 4 elucidates how to insert tables and forms in your web pages so that you can create tables with row span and column span; also, you can create forms with text-boxes, radio buttons, check-boxes, list-boxes, and submit buttons.

Chapter 5 introduces you to the fascinating world of CSS. In this chapter, you will get exposure to the following types of selectors: universal, type, id, class, child, descendant, adjacent sibling, and general sibling. You will also be introduced to internal and external style sheets.

Chapter 6 ensures that you get a good command of various aspects of CSS, including box models, styles for tables, styles for links, styles for outlines, styles for floating, and styles for positioning.

Chapter 7 spells out how to add code to a website project. In this chapter, you will add a good amount of code to the ongoing website project.

Chapter 8 reveals the secrets of responsive web design. It also explains the viewport meta tag and media queries. Finally, you are exposed to a mini project on responsive web design.

Chapter 9 expounds in detail on the process of uploading a website to the hosting server. You get detailed information about buying a domain name, buying a hosting plan, linking a domain name to a hosting plan using cPanel, and uploading files to a hosting server using cPanel.

I wish you the best of luck in your journey into the world of websites.

Code Bundle and Coloured Images

Please follow the link to download the
Code Bundle and the ***Coloured Images*** of the book:

https://rebrand.ly/bae516

The code bundle for the book is also hosted on GitHub at **https://github.com/bpbpublications/My-First-Website-for-Students**. In case there's an update to the code, it will be updated on the existing GitHub repository.

We have code bundles from our rich catalogue of books and videos available at **https://github.com/bpbpublications**. Check them out!

Errata

We take immense pride in our work at BPB Publications and follow best practices to ensure the accuracy of our content to provide with an indulging reading experience to our subscribers. Our readers are our mirrors, and we use their inputs to reflect and improve upon human errors, if any, that may have occurred during the publishing processes involved. To let us maintain the quality and help us reach out to any readers who might be having difficulties due to any unforeseen errors, please write to us at :

errata@bpbonline.com

Your support, suggestions and feedbacks are highly appreciated by the BPB Publications' Family.

Piracy

If you come across any illegal copies of our works in any form on the internet, we would be grateful if you would provide us with the location address or website name. Please contact us at **business@bpbonline.com** with a link to the material.

If you are interested in becoming an author

If there is a topic that you have expertise in, and you are interested in either writing or contributing to a book, please visit **www.bpbonline.com**. We have worked with thousands of developers and tech professionals, just like you, to help them share their insights with the global tech community. You can make a general application, apply for a specific hot topic that we are recruiting an author for, or submit your own idea.

Reviews

Please leave a review. Once you have read and used this book, why not leave a review on the site that you purchased it from? Potential readers can then see and use your unbiased opinion to make purchase decisions. We at BPB can understand what you think about our products, and our authors can see your feedback on their book. Thank you!

For more information about BPB, please visit **www.bpbonline.com**.

Table of Contents

Index

CHAPTER 1

Fascinating World of Websites

Introduction

In this chapter, you will be introduced to the history of the Internet. Here, you begin with the launching of Sputnik by Russia and arrive at the birth of the World Wide Web in 1989. You are also introduced to a brief history of the World Wide Web. Here, you begin with Douglas Englebart's NLS system and learn how Tim Berners-Lee developed World Wide Web while working at CERN. You are also given a quick review of various important technologies that are used in making the websites, namely, HTML, CSS, JavaScript, Java, Python, jQuery, PHP, MySQL, Ajax, ASP.NET, Laravel, AngularJS, Bootstrap, Django, Dreamweaver, FTP, Node.js, React, Ruby, and WordPress. Finally, you are introduced to our website project that you will build in a step-by-step manner in the remaining chapters of this book.

Structure

In this chapter, we will cover the following topics:

- Brief history of the Internet
- Brief history of the World Wide Web
- Quick review of various technologies used in making the websites
- Quick review of a website project that you will execute in this book

Objectives

After reading this chapter, you will learn how the development of the Internet took place, what is ARPANET, and what is its role in the development of the Internet. You will also learn about packet switching technology and the role of TCP/IP in the development of the Internet, who coined the term the Internet, and how the development of the World Wide Web took place, you will learn about the development of ENQUIRE by Tim Berners-Lee and who created fantastic web browsers like ViolaWWW, Mosaic, Netscape, and Internet Explorer, domain name registry, the names of technologies which are used in the making of websites, and quick introduction to our website project.

Internet: because they wanted to share the documents

The Internet was developed because the founding fathers of the Internet just wanted to share documents over the network. Things like WWW are just spin-offs. In this section, we will briefly review the history of the Internet.

The history of the Internet begins with the invention of the modem. The first modem was developed by Bell Labs in 1958. A modem (it is an acronym for modulator and demodulator) is an instrument that

converts signals from computers into signals that can be transported over telephone wires. It also performs the reverse job, that is, it accepts the signals from telephone wires and converts them into signals that can be read by a computer. The invention of a modem made communication between computers possible via telephone lines. The credit for the invention of the modem goes to the celebrated Bell Labs.

In 1957, the then USSR - now Russia - launched the first artificial satellite Sputnik and installed it in an orbit around Earth. The decade of 1960s was witness to serious brawls between USA and USSR. Therefore, the USA seriously decided to do something - in the Defense sector - to score points over USSR. One of the things that the USA did seriously was entrust US Defense **Department's Advanced Research Projects Agency** (**DARPA**) with the task of creating a secure network to link the computers at military research centers.

Before proceeding further, few words about the name DARPA.

In 1971, the name ARPA was changed to DARPA.

In 1993, the name DARPA was changed back to ARPA.

In 1996, the name ARPA was again changed to DARPA.

Our sincere apologies to Shakespeare who thought there is nothing in the name.

The very remarkable feature of this secure network was that it was expected to work (albeit with less capability) even if some of the research centers were destroyed, hence the name secure network. This proposed secure network is named ARPANET.

The very first attempt of ARPANET to connect computers by wire took place on 29th October 1969 and it attempted to connect the machine at **UCLA** (**University of California, Los Angeles**) to the machine at **SRI** (**Stanford Research Institute**). In this attempt, the word LOGIN was typed at the UCLA machine and it was successfully received by the SRI machine. Actually, the network crashed while receiving the third letter G, but the scientists working there managed to fix the problem within a few hours and eventually succeeded in sending and receiving the word LOGIN. Thus, a successful connection was established between these two machines. In the same year, the very first Internet application Telnet (this application allows you to operate a remote machine from your machine using a command-line interface, provided both machines are connected by a network) was demonstrated by its developers.

Within a couple of months, two more machines were added to ARPANET. One machine was from **UCSB** (**University of California, Santa Barbara**) and another machine was from the University of Utah. In the following year (that is, in 1970), there were about ten machines (or nodes, in technical language) connected to ARPANET.

In 1971, **FTP** (**File Transfer Protocol**) - a standard communication protocol to be used in the transfer of files from the server to the client - was released by its developers.

In 1972, at an international conference on computer communications, ARPANET was publicly demonstrated and this demo was a roaring success. In the same year, computer programmer Ray Tomlinson wrote software that could send and receive electronic mail or email. He also used the symbol @ in the address of the email as a separator between the username and domain name.

In 1973, two nodes, outside the USA, became part of ARPANET and ARPANET became international! These nodes were located at the University College of London and the Norwegian Royal Radar Establishment. In the same year, Robert Metcalfe working at Xerox Corporation and his group invented Ethernet technology that was very helpful in the implantation of **LAN** (**Local Area Network**). The

benefits of this technology are:

- Easy to install and manage
- Low priced
- Flexible and scalable

In this year (that is, in 1973), the scientists Vinton Cerf and Robert Kahn developed the TCP/IP technology. This technology has been responsible for making the Internet fail-safe and robust.

TCP in TCP/IP stands for Transmission Control Protocol. IP in TCP/IP stands for Internet Protocol.

Firstly, let us see what is meant by IP. Every machine connected to the Internet has a unique IP address that looks something as follows:

```
n1.n2.n3.n4
```

Where **n1**, **n2**, **n3**, and **n4** are the integers in the range **0** to **255**. Just for fun, if you want to find the IP address of your computer or mobile (that is connected to the Internet), then simply type the question **What is my IP address** in the Google search box and press the Enter key. Then, Google will display the answer to your question as shown in *Figure 1.1*:

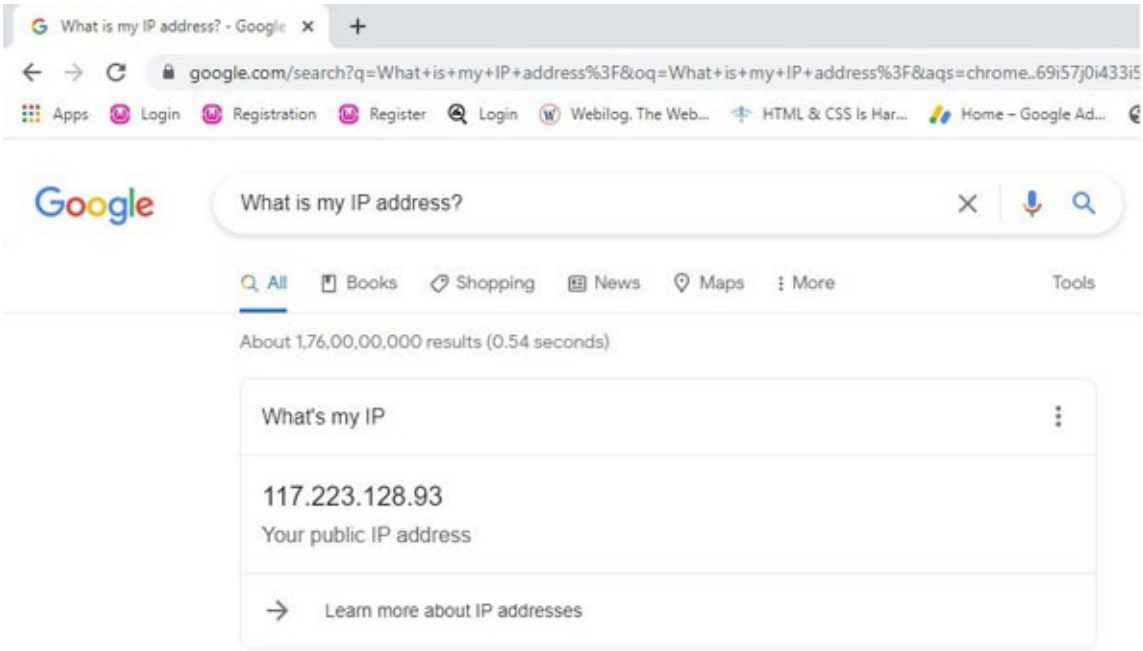

Figure 1.1: *You can find your IP address with the help of Google.*

In the IP address, the left-most number (**n1**) is the most generic and the right-most number (**n4**) is the most specific. Perhaps, the number n4 identifies your computer or mobile. This is simply opposite to the residential address system that we use to address our houses. In our residential address system, the left-most data item is the most specific (your house number) and the right-most data item is the most generic (name of your country).

This IP address system is known as **IPv4**, that is, **Internet Protocol version 4**. Theoretically, it can accommodate (232 = 4,294,967,296 =) 4.3 billion IP addresses. Right now, there are about 4.72 billion Internet users in the world. Clearly, the IPv4 is not capacious enough to accommodate these many users. Hence, a new version of IP - that is more capacious - is coming, it is named as **IPv6** (**Internet Protocol version 6**). The IP address according to this version is something as follows:

```
n1.n2.n3.n4.n5.n6
```

Where **`n1`**, **`n2`**, **`n3`**, **`n4`**, **`n5`**, and **`n6`** are the integers which range from 0 to 65535. This version can accommodate a whopping (2128 = 3.4 x 1038 IP addresses =) 340 trillion trillion trillion IP addresses.

Now. let us see what is meant by TCP. In order to pass the message from one node to another node, we need to use one of the following technologies:

- Circuit switching technology
- Packet switching technology

Suppose a message is to be sent from node A to node B. Then, in circuit switching technology, a well-defined path is prefixed from A to B, then this path is reserved, after that, the message is sent along this path, and once B receives this message, then this path is unreserved.

In packet switching technology, the message to be sent is divided into a number of packets (say 10 packets). Each packet contains the unique ID of this message, the serial number of the packet (from 1 to 10), and the address of recipient node B. There is no prefixed path for these packages and each package follows the path that is easily available to it. Thus, different packages may follow different paths. If one of the paths followed by some packet is broken, then the packet is routed by another path. This is the most remarkable thing about packet switching technology. Packets do never stop; they march toward their destination address using every available path. The Internet is a huge success because the founding fathers of the Internet chose packet switching technology instead of circuit switching technology for the transmission of messages.

The TCP implements the packet switching technology. A part of the credit for the development of TCP also goes to Leonard Kleinrock of MIT who wrote a research paper and a book on this subject.

Kahn introduced the idea of an open-architecture network for the Internet. In an open-architecture network, the designers of the individual network have the freedom to design their networks the way they want and then they can connect their network to the Internet. This open-architecture network concept fuelled the growth of the Internet.

In 1974, the scientists Vinton Cerf and Robert Kahn coined the term ‘Internet’ for Internet in a research paper on data protocols written by them. Also, in this year, the first private network called Telenet (not to be confused with Telnet, the application of Internet) - which was a commercial version of ARPANET - was launched to serve needy customers.

In 1981, the National Science Foundation launched the **Computer Science Network** (**CSNET**). It was created to connect the various computer science departments.

In 1983, the first domain name registry started its operations. In 1985, the first commercial domain name was registered. It was “symbolics.com” registered by a computer system company. Before the invention of domain names, there were IP addresses in the number format. Domain name register is something like a telephone directory. In the telephone directory, we find the names of persons listed along with their telephone numbers. In the domain name register, you can find the names of websites (for example, **`www.webilog.in`**) listed along with their IP addresses (for example, **`n1.n2.n3.n4`** where each n is an integer in the range of **`0`** to **`255`**).

In 1985, Stephen Wolff launched the **National Science Foundation Network** (**NSFNET**). It was created to connect the various Universities to one another via a computer network.

In 1989, computer scientist and software developer Alan Emtage created the very first search engine, Archie. Since then, smarter and smarter search engines have been built by computer scientists and software companies. It is now a million-dollar business.

In 1989, a computer scientist working at CERN, Tim Berners-Lee, invented the now ubiquitous **World Wide Web** (**WWW**). The next section deals with WWW in detail.

WWW: Tim Berners-Lee's exotic creation

Tim Berners-Lee was born in 1955 in England to computer-expert parents Mary Lee and Conway Berners-Lee is largely responsible for the creation of this fantastic thing known as the World Wide Web.

The world wide web was dreamt of by Vannevar Bush - the Director of the Office of Scientific Research and Development, USA - in his article As We May Think published in the magazine "Atlantic Monthly." In this article, he proposed the system memex in which one can store their books, records, and communications and can also consult these documents at the lightning speed. He also hinted at the idea of hypertext in this article. Bush's article created a stir in the computing community and people read this article with great curiosity.

Heavily influenced by Bush's article, in 1945, scientist Douglas Englebart started the research laboratory that was named "Augmentation Research Centre." He developed a system - which he named **NLS** (**oNLine System**) - that was capable of storing all sorts of documents needed by academicians and scientists with the facility of linking these documents to one another. In 1968, he invented the mouse and demonstrated its use in an official demonstration.

The term hypertext was coined by Ted Nelson in 1965. He defined it as text or graphics that is connected to another text or graphics via links. Ted Nelson is well-known for the hypertext system Xanadu developed by him.

Some of the famous hypertext systems developed in the period from 1967 to 1987 are HES, FRESS, ZOG and KMS, Intermedia, GUIDE, and Hypercard.

After the invention of WWW, no other hypertext system was needed because WWW itself was the most comprehensive hypertext system.

Tim Berners-Lee - the inventor of WWW - joined CERN in June 1980 as computer expert. CERN, the center of European Research for Nuclear Physics, Geneva, situated at the boundary of France and Switzerland, was established in 1954 to shift the center of global research from the USA to Europe. CERN was the right place for a creative-minded person like Tim Berners-Lee.

Tim worked at CERN for six months (this was not a regular service but contract-type job). In this period of six months, he developed a hypertext system that he named ENQUIRE. The aim of this system was to store and retrieve (as per requirement) the desired information regarding the research at CERN. In ENQUIRE, all the documents were successfully linked to one another.

The system ENQUIRE was named after the very famous, old Victorian book **Enquire Within Upon Everything** edited by Robert Kemp Philp, first published in 1850 and that was full of cross-indexing. This book contained a large number of tips to deal with various situations in life. There was a copy of this book in Tim's home since his childhood and it influenced Tim a lot.

In 1984, once again, Tim was contracted by CERN for software-related work and he joined CERN in September 1984. This time he decided to improve ENQUIRE. A major shortcoming of ENQUIRE was that all the files needed to be stored on a single machine. He wanted to free ENQUIRE from this constraint.

However, in 1988, Tim decided to stop the work on ENQUIRE and thought about developing WorldWideWeb (a single word and no spaces within). In fact, this WWW was nothing but his ENQUIRE

system free from all the shackles and was able to work worldwide. In March 1989, Tim submitted a proposal to the top authorities of CERN (with a recommendation by Mike Sendall, his boss) for the development and implementation of WorldWideWeb at CERN. But this proposal failed to impress the top authorities of CERN. Undeterred, Tim and his colleagues continued their work, and finally, on 6 August 1991, they were able to put the first web page online. Therefore, this date can be regarded as the birth date of WorldWideWeb.

While fabricating the WorldWideWeb, Tim and his colleagues made use of the following essential components:

- **Internet**: WorldWideWeb is possible because of the Internet. No Internet, no WorldWideWeb. If you want to view a web page, you need a machine that is connected to the Internet.
- **HTML language**: A language in which one can write a web page. Tim derived this language from the then-existing language SGML. He prudently threw away the complexity from SGML.
- **Web browser**: Using this software, one can view the web pages which are available on the Internet.
- **HTTP and URL**: Without these technologies, WorldWideWeb could not exist. **HTTP** stands for **Hyper Text Transfer Protocol**. It deals with the transmission of the web page from a web server to the web client. **URL** stands for **Uniform Resource Locator**. URL is nothing but the address of the web page. If you want to view a web page, then you must know its address. For example, `www.bpbonline.com` is the URL of the website of BPB Publications.

In 1992, the web browser ViolaWWW arrived and it was for the platform of UNIX. It was created by Pei Wei. This browser had its own style sheet language. In the same year, the number of web servers rose to 50.

In 1993, the number of web servers rose to 500. In the same year, the web browser Mosaic arrived. Mosaic was written by Marc Andreesen. It was capable of displaying images.

In 1994, the number of web servers rose to 2500. In the same year:

- W3C (WWW Consortium) began its operations.
- Web browser Netscape 1.0 became available for web users.
- Yahoo! Web directory was started by Jerry Yang and David Filo.

In 1995, web browser Netscape 2.0 was made available by its developers for web users. This state-of-the-art web browser was capable of handling plug-ins, frames, Java applets, and JavaScript.

In 1995, the web browser Internet Explorer 1.0 was released by Microsoft.

In 1996, Macromedia Flash 1.0 was released by its developers. In the same year, the Internet Archive began archiving the WWW.

In 1997, HTML 4.0 was released by its developers. In the same year, the browser war began between Netscape and Internet Explorer.

In 1998, Network Solutions registered the 2 millionth domain name. In the same year, <XML> appeared on the scene and Google search engine was released by its developers, namely, Larry Page and Sergey Brin.

In 2000, the number of indexable web pages rose to 1 billion. In the same year, W3C recommended

XHTML 1.0 for web developers.

In 2001, Wikipedia was made available to web users by Jimmy Wales and Lawrence Sanger. Wikipedia is an online encyclopedia that is available free of cost for web users.

In 2003, various music downloading services were made available by its developers to music lovers all over the world. In the same year, viruses, worms, and trojan horses created many problems for web users.

In 2004, Facebook made its appearance on the scene. It was developed by Mark Zuckerberg. Within the next few years, it took the world of social media by storm.

In 2005, YouTube was released by Steve Chen, Chad Hurley, and Jawed Karim for web users. It revolutionized the world of video sharing. Anyone could upload a video on YouTube and anyone could watch this video without making any payment. The idea was loved by people.

In 2006, Twitter made its appearance on the scene. It introduced the concept of micro-blogging. It imposed the limit of 140 characters (hence, micro-blogging) on the message that can be posted on Twitter.

In 2008, Firefox 3 was released by its developers. It created the world record for the maximum number of downloads in a single day. There were 8 million downloads of this exotic piece of software in a single day. In the same year, Google released the web browser Chrome for web users. Soon it became very popular among web users.

In 2013, out of the total number of web page views made, the web users about 13% web-page-views were made on mobiles.

Websites became hi-tech

When the first website was developed, HTML was the only language that could be used for creating websites. However, with the passage of time, a number of languages and technologies became available to help web programmers in their mission of building complex websites. In this section, we will take a quick review of various languages and technologies that are currently used in making websites.

HTML

HTML is the mother tongue of any website. You just cannot create any website without using HTML. As you will see in the rest of the book, HTML allows us to fill a web page with the desired content. HTML was created by Tim Berners-Lee in 1991 and since 1996, it is maintained by W3C. You don’t need to buy any compiler or interpreter to use HTML because every web browser contains an interpreter to interpret the HTML program.

CSS

CSS allows us to offer the desired look to our web page. Using CSS, you can have the desired fonts for text matter, desired colors for the background and foreground, and desired borders for your pictures. With the help of CSS, a web page becomes more stylish. CSS is the brainchild of HakonWium Lie and he proposed it in October 1994. At that time, he was working at Opera Software Company. CSS was officially accepted by W3C in 1996. CSS is now managed by W3C. Like HTML, you are not needed to buy any compiler or interpreter to use CSS because every web browser contains an interpreter to interpret CSS code. CSS code can be easily embedded in the HTML code.

JavaScript

JavaScript is a scripting language and it has nothing in common with Java. It was developed by Brandan Eich in September 1995 who was a Netscape programmer at that time. In 1996, Netscape and Eich submitted JavaScript before ECMA International Standards Organization. JavaScript is now managed by Oracle Corporation which is the owner of JavaScript. JavaScript is a hot choice for creating dynamic web pages. It can be easily embedded in HTML code.

Java

Java is very popular programming language. Java-based technologies like JSE, JEE, Javabeans, and JSP are quite popular among web programmers. Java was created by James Gosling and his colleagues (working at Sun Microsystems) in 1995. Java is now managed by Oracle Corporation who is the current owner of Java. Java is heavily used for developing dynamic web applications. The website of Java is `www.java.com/en/` and you can download the compiler of Java from this website.

Python

Python was designed by Guido van Rossum of Netherlands in 1991 and developed by Python Software Foundation. Prior to that Guido van, Rossum had implemented the programming language ABC. While using the ABC language with the operating system AMOEBA at CWI, he noticed some shortcomings in ABC. Then, he decided to create a new language that would contain all the goodness of ABC but free from its shortcomings. The first version of Python appeared in 1991. The current version is called Python 3.9.x. Python is now managed by the Python Software Foundation who also owns Python. Its website is **www.python.org** and its interpreter can be downloaded from this website free of cost.

jQuery

jQuery is not a programming language but a JavaScript library. It means it consists of a large number of ready-to-use programs written in JavaScript. Because of its utility, it has become quite popular. jQuery was created by John Resig in January 2006. The popularity of jQuery is really astonishing. In 2021, about 78% of the top 10 million websites were using jQuery. jQuery now is managed by the jQuery Foundation. Its website is `www.jquery.com` and JavaScript library can be downloaded from this website free of cost.

PHP

PHP (**Hypertext Preprocessor**) was created by Rasmus Lerdorf in 1994. Earlier, its name was Personal Home Page. It was officially released in 1995. In 1997, it was transferred into the public domain. The current version of PHP is 8. x and it is quite popular among web programmers. Right now, it is managed by the PHP Development Team, Zend Technologies. Its website is `www.php.net` and PHP interpreter can be downloaded free of cost from this website.

MySQL

MySQL is not a programming language; it is an open-source **RDBMS** (**Relational Data Base Management System**). We can use it to store, retrieve, and modify the desired information in the

database. The internal working of the RDBMS is based on the instructions provided in SQL (Structured Query Language). MySQL can be used with various programming languages like PHP, Python, PERL, C, and C++. However, MySQL is quite friendly with PHP. MySQL was created by Michael Widenius, Allan Larsson, and David Axmark. The first version of MySQL made its appearance in May 1995. Its website is www.mysql.com and we can download the complete source code of MySQL from this website. As it is open-source software, we are not required to make any payment to use it. Currently, it is managed by Oracle Corporation which acquired it in January 2010.

Ajax

Ajax is an acronym for Asynchronous JavaScript and XML. Ajax is not a new technology but just a collection of existing technologies for building highly interacting web sites and web applications. The term Ajax was coined by Jesse James Garrett in February 2005. The aim of Ajax is to build a web application as suitable as desktop application. The various technologies used in Ajax are: HTML (or XHTML), CSS, DOM, JSON or XML, XMLHttpRequest, and JavaScript.

Microsoft ASP.NET

ASP (**Active Server Pages**) is a server-side scripting language created by Microsoft in 1996. Its first version ASP 1.0 appeared in December 1996. Its third version ASP 3.0 appeared in November 2002. In January 2002, Microsoft released ASP.NET which superseded ASP.ASP.NET is an open-source web framework, created by Microsoft, for building modern web apps and services. NET. ASP.NET is cross-platform and runs on Windows, Linux, macOS, and Docker. Its website is **dotnet.microsoft.com/apps/aspnet** and you can download the necessary software from this website. However, it is not absolutely free and you are required to make a payment for using this service. ASP.NET allows us to build large-sized and complex websites. The websites of Microsoft, Godaddy, Visual Studio, and Dell are built using ASP.NET.

Laravel

Laravel is a web application backend PHP framework with expressive, elegant syntax. It was created by TalylorOtwell and its first version appeared in June 2011. Its website is **www.Laravel.com** and you can download the necessary documentation from this website. The benefit of using Laravel is that it is a powerful tool and does some of your work, thus saves your labour as a web programmer. Laravel is free, open-source framework, hence you are not required to make any payment for its use. The source code of Laravel is available at GitHub (**www.github.com/laravel/framework**) and it is licensed under MIT License.

AngularJS

AngularJS is a front-end web framework. It is used for developing single page applications. It is based on JavaScript and also written in JavaScript. It is open source and licensed under MIT License. Its website is **www.angularjs.org** and you can download the required software from this website. AngularJS was created by MiskoHevery at Brat Tech LLC in 2009. Its first version appeared in October 2010. Currently, the stable version of AngularJS is 1.8.2 and it made its appearance in October 2020. AngularJS is now maintained by Google.

Bootstrap

Bootstrap was created by Mark Otto and Jacob Thornton - working at Twitter - in August 2011. Bootstrap is a HTML, CSS, and JavaScript Library. Bootstrap is free and open source software. It is a front-end CSS framework created especially for responsive web development. Currently, its stable version is 5.1.0 released in August 2021. Its website is www.getbootstrap.com and you can download the required software from it. Bootstrap supports responsive web design. The benefit of responsive design is that contents of web page are automatically adjusted taking into account the size of the device (for example, desktop, tab, mobile, and so on) in which the web page is to be displayed. It is licensed under MIT License. Currently, it is managed by Bootstrap Core Team.

Django

Django was created by Adrian Holovaty and Simon Willson in July 2005. Currently, its stable version is 3.2.5 which appeared in July 2021. Django is a Python-based, free, and open-source web framework. Django is particularly useful when you want to develop a secure and maintainable website using Python. The benefit of using Django is that it does some of your work and reduces your labor as a web programmer. It is managed by Django Software Foundation. Its website is `www.djangoproject.com` and you can download the required software from this website. The websites of Mozilla and Instagram are created using Django.

Dreamweaver

Dreamweaver is a software that helps us in making the websites. It is not a programming language; it is a proprietary web development tool. It was created by Macromedia Company. Its first version 1.0 appeared in December 1997. In 2005, Macromedia Company was acquired by Adobe Inc. Its website is `www.adobe.com/products/dreamweaver` and you can download the trial version of this software from this website. Why use Dreamweaver? Read on:

- Dreamweaver is easy to use.
- The web pages created using Dreamweaver are consistent.
- Website can be managed and updated effectively.
- Uploading of files is easy using the built in FTP facility in Dreamweaver.

FTP

FTP (**File Transfer Protocol**) is not a programming language but it is a data transmission protocol. The specification for FTP was written by the Indian Computer Scientist Abhay Bhushan in April 1971. This original version of FTP was published as RFC 114. Since then, it has undergone few revisions. The last revision was made in June 1997. When FTP was introduced in 1971, it was revolutionary and served as a launching pad for other data transmission protocols. However, with the passage of time, FTP has lost its relevance. Now, FTP is no more an important method of data transmission. Currently, on the Internet, TCP/IP is used for data transmission.

Node.js

Node.js is not a programming language. It is also not a framework. It is a run-time, open source, and development platform. It executes the JavaScript code on the server-side. Using Node.js, we can perform tasks like network Input-Output and File Handling which are not possible with JavaScript. It was created by Ryan Dahl in 2009. Its website is www.nodejs.org and you can download the required files from this website. Node.js is currently managed by OpenJS Foundation. Its first version was released in May 2009 and the current stable version (16.4.0) was released on June 2021. Its code repository is at **www.github.com/nodejs/node**.

React

React was created by Jordan Walke in order to deploy it on Facebook's news feed in 2011. In the next year, in 2012, he deployed it on Instagram. React is also called React.js and also ReactJS. It is a free, open source, front-end JavaScript library that is used for building user interfaces. React was released publicly in May 2013. Its current stable version is 17.0.2 which was released in March 2021. Its website is `www.reactjs.org` and its code repository is `www.github.com/facebook/react`. It is managed by Facebook and the React community.

Ruby

Ruby is a high-level, general-purpose programming language. It is an interpreted language. It was created by the Japanese computer scientist Yukihiro Matz Matsumoto. Its first version appeared in 1995. Its current, stable version is 3.0.1 and it was released in April 2021. The website of Ruby is `www.ruby-lang.org` and you can download the required files from this website. Its compiler was written in C. Its license is Ruby License. Ruby is managed by the Ruby Community.

WordPress

WordPress is not a programming language. It is a **CMS** (**Content Management System**). It is free and an open-source software. It is written in PHP. For the management of data, it uses MySQL (a query language) and MariaDB (a database). It is created by Matt Mullenweg and Mike Little. Think of a website as a sort of burger. We need a lot of stuff to fill in this burger. This stuff is provided by CMS. The first version of WordPress appeared in May 2003. The current stable version of WordPress is 5.8 which was released in July 2021. Its website is **www.wordpress.org** and you can download WordPress software from this website. WordPress is managed by WordPress Foundation.

Apart from WordPress, some other CMSs available to us are Joomla, WooCommerce, Drupal, Wix, BigCommerce, Shopify, and Magento. You can choose the CMS depending on your requirement. Not all are free, some of them charge license fees. Also, some of them also provide you with the facility of a shopping cart, something like Amazon.

Our website project

The website that we will develop in this book is already created and placed at the following URL:

www.webilog.in/tbs

This is planned to be a website of a fictitious company Truly Best Software. It consists of three pages, as follows:

- Home page
- Careers page
- Contact page

In the latter part of the book, we will develop this website in a step-by-step manner.

Figure 1.2 shows the home page of our website project:

Figure 1.2: *Home page of our website project.*

Figure 1.3 shows the careers page of our website project:

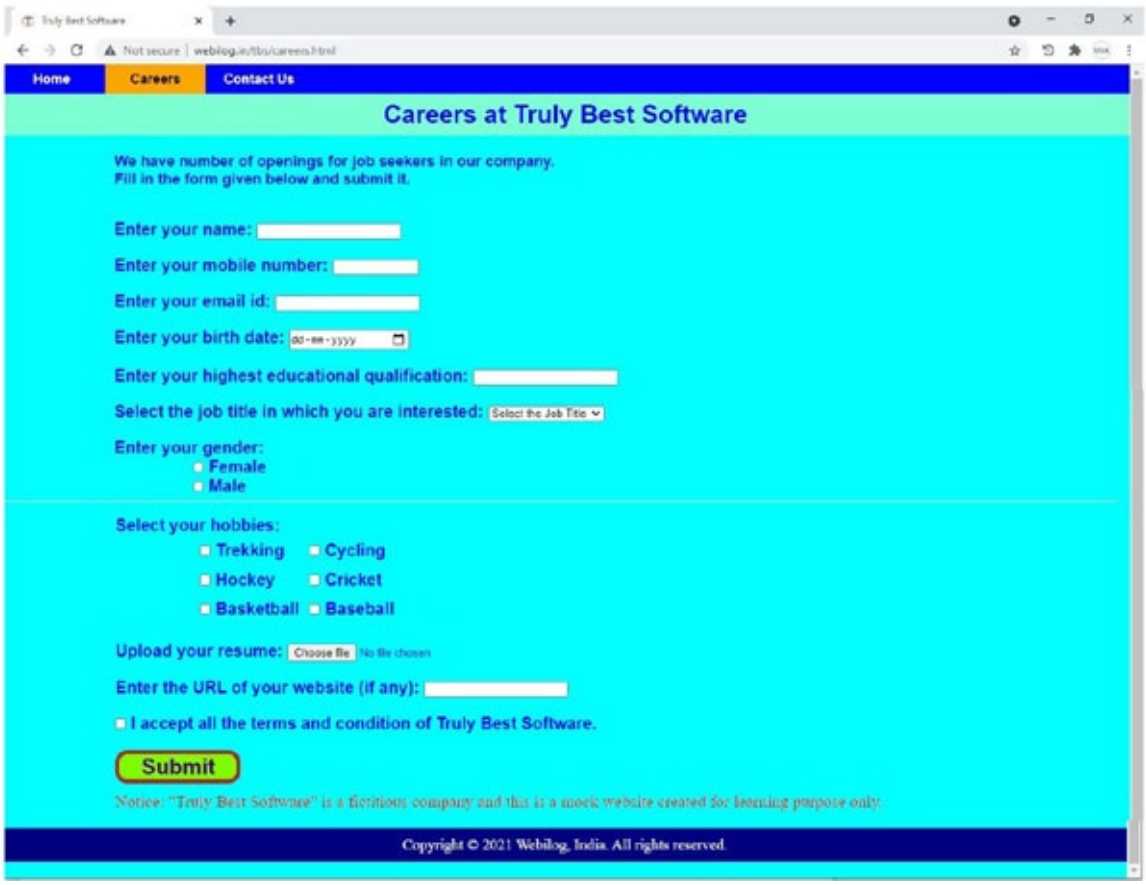

Figure 1.3: *Careers page of our website project.*

Figure 1.4 shows the contact page of our website project:

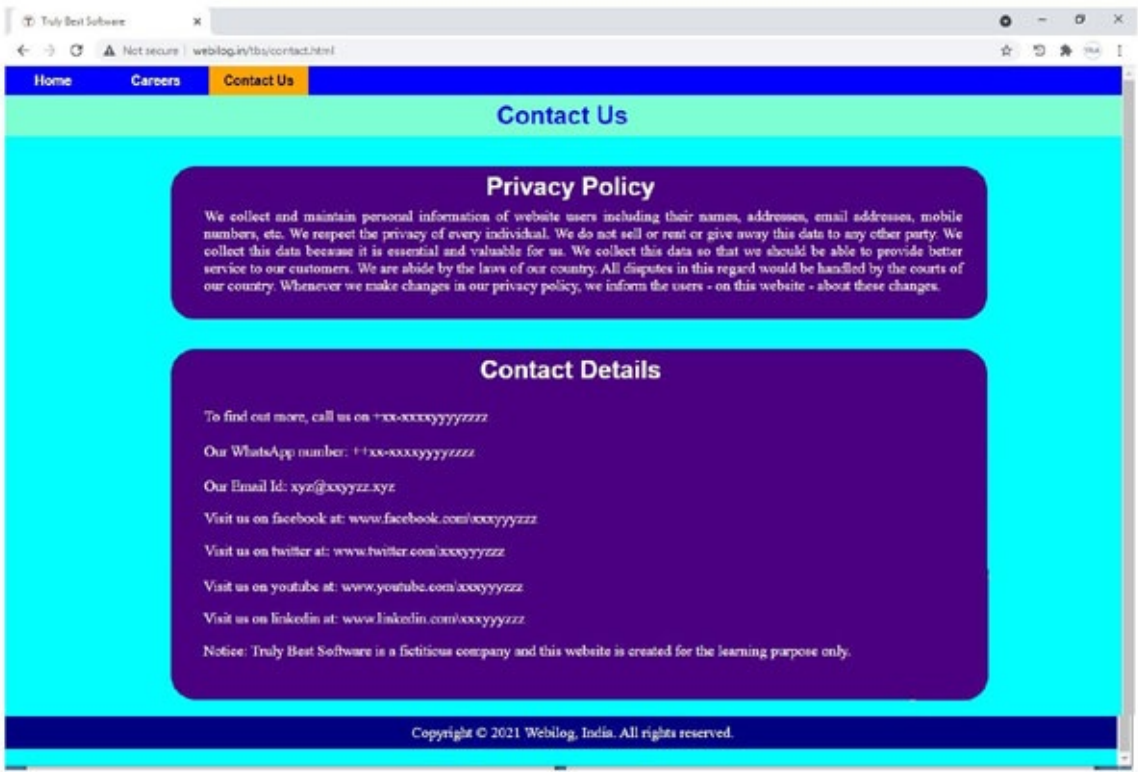

Figure 1.4: *Contact page of our website project.*

Conclusion

In this chapter, you learned the fascinating history of the development of the Internet, including the packet switching technology and TCP/IP protocols. You also learned the equally fascinating history of the development of World Wide Web, including the crucial role played by Tim Berners-Lee. Then, you learned various technologies used in the making of websites. Finally, you were introduced to our website project. In the next chapter, you will create your very first web page, and you will learn to use text formatting elements, block elements and inline elements, inserting special characters in HTML web page, HTML editors, validating web pages, and design your website using sitemaps and wireframes.

Further readings/references

- *Computer: A History of the Information Machine*, by Martin Campbell-Kelly, et al, 3/e, Westview Press, 2014.
- *Where Wizards Stay Up Late: The Origins of The Internet*, by Katie Hafner, et al, Simon and Schuster, 1998.
- *Weaving the Web: The Original Design and Ultimate Destiny of the World Wide Web*, by Tim Berners-Lee, Harper Business, 2000. **https://www.scientificamerican.com/article/long-live-the-web/** This fascinating article is written by Tim Berners-Lee and published on December 1, 2010.
- **https://www.scientificamerican.com/article/day-the-web-was-born/** This fascinating article is written by Larry Greenemeier and published on March 12, 2009.

CHAPTER 2

Creating the Web Pages

Introduction

In this chapter, you will learn how to build your very first web page, use text formatting elements, and use the p element, h element, block elements, inline elements, horizontal rule, and line break. You will learn to apply the desired font style to the text on a webpage using various elements. You will be introduced to special characters, core attributes, HTML editors, and the validation of webpages. Finally, you will learn to use sitemaps and wireframes to design and build your website.

Structure

In this chapter, we will discuss the following topics:

- Our very first webpage
- Text formatting elements
- Block elements and inline elements
- Special characters
- Core attributes
- HTML editors
- Validation of a webpage
- Designing a website using sitemaps and wireframes

Objectives

After reading this chapter, you will be able to create a very basic webpage. You will also learn to format the text on the webpage, use block elements and inline elements, use special characters, use core attributes, use HTML editors to create the webpages, and validate your webpage using the service of W3C. Finally, you will be able to design a website using sitemaps and wireframes.

Our very first web page

Now, we will create our very first web page. Fasten your seat belts.

In this book, I assume that you are using a Windows-based computer. I also assume that at the beginning of every lesson, you have already booted your computer and your desktop displays your familiar background picture.

Before proceeding further, notice the following:

Note: The text that is given in bold font style contains the steps that you are actually required to perform on the computer. Carefully perform every step given in the bold font style on your computer.

We need a folder to store our files. Let us create a folder named **`myFiles`** in the root folder indrive **`C`**. Follow these steps:

Start the Command Prompt window.

Type the following commands and strike the Enter key after each command:

```
cd\
md myFiles
cd myFiles
```

Ensure that your Command Prompt window now looks as shown in *Figure 2.1*. We are now in the folder **`myFiles`**, that is, the folder **`myFiles`** is now the current folder:

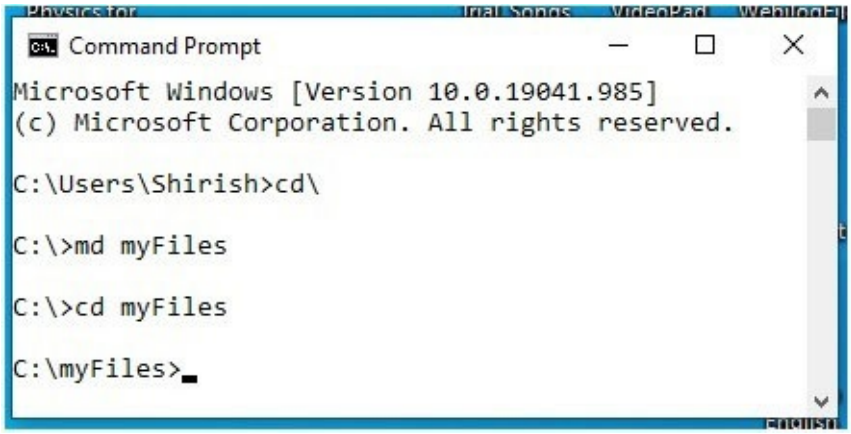

Figure 2.1: *Creating the folder myFiles to store the files*

In order to create a web page, we need to create an HTML file (it is a text file with the filename extension **`.html`**) and this can be done using the Notepad. We will create an HTML file with the filename **`hello.html`**. Type the following command and strike the Enter key:

```
notepad hello.html
```

Now, a message box pops up on the screen asking if you **`Cannot find the hello.html file. Do you want to create a new file?`** as shown in *Figure 2.2*. Click on the button **`Yes`** on this message box as we are interested in creating a new file. Now, this message box disappears from the screen and the Notepad pops up on the screen as shown in *Figure 2.3* (some text is already typed in the Notepad as shown in *Figure 2.3*.

Type the following text in the Notepad window (this is the content of the file **`hello.html`**):

```
<html>
<p>hello, world!</p>
</html>
```

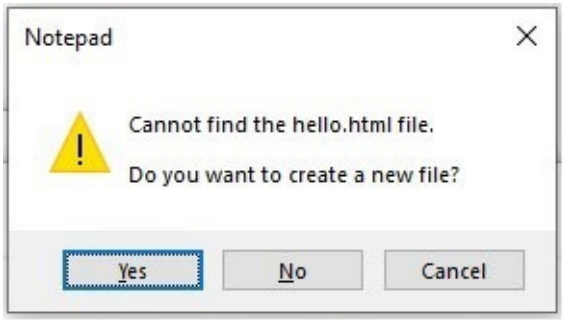

Figure 2.2: *Click on the button Yes to create the file hello.html*

Now, ensure that the Notepad window looks precisely as shown in *Figure 2.3*. If not, then make the necessary corrections:

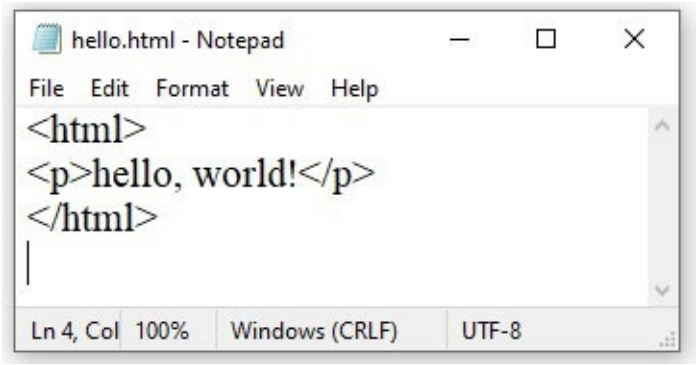

Figure 2.3: *Text is typed in the Notepad window*

Hereafter, the code is shown in the following style (instead of the preceding style):

```
01. <html>
02. <p>hello, world!</p>
03. </html>
```

Code Snippet: *2.1*

The benefit of this style is that every **LOC** (**Line of Code**) is serially-numbered. The first LOC is numbered 01, the second LOC is numbered 02, and the third LOC is numbered 03. I need these numbers for referring to these LOCs while explaining the code. For example, I would say the **`<p>`** element in LOC 2 and you need to immediately understand to which LOC I am referring. Of course, when you type this code in the Notepad window –or any other HTML editor, for that matter –, then you will not type these numbers, you will type only the text that appears on the right-hand side of the vertical, orange-colored bar.

Now, let us save this file. Choose **`File | Save`** (or strike the key combination Ctrl+S) and the file gets saved. Now, let us shut down the Notepad window. Choose **`File | Exit`**, and the Notepad shuts down politely. The Command Prompt window is still there in the background. In case it is minimized and sleeping on the taskbar, then click on it to wake up and it resurrects. Let us check the existence of the file **`hello.html`** using the **`dir`** command. Type the following command:

```
dir
```

The Command Prompt window displays the name of the file **`hello.html`** as shown in *Figure 2.4*:

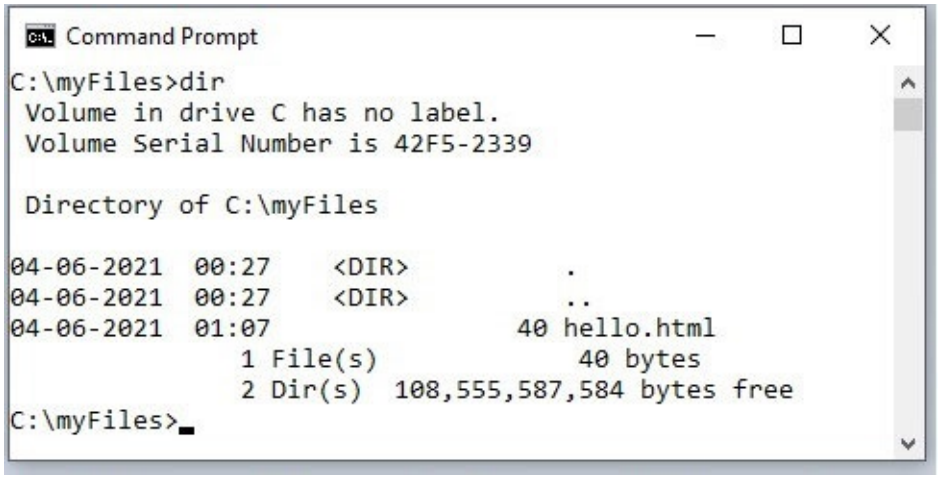

Figure 2.4: *Command Prompt window after typing the dir command*

Now, let us close the Command Prompt window. Type the following command:

```
exit
```

The Command Prompt window shuts down politely.

Now, we are all set to test ride our very first web page. Follow these steps:

1. Start your favorite web browser.
2. Type the following text in the Address bar of your web browser:
   `c:\myFiles\hello.html`

Now, our very first web page is displayed in the web browser as shown in *Figure 2.5*. This web page displays the text **`hello, world!`** which is the only content of this web page. As you read this book, you will learn how to create web pages that are full of text, graphics, and multimedia.

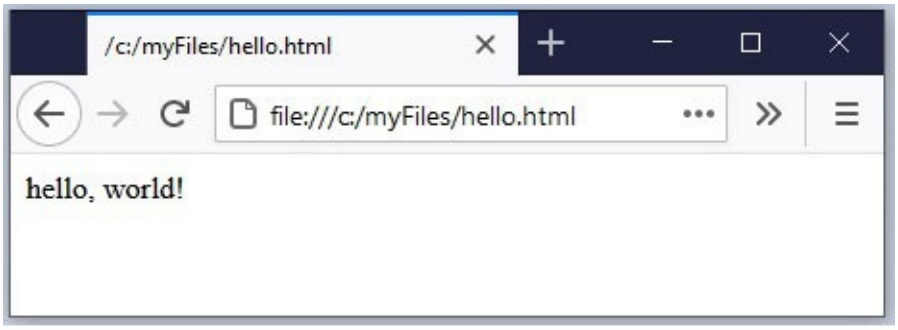

Figure 2.5: *Our very first web page is displayed successfully*

You may wish to pat on your own back for this achievement. Also, show this achievement to your friends and your neighbors. Finally, shut down the web browser by clicking on its Close window icon located at the top right-hand corner.

Now, let us analyze this HTML file for its contents. This HTML file consists of three lines.

The first line consists of the opening tag **`<html>`**. This tag signifies the beginning of the HTML file.

The third line consists of the closing tab **`</html>`**. This tag signifies the end of the HTML file.

The second line consists of a typical HTML element. In this element, the content to be displayed on the screen (**`hello, world!`**) is sandwiched between the opening tag **`<p>`** and the closing tag **`</p>`**. What is this **`p`**? Think of **`p`** as the acronym of the paragraph.

Note: A typical HTML element consists of an opening tag, a closing tag, and content that is sandwiched between the opening tag and the closing tag.

Each tag consists of a pair of angled brackets `<` and `>`. Also, notice the difference between the opening tag and the closing tag. The closing tag consists of a forward slash `/`. Let's take a look at *Figure 2.6* which shows a typical HTML element:

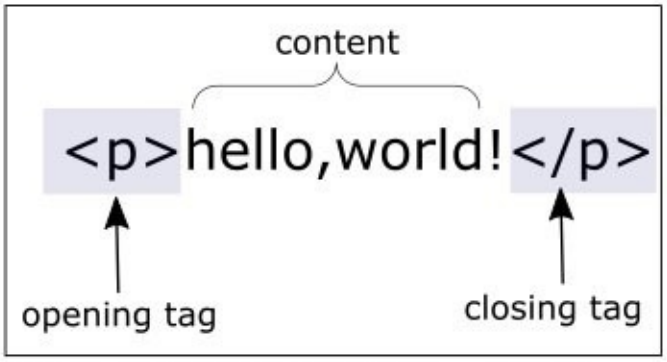

Figure 2.6: *A typical HTML element. Here, p is the name of the element.*

Our very first official web page

The web page we created in the preceding section is really nice. However, it lacks the features of an official web page. In this section, we will create a web page with all the necessary official features according to the requirements of version 5 of HTML(that is, HTML5). Firstly, let us create this web page. Open your Notepad. Type and save the following text in a file named **`official.html`** in our folder **`myFiles`**:

```
01. <!DOCTYPE html>
02. <html lang="en">
03.  <head>
04.     <meta charset="utf-8">
05.     <title>Our Official Web Page</title>
06.   </head>
07.   <body>
08.     <p>We are all set to create an official web page.</p>
09.   </body>
10. </html>
```

Code Snippet: 2.2

All this may sound Greek to you. But don't bother. We will learn about it in detail. Firstly, let us display this web page in our web browser. Open your web browser. Type the following text in the Address bar of the web browser:

```
c:/myFiles/official.html
```

Now, our official web page is displayed in the web browser as shown in *Figure 2.7*. View this web page. Notice where the title of the web page is displayed:

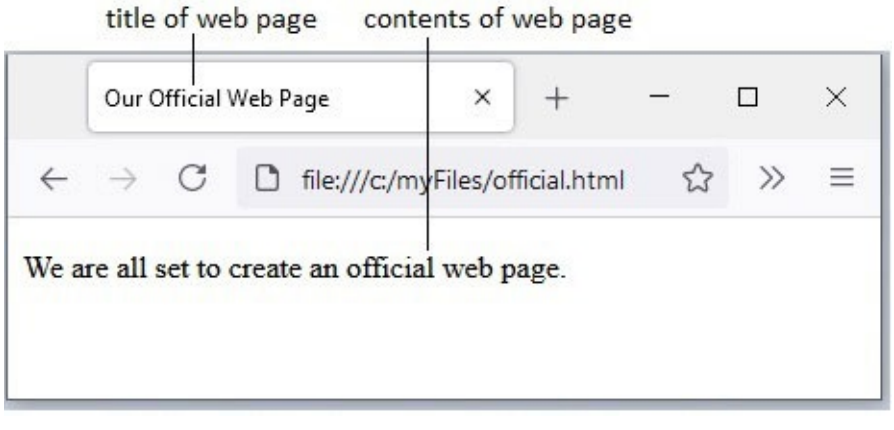

Figure 2.7: *Our official web page is displayed in the web browser*

Now, let us discuss the working of this piece of code.

LOC 1 consists of the 'DOCTYPE declaration.' It is also known as the document-type declaration. It is required.

LOC 2 consists of the opening tag **<html>** and LOC 10 consists of the closing tag **</html>**. Your HTML file is actually packed between these two tags. Everything from line 2 to line 10 is nothing but the root element. LOC 2 is the opening tag of the root element. In LOC 2, we have used the **lang** (**lang** stands for language) attribute and its value is set to **en** (**en** stands for English). This attribute tells the browser that the language English is used on this webpage. Suppose you want to use the French language on your webpage, then you should set the value of the **lang** attribute to **fr**, and so on.

LOC 10 is the closing tag of the root element. LOCs 3 to 9 form the content of this root element.

LOC 3 to LOC 6 consists of the head element.

Note: There is only one head element in an HTML document. Multiple head elements in an HTML document are not needed.

LOC 3 consists of the opening tag **<head>** of the head element. LOC 6 consists of the closing tag **</head>** of the head element. LOCs 4 and 5 form the content of this head element. Think of the head element as the preface of the web page. It includes some information about the contents of the web page; for example, the title of the web page, language (character set) used in the web page, and so on.

LOC 4 consists of the meta element. The meta element is always placed inside the head element. The meta element has only an opening tag **<meta>** and it has no closing tag such as**</meta>**. The meta element consists of metadata about the HTML document that is not displayed on the screen. Metadata means data (information) about data. This metadata is read and used by browsers, search engines, and other web services. The content of this meta element is reproduced as follows for your quick reference:

```
charset="utf-8"
```

The term **charset** is an attribute of the meta element and the term **utf-8** is the value of this attribute. We use meta elements to provide information about the character set, page description, keywords, and author of the document to browsers, search engines, and so on. The syntax of assigning a value to an attribute is as follows:

```
attribute=value
```

Generally, the value is enclosed between double quotes. Here, **utf-8** stands for Unicode Transformation Format - 8 bit and it deals with the character sets that can be used in web pages. utf-8 is very

comprehensive and can represent any character in the Unicode standard.

LOC 5 consists of the title element. The title element is mandatory.

Note: There is only one title element in an HTML document. Multiple title elements in an HTML document are not needed.

The title element consists of the opening tag **`<title>`** followed by the title of the web page, followed by the closing tag **`</title>`**. What should be the title of the web page? It is your choice. However, the title should be meaningful and should give proper information about that web page. Avoid cryptic titles like **`Mktrpc_876`** and use meaningful words in your title something like "History of South India" that clearly throws light on the contents of the web page. The title is displayed on the title bar of the web browser and it is also used by search engines. The title element is always placed inside the head element.

LOC 7 to LOC 9 consist of the body element.

Note: There is only one body element in an HTML document. Multiple body elements in an HTML document are not needed.

The body element begins with the opening tag **`<body>`** and ends with the closing tag **`</body>`**. The body element consists of everything that we want to show on our web page. Here, the content of the body element is only one line of code (LOC 8 which is nothing but a p element). But in reality, hundreds of lines of code can be found in the body element.

LOC 8 consists of a p element. We have already discussed how the p element works in the preceding section.

Note: The elements HTML, head, title, and body are called core elements owing to their importance in a web page. Every HTML document contains these four core elements.

Before concluding this lesson, let us create a template for the HTML file so that we are not required to type the same data again and again. Type and save the following text in a file named **`template.html`** and in the folder **`myFiles`**:

```
01. <!DOCTYPE html>
02. <html lang="en">
03.   <head>
04.     <meta charset="utf-8">
05.     <title> </title>
06.   </head>
07.   <body>
08.   </body>
09. </html>
```

Code Snippet: 2.3

Hereafter, whenever, we want to create an HTML file, we will open this file and save it with the desired name. This will save our labor of typing this data again and again. Also, notice that some of the LOCs are indented (which means they have a margin on the left-hand side compared to other lines which are non-indented).

Note: The indentation of code is not required but I strongly advise it as it is a very useful feature that saves us a lot of trouble in the future.

Text formatting elements

Open the file **`template.html`** and save it with the name **`p01.html`** in the folder **`myFiles`**. A few words about the name of the file **`p01.html`**. Here, **`p`** stands for practice and the number **`01`** stands for the serial number of the practice program. If you think this naming system is a boring one, you can offer some interesting names for your program.

Using a p element

If your web page consists of a lot of text, then probably we will place all this text in **`p`** elements. A single **`p`** element is sort of a single paragraph. Just as an article can have a number of paragraphs, so also an HTML document can have a number of **`p`** elements.

Note: In an HTML document, the body text is generally fitted inside the p elements.

Place the keyboard cursor in the title element and insert the title in it as follows:

```
05.     <title>Formatting the Text</title>
```

Code Snippet: 2.4

Now, place the keyboard cursor in the body element and insert the **`p`** element in it as shown:

```
07.   <body>
08.     <p>Line one.</p>
09.   </body>
```

Code Snippet: 2.5

Ensure that your complete code looks as follows:

```
01. <!DOCTYPE html>
02. <html lang="en">
03.   <head>
04.     <meta charset="utf-8">
05.     <title>Formatting the Text</title>
06.   </head>
07.   <body>
08.     <p>Line one.</p>
09.   </body>
10. </html>
```

Code Snippet: 2.6

Load this HTML document in a web browser and then you will see this sole line of text in the web browser as follows (you will also see the title in the toolbar of the browser):

```
Line one.
```

Now, insert two more **`p`** elements in this body element as shown (and save this file as **`p02.html`**):

```
07.   <body>
08.     <p>Line one.</p>
09.     <p>Line two.</p>
10.     <p>Line three.</p>
11.   </body>
```

Code Snippet: 2.7

Load this HTML document in a web browser and then you will see these three lines of text in the web browser as follows:

```
Line one.
Line two.
Line three.
```

Notice that the content of every **p** element is displayed on a fresh line. This is because **p** is a block element. You will learn something about these block elements in the next subsection.

Now, modify the **body** element as follows (and save this file as **p03.html**). The three **p** elements are now put in a single line:

```
07.   <body>
08.     <p>Line one.</p> <p>Line two.</p> <p>Line three.</p>
09.   </body>
```

Code Snippet: 2.8

Load this HTML document in a web browser and then, surprisingly, you will find that contents of these three **p** elements are displayed not in a single line but in three different lines as follows.

```
Line one.
Line two.
Line three.
```

Notice that the content of every **p** element is displayed on afresh line. This is because **p** is a block element. You will learn something more about these block elements in the subsection *Block element and inline element*.

Using h element

We use the **h** element for the heading. Think of h as an acronym for the heading. The heading of an article is always in bold and big letters, hence in an HTML document also the **h** element displays its contents in bold and big letters.

Note: In HTML documents, headings are generally fitted inside the h elements.

There are six **h** elements as follows: **h1**, **h2**, **h3**, **h4**, **h5**, and **h6**. All these **h** elements are block elements, which means the content of every **h** element is displayed on afresh line.

Note: Heading h1 has the largest font size and heading h6 has the smallest font size.

Modify the body element as follows. Insert six **h** elements in the body element as follows (and save this file as **p04.html**):

```
07.   <body>
08.     <h1>Heading one: h1</h1>
09.     <h2>Heading two: h2</h2>
10.     <h3>Heading three: h3</h3>
11.     <h4>Heading four: h4</h4>
12.     <h5>Heading five: h5</h5>
13.     <h6>Heading six: h6</h6>
14.   </body>
```

Code Snippet: 2.9

Load this HTML document in a web browser and then, you will see the six lines of text displayed in the web browser as headings of varying font size as shown in *Figure 2.8*:

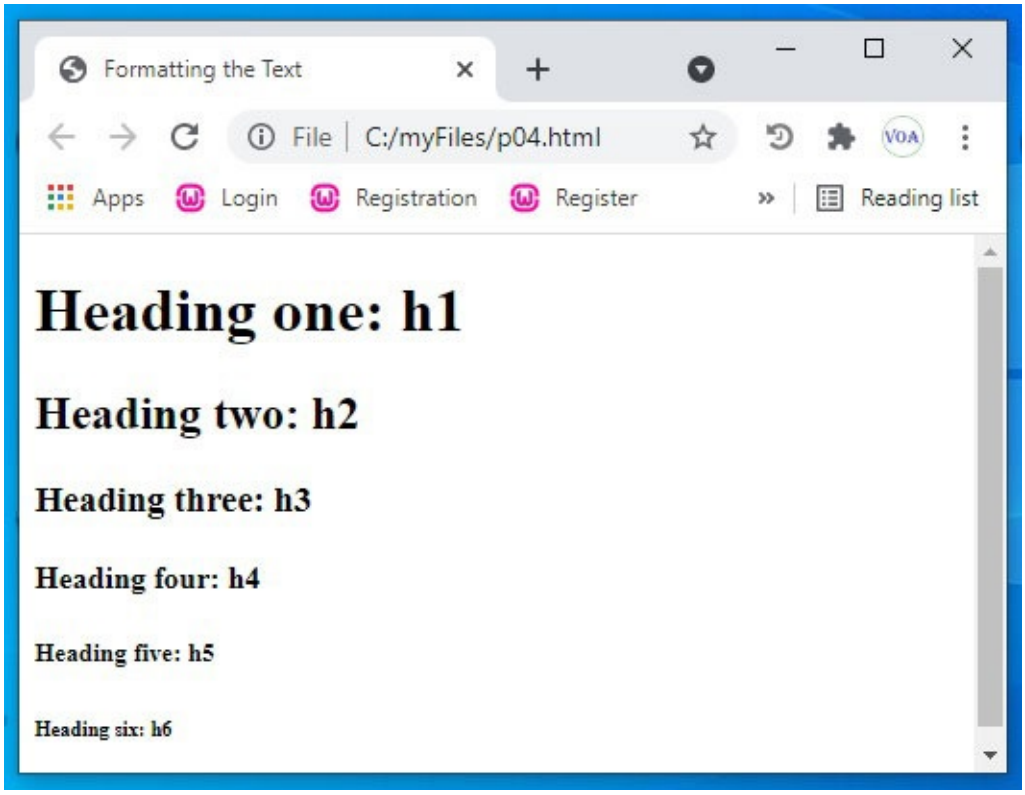

Figure 2.8: *Six headings are displayed in the web browser*

Block elements and inline elements

Every element in an HTML document is either a block element or an inline element.

Note: In the block element, its content is always displayed on a fresh (new) line. In the inline element, its content is displayed in the same line. The content of the inline element flows with the text.

We have already discussed the use of two block elements, namely, the **p** element and the **h** element. We are yet to discuss any inline element. For the sake of example, I introduce the inline element **<b>** right here and now. Notice the following line of text:

```
I want to be an expert in HTML and CSS.
```

Notice that in this line of text, the word **expert** is in bold font style. Now, we will display this line of text in a web browser. Modify the body element as follows. Insert the **p** element in the body element as follows(and save this file as **p05.html**):

```
07.   <body>
08.     <p>I want to be an <b>expert</b> in html and css.</p>
09.   </body>
10. </html>
```

Code Snippet: 2.10

Load this HTML document in a web browser and then, you will see this line of text displayed in the web browser:

I want to be an **expert** in html and css.

Notice that the word **expert** is in bold font style. Also, notice that there is no line break at the **b** element because **b** is an inline element. Think of b as an acronym for bold. The opening and closing tags of the b element are **** and ****.

Horizontal rule and line break

We can insert a horizontal rule in HTML document using the hr element. This element has only an opening tag **<hr>**. It has no closing tag. This is a block element.

We can also force a line break in HTML documents using the **br** element. This element also has only the opening tag **
**. It has no closing tag. This is an inline element.

Modify the body element as follows. Insert the two **p** elements in the body element as follows (and save this file as **p06.html**):

```
07.   <body>
08.     <p>Line one. <hr> Line two.</p>
09.     <p>This line is broken <br> into two lines.</p>
10.   </body>
```

Code Snippet: 2.11

Load this HTML document in a web browser and then, you will see the effect of **<hr>** and **
** in the web page displayed in the web browser as shown in *Figure 2.9*:

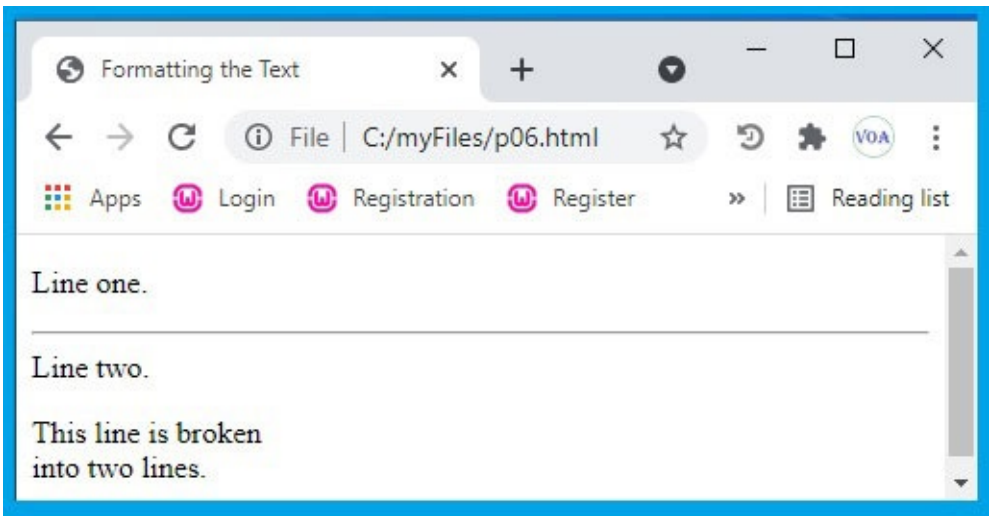

Figure 2.9: *The tags <hr> and
 are used in this web page. The tag <hr> inserts the horizontal rule and the tag
 forces the line break.*

Using various elements for desired font styles

There are various elements that are used for applying the desired font style to the text on the web page. Of course, now very powerful styles are available under CSS for formatting the text. But still, you will love these elements which can be used easily for desired font styles on the web page.

- Element **`b`** is used to make the text **bold**.
- Element **`i`** is used to make the text *italic*.
- Element **`s`** is used to strikethrough the ~~text~~.
- Element **`u`** is used to underline the text.
- Element **`small`** is used to make the font size smaller.
- Element **`mark`** is used to mark up the text.
- Element **`strong`** is used to make the text **bold**.
- Element **`em`** is used to make the text *italic*.
- Element **`sup`** is used to set the font style to $^{\text{superscript}}$.
- Element **`sub`** is used to set the font style to $_{\text{subscript}}$.

Now, let us implement these elements in an HTML document. Modify the body element as follows. Insert a few **`p`** elements in the body element as follows (and save this file as **`p07.html`**):

```
07.   <body>
08.     <p>Let us make the text <b>bold</b>.</p>
09.     <p>Let us make the text <i>italic</i>.</p>
10.     <p>Let us <s>strikethrough</s> the text.</p>
11.     <p>Let us <u>underline</u> the text.</p>
12.     <p>Let us make the text <small>smaller</small>.</p>
13.     <p>Let us <mark>mark up</mark> the text.</p>
14.     <p> Let us make the text <strong>bold</strong>.</p>
15.     <p> Let us make the text <em>italic</em></p>
16.     <p>Let us make the text <sup>superscript</sup>.</p>
17.     <p>Let us make the text <sub>subscript</sub>.</p>
18.   </body>
```

Code Snippet: 2.12

Load this HTML document in a web browser and then you will see the effect of the various elements on the font style of text displayed on the web page as shown in *Figure 2.10*:

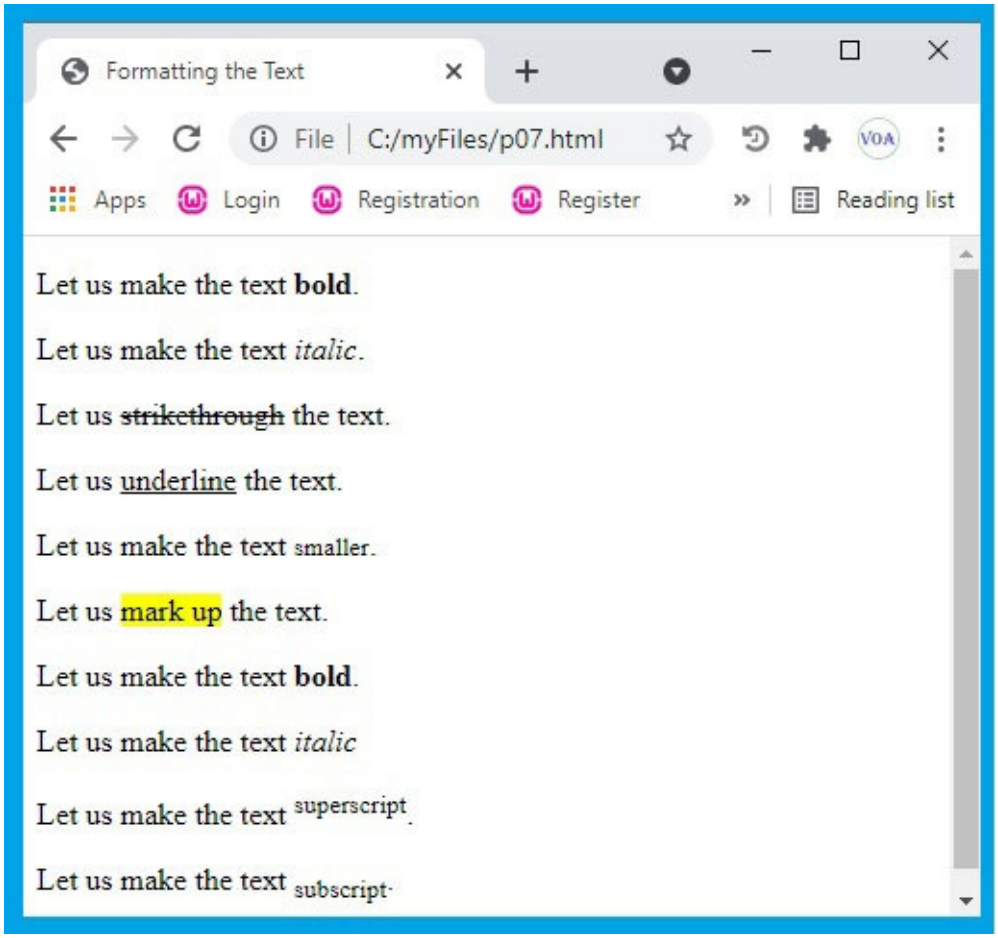

Figure 2.10: *Various elements are used to set the font styles*

Using the pre element to retain the white spaces

If we insert white spaces in our HTML document, then these white spaces are done away with. What is this white space?

Note: The output of the space bar, Tab key, and Enter key is termed white space.

Let us check how these white spaces are done away with. Modify the **`body`** element as follows. Insert the two **`p`** elements in the **`body`** element as shown (and save this file as **`p08.html`**):

```
07.   <body>
08.     <p>I      love      white      spaces.</p>
09.
10.
11.
12.     <p>Ha     Ha     Ha     Ha     Ha.</p>
13.   </body>
```

Code Snippet: 2.13

Notice that there are five spaces between two consecutive words. Also, there are three blank lines between the two **`p`** elements. Load this HTML document in a web browser and then you will get the following output on the web page:

```
I love white spaces.
Ha Ha Ha Ha Ha.
```

Notice that now there is only a single space between the two consecutive words. Also, the blank lines between the two **p** elements are also removed. This is how the web browser ruthlessly kicked out our white spaces from the HTML document.

Suppose you want to retain these white spaces. How to do it? We can do it using the element **pre**. Simply put these two **p** elements in a **pre** element and then these white spaces will be carefully preserved by the web browser. Modify the body element as follows (and save this file as **p09.html**):

```
07.   <body>
08.     <pre>
09.       <p>I      love      white      spaces.</p>
10.
11.
12.
13.       <p>Ha     Ha     Ha     Ha     Ha.</p>
14.     </pre>
15.   </body>
```

Code Snippet: 2.14

Notice that the two **p** elements are now put in a **pre** element. Load this HTML document in a web browser and then you will get the following output on the web page:

```
I      love      white      spaces.
Ha      Ha      Ha      Ha      Ha.
```

Notice that our lovely white spaces are now preserved by the web browser. This feature is very useful when you are designing advertisements and flyers with some fancy-looking matter in them.

Using the elements ins, del, and wbr

The element **ins** is used to insert the text in the existing text. The element **del** is used to delete the text from the existing text. The element **wbr** is used to break the very long word, just in case the break is required.

Firstly, let us tinker with the elements **ins** and **del**. Modify the **body** element as follows (and save this file as **p10.html**):

```
07.   <body>
08.     <p>I like <del>chicken</del><ins>mutton</ins>.</p>
09.   </body>
```

Code Snippet: 2.15

Here, in the original sentence **I like chicken**, the word **chicken** is deleted and the new word **mutton** is inserted which will take the place of **chicken**. The resulting sentence is now **I like mutton**. Load this HTML document in a web browser and the web page displays this sentence as shown in *Figure 2.11*. Notice how the word **chicken** is deleted and the word **mutton** is inserted in the text:

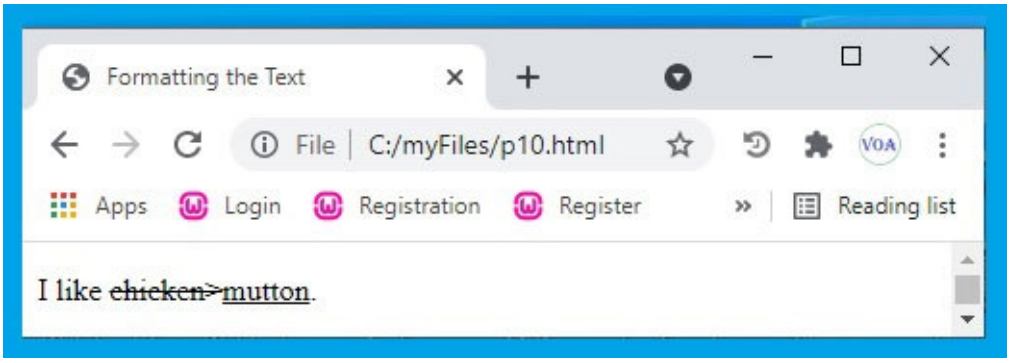

Figure 2.11: *Using the elements ins and del in an HTML document*

Now, let us experiment with the element **`wbr`**. This element breaks the very long word, provided such a break is required. Modify the **`body`** element as follows (and save this file as **`p11.html`**):

```
07.   <body>
08.     <p>This is veryveryvery<wbr>veryverylongword.</p>
09.   </body>
```

Code Snippet: 2.16

Load this HTML document in a web browser and then you will get the following output on the web page:

`This is veryveryveryveryverylongword.`

Word is not broken at the specified location. This is because the width of the web browser is comfortably large and hence the web browser has not broken this word. However, if you resize the web browser (make it too small to accommodate the word **`veryveryveryveryverylongword`**), then this word will break at the specified location.

Using special characters

We can display special characters like ©,♥, or ♠ on our web pages! If you want to insert the symbol ©, then simply type **`&#169`** in your HTML document, and then the web browser will display © in place of **`&#169`**. In this section, we will tinker with a few special characters.

Modify the **`body`** element as follows (and save this file as **`p12.html`**):

```
07.   <body>
08.     <p>This is copyright symbol: &copy and &#169</p>
09.     <p>This is registration symbol: &reg and &#174</p>
10.     <p>This is left arrow symbol: &#8592</p>
11.     <p>This is up arrow symbol: &#8593</p>
12.     <p>This is heart symbol: &#9829</p>
13.     <p>This is spade symbol: &#9824</p>
14.     <p>This is emoji smiley symbol: &#128540</p>
15.     <p>This is also emoji smiley symbol: &#128527</p>
16.   </body>
```

Code Snippet: 2.17

Load this HTML document in a web browser and the web page displays this HTML document as shown in *Figure 2.12*:

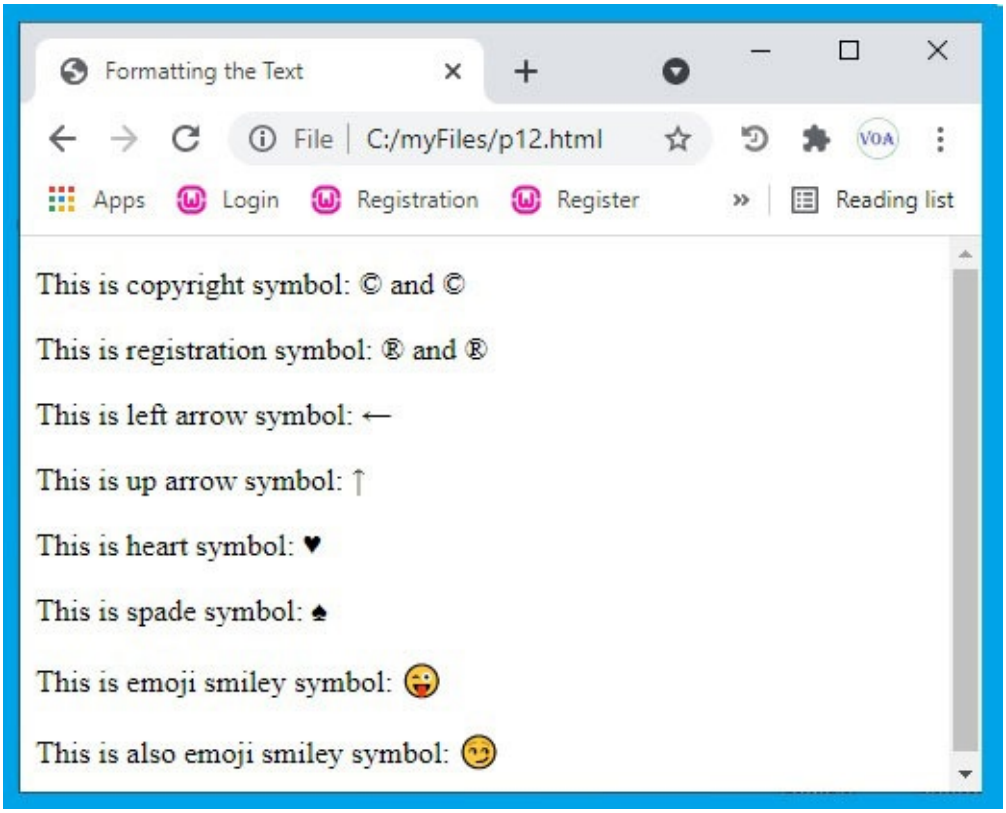

Figure 2.12: *Various symbols are displayed on the web page*

Using grouping elements

In HTML, we use the following grouping elements to group the various items:

- ** **
- **<header>**
- **<hgroup>**
- **<nav>**
- **<section>**
- **<article>**
- **<hr>**
- **<blockquote>**
- **<aside>**
- **<footer>**
- **<address>**

In the upcoming chapters of this book, we will discuss the use of these grouping elements.

Using core attributes

In HTML, we will use the following core attributes to modify or add the functionality to HTML elements:

- **id**

- **class**
- **title**
- **style**
- **dir**
- **lang**

In the succeeding chapters, you will learn how to use these attributes.

Using HTML editors

You can create an HTML file in any text editor. Notepad is the most obvious choice for Windows users. However, there are some HTML editors which offer some facilities to facilitate the creation of HTML pages. A few of these HTML editors and the website URLs to download them are listed as follows:

- Atom: **https://www.atom.io**
- Notepad++: **https://www.notepad-plus-plus.org/downloads/**
- Sublime Text: **https://www.sublimetext.com/**
- Visual Studio Code: **https://code.visualstudio.com/**
- Froala: **https://www.froala.com/**
- CoffeeCup: **https://www.coffeecup.com/html-editor/**

All these are very simple to use like Notepad and offer a number of facilities to create HTML files. Try a few of them and choose the one that you like the most.

Validate your web page

World Wide Web Consortium has provided the validation service for HTML pages. Using this service, we can check the validity of our HTML pages. In order to do so, log on to the following URL:

https://validator.w3.org/

Now, the web page of Markup Validation Service will appear on your screen. You can validate your HTML page using one of the following three methods.

Validate by URI (Uniform Resource Identifier)

We can use this method for the HTML pages which are already uploaded on websites. In order to validate your HTML page using this method, follow these steps:

1. Click on the tab **`Validate by URI`** to bring the corresponding page to the front.
2. Type the address (URI) of the web page to be validated in the edit-box Address (say, `https://www.xyz.com/mypage.html`)
3. Click on the button **`Check`**.

Now, this service checks the web page suggested by you and displays the results (which is nothing but a

list of warnings and errors) on the screen.

Validate by File Upload

We can use this method for the HTML pages which are stored on your computer. In order to validate your HTML page using this method, follow these steps:

1. Click on the tab **`Validate by File Upload`** to bring the corresponding page to the front.
2. Click on the button **`Choose File`** and the dialog box **`Open`** pops up on the screen. In this dialog box, enter the name of the HTML file to be validated and then click on the button **`Open`** provided on it. Now, the dialog box disappears from the screen and the name of this HTML file appears in front of the button **`Choose File`**.
3. Click on the button **`Check`**.

Now, this service checks the HTML page suggested by you and displays the results (which is nothing but a list of warnings and errors) on the screen.

Validate by Direct Input

We can use this method for the HTML pages when the HTML page is open in a text editor. In order to validate your HTML page using this method, follow these steps:

1. Click on the tab **`Validate by Direct Input`** to bring the corresponding page to the front.
2. Copy and paste the contents of an HTML page in the edit box titled **`Enter the Markup to validate`**.
3. Click on the button **`Check`**.

Now, this service checks the HTML page suggested by you and displays the results (which is nothing but a list of warnings and errors) on the screen.

Try to remove the warnings and errors from your web page. However, remember:

Note: Do not take the advice of validating service with religious zeal. Because, at times, you will find it difficult to satisfy the validating service. Even reputed websites have web pages that fail to validate when subjected to validation tests.

Designing the website using sitemaps and wireframes

Web design is a five-step process:

- **Fixing the goals**: Fix your goals. What is the purpose of your website? For example, if you are creating a website for eTutoring, then prepare the list of courses for which you will offer guidance on your website.
- **Fixing the scope**: Fix the scope of the website. For example, write down whether you want to teach everything on your website or just want to conduct demo examinations.
- **Sitemap and wireframe creation**: Prepare a sitemap and wireframe for your website. In this step,

you will decide the overall layout of the website and the number of pages on your website.

- **Content creation**: Next, create the content for your website. For example, if you plan to teach everything from scratch to students, then prepare the video lectures or written notes or guides covering the complete syllabus. If you just want to conduct the demo examinations, then prepare the question papers so that online demo examinations can be conducted.
- **Visual elements**: Prepare a list of visual elements that you plan to put on your website. A picture is worth a thousand words, therefore, always put a few appealing pictures on your website.

The sitemap for our website project

There are two types of sitemaps we use in website making:

- A sitemap to assist the building of the website.
- A sitemap to improve the **SEO** (**Search Engine Optimization**) of the website.

Right now, we are dealing with the sitemap of the first type. We will deal with the sitemap of the second type in the latter part of the book.

The sitemap of a website (of the first type) is nothing but a block diagram of a website. *Figure 2.13* shows the sitemap for our website project. This website (that we intend to develop as a project in this book) is already placed at the URL **www.webilog.in/tbs** and you may wish to view it before proceeding further. Also, Figures 1.2, 1.3, and 1.4 show the images of the **`Home`** page, **`Careers`** page, and **`Contact Us`** page respectively. You may wish to view these figures. In this figure (that is, Figure 2.13) you can see that there are three columns of boxes. The first column is regarding the design of the **`Home`** page. The second column is regarding the design of the **`Careers`** page. The third column is regarding the design of the **`Contact Us`** page. Each box in a column represents some item to be displayed on that web page.

Consider the first column of boxes. The topmost box **`Home`** is nothing but the name of this page. The second box **`Running Line`** represents the line of text that runs continuously from right to left. The third box **`Name of Company`** represents the name of the company **`Truly Best Software`** which will be displayed on the **`Home`** page, just below the running line of text. Proceeding in this manner, you will find the items **`Division Names`**, **`Image`**, **`Division Details`**, **`Tagline`**, and **`Copyright Notice`**, as shown in the following figure:

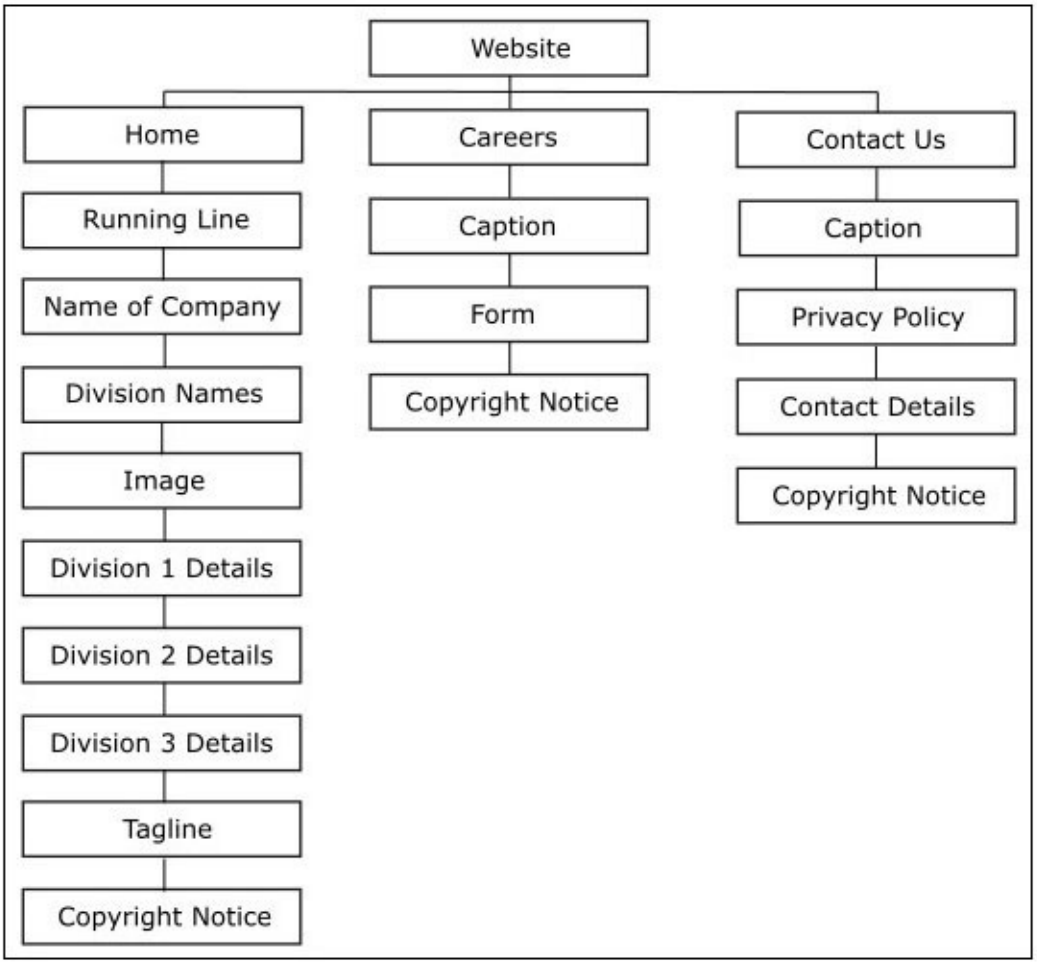

Figure 2.13: *Sitemap for our website project*

In the second column of boxes, you will find the items **`Careers`**, **`Caption`**,**`Form`**, and **`Copyright Notice`**. This column deals with the **`Careers`** page.

In the third column of boxes, you will find the items **`Contact Us`**, **`Caption`**, **`Privacy Policy`**, **`Contact Details`**, and **`Copyright Notice`**. This column deals with the **`Contact Us`** page.

Wireframes for our website project

A wireframe is nothing but a blueprint of the webpage. The various elements mentioned in the sitemap are placed at proper locations in the wireframe. Also, the space allotted to every item is proportional to its actual size. The wire frame for the Home page of our website project is shown in *Figure 2.14*. You will notice that all the items mentioned in the first column in *Figure 2.13* are placed at the proper locations in *Figure 2.14*. Compare the wireframe shown in *Figure 2.14* with the image of the webpage (Home page) shown in *Figure 1.2*:

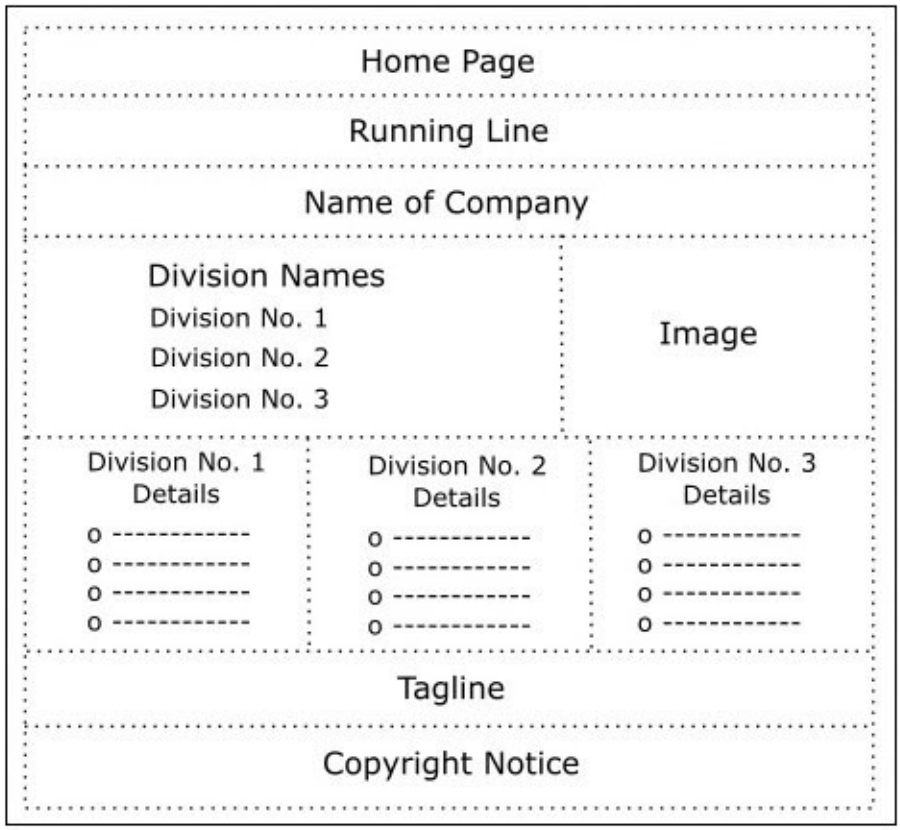

Figure 2.14: *Wireframe for the Home page of our website project*

The wireframe for the Careers page of our website project is shown in *Figure 2.15*. Verify that all the items mentioned in the second column in *Figure 2.13* are placed at the proper locations in *Figure 2.15*. Also, compare the wireframe shown in *Figure 2.15* with the image of the webpage (Careers page) shown in *Figure 1.3*.

Figure 2.15: *Wireframe for the Careers page of our website project*

The wireframe for the Contact Us page of our website project is shown in *Figure 2.16*. Verify that all the items mentioned in the third column in *Figure 2.13* are placed at the proper locations in *Figure 2.16*. Also, compare the wireframe shown in *Figure 2.16* with the image of the webpage (Contact Us page)

shown in *Figure 1.4*:

Figure 2.16: *Wireframe for the Contact Us page of our website project*

Conclusion

In this chapter, we learned to create a very basic but fully functional webpage. We also learned the usage of text formatting elements, block elements, inline elements, special characters, core attributes, HTML editors, and webpage validating services by W3C. Finally, we learned to use sitemaps and wireframes in order to design a website. In the next chapter, you will learn to use the links for linking within the same webpage, linking to other webpages, and linking to other websites. You will also add links to our website project.

Further readings/references

- *HTML & CSS: The Complete Reference*, by Thomas A. Powell, 5/e, McGraw Hill, 2010.
- *Build Your Own Website the Right Way using HTML & CSS*, by Ian Lloyd, 2/e, Sitepoint, 2008.
- *HTML: A Beginner's Guide*, by Wendy Willard, 4/e, McGraw Hill, 2009.
- *HTML5: The Missing Manual*, by Matthew Mas Donald, 2/e, 2014.
- *Creating Your First Page*, this fascinating chapter (think of this chapter as a magazine article) is available at: **https://www.oreilly.com/library/view/creating-a-website/9781449374525/ch01.html**.

CHAPTER 3

Using Images, Audio, Video, and Links

Introduction

In this chapter, you will learn to insert an image on your web page, insert an audio clip on your web page, insert a video clip on your web page, and insert links on your web page. You will learn to create links to web pages on your website, create links to email IDs, create in-page links, and create links to other websites. You will learn the difference between absolute URLs and relative URLs. Finally, you will create three HTML files as a part of your website project.

Structure

In this chapter, we will discuss the following topics:

- Adding an image to a web page
- Adding audio to a web page
- Adding a video to a web page
- Displaying a web page on a web page
- Linking to other websites
- Linking to the email ID
- Linking to other web pages on the same website
- Directory, subdirectory, parent directory, root directory, and current directory
- Creating the in-page links
- Creating your website project

Objectives

After reading this chapter, you will be able to add an image to a web page, add an audio clip to a web page, add a video clip to a web page, and add various types of links to a web page. You will be able to create links to other websites, to email IDs, and to various web pages of your website. You will also learn to create in-page links. Finally, you will create three HTML files as a part of your website project.

Adding an image to a web page

A picture is worth a thousand words. Therefore, posting a picture on the webpage is a fantastic idea to hook the readers to your webpage. In the beginning, web browsers used to be text-based only. The very first web browser that supported images was Mosaic which made its appearance in 1993 and became

extremely popular.

Take your fantastic selfie and put the image in the folder **myFiles** with the name **pic01.jpg**. Do not bother about the size and memory requirement of the image, anything will do. Here, I will use any available image on the Internet for **pic01.jpg**.

Open the file **template.html** and save it with the name **p15.html** in the folder **myFiles**. The last program we saved was **p12.html** and this program is named **p15.html**, purportedly leaving some space so that more programs can be appended to the preceding chapter.

Place the keyboard cursor on the title element and insert the title in it as follows:

```
05.     <title>Using Images, Audio, Video and Links</title>
```

Code Snippet: 3.1

Throughout this chapter, we will use this title.

Now, place the keyboard cursor in the body element and insert the two LOCs in it as follows:

```
07.   <body>
08.    <h1>A programmer at work!</h1>
09.    <img src="pic01.jpg" height="200" width="400" alt="Programmer at Work">
10.   </body>
```

Code Snippet: 3.2

Ensure that your complete code looks as shown in the following code snippet:

```
01. <!DOCTYPE html>
02. <html lang="en">
03.   <head>
04.     <meta charset="utf-8">
05.     <title>Using Images, Audio, Video and Links</title>
06.   </head>
07.   <body>
08.    <h1>A programmer at work!</h1>
09.    <img src="pic01.jpg" height="200" width="400" alt="Programmer at Work">
10.   </body>
11. </html>
```

Code Snippet: 3.3

Load this HTML document in a web browser and then you will see a line of text followed by a beautiful image as shown in *Figure 3.1*:

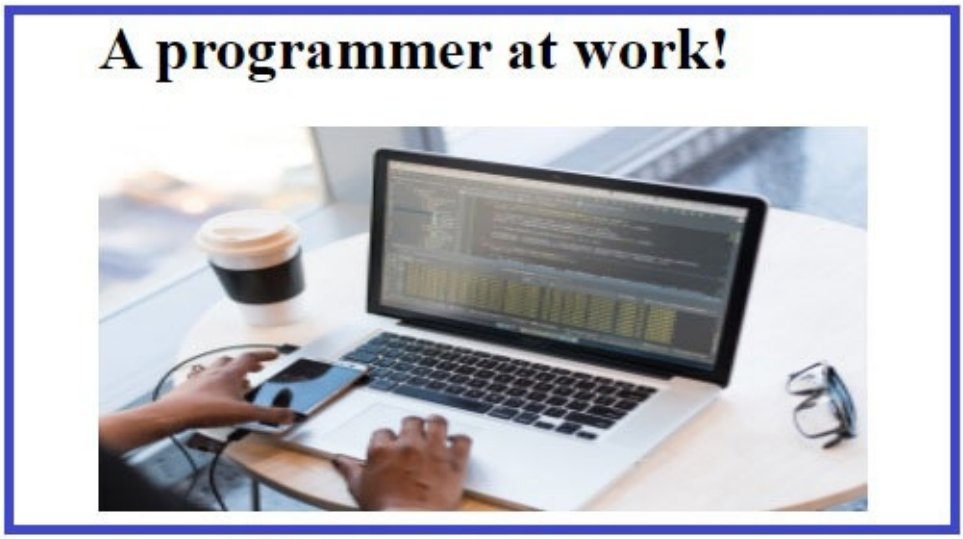

Figure 3.1: *Image is displayed on the web page successfully*

Now, let us discuss the working of this code.

LOC 8 displays the line of text **`A Programmer at Work!`** in bold and large font style. This is so because the element used here is **`<h1>`** which displays the text in bold and large font size. Let's take a look at the section *Using h Element* in *Chapter 2 (Creating the Web Pages)* for details about the **`h1`** element.

LOC 9 displays the image of a programmer at work. This LOC consists of the element **`<img>`**.

The **`img`** element is an empty element and hence, it has no **`eng`** tag like **`</img>`**.

The important attributes of the **`img`** element are: **`src`**, **`alt`**, **`height`**, and **`width`**. We are required to assign the suitable values to these attributes as follows:

Attribute	Value
`src`	Image file name with path
`alt`	Description of the image
`height`	Desired height of the image on a webpage in pixels
`width`	Desired width of the image on webpage in pixels

Table 3.1: *suitable values for the attributes*

The attributes **`src`** and **`alt`** are required. Other attributes are optional. The **`src`** attribute tells the web browser the name of the image file and also its location. If an image file is placed in the current folder, then the path name is not required, only the file name will do (as is the case in LOC 9).

The attribute **`alt`** provides the text to the web browser which is displayed on the screen just in case the image fails to appear on the screen; **`alt`** stands for **alternative text**.

The **`height`** and **`width`**attributes offer us an easy way to resize the image the way we want. We set their values in pixels.

Various image formats

Here, we displayed a jpg image on a webpage. But jpg is not the only format that is supported by web

browsers. The various image formats supported by web browsers are listed as follows:

Image format	Filename extension
APNG (Animated Portable Network Graphics)	`.apng`
GIF (Graphics Interchange Format)	`.gif`
ICO (Microsoft Icon)	`.ico`, `.cur`
JPEG (Joint Photographic Expert Group)	`.jpg`, `.jpeg`, `.jtif`, `.pjpeg`, `.pjp`
PNG (Portable Network Graphics)	`.png`
SVG (Scalable Vector Graphics)	`.svg`

Table 3.2: *image formats supported by web browsers*

You can choose the format that is suitable for your requirement.

Adding audio to a web page

Now, let us add audio to a webpage. Place an mp3 file of your favorite song in the folder **`myFiles`** and rename that file as **`music01.mp3`**.

Open the file **`template.html`** and save it with the name **`p16.html`** in the folder **`myFiles`**. Update the title element as mentioned in the first section of this chapter.

Now, place the keyboard cursor in the body element and insert the few LOCs in it as shown in the following code snippet:

```
07.   <body>
08.    <h1>Listen to My Favourite Song</h1>
09.     <audio controls>
10.       <source src="music01.mp3" type="audio/mpeg">
11.       Your browser does not support the audio element.
12.     </audio>
13.   </body>
```

Code Snippet: 3.4

Load this HTML document in a web browser and then you will see a line of text followed by an audio player as shown in *Figure 3.2*:

Figure 3.2: *Audio is being played on the web page. Click on the play button to listen to your favorite song.*

Now, let us discuss the working of this code. LOC 8 displays a line of text **`Listen to My Favorite`**

Song by making use of the **<h1>** element. LOCs 9-12 consist of the **<audio>** element that is responsible for the functioning of the audio player shown in *Figure 3.2*. LOC 9 consists of an opening tag **<audio>**. LOC 12 consists of a closing tag **</audio>**. LOC 11 consists of a line of the text **Your browser does not support the audio element** which is displayed on the screen just in case your browser doesn't support the **audio** element. However, all the major browsers do support the audio element.

LOC 10 is the real stuff of this audio burger. LOC 10 consists of an empty element **<source>**. The **<source>** element tells the web browser the name of the audio file to be played and also that the type of this media file is audio/MPEG. The **<source>** element has two important attributes, namely, **src** and **type**. The name of the audio file with pathname is assigned to attribute **src**. However, if an audio file is placed in the current folder, then a pathname is not required, only file name is sufficient (as is the case in this program). The attribute **type** tells the web browser the type of this media file. Here, the text audio/MPEG is assigned to this attribute in LOC 10 and this text tells the web browser that the file to be played is an audio file.

Adding a video to a web page

In this section, we will add a video to a web page. Place an mp4 file of your choice in the folder **myFiles** and rename that file as **video01.mp3**.

Open the file **template.html** and save it with the name **p17.html** in the folder **myFiles**. Update the title element as mentioned in the first section of this chapter.

Now, place the keyboard cursor in the body element and insert a few LOCs in it as shown in the following code snippet:

```
07.   <body>
08.    <h1>What a Fantastic Video!</h1>
09.     <video width="400" height="300" controls>
10.       <source src="video01.mp4" type="video/mp4">
11.       Your browser does not support the video element.
12.     </video>
13.   </body>
```

Code Snippet: 3.5

The following is the representation of a video being played on the web page:

Figure 3.3: *Video is being played on the web page. Click on the play button to start the video*

Load this HTML document in a web browser and then you will see a line of text followed by an audio player as shown in *Figure 3.3*.

Now, let us discuss the working of this code. LOC 8 displays a line of text **`What a Fantastic Video!`** by making use of the **`<h1>`** element. LOCs 9-12 consist of the **`<video>`** element that is responsible for the functioning of the video player shown in *Figure 3.3*. LOC 9 consists of an opening tag **`<video>`**. LOC 12 consists of a closing tag **`</video>`**. LOC 11 consists of a line of the text **`Your browser does not support video element`** which is displayed on the screen just in case your browser doesn't support the video element. However, all the major browsers do support the **`video`** element.

LOC 10 is the real stuff of this video burger. LOC 10 consists of an empty element **`<source>`**. The **`<source>`** element tells the web browser the name of the video file to be played and also that the type of this media file is video/mp4. The **`<source>`** element has two important attributes, namely, **`src`** and **`type`**. The name of the video file with pathname is assigned to the attribute **`src`**. However, if the video file is placed in the current folder, then the pathname is not required, only the filename is sufficient (as is the case in this program). The attribute **`type`** tells the web browser the type of this media file. Here, the text video/mp4 is assigned to this attribute in LOC 10 and this text tells the web browser that the file to be played is a video file.

Displaying a web page on a web page

We can display a web page (or web pages) on a web page using the **`<iframe>`** element. Let us see how to do it. Open the file **`template.html`** and save it with the name **`p18.html`** in the folder **`myFiles`**. Update the title element as mentioned in the first section of this chapter.

Now, place the keyboard cursor in the body element and insert a couple of LOCs in it as shown in the following code snippet:

```
07.   <body>
08.    <h2>Website of Webilog:</h2>
09.     <iframe src="http://www.webilog.in" width="600" height="200"></iframe>
10.   </body>
```

Code Snippet: 3.6

Load this HTML document in a web browser and then you will see the home page of the website **www.webilog.in**displayed in a small window on your web page as shown in *Figure 3.4*:

Figure 3.4: *A web page is inserted intoa web page.*

Now, let us discuss the working of this code. LOC 8 displays a line of text **`Website of Webilog`**: by making use of the **`<h2>`** element. LOC 9 consists of the <iframe> element that is responsible for the insertion of the home page of the website "www.webilog.in" in our web page as shown in *Figure 3.4*. LOC 9 consists of both the opening tag **`<iframe>`** as well as the closing tag **`</iframe>`**. The term **`iframe`** stands for the **inline frame**. The element **`<iframe>`** has three important attributes, namely, **`src`**, **`width`**, and **`height`**. We are required to assign suitable values to these attributes as follows:

Attribute	Value
`src`	The URL of the website to be displayed on our web page.
`height`	The desired height of the window in which the website is to be displayed.
`width`	The desired width of the window in which the website is to be displayed.

Table 3.3: *suitable values for the attributes*

Here, we have assigned the text string **`http://www.webilog.in`** to the attribute **`src`** and this text string is nothing but the URL of the website of the webilog company. The values of **`height`** and **`width`** are set in pixels.

You can display a number of web pages in small windows on your web page. Let us display two web pages on a web page at a time. Open the file **`template.html`** and save it with the name **`p19.html`** in the folder **`myFiles`**. Update the title element as mentioned in the first section of this chapter.

Now, place the keyboard cursor in the **`body`** element and insert a few LOCs in it as shown in the following code snippet:

```
07.   <body>
08.    <h2>Website of Webilog:</h2>
09.     <iframe src="http://www.webilog.in" width="600" height="200"></iframe>
10.     <br> <br>
11.     <h2>Website of Astrologer Chandrakant Jadhav:</h2>
12.     <iframe src="http://www.astrojadhav.org" width="600" height="200"></iframe>
13.   </body>
```

Code Snippet: 3.7

Load this HTML document in a web browser and then you will see the home page of the website **www.webilog.in**displayed in a small window on your web page as shown in *Figure 3.5*:

Figure 3.5: *Two web pages are displayed on oneweb page at a time.*

We have already discussed the working of the **<iframe>** element. Here, instead of one, two **<iframe>** elements are used resulting in the display of two web pages at a time on our web page. The element **
** used in LOC 10 inserts a blank line. There are actually two **
** elements in LOC 10 that insert two blank lines and this is done for the want of desired spacing between the two windows that display the websites.

Linking to other websites

Linking is a very important aspect of an HTML document. In fact, the web was basically developed as a network of linked documents. In this section, we will create a link to an external website**https://in.bpbonline.com** which is the website of BPB Publications, New Delhi.

Open the file **template.html** and save it with the name **p20.html** in the folder **myFiles**. Update the title element as mentioned in the first section of this chapter.

Now, place the keyboard cursor in the body element and insert a couple of LOCs in it as shown in the

following code snippet:

```
07.   <body>
08.    <h2>Link to BPB Publications, New Delhi.</h2>
09.    <a href="https://in.bpbonline.com">Home Page of BPB Publications.</a>
10.   </body>
```

Code Snippet: 3.8

Load this HTML document in a web browser and then you will see a couple of lines of text on your web page as shown in *Figure 3.6*:

Figure 3.6: *Link to go to the home page of BPB Publications, New Delhi.*

The first line of text is the output of the **<h2>** element used in LOC 08. The second line of text is the output of the **<a>** element used in LOC 09. This second line of text is not an ordinary line of text, it is a link. Notice that it is underlined and its color is blue. These are distinctive features of a typical link on any web page. If you place a mouse cursor on this link, its shape will change from a pointer icon to a raised finger icon. Click on this link and within a few seconds, you will be taken to the home page of the website of BPB Publications (provided that your machine is connected to the Internet) as shown in *Figure 3.7*:

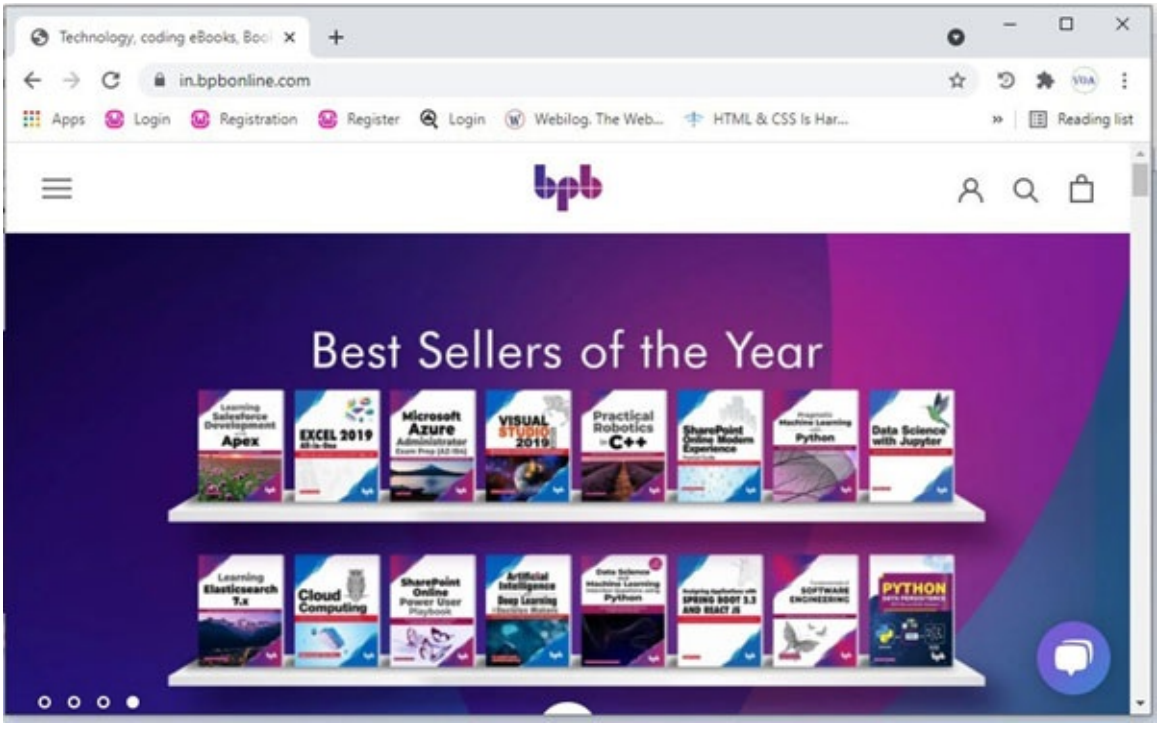

Figure 3.7: *Home page of the website of BPB Publications, New Delhi.*

After viewing this website for a while, click on the Back button (located on the left-hand side of the Address bar) on the Address bar and you will be taken back to your web page as shown in *Figure 3.6*.

Now, let us discuss the code used in this program. The element **<a>** (**a** stands for **anchor**) is used for creating the hyperlinks. The element **<a>** has only one important attribute and it is **href**. The value we

assign to the attribute **href** is nothing but the URL of the destination. For example, in this program, we have assigned the value **https://in.bpbonline.com** to the attribute **href** and this value is nothing but the URL of the website of BPB Publications. The text between the opening tag **<a>** and closing tag **** is displayed on the web page as the linked text in blue color with an underline. When we click on this linked text, we are taken to the specified destination.

Linking to the email ID

We can create a link to an email ID. In this section, we will create a link to the email ID**lina.the.crazy@gmail.com**.

Open the file "template.html" and save it with the name **p21.html** in the folder **myFiles**. Update the title element as mentioned in the first section of this chapter.

Now, place the keyboard cursor in the **body** element and insert a couple of LOCs in it as shown in the following code snippet:

```
07.   <body>
08.    <h2>Link to Email Id of Lina.</h2>
09.    <a href="mailto:lina.the.crazy@gmail.com">Send Email to Lina</a>
10.   </body>
```

Code Snippet: 3.9

Load this HTML document in a web browser and then you will see a couple of lines of text on your web page as shown in *Figure 3.8*:

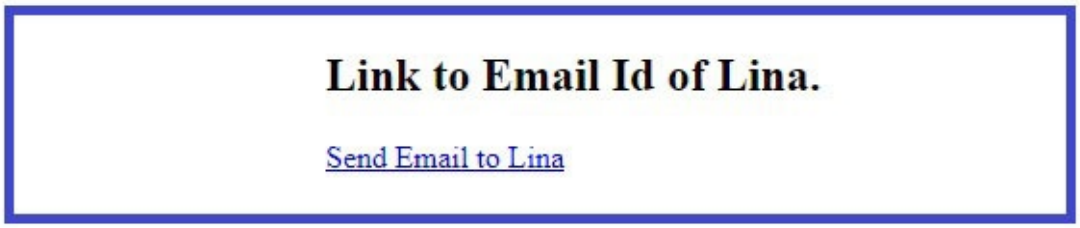

Figure 3.8: *Link to the email ID of Lina is created here.*

The first line of text is nothing but the output of the element **<h2>** used in LOC 08. The second line of text is nothing but the output of the element **<a>** used in LOC 09. This second line of text is not an ordinary line of text; it is a link. You can easily identify a link as it is blue in color and it is underlined. Click on this link and within a few seconds, your default email software opens with the email ID of Lina already typed in the edit box **To**. Now, you can type the desired content of the email as usual in the edit boxes provided and send this email to Lina.

Linking to other web pages on the same website

In this section, you will learn to create links to other web pages on the same website. In what follows now, we will create three HTML documents, namely, index.html, **careers.html**, and contact.html. The file **index.html** will represent the Home page. The file **careers.html** will represent the Careers page. The file **contact.html** will represent the Contact Us page. We will create two links on every web page so that by clicking on the corresponding link, we can access the desired web page, as follows.

On the Home page, we will create the links **Careers** and **Contact Us**. When we click on the **Careers** link, we will go to the Careers page, and when we click on the link **Contact Us,** we will go to the Contact Us page.

On the Careers page, we will create the links **Home** and **Contact Us**. When we click on the **Home** link, we will go to the Home page, and when we click on the link **Contact Us,** we will go to the Contact us page.

On the Contact Us page, we will create the links **Home** and **Careers**. When we click on the **Home** link, we will go to the Home page, and when we click on the **Careers** link, we will go to the Careers page.

In what follows now, we will create three files, namely, **index.html**, **careers.html**, and **contact.html**, one by one.

Firstly, let us create the file **index.html**. Open the file **template.html** and save it with the name **index.html** in the folder **myFiles**. Update the title element as mentioned in the first section of this chapter.

Now, place the keyboard cursor in the **body** element and insert a few LOCs in it as shown in the following code snippet:

```
07.   <body>
08.    <h2>This is Home Page of Our Website</h2>
09.    <a>Home</a>
10.    <a href="careers.html">Careers</a>
11.    <a href="contact.html">Contact Us</a>
12.   </body>
```

Code Snippet: 3.10

Now, let us create the file careers.html. Open the file **template.html** and save it with the name **careers.html** in the folder **myFiles**. Update the title element as mentioned in the first section of this chapter.

Now, place the keyboard cursor in the **body** element and insert a few LOCs in it as follows:

```
07.   <body>
08.    <h2>This is Careers Page of Our Website</h2>
09.    <a href="index.html">Home</a>
10.    <a>Careers</a>
11.    <a href="contact.html">Contact Us</a>
12.   </body>
```

Code Snippet: 3.11

Finally, let us create the file **contact.html**. Open the file **template.html** and save it with the name **contact.html** in the folder **myFiles**. Update the title element as mentioned in the first section of this chapter.

Now, place the keyboard cursor in the **body** element and insert a few LOCs in it as follows:

```
07.   <body>
08.    <h2>This is Contact Us Page of Our Website</h2>
09.    <a href="index.html">Home</a>
10.    <a href="careers.html">Careers</a>
11.    <a>Contact Us</a>
12.   </body>
```

Code Snippet: 3.12

Now, let us test these files. Load the HTML document **`index.html`** in the web browser and Home Page appears on the screen as shown in *Figure 3.9*. Notice that a couple of lines of text are displayed on this web page (*Figure 3.9*):

Figure 3.9: *Home Page is displayed on the screen. Click on the link Careers and Careers Page will be displayed on the screen (Figure 3.10). Click on the link Contact Us and the Contact Us Page will be displayed on the screen (Figure 3.11).*

The first line **`This is Home Page of Our Website`** is the output of the **`<h2>`** element used in LOC 08. The second line consists of three phrases: **`Home`**, **`Careers`**, and **`Contact Us`**. The phrase Home is just ordinary text. However, the remaining two phrases Careers and ContactUs are not ordinary text but links; and you must have correctly guessed this from their appearance. The phrases Careers and Contact Us are blue coloured and underlined as these phrases are links. The first phrase Home is not linked because you are already on the Home Page and you do not need any link to go to Home Page.

Click on the link **`Careers`** and **`Careers`** page and a page appears on the screen (it replaces the **`Home`**page) as shown in *Figure 3.10*:

Figure 3.10: *Careers Page is displayed on the screen. Click on the link Home and the Home Page will be displayed on the screen (Figure 3.9). Click on the link Contact Us and the Contact Us page will be displayed on the screen (Figure 3.11).*

If you see the Address bar, then you will notice the address of this web page to be **`C:/Myfiles/careers.html`** as expected. A couple of lines of text are displayed on this web page. The first line of text **`This is Careers Page of Our Website`** is the output of the **`<h2>`** element used in the LOC 08. The second line consists of three phrases: **`Home`**, **`Careers`**, and **`Contact Us`**. The second phrase **`Careers`** is just ordinary text. However, the remaining two phrases **`Home`** and **`ContactUs`** are not ordinary text but links, and you must have correctly guessed this from their appearance. The phrases **`Home`** and **`Contact Us`** are blue-coloured and underlined as these phrases are links. The second phrase Careers is not linked because you are already on the Careers page and you do not need any link to go to the Careers page.

Click on the link **`Contact Us`** and the Contact Us page appears on the screen (it replaces the Home Page) as shown in *Figure 3.11*:

Figure 3.11: *Contact Us Page is displayed on the screen. Click on the link Home and the Home Page will be displayed on the screen (Figure 3.9). Click on the link Careers and Careers Page will be displayed on the screen (Figure 3.10).*

If you see the Address bar, then you will notice the address of this web page to be **`C:/Myfiles/contact.html`** as expected. A couple of lines of text are displayed on this web page. The first line of text **`This is Contact Us Page of Our Website`** is the output of the **`<h2>`** element used in the LOC 08. The second line consists of three phrases: **`Home`**, **`Careers`** and **`Contact Us`**. The third phrase **`Contact Us`** is just ordinary text. However, the remaining two phrases **`Home`** and **`Careers`** are not ordinary text but links, and you must have correctly guessed this from their appearance. The phrases **`Home`** and **`Careers`** are blue color and underlined as these phrases are links. The third phrase **`Contact Us`** is not linked because you are already on the Contact Us page and you do not need any link to go to the Contact Us page.

Play with all the links for a while and see how a new web page is displayed when we click on a corresponding link.

Now, let us discuss the working of this code. Let us begin with the code of file **`index.html`** and the relevant portion of this code is reproduced for your quick reference as follows:

```
07.   <body>
08.    <h2>This is Home Page of Our Website</h2>
09.    <a>Home</a>
10.    <a href="careers.html">Careers</a>
11.    <a href="contact.html">Contact Us</a>
12.   </body>
```

Code Snippet: 3.13

This code after execution displays the web page shown in *Figure 3.9*. This web page displays two lines of text. The first line of text is nothing but the output of LOC 08 which consists of an element **`<h2>`**. The second line of text is the output of three LOCs, namely, LOC 09, LOC 10, and LOC 11. This second line of text consists of three words, namely, **`Home`**, **`Careers`**, and **`Contact Us`**. As discussed earlier, the first word is simply text, and the remaining two words are links. The word Home is the output of LOC 09.

LOC 09 consists of element **`<a>`** which is used for creating the links but here no link is created because we have not assigned any value to the attribute **`href`**, that is, we have not set the **`href`** attribute. As a result, the text enclosed between **`<a>`** and **`</a>`**behaves like ordinary text. Right now, we are already on the Home page, and hence no link is required to go to the Home page, that's why we have not created a Home link but left it as ordinary text.

LOC 10 consists of the element **`<a>`** and this element creates the link **`Careers`**. When we click on this link, then we are simply taken to the Careers page because in this element, we have set the **`href`** attribute with the value **`careers.html`** and this value is nothing but the name of the HTML file that creates the web page Careers page.

Similarly, LOC 11 consists of element **`<a>`,** and this element creates the link **`Contact Us`**. When we click on this link, then we are simply taken to the Contact Us page because in this element, we have set the **`href`**

attribute with the value **`contact.html`** and this value is nothing but the name of the HTML file that creates the web page Contact Us page.

Here, we have assigned only the names of HTML files to the attribute **`href`** because all these files are in the same folder. But, just in case, if these files are placed in different folders, then you are required to prefix the filename with a pathname. Remember:

If files to be accessed are placed in different folders, then we are required to prefix the filename with a pathname.

In the next section, we will discuss how to prefix the filename with the pathname when files are placed in different folders.

The code in the files **`careers.html`** and **`contact.html`** is very much similar to the code in the file **`index.html`** and hence, there is no need to repeat that discussion.

Directory, subdirectory, parent directory, root directory, and current directory

We have already created the folder **`myFiles`** under the root directory of **`C:`** drive. In what follows now, we will create a few nested folders. Start the Command Prompt window and see to it that the folder **`myFiles`** is the current folder as shown in *Figure 3.12*:

Figure 3.12: *Command Prompt window. The folder myFiles is now the current folder.*

Now, give the following commands and strike the Enter key after each command:

```
md A
cd A
md B
cd B
md C
cd C
md D
cd D
```

Ensure that your Command Prompt window looks as shown in *Figure 3.13*:

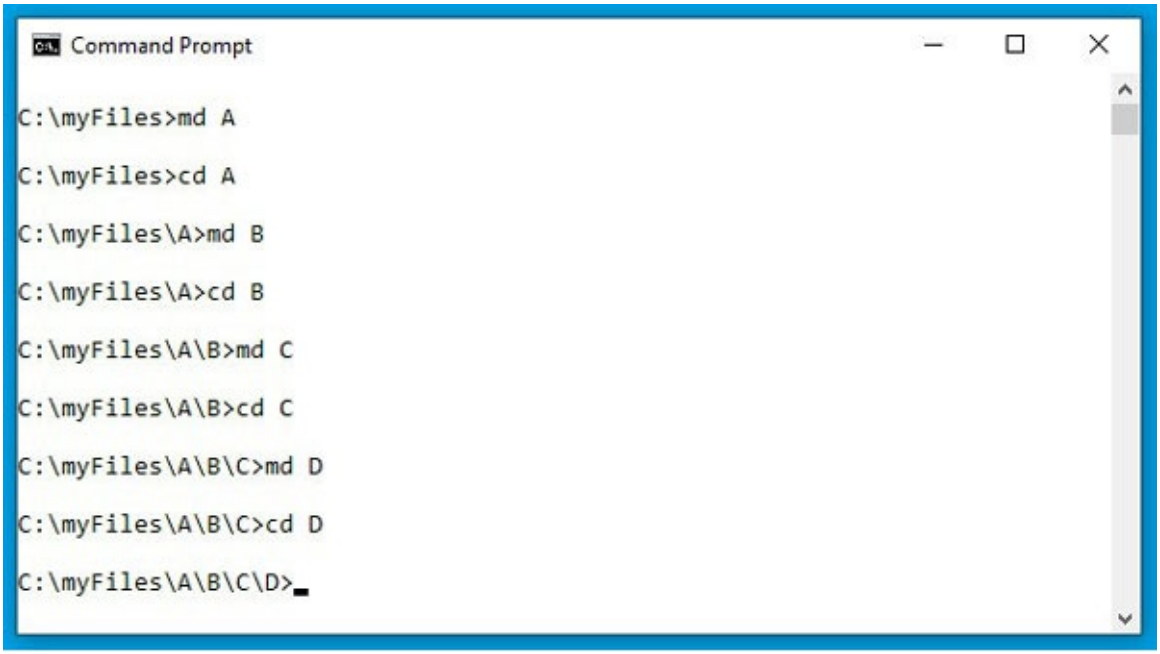

Figure 3.13: *Command Prompt window. We have created the nested folders A, B, C, and D.*

Now, close the Command Prompt window. Now, we will create the four HTML files, namely, **`a.html`**, **`b.html`**, **`c.html`**, and **`d.html`**, in the folders **`A`**, **`B`**, **`C`**, and **`D`**, respectively.

Firstly, let us create the file **`a.html`**. Open the file **`template.html`** and save it with the name **`a.html`** in folder **`A`** that we just created. Update the title element of this file as discussed in the first section of this chapter.

Now, place the keyboard cursor in the body element of this file and insert a few LOCs in it as shown in the following code:

```
07.    <body>
08.     <h2>This is Web Page A Stored in Folder A</h2>
09.     <a>Web Page A</a>
10.     <a href="B/b.html">Web Page B</a>
11.     <a href="B/C/c.html">Web Page C</a>
12.     <a href="B/C/D/d.html">Web Page D</a>
13.    </body>
```

Code Snippet: 3.14

Now, let us create the file **`b.html`**. Close the file **`a.html`**. Open the file **`template.html`** and save it with the name **`b.html`** in the folder **`B`** that we just created. Update the **`title`** element of this file as discussed in the first section of this chapter.

Now, place the keyboard cursor in the **`body`** element of this file and insert a few LOCs in it as shown in the following code:

```
07.    <body>
08.     <h2>This is Web Page B Stored in Folder B</h2>
09.     <a href="../a.html">Web Page A</a>
10.     <a>Web Page B</a>
11.     <a href="C/c.html">Web Page C</a>
12.     <a href="C/D/d.html">Web Page D</a>
13.    </body>
```

Code Snippet: 3.15

Now, let us create the file **c.html**. Close the file **b.html**. Open the file **template.html** and save it with the name **c.html** in the folder **C** that we just created. Update the title element of this file as discussed in the first section of this chapter.

Now, place the keyboard cursor in the **body** element of this file and insert a few LOCs in it as shown in the following code:

```
07.   <body>
08.    <h2>This is Web Page C Stored in Folder C</h2>
09.    <a href="../../a.html">Web Page A</a>
10.    <a href="../b.html">Web Page B</a>
11.    <a>Web Page C</a>
12.    <a href="D/d.html">Web Page D</a>
13.   </body>
```

Code Snippet: 3.16

Now, let us create the file **d.html**. Close the file **c.html**. Open the file **template.html** and save it with the name **d.html** in the folder **D** that we just created. Update the title element of this file as told in the first section of this chapter.

Now, place the keyboard cursor in the **body** element of this file and insert a few LOCs in it as shown as follows:

```
07.   <body>
08.    <h2>This is Web Page D Stored in Folder D</h2>
09.    <a href="../../../a.html">Web Page A</a>
10.     <a href="../../b.html">Web Page B</a>
11.    <a href="../c.html">Web Page C</a>
12.    <a>Web Page D</a>
13.   </body>
```

Code Snippet: 3.17

Close this file (**d.html**) also. We have successfully created the four HTML files and placed them in suitable folders.

Now, let us experiment with these files. Load the file **a.html** (which is in folder **A**) in a web browser and then you will see a couple of lines of text on your web page as shown in *Figure 3.14*:

Figure 3.14: *Web page A (HTML document a.html) is displayed in the web browser. If you click on the link Web Page B, then you will be taken to web page B, and so on. The file "a.html" is in folder A and the current folder is A.*

Click on the link **Web Page B** and web page **B** will appear on the screen as shown in *Figure 3.15*. Click on the link **Web Page B** and web page **B** will appear on the screen as shown in *Figure 3.16*. Click on the link **Web Page B** and web page **B** will appear on the screen as shown in *Figure 3.17*. Click on the link **Web Page A** and web page **A** will appear on the screen, once again, as shown in *Figure 3.14*:

Figure 3.15: *Web page B (html document b.html) is displayed in the web browser. If you click on the link Web Page A, then you will be taken to web page A, and so on. The file "b.html" is in folder B and the current folder is B.*

Figure 3.16: *Web page C (HTML document c.html) is displayed in the web browser. If you click on the link Web Page A, then you will be taken to web page A, and so on. The file "c.html" is in folder C and the current folder is C.*

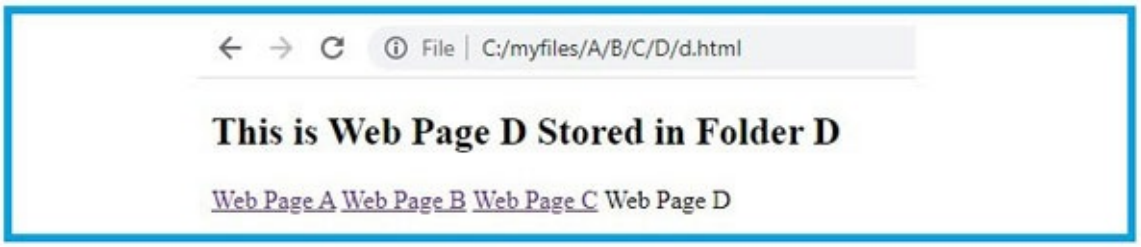

Figure 3.17: *Web page D (HTML document c.html) is displayed in the web browser. If you click on the link Web Page A, then you will be taken to web page A, and so on. The file "d.html" is in folder D and the current folder is D.*

Now, let us discuss the working of this code. Firstly, consider the code in the HTML document **`a.html`**. Here, the relevant portion of the code (from file **`a.html`**) is reproduced for your quick reference as follows:

```
07.   <body>
08.    <h2>This is Web Page A Stored in Folder A</h2>
09.    <a>Web Page A</a>
10.    <a href="B/b.html">Web Page B</a>
11.    <a href="B/C/c.html">Web Page C</a>
12.    <a href="B/C/D/d.html">Web Page D</a>
13.   </body>
```

Code Snippet: 3.18

Take a look at *Figure 3.14*. The first line of text **`This is a Web Page A Stored in Folder A`** is the output of LOC 08 which consists of an **`<h2>`** element.

The second line of text is actually the output of four LOCs, namely, LOC 09, LOC 10, LOC 11, and LOC 12. This second line of text consists of four phrases, literally, **`Web Page A`**, **`Web Page B`**, **`Web Page C`**, and **`Web Page D`**. Out of these four phrases, the first phrase is ordinary text and the remaining three phases are links. If you want to go to web page **`B`**, then click on the link **`Web Page B`**. If you want to go to web page **`C`**, then click on the link **`Web Page C`**. If you want to go to web page **`D`**, then click on the link **`Web Page D`**. You are already on web page **`A`** and hence no link is provided to go to web page A. That's why the first phrase **`Web Page A`** is ordinary text and not a link. You have already studied the working of the

<a> element. The only thing you are going to learn here is the pathname that is prefixed to the filename so that files in other directories can be accessed.

Take a look at LOC 10. Here, the pathname is prefixed to the filename **b.html** is **B/**. This pathname tells the browser that the file **b.html** is placed in folder **B** which is a subfolder of the current folder.

Take a look at LOC 11. Here, the pathname is prefixed to the filename **c.html** is **B/C/**. This pathname tells the browser that the file **c.html** is located in folder **C** which is a subfolder of folder **B** which in turn is a subfolder of the current folder.

Take a look at LOC 12. Here, the pathname is prefixed to the filename **d.html** is **B/C/D/**. This pathname tells the browser that the file **d.html** is located in folder **D** which is a subfolder of folder **C** which in turn is a subfolder of folder **B** which in turn is a subfolder of the current folder.

Now, consider the code in the file **b.html**. The relevant portion of the code from this file is reproduced for your quick reference as follows:

```
07.   <body>
08.    <h2>This is Web Page B Stored in Folder B</h2>
09.    <a href="../a.html">Web Page A</a>
10.    <a>Web Page B</a>
11.    <a href="C/c.html">Web Page C</a>
12.    <a href="C/D/d.html">Web Page D</a>
13.   </body>
```

Code Snippet: 3.19

Notice *Figure 3.15* which shows the web page created by this HTML file. As all the four files, **a.html**, **b.html**, **c.html**, and **d.html** are very much similar, we need not repeat the discussion in the case of every file. The only things to note are how the pathnames are prefixed to filenames in order to access the files in other folders.

In LOC 09, the pathname **../** is prefixed to the filename **a.html**. This pathname tells the browser that file **a.html** is in the parent folder of the current folder. Right now, the current folder is **B** and its parent folder is **A**.

In LOC 11, the pathname **C/** is prefixed to the filename **c.html**. This pathname tells the browser that the file **c.html** is in folder **C** which is a subfolder of the current folder.

In LOC 12, the pathname **C/D/** is prefixed to the filename **d.html**. This pathname tells the browser that file **d.html** is in folder **D** which is a subfolder of folder **C** which in turn is the subfolder of the current folder.

Now, consider the code in the file **c.html**. The relevant portion of the code from this file is reproduced for your quick reference as follows:

```
07.   <body>
08.    <h2>This is Web Page C Stored in Folder C</h2>
09.    <a href="../../a.html">Web Page A</a>
10.    <a href="../b.html">Web Page B</a>
11.    <a>Web Page C</a>
12.    <a href="D/d.html">Web Page D</a>
13.   </body>
```

Code Snippet: 3.20

Notice *Figure 3.16* which shows the web page created by this HTML file. In LOC 09, the pathname

../../ is prefixed to the filename **a.html**. This pathname tells the browser that file **a.html** is located in the parent folder (folder **A**) of the current folder (folder **C**). Right now, the current folder is **C**, its parent folder is **B** and its parent folder is **A**.

In LOC 10, the pathname **../** is prefixed to the filename **b.html**. This pathname tells the browser that the file **b.html** is located in the parent folder (folder **B**) of the current folder (folder **C**).

In LOC 12, the pathname **D/** is prefixed to the filename **d.html**. This pathname tells the browser that file **d.html** is located in the subfolder **D** of the current folder **C**.

Now, consider the code in the file **d.html**. The relevant portion of the code from this file is reproduced for your quick reference as follows:

```
07.   <body>
08.    <h2>This is Web Page D Stored in Folder D</h2>
09.    <a href="../../../a.html">Web Page A</a>
10.     <a href="../../b.html">Web Page B</a>
11.    <a href="../c.html">Web Page C</a>
12.    <a>Web Page D</a>
13.   </body>
```

Code Snippet: 3.21

Notice *Figure 3.17* which shows the web page created by this HTML file. In LOC 09, the pathname **../../../** is prefixed to the filename **a.html**. This pathname tells the browser that the file **a.html**is located in the parent folder (folder **A**) of the current folder (folder **D**). Right now, the current folder is **D**, its parent folder is **C**, its parent folder is **B**, and its parent folder is **A**.

In LOC 10, the pathname **../../** is prefixed to the filename **b.html**. This pathname tells the browser that the file **b.html** is located in the parent folder (folder **B**) of the current folder (folder **D**).

In LOC 11, the pathname **../** is prefixed to the filename **c.html**. This pathname tells the browser that the file **c.html** is located in the parent folder (folder **C**) of the current folder (folder **D**).

Before concluding this section, notice the difference between absolute URLs and relative URLs. An absolute URL contains the absolute address of the desired resource. For example, notice the example of the absolute address given as follows:

```
<p><img src="http:\\www.webilog.in\tbs\images\programmer.jpg" /></p>
```

You need an absolute URL while linking to other websites.

The relative URL contains the address of the desired resource relative to the current page. For example, notice the example of the relative address given as follows:

```
<p><img src="images\programmer.jpg" /></p>
```

Here, the image-file**programmer.jpg** is located in the folder **images** which is a subfolder of the current folder **tbs**. You need a relative URL while linking to web pages on your website.

The LOC 09 in the file **p15.html** also makes use of a relative URL. In this LOC, the image file is located in the current folder; hence, the only image file name is stated.

Creating the in-page links

While browsing the web pages on the Internet, you must have come across a link that says **Click here to go to the top of the page** and when you click on this link, you are actually taken to the top of

that web page. This is an example of in-page linking. In this section, you will learn how to create in-page links. Firstly, let us create a suitable web page for experimentation purposes. The outline of this web page is shown in *Figure 3.18*. On this web page, there will be three links at the top, namely:

- Article on New Delhi
- Article on Mumbai
- Article on Chennai

Followed by these links, there will be three articles, and each article consisting of about 20 to 25 lines of text. When you click on any of the preceding links, the corresponding article will be displayed on the screen. At the end of each article, there is a link that says **`Click here to go to the top of the page`**. When you click on this link, then you are taken to the top of the page. The outline of this web page is shown in *Figure 3.18*:

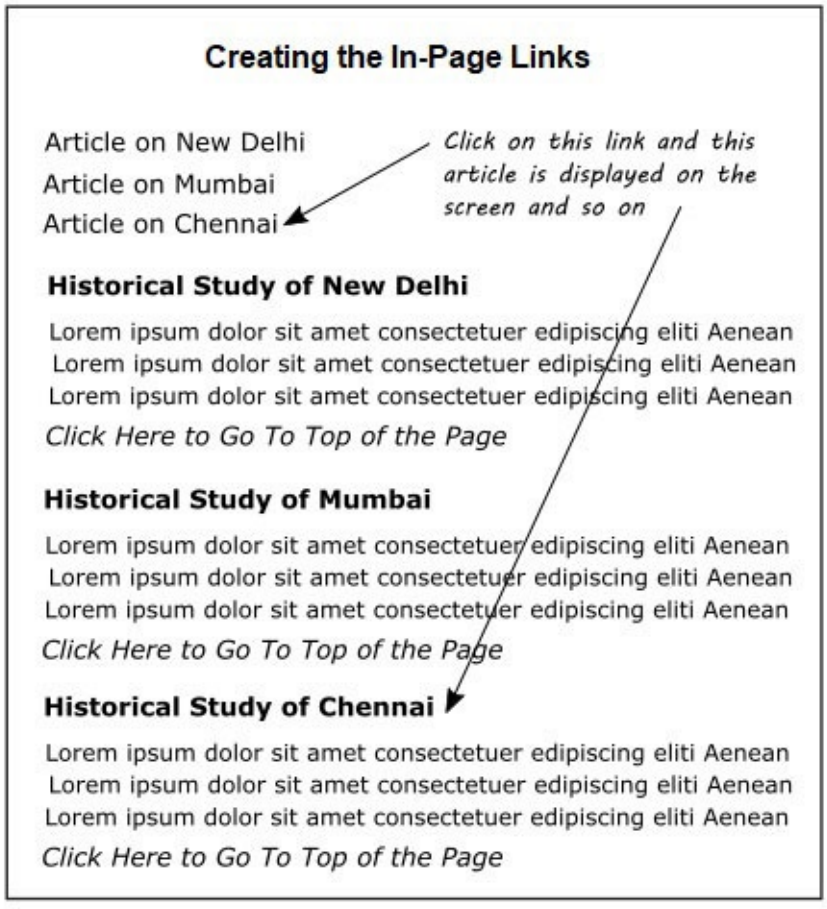

Figure 3.18: *Outline of the web page for creating links within the same web page. If you click on the link, say, "Article on Mumbai" then the article "Historical Study of Mumbai" will be displayed on your screen. If you click on the link, "Click Here to Go To Top of the page" then you will be taken to the top of the page.*

Open the file **`template.html`** and save it with the name **`p22.html`** in the folder **`myFiles`**. Update the **`title`** element as discussed in the first section of this chapter.

Now, place the keyboard cursor in the **`body`** element and insert a few LOCs in it as shown in the following code(as the code is lengthy, I have broken it into four pieces of code):

The first piece of code:

```
<body>
 <h2>Creating the In-Page Links</h2>
 <a href="#delhi">Article on New Dehli</a>
 <br>
 <a href="#mumbai">Article on Mumbai</a>
 <br>
 <a href="#chennai">Article on Chennai</a>
 <br>

```

Code Snippet: 3.22

The second piece of code:

```

 <h2 id ="delhi">Historical Study of New Delihi</h2>
 <p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit.<br>
 Insert about 20 lines of dummy text here using copy & paste method<br>
 consequat vitae, eleifend ac, enim. Aliquam lorem ante, dapibus in, <br>
 </p>
 <a href="#top">Click Here to Go To Top of The Page</a>

```

Code Snippet: 3.23

The third piece of code:

```

 <h2 id ="mumbai">Historical Study of Mumbai</h2>
 <p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit.<br>
 Insert about 20 lines of dummy text here using copy & paste method<br>
 consequat vitae, eleifend ac, enim. Aliquam lorem ante, dapibus in, <br>
 </p>
 <a href="#top">Click Here to Go To Top of The Page</a>

```

Code Snippet: 3.24

The fourth piece of code:

```

 <h2 id ="chennai">Historical Study of Chennai</h2>
 <p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit.<br>
 Insert about 20 lines of dummy text here using copy & paste method<br>
 consequat vitae, eleifend ac, enim. Aliquam lorem ante, dapibus in, <br>
 </p>
 <a href="#top">Click Here to Go To Top of The Page</a>

</body>
```

Code Snippet: 3.25

Now, let us experiment with this web page. Load the file **`p22.html`** in a web browser and then you will see the web page as shown in *Figure 3.18*. Click on the link **`Article on Chennai`** and the article **`Historical Study of Chennai`** will appear on the screen. Click on the link **`Click Here to Go To Top of the Page`** at the end of this article, and you will be taken back to the top of the web page. Play with all the links on this web page for a while and see how these links work.

While working with in-page links, we are required to classify into two groups:

- Source links
- Destination links

When you click on the source link, you are taken to the destination link. For example, when you click on **`Article on Chennai`** (which is the source link created in LOC 13), you are taken to the title of the article **`Historical Study of Chennai`** (which is the destination link created in LOC 30). See the LOCs 13 and 30 to get some insight. So, if you click on the link **`Article on New Delhi`**(which is the source link created in LOC 09), then you are taken to the title of the article **`Historical Study of New Delhi`** (which is the destination link created in LOC 16). Now, notice the difference between source link and destination link:

In the source link, we set the href attribute with some value (for example, href = "#delhi") and in the destination link, we set the id attribute with the same value (for example, id = "delhi"). However, the value of the href attribute is prefixed with a pound or hash symbol (#).

Now, notice the source links in LOCs 21, 28, and 35. These links say **`Click Here to Go To Top of the Page`**. The value assigned to the attribute **`href`** in these LOCs is **`#top`**; however, there is a corresponding destination link where the **`id`** attribute is set with the value **`top`**. This is so because the value **`top`** has a predefined meaning in HTML, and when you click on this source link, you are taken to the top of the web page. This saves your labor of creating the destination link.

Creating your website project

Now, we are all set to begin the development of our website project. We have been storing our files in the folder **`myFiles`**. But to store the files of our website project, we will create the new folder **`website`** with the path **`C:\website`**. In what follows now, we will create three HTML files, namely, **`index.html`**, **`careers.html`**, and **`contact.html`**; and place them in the folder **`website`**. We will also create the folder **`images`** as a subfolder of the folder **`website`** and place the image file **`pic01.jpg`** in the folder **`images`**.

Create the folder **`website`** off the root folder of **`C:`** drive. Create the folder **`images`** off the folder **`website`**. Place the image file **`pic01.jpg`** in the folder **`images`**. Open the file **`template.html`** and save it with the name **`index.html`** in the folder **`website`**. Modify it carefully as follows:

```
<!DOCTYPE html>
<html lang="en">
 <head>
  <title>Truly Best Software</title>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
 </head>
 <body>
  <div class="menu">
   <ul>
    <li><a href="index.html" class="active">Home</a></li>
    <li><a href="careers.html">Careers</a></li>
    <li><a href="contact.html">Contact Us</a></li>
   </ul>
  </div>
  <p><img src="images/pic01.jpg" height="300" alt="programmer"/></p>
 </body>
</html>
```

Code Snippet: 3.26

Open the file **`template.html`** and save it with the name **`careers.html`** in the folder **`website`**. Modify it carefully as shown in the following code:

```
<!DOCTYPE html>
<html lang="en">
    <head>
      <title>Truly Best Software</title>
      <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
   </head>
  <body>
  <div class="menu">
   <ul>
    <li><a href="index.html">Home</a></li>
    <li><a href="careers.html" class="active">Careers</a></li>
    <li><a href="contact.html">Contact Us</a></li>
   </ul>
  </div>
 </body>
```

Code Snippet: 3.27

Open the file **`template.html`** and save it with the name **`contact.html`** in the folder **`website`**. Modify it carefully as shown in the following code:

```
01. <!DOCTYPE html>
02. <html lang="en">
03.  <head>
04.   <title>Truly Best Software</title>
05.   <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
06.  </head>
07.  <body>
08.   <div class="menu">
09.    <ul>
10.     <li><a href="index.html">Home</a></li>
11.     <li><a href="careers.html">Careers</a></li>
12.     <li><a href="contact.html" class="active">Contact Us</a></li>
13.    </ul>
14.   </div>
15.  </body>
16. </html>
```

Code Snippet: 3.28

This type of code - which contains links - is familiar to you. Let us experiment with these files. Load the file **`index.html`** in a web browser and then you will see the web page as shown in *Figure 3.19 (a)*. This web page consists of three links at the top of the web page, namely, **`Home`**, **`Careers`**, and **`Contact Us`**. Click on the link **`Careers`** and the web page **`Career`**s appears on the screen as shown in *Figure 3.19 (b)*. Click on the link **`Contact Us`** and the web page **`Contact Us`** appears on the screen as shown in *Figure 3.19 (c)*. The links on all three pages are working. Also, there is an image on the web page **`Home`**. In the successive chapters of this book, you will add a number of things to these web pages.

Each of these three HTML files - being discussed - consists of LOC 05 reproduced for your quick reference as follows:

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
```

This LOC consists of the element **`<meta>`**. This element consists of metadata. Metadata means data about data. The information in this element is never displayed on the web page. This metadata is helpful to browsers and search engines; hence, we should not omit this LOC from a web page. We will revisit and discuss this element in the latter part of this book.

LOL 08 of each of these web pages consist of the opening tag of the **`<div>`** element and this LOC is reproduced for your quick reference as follows:

```
<div class="menu">
```

We use the **`<div>`** element as a sort of suitcase. We put (or pack) the number of items in this element; between the opening tag **`<div>`** and closing tag **`</div>`**. The attribute **`class`** of this element is assigned the value**`menu`**. The attributes **`class`** and **`id`** are frequently used in any HTML document. Here, we have assigned the value "menu" to attribute "class" because we need to set the properties of the contents of this **`<div>`** element using **CSS** (**Cascading Style Sheets**). We will do this in the latter part of the book.

Similarly, in LOC 10 in the file **`index.html`**, in LOC 11 in the file **`careers.html`**, and in LOC 12 in the file **`contact.html`**, you will notice that the attribute **`class`** is set with the value **`active`** because we want to set the properties of the concerned link using CSS in the latter part of this book:

Figure 3.19: *The three web pages index.html, careers.html, and contact.html are created which are part of our website project. These files are stored in the folder website.*

Conclusion

In this chapter, you learned to add an image to a web page, add an audio clip to a web page, add a video clip to a web page, create links to other websites, create links to web pages of your website, and create in-page links. You also embarked on the process of creating a project website. Tables and forms are highly useful constructs in making a website. In the table, you can display the data in a structured manner. Forms allow us to accept the data from the users of the website in a systematic manner. In the next chapter, you will learn to put tables and forms on a web page.

Further readings/ references

- *HTML & CSS: The Complete Reference*, by Thomas A. Powell, 5/e, McGraw Hill, 2010.
- *HTML: A Beginner's Guide*, by Wendy Willard, 4/e, McGraw Hill, 2009.
- **https://www.technologyuk.net/computing/website-development/html-basics/audio-and-video-in-web-pages.shtml**.
- **https://developer.mozilla.org/en-US/docs/Learn/HTML/Multimedia_and_embedding/Video_and_audio_content**.
- **https://developer.mozilla.org/en-US/docs/Learn/HTML/Multimedia_and_embedding/Images_in_HTML**.
- **https://www.carmelon-digital.com/digital-marketing-magazine/the-importance-of-images-on-a-website/**.

CHAPTER 4

Using Tables and Forms

Introduction

In this chapter, you will learn to create a table, create a table with a column span and a row span, and create a professional table. You will also learn to create a form, create various form controls for inputting the data such as text, password, email, URL, file, phone number, and date. In addition, you will learn to create radio buttons, checkboxes, and labels. Finally, you will learn to group the form controls.

Structure

- The following topics are covered in this chapter:
- Creating a table on a web page
- Creating a table with a column span and row span
- Creating a professional table
- Creating a form
- Creating the form controls for inputting text, password, email, URL, file, phone number, and date
- Using the radio buttons, checkboxes, listboxes, labels in a form
- Grouping the controls in a form

Objectives

After reading this chapter, you will be able to create a table on a web page, create a table with a column span and row span, and create a professional table. You will also be able to create a form on a web page and put various form controls on it to accept text, password, email-id, URL, phone number, and date. In addition, you will be able to place the following controls on a form: radio buttons, checkboxes, list boxes, and labels. Finally, you will be able to group the various controls in the form.

Creating a table in a web page

Tables are a must in homes, offices, restaurants, and documents. Here, we are interested only in the tables to be placed in web pages. In order to create a table in an HTML document, we use the following tags:

- Tags to create a table: **`<table>`** and **`</table>`**
- Tags to create a header in a table: **`<th>`** and **`</th>`**
- Tags to create a row in a table: **`<tr>`** and **`</tr>`**
- Tags to create a data item in a table: **`<td>`** and **`</td>`**

Let us create a simple table shown in *Figure 4.1*:

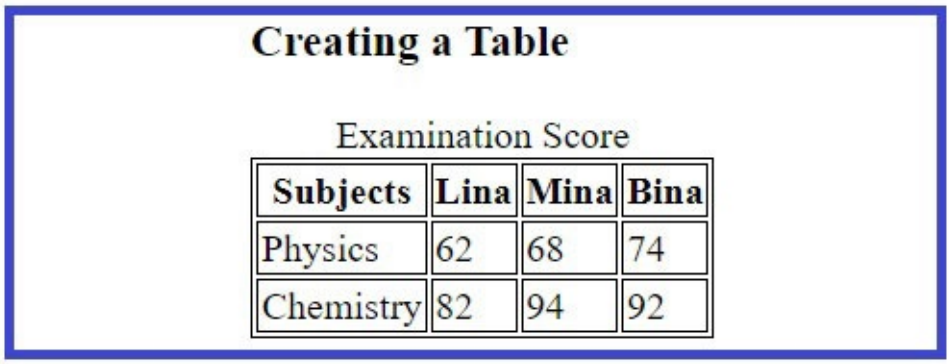

Figure 4.1: *A simple table is created that shows examination scores by students.*

Open the file **`template.html`** and save it with the filename **`template2.html`** in the folder **`myFiles`**. Modify the file **`template2.html`** as follows and then close it:

```
01. <!DOCTYPE html>
02. <html lang="en">
03.   <head>
04.     <meta charset="utf-8">
05.     <title>Using Tables and Forms</title>
06.     <style type="text/css">
07.         table, th, td {
08.           border: 1px solid black;
09.         }
10.     </style>
11.   </head>
12.   <body>
13.   </body>
14. </html>
```

Code Snippet: 4.1

Before proceeding further, a few words about the <style> element used in this template in the LOCs 06-10. Actually, styles will be officially introduced to you in *Chapter 5, Welcome to CSS*. Here, I have used styles in this program for the want of borders for the table. Without this piece of code in the LOCs 06-10, the table in *Figure 4.1* would appear borderless. A world without borders certainly looks beautiful, but a table without borders looks miserable. In this piece of code, the LOC 06 consists of the opening tag **`<style>`**, and LOC 10 consists of the closing tag **`</style>`**. In LOC 06, the attribute **`type`** specifies the media type of **`<style>`** tag. Here, the media type is text/CSS.

LOCs 07-09 consist of a **CSS**(**Cascading Style Sheets**) rule. Let us peep inside a CSS rule. A CSS rule consists of two parts:

- Selector(s)
- Declaration

Here, selectors and declaration are as follows:

Selectors:**`table, th, td`**

Declaration: **`{ border: 1px solid black ;}`**

Here, the declaration is packed in a single line. If you want, you can also pack the complete CSS rule in a

single line as follows:

- CSS rule:**`table, th, td { border: 1px solid black ;}`**

And still, it works. However, for the want of clarity, we will format the LOCs with ample white space in them as in LOCs 07-09.

Here, the selectors are **`table`**, **`hd`**, and **`td`**. The property (or properties) set in the declaration part will be applied to these three elements, namely, **`<table>`**, **`<hd>`**, and **`<td>`**, throughout the HTML file.

Now, notice the syntax of the declaration part. The declaration part stated herehas the following constituents:

- Opening curly bracket:**`{`**
- Property and value:**`border: 1px solid black ;`**
- Closing curly bracket: **`}`**

In the declaration part, there is a pair of curly brackets and one or more properties set between this pair of curly brackets. The property and value used in this declaration part are as follows:

- Property:**`border`**
- Value:**`1px solid black`**

Also, there is a colon (`:`) after the property and a semicolon after the value (`;`). Thus, the generic syntax of the property setting LOC is as follows:

```
property : value ;
```

Here, the value used is three-fold, that is, it consists of three parts as follows:

- **`1px`**:This part tells the browser to set the width of the borderline to 1 pixel.
- **`solid`**:This part tells the browser that the line of the border should be a solid line.
- **`black`**:This part tells the browser that the color of the borderline should be black.

You can set the width, style, and color of the borderline as per your choice. You can have a thick line of, say, 3 px for your border. Apart from a solid line, you can also have a dashed line or dotted line for your border. Apart from the black color, you can also have a border of almost any color you want, for example, blue, green, red, yellow, and so on.

Now, notice one more example of the declaration given next. This is a declaration block. In a declaration block, two or more properties are set simultaneously:

```
p {
  font-family: Arial ;
  color: yellow;
  }
```

Here, the selector is **`<p>`**. The first property set is the font of the text and the second property set is the color of the text. This declaration block will cause all the **`<p>`** elements used in the HTMLfile to have Arial font and yellow color.

This was a quick introduction to CSS. Now, we stop the discussion on CSS and return to tables. For details onCSS, please see *Chapter 5, Welcome to CSS* and *Chapter 6, Getting Command on CSS*.

Open the file **`template2.html`** and save it with the name **`p26.html`** in the folder **`myFiles`**. The last

program we saved was **p22.html** and this program is named **p26.html**. I am purportedly leaving some space so that more programs can be appended to the preceding chapter.

Now, place the keyboard cursor in the **body** element and insert a few LOCs in it as follows:

```
12. <body>
13.    <h3>Creating a Table</h3>
14.   <table>
15.   <caption>Examination Score</caption>
16.   <tr>
17.    <th>Subjects</th>
18.    <th>Lina</th>
19.    <th>Mina</th>
20.    <th>Bina</th>
21.   </tr>
22.   <tr>
23.    <td>Physics</td>
24.    <td>62</td>
25.    <td>68</td>
26.    <td>74</td>
27.   </tr>
28.   <tr>
29.    <td>Chemistry</td>
30.    <td>82</td>
31.    <td>94</td>
32.    <td>92</td>
33.   </tr>
34.   </table>
35. </body>
```

Code Snippet: 4.2

Load this HTML document in a web browser and then you will see the web page with a table displayed on it as shown in *Figure 4.1*.

The whole table is enclosed between LOC 14 and LOC 34. LOC 14 consists of the opening tag **<table>** and LOC 34 consists of the closing tag **</table>**. This table consists of three rows. Each row is enclosed between the opening tag **<tr>** and closing tag **</tr>**. Here, **tr** stands for **table row**.

The first row is enclosed between LOCs 16-21. The second row is enclosed between LOCs 22-27. The third row is enclosed between LOCs 28-33. Each row is enclosed between the opening tag **<tr>** and closing tag **<tr>**. Each row consists of four data items.

In the first row (LOCs 16-21), every data item is enclosed between the opening tag **<th>** and closing tag **</th>**. Here, **th** stands for **table heading**. The first row in the table is designated as the table heading and its text is set to be bold (see *Figure 4.1*). This first row consists of four data items, namely, Subjects, Lina, Mina, and Bina.

In the second row (LOCs 22-27) and third row (LOCs 28-33), every data item is enclosed between the opening tag **<td>** and closing tag **</td>**. Here, **td** stands for **table data**.

The purpose of LOC 15 is only to add a caption to the table. Here, the caption of the table is **Examination Score** (see *Figure 4.1*). The caption is an optional, but very useful feature.

Creating a table with a column span

We use the column span procedure in order to join the multiple cells to create a single cell horizontally, as

shown in *Figure 4.2*. This table consists of two rows and three columns. The three cells in the first row are merged using the column span procedure:

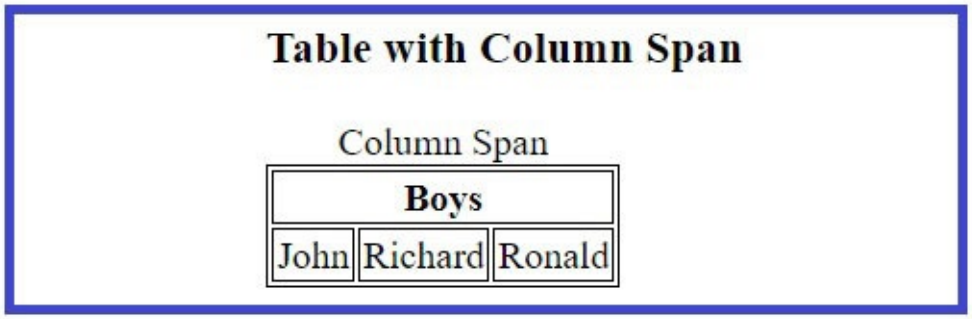

Figure 4.2: *The column span procedure is used in this table. Three cells in the first row are merged to create a single cell.*

Let us see how to span the columns in a table. Open the file **`template2.html`** and save it with the name **`p27.html`** in the folder**`myFiles`**. Now, place the keyboard cursor in the **`body`** element and insert a few LOCs in it as follows:

```
12.   <body>
13.     <h3>Table with Column Span</h3>
14.     <table>
15.       <caption>Column Span</caption>
16.       <tr>
17.         <th colspan="3">Boys</th>
18.       </tr>
19.       <tr>
20.         <td>John</td>
21.         <td>Richard</td>
22.         <td>Ronald</td>
23.       </tr>
24.     </table>
25.   </body>
```

Code Snippet: 4.3

Load this HTML document in a web browser and then you will see the web page with a table displayed on it as shown in *Figure 4.2*. Let us see how the procedure of column span works. In LOCs 16-18, the first row is fitted. In LOCs 19-23, the second row is fitted. In LOC 17, the attribute **`colspan`** is set with the value of integer **`3`**. This attribute is responsible for the merging of three cells in a single cell horizontally. We can use the attribute **`colspan`** with the elements **`<th>`** and **`<td>`**.

The generic syntax of setting the value of attribute **`colspan`** is as follows:

```
colspan = "n"
```

Where **`n`** = integer that specifies the number of cells to be merged horizontally.

Creating a table with a row span

We use the row span procedure in order to join the multiple cells to create a single cell vertically, as shown in *Figure 4.3*. This table consists of -three rows and two columns. The three cells in the first columns are merged using the row span procedure:

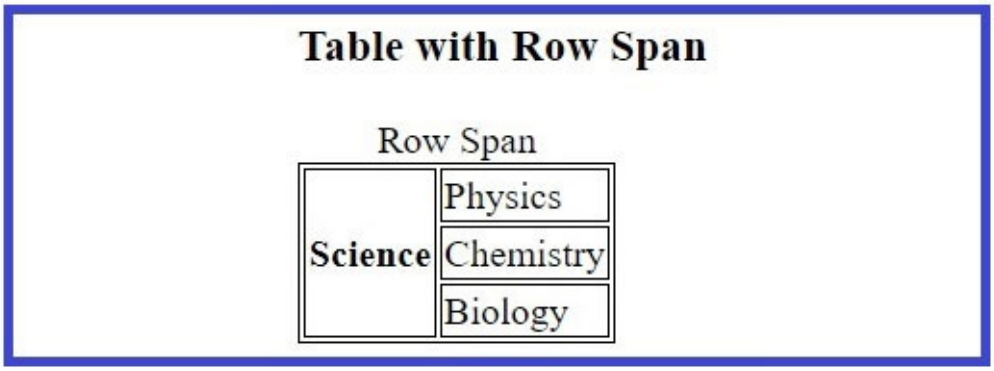

Figure 4.3: *The row span procedure is used in this table. Three cells in the first column are merged to create a single cell.*

Let us see how to span the rows in a table. Open the file **`template2.html`** and save it with the name **`p28.html`** in the folder **`myFiles`**. Now, place the keyboard cursor in the **`body`** element and insert a few LOCs in it as follows:

```
12.   <body>
13.     <h3>Table with Row Span</h3>
14.     <table>
15.       <caption>Row Span</caption>
16.       <tr>
17.         <th rowspan="3">Science</th>
18.         <td>Physics</td>
19.       </tr>
20.       <tr>
21.         <td>Chemistry</td>
22.       </tr>
23.       <tr>
24.         <td>Biology</td>
25.       </tr>
26.     </table>
27.   </body>
```

Code Snippet: 4.4

Load this HTML document in the web browser and then you will see the web page with a table displayed on it as shown in *Figure 4.3*. Let us see how the procedure of row span works.

In LOCs 16-19, the first row is fitted. In LOCs 20-22, the second row is fitted. In LOCs 23-25, the third row is fitted. In LOC 17, the attribute **`rowspan`** is set with the value of integer **`3`**. This attribute is responsible for the merging of three cells in a single cell vertically. We can use the attribute **`rowspan`** with the elements **`<th>`** and **`<td>`**.

The generic syntax of setting the value of attribute **`rowspan`** is as follows:

```
rowspan = “n”
```

Where **`n`** = integer that specifies the number of cells to be merged vertically.

Here, the first row consists of two data items, namely, Science and Physics, the second row consists of only one data item, namely, Chemistry, and the third row consists of only one data item, namely, Biology. There is only one data item in the second and third rows because the first cell in these rows is lost in the merging of cells.

Creating a professional table

Now, we are all set to create a professional table shown in *Figure 4.4*. You have acquired enough mastery to create this table:

Examination Score of Boys and Girls

Examination Score

Faculty	Subjects	Boys		Girls	
		Dick	Bob	Rina	Tina
Science	Physics	52	56	63	75
	Chemistry	58	55	82	94
Arts	Music	52	56	63	75
	Painting	58	55	82	94

Figure 4.4: *Professional table with column span and row span applied to it.*

Open the file **`template2.html`** and save it with the name **`p29.html`** in the folder **`myFiles`**. Now, place the keyboard cursor in the **`body`** element and insert a few LOCs in it as follows:

```
12.   <body>
13.     <h4>Examination Score of Boys and Girls</h4>
14.     <table>
15.       <caption>Examination Score</caption>
16.       <tr>
17.         <th rowspan="2">Faculty</th>
18.         <th rowspan="2">Subjects</th>
19.         <th colspan="2">Boys</th>
20.         <th colspan="2">Girls</th>
21.       </tr>
22.       <tr>
23.         <td>Dick</td>
24.         <td>Bob</td>
25.         <td>Rina</td>
26.         <td>Tina</td>
27.       </tr>
28.       <tr>
29.         <td rowspan="2">Science</td>
30.         <td>Physics</td>
31.         <td>52</td>
32.         <td>56</td>
33.         <td>63</td>
34.         <td>75</td>
35.       </tr>
36.       <tr>
37.         <td>Chemistry</td>
38.         <td>58</td>
39.         <td>55</td>
40.         <td>82</td>
41.         <td>94</td>
42.       </tr>
43.       <tr>
44.         <td rowspan="2">Arts</td>
45.         <td>Music</td>
46.         <td>52</td>
47.         <td>56</td>
48.         <td>63</td>
49.         <td>75</td>
50.       </tr>
51.       <tr>
52.         <td>Painting</td>
53.         <td>58</td>
54.         <td>55</td>
55.         <td>82</td>
56.         <td>94</td>
57.       </tr>
58.     </table>
59.   </body>
```

Code Snippet: 4.5

Load this HTML document in a web browser and then you will see the web page with a table displayed on it as shown in *Figure 4.4*.

This professional table consists of six columns and six rows. The first row is fitted in the LOCs 16-21. The second row is fitted in the LOCs 22-27. The third row is fitted in the LOCs 28-35. The fourth row is fitted in the LOCs 36-42. The fifth row is fitted in the LOCs 43-50. Finally, the sixth row is fitted in the LOCs 51-57.

In the first row (LOCs 16-21), row spanning is performed at two places and column spanning is also performed at two places. Apart from this, row spanning is performed in the third row and fifth row. The procedure of row spanning and column spanning is already explained and there is no need to repeat that explanation here again.

Creating a form

The form is a very useful feature supported by HTML. We need a form when we want to collect some information from the user of the website. With HTML, you can create a professional form that is laced with various form controls like textboxes, radio buttons, checkboxes, drop-down buttons, submit buttons, and so on. The most important element in this regard is **<form>**. Your form on a web page is sandwiched between the opening tag **<form>** and the closing tag **</form>**. The user of the website enters information in this form and this information is passed to the server-side program (so that it can be processed) after the user clicks on the **Submit** button located at the end of the form. Let us create a very simple form.

Open the file **template.html**, modify it as follows, save it with the name **template3.html** in the folder **myFiles**, and then close it:

```
01. <!DOCTYPE html>
02. <html lang="en">
03.   <head>
04.     <meta charset="utf-8">
05.     <title>Using Forms</title>
06.   </head>
07.   <body>
08.   </body>
09. </html>
```

Code Snippet: 4.6

Open the file **template3.html** and save it with the name **p30.html** in the folder **myFiles**. Now, place the keyboard cursor in the **body** element and insert a few LOCs in it as follows:

```
07.   <body>
08.     <form action="" method="get">
09.       <p>Enter your name:
10.         <input type="text" name="myname" size="20" maxlength="80"/>
11.       </p>
12.     </form>
13.   </body>
```

Code Snippet: 4.7

Load this HTML document in the web browser and then you will see the web page with a form displayed on it as shown in *Figure 4.5*. This form consists of a text item **Enter your name** and a rectangular text box in which you are supposed to type your name. Simply click on this text box and a blinking keyboard cursor will appear in it. Now, type your name in it:

Figure 4.5: *A simple form that consists of a text item Enter your name and a rectangular-shaped textbox.*

The code that is responsible for this form is contained in LOCs 08-12. LOC 08 consists of the opening tag **<form>** and LOC 12 consists of the closing tag **</form>**. Two important attributes of the **<form>** elements are **action** and **method**.

In LOC 08, these attributes, namely, **action** and **method** are assigned suitable values. Actually, in LOC 08, a null string is assigned to the attribute **action** to keep things simple. Ideally, the URL of the server-side program is assigned to the attribute **action** something like the following:

```
action="http://www.webilog.in/tbs/processform.php"
```

In the preceding example, the URL of the program **processform.php** is assigned to the attribute **<action>**. This program processes the data entered in the form by the user of the website.

The attribute **method** can have only two values, namely,**get** and **post**. If the data collected in the form is sensitive (confidential), then use the value **post** otherwise, use the value **get** for the attribute **method**.

The LOCs 09 and 11 contain the element **<p>**. The purpose of this element is:

To display the text item: **Enter your name**.

To enforce a line break so that the next item to be displayed on the screen should appear on the next line. Notice that the element **<p>** is a block-level element whereas form elements are inline elements. The block element always enforces a line break whereas the inline element doesn't enforce a line break.

LOC 10 consists of the element <input>. LOC 10 is reproduced as follows for your quick reference:

```
<input type="text" name="myname" size="20" maxlength="80"/>
```

The element **<input>** is an empty element. It means it has no closing tag like **</input>**. Instead of a closing tag, there is a forward slash / at the end.

Here, we have used four attributes of element **<input>**, namely, **type**, **name**, **size**, and **maxlength**. We will use these attributes for most of the form elements. Let's learn about these attributes:

- **type**: This attribute determines the type of input. Here, the type of input is text. Some other types are date, time, color, email, and so on. See *Table 4.1* for the list of input types.
- **name**: Using this attribute, we assign some suitable names to this input element. With the help of this name, the programmer (who writes the server-side program **processform.php**) retrieves the text typed in this text box and stores it at a suitable location. What name should be assigned to this attribute? It is your authority to name the elements suitably. You can give any name to this element, for example, Tom, Jerry, Lisa, ghost, witch, and so on, but a sensible and meaningful name will be helpful to you and the server-side programmer also. Here, we have named this element **myname** because this element (textbox) accepts the name of the user.
- **size**: This attribute specifies the length of the textbox. Here, we have specified the length of the textbox to be 20 characters.
- **maxlength**: This attribute specifies the maximum length of the text that you can type in this element (textbox). Here, we have specified the **maxlength** to be 80 characters. It means you cannot type more than 80 characters in this textbox.

Element <input>

The element **<input>** is very useful in creating a form as it provides us with an easy way to accept the various types of data in a form. *Table 4.1* enlists the various types associated with the element **<input>**:

Type	Purpose
text	To create a textbox to accept a single line of text.
password	To create a textbox to accept a password.

email	To create an email ID input box.
url	To create a URL input box.
file	To create a file input box.
tel	To create a phone number input box.
date	To create a date input box.
submit	To create a submit button forthe form.
checkbox	To create a checkbox.
radio	To create a radio button.
search	To create a search box.
image	To create a submit button with the desired image on it.
datetime	To create a date and time input box.
datetimelocal	To create a local date and time input box.
month	To create a monthlyinput box.
week	To create a weekly input box.

Table 4.1: *Various types (Form Controls) associated with the element <input>*

Text, password, email, URL, file, phone date, and submit

In this subsection, we will write a program that uses the element **`<input>`** with the following types: text, password, email, URL, file, date, and submit. In the preceding section, you have already learned something about the element **`<input>`** with the type text. Now, you will learn about another type of element **`<input>`**.

Open the file **`template3.html`** and save it with the name **`p31.html`** in the folder **`myFiles`**. Now, place the keyboard cursor in the **`body`** element and insert a few LOCs in it as follows:

```
07.   <body>
08.     <form action="" method="post">
09.       <p>Enter your name:
10.         <input type="text" name="myname" size="20" maxlength="80" required/>
11.       </p>
12.       <p>Enter your password:
13.         <input type="password" name="mypassword" size="20" maxlength="100" />
14.       </p>
15.       <p>Enter your email id:
16.         <input type="email" name="myemail" size="20" maxlength="60" required/>
17.       </p>
18.       <p>Enter URL of your website:
19.         <input type="url" name="myurl" size="20" maxlength="60" required/>
20.       </p>
21.       <p>Upload file of your CV (Max Size: 10 MB):
22.         <input type="file" name="mycvfile" />
23.       </p>
24.       <p>Enter your mobile number:
25.         <input type="tel" name="myphone" pattern="[0-9]{10}" size="10" maxlength="10"/>
26.       </p>
27.       <p>Enter your birth date:
28.         <input type="date" name="mybirthdate" required />
29.       </p>
30.       <p>Now click this Submit button:
31.         <input type="submit" name="mysubmit" value="Submit"/>
32.       </p>
33.     </form>
34.   </body>
```

Code Snippet: 4.8

Load this HTML document in a web browser and then you will see the web page with a form displayed on it as shown in *Figure 4.6*. This form consists of eight **<input>** elements placed on it:

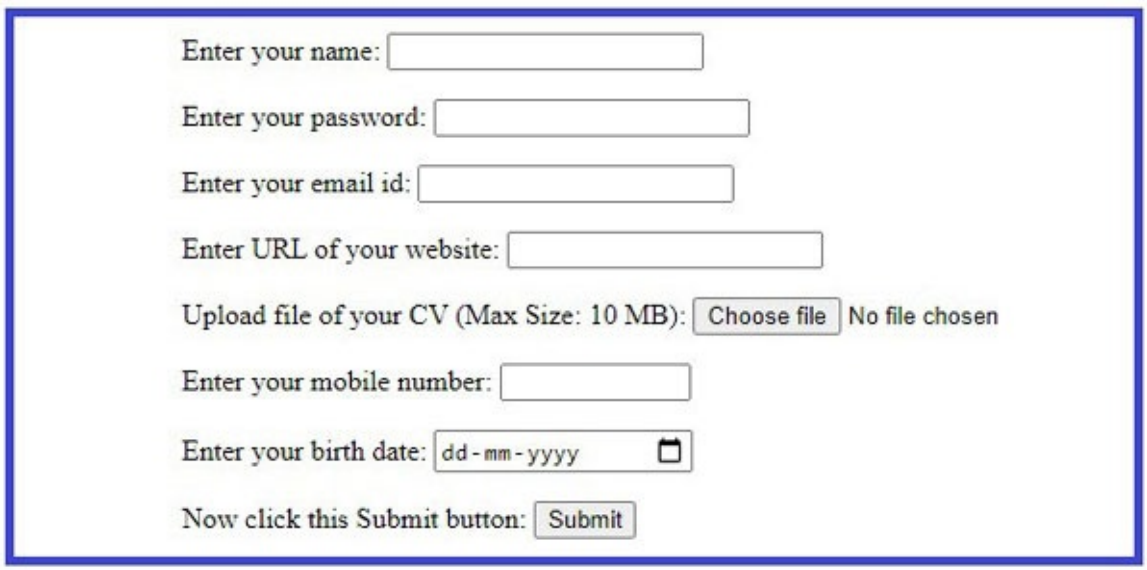

Figure 4.6: *A simple form that consists of various form controls.*

LOCs 09-11 consist of the **<input>** element of the type text. You have used this element in the preceding section. Notice the attribute **required** used in the LOC 10. This attribute makes it mandatory for the user to fill in the legitimate value in the input box before clicking on the **Submit** button. If you click on the **Submit** button without filling in legitimate value in the input box, a small message box pops up on the

screen as shown in *Figure 4.7* asking you to **`Please fill in this field`**. Here, we have not assigned any value to this attribute but if you want, you can assign the value **`required`** to this attribute as follows:

```
required = “required”
```

Notice that this attribute is also used in the LOCs 16, 19, and 28:

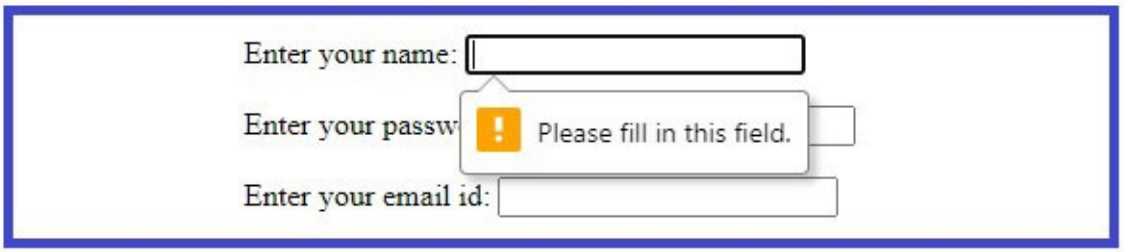

Figure 4.7: *The required attribute reminds you to enter the data in the field.*

LOCs 12-14 consist of the **`<input>`** element of type password. This input element is also like a textbox. However, as this textbox is meant for the password, a chain of asterisks (*******) is displayed in it when you type some text in it.

LOCs 15-17 consist of the **`<input>`** element of type email. This input element is meant for accepting your email ID. Type your valid email ID in it (for example, **`xyz@zyx.com`**).

LOCs 18-20 consist of the **`<input>`** element of type URL. This input element is meant for accepting the **URL** (**Uniform Resource Locator**) of your website. Type a valid URL of your website in it (for example, **`http://www.webilog.in`**).

LOCs 21-23 consist of the **`<input>`** element of type file. This input element is meant for uploading the files. In order to upload a file, simply click on the button **`Choose File`** displayed on the screen (*Figure 4.6*) and the dialog box **`Open`** pops up on the screen. Type the name of the file to be uploaded with the path in the edit-box **`Filename`** and click on the button **`Open`**. Now, the file gets uploaded to the server. When this input type is used, the attribute **`method`** should be set with the value post.

LOCs 24-26 consist of the **`<input>`** element of type tel. This input element is meant for accepting your mobile number. Type your valid mobile number in it without the country code. Notice the LOC 25 in which the attribute **`pattern`** is assigned some value as follows:

```
pattern = “[0-9]{10}”
```

This value assigned to the attribute **`pattern`** tells the browser that the user should type ten digits in the input box. Contents of the first bracket [0-9] tell the browser that this input box should accept only digits from 0 to 9. Contents of the second bracket {10} tell the browser that this input box should accept 10 characters (here, digits).

LOCs 27-29 consist of the **`<input>`** element of type date. This input element is meant for accepting a date. Type the required date (here, birthdate) directly in the input box by overwriting the letters **`dd-mm-yyyy`**. Alternatively, click on the icon of the calendar located at the right end of this input box and the calendar drops down. Now, set the date in this calendar.

LOCs 30-32 consist of the **`<input>`** element of the type submit. This input element is meant for submitting the contents of the form to the server. Click on this button and all the data filled in the form will be submitted to the server.

Radio buttons

We use radio buttons when the user is expected to select only one option given the two or more options. In this subsection, we will develop a simple form that consists of a group of three radio buttons (*Figure 4.8*). Therefore, let us develop a suitable program accordingly. Open the file **`template3.html`** and save it with the name **`p32.html`** in the folder **`myFiles`**. Now, place the keyboard cursor in the **`body`** element and insert a few LOCs in it as follows:

```
07.   <body>
08.     <form action="" method="post">
09.       <p>Select your favorite color:<br/>
10.         <input type="radio" name="mycolor" value="blue" checked />Blue<br/>
11.         <input type="radio" name="mycolor" value="red" />Red<br/>
12.         <input type="radio" name="mycolor" value="green" />Green<br/>
13.       </p>
14.     </form>
15.   </body>
```

Code Snippet: 4.9

Load this HTML document in a web browser and then you will see the web page with a form displayed on it as shown in *Figure 4.8*. This form consists of a group of three radio buttons which are captioned as **`Blue`**, **`Red`**, and **`Green`**. The radio button **`Blue`** is selected by default. You can click on any radio button to select it:

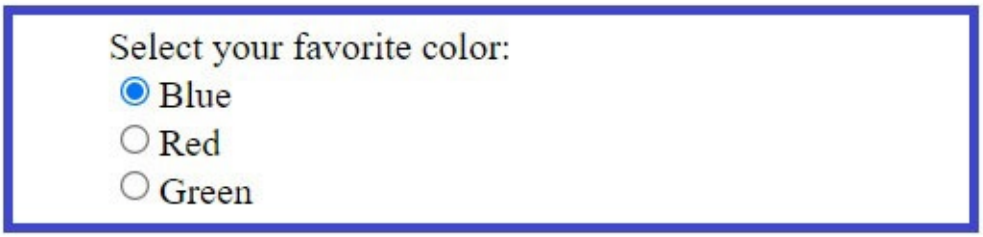

Figure 4.8: *A simple form that consists of a group of three radio buttons. Only one radio button can be selected in a group of radio buttons. The radio button Blue is selected by default.*

LOC 09 displays the text **`Select your favorite color`**: on the screen and the element **`<br/>`** ensures that the next item would be displayed on the fresh line.

LOC 10 displays the radio button **`Blue`**. LOC 11 displays the radio button **`Red`**. LOC 12 displays the radio button **`Green`**. Notice that the name of all radio buttons in a group is the same and here it is **`mycolor`**. However, the value of each radio button is different as you can see in LOCs 10-12. The value of the selected radio button is sent to **SSP(Server Side Program)** for processing. The radio button **`Blue`** is selected by default because we have used the attribute **`checked`** in LOC 09. We have not assigned any value to attribute **`checked`** as it is not required.

Checkboxes

We use checkboxes when the user is expected to select zero, one, or more options from a given set of options (unlike radio buttons where you can select only one option). In this subsection, we will develop a simple form that consists of a group of three checkboxes (*Figure 4.9*). Therefore, let us develop a suitable program accordingly. Open the file **`template3.html`** and save it with the name **`p33.html`** in the folder **`myFiles`**. Now, place the keyboard cursor in the **`body`** element and insert a few LOCs in it as follows:

```
07.   <body>
08.     <form action="" method="post">
09.       <p>Select your favorite color:<br/>
10.         <input type="checkbox" name="mycolor" value="blue" />Blue<br/>
11.         <input type="checkbox" name="mycolor" value="red" />Red<br/>
12.         <input type="checkbox" name="mycolor" value="green" />Green<br/>
13.       </p>
14.     </form>
15.   </body>
```

Code Snippet: 4.10

Load this HTML document in the web browser and then you will see the web page with a form displayed on it as shown in *Figure 4.9*. This form consists of a group of three checkboxes which are captioned as **`Blue`**, **`Red`**, and **`Green`**. No checkbox is checked by default. You can click on any checkbox to put a checkmark in it:

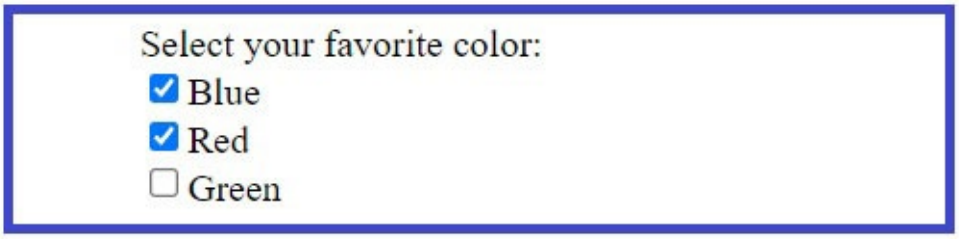

Figure 4.9: *A simple form that consists of a group of three checkboxes. You can select any number of checkboxes in this group (0, 1, 2, or 3). Here, the user has clicked on two checkboxes (Blue and Red) to select them.*

LOC 09 displays the text **`Select your favorite color`** on the screen and the element **`<br/>`** ensures that the next item would be displayed on the fresh line.

LOC 10 displays the checkbox **`Blue`**. LOC 11 displays the checkbox **`Red`**. LOC 12 displays the checkbox **`Green`**. Notice that the name of all checkboxes in a group is the same and here it is **`mycolor`**. However, the value of each check box is different as you can see in LOCs 10-12. The values of the selected checkboxes are sent to SSP for processing. If you want to put a checkmark in any checkbox by default, then simply put the attribute **`checked`** in the corresponding LOC as shown in LOC 10 of the preceding program **`p32.html`**.

Select boxes or list boxes

Select boxes are nothing but the listboxes. We use the select boxes when we want to fit all the options in a short space. There are two types of select boxes. In the first type, you can select only one option. In the second type, you can select multiple options. Let us deal with the first type, firstly. Open the file **`template3.html`** and save it with the name **`p34.html`** in the folder **`myFiles`**. Now, place the keyboard cursor in the **`body`** element and insert a few LOCs in it as follows:

```
07.   <body>
08.     <form action="" method="post">
09.       <p>Select your most favorite singer:<br/></p>
10.       <select name="singer">
11.        <option value="rafi">Mohammad Rafi</option>
12.        <option value="kishore">Kishore Kumar</option>
13.        <option value="mahrendra">Mahendra Kapoor</option>
14.        <option value="talat">Talat Mahmood</option>
15.       </select>
16.     </form>
17.   </body>
```

Code Snippet: 4.11

Load this HTML document in the web browser and then you will see the web page with a form displayed on it as shown in *Figure 4.10 (a)*. This form consists of a drop-down listbox in which the list item Mohammad Rafi is selected by default. Click on the drop-down button on it (the downward-pointing arrowhead on the list box is called a drop-down button) and a list falls down as shown in *Figure 4.10 (b)*. Now, click on the desired list item to select it. As you click on the desired list item, the clicked list item gets selected and the list winds up as before:

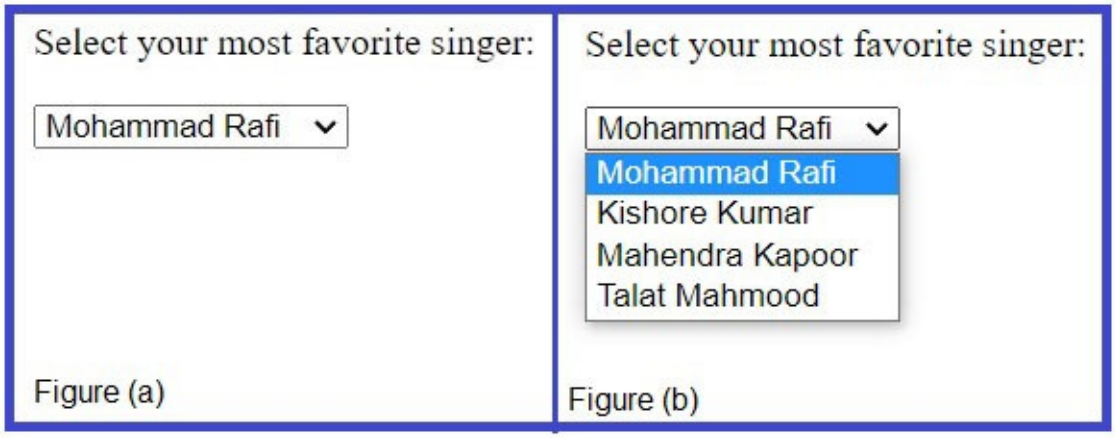

Figure 4.10: *A drop-down listbox in action. Click on the drop-down button (downward pointing arrow head) on the listbox and a list drops down as shown in Figure (b). Now, click on the desired list item to select it. The first list item (here, "Mohammad Rafi") is selected by default.*

This list box is fitted in the LOCs 10-15. LOC 10 consists of the opening tag **`<select>`** and LOC 15 consists of the closing tag **`</select>`**. In LOC 10, the attribute **`name`** is assigned the value **`singer`**. This is an important attribute and we are required to assign a suitable value to this attribute. LOC 11 consists of the element **`<option>`**. In fact, we have fitted the opening tag **`<option>`** closing tag </option> and the content of this element in a single LOC to save space. In this LOC (that is, LOC 11), the attribute **`value`** is assigned the value **`rafi`**. We are required to assign a distinct value to this attribute for every list item. This LOC is responsible for the generation of the list item **`Mohammad Rafi`** in this list box. Similarly, the LOCs 12-14 generate three more list items, namely, **`Kishore Kumar`**, **`Mahendra Kapoor`**, and **`Talat Mahmood`**.

In this list box, you can select only one list item akin to a group of radio buttons. Here, the first list item is selected by default as shown in *Figure 4.10 (a)*. However, you can force any list item to be selected by default simply by putting the attribute **`selected`** in the element **`<option>`**. For example, suppose you want the list item **`Mahendra Kapoor`** to be selected by default, then replace the LOC 13 in this HTML document with the following LOC:

```
13.         <option value="mahrendra" selected>Mahendra Kapoor</option>
```

Code Snippet: 4.12

Run this HTML document in the web browser again, and you will find that this time list item **`Mahendra Kapoor`** is selected by default, instead of **`Mohammad Rafi`**.

Now, let us see how to create a list box of the second type in which we can select multiple list items. In order to do so, simply replace the LOC 10 in this HTML document with the following LOC:

```
10.       <select name="singers" size="4" multiple>
```

Code Snippet: 4.13

Run this HTML document in the web browser again, and you will find that this time a list box shown in *Figure 4.11* is displayed on the screen. This is not a drop-down list box as the list is already dropped down. Instead of a drop-down button, this list box is equipped with a scrollbar. The height of this list box is such that 4 list items are displayed. This is because we have set the attribute **`size`** with the value **`4`** as shown in the preceding LOC 10. In order to select the multiple list items, press the Control key and hold it down, then click on all the list items to be selected. Finally, release the Control key. In this list box, you can select multiple list items because we have inserted the attribute **`multiple`** in the LOC 10 previously:

Figure 4.11: *A list box in which you can select multiple list items and not just one. Press and hold down the Control key. Then, click on all the list items to be selected, one by one. Finally, release the Control key.*

Using labels

Labels are used for (what else!) labeling the items. HTML provides the element **`<label>`** for creating the labels. We can use the **`<label>`** element in the following two ways:

- Label embedding another form control
- Label that is separate from form control

Let us deal with these two ways, one by one. Firstly, let us develop a simple HTML document in which the label has embedded a form control (first way). Open the file **`template3.html`** and save it with the name **`p35.html`** in the folder **`myFiles`**. Now, place the keyboard cursor in the **`body`** element and insert a few LOCs in it as follows:

```
07.   <body>
08.     <form action="" method="post">
09.       <label>Enter your age: <input type="text" name="age" /></label>
10.     </form>
11.   </body>
```

Code Snippet: 4.14

Load this HTML document in the web browser and then you will see the web page with a form displayed on it as shown in *Figure 4.12*. This form consists of a label (**`Enter your age`**) and a textbox. Notice the LOC 09. The **`<input>`** element is completely embedded in the element **`<label>`**. This is how the form control is embedded in a label element:

Figure 4.12: *A label and a textbox. This textbox is embedded in a label element.*

Now, let us develop another HTML document to demonstrate the use of the label element which is separate from form control (the second way). Open the file **`template3.html`** and save it with the name **`p36.html`** in the folder **`myFiles`**. Now, place the keyboard cursor in the **`body`** element and insert a few LOCs in it as follows:

```
07.   <body>
08.     <form action="" method="post">
09.       <label>Select your gender:</label>
10.       <input id="female" type="radio" name="gender" value="f"/>
11.       <label for="female">Female</label>
12.       <input id="male" type="radio" name="gender" value="m"/>
13.       <label for="male">Male</label>
14.     </form>
15.   </body>
```

Code Snippet: 4.15

Load this HTML document in the web browser and then you will see the web page with a form displayed on it as shown in *Figure 4.13*. This form consists of two radio buttons and three labels. The three labels are as follows: (a) **`Select your gender`**, (b) **`Female`**, and (c) **`Male`**. Here, radio buttons are not embedded in labels. The first label provides the general instruction to users of the website and this label is not linked to any radio button. However, the label **`Female`** is linked to the first radio button and the label **`Male`** is linked to the second radio button. This linking is performed using the common value for attributes **`id`** and **`for`**. Notice that in the LOCs 10 and 11, the value **`female`** is assigned to attributes **`id`** and **`for`**. Similarly, in the LOCs 12 and 13, the value **`male`** is assigned to attributes **`id`** and **`for`**.

For both radio buttons, the attribute **`name`** is assigned a common value **`gender`**. The **`name`** is the same for both radio buttons because these radio buttons belong to a single group and hence, they must have a common name.

The attribute **`value`** has different values for both radio buttons. For the first radio button, the attribute **`value`** has assigned the value **`f`** (LOC 10) and for the second radio button, the attribute **`value`** has assigned the value **`m`** (LOC 12). These values must be different because using these values the information about the selected radio button is sent to the server-side program:

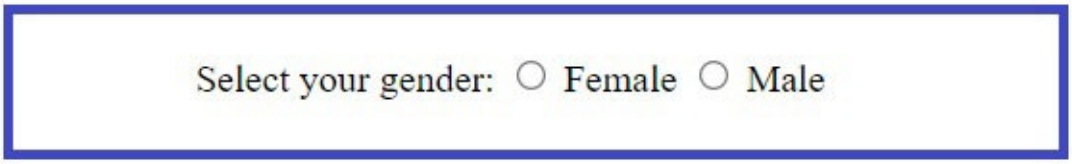

Figure 4.13: *In this form, the form controls (radio buttons) are not embedded in the label.*

Grouping the controls

Sometimes, we are required to group a few controls in a form in a stylish way. For example, *Figure 4.14* shows contact details grouped in a stylish way. In *Figure 4.14*, three contact details, namely, Email, Mobile, and Telephone are grouped in a stylish way. This can be done by using the elements **`<fieldset>`** and **`<legend>`**. The element **`<fieldset>`** draws a border around the group of form controls. The element **`<legend>`** inserts a legend (that is, a caption). Here, the legend is **`Contact Details`**.

Now, let us develop a simple HTML document to demonstrate the use of elements **`<fieldset>`** and **`<legend>`**. Open the file **`template3.html`** and save it with the **`name p37.html`** in the folder **`myFiles`**. Now, place the keyboard cursor in the **`body`** element and insert a few LOCs in it as follows:

```
07.   <body>
08.     <form action="" method="post">
09.       <fieldset>
10.         <legend>Contact Details</legend>
11.           <label>Email:<br/>
12.            <input type="text" name="email"/>
13.           </label><br/>
14.           <label>Mobile:<br/>
15.            <input type="text" name="mobile"/>
16.           </label><br/>
17.           <label>Telephone:<br/>
18.             <input type="text" name="telephone"/>
19.           </label>
20.       </fieldset>
21.     </form>
22.   </body>
```

Code Snippet: 4.16

Load this HTML document in the web browser and then you will see the web page with a form displayed on it as shown in *Figure 4.14*. Here, we have not set the width of the fieldset; hence, it occupies the complete width of the screen available to it. In the latter part of the book, you will learn how to control the width of the fieldset using cascading style sheets. LOCs 09 and 20 consist of the opening tag **`<fieldset>`** and closing tag **`</fieldset>`**, respectively. LOC 10 consists of the complete element **`<legend>`**. This fieldset encloses three textboxes and three labels. There are three labels, namely, Email, Mobile, and Telephone, and each label embeds a textbox:

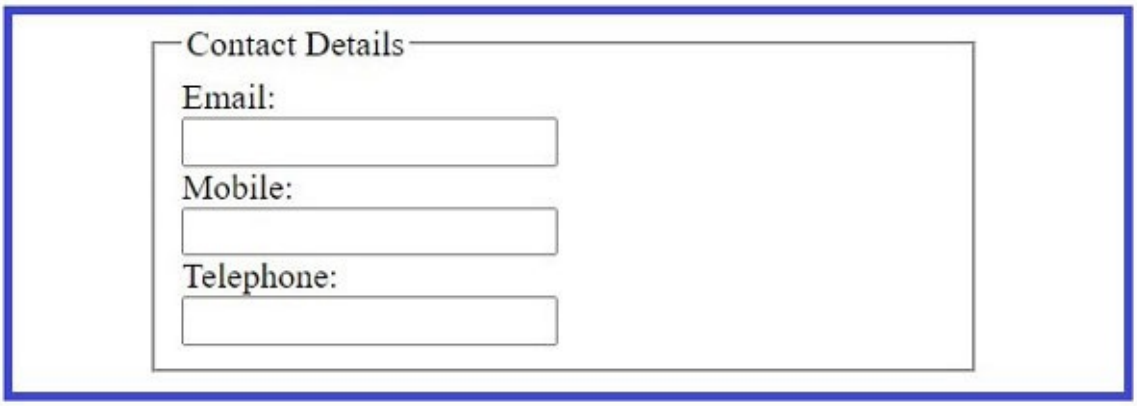

Figure 4.14: *In this form, the elements <fieldset> and <legend> are used to group the contact details in style.*

Conclusion

In this chapter, you learned to create a table on a web page, create a table with a column span and row span, and create a professional table. You also learned to create a form on a web page and put various form controls on it to accept text, password, email ID, URL, phone number, and date. In addition, you learned to place the following controls on a form: radio buttons, checkboxes, list boxes, and labels. Finally, you learned to group the various controls in the form. In the next chapter, you will learn to use the cascading style sheets in order to improve the look of web pages. Particularly, you will learn to set fonts, font sizes, font colors, background colors, alignment of text, using selectors, and using the box model.

Further readings/references

- *HTML & CSS: The Complete Reference*, by Thomas A. Powell, 5/e, McGraw Hill, 2010.
- **https://www.internetingishard.com/**.
- *Beginning HTML and CSS*, by Rob Larsen, Wrox, John Wiley and Sons, 2013.
- *HTML: A Beginner's Guide*, by Wendy Willard, 4/e, McGraw Hill, 2009.

CHAPTER 5

Welcome to CSS

Introduction

In this chapter, you will learn to use internal and external style sheets, universal selectors, type selectors, id selectors, and class selectors to format the text using style sheets. You will also learn to use various CSS combinators, namely, child selectors, descendant selectors, adjacent sibling selectors, and general sibling selectors.

Structure

In this chapter, we will discuss the following topics:

- Internal and external style sheets
- ID selector and class selector
- Formatting the text using style sheets
- Universal selector
- Type selector
- ID selector
- Class selector
- CSS combinators
- Child selector
- Descendant selector
- Adjacent sibling selector
- General sibling selector

Objectives

After reading this chapter, you will be able to use internal and external style sheets, universal selectors, type selectors, ID selectors, and class selectors to format the text using style sheets. Finally, you will be able to use various CSS combinators, namely, child selectors, descendant selectors, adjacent sibling selectors, and general sibling selectors.

Internal and external style sheets

You are already introduced to styles in the section *Creating a table in a web page* in *Chapter 4, Using Forms and Tables*. You are advised to read this specified section before reading this chapter.

The basic idea behind using **CSS** (**Cascading Style Sheets**) is to separate contents from styling them. According to this idea, the content of the web page should be stuffed in HTML files and the appearance of the web page should be left to CSS files.

We can use CSS in the following two ways:

- Using internal style sheets
- Using external style sheets

The style sheet used in the section *Creating a table in a web page* in *Chapter 4, Using Forms and Tables* is an example of an internal style sheet because it is included in the HTML file, namely, p26.html. When you use an external style sheet, you are required to create a separate file with the filename extension **`.css`** and also mention the name of that CSS file in the HTML file. In what follows now, we will write two simple programs to demonstrate these two ways, namely, internal style sheets (**`p40.html`**) and external style sheets (**`p41.html`** and **`p41.css`**).

In these two programs, styles will be applied to the **`<p>`** element and the **`<h3>`** element.

Firstly, let us create the file **`p40.html`**. Open the file **`template.html`**, modify it as follows, and save it with the filename **`p40.html`** in the folder **`myFiles`** and then close it:

```
01. <!DOCTYPE html>
02. <html lang="en">
03.   <head>
04.     <meta charset="utf-8">
05.     <title>Using Cascading Style Sheets</title>
06.     <style type="text/css">
07.       p, h3 {
08.         font-family: arial;
09.         font-size: 30px;
10.         color: blue;
11.       }
12.     </style>
13.   </head>
14.   <body>
15.     <p>Hello Friends</p>
16.     <h2>Hello Boys</h2>
17.     <h3>Hello Girls</h3>
18.   </body>
19. </html>
```

Code Snippet: 5.1

Load this HTML document in a web browser and your screen will display the three lines of text as shown in *Figure 5.1*:

Figure 5.1: *This web page has used styles to set the font, font size, and color of the first line and third line of text.*

LOCs 06-12 consist of the **<style>** element. LOCs 07-11 consist of CSS rules. As stated in LOC 07, **p** and **h3** are selectors. LOCs 07-11 (sans the selectors **p** and **h3**) consist of declaration. In this declaration, three properties are set with suitable values. In LOC 08, the property **font-family** is set to the value **Arial**. You can set this property using multiple values as follows:

```
font-family: Arial, "Times New Roman", Helvetica;
```

In this case, if font Arial is not available, then font Times New Roman will be used and if it is also not available, then font Helvetica will be used. You can certainly write font names in lowercase as we did in LOC 08. Also, if the font name is multi-word, then it must be enclosed between double quotes like **Times New Roman**.

In LOC 09, the property **font-size** is set to the value **30px**. Here, **px** stands for **pixels**.

You can also set the font-size as a percentage of the font-size of the parent element. For example, the following LOC:

```
font-size: 150%;
```

will increase the font-size to 1.5 times that of the font-size of the parent element.

There are a number of keywords that you can use to set the font-size as follows: xx-small, x-small, small, medium, large, x-large, xx-large. You can use these keywords to set the font-size as follows:

```
font-size: small;
```

In LOC 10, the property **color** is set to the value **blue**. This property sets the color of the font (text). Using keywords such as **blue** is the easiest way to set the value of this property. There is a good number of keywords to set the color in CSS such as white, black, blue, green, red, yellow, cyan, magenta, brown, aqua, azure, and so on.

You can also set this property using the three decimal numbers D, D, and D as follows:

```
color: rgb(D, D, D);
```

Where **D** is a decimal number that ranges from 0 to 255. Here, **rgb** stands for **red green blue**. The first **D** sets the value of red, the second **D** sets the value of green and the third **D** sets the value of blue. The final color is the resultant of red, green, and blue.

You can also set this property using the three hexadecimal numbers D, D, and D as follows:

```
color: #DDD;
```

Where **D** is a hexadecimal number that ranges from 0 to FF. For example:

```
color: #00890F;
```

Here, **00** is the value of red, **89** is the value of green, and **0F** is the value of blue.

These properties are applied to LOC 15 (element **p**) and LOC 17 (element **h3**). However, these properties are not applied to LOC 16 because it contains the element **h2** and it is not in the list of selectors in LOC 07.

Now, let us create a similar web page using an external style sheet. Open the file **template.html**, and modify it as follows:

```
01. <!DOCTYPE html>
02. <html lang="en">
03.   <head>
04.     <meta charset="utf-8">
05.     <title>Using Cascading Style Sheets</title>
06.     <link rel="stylesheet" type="text/css" href="pXX.css" media="screen" />
07.   </head>
08.   <body>
09.   </body>
10. </html>
```

Code Snippet: 5.2

Now, save this file with the filename **template4.html** in the folder **myFiles** and then close it.

Open the file **template4.html** and save it with the name **p41.html** in the folder **myFiles**. Place the keyboard cursor in the LOC 06 and change the name of the CSS file as follows (notice that the name of the CSS file is changed from **pXX.css** to **p41.css**):

```
06.     <link rel="stylesheet" type="text/css" href="p41.css" media="screen" />
```

Code Snippet: 5.3

Place the keyboard cursor in the **body** element and insert a few LOCs in it as follows:

```
08.   <body>
09.     <p>Hello Friends</p>
10.     <h2>Hello Boys</h2>
11.     <h3>Hello Girls</h3>
12.   </body>
```

Code Snippet: 5.4

Now, let us create the CSS file **p41.css**. Open Notepad (or your favorite editor) and type the following text in it:

```
01. p, h3 {
02.   font-family: arial ;
03.   font-size: 30px ;
04.   color: blue ;
05. }
```

Code Snippet: 5.5

Save this file with the filename **p41.css** in the folder **MyFiles**. Load the file **p41.html** in a web browser and your screen will display the three lines of text as shown in *Figure 5.1*. The only difference between the programs **p40.html** and **p41.html** is that **p40.html** uses an internal style sheet and **p41.html** uses an external style sheet.

Before concluding this section, let me explain the syntax of LOC 06 in the file **p41.html**. This LOC is reproduced for your quick reference as follows:

```
06.     <link rel="stylesheet" type="text/css" href="p41.css" media="screen" />
```

Code Snippet: 5.6

This LOC consists of the **<link>** element. This LOC informs the web browser that this HTML document uses the external style sheet and the concerned CSS file is nothing but **p41.css**. This element consists of four attributes, namely, **rel**, **type**, **href**, and **media**.

The attribute **rel** informs the web browser about the relationship between the current document (**p41.html**) and the linked document (**p41.css**). Here, the value assigned to this attribute is **stylesheet** because the file **p41.css** is nothing but an external style sheet.

The attribute **type** informs the web browser about the type of content of the linked document (**p41.css**). Here, the linked document **p41.css** consists of text and this text is nothing but CSS; hence, the value assigned to this attribute is **text/css**.

The attribute **href** informs the web browser about the name of the linked document with the path. Here, the name of the linked document is **p41.css** and hence it is assigned as a value to this attribute. The path is not required here as the current document and linked document are in the same folder.

The attribute **media** informs the web browser about the type of media on which the web page will be displayed. Here, we intend to display the web page on the screen of the computer or mobile; hence, the value assigned to this attribute is **screen**.

ID selector and class selector

A web page consists of a number of **<p>** elements. Generally, we want different styles to be applied to different **<p>** elements. We can do this using either the **id** selector or the **class** selector. Firstly, let us see how we can do it using the **id** selector (**id** for identifier). We will create the files **p42.html** and **p42.css** to demonstrate the use of the **id** selector.

Open the file **template4.html** and save it with the name **p42.html** in the folder **myFiles**. Place the keyboard cursor in the LOC 06 and change the name of the CSS file as follows (notice that the name of the CSS file is changed from **pXX.css** to **p42.css**):

```
06.     <link rel="stylesheet" type="text/css" href="p42.css" media="screen" />
```

Code Snippet: 5.7

Place the keyboard cursor in the **body** element and insert a few LOCs in it as follows:

```
08.   <body>
09.     <p id="one">This line of text is red in color.</p>
10.     <p id="two">This line of text is green in color.</p>
11.     <p id="three">This line of text is blue in color.</p>
12.     <h3 id="four">This heading is red in color.</h3>
13.     <h3 id="five">This heading is green in color.</h3>
14.     <h3 id="six">This heading is blue in color.</h3>
15.   </body>
```

Code Snippet: 5.8

Now, let us create the CSS file **`p42.css`**. Open Notepad (or your favorite editor) and type the following text in it:

```
01. #one, #four{
02.  color: red ;
03. }
04. #two, #five{
05.  color: green ;
06. }
07. #three, #six{
08.  color: blue ;
09. }
```

Code Snippet: 5.9

Save this file with the filename **`p42.css`** in the folder **`MyFiles`**. Load the file **`p42.html`** in a web browser and your screen will display the six lines of text as shown in *Figure 5.2*:

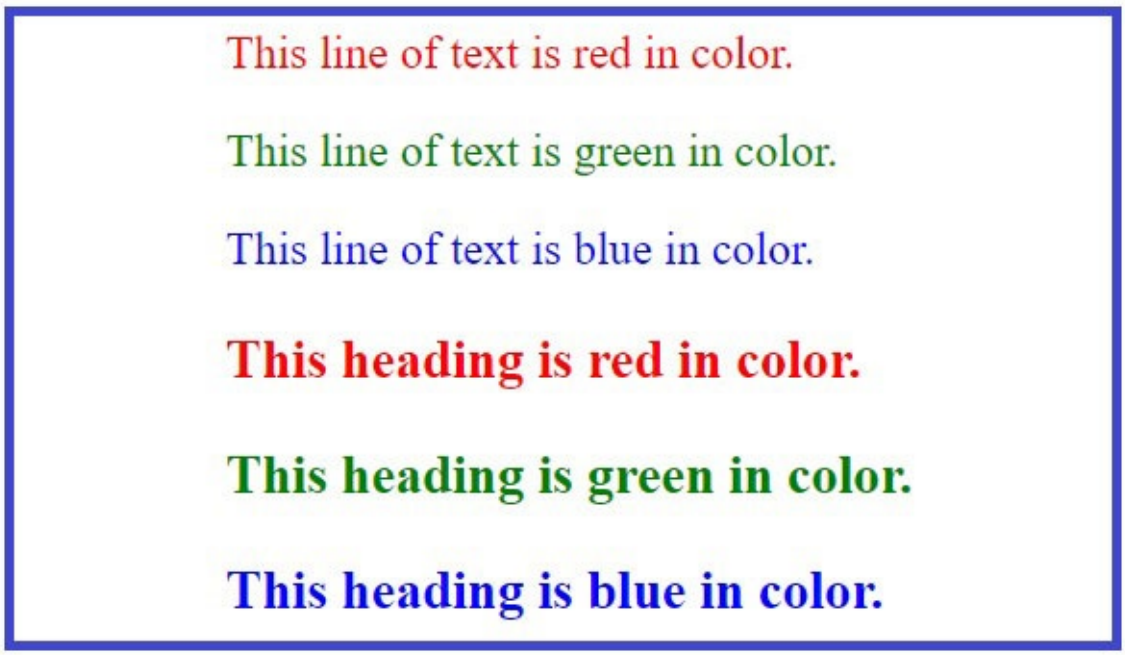

Figure 5.2: *In this web page, the id selector is used to distinguish between various <p> elements, and also between various <h3> elements.*

Notice the LOCs 09-14 in the file **`p42.html`**. In LOC 09, the **`<p>`** element has the attribute **`id`** and this attribute is assigned the value **`one`**. It means the ID name of this **`<p>`** element is **`one`**. Similarly, the ID names of the **`<p>`** elements in LOCs 10 and 11 are **`two`** and **`three`**, respectively.

Similarly, the ID names of the **`<h3>`** elements in LOCs **`12`**, **`13`**, and **`14`** are **`four`**, **`five`**, and **`six`**, respectively.

Note: The ID name must be unique in an HTML document. It means the same ID name should not be used at two places in the HTML file. That's why we have used different ID names ("four," "five," and "six") in LOCs 12, 13, and 14, instead of using the ID names "one", "two", and "three" which are used already in the LOCs 9, 10, and 11.

In the file **`p42.css`**, the LOCs 01-03 consist of CSS rules for **`id`** selectors (or id-names) **`one`** and **`four`**,

the LOCs 04-06 consist of CSS rules for **id** selectors (or id-names) **two** and **five**, and LOCs 0-09 consist of CSS rules for **id** selectors (or id-names) **three** and **six**. The contents of these CSS rules are self-explanatory and hardly need any explanation. In the CSS rule for the **id** selectors **one** and **four**, the color of the font is set to red (LOCs01-03). In the CSS rule for the **id** selectors **two** and **five**, the color of the font is set to green (LOCs04-06). In the CSS rule for **id** selectors **three** and **six**, the color of the font is set to blue (LOCs07-09).

Note: In the CSS file (or in the CSS rule), the ID name is always prefixed with an ampersand symbol (#) as shown in the LOCs 01, 04, and 07 in the file p42.css. Also, in the CSS file (or in CSS rule) the class name is always prefixed with a period (.).

Now, let us develop the programs **p43.html** and **p43.css** in order to demonstrate the use of the **class** selectors.

Open the file **template4.html** and save it with the name **p43.html** in the folder **myFiles**. Place the keyboard cursor in the LOC 06 and change the name of the CSS file as follows (notice that the name of the CSS file is changed from **pXX.css** to **p43.css**):

```
06.     <link rel="stylesheet" type="text/css" href="p43.css" media="screen" />
```

Code Snippet: 5.10

Place the keyboard cursor in the **body** element and insert a few LOCs in it as follows:

```
08.   <body>
09.    <p class="one">This line of text is red in color.</p>
10.    <p class="two">This line of text is green in color.</p>
11.    <p class="three">This line of text is blue in color.</p>
12.    <h3 class="one">This heading is red in color.</h3>
13.    <h3 class="two">This heading is green in color.</h3>
14.    <h3 class="three">This heading is blue in color.</h3>
15.   </body>
```

Code Snippet: 5.11

Now, let us create the CSS file **p43.css**. Open Notepad (or your favorite editor) and type the following text in it:

```
01. .one{
02.  color: red ;
03. }
04. .two{
05.  color: green ;
06. }
07. .three{
08.  color: blue ;
09. }
```

Code Snippet: 5.12

Save this file with the filename **p43.css** in the folder **MyFiles**. Load the file **p43.html** in a web browser and your screen will display the six lines of text as shown in *Figure 5.2*.

The explanation given for the program **p42.html** holds good for the program **p43.html**. The only

difference is that in the program **`p42.html`**, the id selectors are used, whereas, in the program **`p43.html`**, the **`class`** selectors are used. In CSS files, the ID names are prefixed by an ampersand symbol (`#`) as in the file **`p42.css`** and class names are prefixed by period (`.`) as in the file **`p43.css`**. Also, unlike **`id-name`**, **`class-name`** can be repeated in an HTML file, that's why the class names **`one`** and **`two,`** and **`three`** - used in LOCs 9, 10, and 11 - are repeated in LOCs 12, 13, and 14, respectively.

Formatting the text using a style sheet

In order to format the text using the style sheets, we need to set the various properties. These properties are as follows:

- **`font-family`**: This property is used to set the font of the text. The values to be assigned are nothing but a comma-separated list of font names. If the font name is a multi-word, then enclose it in double quotes. The very first available font in the list will be used for setting the font of the text. Example:
  ```
  font-family : arial, “times new roman”, helvetica, serif ;
  ```
- **`font-size`**: This property is used to set the size of the font of the text. Generally, the value in the unit of px (px for pixel) is assigned to this property. Example:
  ```
  font-size : 24px ;
  ```
- **`font-weight`**: This property is used to set the weight of the font of the text, that is, to make the font bold, light, and so on. The comma-separated list of valid values for this property is mentioned here:
  ```
  normal, bold, bolder, lighter, 100, 200, 300, 400, 500, 600, 700, 800, 900
  ```

For example:
```
font-weight :bolder;
```

`font-style`: This property is used to set the style of the font of the text, that is, to make the font normal, italic, oblique, and so on. The comma-separated list of valid values for this property is given here:
```
normal, italic, oblique
```

For example:
```
font-style :oblique;
```

`font-variant`: This property is used to convert the text into **`small-caps`**. The comma-separated list of valid values for this property is mentioned here:
```
normal, small-caps
```

For example
```
font-variant :small-caps;
```

After setting the **`font-variant`** property to **`small-caps`**, the lowercase letters in the text are converted into uppercase, but in smaller font size. The letters which are already capitalized are left undisturbed.

`color`: This property is used to set the color of the text. We have already discussed this property in the section *Internal and external style sheets*.

For example:
```
font-color :red;
```

`text-decoration`: This property is used to decorate the text, that is, to convert the text into underlined, overlined, and so on. The comma-separated list of valid values for this property is given here:
```
none, underline, overline, line-through
```

For example:
```
text-decoration :underline;
```

text-indent: This property is used to indent the text, that is, to apply the left side margin to the first line of a paragraph. An example of setting the value of this property is mentioned here :
```
text-indent :20px;
```

Apart from **px**, you can use any unit of length that is supported by CSS.

text-shadow: This property is used to create a shadow of text. An example of setting the value of this property is given here :
```
text-shadow :blue 4px 6px 3px;
```

The value consists of four items separated by single spaces. The first item (blue) is the color of the shadow. Here, the shadow will be blue in color. The second and third items (**4px** and **6px**) are the X and Y coordinates of the location where the shadow should fall. The fourth item (**3px**) specifies the degree of blur. Apart from **px**, you can use any unit of length that is supported by CSS.

text-transform: This property is used to change the case of text, that is, to make it uppercase, lowercase, and so on. The comma-separated list of valid values for this property is given here:
```
none, capitalize, uppercase, lowercase
```

For example:
```
text-transform :uppercase;
```

letter-spacing: This property is used to set the spacing between two consecutive letters (also known as tracking). For example, use this LOC to set the spacing between two consecutive letters to be **4px**:
```
letter-spacing :4px;
```

Apart from **px**, you can use any unit of length that is supported by CSS.

word-spacing: This property is used to set the spacing between two consecutive words. For example, use this LOC to set the spacing between two consecutive words to **5px**:
```
word-spacing : 5px;
```

Apart from **px**, you can use any unit of length that is supported by CSS.

white-space: It is a habit of HTML that it replaces all occurrences of white spaces with a single space. The output of the Tab key, Enter key, and Space bar is termed white space. However, this property allows us to control the occurrences of white spaces on the web page.

The comma-separated list of valid values for this property is given here:
```
normal, pre, nowrap
```

For example:
```
white-space :pre;
```

The value **none** refers to the default situation (that is, occurrences of white spaces are replaced by a single space). The value **pre** preserves the white spaces in the text. The value **nowrap** allows a line-break only if the element **
** is used.

direction: Normally, the text flows from left to right. But sometimes, you need to change the direction of the flow of text and this property helps you in doing so.

The comma-separated list of valid values for this property is as follows:
```
ltr, rtl, inherit
```

For example:
```
direction :rtl;
```

The value **ltr** refers to the default situation (that is, from left to right). The value **rtl** sets the flow of text

from right to left. The value **inherit** sets the direction of the flow of text the same as in the parent element.

text-align: This property sets the alignment of text with respect to the containing element. The comma-separated list of valid values for this property is mentioned here:

```
left, right, center, justify, start, end
```

For example:

```
text-align :right;
```

The value **left** aligns the text with the **left** border of the containing element, and so on.

vertical-align: This property is used to vertically align the text with respect to an inline image. The comma-separated list of valid values for this property is as follows:

```
baseline, sub, super, top, text-top, middle, bottom, text-bottom
```

For example:

```
vertical-align :middle;
```

The value **baseline** aligns the text to the baseline of the inline image, and so on.

Using selectors

You are already familiar with the selectors. In this section, we again deal with the selectors.

Universal selector

The Universal selector represents all elements in a web page. It is something like a wild card. Let us experiment with it. Open the file **template4.html** and save it with the name **p44.html** in the folder **myFiles**. Place the keyboard cursor in the LOC 06 and change the name of the CSS file as follows (notice that the name of the CSS file is changed from **pXX.css** to **p44.css**):

```
06.     <link rel="stylesheet" type="text/css" href="p44.css" media="screen" />
```

Code Snippet: 5.13

Place the keyboard cursor in the **body** element and insert a few LOCs in it as follows:

```
08.   <body>
09.    <p>Line of text for demonstration.</p>
10.    <h2>Line of text for demonstration.</h2>
11.    <h3>Line of text for demonstration.</h3>
12.    <h4>Line of text for demonstration.</h4>
13.   </body>
```

Code Snippet: 5.14

Now, let us create the CSS file **p44.css**. Open Notepad (or your favorite editor) and type the following text in it:

```
01. *{
02.  color: blue ;
03. }
```

Code Snippet: 5.15

Save this file with the filename **`p44.css`** in the folder **`MyFiles`**. Load the file **`p44.html`** in a web browser and your screen will display the four lines of text as shown in *Figure 5.3*. Notice that all four lines are now blue because the selector in LOC 01 (in the file **`p44.css`**) is universal selector **`*`** and hence, the property setting in LOC 02 is applied to all four elements in the file **`p44.html`**, namely, **`<p>`**, **`<h2>`**, **`<h3>`**, and **`<h4>`**:

Figure 5.3: *On this web page, all four lines of text are blue. This web page demonstrates the use of a universal selector *.*

Type selector

We have already used the type selectors in programs **`p40.html`** and **`p41.html`**. The **`type`** selectors used in these programs are **`<p>`** and **`<h3>`**.

ID selector

We have already used the ID selectors in the program **`p42.html`**. The **`id`** selectors used in this program are **`one`**, **`two`**, **`three`**, **`four`**, **`five`**, and **`six`**.

Class selector

We have already used the **`class`** selectors in the program **`p43.html`**. The class selectors used in this program are **`one`**, **`two`**, and **`three`**.

We can use a **`class`** selector using one of the two methods:

- **First method:** In this method, we assign only one value to the attribute **`class`**. We have used this method in the programs **`p43.html`** and **`p43.css`**. Also, the program **`p45.html`** again demonstrates the use of this method.
- **Second method:** In this method, we can assign multiple values to the attribute **`class`**. For example, notice this piece of code:

```
<p class="one two">This is line of text.</p>
```

Code Snippet: 5.16

Let the linked CSS file contain the following code:

```
.one{
color : blue ;
}
.two{
font-style : italic ;
}
```

Code Snippet: 5.17

Now, the line of text displayed on the web page will be as follows (it will be blue in color and also have the Italics font style):

```
This is line of text
```

Here, the two class selectors **`one`** and **`two`** are assigned to the **`class`** attribute of **`<p>`** element in a single line of code. Hence, both the properties of the **`<p>`** element are set accordingly (**`color`** as well as **`font-style`** of the **`<p>`** element is set accordingly).

The programs **`p45.html`** (demonstrates the first method) and **`p46.html`** (demonstrates the second method) are mentioned in the next line.

Open the file **`template4.html`** and save it with the name **`p45.html`** in the folder **`myFiles`**. Place the keyboard cursor in the LOC 06 and change the name of the CSS file as follows (notice that the name of the CSS file is changed from **`pXX.css`** to **`p45.css`**):

```
    <link rel="stylesheet" type="text/css" href="p45.css" media="screen" />
```

Code Snippet: 5.18

Place the keyboard cursor in the **`body`** element and insert a few LOCs in it as follows:

```
  <body>
   <p class="one">This line of text is red in color.</p>
   <p class="two">This line of text is green in color.</p>
   <p class="three">This line of text is blue in color.</p>
   <h3 class="one">This line of text is yellow in color.</h3>
   <h3 class="two">This line of text is green in color.</h3>
   <h3 class="three">This line of text is blue in color.</h3>
  </body>
```

Code Snippet: 5.19

Now, let us create the CSS file **`p45.css`**. Open Notepad (or your favorite editor) and type the following text in it:

```
p.one{
 color: red ;
}
h3.one{
 color: yellow ;
}
.two{
 color: green ;
}
.three{
 color: blue ;
}
```

Code Snippet: 5.20

Save this file with the filename **p45.css** in the folder **MyFiles**. Load the file **p45.html** in a web browser and your screen will display the six lines of text as shown in *Figure 5.4*:

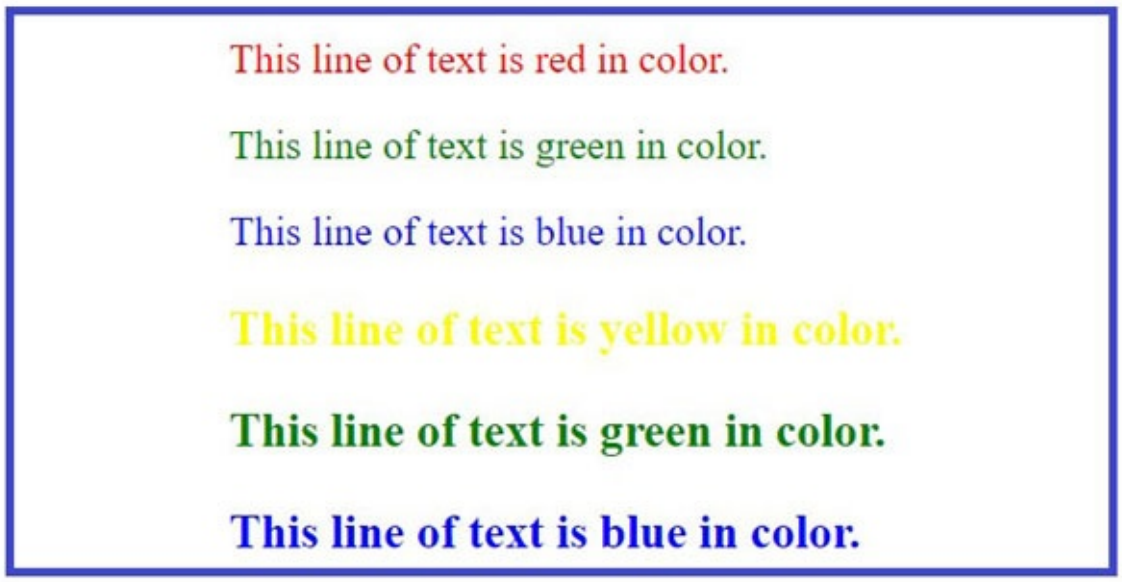

Figure 5.4: *This web page demonstrates the use of class selectors according to the first method. See program p45.html.*

Notice that the selector **p.one** applies only to LOC 09 and the selector **h3.one** applies only to LOC 12 in the file **p45.html**. Selector **two** applies to LOCs 10 and 13, and the selector **three** applies to LOCs 11 and 14 in the file **p45.html**.

Now, let us create the program **p46.html**. Open the file template4.html and save it with the name **p46.html** in the folder **myFiles**. Place the keyboard cursor in the LOC 06 and change the name of the CSS file as follows (notice that the name of the CSS file is changed from **pXX.css** to **p46.css**):

```
06.     <link rel="stylesheet" type="text/css" href="p46.css" media="screen" />
```

Code Snippet: 5.21

Place the keyboard cursor in the **body** element and insert a few LOCs in it as follows:

```
08.   <body>
09.    <p class ="one two">This is line of text.</p>
10.   </body>
```

Code Snippet: 5.22

Now, let us create the CSS file **p46.css**. Open Notepad (or your favorite editor) and type the following text in it:

```
01.  .one {
02.    color : blue ;
03.  }
04.  .two {
05.    font-style : italic ;
06.  }
```

Code Snippet: 5.23

Save this file with the filename **`p46.css`** in the folder **`MyFiles`**. Load the file **`p46.html`** in a web browser and your screen will display the line of text as follows:

```
This is line of text
```

CSS combinators

CSS combinator defines the relationship between the selectors. There are four different combinators in CSS as follows:

- **Child selector (>)**: Here, the symbol is rightward pointing arrowhead.
- **Descendant selector ( )**: Here, the symbol is nothing but a single space.
- **Adjacent sibling selector (+)**: Here, the symbol is the summation symbol.
- General sibling selector (~): Here, the symbol is tilde.

Notice the following piece of code:

```
<div>
  <section>
    <p>First line of text.</p>
    <p>Second line of text.</p>
    <h1>Third line of text.</h1>
  </section>
</div>
```

Code Snippet: 5.24

In this piece of code, we have the following elements (refer to *Figure 5.5*):

- One **`<div>`** element
- One **`<section>`** element
- Two **`<p>`** elements
- One **`<h1>`** element

Tree diagram of these elements is shown in *Figure 5.5*:

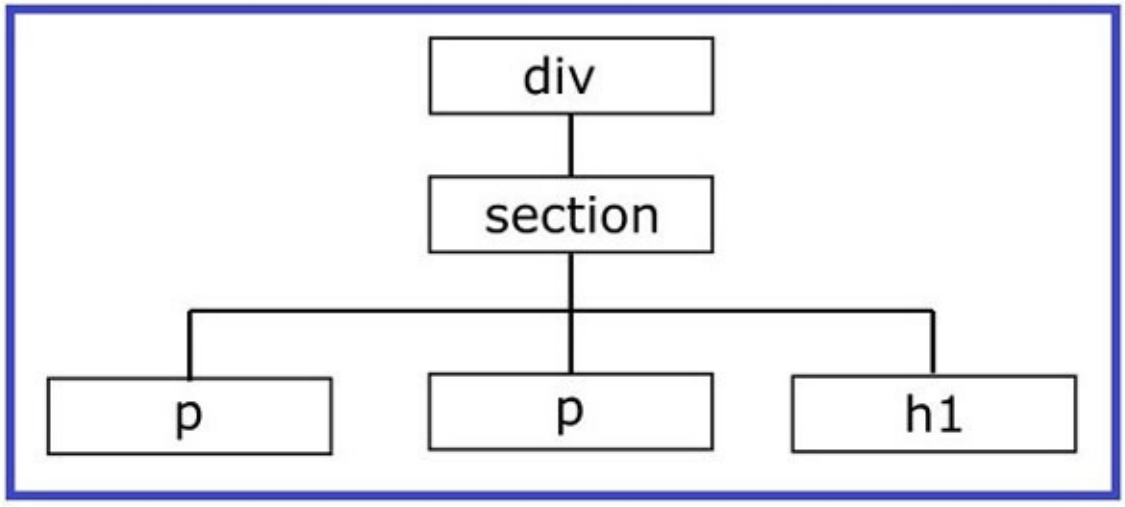

Figure 5.5: *Tree diagram of various elements used in the HTML source code.*

- **Child elements**: Let's take a look at *Figure 5.5*. Here, the **<section>** element is a child element of the ** ** element and the ** ** element is the parent element of the **<section>** element. Also, the two **<p>** elements and **<h1>** element are children elements of the **<section>** element and the **<section>** element is the parent element of the two **<p>** elements and the **<h1>** element. Finally, the two **<p>** elements and **<h1>** element have no child elements.
- **Descendant elements**: Let's take a look at *Figure 5.5*. Here, the **<section>** element, two **<p>** elements, and the **<h1>** element are descendant elements of the ** ** element; the ** ** element is the ancestor element of the **<section>** element, and the two **<p>** elements and **<h1>** element. Also, the two **<p>** elements and the **<h1>** element are descendant elements of the **<section>** element, and the **<section>** element is the ancestor element of two **<p>** elements and the **<h1>** element. Finally, the two **<p>** elements and the **<h1>** element have no descendant elements.
- **Sibling elements**: If two or more elements have a common parent element, then these children elements are known as sibling elements. Refer to *Figure 5.5*. Here, the ** ** element has no sibling. Also, the **<section>** element has no sibling. Finally, the two **<p>** elements and the **<h1>** element are siblings of each other because they have a common parent **<section>** element.
- **Adjacent sibling element**: If sibling **Two** immediately follows sibling **One,** then sibling **Two** is the adjacent sibling element of sibling **One**. Refer to *Figure 5.5*. Here, the second **<p>** element is an adjacent sibling of the first **<p>** element. Also, the **<h1>** element is an adjacent sibling of the second **<p>** element.
- **General sibling element**: If elements **A** and **B** are siblings, and if **B** follows **A** (not necessarily immediately), then **A** is the general sibling of **A**.

Child selector

The meaning of child selector is explained in the preceding section. Now, let us implement child selectors on a suitable web page.

Open the file **template4.html** and save it with the name **p47.html** in the folder **myFiles**. Place the keyboard cursor in the LOC 06 and change the name of the CSS file as follows (notice that the name of the CSS file is changed from **pXX.css** to **p47.css**):

```
06.     <link rel="stylesheet" type="text/css" href="p47.css" media="screen" />
```

Code Snippet: 5.25

Place the keyboard cursor in the **body** element and insert a few LOCs in it as follows:

```
08.  <body>
09.   <p>This line of text is black in color.</p>
10.    <div>
11.      <p>This line of text is red in color.</p>
12.    </div>
13.    <div>
14.      <section>
15.        <p>This line of text is black in color.</p>
16.      </section>
17.    </div>
18.    <h3>This line of text is black in color.</h3>
19.    <div>
20.      <h3>This line of text is blue in color.</h3>
21.    </div>
22.    <div>
23.      <section>
24.        <h3>This line of text is black in color.</h3>
25.      </section>
26.    </div>
27.  </body>
```

Code Snippet: 5.26

Now, let us create the CSS file **`p47.css`**. Open Notepad (or your favorite editor) and type the following text in it:

```
01. div > p{
02.  color: red ;
03. }
04. div > h3{
05.  color: blue ;
06. }
```

Code Snippet: 5.27

Save this file with the filename **`p47.css`** in the folder **`MyFiles`**. Load the file **`p47.html`** in a web browser and your screen will display the six lines of text as shown in *Figure 5.6*.

The CSS file **`p47.css`** consists of two CSS rules. The first rule is in the LOCs 01-03. The second rule is in the LOCs 04-06.

In the first rule, the color of the text is set to red. Also, in the first rule, the selector is every **`<p>`** element which is a child of the **`<div>`** element. This condition is satisfied only by the **`<p>`** element in the LOC 11 in the file **`p47.html`**. Hence, this line of text appears to be red on the web page displayed on the screen.

In the second rule, the color of the text is set to blue. Also, in the second rule, the selector is every **`<h3>`** element which is a child of the **`<div>`** element. This condition is satisfied only by the **`<h3>`** element in the LOC 20 in the file **`p47.html`**. Hence, this line of text appears to be blue on the web page displayed on the screen. All the remaining lines of text appear in the default black color:

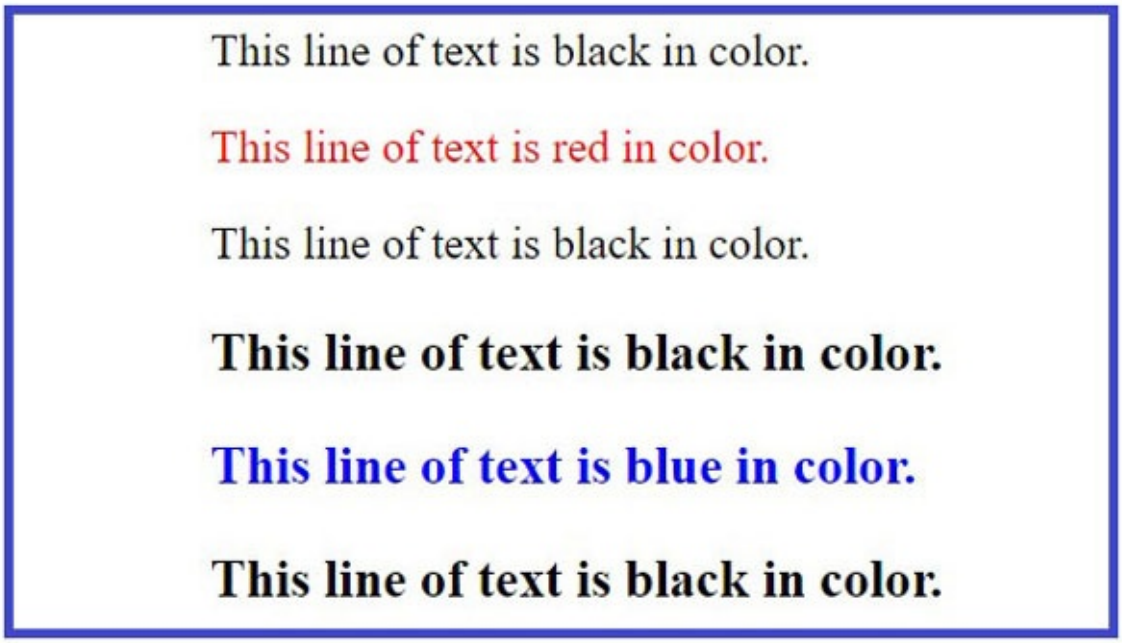

Figure 5.6: *Demonstration of a child selector. See the program p47.html.*

Descendant selector

The meaning of a descendant selector is explained in the preceding section. Now, let us implement descendant selectors on a suitable web page.

Open the file **`template4.html`** and save it with the name **`p48.html`** in the folder **`myFiles`**. Place the keyboard cursor in the LOC 06 and change the name of the CSS file as follows (notice that the name of the CSS file is changed from **`pXX.css`** to **`p48.css`**):

```
06.     <link rel="stylesheet" type="text/css" href="p48.css" media="screen" />
```

Code Snippet: 5.28

Place the keyboard cursor in the **`body`** element and insert a few LOCs in it as follows:

```
08.   <body>
09.    <p>This line of text is black in color.</p>
10.     <div>
11.       <p>This line of text is red in color.</p>
12.     </div>
13.     <div>
14.       <section>
15.         <p>This line of text is red in color.</p>
16.       </section>
17.     </div>
18.     <h3>This line of text is black in color.</h3>
19.     <div>
20.       <h3>This line of text is blue in color.</h3>
21.     </div>
22.     <div>
23.       <section>
24.         <h3>This line of text is blue in color.</h3>
25.       </section>
26.     </div>
27.   </body>
```

Code Snippet: 5.29

Now, let us create the CSS file **`p48.css`**. Open Notepad (or your favorite editor) and type the following text in it:

```
01. div p{
02.  color: red ;
03. }
04. div h3{
05.  color: blue ;
06. }
```

Code Snippet: 5.30

Save this file with the filename **`p48.css`** in the folder **`MyFiles`**. Load the file **`p48.html`** in a web browser and your screen will display the six lines of text as shown in *Figure 5.7*.

The CSS file **`p48.css`** consists of two CSS rules. The first rule is in the LOCs 01-03. The second rule is in the LOCs 04-06.

In the first rule, the color of the text is set to red. Also, in the first rule, the selector is every **`<p>`** element which is a descendant of the **`<div>`** element. This condition is satisfied only by the **`<p>`** element in the LOC 11 in the file **`p48.html`**. Hence, this line of text appears to be red on the web page displayed on the screen.

In the second rule, the color of the text is set to blue. Also, in the second rule, the selector is every **`<h3>`** element which is a descendant of the **`<div>`** element. This condition is satisfied only by the **`<h3>`** element in the LOC 20 in the file **`p48.html`**. Hence, this line of text appears to be blue on the web page displayed on the screen.

All the remaining lines of text appear in the default black color:

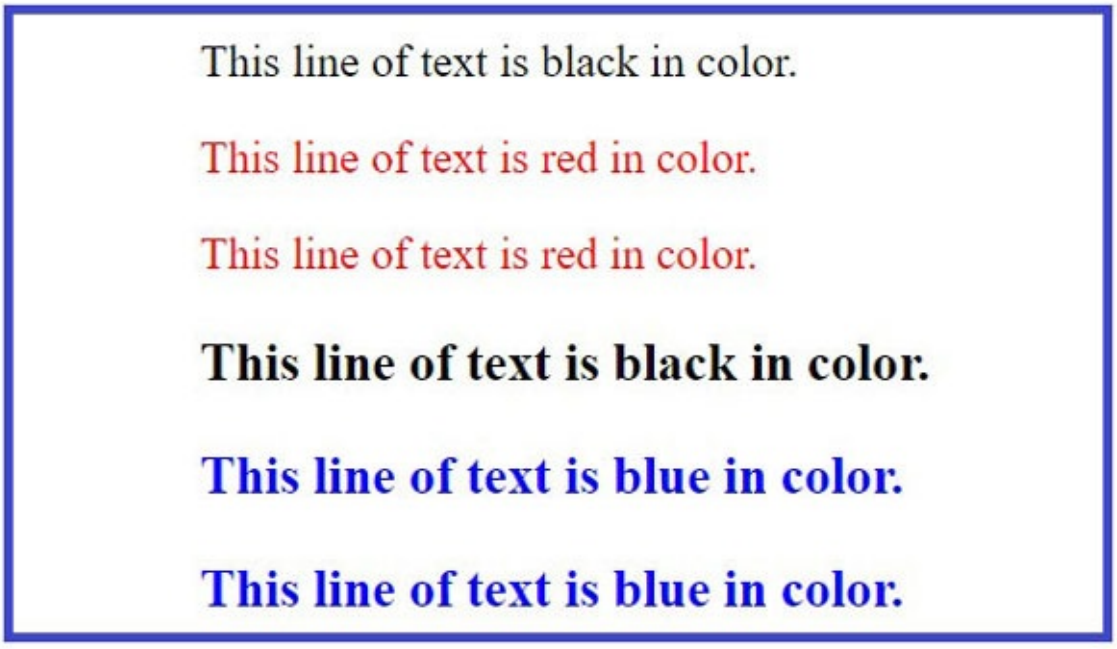

Figure 5.7: *Demonstration of the descendant selector. See the program p48.html.*

Adjacent sibling selector

The meaning of an adjacent sibling selector is explained in the preceding section. Now, let us implement adjacent sibling selectors on a suitable web page.

Open the file **`template4.html`** and save it with the name **`p49.html`** in the folder **`myFiles`**. Place the keyboard cursor in the LOC 06 and change the name of the CSS file as follows (notice that the name of the CSS file is changed from **`pXX.css`** to **`p49.css`**):

```
06.     <link rel="stylesheet" type="text/css" href="p49.css" media="screen" />
```

Code Snippet: 5.31

Place the keyboard cursor in the **`body`** element and insert a few LOCs in it as follows:

```
08.   <body>
09.    <p>This line of text is black in color.</p>
10.     <div>
11.       <p>This line of text is black in color.</p>
12.     </div>
13.     <p>This line of text is red in color.</p>
14.     <p>This line of text is black in color.</p>
15.     <h3>This line of text is black in color.</h3>
16.     <div>
17.       <h3>This line of text is black in color.</h3>
18.     </div>
19.     <h3>This line of text is blue in color.</h3>
20.     <h3>This line of text is black in color.</h3>
21.   </body>
```

Code Snippet: 5.32

Now, let us create the CSS file **`p49.css`**. Open Notepad (or your favorite editor) and type the following

text in it:

```
01. div + p{
02.  color: red ;
03. }
04. div + h3{
05.  color: blue ;
06. }
```

Code Snippet: 5.33

Save this file with the filename **`p49.css`** in the folder **`MyFiles`**. Load the file **`p49.html`** in a web browser and your screen will display the six lines of text as shown in *Figure 5.8*.

The CSS file **`p49.css`** consists of two CSS rules. The first rule is in the LOCs 01-03. The second rule is in the LOCs 04-06.

In the first rule, the color of the text is set to red. Also, in the first rule, the selector is every **`<p>`** element which is an adjacent sibling of the **`<div>`** element. This condition is satisfied only by the **`<p>`** element in LOC 13 in the file **`p49.html`**. Hence, this line of text appears to be red on the web page displayed on the screen.

In the second rule, the color of the text is set to blue. Also, in the second rule, the selector is every **`<h3>`** element which is an adjacent sibling of the **`<div>`** element. This condition is satisfied only by the **`<h3>`** element in the LOC 19 in the file **`p49.html`**. Hence, this line of text appears to be blue on the web page displayed on the screen.

All the remaining lines of text appear in the default black color:

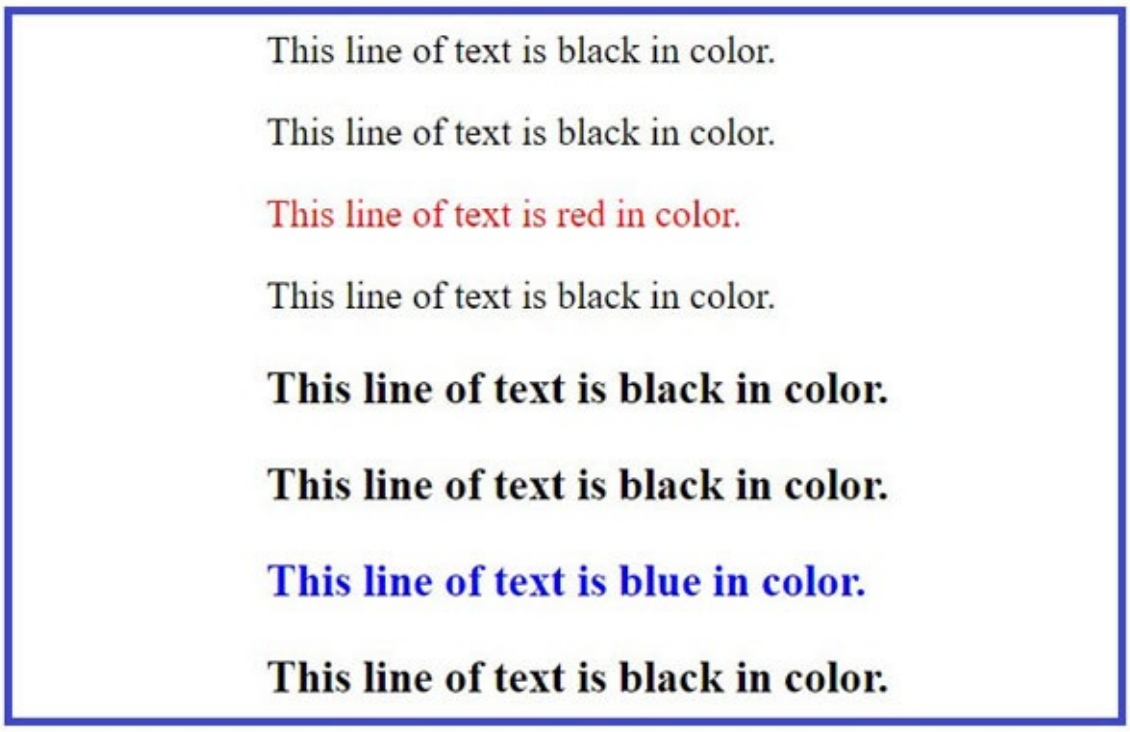

Figure 5.8: *Demonstration of an adjacent sibling selector. See the program p49.html.*

General sibling selector

If elements **`A`** and **`B`** are siblings, and if **`B`** follows **`A`** (not necessarily immediately), then **`B`** is the general

sibling of **A**.

Open the file **template4.html** and save it with the name **p50.html** in the folder **myFiles**. Place the keyboard cursor in the LOC 06 and change the name of the CSS file as follows (notice that the name of the CSS file is changed from **pXX.css** to **p50.css**):

```
06.     <link rel="stylesheet" type="text/css" href="p50.css" media="screen" />
```

Code Snippet: 5.34

Place the keyboard cursor in the **body** element and insert a few LOCs in it as follows:

```
08.   <body>
09.    <p>This line of text is black in color.</p>
10.     <div>
11.       <p>This line of text is black in color.</p>
12.     </div>
13.     <p>This line of text is red in color.</p>
14.     <p>This line of text is red in color.</p>
15.     <h3>This line of text is blue in color.</h3>
16.     <div>
17.       <h3>This line of text is black in color.</h3>
18.     </div>
19.     <h3>This line of text is blue in color.</h3>
20.     <h3>This line of text is blue in color.</h3>
21.   </body>
```

Code Snippet: 5.35

Now, let us create the CSS file **p50.css**. Open Notepad (or your favorite editor) and type the following text in it:

```
01. div ~ p{
02.  color: red ;
03. }
04. div ~ h3{
05.  color: blue ;
06. }
```

Code Snippet: 5.36

Save this file with the filename **p50.css** in the folder **MyFiles**. Load the file **p50.html** in a web browser and your screen will display the six lines of text as shown in *Figure 5.9*.

The CSS file **p50.css** consists of two CSS rules. The first rule is in the LOCs 01-03. The second rule is in the LOCs 04-06.

In the first rule, the color of the text is set to red. Also, in the first rule, the selector is every **<p>** element which is a general sibling of the ** ** element. This condition is satisfied by two **<p>** elements, one in the LOC 13 and the other in the LOC 14, in the file **p50.html**. Notice that these two <p> elements are general siblings of the ** ** element situated in the LOCs10-12. Hence, these two lines of text appear to be red on the web page displayed on the screen.

In the second rule, the color of the text is set to blue. Also, in the second rule, the selector is every **<h3>** element which is a general sibling of the ** ** element. This condition is satisfied by three **<h3>**

elements, one in the LOC 15, the other in the LOC 19, and another in the LOC 20, in the file **`p50.html`**. Notice that the **`<h3>`** element in the LOC 15 is a general sibling of the **`<div>`** element situated in LOCs 10-12. The two **`<h3>`** elements in the LOCs 19 and 20 are general siblings of the **`<div>`** element situated in LOCs 16-18. Hence, these three lines of text appear to be blue on the web page displayed on the screen.

All the remaining lines of text appear in the default black color:

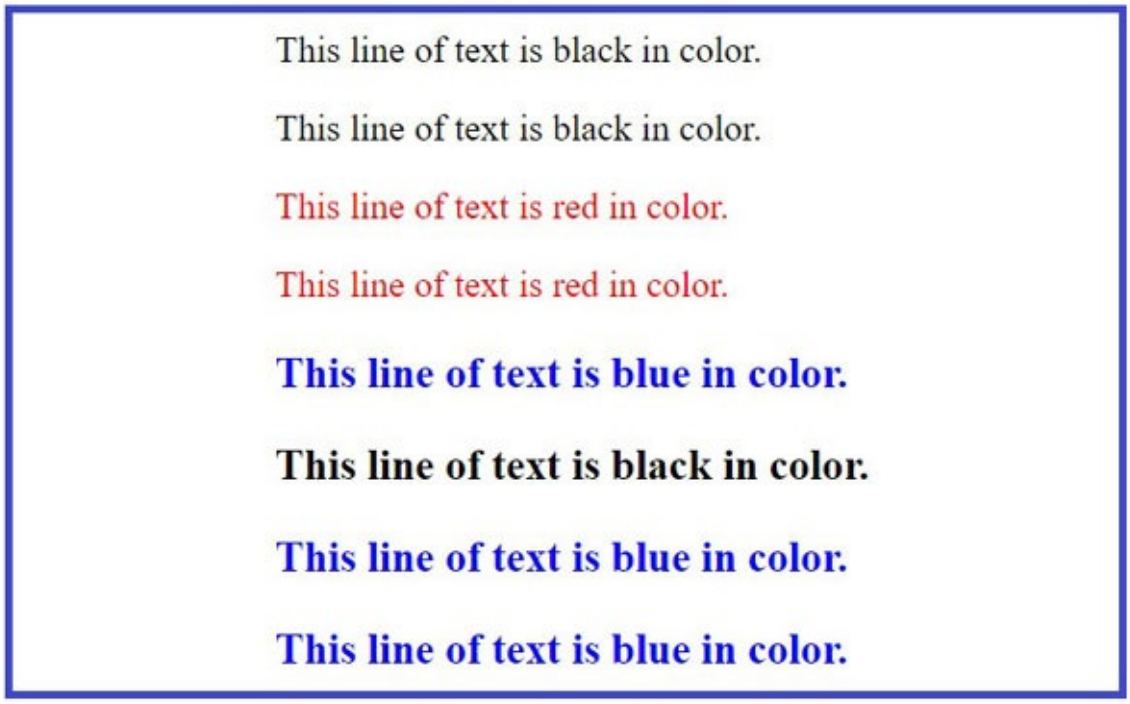

Figure 5.9: *Demonstration of a general sibling selector. See the program p50.html.*

Conclusion

In this chapter, we learned to use internal and external style sheets, **`universal`** selector, **`type`** selectors, **`id`** selectors, and **`class`** selectors to format the text using style sheets. Finally, we learned how to use various CSS combinators, namely, child selectors, descendant selectors, adjacent sibling selectors, and general sibling selectors. In the next chapter, we will learn to use the box model to apply styles to lists, tables, links, outlines, positioning, and multi-column layouts.

Further readings/references

- *HTML & CSS: The Complete Reference*, by Thomas A. Powell, 5/e, McGraw Hill, 2010.
- *Beginning HTML and CSS*, by Rob Larsen, Wrox, John Wiley and Sons, 2013.
- *HTML5: The Missing Manual*, by Matthew MacDonald, 2/e, 2014.
- https://www.internetingishard.com/

CHAPTER 6

Getting Command on CSS

Introduction

In this chapter, you will learn to use the box model, use the lists, apply styles to lists, apply styles to tables, create tables with single borders, create a hoverable table using the pseudo-class :hover, create striped tables, apply styles to links, apply styles to outlines, apply styles to elements in order to float them horizontally, and finally to apply styles to elements for positioning them on a webpage.

Structure

In this chapter, we will discuss the following topics:

- Using the box model
- Using the lists
- Styles for lists
- Styles for tables
- Table with single borders
- Hoverable table and pseudo-class :hover
- Striped table
- Styles for links
- Styles for outlines
- Styles for floating
- Styles for positioning

Objectives

After reading this chapter, you will be able to learn, how to use the box model, use the lists, apply styles to lists and apply styles to tables. You will also understand how to create tables with single borders, a hoverable table by using pseudo-class :hover**.** You will learn how to create a striped table, apply styles to links, apply styles to outlines, apply styles to elements in order to float them horizontally. Finally, you will learn how to apply styles to elements for positioning them on a webpage.

Using the box model

In HTML, every element is a box. Earlier, we talked about two types of elements, namely, block elements and inline elements. You can also refer to these items as block boxes and inline boxes. In CSS, there are a

number of properties of the box that you can set as per your requirement, namely, width, padding, border, margin, background-color, and so on. Let us write a simple program that illustrates this concept of a box in HTML.

Open the file **`template4.html`** and save it with the name **`p55.html`** in the folder **`myFiles`**. Place the keyboard cursor in LOC 06 and change the name of the CSS file as follows (notice that the name of the CSS file is changed from **`pXX.css`** to **`p55.css`**):

```
06.     <link rel="stylesheet" type="text/css" href="p55.css" media="screen" />
```

Code Snippet: 6.1

Place the keyboard cursor in the **`body`** element and insert a few LOCs in it as follows:

```
08.   <body>
09.     <div>
10.       <p>This is paragraph 1. This is demonstration of box model.</p>
11.       <p>This is paragraph 2. This is demonstration of box model.</p>
12.       <p>This is paragraph 3. This is demonstration of box model.</p>
13.     </div>
14.   </body>
```

Code Snippet: 6.2

Now, let us create the CSS file **`p55.css`**. Open Notepad (or your favourite editor) and type the following text in it:

```
01.  * {
02.    margin: 0;
03.    padding: 0;
04.    box-sizing: border-box;
05.  }
06.  div{
07.   background-color: yellow;
08.   width: 500px;
09.   padding: 25px;
10.   border: 10px solid blue;
11.   margin: 50px;
12.  }
13.  p{
14.   background-color: cyan;
15.  }
```

Code Snippet: 6.3

Save this file with the filename **`p55.css`** in the folder **`MyFiles`**. Load the file **`p55.html`** in a web browser and your screen will display the three lines of text inside a beautiful box as shown in *Figure 6.1*:

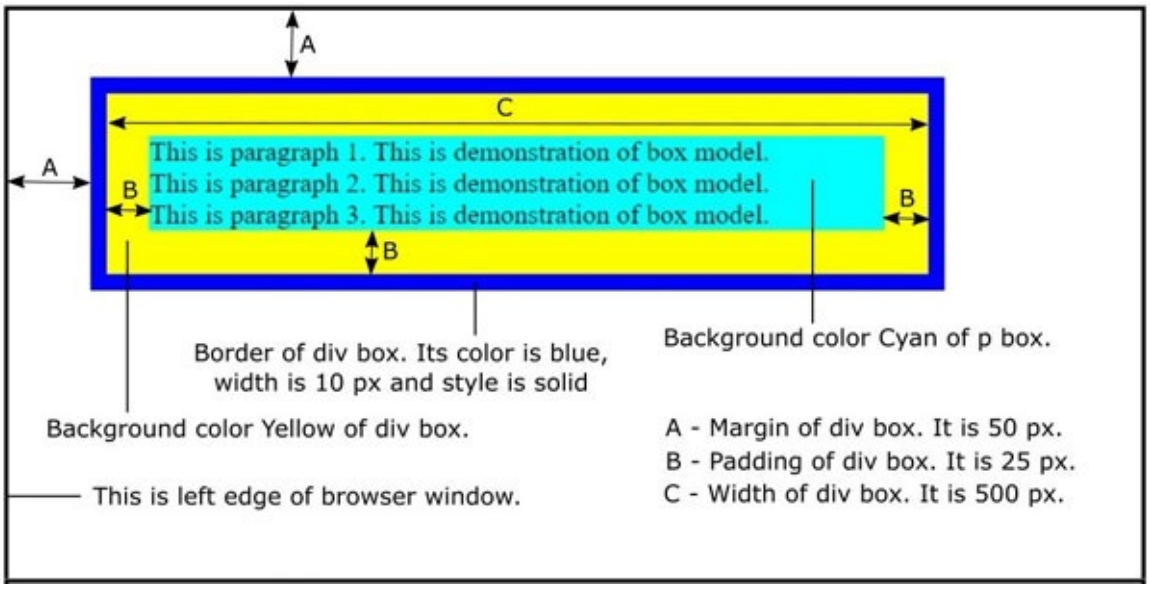

Figure 6.1: *The webpage p55.html is displayed on the screen. It illustrates the box model in CSS. Notice that div-box is set with the following properties: margin, width, padding, border, and background-color. The group of three lines, with the background-color Cyan, is actually the content of this box.*

Observe how the various properties set in code actually work. The margin of the box is always with reference to its parent container. Here, the parent container of the div-box is a browser window. Notice how the margin (shown by letter A) is set with reference to the browser window. Just for the sake of information also look that the parent container of the p-box (in this program) is the div-box. However, here, we have set the properties of the div-box only. Only one property of the p-box is set here: the background-color. The group of three lines is actually the content of this box.

Also, this figure shows how the border, width, and padding of the div-box are set.

Notice the LOCs 09 to 13 in the file **`p55.html`**. There is one div-box and three p-boxes. The div-box is the parent container of three p-boxes. Also, the parent container of the div-box is a browser window itself. Notice how the border property is set in LOC 10 and three values are assigned to this property. This first value 10px is the width of the border, the second value solid is the style of the border (there are other border styles available such as dotted, dashed, double, groove, ridge, inset, and outset), and the third value is nothing but the color of the border, which is blue here.

Now take a look at the file **`p55.css`**, Notice the LOCs 01 to 05. In this piece of code, we have set the margin and padding of everything to 0. Also, the **`box-sizing`** property is set to the value **`border-box`**. What is the purpose of this piece of code?

Note: Every web browser has default margin and padding settings. This results in, your web page appearing differently in different web browsers. The purpose of this piece of code is to remove the default margin and padding set by different web browsers. If you use this piece of code at the beginning of the CSS file, then your webpage appears the same in all web browsers.

In LOCs 06 to 12, the div-box properties are set. In LOCs 13 to 15, a single property of the p-box is set.

Using the lists

A list is a very useful feature in HTML. Using HTML, we can create the beautiful lists as shown in *Figure 6.2*. Let us do so.

Firstly, let us create the file **p56.html**. Open Notepad and type the following text in it:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>Using Lists</title>
  </head>
  <body>
    <p>This is unordered list of names of programmers</p>
    <ul>
      <li>Lina</li>
      <li>Mina</li>
      <li>Bina</li>
      <li>Tina</li>
    </ul>
    <p>This is ordered list of names of programmers</p>
    <ol>
      <li>Lina</li>
      <li>Mina</li>
      <li>Bina</li>
      <li>Tina</li>
    </ol>
  </body>
</html>
```

Code Snippet: 6.4

Save this file with the name **p56.html** in the folder **myFiles**.

Load this HTML document (**p56.html**) in a web browser and your screen will display the two beautiful lists as shown in *Figure 6.2*:

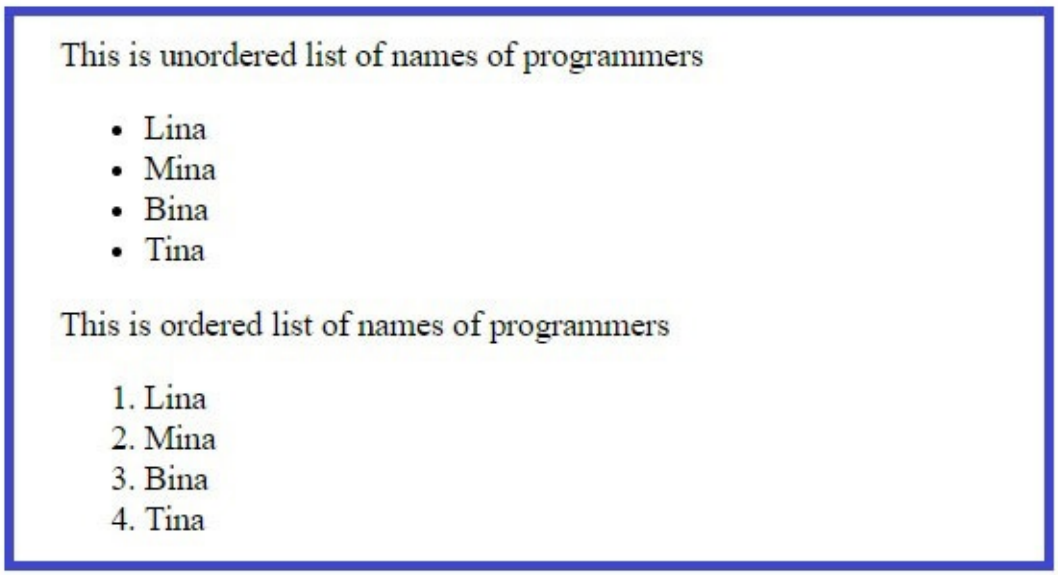

Figure 6.2: *Two beautiful lists are created on this webpage using HTML code. The upper list is called as unordered list and the lower*

list is called as an ordered list.

There are two types of lists in HTML:

- Unordered list (upper list shown in *Figure 6.2*)
- Ordered list (lower list shown in *Figure 6.2*)

In an unordered list, the list items are not numbered but bulleted. In an ordered list, the list items are numbered.

Unordered list

In order to create an unordered list, we need two types of HTML elements:

- ul
- li

The element **`ul`** has the opening tag **`<ul>`** and closing tag **`</ul>`**. In order to create an unordered list, we need just one **`ul`** element.

The element li has the opening tag **`<li>`** and closing tag **`</li>`**. In order to create an unordered list, we need a number of **`li`** elements. Precisely, if the list consists of **`N`** list items, then we need **`N li`** elements to create this list.

The piece of code 09 to 14 creates the unordered list as shown in *Figure 6.2*. In this piece of code, LOCs 09 and 14 consist of the opening tag **`<ul>`** and closing tag **`</ul>`**, respectively. LOCs 10, 11, 12, and 13 each consist of an element li. The content of the element li is nothing but list item. In this list, there are four list items, namely, Lina, Mina, Bina, and Tina. This is how an unordered list is created.

Ordered list

In order to create an ordered list, we need two types of HTML elements:

- ol
- li

The element **`ol`** has the opening tag **`<ol>`** and closing tag **`</ol>`**. In order to create an unordered list, we need just one **`ol`** element.

The behaviour of the element **`li`** is the same in an unordered list, as well as, in the ordered list.

The piece of code 16 to 21 creates the ordered list as shown in *Figure 6.2*. In this piece of code, LOCs 16 and 21 consist of the opening tag **`<ol>`** and closing tag **`</ol>`**, respectively. LOCs 17, 18, 19, and 20 each consist of an element **`li`**. The content of the element **`li`** is nothing but a list item. In this list, there are four list items, namely, Lina, Mina, Bina, and Tina. This is how an ordered list is created.

Styles for lists

CSS offers a number of styles for lists. In this section, we will apply some styles to the lists created in the previous section using the facilities in CSS. These styled lists will look as shown in *Figure 6.3*.

Open the file template4.html and save it with the name **p57.html** in the folder **myFiles**. Place the keyboard cursor in the LOC 06 and change the name of the CSS file as follows (notice that the name of the CSS file is changed from **pXX.css** to **p57.css**):

```
06.     <link rel="stylesheet" type="text/css" href="p57.css" media="screen" />
```

Code Snippet: 6.5

Place the keyboard cursor in the **body** element and insert a few LOCs in it as follows:

```
08.   <body>
09.     <p>These are unordered lists of names of programmers</p>
10.     <ul class="one">
11.       <li>Lina</li>
12.       <li>Mina</li>
13.       <li>Bina</li>
14.       <li>Tina</li>
15.     </ul>
16.     <ul class="two">
17.       <li>Lina</li>
18.       <li>Mina</li>
19.       <li>Bina</li>
20.       <li>Tina</li>
21.     </ul>
22.     <p>These are ordered lists of names of programmers</p>
23.     <ol class="three">
24.       <li>Lina</li>
25.       <li>Mina</li>
26.       <li>Bina</li>
27.       <li>Tina</li>
28.     </ol>
29.     <ol class="four">
30.       <li>Lina</li>
31.       <li>Mina</li>
32.       <li>Bina</li>
33.       <li>Tina</li>
34.     </ol>
35.   </body>
```

Code Snippet: 6.6

Now, let us create the CSS file **p57.css**. Open Notepad (or your favourite editor) and type the following text in it:

```
01. ul.one {
02.   list-style-type: circle;
03. }
04. ul.two {
05.   list-style-type: square;
06. }
07. ol.three {
08.   list-style-type: lower-roman;
09. }
10. ol.four {
11.   list-style-type: lower-alpha;
12. }
```

Code Snippet: 6.7

Save this file with the filename **`p57.css`** in the folder **`MyFiles`**. Load the file **`p57.html`** in a web browser and your screen will display the beautifully styled lists as shown in *Figure 6.3*:

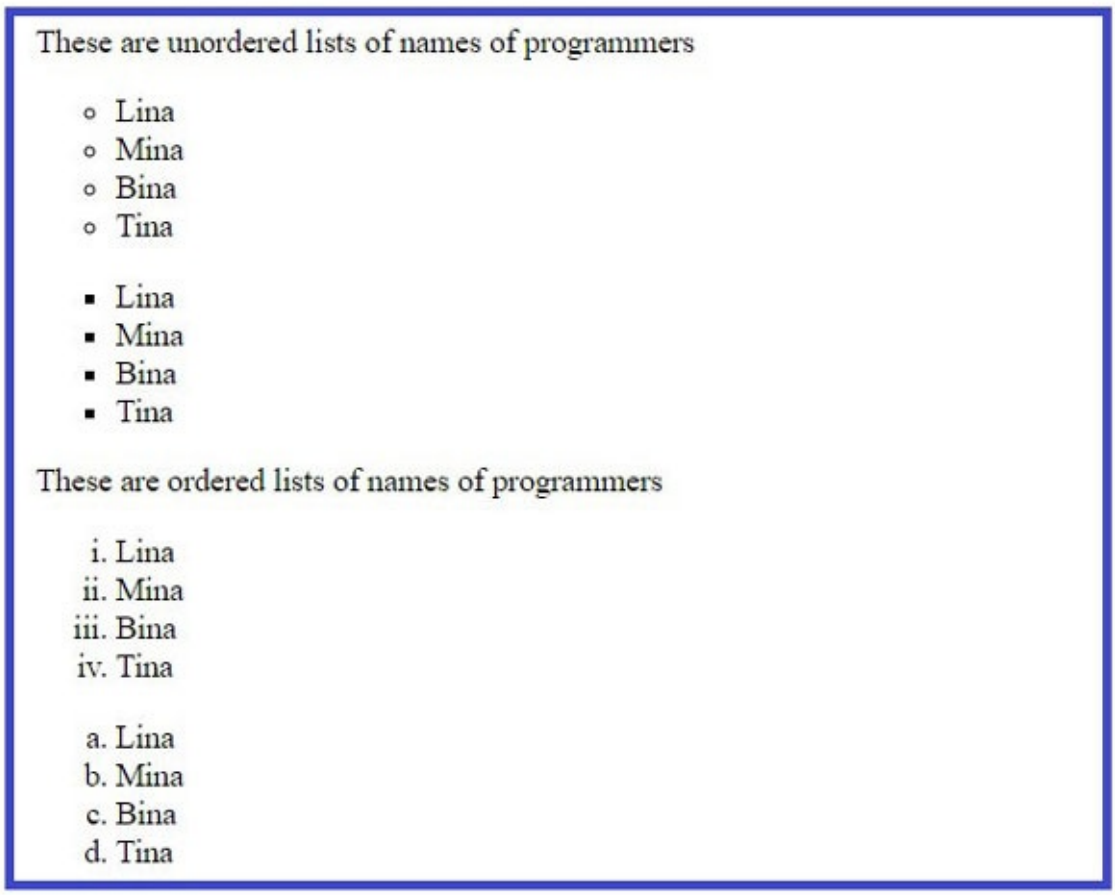

Figure 6.3: *Lists are styles using CSS. These styles are (from top to bottom) circle, square, lower-roman, and lower-alpha.*

In this program (that is, **`p57.html`**), four lists are created; two lists are unordered and the remaining two lists are ordered. For the first list, its **`class`** attribute is set with the value **`one`**. For the second list, its **`class`** attribute is set with the value **`two`**. For the third list, its **`class`** attribute is set with the value **`three`**. For the fourth list, its **`class`** attribute is set with the value **`four`**.

Now, take a look at the CSS file **`p57.css`**. In this program:

In the LOCs 01 to 03, for the selector **`ul.one`**, the property **`list-style-type`** is set with the value **`circle`**. It means the unordered list (**`ul`**) whose class name is **`one`** should be styled using **`circles`** instead of bullets. The first list is styled according to this piece of code.

In the LOCs 04 to 06, for the selector **`ul.two`**, the property **`list-style-type`** is set with the value **`circle`**. It means the unordered list (**`ul`**) whose class name is **`two`** should be styled using **`squares`** instead of bullets. The second list is styled according to this piece of code.

In the LOCs 07 to 09, for the selector **`ol.three`**, the property **`list-style-type`** is set with the value **`circle`**. It means the ordered list (**`ol`**) whose class name is **`three`** should be styled using lower-roman letters (that is, i, ii, iii, and so on) instead of numbers 1, 2, 3, and so on. The third list is styled according to this piece of code.

In the LOCs 10 to 12, for the selector **`ol.four`**, the property **`list-style-type`** is set with the value **`circle`**. It means the ordered list (**`ol`**) whose class name is **`four`** should be styled using lower-alpha letters (that is, a, b, c, and so on) instead of numbers 1,2 3, and so on. The fourth list is styled according to this piece of code.

Styles for tables

You have already created tables in *Chapter 4, Using Tables and Forms*. In this section, you will apply styles to create beautiful and stylish tables using the facilities in CSS. Notice the table shown in *Figure 6.1*. It's a nice one. However, in this table, all the lines are double. Actually, a table consists of a single line. The double lines occurred because every cell is bordered; as a result, we get double lines in a table. Now, we will create a table with single lines (single borders) as shown in *Figure 6.4*.

Table with single borders

Open the file **`template4.html`** and save it with the name **`p58.html`** in the folder **`myFiles`**. Place the keyboard cursor in LOC 06 and change the name of the CSS file as follows (notice that the name of the CSS file is changed from **`pXX.css`** to **`p58.css`**):

```
06.     <link rel="stylesheet" type="text/css" href="p58.css" media="screen" />
```

Code Snippet: 6.8

Place the keyboard cursor in the **`body`** element and insert a few LOCs in it as follows:

```
<body>
 <table>
  <caption>Examination Score</caption>
  <tr>
   <th>Subjects</th>
   <th>Lina</th>
   <th>Mina</th>
   <th>Bina</th>
  </tr>
  <tr>
   <td>Physics</td>
   <td>62</td>
   <td>68</td>
   <td>74</td>
  </tr>
  <tr>
   <td>Chemistry</td>
   <td>82</td>
   <td>94</td>
   <td>92</td>
  </tr>
 </table>
</body>
```

Code Snippet: 6.9

Now, let us create the CSS file **`p58.css`**. Open Notepad (or your favourite editor) and type the following text in it:

```
table, th, td{
 border: 3px solid blue;
}
table{
 width: 40%;
 border-collapse: collapse;
}
th, tr{
 height: 30px;
}
td{
 text-align: center;
}
```

Code Snippet: 6.10

Save this file with the filename **`p58.css`** in the folder **`MyFiles`**. Load the file **`p58.html`** in a web browser and your screen will display the beautifully styled table as shown in *Figure 6.4:*

Examination Score

Subjects	Lina	Mina	Bina
Physics	62	68	74
Chemistry	82	94	92

Figure 6.4: *A stylish table. Notice the single borders. Also, contents of each are aligned at center, horizontally, as well as, vertically.*

Take a look at the piece of code in LOCs 04 to 07 (file **`p58.css`**). This is a CSS rule for the selector **`table`**. Notice the LOC 6 in this piece of code in which a property **`border-collapse`** is set with the value **`collapse`**. This property set in LOC 6 is responsible for replacing the double lines in the table (see *Figure 4.1*) with single lines (see *Figure 6.1*).

Notice the piece of code in the LOCs 08 to 10 (file **`p58.css`**). This is a CSS rule in which selectors are **`th`** and **`tr`**. Here, **`th`** means **`table heading`** and **`tr`** means **`table row`**. The topmost row in *Figure 6.4* is the table heading and the remaining two rows in *Figure 6.4* are "table rows." In this piece of code, the height of the table heading and table rows are set with the value 30 px.

Observe the piece of code in LOCs 11 to 13 (file **`p58.css`**). This is a CSS rule in which the selector is **`td`** (which stands for **`table data`**). Table data refers to all data in the table except in the topmost row (which is nothing but a heading). In this CSS rule, we have horizontally aligned the contents of cells the center. This is done by setting the property **`text-align`** to **`center`**. As regards, vertical alignment, all the data in the table is vertically aligned to the center by default. If we delete the LOC 12 in the file **`p58.css`**, then the data in the second and third rows will get aligned to the left which is the default choice.

Hoverable table and pseudo-class :hover

The hoverable table is a very useful feature. When you place the mouse cursor on any cell in the hoverable table, then that complete row changes its color (see *Figure 6.5*). This feature is particularly useful when you are browsing a large table with a good number of data items. Let us create a hoverable table.

Before proceeding further, you should learn something about pseudo-classes because in our next program, we will use the pseudo-class **`hover`**. What is this pseudo-class?

Note: Pseudo-class is a facility in CSS that allows us to define a special state of an element. For example, the pseudo-class hover allows us to define a state of an element in which the mouse cursor is placed on that element.

Some other pseudo-classes are **`active`**, **`visited`**, **`link`**, and so on. These pseudo-classes are particularly used with links.

Now come back to the pseudo-class **`hover`**. Notice the piece of code (to be placed in the CSS file) as follows:

```
p:hover{
color: yellow;
}
```

The effect of this piece of code is that when you hover the mouse cursor (that is, place the mouse cursor)

on any **p** element on this webpage, then the text in that paragraph looks yellow.

Now, let us develop the program **p59.html** that demonstrates a hoverable table.

Open the file **template4.html** and save it with the name **p59.html** in the folder **myFiles**. Place the keyboard cursor in LOC 06 and change the name of the CSS file as follows (notice that the name of the CSS file is changed from pXX.css to **p59.css**):

```
06.     <link rel="stylesheet" type="text/css" href="p59.css" media="screen" />
```

Code Snippet: 6.11

Place the keyboard cursor in the **body** element and insert a few LOCs in it as follows:

```
08. <body>
09.  <table>
10.   <caption>Examination Score</caption>
11.   <tr>
12.    <th>Subjects</th>
13.    <th>Lina</th>
14.    <th>Mina</th>
15.    <th>Bina</th>
16.   </tr>
17.   <tr>
18.    <td>Physics</td>
19.    <td>62</td>
20.    <td>68</td>
21.    <td>74</td>
22.   </tr>
23.   <tr>
24.    <td>Chemistry</td>
25.    <td>82</td>
26.    <td>94</td>
27.    <td>92</td>
28.   </tr>
29.  </table>
30. </body>
```

Code Snippet: 6.12

Now, let us create the CSS file **p59.css**. Open Notepad (or your favourite editor) and type the following text in it:

```
01. table{
02.  width: 30%;
03.  text-align: center;
04. }
05. th, td{
06.  border-bottom: 2px solid red;
07. }
08. tr:hover{
09.  color: white;
10.  background-color: orange;
11. }
```

Code Snippet: 6.13

Save this file with the filename **`p59.css`** in the folder **`MyFiles`**. Load the file **`p59.html`** in a web browser and your screen will display the hoverable table as shown in *Figure 6.5*:

Examination Score

Subjects	Lina	Mina	Bina
Physics	62	68	74
Chemistry	82	94	92

Figure 6.5: *Hoverable table. Last row in this table has changed its color because the mouse cursor is placed somewhere on the last row. The mouse cursor is not shown in the figure. Also, notice that vertical lines are missing in this table.*

Notice that vertical lines are missing in this table and this is because of the piece of code in LOCs 05 to 07 (file **`p59.css`**). In LOC 06, the property **`border-bottom`** is set with a suitable value, as a result, only the bottom border is shown in the table and other borders are removed.

The piece of code in the LOCs 08 to 11 is responsible for the behavior of this table as a hoverable table. This is a CSS rule in which the selector is **`tr:hover`**. Here, **`tr`** means a table row and **`hover`** is a pseudo class. We have already discussed the meaning of pseudo class **`hover`**. In LOCs 09 and 10, the properties color and background-color is set with suitable values. When we hover the mouse cursor on any row in this table, then the background-color of this row changes in accordance with the property settings in LOCs 09 and 10. This is how the hoverable table works.

Striped table

In a striped table, the rows are colored alternately with the desired color. I call it a zebra table because like a zebra, this table has stripes (see *Figure 6.6*). In order to create a striped table, we need a very special type of selector as follows:

```
:nth-child(even)
```

This selector picks up every even numbered row in a table and changes its colors accordingly. As you can see in *Figure 6.6*, the second and fourth rows are colored. Here, the heading is counted as row 1. If you replace even by odd in this selector, then the first and third rows will get colored. Now, let us write a program to create this table.

Open the file **`template4.html`** and save it with the name **`p60.html`** in the folder **`myFiles`**. Place the keyboard cursor in LOC 06 and change the name of the CSS file as follows (notice that the name of the CSS file is changed from **`pXX.css`** to **`p60.css`**):

```
06.     <link rel="stylesheet" type="text/css" href="p60.css" media="screen" />
```

Code Snippet: 6.14

Place the keyboard cursor in the **`body`** element and insert a few LOCs in it as follows:

```
<body>
 <table>
  <caption>Examination Score</caption>
  <tr>
   <th>Subjects</th>
   <th>Lina</th>
   <th>Mina</th>
   <th>Bina</th>
  </tr>
  <tr>
   <td>Physics</td>
   <td>62</td>
   <td>68</td>
   <td>74</td>
  </tr>
  <tr>
   <td>Chemistry</td>
   <td>82</td>
   <td>94</td>
   <td>92</td>
  </tr>
  <tr>
   <td>Maths</td>
   <td>74</td>
   <td>78</td>
   <td>82</td>
  </tr>
 </table>
</body>
```

Code Snippet: 6.15

Now, let us create the CSS file **`p60.css`**. Open Notepad (or your favourite editor) and type the following text in it:

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

Code Snippet: 6.16

Save this file with the filename **`p60.css`** in the folder **`MyFiles`**. Load the file **`p60.html`** in a web browser and your screen will display the hoverable table as shown in *Figure 6.6*:

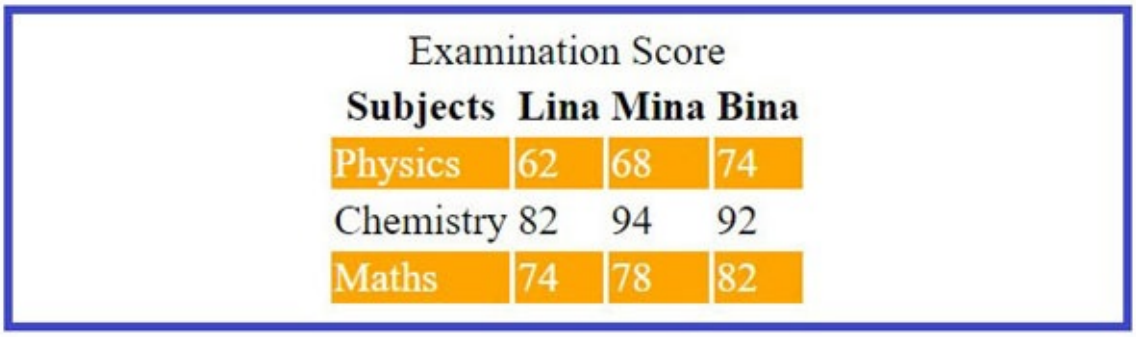

Figure 6.6: *Striped table. It has stripes like a zebra; hence I call it a zebra table. In this table, even numbered rows are painted with different colors. Heading is considered as the first row, and so on.*

Now, let us analyze the file p60.css. It consists of a CSS rule in the LOCs 01 to 04. In this CSS rule, the selector is **`tr:nth-child(even)`**. The properties color and background-color are set in LOCs 02 and 03. As explained earlier, the selector picks only even numbered rows in the table and changes their color and background-color to white and orange, respectively.

Styles for links

While browsing the webpages you must have noticed that generally, links are blue. However, when you click on any link, its color changes. Also, when you hover the mouse cursor over a link, then temporarily the color of the link changes. This is done using styles for links. While using styles for links, we generally use the pseudo-classes link, **`visited`**, **`hover`**, and **`active`** in collaboration with the element **`a`** as follows:

```
a:link
a:visited
a:hover
a:active
```

The selector **`a:link`** is used to set the color of normal, unvisited link. However, instead of the selector **`a:link`**, you can use the selector **`a`** only and you still get the same effect.

The selector **`a:visited`** is used to set the color of a visited link. Once you click on the link, its color is changed. If you reload the same webpage again, then the original color of the link is not restored because the web browser remembers that you have already clicked on this link. How does a web browser remember this thing? It simply notes it in the cookie.

The selector **`a:hover`** is used to set the color of the link while you hover the mouse-cursor over it.

The selector **`a:active`** is used to set the color of the link while the link is clicked. This change in color is only momentary. When you release the mouse button, the link's earlier color is restored. This selector is not of much use; therefore, we will not use it in our next program. We will use the other three selectors, namely, **`a:link`**, **`a:visited`**, and **`a:hover`** in our next program that demonstrates the links with styles.

Firstly, let us create the file **`p61.html`**. Open Notepad (or your favorite editor) and type the following text in it:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>Using Cascading Style Sheets</title>
    <link rel="stylesheet" type="text/css" href="p61.css" media="screen" />
  </head>
  <body>
    <h3>This is First Page</h3>
    <a href="p61b.html">Go to Second Page</a>
  </body>
</html>
```

Code Snippet: 6.17

Save this file with the filename **`p61.html`** in the folder **`MyFiles`**.

Now, let us create the file **`p61b.html`**. Open Notepad (or your favorite editor) and type the following text in it:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>Using Cascading Style Sheets</title>
    <link rel="stylesheet" type="text/css" href="p61.css" media="screen" />
  </head>
  <body>
    <h3>This is Second Page</h3>
    <a href="p61.html">Go to First Page</a>
  </body>
</html>
```

Code Snippet: 6.18

Save this file with the filename **`p61b.html`** in the folder **`MyFiles`**.

Now, let us create the file **`p61.css`**. Open Notepad (or your favorite editor) and type the following text in it:

```
a:link{
  color: blue;
}
a:visited{
  color:red;
}
a:hover{
  color: yellow;
  background-color: blue;
}
```

Code Snippet: 6.19

Save this file with the filename **`p61.css`** in the folder **`MyFiles`**. Load the file **`p61.html`** in a web browser and your screen will display the webpage as shown in *Figure 6.7 (a):*

Figure 6.7: *(a) The link is blue as it is not clicked yet. Once you click on it, it will turn red. As you click on this link, you will be taken to the second page as shown in the part (b) of this figure. (b) If you click on this link, then you will be taken to the first page again. (c) If you hover the mouse-cursor over any link, then its color and background-color will change as shown in this figure.*

Notice that the link **`Go to Second Page`** is blue in color because it is not clicked yet. Once you click on it, it turns red. Click on the link **`Go to Second Page`** and the second page appears on the screen as shown in *Figure 6.7 (b)*. Hover the mouse-cursor over the link **`Go to First Page`** and then the color and background-color of this link change as shown in *Figure 6.7 (c)*. Click on the link **`Go to First Page`** and you will be taken back to the first page. Now, the link **`Go to Second Page`** appears to be red as it is already clicked.

Styles for outlines

We can offer styles to an outline as shown in *Figure 6.8*. Notice the piece of code given as follows:

```
p {
  outline: magenta solid 8px;
}
```

This piece of code, which consists of a CSS rule, styles the outline. In this piece of code, the selector is **`p`**. Only one property is set in this piece of code and this property is nothing but the outline. The triple-value assigned to this property is nothing but magenta solid 8px. The first term in this value is the color of the outline (here, it is magenta). The second term in this value is the style of the outline (here, it is solid). The third term in this value is the thickness of this outline (here, it is 8 px).

Now, let us write a suitable program to demonstrate the styling for the outline. Open the file template4.html and save it with the name **`p62.html`** in the folder **`myFiles`**. Place the keyboard cursor in LOC 06 and change the name of the CSS file as follows (notice that the name of the CSS file is changed from **`pXX.css`** to **`p62.css`**):

```
06.     <link rel="stylesheet" type="text/css" href="p62.css" media="screen" />
```

Code Snippet: 6.20

Place the keyboard cursor in the **`body`** element and insert a few LOCs in it as follows:

```
08.   <body>
09.     <h2>Styling the Outline</h2>
10.     <p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit.
11.        Aenean commodo ligula eget dolor. Aenean massa. Cum sociis
12.        natoque penatibus et magnis dis parturient montes, nascetur
13.        ridiculus mus.
14.     </p>
15.   </body>
```

Code Snippet: 6.21

Now, let us create the CSS file **p62.css**. Open Notepad (or your favorite editor) and type the following text in it:

```
01. p {
02.   border: 6px dashed orange;
03.   outline: magenta solid 8px;
04.   margin: auto;
05.   padding: 10px;
06.   text-align: justify;
07. }
```

Code Snippet: 6.22

Save this file with the filename **p62.css** in the folder **MyFiles**. Load the file **p62.html** in a web browser and your screen will display the webpage with styled outline as shown in *Figure 6.8*:

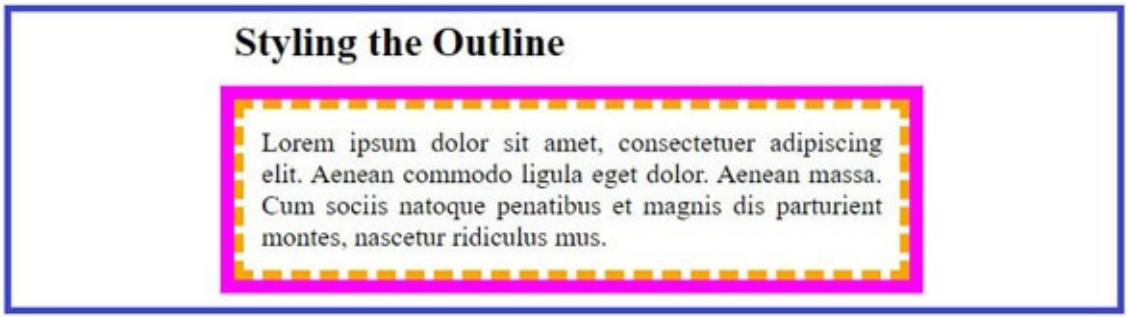

Figure 6.8: *Styled outline. The orange colored dashed line is border. The magenta colored solid line is outline. Notice that outline encloses the border.*

We have already discussed the working of the CSS code. Notice the LOC 03 in the file **p62.css**. In this LOC, the property **outline** is set with the value **magenta solid 8px**. As a result, the paragraph on this webpage is equipped with an outline whose color is magenta, whose style is solid and width is 8 px, as shown in *Figure 6.8*.

Styles for floating

In the section “Using the Box Model”, you learned about boxes. Particularly, you learned that in HTML, every element is a box. Block elements and inline elements are actually block boxes and inline boxes. In the HTML file, we introduce the number of boxes, one by one; and all these boxes are listed on a web page, one by one, from top to bottom. In the technical language, we describe this situation as a vertical flow and this is the default style of flow.

But sometimes, we need to alter the style of the flow from vertical to horizontal. For example, let’s take a look at *Figure 6.9*. In this figure, you can see that boxes Block1 and Block2 are at the same horizontal level because their flow is horizontal. Also, the boxes Block3, Block4, and Block5 are at the same horizontal level because their flow is horizontal. We need this type of flow for a multicolumn layout for the webpage.

The two important properties which we use to alter the floating style are float and clear. The default value of float is none. With this default value, all the blocks experience vertical flow. When we want the blocks to experience the horizontal flow, then we need to set the property **float** of those blocks with the value **left** or **right**. When the property **float** is set with the value **left**, then all these boxes experience

horizontal flow with left alignment. When the property **float** is set with the value **right**, then all these boxes experience horizontal flow with the right alignment.

When we set the property **float** of a box (say **box1**) with some suitable value, then the boxes that appear after **box1** inherits this **float** property.

When the horizontal flow is no more needed for a box (say, **box4**), then simply set the **clear** property of that box (that is, of **box4**) with the value **both**.

For example, suppose your HTML file provides four boxes as follows (notice that the class **box1** is common to **Box1**, **Box2**, and **Box3**):

```
<div class="box1">Box1</div>
<div class="box1">Box2</div>
<div class="box1">Box3</div>
<div class="box4">Box4</div>
```

Normally, on the screen, these boxes would appear according to vertical flow as follows:

```
Box1
Box2
Box3
Box4
```

But suppose if you want to put **Box1**, **Box2**, and **Box3** in a single line, then set their **float** property with the value **left**. As you do not want the **Box4** to join this trio, set the **clear** property of **Box4** with the value **both** as follows:

```
.box1 {
float: left;
}
.box4{
clear: both;
}
```

When you display this webpage on the screen, you will find these boxes placed as follows:

```
Box1 Box2 Box3
Box4
```

What if these three boxes **Box1**, **Box2**, and **Box3** are large enough and hence cannot be accommodated in a single line? In this case, the third box will be forced to appear on the next line as follows:

```
Box1 Box2
Box3
Box4
```

If you set the **float** property of the class **box1** to **right** as follows:

```
float: right;
```

Then the webpage would display these boxes as follows:

```
Box1 Box2 Box3
Box4
```

Observe that now **Box1**, **Box2**, and **Box3** are right aligned.

Now, we will build the webpage as shown in *Figure 6.9*. On this webpage, there are eight boxes as follows (in fact, there is one more box which is "page" and it is the parent of all these eight boxes, but to keep the things simple, we have ignored it):

```
Menu
Block1 Block2
Middle_Section
Block3 Block 4 Block5
Footer
```

Notice that boxes **Block1** and **Block2** in the second line are floated horizontally. Also, the boxes **Block3**, **Block4**, and **Block5** in the fourth line are also floated horizontally. Therefore, for these boxes, we need to set their property **float** with the value **left** or **right**. Also, we need to set the property **clear** of the boxes **Middle_Section** and **Footer** with the value **both** so that they should not inherit the float property of the preceding boxes. As the box **Menu** is the first box on this webpage, it is all right with its default property.

In the actual code in the file **p63.css**, you will find the boxes **Block2** and **Block5** have their **float** property set with the value **right**. This is my personal choice, if you want, you can set their property **float** with the value **left** also. After all, the last box in each line (second line and fourth line) is expected to touch the right boundary of the page.

Before proceeding further, notice that the property **float** has the following possible values: none (this is the default value, it means no horizontal flow), left, right, and initial (it always means the default value) and inherit (it inherits this property from its parent).

Also, observe that property **clear** has the following possible values: none (default value), left (element is pushed below the left floated elements), right (element is pushed below the right floated elements), both (element is pushed below both the left and right floated elements), initial (it always means a default value), and inherit (it inherits this property from its parent element).

Now, let us build the webpage that displays on the screen the boxes as shown in *Figure 6.9*. Open the file **template4.html** and save it with the name **p63.html** in the folder **myFiles**. Place the keyboard cursor in LOC 06 and change the name of the CSS file as follows (notice that the name of the CSS file is changed from pXX.css to p63.css):

```
06.     <link rel="stylesheet" type="text/css" href="p63.css" media="screen" />
```

Code Snippet: 6.23

Place the keyboard cursor in the **body** element and insert a few LOCs in it as follows:

```
08.   <body>
09.     <div class='page'>
10.       <div class='menu'>Menu</div>
11.       <div class='block1'>Block1</div>
12.       <div class='block2'>Block2</div>
13.       <div class='middle'>Middle Section</div>
14.       <div class='block3'>Block3</div>
15.       <div class='block4'>Block4</div>
16.       <div class='block5'>Block5</div>
17.       <div class='footer'>Footer</div>
18.     </div>
19.   </body>
```

Code Snippet: 6.24

Now, let us create the CSS file **p63.css**. Open Notepad (or your favorite editor) and type the following text in it:

```css
* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
}
.page {
  width: 900px;
  margin: 0 auto;
  border: 3px solid red;
  text-align: center;
  font-size: 30px;
}
.menu {
  height: 50px;
  background-color: lightblue;
  line-height: 50px;
}
.block1 {
  float: left;
  width: 441px;
  height: 60px;
  background-color: lightgreen;
  border: 3px solid blue;
  line-height: 60px;
}
.block2 {
  float: right;
  width: 441px;
  height: 60px;
  background-color: lightgreen;
  border: 3px solid blue;
  line-height: 60px;
}
.middle {
  clear: both;
  height: 50px;
  background-color: lightblue;
  line-height: 50px;
}
.block3 {
  margin-right: 12px;
  float: left;
  width: 290px;
  height: 60px;
  background-color: lightgreen;
  border: 3px solid blue;
  line-height: 60px;
}
```

```
.block4 {
  float: left;
  width: 290px;
  height: 60px;
  background-color: lightgreen;
  border: 3px solid blue;
  line-height: 60px;
}
.block5 {
  float: right;
  width: 290px;
  height: 60px;
  background-color: lightgreen;
  border: 3px solid blue;
  line-height: 60px;
}
.footer {
  clear: both;
  height: 50px;
  background-color: yellow;
  line-height: 50px;
}
```

Code Snippet: 6.25

Save this file with the filename **`p63.css`** in the folder **`MyFiles`**. Load the file **`p63.html`** in a web browser and your screen will display the webpage as shown in *Figure 6.9*:

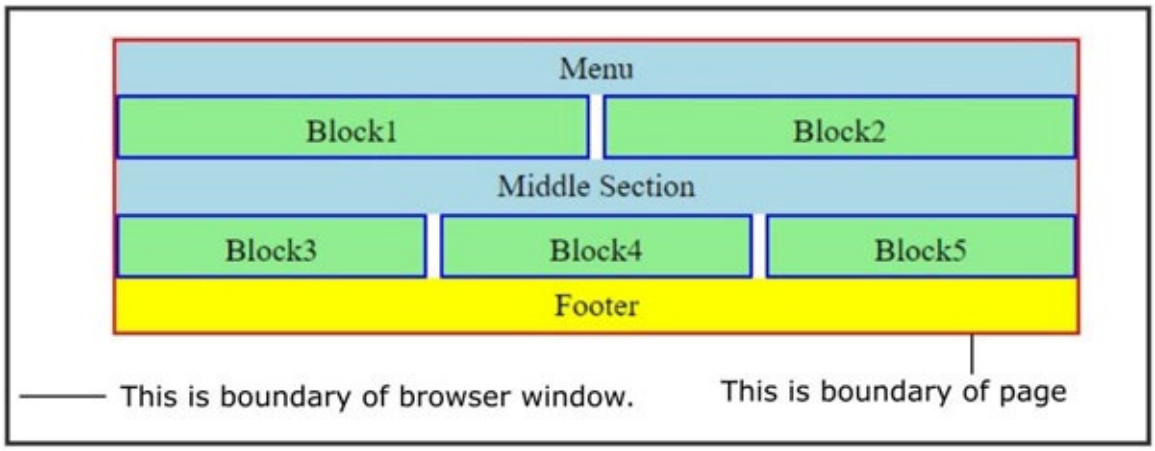

Figure 6.9: *Floating the boxes horizontally. Here, the boxes Block1 and Block2 in the second line are floated horizontally. Also, the boxes Block3, Block4, and Block5 in the fourth line are also floated horizontally. The page is smaller than a browser window. In the absence of the page, the boxes Menu, Middle Section, and Footer would have occupied the full width of the browser window.*

Observe the properties **`height`** and **`line-height`** in almost every class; the values assigned to these properties are the same. For example, in the class **`menu`**, the value assigned to these properties is 50 px; in the class **`block1`**, the value assigned to these properties is 60 px; and so on. The property **`height`** maintains the height of box. The property **`line-height`** ensures that any text typed in the box is vertically aligned at the center of the box, provided that value assigned to the properties **`height`** and **`line-height`** is the same.

What about the width of these boxes? Well, in order to set the width of these boxes you are required to do some calculations, backed by the trial and error method. For example, in the class **`block3`**, we have set the right margin to be 12 px and the width to be 290 px. You are required to fiddle with these values till you get the desired layout on the screen.

Finally, notice the following:

If you want a scalable layout, then replace all the fixed lengths (such as 900 px or 60 px, and so on) by the percentages (for example, 90% or 12%, and so on). In the scalable layout, when the size of the browser window changes, then all the boxes on the webpage are automatically resized accordingly. Here, percentage refers to the percentage width of the parent element. For example, if you set the "width" of "Page" to 70%, then the browser sets the width of "Page" to 70% of the width of the browser window because the browser window is the parent element of "Page." If you set the "width" of "Menu" to 90%, then the browser sets the width of "Menu" to 90% of the width of "Page" as "Page" is the parent element of "Menu." Here, the browser window is the parent element of "Page" and "Page" is the parent element of all the eight boxes, namely, "Menu," "Block1," "Block2," "Middle Section," "Block3," "Block4," "Block5," and "Footer."

Styles for positioning

We can position an element anywhere on the webpage using the property **`position`**. This property can take the following values: **`static`**, **`absolute`**, **`fixed`**, **`relative`**, **`initial`**, and **`inherit`**. The meaning of these values is described as follows:

- **`static`**: This is the default value. Elements with this value (for the **`position`** property) appear in the same order on the webpage as they appear in the HTML file.
- **`absolute`**: The element with this value (for the **`position`** property) is placed relative to its parent element.
- **`fixed`**: The element with this value (for the **`position`** property) is positioned relative to the browser window.
- **`relative`**: The element with this value (for the **`position`** property) is positioned relative to its normal position.
- **`initial`**: It always means a default value.
- **`inherit`**: It inherit this property from its parent.

Now, let us develop a simple program to demonstrate the capabilities of the **`position`** property. Open the file **`template4.html`** and save it with the name **`p64.html`** in the folder **`myFiles`**. Place the keyboard cursor in the LOC 06 and change the name of the CSS file as follows (notice that the name of the CSS file is changed from pXX.css to p64.css):

```
06.     <link rel="stylesheet" type="text/css" href="p64.css" media="screen" />
```

Code Snippet: 6.26

Place the keyboard cursor in the **`body`** element and insert a few LOCs in it as follows:

```
  <body>
    <div class='block1'>
      Block1
      <div class='block2'>Block2</div>
      <div class='block3'>Block3</div>
    </div>
    <div class='block4'>Block4</div>
  </body>
```

Code Snippet: 6.27

Now, let us create the CSS file **`p64.css`**. Open Notepad (or your favorite editor) and]type the following text in it:

```
* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
}
.block1 {
  position: fixed;
  top: 40px;
  left: 60px;
  width: 400px;
  height: 300px;
  background-color: yellow;
  border: 3px solid blue;
}
.block2 {
  position: absolute;
  top: 50px;
  left: 60px;
  width: 300px;
  height: 200px;
  background-color: snow;
  border: 3px solid blue;
}
.block3 {
  position: fixed;
  top: 150px;
  left: 170px;
  width: 200px;
  height: 100px;
  background-color: lemonchiffon;
  border: 3px solid blue;
}
.block4 {
  position: static;
  width: 200px;
  height: 150px;
  background-color: paleturquoise;
  border: 3px solid blue;
}
```

Code Snippet: 6.28

Save this file with the filename **`p64.css`** in the folder **`MyFiles`**. Load the file **`p64.html`** in a web browser and your screen will display the webpage with the styled outline as shown in *Figure 6.10*:

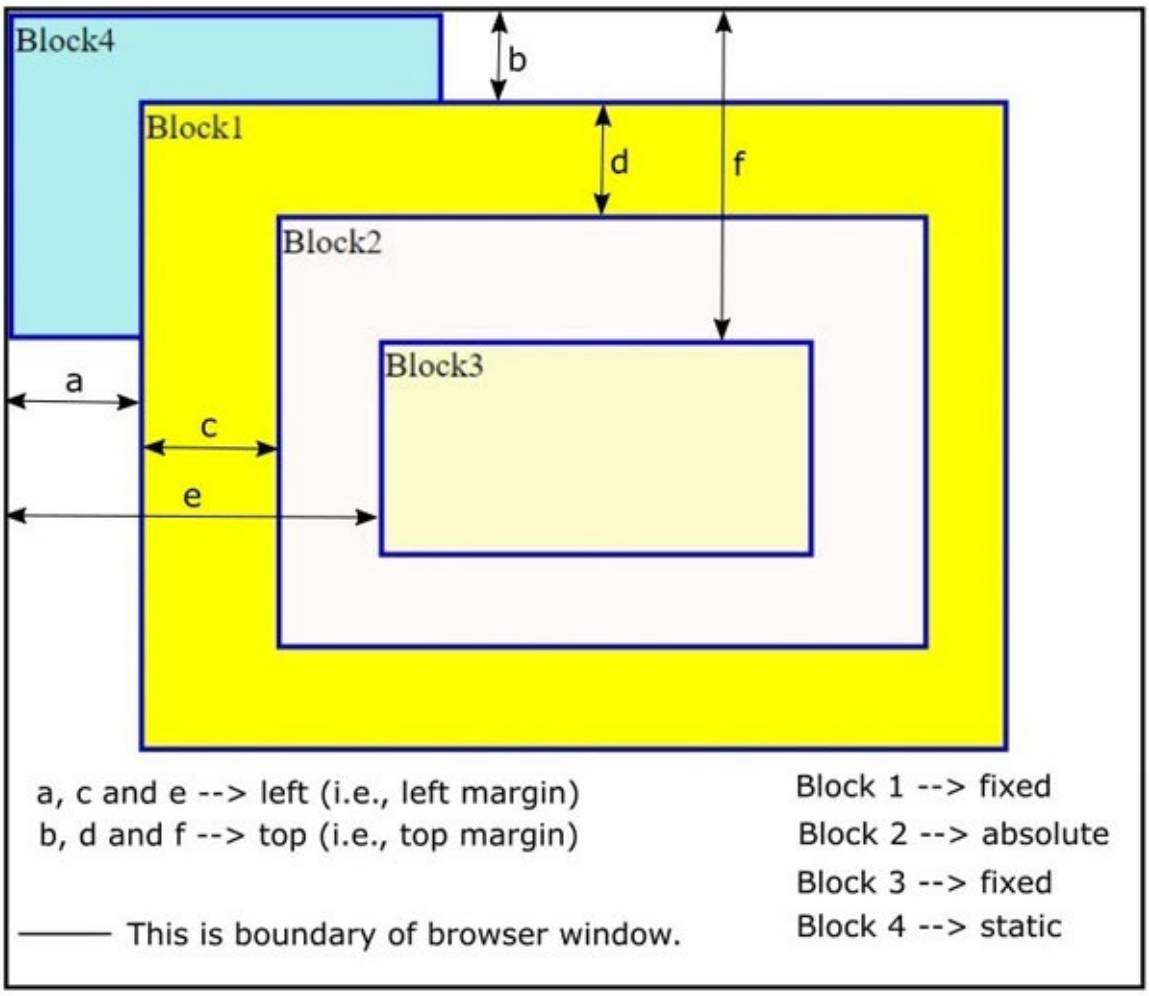

Figure 6.10: *Positioning the elements (boxes) on a webpage. The "static" is the default value of the "position" property. For "fixed" element (box), left and top margins are measured with reference to the browser window. For "absolute" element (box), left and top margins are measured with reference to its parent element. Here, Block1 is the parent element of Block2 (Block2 is absolute) hence Block2's left and top margins are measured with reference to Block1.*

Notice that Block 1 is fixed, Block 2 is absolute, Block 3 is fixed, and Block 4 is static. Also, Block 1 is the parent element of Block 2 and Block 3, Block 4 is placed at the top, left-hand corner of the browser window (that is, Block 4's top, left corner coincides with the top, left-hand corner of the browser window). As Block 4 is a static one, even if you assign suitable values to its **`left`** and **`top`** properties, it will not move from its position. Block1 and Block 3 are placed at the specified left and top margins from the browser window. You can see these margins (their values) in the LOCs 08,09, 26, and 27 in the file **`p64.css`**. Block 2 is placed at the specified left and top margins from Block 1. You can see these margins (their values) in LOCs 17 and 18 in the file **`p64.css`**.

In *Figure 6.10*, you can see that all the elements (or boxes) are overlapping one another. Block 3 is at the top (not obstructed by any other element). Block 2 is just below Block 3, Block 1 is just below Block 2 and finally, Block 4 is just below Block 1. Can we change this order? Can we bring, say, Block 1 to the top? Certainly, we can do so by setting the property **`z-index`** of overlapping elements. If there are three overlapping boxes, then set their **`z-index`** properties as follows:

- For box1: **`z-index = 1;`**

- For box2: **`z-index = 2;`**
- For box3: **`z-index = 3;`**

Now, the element (or box) with the largest **`z-index`** value (here, **`box3`**) will be at the top and the element (or box) with the smallest **`z-index`** value (here, **`box1`**) will be at the bottom. You can change this order by changing the **`z-index`** values of the concerned elements.

Let us experiment with this newly acquired knowledge by creating a simple program. Open the file **`template4.html`** and save it with the name **`p65.html`** in the folder **`myFiles`**. Place the keyboard cursor in the LOC 06 and change the name of the CSS file as follows (notice that the name of the CSS file is changed from **`pXX.css`** to **`p65.css`**):

```
06.     <link rel="stylesheet" type="text/css" href="p65.css" media="screen" />
```

Code Snippet: 6.29

Place the keyboard cursor in the **`body`** element and insert a few LOCs in it as follows:

```
08.   <body>
09.     <div class='block1'>Block1</div>
10.     <div class='block2'>Block2</div>
11.     <div class='block3'>Block3</div>
12.   </body>
```

Code Snippet: 6.30

Now, let us create the CSS file **`p65.css`**. Open Notepad (or your favorite editor) and type the following text in it:

```css
* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
}
.block1 {
  position: fixed;
  top: 40px;
  left: 60px;
  width: 300px;
  height: 200px;
  background-color: cyan;
  border: 3px solid blue;
  z-index: 1;
}
.block2 {
  position: fixed;
  top: 80px;
  left: 100px;
  width: 300px;
  height: 200px;
  background-color: white;
  border: 3px solid blue;
  z-index: 2;
}
.block3 {
  position: fixed;
  top: 120px;
  left: 140px;
  width: 300px;
  height: 200px;
  background-color: yellow;
  border: 3px solid blue;
  z-index: 3;
}
```

Code Snippet: 6.31

Save this file with the filename **`p65.css`** in the folder **`MyFiles`**. Load the file **`p65.html`** in a web browser and your screen will display the webpage as shown in *Figure 6.11 (a):*

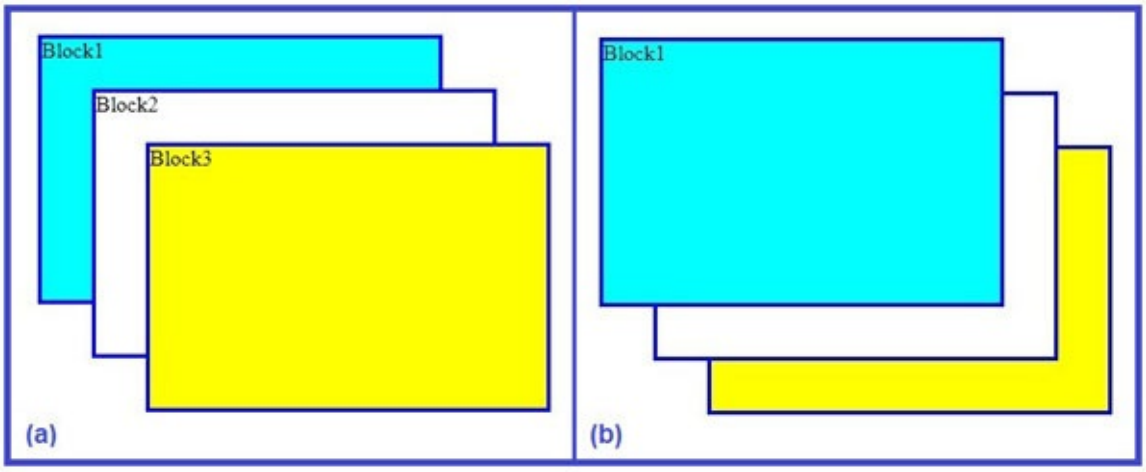

Figure 6.11: *In part (a) of figure, Block1 has z-index value set to 1. Block 2 has z-index value set to 2. Block 3 has z-index value set to 3. In part (b) of figure, Block 1 has z-index value set to 3. Block 2 has z-index value set to 2. Block 3 has z-index value set to 1. Element with the largest z-index value is placed at top and element with smallest z-index value is placed at the bottom.*

Notice that in *Figure 6.11 (a)*, Block 3 is at the top as its z-index value is 3 and Block 1 is at the bottom as its **`z-index`** value is **`1`**. Now, change the **`z-index`** values as follows:

- Block1: **`z-index = 3`**
- Block2: **`z-index = 2`**
- Block3: **`z-index = 1`**

You are required to make changes only in the LOCs 14 and 34 in the file **`p65.css`**. Now, save the changes and refresh the webpage, and you will see the blocks are rearranged as shown in *Figure 6.11 (b)*.

Conclusion

In this chapter, you learned to use the box model and the lists You also learned to apply styles to lists, styles to tables, create a table with single borders, create a hoverable table using the pseudo class **`:hover`**, create a striped table, apply styles to links, apply styles to outlines, apply styles to elements for floating them horizontally, and finally to apply styles to elements to position them on a webpage.

In the next chapter, you will add code to our website project ‘Truly Best Software’.

Further readings/references

- *HTML & CSS: The Complete Reference*, by Thomas A. Powell, 5/e, McGraw Hill, 2010.
- *Beginning HTML and CSS*, by Rob Larsen, Wrox, John Wiley and Sons, 2013.
- *CSS3: The Missing Manual*, by David Sawyer McFarland, 3/r, O’Reilly, 2012.
- https://www.internetingishard.com/

CHAPTER 7

Adding Code to a Website Project

Introduction

In this chapter, you will learn how to add code to various files of our website project, namely, file index.html, file style_index.css, file careers.html, file style_careers.css, file contact.html, and file style_contact.css. You will also test our website offline. Finally, you will be given a few useful tips and suggestions on coding a website.

Structure

In this chapter, we will discuss the following topics:

- Adding code to various files of our website project
- Adding code to the file index.html
- Adding code to the file style_index.css
- Adding code to the file careers.html
- Adding code to the file style_careers.css
- Adding code to the file contact.html
- Adding code to the file style_contact.css
- Testing your website offline
- How to code a website

Objectives

After reading this chapter, you will be able to add the code to various files of our website project. Particularly, you will be able to add code to the file index.html, file style_index.css, file careers.html, file style_careers.css, file contact.html, and file style_contact.css. You will also be able to test your website offline.

Adding code to the file index.html

We have already developed the file **`index.html`** and stored it in the folder **`website`** for our website project in *Chapter 3, Using Images, Audio, Video, and Links*. Now, we will add most of the code to this file. Instead of modifying this file, let us simply delete the contents of the file index.html and recreate the whole file again. Open the file **`index.html`** placed in the folder **`website`**, delete its contents, and type the following code in it:

```
01. <!DOCTYPE html>
02. <html lang="en">
03.   <head>
04.     <title> Truly Best Software </title>
05.     <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
06.     <link rel="shortcut icon" type="image/x-icon" href="images/tbs.ico"/>

07.     <link href='https://fonts.googleapis.com/css?family=Gelasio&effect=neon|outline|emboss|shadow-multiple' rel='stylesheet'>
08.     <link href='https://fonts.googleapis.com/css?family=Rokkitt' rel='stylesheet'>
09.     <link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Sofia&effect=neon|outline|emboss|shadow-multiple|fire">
```

Code Snippet: 7.1

The LOCs 7 and 9 are long and may not be visible clearly to you; hence, these lines are given in the next split in two lines. However, when you type these lines in the code window, ensure that every LOC is typed in a single line. Do not break any LOC into two lines:

```
07.     <link href='https://fonts.googleapis.com/css?family=Gelasio&effect=neon|
            outline|emboss|shadow-multiple' rel='stylesheet'>

08.     <link href='https://fonts.googleapis.com/css?family=Rokkitt' rel='stylesheet'>
09.     <link rel="stylesheet" href="https://fonts.googleapis.com/
            css?family=Sofia&effect=neon|outline|emboss|shadow-multiple|fire">
```

Code Snippet: 7.2

And now LOC 10 and onwards will remain in the usual style:

```
    <link rel="stylesheet" type="text/css" href="style_index.css" media="screen" />
  </head>
  <body>
    <div class="menu">   <!-- div 1 begins -->
      <ul>
        <li> <a href="index.html" class="active">Home</a>  </li>
        <li> <a href="careers.html">Careers</a>  </li>
        <li> <a href="contact.html">Contact Us</a>  </li>
      </ul>
    </div>    <!-- div 1 ends -->
    <div id="head_line">    <!-- div 2 begins -->
      <marquee behavior="scroll" direction="left"> <p id="head_line_text">
      Use The High Quality and Reliable Products Created by the Inquisitive
      Programmers of Truly Best Software.  </p></marquee>
    </div>  <!-- div 2 ends -->
    <h1 class="font-effect-shadow-multiple">Truly Best Software!</h1>
    <p><img src="images/pic01.jpg" alt="programmer"/></p>
    <div class="systems">  <!-- div 3 begins -->
      <p class="font-effect-fire"># Logistic Systems</p>
      <p class="font-effect-fire"># Medical Systems</p>
      <p class="font-effect-fire"># Educational Systems</p>
    </div>  <!-- div 3 ends -->
    <div class="container">  <!-- div 4 begins -->
      <div id="div_text">  <!-- div 5 begins -->
        <h2>Logistic Systems</h2>
        <p>&#9632; Transport Management System</p>
        <p>&#9632; Fleet Management System</p>
        <p>&#9632; Warehouse Management System</p>
        <p>&#9632; Frieght Forwarding System</p>
      </div>  <!-- div 5 ends -->
      <div id="div_text">  <!-- div 6 begins -->
        <h2>Medical Systems</h2>
        <p>&#9632; Diagnosis System</p>
        <p>&#9632; Health Care System</p>
        <p>&#9632; Dental Care System</p>
        <p>&#9632; Surgical Equipment System</p>
      </div>  <!-- div 6 ends -->
      <div id="div_text">  <!-- div 7 begins -->
        <h2>Educational Systems</h2>
        <p>&#9632; Classroom Teaching System</p>
        <p>&#9632; Online Teaching System</p>
        <p>&#9632; Online Examination System</p>
        <p>&#9632; School Management System</p>
      </div>  <!-- div 7 ends -->
    </div>  <!-- div 4 ends -->
    <div class="notice">  <!-- div 8 begins -->
      <p class="font-effect-outline"> One stop shop for all your software needs.
      Visit our office or call us. </p>
    </div>  <!-- div 8 ends -->
    <div class = "copyright">  <!-- div 9 begins -->
      <p> Copyright &copy; 2022 Webilog, India. All rights reserved. </p>
    </div>  <!-- div 09 ends -->
  </body>
</html>
```

Code Snippet: 7.3

The working of the code in the file **`index.html`** is explained in the next section. This is because the working of the code in the file **`index.html`** is dependent on the code in the file **`style_index.css`**. In the next section, we will develop the file **`style_index.css`**.

Adding code to the file style_index.css

Now, let us develop the CSS file **`style_index.css`** that is linked to the HTML file **`index.html`** (see LOC 12 in the file **`index.html`**). Open Notepad (or your favourite editor) and type the following text in it (this text is nothing but the code of the file **`style_index.css`**):

```css
* {
   margin: 0;
   padding: 0;
}
body {
  background-color: cyan;
  color: rgb(0, 0, 255);
}
h1 {
  line-height: 80px;
  font-family: arial;
  font-size: 60px;
  color:green;
  background-color: lime;
  text-align: center;
  font-style: italic;
}
h2 {
  text-align:center;
}
.menu {
  width: 100%;
  background-color: blue;
  color: white;
}
ul {
  text-align: left;
}
ul li {
  width: 120px;
  display: inline-block;
  height: 35px;
  line-height: 35px;
  text-align: center;
}
ul li a {
  font-family: arial;
  font-weight: bold;
  color: white;
  text-decoration: none;
```

```
41.   display: block;
42. }
43. ul li a:hover, ul li a.active {
44.   background-color: orange;
45.   color: black;
46. }
47. #head_line {
48.   height:50px;
49.   background-color: aquamarine;
50.   line-height:50px;
51. }
52. #head_line_text {
53.   font-family: arial;
54.   font-size: 20px;
55.   color:blue;
56.   text-align: center;
57. }
58. img {
59.   height: 300px;
60.   float: right;
61. }
62. .systems p {
63.   font-family: 'Gelasio';
64.   margin-left: 100px;
65.   margin-top: 25px;
66.   text-align: left;
67.   font-size: 50px;
68. }
69. .container {
70.   width: 100%;
71.   clear: both;
72.   background-color: LemonChiffon;
73. }
74. #div_text {
75.   font-family: 'Rokkitt';
76.   width: 25%;
77.   margin-left: 5%;
78.   display: inline-block;
79.   box-shadow: 5px 5px 10px grey;
80.   padding: 10px;
81.   margin-bottom: 15px;
82.   margin-top: 15px;
83.   text-align: left;
84. }
85. #div_text p {
86.   text-align: left;
87.   font-size: 15px;
88. }
89. .notice {
90.   font-family: 'Gelasio';
```

```
91.     line-height: 50px;
92.     text-align: center;
93.     margin-top:15px;
94.     font-size: 30px;
95.     background-color: red;
96.     color: white;
97.     clear: both;
98.  }
99.  .copyright {
100.    line-height: 40px;
101.    margin-bottom: 20px;
102.    font-size: 18px;
103.    text-align: center;
104.    font-family: serif;
105.    background-color: navy;
106.    color: white;
107.  }
```

Code Snippet: 7.4

Save this file with the filename **`style_index.css`** in the folder **`website`**.

Before proceeding further, notice the rule of thumb to study from the given code files.

NOTE: If you want to know the purpose of any LOC in a given code file, then simply delete that LOC, save that code file, refresh the screen, and see how the deletion of that LOC has affected the look and behavior of that web page. Finally, restore the deleted LOC.

Now, follow the explanation of the code in the files **`index.html`** and **`style_index.css`**.

Notice the LOC 6 (file: **`index.html`**) reproduced as follows for your quick reference:

```
<link rel="shortcut icon" type="image/x-icon" href="images/tbs.ico"/>
```

This LOC is responsible for the display of a favicon displayed in the browser's tab as shown in *Figure 7.1*. We created the image **`tbs.ico`** and placed it in the folder **`images`** for this purpose. Notice that traditionally the favicon image must be of type **`ico`** and it must have a size of 16 by 16 pixels. However, this rule is now relaxed and you can also create a favicon with the **`ico`** type image with a file size of 32 by 32 pixels. Apart from this now, you can also create a favicon with the PNG type image provided that the file size is 32 by 32 pixels. Here, the letter **`T`** (the first letter of our company name **`TBS`**) is fitted in the **`ico`** image:

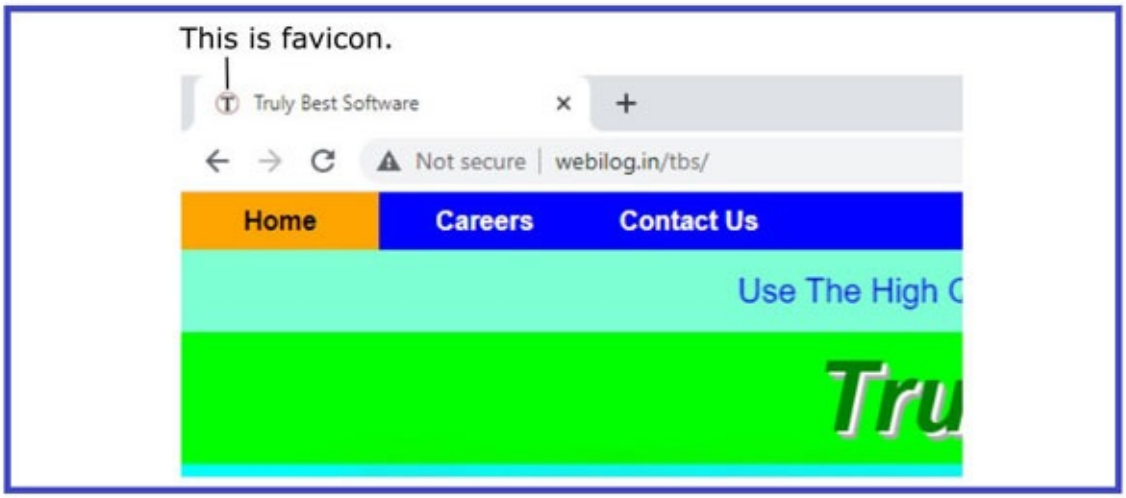

Figure 7.1: *Favicon of our website. It is nothing but an encircled letter T. Size of the image file is 16 x 16 pixels. The extension of the image filename is ico.*

Notice the LOCs 07, 08, and 09 (file: **`index.html`**). These LOCs allow us to use the exotic fonts available on the Internet. Notice the three eye-catching lines, namely:

```
# Logistic Systems
# Medical Systems
# Educationl Systems
```

displayed on the Home page of our website. The exotic fonts used in these lines (and elsewhere on the website) are available to us because of these LOCs. Do not try to dissect these LOCs, just use these LOCs as black boxes wherever you need them.

Notice the LOC 15 (file: **`index.html`**). In this LOC the class **`active`** is created in the list-item Home. This is because when the Home page is displayed on the screen, the Home tab need to look different from the remaining tabs (namely, Careers and Contact). The CSS code for this class is available in the LOCs 43-46 (file: **`style_index.css`**). These LOCs change the color and background color of the selected tab. Similarly, you will notice that in the file, the**`careers.html`** the **`active`** class is created in the list-item Careers, and in the file **`contact.html`**, the **`active`** class is created in the list-item Contact.

The menu bar is small in size but good amount of code is needed to create it. The LOCs 15-19 in the file **`index.html`** and the LOCs 21-46 in the file **`style_index.css`** are responsible for the look and behavior of the menu bar.

LOCs 21-25 (file: **`style_index.html`**) create a CSS rule for the class **`menu`**. This CSS rule applies to LOCs 13-19 in the file **`index.html`** as these LOCs consist of a **`div`** element with a class attribute **`menu`**.

LOCs 26-28 (file: **`style_index.css`**) create a CSS rule for the selector **`ul`** (unordered list) and it applies to LOCs 14-18 in the file **`index.html`** as these LOCs consist of an unordered list.

LOCs 29-35 (file: **`style_index.html`**) create a CSS rule for selector **`ul li`** (for list-item in the unordered list) and it applies to LOCs 15-17 in the file **`index.html`** as each of these LOCs consists of a list-item of an unordered list. Suppose there is an ordered list created somewhere in the file **`index.html`**, then this code (LOCs 29-35) will not apply to list-items in this list because the selector in this CSS rule is **`ul li`**. For the list items in the ordered list, we are required to write separate CSS rules with the selector **`ol li`**.

Let us take a closer look at the code in the LOCs 29-35 (file: style_index.css). In LOC 30, the width of each block on the menu bar is set to 120 px. Notice that every list item in the unordered list (created in the LOCs 15-17 in the file **`index.html`**) is a block. Generally, this width is set by trial and error method.

In LOC 31, the property **`display`** is set with the value **`inline-block`**. Because of this LOC, the list-items Home, Careers, and Contact (in the menu bar) are displayed in a single line. If you delete LOC 31, then these list items would be displayed in three separate lines, one below the other.

Display property

The display property decides how an element is to be displayed on the screen. The four most popular values that are generally assigned to the display property are as follows:

- **none**: When the **`display`** property is set with this value, then that element is not displayed on the screen at all.
- **inline**. When the **`display`** property is set with this value, then that element is displayed as an inline element. For example, the **`p`** element is a block element. But if we set the display property of the **`p`** element with the value **`inline`**, then it behaves like an inline element.
- **block**. When the **`display`** property is set with this value, then that element is displayed as a block element. For example, the element **`img`** is an inline element. But if we set the display property of the **`img`** element with the value **`block`**, then it behaves like a block element.
- **inline-block**. When the **`display`** property is set with this value, then that element is displayed as an inline element but at the same time, it also exhibits a block-like behavior. For example, we can set its properties like **`height`** and **`width`** with suitable values. Notice that the properties **`height`** and **`width`** are available only for block elements. In LOC 31, the property **`display`** is set with this value.

In LOC 32, the property **`height`** of the menu bar is set to **`35`** pixels. This ensures that the menu bar has sufficient height to display any text written on it.

In LOC 33, the property **`line-height`** is set to **`35`** pixels. This ensures that the text written on the menu bar is vertically aligned at the center. Otherwise, by default, the text written on the menu bar is aligned to the top. For this to happen, it is essential that properties **`height`** and **`line-height`** should be set with the same value (here, this value is **`35`** pixels).

In LOC 34, the property **`text-align`** is set with the value **`centre`**. This ensures that the text written in each block on the menu bar is horizontally aligned at the center. Otherwise, by default, this text is aligned to the left in each block.

LOCs 36-42 (file: **`style_index.css`**) consist of another CSS rule. In this CSS rule, the selector is **`ul li a`**. The selector refers to a link that is inside the list item of the unordered list. LOCs 37-40 need no explanation. In LOC 41, the property **`display`** is assigned the value **`block`**. This forces the inline element (anchor or link) to behave like a block. If we delete LOC 41, then the menu bar looks as shown in *Figure 7.2*. Now, we are required to click on text only (Home, Careers, and so on) to activate that menu item. However, because of LOC 41, we can comfortably click anywhere on the block of width 120 pixels and height of 35 pixels to activate that menu item:

Figure 7.2: *Each menu-item in the menu bar looks like a block because of the LOC 41 in the file style_index.css as shown in part (a) of this figure. If we delete this LOC, then menu items look as shown in part (b) of this figure.*

LOCs 43-47 (file: **`style_index.css`**) consist of a CSS rule. In this CSS rule, the selectors are (a) **`ul li a:hover`** and **`ul li a.active`**. The first selector is invoked when we hover a mouse cursor over a menu item and the second selector is invoked when we click on a menu item and bring the concerned page to the front. This CSS rule simply changes the color and background color of the menu item as stated in LOCs 44 and 45. Notice that **`:hover`** is a pseudo class whereas **`active`** is an ordinary class that is created in the LOC 15 (file: **`index.html`**).

Now, take a look at the code file **`index.html`**. LOCs 13-19 consist of a **div** element. We have named it div 1. This **`div`** consists of code related to the menubar. LOCs 20-24 consist of another **`div`** (it's **`div 2`**). This **`div`** consists of an element **`marquee`**. The **`marquee`** element causes the text to scroll from right to left automatically. You can see a line of text – just below the menubar – that runs from right to left and reads **`Use The High Quality and Reliable...`** This line runs because it is inside the **`marquee`** element. We can also make it scroll from left to right, as well as, scroll vertically.

This **`div`** element (**`div 2`**) has an **`id`** attribute that is named **`head_line`**. The CSS rule for the id selector **`head_line`** is given in the LOCs 47-51 in the file **`style_index.css`**. This CSS rules set the height and line height for this **`div`** element to **`50`** pixels. It also sets the background color of this **`div`** element to aquamarine. There is a **`p`** element that is a child element of the **`marquee`** element (see the LOC 21, file: **`index.html`**). This **`p`** element has an **`id`** attribute that is named **`head_line_text`**. The CSS rule for the selector **`head_line_text`** is given in the LOCs 52-57 in the file **`style_index.css`**. This CSS rule sets the font of this running text to **`arial`**, **`font-size`** to **`20`** pixels, and color of the text to blue (see the LOCs 53-55).

LOC 25 (file: **`index.html`**) is responsible for this head line. The LOC 25 consists of the **`h1`** element. This h1 element has the class attribute named **`font-effect-shadow-multiple`**. We have written a CSS rule for the selector **`h1`** in the LOCs 09-17 in the file **`style_index.css`**. However, we have not written a CSS rule for the class selector **`font-effect-shadow-multiple`** in the file **`style_index.css`** because in the LOCs 07-09 (in the file **`index.html`**), we have downloaded a number of utilities, including the CSS rule for the class selector **`font-effect-shadow-multiple`**.

The LOC 26 in the file **`index.html`** is responsible for the display of an image of the programmer just below the headline. LOC 26 consists of a **`p`** element. This **`p`** element, in turn, consists of an **`img`** element (that is, here the **`img`** element is a child element of the **`p`** element) and this **`img`** element displays this image on the screen. Notice the CSS rule written for the selector **`img`** in the LOCs 58-61 in the file

style_index.css. This CSS sets the height of the image to 300 pixels in LOC 59. The width of the image is adjusted accordingly. In LOC 60, the image is floated to the right by setting the property **float** with the value **right**.

LOCs 27-31 (file: **index.html**) consist of a **div** element (**div 3**). This **div** element has a class attribute named **systems** (see LOC 27). This **div** element, in turn, consists of three **p** elements, and each of these **p** elements has a class attribute named **font-effect-fire** (see the LOCs 28-30). The CSS rule for the class selector **systems p** is written in the LOCs 62-68 in the file **style_index.css**. We have not written the CSS rule for the class selector **font-effect-fire** because in the LOCs 07-09 (in the file **index.html**) we have downloaded a number of utilities, including the CSS rule for the class selector **font-effect-fire**. All this code is responsible for the exotic looking three lines of text (to the left of the image of the programmer).

LOCs 32-54 (file: index.html) consist of the **div** element (**div 4**). This **div** element has a class attribute named **container**. This **div**, in turn, consists of three **div** elements, namely, **div 5**, **div 6**, and **div 7**. Each of these three **div** elements (**div 5**, **div 6**, and **div 7**) has an id attribute named **div_text**. The CSS rule for the class selector **container** is written in the LOCs 69-73 (file: **style_index.css**). The CSS rule for the id selector **div_text** is written in the LOCs 74-84 (file: **style_index.css**). Also, the CSS rule for the selector **div_text p** is written in the LOCs 85-88. Notice the LOC 78 in this code which sets the property **display** with the value **inline-block**. This LOC is responsible for putting these three **div**elements in the same line (that is, at the same horizontal level as three columns or three boxes). If you remove this LOC, then these three **div** elements would appear one below the other.

Consider the LOC 35 that is reproduced as follows:

```
<p>&#9632; Transport Management System</p>
```

This is a **p** element. The number **&#9632** represents the symbol **solid square**. Here, we use this symbol as a sort of bullet. For details, see the section *Using special characters* in *Chapter 2, Creating the Web Pages*.

In these three **div** elements (**div 5**, **div 6**, and **div 7**), we have displayed the lists and each list consists of four listitems. However, for the want of more control on the process, we have not used the element **ul** (unordered list) and instead used the **p** elements. However, if you wish, you can use the element **ul** instead of the element **p**.

The remaining code in these two files (namely, **index.html** and **style_index.css**) hardly needs any explanation.

Adding code to the file careers.html

We are now going to write code for the file “careers.html” which is responsible for the display of Careers-page on screen. We have already developed the file **careers.html** and stored it in folder **website** for our website project in *Chapter 3, Using Images, Audio, Video, and Links*. Now, we will add most of the code to this file. Instead of modifying this file, let us simply delete the contents of the file **careers.html** and recreate the whole file again. Open the file **careers.html** placed in the folder **website**, delete its contents, and type the following code in it:

```
<!DOCTYPE html>
  <head>
    <title>Truly Best Software</title>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
    <link rel="shortcut icon" type="image/x-icon" href="images\tbs.ico" />
    <link rel="stylesheet" type="text/css" href="style_careers.css" media="screen" />
  </head>
  <body>
    <div class="menu">    <!-- div 1 begins -->
      <ul>
      <li> <a href="index.html">Home</a>  </li>
      <li> <a href="careers.html" class="active">Careers</a>  </li>
      <li> <a href="contact.html">Contact Us</a>  </li>
     </ul>
   </div>  <!-- div 1 ends  -->
   <h2>Careers at Truly Best Software</h2>
   <p><br></p>
   <div class="container">  <!-- div 2 begins -->
     <p id="topline">We have number of openings for job seekers in our company. </p>
     <p id="topline">Fill in the form given below and submit it.</p> <br>
     <form action="" method="get">  <!-- form begins -->
      <p id="form">Enter your name:
       <input type="text" name="myname" size="20" maxlength="80"/>
      </p>
      <p id="form">Enter your mobile number:
       <input type="tel" name="mymobile" size="10" maxlength="15"/>
      </p>
      <p id="form">Enter your email id:
       <input type="email" name="myemail" size="20" maxlength="60"/>
      </p>
      <p id="form">Enter your birth date:
       <input type="date" name="mybirthdate" size="20" maxlength="60"/>
      </p>
      <p id="form">Enter your highest educational qualification:
       <input type="text" name="myeducation" size="20" maxlength="60"/>
      </p>
      <p id="form">Select the job title in which you are interested:
       <select name="jobtitle">  <!-- drop down list begins -->
        <option value="Select Job Title">
         Select the Job Title
        </option>
        <option value="System Analyst">
                System Analyst
        </option>
        <option value="Hardware Engineer">
                Hardware Engineer
        </option>
        <option value="Software Engineer">
                Software Engineer
        </option>
```

```
51.          <option value="Typist">
52.                  Typist
53.          </option>
54.          <option value="Canteen Manager">
55.                  Canteen Manager
56.          </option>
57.          <option value="Receptionist">
58.                  Receptionist
59.          </option>
60.        </select>  <!-- drop down list ends -->
61.        </p>
62.        <p id="form">Enter your gender: <br>
63.         <input id="female" type="radio" name="mygender" value="f"/>
64.         <label for="female">Female</label> <br>
65.         <input id="male" type="radio" name="mygender" value="m"/>
66.         <label for="male">Male</label>
67.        </p>
68.        <p id="form">Select your hobbies: <br>
69.        <table>  <!-- table begins -->
70.         <tr>
71.          <td>
72.           <input type="checkbox" name="hobbies" value="trekking"/> Trekking
73.          </td>
74.          <td>
75.           <input type="checkbox" name="hobbies" value="cycling"/> Cycling
76.          </td>
77.         </tr>
78.         <tr>
79.          <td>
80.           <input type="checkbox" name="hobbies" value="hockey"/> Hockey
81.          </td>
82.          <td>
83.           <input type="checkbox" name="hobbies" value="cricket"/> Cricket
84.          </td>
85.         </tr>
86.         <tr>
87.          <td>
88.           <input type="checkbox" name="hobbies" value="ball"/> Basketball
89.          </td>
90.          <td>
91.           <input type="checkbox" name="hobbies" value="chess"/> Chess
92.          </td>
93.         </tr>
94.        </table>  <!-- table ends -->
95.        </p>
96.        <p id="form">Upload your resume:
97.         <input type="file" name="myresume" />
98.        </p>
99.        <p id="form">Enter the URL of your website (if any):
100.        <input type="url" name="myurl" />
```

```
      </p>
      <p id="form">
       <input type="checkbox" name="myhobbies" value="trekking"/>
        I accept all the terms and condition of Truly Best Software. <br>
      </p>
      <p id="form">
       <input type="submit" name="mysubmit" value="Submit" id="submit"/>
      </p>
     </form>  <!-- form ends -->
     <p class="notice">Notice: "Truly Best Software" is a fictitious company
       and this is a mock website created for learning purpose only.
     </p>
   </div>    <!-- div 2 ends -->
   <div class = "copyright">  <!-- div 3 begins -->
    <p> Copyright &copy; 2022 Webilog, India. All rights reserved. </p>
   </div>  <!-- div 3 ends -->
  </body>
</html>
```

Code Snippet: 7.5

Working of the code in the file **`careers.html`** is explained in the next section. This is because the working of the code in the file **`careers.html`** is dependent on the code in the file **`style_careers.css`**. In the next section, we will develop the file **`style_careers.css`**.

Adding code to the file style_careers.css

We are now going to write code for the file **`style_careers.css`**. This file works in close cooperation with the file **`careers.html`** which have already coded. Now, let us develop the CSS file **`style_careers.css`** that is linked to the HTML file **`careers.html`** (see LOC 08 in the file **`careers.html`**). Open Notepad (or your favourite editor) and type the following text in it (this text is nothing but the code of the file **`style_careers.css`**):

```
* {
  margin: 0;
  padding: 0;
}
body {
  background-color: cyan;
  color: rgb(0, 0, 255);
}
h2 {
  font-family: arial;
```

```
    font-size: 30px;
    color: rgb(0, 0, 255);
    height:50px;
    background-color: magenta;
    background-color: aquamarine;
    line-height:50px;
    text-align: center;
}
p {
    font-size: 18px;
    font-color: blue;
}
.menu {
    width: 100%;
    background-color: blue;
    color: white;
}
ul {
    text-align: left;
}
ul li {
    width: 120px;
    display: inline-block;
    height: 35px;
    line-height: 35px;
    text-align: center;
}
ul li a {
    font-family: arial;
    font-weight: bold;
    color: white;
    text-decoration: none;
    display: block;
}
ul li a:hover, ul li a.active {
    background-color: orange;
    color: black;
}
.container {
    width: 70%;
    margin-left: 10%
}
#topline {
    color: blue;
    font-size: 20px;
    font-family: arial;
    font-weight: bold;
    }
#form {
    color: blue;
```

```
    font-size: 20px;
    margin-top: 20px;
    font-family: arial;
    font-weight: bold;
  }
  #female {
    margin-left: 10%;
  }
  #male {
    margin-left: 10%;
  }
  table {
    margin-left: 10% ;
    color: blue;
    font-size: 20px;
    font-family: arial;
    font-weight: bold;
    column-width: 1000px;
    }
  td {
    padding: 5px;
  }
  #submit {
    width: 150px;
    height: 40px;
    font-size: 26px;
    background-color: chartreuse;
    color: indigo;
    font-weight: bold;
    border: 5px solid brown;
    border-radius: 15px;
    cursor: pointer;
  }
  .notice {
    margin-top: 10px;
    font-size: 20px;
    color: red;
  }
  .copyright {
    line-height: 40px;
    margin-top: 20px;
    margin-bottom: 20px;
    text-align: center;
    font-family: serif;
    background-color: navy;
    color: white;
  }
```

Code Snippet: 7.6

Save this file with the filename **style_careers.css** in the folder **website**.

Now, follow the explanation of the code in the files **careers.html** and **style_careers.css**.

We have already discussed the working of the menubar in detail in the context of the Home page (files: **index.html** and **style_index.css**) and that discussion will not be repeated here. LOCs 18-113 (file: **careers.html**) consist of a **div** element (**div 2**). This **div** element has a class attribute named **container**. The CSS rule for this class selector is given in the LOCs 49-52 in the file **style_careers.css**. This **div** element, in turn, consists of a **form** element that is coded in the LOCs 21-109. We have already discussed the forms in detail in *Chapter 4, Using Tables and Forms*, and that discussion will not be repeated here.

Notice the LOCs 106-108 (file: **careers.html**). This piece of code consists of a **p** element and this **p** element, in turn, consists of an **input** element of type **submit** in LOC 107. To put it simply, LOC 107 consists of a **submit** button. We need to click on this **submit** button after filling out the form to send the information filled in the form to the server. This **input** element (of type **submit**) has an **id** attribute named **submit**. The CSS rule for this **id** selector is written in the LOCs 83-93 in the file **style_careers.css**. In this CSS rule, we have set the following properties of this **submit** button: width (of **submit** button), height (of **submit** button), font-size (of text written on **submit** button), background-color (this will be color of **submit** button), color (this will be the color of the text written on the **submit** button), border (thickness, style, and color of the border of the **submit** button), border-radius (radius of the corner of the **submit** button, notice that the corners of the **submit** button are rounded), cursor (when we hover the mouse cursor over this button, then it changes its shape as specified here, it changes its shape from the default arrow look to a pointer - raised finger - look).

The remaining code in these two files (namely, **careers.html** and **style_careers.css**) hardly needs any explanation.

Adding code to the file contact.html

We are now going to write code for the file **contact.html** which is responsible for the display of Contact-page on screen. We have already developed the file **contact.html** and stored it in folder **website** for our website project in *Chapter 3, Using Images, Audio, Video, and Links*. Now, we will add most of the code to this file. Instead of modifying this file, let us simply delete the contents of the file **contact.html** and recreate the whole file again. Open the file **contact.html** placed in the folder **website**, delete its contents, and type the following code in it:

```
<!DOCTYPE html>
  <head>
    <title>Truly Best Software</title>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
    <link rel="shortcut icon" type="image/x-icon" href="images\tbs.ico" />
    <link rel="stylesheet" type="text/css" href="style_contact.css" media="screen" />
  </head>
  <body>
    <div class="menu">   <!-- div 1 begins -->
      <ul>
    <li> <a href="index.html">Home</a></li>
    <li> <a href="careers.html">Careers</a></li>
    <li> <a href="contact.html" class="active">Contact Us</a></li>
   </ul>
  </div>  <!-- div 1 ends -->
  <h2>Contact Us</h2> <br><br>
  <div class="container">  <!-- div 2 begins -->
   <h1>Privacy Policy</h1>
   <p id="privacy">
    We collect and maintain personal information of website users including their
    names, addresses, email addresses, mobile numbers, etc. We respect the privacy
    of every individual. We do not sell or rent or give away this data to any other
    party. We collect this data because it is essential and valuable for us. We
    collect this data so that we should be able to provide better service to our
    customers. We are abide by the laws of our country. All disputes in this regard
    would be handled by the courts of our country. Whenever we make changes in our
    privacy policy, we inform the users - on this website - about these changes.
   </p>
  </div>  <!-- div 2 ends -->
  <br><br>
  <div class="container">  <!-- div 3 begins -->
   <h1>Contact Details</h1>
   <p> <br>
    To find out more, call us on +xx-xxxxyyyyzzzz <br><br>
    Our WhatsApp number: ++xx-xxxxyyyyzzzz  <br><br>
    Our Email Id: xyz@xxyyzz.xyz </p> <br>
   <p>Visit us on facebook at: www.facebook.com\xxxyyyzzz </p> <br>
   <p>Visit us on twitter at: www.twitter.com\xxxyyyzzz </p> <br>
   <p>Visit us on youtube at: www.youtube.com\xxxyyyzzz </p> <br>
   <p>Visit us on linkedin at: www.linkedin.com\xxxyyyzzz </p> <br>
   <p>Notice: Truly Best Software is a fictitious company and this website is
      created for the learning purpose only. </p> <br>
  </div>  <!-- div 3 ends -->
  <div class = "copyright">  <!-- div 4 begins -->
   <p> Copyright &copy; 2022 Webilog, India. All rights reserved. </p>
  </div>  <!-- div 4 ends -->
 </body>
</html>
```

Code Snippet: 7.7

The working of the code in the file **`contact.html`** is explained in the next section. This is because the working of the code in the file **`contact.html`** is dependent on the code in the file **`style_contact.css`**. In the next section, we will develop the file **`style_contact.css`**.

Adding code to the file style_contact.css

We are now going to write code for the file **`style_contact.css`**. This file works in close cooperation with the file "contact.html" which have already coded. Now, let us develop the CSS file **`style_contact.css`** that is linked to the HTML file **`contact.html`** (see LOC 08 in the file **`contact.html`**). Open Notepad (or your favourite editor) and type the following text in it (this text is nothing but the code of the file **`style_contact.css`**):

```
01. * {
02.   margin: 0;
03.   padding: 0;
04. }
05. body {
06.   background-color: cyan;
07.   color: rgb(0, 0, 255);
08. }
09. h1 {
10.   font-family: arial;
11.   font-size: 30px;
12.   color: rgb(0, 0, 255);
13.   height:50px;
14.   background-color: indigo;
15.   color: white;
16.   line-height:50px;
17.   text-align: center;
18.   border-radius: 30px;
19. }
20. h2 {
21.   font-family: arial;
22.   font-size: 30px;
23.   color: rgb(0, 0, 255);
24.   height:50px;
25.   background-color: aquamarine;
26.   line-height:50px;
27.   text-align: center;
28. }
29. p {
30.   font-size: 18px;
```

```
    margin-left:30px;
}
.menu {
    width: 100%;
    background-color: blue;
    color: white;
}
ul {
    text-align: left;
}
ul li {
    width: 120px;
    display: inline-block;
    height: 35px;
    line-height: 35px;
    text-align: center;
}
ul li a {
    font-family: arial;
    font-weight: bold;
    color: white;
    text-decoration: none;
    display: block;
}
ul li a:hover, ul li a.active {
    background-color: orange;
    color: black;
}
.container {
    width: 70%;
    margin-left: 15%;
    background-color: indigo;
    color: white;
    padding-right: 30px;
    padding-left: 10px;
    padding-bottom: 30px;
    border-radius: 30px;
}
#privacy {
    text-align: justify ;
}
.copyright {
    line-height: 40px;
    margin-top: 20px;
    margin-bottom: 20px;
    text-align: center;
    font-family: serif;
    background-color: navy;
    color: white;
}
```

Code Snippet: 7.8

Save this file with the filename **`style_contact.css`** in the folder **`website`**.

Now, follow the explanation of the code in the files **`contact.html`** and **`style_conact.css`**.

We have already discussed the working of the menubar in detail in the context of the Home page (files: **`index.html`** and **`style_index.css`**) and that discussion will not be repeated here.

The remaining code in these two files (namely, **`contact.html`** and **`style_contact.css`**) hardly needs any explanation.

Test your website offline

For offline testing of the website, follow the given steps:

- Place the following six files in the folder **`C:\website`** :
    - **`index.html`**
    - **`style_index.css`**
    - **`careers.html`**
    - **`style_careers.css`**
    - **`contact.html`**
    - **`style_contact.css`**
- Place the following two files in the folder **`C:\website\images`**:
    - **`pic01.jpg`**
    - **`tbs.ico`**
- Load the file **`index.html`** in your web browser and ensure that the home page of our website is displayed on the screen successfully as shown in *Figure 1.2*.

How to code a website?

How to code a website? Well, you have already coded a functional website of Truly Best Software and therefore, you know how to code a website. Still, remember the following tips/suggestions while coding a new website:

- **Do not reinvent the wheel**: Do not try to create any code file from scratch. This is like reinventing a wheel. Instead, find any website on the Internet that is very close to your website. Copy all the code files to your machine. In order to access any code file, simply right click on the corresponding web page, a menu pops up on the screen, then click on the menuitem **`View page source`** (alternatively, click on the key combination Ctrl + U) and the concerned code file appears on the screen. Make the necessary changes in these code files depending on the requirements of your website. Delete any irrelevant portion in this code file. Take care that the appearance of your website should not be exactly the same as any other website on the Internet, as this may invoke copyright disputes. Finally, take care that there is no copyright infringement anywhere in your

code.

- **Use a suitable template**: There are a number of websites on the Internet that allow you to download the templates for various types of websites. Whatever may be your need, you will always find a template that is suitable for your need. To begin with, use only free templates. As you become experienced, go for paid templates.
- **Join the groups of website makers**: There are a number of groups of website makers on social platforms like Facebook or LinkedIn. Join the suitable groups. The benefit of joining these groups is that they offer you technical help in making the websites. They also help you in searching the website market for suitable projects. The membership of these groups is free. You can also take the help of companies like Turing or Microverse in order to get website-making projects. But do not make any upfront payment for any project.

Conclusion

In this chapter, we learned to add code to our website project. We added code to the file index.html, file `style_index.css`, file `careers.html`, file `style_careers.css`, file `contact.html`, and file `style_contact.css`. We also learned to test our website offline. Finally, we were given a few tips and suggestions on coding a website. In the next chapter, we will learn the responsive design for mobile and tablet web pages, using the viewport meta tag, using media queries, and testing the responsive web pages.

Further readings/references

- *HTML & CSS: The Complete Reference*, by Thomas A. Powell, 5/e, McGraw Hill, 2010.
- **https://www.registerednursern.com/how-to-build-a-website-from-start-to-finish/**
- **https://forgeandsmith.com/blog/the-finishing-touches-to-your-website/**
- **https://wpmudev.com/blog/finish-website-happy-client/**
- **https://www.aztekweb.com/blog/post/tips-to-finish-your-website-project-faster-and-with-better-results/**

CHAPTER 8

Responsive Web Design for Mobile and Tablet Web Pages

Introduction

In this chapter, you will learn about responsive web design, viewport meta tag, and media queries. In responsive web design, you will learn how to design a web page that adjusts its visibility depending on the size of the screen of the device. We need responsive web design because the web page that is suitable for the desktop screen just fails to fit on the tiny screen of the cell phone. In responsive web design, we delete the less important items in the web page so that the remaining portion of the web page can be displayed on the screen of a cell phone without any congestion. In the viewport meta tag, you will learn how to use this meta tag to inform the browser whether to resize the web page or not. This is a very important meta tag that helps us in responsive web design. Finally, the media query allows us to use If-Then structure in CSS that helps us in identifying whether the user uses a cell phone, tab, or desktop computer.

Structure

In this chapter, we will discuss the following topics:

- What is responsive web design?
- Viewport meta tag
- Using media queries
- Mini project on responsive web design

Objectives

After reading this chapter, you will have workable knowledge about responsive web design, viewport meta tag, and media queries. You will also build a mini project on responsive web design.

What is responsive web design?

Ever since the invention of the World Wide Web, websites have been viewed on computer screens. However, with the advent of time, mobiles and tabs also entered the fray. Mobile and tab users found it inconvenient to view the websites meant for big-sized computer screens on tiny screens of mobiles and tabs. Webmasters realized this problem and decided to do something in this direction. Webmaster Ethan Marcotte, in his famous article published on the website **www.alistapart.com** (A List Apart) in 2010 discussed this problem and also coined the term ‘responsive web design’ in this article. It is worth noting

that in 2020, of all the visits to websites, 68% visits were from mobile users.

Note: Responsive web design refers to a process of creating web pages that look good on all devices. When we view such a website on a smaller screen, then some of the less important items are resized to look smaller and the remaining less important items simply remain hidden from us to make ample space for more important items.

In the remaining part of this chapter, you will learn the techniques of responsive web design. You will also build a mini project on responsive web design.

Viewport meta tag

Notice a typical LOC that uses the **`viewport meta`** tag, given next. When required, this LOC is included in the **`head`** section of the HTML file:

```
<meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1" />
```

Code Snippet: 8.1

Looking at this LOC, you can get some insight into the viewport **`meta`** tag.

When we display any web page on a mobile screen, the mobile browser extensively reduces the size of everything that is on a web page (text, images, and so on) in a desperate attempt to fit the web page on a tiny screen of the mobile. Whenever we are creating a web page with a responsive web design, we need to tell the mobile browser that it should not resize this web page because we are taking care of its display on tiny screens. The preceding LOC tells the mobile browser to display the web page at a scale of 1: 1 (that is, the actual size of the web page).

The part **`width = device-width`** in this LOC sets the width of the page equal to **`device-width`**.

The part **`initial-scale = 1`** sets the initial zoom level to **`1`**, which means no contents of web pages are to be resized, initially.

The part **`maximum-scale=1`** prevents the browsers from enlarging the contents of web pages larger than their original size.

Viewport means a part of a web page that is displayed on your device. Using the viewport meta tag, we can control the contents to be displayed on Viewport.

Using media queries

If you are already a programmer and know at least one high-level language like C, C++, Java, or Python, then you know what is meant by a conditional statement. For example, notice a conditional statement (this one is from C language):

```
if (x > 10)
printf("Value of x is larger than 10.") ;
```

In this conditional statement, the value of the variable **`x`** is checked. If this value of **`x`** is larger than **`10`**, then the line of the text **`Value of x is larger than 10`**. is displayed on the screen. Even if you have never learned the C language, you can still understand the meaning of this statement.

The media queries allow us to use similar conditional statements in the CSS file. The syntax of these

conditional statements (using media queries) is something as shown in the following code snippet (here, items mean items on a web page like text, image, and so on):

```
if (device is computer)
{Display all the items on the screen with full size specified in CSS file.)
if (device is tab)
{Display only these items on the screen with such and such size.)
if (device is mobile)
  {Display only these items on the screen with such and such size.)
```

Now, let us actually write few LOCs using media queries so that you can get some insight about this wonderful thing. Notice the following piece of code:

```
01. @media only screen and (max-width : 600px),
02. only screen and (max-device-width : 600px) {
03.     .text {
04.         font-size: 40px;
05.     }
06.     .picture1{
07.         height: 60px;
08.     }
09.     .picture2{
10.         height: 60px
11.     }
12. }
13. @media only screen and (max-width : 520px),
14. only screen and (max-device-width : 520px) {
15.     .text {
16.         font-size: 20px;
17.     }
18.     .picture1{
19.         display: none;
20.     }
21.     .picture2{
22.         height: 30px
23.     }
24. }
```

Code Snippet: 8.2

This piece of code consists of two blocks of code: one block of code spans LOC 01 to LOC 12 and the second block of code spans from LOC 13 to LOC 24. Consider only the first block of code from LOC 01 to LOC 12.

LOC 01 tells the browser that the following block of code will be effective only if the media is a screen and the maximum width of the screen is 600 pixels. It means that if the media is not the screen but, say, the printer, then the following block of code will not be effective. Also, if the screen size of the device is more than 600 pixels, then also the following block of code will not be effective.

LOC 02 tells the same thing as LOC 01, but it is included for backward compatibility with old browsers. Old browsers know the term **`max-device-width`** and the term **`max-width`** are relatively new and

unknown to them.

LOCs 03-04 create a CSS rule and the selector for this CSS rule is the class **`text`**. This CSS rule sets the **`font-size`** to 40 pixels. Thus, the text inside the HTML element whose class is **`text`** will be affected by this rule and its **`font-size`** will be set to 40 pixels.

LOCs 06-08 create a CSS rule and the selector for this CSS rule is the class **`picture1`**. This CSS rule sets the height of an image (whose class is **`picture1`**) to 60 pixels.

LOC 09-11 does the same thing as LOCs 06-08, but this time, the class name is different. It is **`picture2`** instead of **`picture1`**.

The block of code LOCs 13-24 does almost the same things as block 01-12 but with the following differences:

- This block of code is effective only if the screen size is 520 pixels or less. The preceding block of code is effective only if the screen size is 600 pixels or less. See the LOCs 13 and 14.
- In this block code, the text size is set to 20 pixels instead of 40 pixels. See LOC 16.
- In this block of code, the image with the class **`picture1`** will not be displayed on the screen. See LOC 19.
- In this block code, the image with the class **`picture2`** will have a height of 30 pixels. See LOC 22.

Now, let us implement this newly acquired knowledge in a program. Type the following text in Notepad (or your favorite editor) and save this file with the file name **`p70.html`** in the folder **`MyFiles`**:

```
01. <!DOCTYPE html>
02. <html lang="en">
03.  <head>
04.   <meta charset="utf-8">
05.   <title>Responsive Design for Mobiles and Tablets</title>
06.   <meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1" />
07.   <link rel="stylesheet" type="text/css" href="p70.css" media="screen" />
08.  </head>
09.  <body>
10.   <div class='text'>Webilog</div>
11.    <div>
12.     <img class="pic" src="pic01.jpg" alt="programmer" />
13.     <br/>
14.     <img class="logo" src="webilog.jpg" alt="Webilog" />
15.   </div>
16.  </body>
17. </html>
```

Code Snippet: 8.3

Save this file with the filename **`p70.html`** in the folder **`MyFiles`**.

Now, let us create the CSS file **`p70.css`**. Open Notepad (or your favorite editor) and type the following text in it:

```
.text {
  font-family: arial;
  font-size: 60px;
  font-weight: bold;
  color: blue;
  margin: 20px;
  padding: 10px;
}
.pic{
  display: show;
  height: 200px;
}
.logo{
  height: 130px;
}
@media only screen and (max-width : 600px),
only screen and (max-device-width : 600px) {
  .text {
    font-size: 40px;
    font-style: italic;
    color: red;
  }
  .pic{
    display: none;
  }
  .logo{
    height: 80px;
  }
}
@media only screen and (max-width : 520px),
only screen and (max-device-width : 520px) {
  .text {
    font-size: 30px;
    font-style: normal;
    text-decoration: underline;
    color: hotpink;
  }
  .pic{
    display: none;
  }
  .logo{
    height: 50px;
  }
}
```

Code Snippet: 8.4

Save this file with the filename **`p70.css`** in the folder **`MyFiles`**. Load the file **`p70.html`** in a web browser

and your screen will display the web page as shown in *Figure 8.1 (a)*. Now, resize the browser window gradually in order to decrease its width. As the width becomes equal to (or less than) 600 pixels, the browser windows display the web page as shown in *Figure 8.1 (b)*. Continue resizing the browser window. As the width becomes equal to (or less than) 520 pixels, the browser window displays the web page as shown in *Figure 8.1 (c)*:

Figure 8.1: *Part (a) of this figure shows the web page displayed on the computer screen. Part (b) of this figure shows the web page displayed on a device with a width of less than 600 pixels. Part (c) of this figure shows the web page displayed on the device with a width of less than 520 pixels.*

Now, let us discuss the working of the program **`p70.html`**.

In the **`p70.html`** file, there are two **`div`** elements. The first **`div`** element has the attribute **`class`** and this attribute is assigned the value **`text`**. The second **`div`** element consists of two **`img`** elements. Each of these **`img`** elements has an attribute **`class`**. The **`class`** attribute of the first **`img`** element is named **`pic`** and the **`class`** attribute of the second **`img`** element is named **`logo`**.

Now, consider the file **`p70.css`**. For each selector, three different CSS rules are provided in this file.

For the class selector (or class-name) **`text`**, the three CSS rules are provided in the following LOCs:

- **First CSS rule**: LOCs 01-08. This is for a computer screen.
- **Second CSS rule**: LOCs 18-22. This is for a device with a width of less than 600 pixels.
- **Third CSS rule**: 32-37. This is for a device with a width of less than 520 pixels.

Text (which is nothing but a single word **`Webilog`**) is formatted according to these CSS rules and the meaning of these CSS rules is straightforward.

For the **`class`** selector (or **`class-name`**) **`pic`**, the three CSS rules are provided in the following LOCs:

- **First CSS rule**: LOCs 09-12. This is for a computer screen.
- **Second CSS rule**: LOCs 23-25. This is for a device with a width of less than 600 pixels.
- **Third CSS rule**: 38-40. This is for a device with a width of less than 520 pixels.

Image (this is image of programmer stored in the file **`pic01.jpg`**) is formatted according to these CSS rules and the meaning of these CSS rules is straightforward. In the second and third CSS rules, the property **`display`** is set to **`none`**. The effect of this property setting is that the image of the programmer simply disappears from the screen when the second and third CSS rules are invoked. However, the image again appears on the screen when the first CSS rule is invoked because in the first CSS rule, in LOC 10, the **`display`** property is set to **`show`**.

For the **`class`** selector (or **`class-name`**) **`logo`**, the three CSS rules are provided in the following LOCs:

- **First CSS rule**: LOCs 13-15. This is for a computer screen.
- **Second CSS rule**: LOCs 26-28. This is for a device with a width of less than 600 pixels.
- **Third CSS rule**: 41-43. This is for a device with a width of less than 520 pixels.

Image (this is an image of the Webilog logo stored in the file **`logo.jpg`**) is formatted according to these CSS rules and the meaning of these CSS rules is straightforward. In the second CSS rule, the property **`height`** is set to **`80px`**. In the third CSS rule, the property **`height`** is set to **`50px`**.

This is how the program **`p70.html`** works.

Mini project on a responsive web design

In the preceding section, we developed the program based on a responsive web design. In this section also, we will develop a program based on a responsive web design, but this time, it will be more complex. Almost every website comes with a horizontal menu bar. For example, our project website of TBS consists of a horizontal menu bar with three menu items, namely, Home, Careers, and Contact. In a mobile, the width of the screen is very small and the horizontal menu bar just fails to fit in it despite resizing. Therefore, in a responsive web design, the horizontal menu bar is simply converted into a vertical menu bar like a ladder. Now, this vertical menu bar easily fits on a mobile screen. This is what we want to accomplish in this mini project on a responsive web design.

We have already discussed the basic concepts of responsive web design. You also know how to create a horizontal menu bar. Therefore, now let us create the three code files needed to implement this mini project. After creating the code files, we will test the behavior of this project on screens of various widths. Once we do this, you will get a lot of insight into these code files. Finally, we will discuss the working of the code files.

In what follows now, we will create three code files, namely, **`p71.html`**, **`p71_layout.css`**, and **`p71_resize.css`**. The layout of a web page is covered in the code file **`p71_layout.css`** and the code related to resizing of a web page is put in a separate code file, namely, **`p71_resize.css`**. This is done for the want of clarity. In the preceding program, being a tiny program, the code for layout was put in the file **`p70.css`** in the LOCs 01-15 and the code for resizing was put in the same file in LOCs 16-44.

Type the following text in Notepad (or your favorite editor) and save this file with the file name **`p71.html`** in the folder **`MyFiles`**:

```
<!DOCTYPE html>
<html lang="en">
 <head>
  <title>Webilog</title>
  <meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1" />
  <link rel="stylesheet" href="p71_layout.css" type="text/css" media="screen" />
  <link rel="stylesheet" href="p71_resize.css" type="text/css" media="screen" />
 </head>
 <body>
  <h1>Welcome to Webilog</h1>
  <div class="header">
   <img class="logo" src="webilog.jpg" alt="Webilog" />
  </div>
  <ul id="navigation">
   <li><a href="index.html">Home</a></li>
   <li><a href="index.html">Careers</a></li>
   <li><a href="index.html">Shopping Cart</a></li>
   <li><a href="index.html">Contact Us</a></li>
  </ul>
 </body>
</html>
```

Code Snippet: 8.5

Save this file with the filename **`p71.html`** in the folder **`MyFiles`**.

Now, let us create the CSS file **`p71_layout.css`**. Open Notepad (or your favorite editor) and type the following text in it:

```
body {
  margin: 0px;
  background: white;
  font-family: arial;
  font-size: 14px;
}
```

```
h1 {
  font-size: 28px;
  margin-left: 20px;
}
div.header {
    float: left;
    position: relative;
    width: 94%;
    padding: 8px 3%;
}
div.header img.logo {
  float: left;
  max-width: 70%;
}
ul#navigation {
    float: left;
    width: 94%;
    margin: 0px;
    padding: 0px 3%;
    background: greenyellow;
}
ul#navigation li {
    float: left;
    list-style: none;
    margin: 0px;
    padding: 0px;
}
ul#navigation li a {
    float: left;
    padding: 6px 12px;
    font-size: 20px;
    font-weight: bold;
    color: darkblue;
    text-decoration: none;
}
ul#navigation li a:hover {
    background: black;
    color: cyan;
}
ul#navigation li a:active {
    background: cyan;
    color: black;
}
```

Code Snippet: 8.6

Save this file with the filename **`p71_layout.css`** in the folder **`MyFiles`**.

Now, let us create the CSS file **`p71_resize.css`**. Open Notepad (or your favorite editor) and type the following text in it:

```
@media only screen and (max-width : 600px),
only screen and (max-device-width : 600px) {
  h1 {
    font-size: 22px;
  }
  ul#navigation li a {
    font-size: 14px;
    font-weight: bold;
  }
}
@media only screen and (max-width : 520px),
only screen and (max-device-width : 520px) {
  ul#navigation {
    padding: 0px;
    width: 100%;
  }
  ul#navigation li {
    width: 100%;
  }
  ul#navigation li a {
    width: 100%;
    padding: 12px 0px;
    text-align: center;
    border-top: solid 1px blue;
    border-bottom: solid 1px blue;
  }
}
```

Code Snippet: 8.7

Save this file with the filename **`p71_resize.css`** in the folder **`MyFiles`**. All the three code files of this mini project are now successfully created.

Load the file **`p71.html`** in a web browser and your screen will display the web page as shown in *Figure 8.2*. Now, resize the browser window gradually in order to decrease its width. As the width becomes equal to (or less than) 600 pixels, the browser windows display the web page as shown in *Figure 8.3*. Continue resizing the browser window.

As the width becomes equal to (or less than) 520 pixels, the browser window displays the web page as shown in *Figure 8.4*:

Figure 8.2: *Web page of our mini project is displayed on the screen. This display is meant for a regular computer screen. Here, everything is in full size.*

The following figure displays the tabs with width of less than 600 pixels:

Figure 8.3: *Web page of our mini project is displayed on the screen. This display is meant for tabs with width of less than 600 pixels. Here, everything - text and image - is resized. Compare this figure with Figure 8.2.*

The following figure displays the tabs with a width of less than 520 pixels:

Figure 8.4: *Web page of our mini project is displayed on the screen. This display is meant for tabs with a width of less than 520 pixels. Here, everything - text and image - is resized. Also, the horizontal menu bar is converted into a vertical menu bar. Compare this figure with Figures 8.2 and 8.3.*

Now, let us discuss the working of code in this program. Consider the code file **`p71.html`**. In LOC 5, the meta viewport tag is included. In the LOCs 06 and 07, this file (**`p71.html`**) is linked to CSS files **`p71_layout.css`** and **`p71_resize.css`**. The LOC 10 consists of an element **`h1`** which, in turn, includes the text **`Welcome to Webilog`**. The LOCs 11-13 consist of an element **`div`** which, in turn, consists of an

element **img**. This element is all set to display the image stored in the file **webilog.jpg**. The **class** attribute of this **img** element is **logo**. The LOCs 14-19 consist of the element **unordered list** and the **id** attribute of this element is **navigation**.

This element is all set to display a horizontal menu bar that consists of four menu items, namely, Home, Careers, Shopping Cart, and Contact Us. However, the property **href** in the case of each of these items is set to **index.file**. It means that if you click on any of these menu items, then you will stay on the home page and will not go elsewhere. This is done to keep the project as simple as possible. The purpose of this project is only to demonstrate the capabilities of responsive web design.

Consider the code file **p71_layout.css**. The code in this file looks after the layout of the web page that is meant for a computer screen. This code ensures that your web page looks as shown in *Figure 8.2*. We have already discussed the creation of a horizontal menu bar in detail in the previous chapters.

Consider the code file **p71_resize.css**. The code in this file is responsible for the responsive web design of this web page. The code in this file consists of two code blocks. The first code block consists of LOCs 01-10. The second code consists of LOCs 11-27. The first code block is responsible for the behavior of the web page when the width of the device screen is 600 pixels or less than 600 pixels. The second code block is responsible for the behavior of the web page when the width of the device screen is 520 pixels or less.

Firstly, consider the first code block. In this code block, the font size of the **h1** element is set to 22 pixels. It means now the text under element **h1** will look smaller. Recall the font size of the text in the **h1** element is set to 28 pixels in the LOC 09 in the file **p71_layout.css**.

In LOCs 07, the font size of the text in menu items is set to 14 pixels. It means now the text in menu items will look smaller. Recall the font size of the text in menu items is set to 20 pixels in the LOC 38 in the file **p71_layout.css**.

The image of the logo gets resized automatically as we resize the browser window; hence, there is no need for code to make this image smaller.

The second code block is more complex compared to the first code block. This is because in the second code block, the provision is made for converting the horizontal menu bar into a vertical menu bar. The font size of the text is not reset in the second block of code.

How the horizontal menu bar is converted into a vertical menu bar? The secret lies in the LOCs 15, 18, and 21. Each of these LOCs sets the width to 100%. The LOC 15 sets the width of the “unordered list” to 100%. The LOC 18 sets the width of each list item in “unordered list: to 100%. The LOC 21 sets the width of each “link” in “list item” in “unordered list” to 100%. The total effect of these LOCs is that the horizontal menu bar is converted into a vertical menu bar and each menu item is horizontally centered. In LOC 18, we force each menu item to occupy all the width available to the menu bar; hence, only one menu item can be accommodated in one line. Suppose we set the width to 50% in LOC 18, then there will be two menu items (instead of one) in each horizontal line. The LOC 21 forces each menu item to be at the center. Suppose we set the width to 50% in LOC 21, then the whole menu will be fitted on the left side in 50% of the available width. This is how the horizontal menu bar is converted into a vertical menu bar.

Conclusion

In this chapter, we learned about responsive web design, the viewport meta tag, and media queries. We also built a mini project on responsive design. In the next chapter, we will learn about buying a domain name, buying a hosting plan, linking the domain name to the hosting plan, and finally, uploading the

website files to the hosting server.

Further readings / references

- *Beginning Responsive Web Design with HTML5 and CSS3*, by Jonathan Fielding, Apress, 2014.
- *Responsive Web Design with HTML5 and CSS*, by Ben Frain, 3/e, Packt, 2020.
- *Responsive Web Design*, by Ethan Marcotte, 2/e, A Book Apart, 2014.
- *Responsive Web Design with HTML 5 and CSS*, by Jessica Minnick, Cengage, 2020.
- *Learning Responsive Web Design: A Beginner's Guide*, by Clarissa Peterson, O'Reilly, 2014.
- *Responsive Web Design in 24 Hours*, by Kyrnin Jennifer, Sams Publishing, 2014.

CHAPTER 9

Uploading a Website to a Hosting Server

Introduction

In this chapter, you will learn to buy a domain name, buy a hosting plan, link the domain name to the hosting plan using the software cPanel, upload the files to the hosting server using the software cPanel, upload the files to the domain **`ourbusinesswebsite.com`**, upload the files to the URL **`www.webilog.in/tbs/`**, and make our website online.

Structure

In this chapter, we will discuss the following topics:

- Buying a domain name
- Buying a hosting plan
- Linking a domain name to a hosting plan using cPanel
- Uploading the files to the hosting server using cPanel
- Uploading the files to the domain **`ourbusinesswebsite.com`**
- Uploading the files to URL **`www.webilog.in/tbs/`**
- Making the website online

Objectives

After reading this chapter, you will be able to buy a domain name from a suitable registrar, buy a suitable hosting plan from this registrar, link a domain name to the hosting plan, upload the files to the hosting server using cPanel, upload the files to the domain **`ourbusinesswebsite.com`**, upload the files to URL **`www.webilog.in/tbs/`**, and finally, you will be able to make our website online on the Internet.

Buying a domain name

In order to launch your website, you are required to buy (or register, these words are synonyms in this context) a suitable domain name for your website. For example, **www.bpbonline.com**, **www.webilog.in**, and **www.astrojadhav.org** are the domain names of existing websites. While choosing a domain name, take care of the following:

- Your domain name should be simple and easy to pronounce and spell.
- Do not use any copyrighted word or registered trademark as a domain name. For example, you should avoid the use of the word Tata in your domain name as Tata is a registered trademark.

- It should advertise your business.
- The domain name you want to buy must be unique. You cannot go for a domain name that is already bought by somebody. For example, you cannot buy the domain name www.facebook.com because somebody has already bought this domain name.

We are required to buy the domain name for one year by making some payment to the domain name registrar. To keep this domain name alive, we are required to continue the registration every year by making some payment to the domain name registrar. Fees of the domain name registration vary from registrar to registrar and from extension to extension. On an average, you are required to pay about Rs. 1200 per year for a domain name.

The domain registrar must be authorized by ICANN. There are a number of domain registrars available in India such as GoDaddy, Hostinger, and so on. Before finalizing your domain registrar, compare their prices. Some registrars give first-year discounted prices in bold letters and regular non-discounted prices in small font sizes; therefore, be careful. Apart from the price of the domain, the reputation of the registrar is also important. Do not purchase the domain name from a totally unknown registrar who can abscond anytime by closing his shop.

Note: Remember, in order to create your website, you are required to buy not only the domain name but also a hosting plan. It is advisable to buy these both items from a single party to avoid the hassle of transferring of the domain name. Therefore, before finalizing your registrar, check the prices of domain names, as well as, hosting plans. Only after that, go ahead and buy the domain name.

After finalizing on your registrar (for example, Godaddy, Hostinger, and so on), go to their website and follow the given steps:

1. Sign up for your account on the website of the registrar.
2. Log in to your account on the website of the registrar.
3. Type the domain name of your choice in the edit-box provided and check its availability.
4. Once you finalize the domain name that is available and within your budget, add it to the cart.
5. Go ahead and buy it.

Now, congratulate yourself as you have successfully bought the domain name. If there is any problem, contact the help center of your registrar. The link of the help center is provided on their website.

Let us assume that the domain name you have bought is **`ourbusinesswebsite.com`**. Also, notice that this is a fictitious domain name and you have not actually bought it but just mimicked the procedure of buying it.

Buying a hosting plan

While buying a hosting plan, check for the following features:

- What type of hosting do they offer? Prefer managed hosting over unmanaged hosting. In managed hosting, your registrar handles the troublesome jobs like the security of software updates. If you are a newbie, then go for a shared hosting plan that serves your purpose. Experienced webmasters

can go for **VPS** (**Virtual Private Server**) or cloud hosting.

- Look for high quality, round-the-clock support so that if the website fails or any problem occurs, then their support team should be available to solve your problem. They should provide you support on phone, chat window, WhatsApp, and Telegram.
- There should be a facility for automatic backups. This is particularly helpful if some stupid hacks your website or server failure occurs resulting in loss of data.
- For storage of data, they should provide you **SSD** (**Solid State Disks**) rather than old-fashioned **HDD** (**Hard Disk Drives**). HDDs are prone to failure and their speed is also low. SSDs are very fast and reliable.
- We need software to control the website. Nowadays, the software cPanel is mostly provided by registrars and it is good, particularly for newbies. However, if you are an experienced webmaster, then you can go for other control software.
- Look for the feature **`one-click installation of website`**. This feature is particularly helpful for newbies.
- Choice of server location. Your server should be located close to your audience. Therefore, your registrar should offer you the choice of server location so that you select the server location that is suitable for your audience.
- Your registrar should be aware of website security. If not, you will face many difficulties in managing your website.
- Check whether your hosting plan supports only one website or multiple websites. If you intend to be a website professional, then go for a plan that supports multiple websites. By hosting multiple websites, you can earn more money.
- Also, check for disk space, size of data transfer, number of email accounts that can be created, number of databases that can be created, free **SSL** (**Secure Sockets Layer**) certificate, and support for PHP and MySQL.

Linking the domain name to a hosting plan using cPanel

The procedure of linking the domain name to the hosting plan differs from registrar to registrar and also from hosting plan to hosting plan. Therefore, it is not possible to give precise steps for linking the domain name to the hosting plan. Let us proceed further assuming:

- That you have bought a shared hosting plan because most of the domain names are linked to a shared hosting plan.
- That your registrar uses cPanel to control the website.
- That the primary domain is already added to your shared hosting plan and now you are going to add your domain name as Addon Domain to your shared hosting plan.

With these assumptions, let us see how to add your domain name to your shared hosting plan as an Addon Domain. Follow these steps:

1. Log on to your account on the registrar's website and access the **`My Products`** page of your account.

2. Find the section **`Web Hosting`** (see *Figure 9.1*). Find the shared hosting plan you have bought. Here, we have bought two hosting plans:
    - **`Economy Linux Hosting with cPanel`** (for a single website)
    - **`Deluxe Linux Hosting with cPanel`** (for multiple websites), as shown in the following *Figure 9.1*:

Figure 9.1: *Click on the button Manage in front of Deluxe Linux Hosting with cPanel and Dashboard page opens on the screen.*

3. Click on the **`Manage`** button in front of **`Deluxe Linux Hosting with cPanel`**. Now, the dashboard page opens with the text **`myprimary.com`** in the top, left corner (this is the primary domain already attached to the shared hosting plan, in my shared hosting plan, the primary domain is **`shirishchavan.in`**; in yours, it will be different). Click on the button **`cPanel Admin`** on this page and a new page (**`cPanel`** page) opens.
4. In the **`cPanel`** page, find the section **`DOMAINS`**. It is the third section from the top (see *Figure 9.2*). The first section is the **`FILES`** and the second section is the **`DATABASES`**:

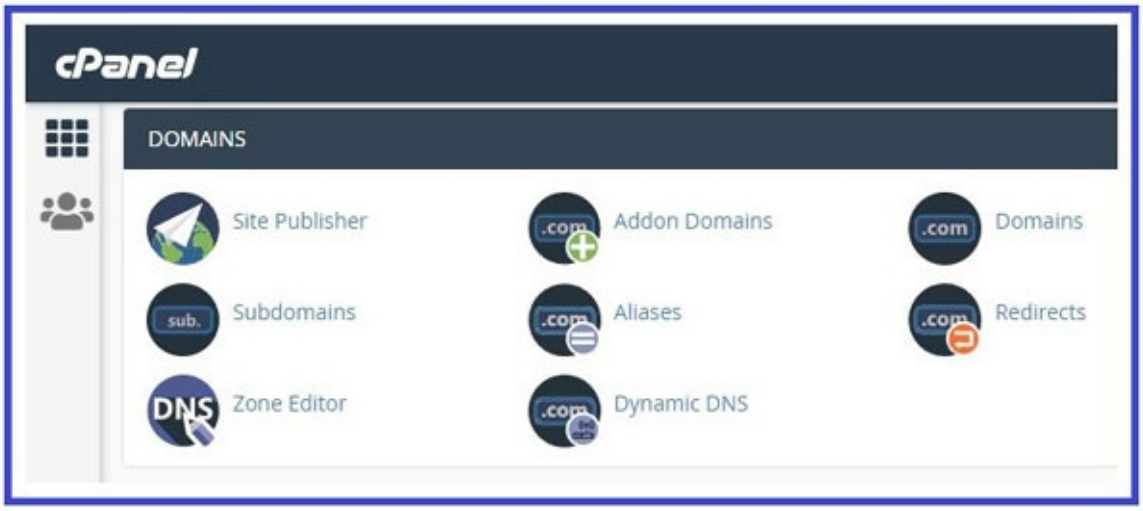

Figure 9.2: *Click on the icon Addon Domains and a new page titled Addon Domains opens on the screen (see Figure 9.3).*

5. In the **`DOMAINS`** section, click on the icon **`Addon Domains,`** and a new page titled **`Addon Domains`** opens (see *Figure 9.3*):

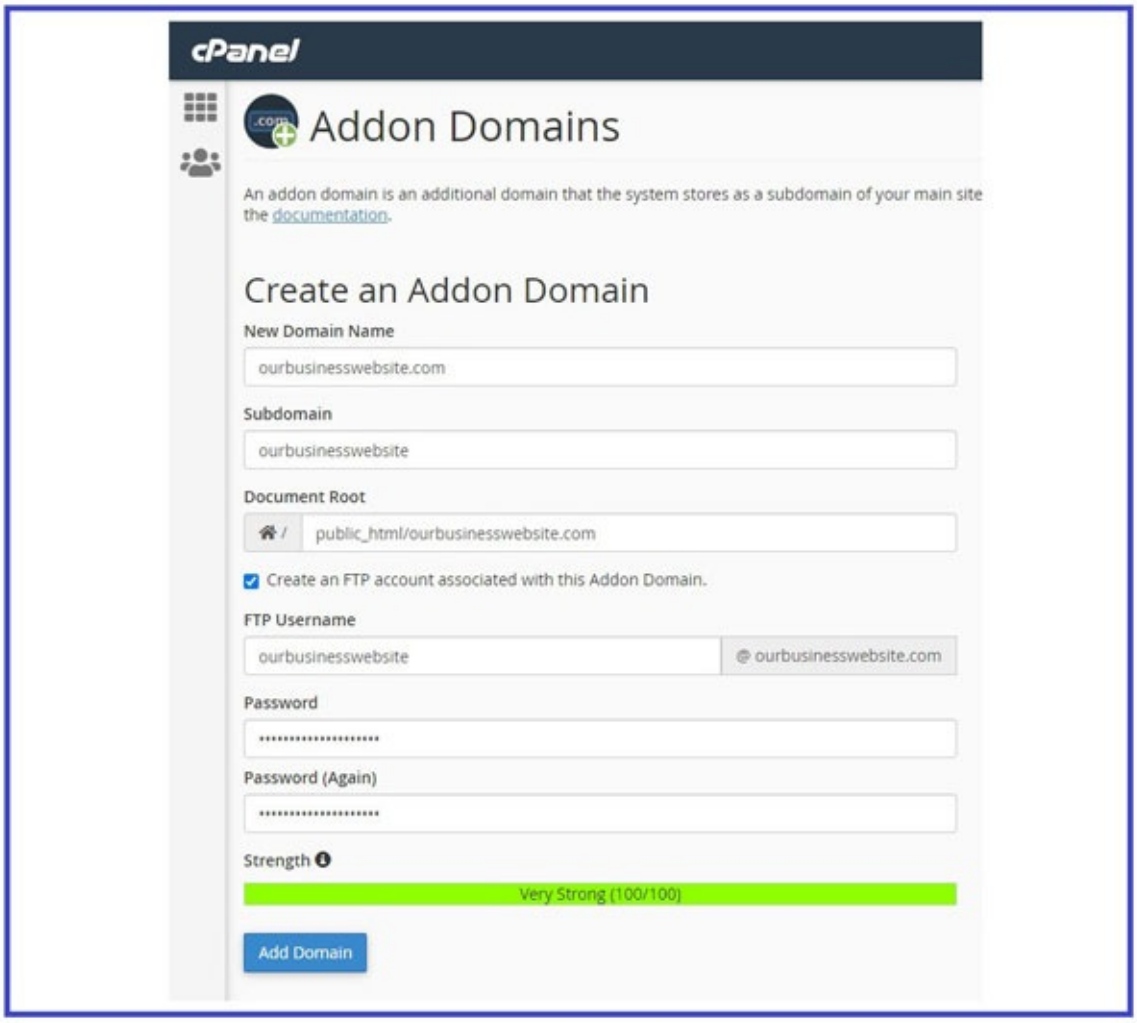

Figure 9.3: *Fill the various edit-boxes on the Addon Domains page as shown here. Finally, click on the Add Domain button.*

6. This page consists of a number of edit-boxes (see *Figure 9.3*). Click on the top-most edit-box labeled as **`New Domain Name`** and type the following domain name in it (this is the secondary domain we want to link with this shared hosting plan; remember, you bought this domain name):
   `ourbusinesswebsite.com`
7. Click on the next edit-box labeled as **`Subdomain`** and type the following text in it:
   `ourbusinesswebsite`
8. Edit-box is labeled as **`Document Root`**. Click on this edit-box and type the following text in it:
   `ourbusinesswebsite.com`
9. Next, there is a checkbox labeled **`Create an FTP account associated with this Addon Domain`**. Click on this checkbox to check it.
10. Now, four new edit-boxes are created. The first of these four edit-boxes are labeled FTP Username. Click on this edit-box and type the following name in it:
    `ourbusinesswebsite`
11. The next edit-box is labeled **`Password`**. Click on this edit-box and type the following text in it (this is your password):
    `xxxxxxxxxxxxxx`
12. The next edit-box is labeled **`Password (Again)`**. Click on this edit-box and type the following text in it:
    `xxxxxxxxxxxxxx`
13. Verify that the strength of the password displayed is good. Ideally, it should be 100/100.

14. Next, there is a blue button labeled **`Add Domain`**. Click on this button.
15. Within a few seconds, the appearance of this page changes. Now, the page displays the following messages:

The addon domain "ourbusinesswebsite.com" has been created. The FTP account"ourbusinesswebsite@ourbusinesswebsite.com" has been created.

If you would like to manage the files for this domain, you can do so here: File Manager

Now, log out of your account.

Uploading the files to the hosting server using cPanel

Firstly, we will upload the files to the fictitious domain name "www.ourbusinesswebsite.com." But as this domain name is fictitious, you cannot get an online test of this website. Therefore, we will upload the files again to the folder **`tbs`** on the website **www.webilog.in** which is a real website. Then, you can actually view this website online on the following URL: **www.webilog.in/tbs**.

Uploading the files to the domain ourbusinesswebsite.com

Let us upload the files to the domain **`ourbusinesswebsite.com`**. Follow the given steps:

1. Ensure that the folder **`C:\website`** consists of the following files:
    - index.html
    - style_index.css
    - careers.html
    - style_careers.css
    - contact.html
    - style_contact.css
2. Also, ensure that the folder **`C:\website\images`** consists of the following files:
    - pic01.jpg
    - tbs.ico
3. Load the file **`index.html`** in a web browser and ensure that our website is working flawlessly. If there is any error, then spot it and correct it. Before uploading any file always ensure that there is no error in that file.
4. Log on to your account on the website of your domain name registrar (or hosting provider, as both parties are the same).
5. Go to **`My Products`** page by clicking on the appropriate link.
6. Find the section **`Web Hosting`** (see *Figure 9.1*). Find the **`Shared Hosting Plan`** you have bought. We have bought two hosting plans:
    - **`Economy Linux Hosting with cPanel`** (for a single website)

- **Deluxe Linux Hosting with cPanel** (for multiple websites)

7. Click on the **Manage** button in front of **Delux Linux Hosting with cPanel**. Now, the dashboard page opens with the text **myprimary.com** on the top, left corner (this is the primary domain already attached to a shared hosting plan. In my shared hosting plan, the primary domain is **shirishchavan.in**; in yours, it will be different). Click on the link **File Manager** in the top row on this page.
8. Now, the **File Manager** page opens as shown in *Figure 9.4*. On the **File Manager** page, in the left pane, the name of the current folder looks bold indicating that it is the current folder. The contents of the current folder are displayed in the right pane. In *Figure 9.4*, the folder **/home/vgwsk4nru9ef** is the current folder and hence, this folder name in the left pane looks bold and its contents are displayed in the right pane. In *Figure 9.5*, the folder name **public_html** in the left pane looks bold because it is the current folder and its contents are displayed in the right pane. In *Figures 9.6*, *9.7*, and *9.13*, the folder name **ourbusinesswebsite.com** in the left pane looks bold because it is the current folder and its contents are displayed in the right pane. Also, notice that the uploaded files are placed in the current folder, therefore before uploading any file always ensure that the desired folder is the current folder.

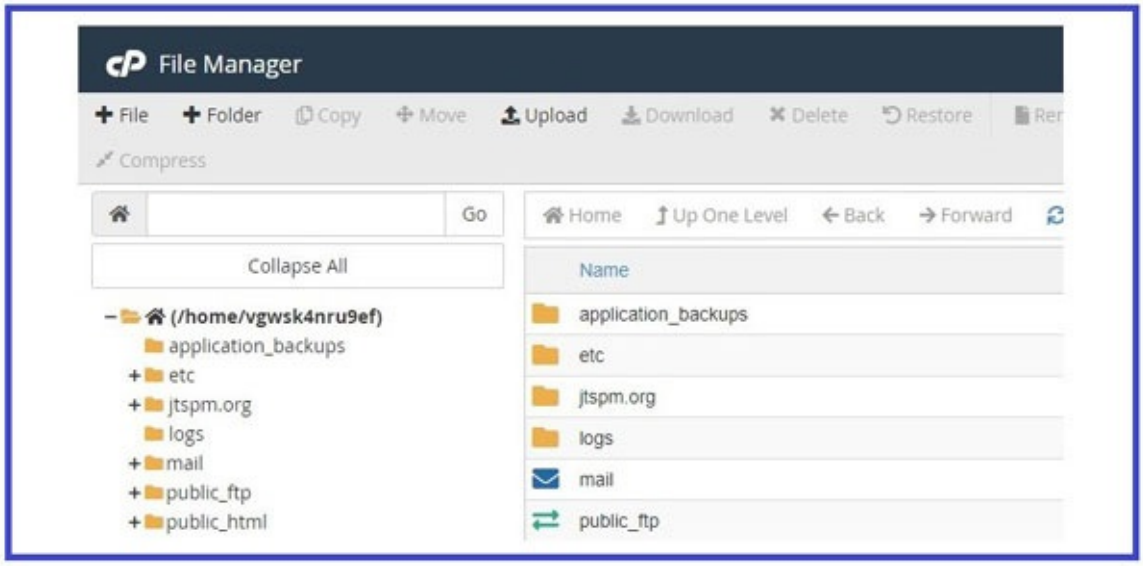

Figure 9.4: *This is the File Manager page. Click on the folder name public_html (while clicking, place the mouse cursor on the folder name) located in the left pane of this page the contents of this folder are then displayed in the right pane as shown in Figure 9.5.*

9. On this page, notice the folder icon **public_html** located in the left pane of this page. Click on the folder name **public_html** (while clicking, place the mouse cursor on the folder name) and the contents of this folder are then displayed in the right pane of this page as shown in *Figure 9.5*:

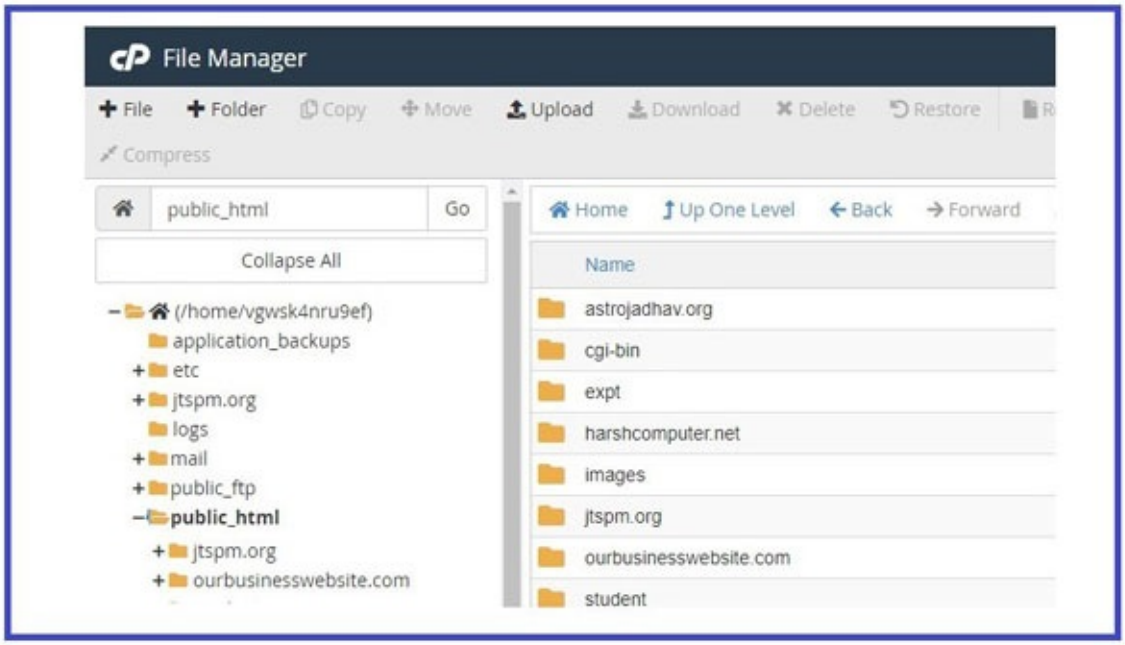

Figure 9.5: *The contents of the folder public_html is displayed in the right pane.*

10. Now, click on the folder name **`ourbusinesswebsite.com`** (while clicking, place the mouse cursor on the folder name) located in the left pane of this page, and the contents of this folder (which is nothing but a lone folder **`cgi-bin`**) are then displayed in the right pane of this page as shown in *Figure 9.6*:

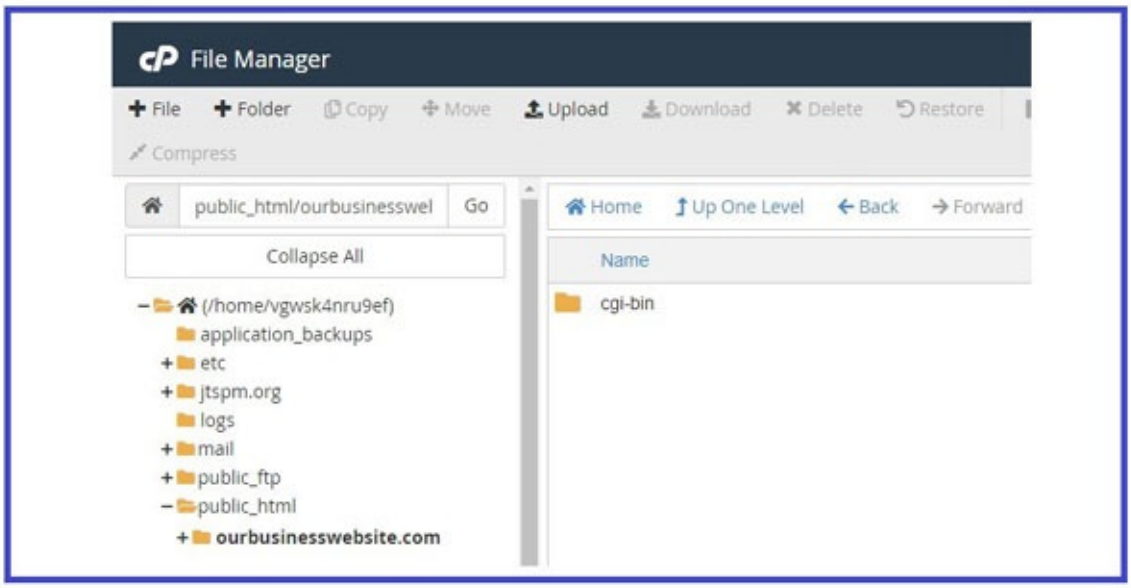

Figure 9.6: *The contents of the folder ourbusinesswebsite.com are displayed in the right pane. Click on the menu-item Folder in the menu bar of this page and the dialog-box "New Folder" pops up on the screen as shown in Figure 9.7.*

11. We want to transfer all the contents of the folder **`C:\Website`** to the current folder **`ourbusinesswebsite.com`**. Firstly, let us create the folder **`images`** to place the image files. Click on the menu-item **`Folder`** in the menu bar of this page (located at the top of the page) and the dialog-box **`New Folder`** pops up on the screen as shown in *Figure 9.7*. Type the **`images`** text in the edit-box **`New Folder Name`** (this is the name of our new folder) and click on the button **`Create New Folder`** on this dialog-box. Now, this dialog box disappears from the page:

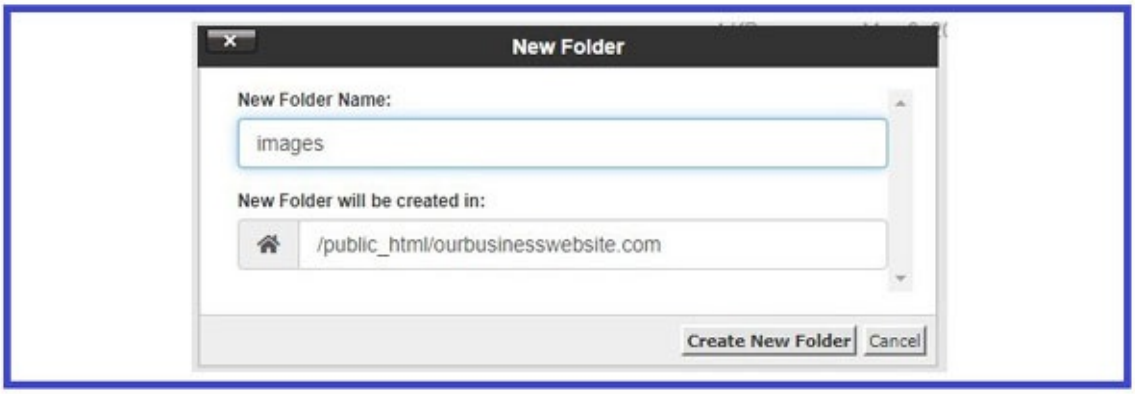

Figure 9.7: *The dialog box New Folder. Type the text images in the edit-box provided and click on the button Create New Folder to create the folder images.*

12. Now, the right pane shows the folder **`images`** that we created just now as shown in *Figure 9.8*. Now, this is the right time to upload the files to a hosting server. Ensure that the folder **`ourbusinesswebsite.com`** is the current folder (the name of the current folder looks bold in the left pane). Click on the menu item **`Upload`** in the menu bar located at the top and the **`File Upload`** page appears on the screen as shown in *Figure 9.9*:

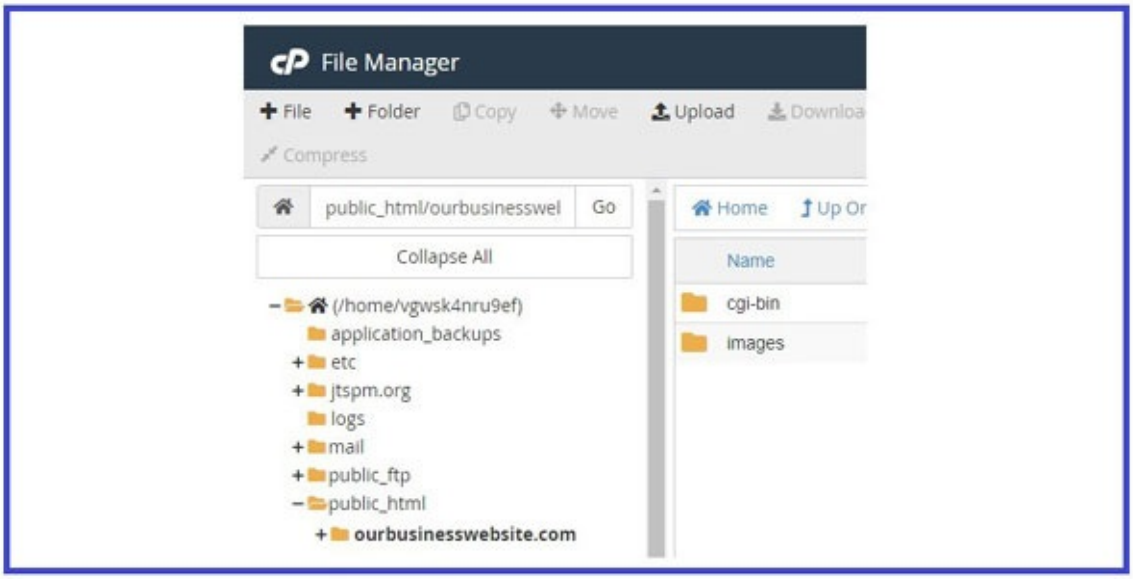

Figure 9.8: *The folder name images is displayed in the right pane. Now, click on the menu item Upload in the menu bar at the top to upload the files to a hosting server and the File Upload page appears on the screen as shown in Figure 9.9.*

13. Now in order to upload the files, you can either drop the files in the dotted rectangle provided on this page or simply click on the button **`Select File`** and make the appropriate entries in the dialog box Open. Let us use the second option. Click on the button **`Select File`** and the dialog box **`Open`** pops up on the screen as shown in *Figure 9.10*:

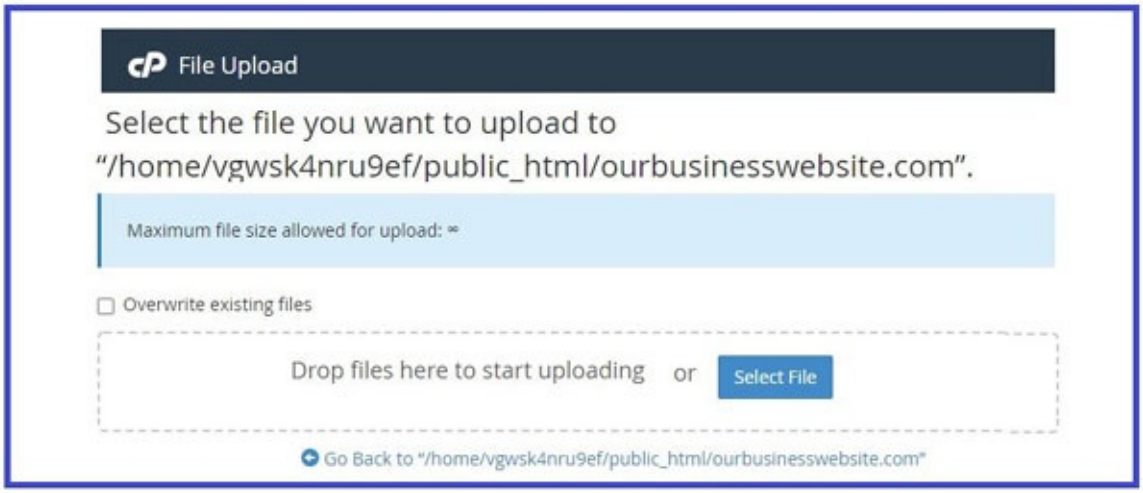

Figure 9.9: *The File Upload page. Now, click on the button Select File and the dialog box Open pops up on the screen as shown in Figure 9.10.*

14. Set the folder **`C:\website`** as the current folder in this dialog box **`Open`**. Now, the contents of the folder **`C:\website`** are displayed in the pane of this dialog box as shown in *Figure 9.10*:

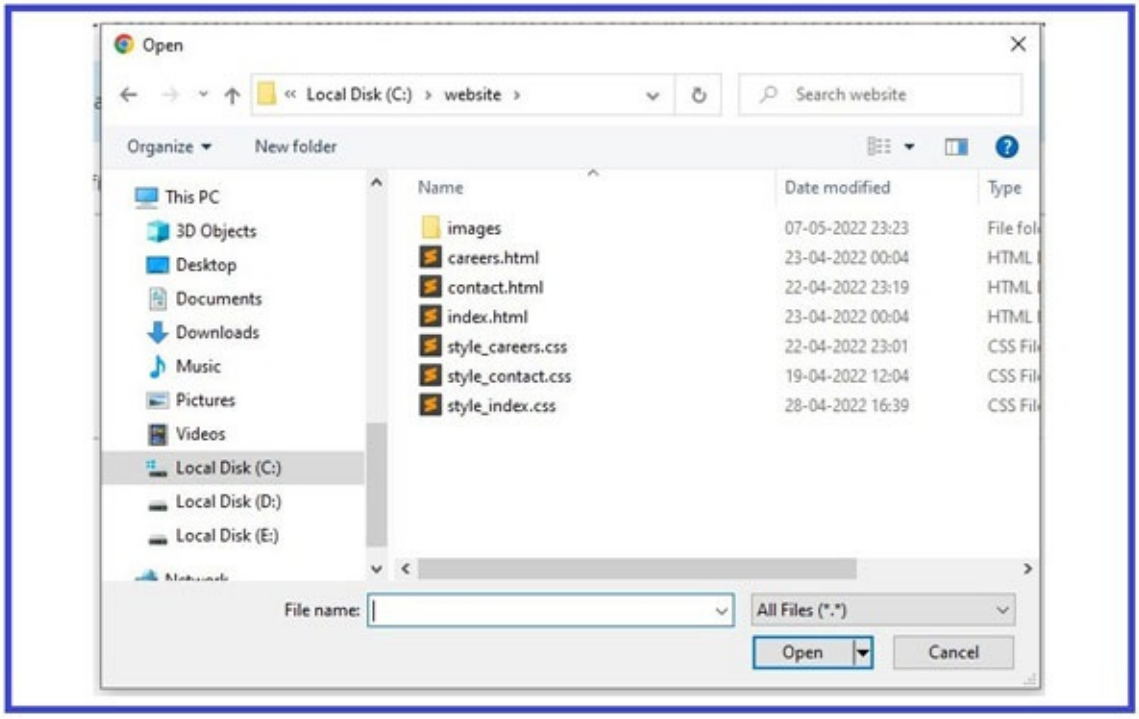

Figure 9.10: *The dialog box Open. Set the folder C:\website as the current folder and then contents of this folder are displayed in the pane. Now, double click on the filename careers.html displayed in the pane, and this file gets uploaded to the hosting server.*

15. Double click on the filename **`careers.html`**, dialog box **`Open`** disappears from the screen and this file gets uploaded to the hosting server, in general, and to the folder **`ourbusinesswebsite.com`**, in particular. Notice the dark bar that appears at the bottom of **`File Upload`** page as shown in *Figure 9.11*:

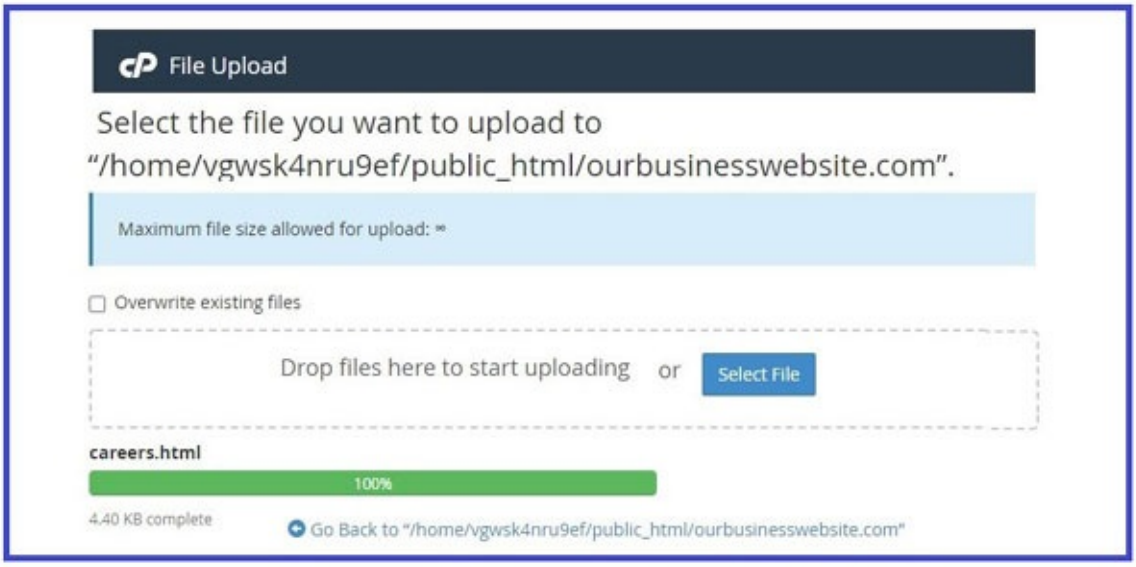

Figure 9.11: *The file careers.html is uploaded successfully as indicated by the dark bar at the bottom on which the text 100% is printed.*

16. Now, repeat this procedure (stated in steps 11, 12, and 13) for the remaining five files. When you do this, you will see six dark bars at the bottom of the **`File Upload`** page as shown in *Figure 9.12* indicating that all these six files are uploaded successfully to the folder **`ourbusinesswebsite.com`**:

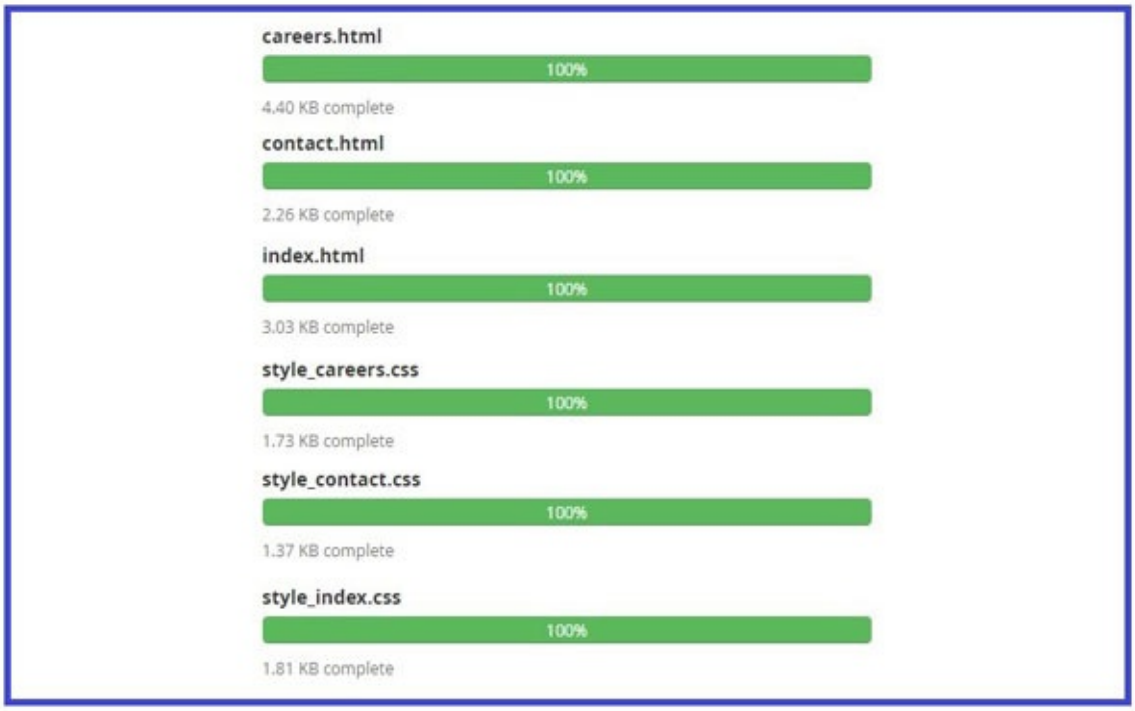

Figure 9.12: *Now, all the six files are uploaded to the folder ourbusinesswebsite.com as indicated by the six dark bars on the File Upload page as shown in this figure.*

17. Now, click on the close-window-button on the tab of the **`File Upload`** page (see *Figure 9.13*) and this page shuts down:

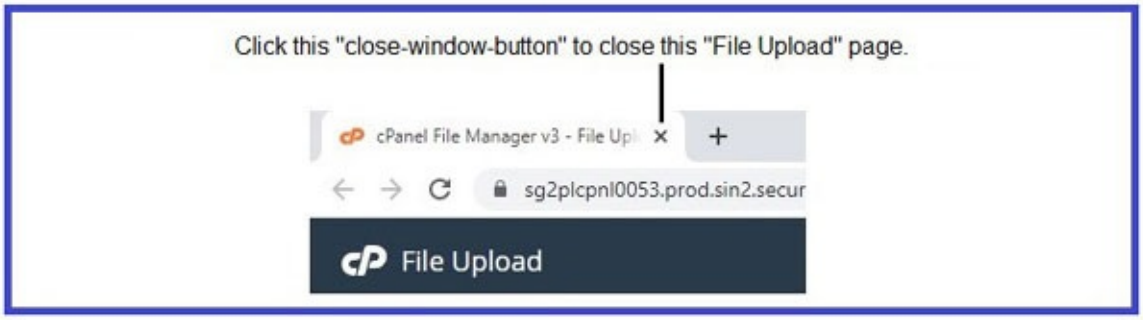

Figure 9.13: *Click on the close-window-button on the tab of this page (that is, the File Upload page) to close this page. Now, the File Manager page comes to the front as shown in Figure 9.14.*

18. Now, the **`File Manager`** page comes to the front as shown in *Figure 9.14*:

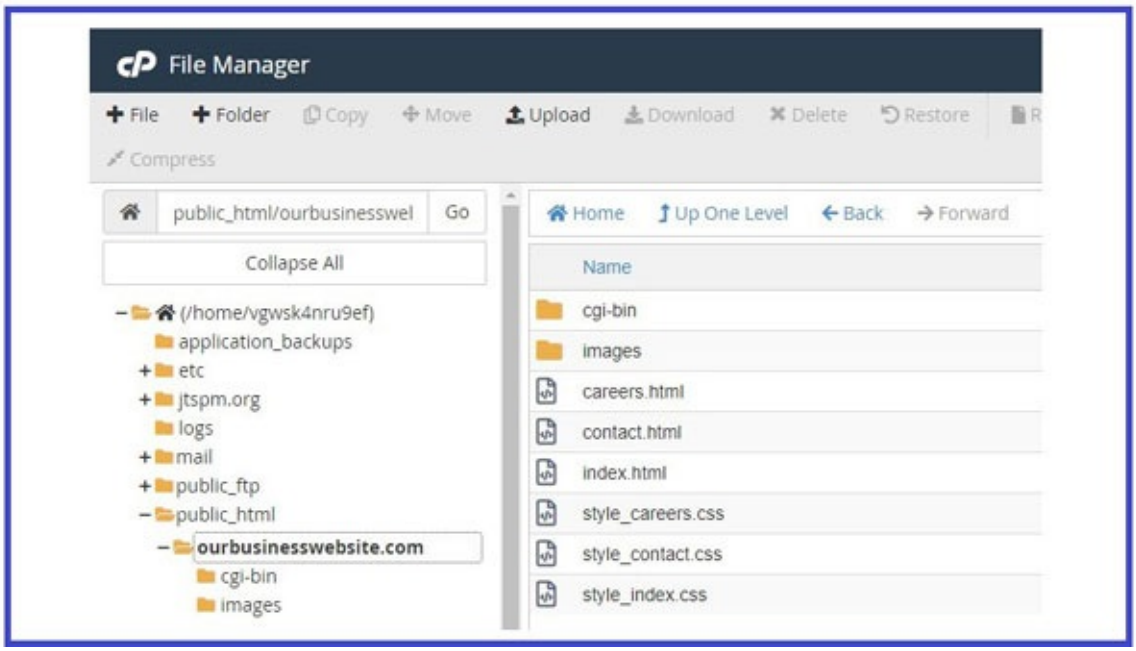

Figure 9.14: *File Manager page. Click on the folder name ourbusinesswebsite.com in the left pane and the names of the uploaded files are displayed in the right pane.*

19. On the **`File Manager`** page, click on the folder name **`ourbusinesswebsite.com`** in the left pane, and the names of the uploaded files are displayed in the right pane (see *Figure 9.14*). Now, double click on the folder name **`images`** in the right pane, and this folder becomes the current folder as shown in *Figure 9.15*:

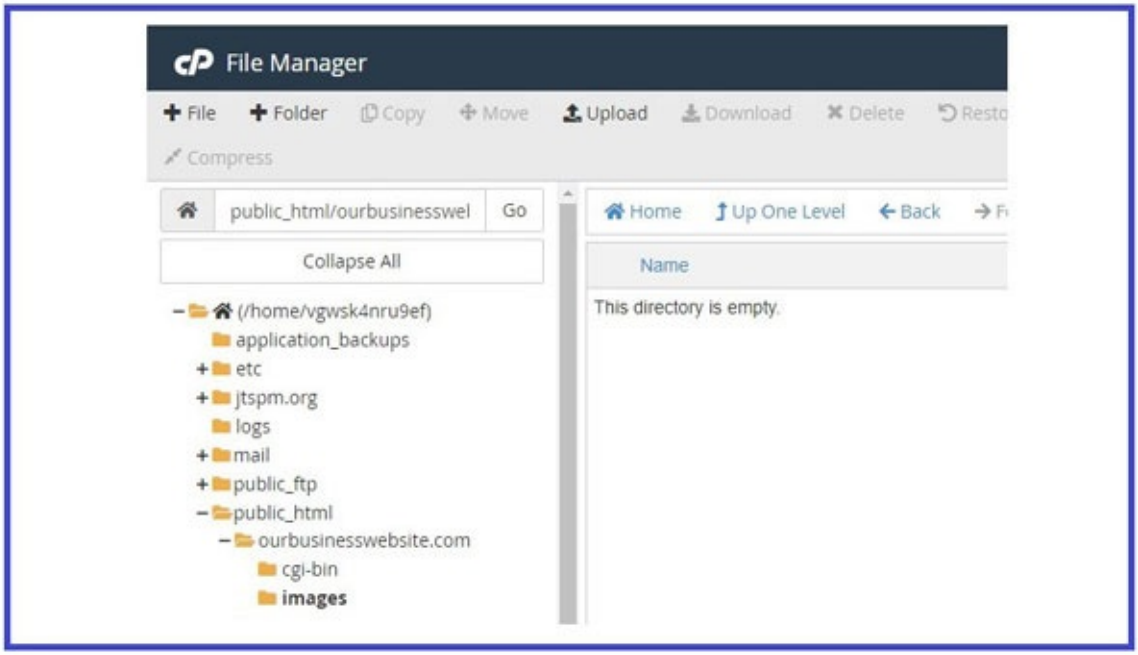

Figure 9.15: *File Manager page. The folder images are the current folder and its contents are displayed in the right pane. As the folder images is empty, nothing is listed in the right pane.*

20. Ensure that the folder **`images`** are the current folder, and then click on the menu-item **`Upload`** in the menu bar located at the top of this page and the page **`File Upload`** appears on the screen. Click on the **`File Select`** button on this page (*Figure 9.11*) and the dialog box **`Open`** pops up on the screen:

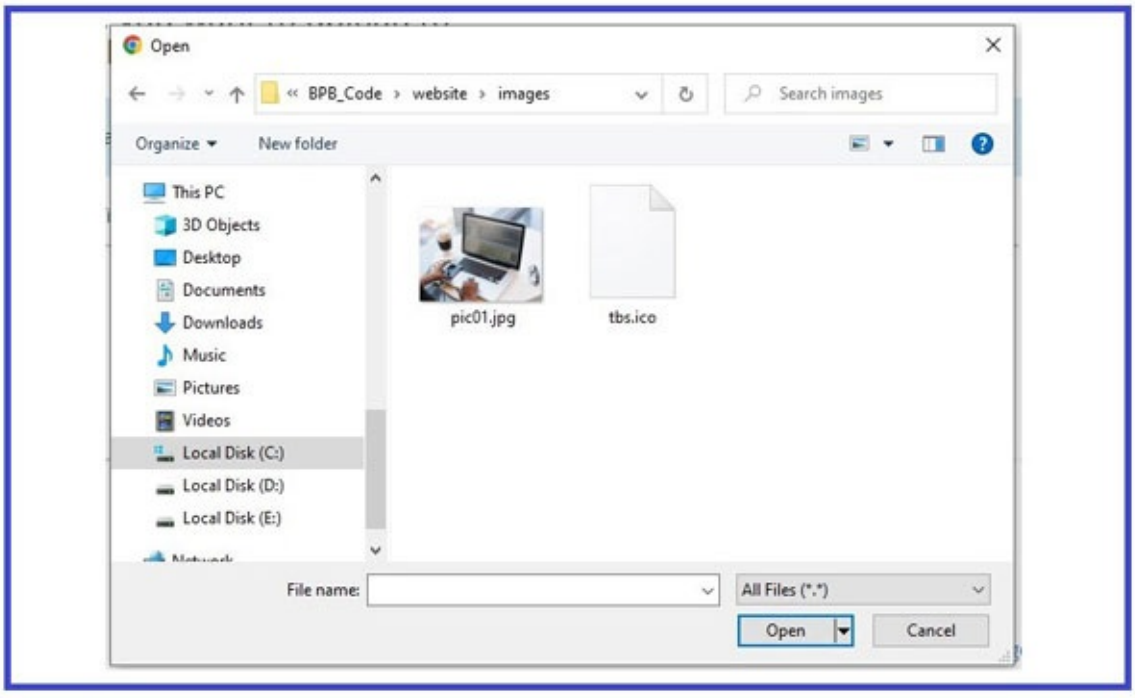

Figure 9.16: *Dialog box Open. Folder C:\website\images is made to be current folder in this dialog-box and hence its contents are displayed in the pane. Double click on the filename pic01.jpg and this file gets uploaded to the folder images.*

21. In the dialog box **`Open`**, set the folder **`C:\website\images`** to be the current folder and the contents of this folder (which are nothing but two image files) are then displayed in the pane of this dialog box as shown in *Figure 9.17*. Double click on the filename **`pic01.jpg`** displayed in this pane. Now, this dialog box disappears from the screen and the file **`pic01.jpg`** is uploaded to the folder **`images`**. Repeat this procedure for the image file **`tbs.ico`** also and upload this file to

the folder **images**. Now, the task of the **File Upload** page is complete. Let us shut down this page. Click on the close-window-button on the tab of the **File Upload** page (see *Figure 9.13*) and this page shuts down:

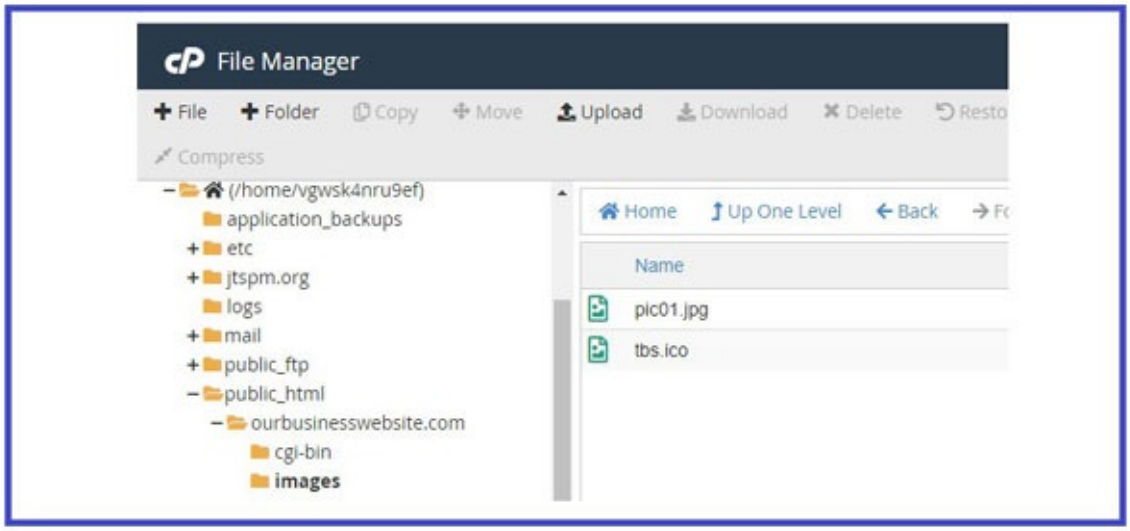

Figure 9.17: *"File Manager" page. Click on the folder name "images" in the left pane and its contents will be displayed in the right pane as shown here. Now, the task of uploading the files is complete.*

22. Now, the **File Manager** page comes to the front. Click on the folder name **images** displayed in the left pane and the right pane shows the names of two image files we just uploaded as shown in *Figure 9.17*. Now, the task of uploading the files is complete. Click on the close-window-button on the tab of the **File Manager** page (see *Figure 9.13*) and this page also shuts down.
23. Now, log out of your account. You have successfully uploaded all the files of the website project to the hosting server.

The task of uploading the files to the hosting server is fully complete but still, you cannot view this website online because the domain name **ourbusinesswebsite.com** is not actually bought by us. We have just mimicked the buying procedure. Buying and then maintaining the domain name for many years is a costly affair therefore we have not bought this domain name.

However, it is essential that you should be able to view this website online. In order to make this website visible online, we will place this website under another website that is already online. This website is **www.webilog.in**. We will create the folder **tbs** in the root folder of this website and then upload all the files to it. The procedure for uploading the files is almost the same as discussed in this subsection. Then, we will be able to view this website online at the following URL:
www.webilog.in/tbs/

We will do this in the next subsection.

Uploading the files to the URL www.webilog.in/tbs/

Let us upload the website project files to URL: **www.webilog.in/tbs/**. Follow the given steps:

1. Log on to your account on the website of your domain name registrar (or hosting provider, as both parties are same).
2. Go to the **My Products** page by clicking on the appropriate link.
3. Find the section **Web Hosting** (see *Figure 9.1*). Find the **Economy Linux Hosting** you have

bought. I have bought two hosting plans:

- **Economy Linux Hosting with cPanel** (for a single website)
- **Delux Linux Hosting with cPanel** (for multiple websites)

4. Click on the **Manage** button in front of **Economy Linux Hosting with cPanel** because the website **www.webilog.in** is stored in this hosting plan. Now, the Dashboard page opens with the text **webilog.in** on the top, left corner of the Dashboard page. Click on the link **File Manager** in the top row on this page.
5. Now, the **File Manager** page opens which looks something as shown in *Figure 9.4*.
6. In the **File Manager** page, double click on the folder name **public_html** displayed in the left pane and the contents of this folder and then displayed in the right pane. This folder contains the files of the website **www.webilog.in**. As this hosting plan is meant for a single website, the folder with the name **webilog.in** is not created. Instead, all the files of this website are directly placed in the folder **public_html**.
7. Here, we are required to create the folder **tbs** that will reside in the folder **public_html**. Ensure that the folder **public_html** is the current folder and then click on the menu item **Folder** and the dialog box **New Folder** pops up on the screen (*Figure 9.7*). Type the **tbs** text in the edit-box provided and click on the button **Create New Folder**. Now, this dialog box disappears from the screen and the folder name **tbs** appears in the right pane as well as the left pane.
8. Double click the folder name **tbs** displayed in the right pane and the folder **tbs** becomes the current folder and its contents are listed in the right pane. As folder “tbs” is empty, nothing is listed in the right pane.
9. Now, repeat all the steps from step no. 9 to step no. 20 stated in the previous subsection in order to upload the website project files to the folder **tbs**. The only difference between the previous subsection and this subsection is that in the previous subsection, you uploaded the files to the folder **ourbusinesswebsite.com** and in this subsection, you will upload these files to the folder **tbs**:

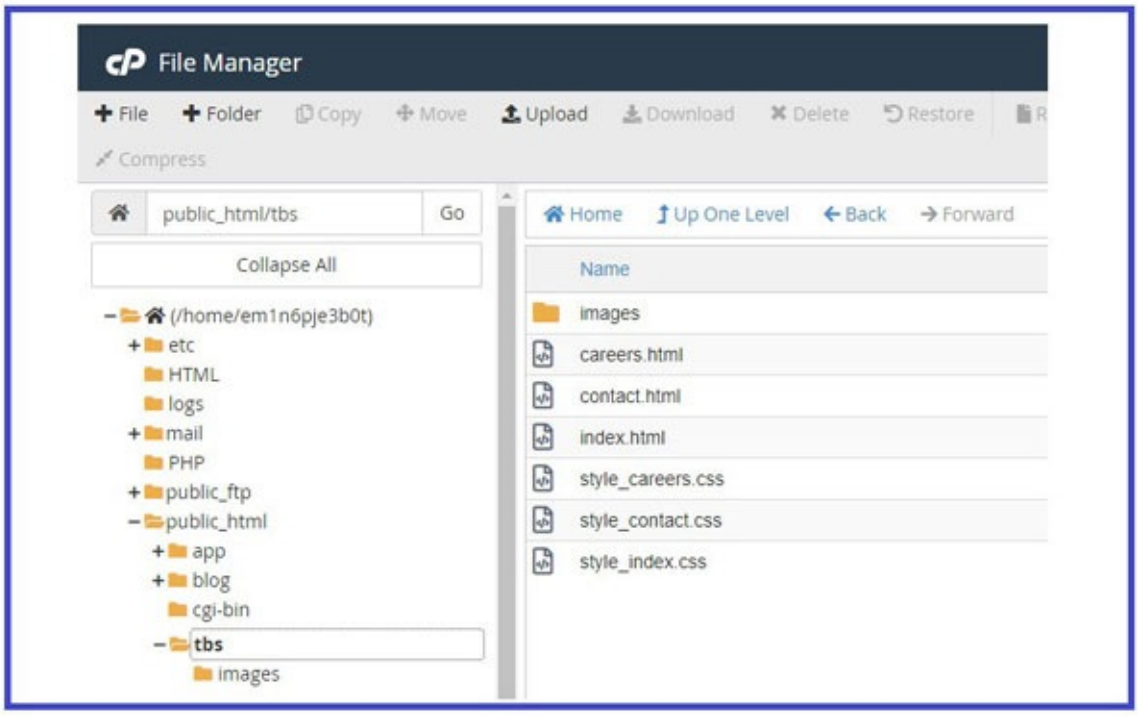

Figure 9.18: *File Manager page. Folder tbs is the current folder and its contents are displayed in the right pane. Also, two image files are placed in the folder images. Our website is now truly online.*

10. Before shutting the "File Manager" page (in step 20) ensure that the folder **`tbs`** contains all six files (3 HTML files and 3 CSS files) and folder **`images`**. Also, the folder **`image`** should consist of two image files as shown in *Figure 9.18*.
11. Believe me. Our website project is now truly online.
12. Log out of your account.

Our website is now online

Our website is now online. Type the following text in the address bar of your web browser:

www.webilog.in/tbs/

Strike the Enter key. Now, the home page of our website project gets displayed on the screen as shown in *Figure 9.19*:

Figure 9.19: *Our website is now truly online. The home page of our website is now displayed in the browser window. Also, access the Careers page and Contact page by clicking on the respective menu items and viewing these pages carefully.*

Website making business

Website making is a multi-billion-dollar business. However, in order to make good amount of money in this business, you are required to learn many things apart from HTML/CSS. See the list of books and articles given below in the section *Further Readings/References*. To begin with, do a job in some website-making company for a couple of years, learn the ropes, and then start your company.

Conclusion

In this chapter, you learned to buy a domain name from an authorized registrar, buy a hosting plan from a hosting service provider, link the domain name to your hosting plan using cPanel, upload the files to the

hosting server using cPanel, upload the files to the domain **`ourbusinesswebsite.com`**, upload the files to the URL **www.webilog.in/tbs/**. Finally, you learned to make your website online on the Internet.

Further readings/references

- *Running a Web Design Business from Home*, by Rob Cubbon, Createspace Independent Pub, 2013.
- *JavaScript: The Complete Reference*, by Thomas Powell, et al, 3/e, McGraw Hill, 2001.
- *PHP: The Complete Reference*, by Steven Holzner, McGraw Hill, 2017.
- *MySQL: The Complete Reference*, by Vikram Vaswani, McGraw Hill, 2017.
- *Python: The Complete Reference*, by Martin Brown, McGraw Hill, 2018.
- *Search Engine Optimization, An Hour a Day*, by Jennifer GRappone, 3/e, Wiley, 2011.
- *Web Security for Developers: Real Threats, Practical Defense*, by Malcolm McDonald, No Starch Press, 2020.
- https://elementor.com/blog/start-web-design-business/
- https://howtostartanllc.com/business-ideas/web-development
- https://techjury.net/blog/website-design-industry-statistics/

Index

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

R

S

T

U

V

W