HTML For Novices
By Novices

Creating Web Pages with Ease

Mike Abelar
For my parents and my friends, Ricky and Tiffany
# Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction</td>
<td>4</td>
</tr>
<tr>
<td>Chapter 1</td>
<td>7</td>
</tr>
<tr>
<td>Chapter 2</td>
<td>16</td>
</tr>
<tr>
<td>Chapter 3</td>
<td>25</td>
</tr>
<tr>
<td>Chapter 4</td>
<td>32</td>
</tr>
<tr>
<td>Chapter 5</td>
<td>37</td>
</tr>
<tr>
<td>Chapter 6</td>
<td>47</td>
</tr>
<tr>
<td>Chapter 7</td>
<td>52</td>
</tr>
<tr>
<td>Chapter 8</td>
<td>58</td>
</tr>
<tr>
<td>Conclusion</td>
<td>67</td>
</tr>
</tbody>
</table>
Introduction

Hello future web programmer! Welcome to the wonderful world of the internet! In this book, you are not only going to learn how to create websites, but also how to think and problem solve like a programmer. As you have read the title, I consider myself to be a novice. Even more so, I am a fifteen year old teenager in high school. This of course raises the questions: “What am I doing here?” and “Why would I ever want to learn programming from a teenager who calls himself a novice?” Those are all essential questions that you may want to be answered before you move on. The point of this book series is to explain programming from my knowledge. I get programming, I self-taught myself an array of languages and I know what works and what does not. I know what materials and methods helped completely code-illiterate people like me from the start. If I were to wait to write this book series, I would have simply learned too much to be the best programming instructor I can be. If you read this series, I will share the knowledge that took me months to try and understand in only a matter of pages. If anything is to be taken from situation, it is that you should learn from me because the knowledge is still fresh in my mind and I am ready to share with you what works and what does not for learning web programming. Now as a fifteen year old, I get that long introductions are boring; let’s jump into programing websites!

What is the Internet?

Before we start programming websites, we first need to understand how the internet operates. Now, here is a boring definition of the internet from Wikipedia: “The Internet is a global system of interconnected computer networks that use the standard Internet protocol suite (TCP/IP) to link several billion devices
worldwide.” Confusing right? Okay, let’s try to formulate a definition of the internet based on some basic facts that you may or may not know about the internet.

First, we know that when we are using the internet, we are on a web browser. Whether it be Google Chrome, Mozilla Firefox, Internet Explorer, Safari, Opera or any other type of browser, we use that browser to navigate on the internet. With this information, we can make a basic definition of the internet: “A global network in which devices with internet browsers connect to.”

However, we can add to this definition. What about the devices and hardware that do not use an internet browser yet still seem to be connected to the internet. Take cell phones for example, they don’t necessarily always use a web browser, yet you still get notifications from Facebook and other internet-based platforms for example. If we include this set of devices, we are now talking billions of collective devices which are connected to the internet. Now we can expand out definition to: “A global network in which billions of devices are connected to for information.” In this case, I added information because all of these connected devices are looking to display and get data. For example, that Facebook notification on your phone is a result of your phone being connected to the internet for that information.

Finally, there is one more component which I would like to add to our working definition, and that is a web server. A web server is simply a large machine which “contains” the internet. Think of it like this: say I want to go onto Google, what does my device do to get the Google page when I type the URL into my browser? First, the device will connect to the internet to find where the Google website exists. Once it finds its address (or IP address), it then goes to the web server containing that address and requests the Google website be delivered to the device. Then, the contents of the Google website are transferred to the device requesting Google.

Without web servers, or computers that host the websites of the internet, we would not be able to get our websites and data we want. Hence, with this in mind, our final definition turns out to be: “A global network of connected computers and systems in which billions of devices are connected to for information.” I added the part about web servers as they are the connected systems of the internet. In addition, if you look back on that Wikipedia definition, you will notice that you can now understand it because we have now briefly covered everything including the IP address.

Enough about that, let’s start programming!
Code for This Book
Can be found at https://github.com/mike2151/HTML-for-novices-by-novices
Just look to the right side of the website and click “Download Zip.”
Chapter 1: Structure of HTML

Finally! We are now ready to start programming such great websites that will amaze everyone right? Well, not quite, just like you need to learn the structure of a foreign language before you begin to speak it, we must do the same. The structure of HTML can summarized in one word: tags. What do I mean? Let us think of this analogy: Say you are at a supermarket and as you walk around, you notice the aisles and the content in the aisles. You look at the frozen food aisle, what would you expect to find in this aisle? Frozen food of course. Let's create an example HTML tag for frozen food: <frozen food>TV dinner</frozen food>

Let's take this tag part by part:
- <frozen food> - This tag indicates that we are talking about frozen food. It is like the start of the aisle.
- TV dinner - This part indicates what's in the aisle, in this case a TV dinner is in the frozen food aisle.
- </frozen food> This tag indicates we are stopping talking about frozen food. It indicates the end of the aisle.

Another example:
<bookshelf> HTML For Novices By Novices</bookshelf>

Here we have a bookshelf tag which describes that we will be working with bookshelves. Inside this given bookshelf tag, we have this book. This book will now be identified as being in this bookshelf.

In summary, we can think of HTML tags as the following: <start tag>Content in the tag</end tag>

**Remember:** The end tag must always have a “/“ following the “<“.

Are you confused by my analogy? How about this, let’s just jump into some actual code! Don’t worry, by doing this some more, you will understand it. Open your choice of text editor (Windows: Notepad or Mac: TextEdit) and type the following:
<p>This is my first website!</p>

Then, save the file as “first.html”. Make sure that you include the “.html” extension when you save your file. If you do not, then the website will not work. Then, open with a web browser by right clicking on the file and clicking “Open with…” then select the web browser of your choice. You should see the following screen:

You can find this code as 1.1.html from the downloaded code for this book.

Congratulations! You have made your first website. You can now tell all of your friends that you were up all night making websites. You will seem like such a cool person. Now, you might be wondering “I can now make websites, the book is over right?” Wrong. You still need to learn more about the ways of the web programming. Now let’s take the code from above and break it down:

The first part of the line of code: “<p>” means that we are using a p tag which stands for paragraph. Whenever you see the p tag, that means you are going to be writing text on the web page. Paragraph stands for text on the webpage. In the inside of the tags, there is the text we want to write. As you could see from the web browser, whatever was in the p tags was written to the web page. Finally, we have closing p tag. This tells the computer that we are going to stop writing text to
the webpage. In conclusion, if you want to write something to the webpage use this basic formula:

<p>Thing you want to write here</p>

And that’s it! Now of course no website has just one sentence. We need to write paragraphs in order to create an actual website. Using the <p>, let us try to create 3 paragraphs on the website. For this example I will choose to use these three random paragraphs I found online (Don’t worry if you do not understand them; the paragraphs are written in Latin):


Based on what you have learned so far, you may say “just put the three paragraphs inside the p tag.” Well, let’s try it and see what issues/problems we will encounter. It is best to experiment with the code to see for example, if paragraphs will be properly displayed. This is the entire point of programming: to try and solve problems you will encounter. Of course, as you advance as a web
programmer, problems are going to be a lot harder than trying to write paragraphs to a webpage. However, you need to try new approaches and use what you know to become a great programmer who is able to figure how to program whatever you want. Here is the proposed code:

```html

Ut gravida magna sed tortor posuere…
</p>
```

Here is what happens when we save the code to the html file and then open that html file with our web browser:

Well, that is not what we expected. We have hit the enter key to separate the paragraphs in the code but why doesn’t it show up in the webpage? To tackle this problem we need to figure out how to insert line breaks into code so that it mirrors on the website. Now, I could tell you how to code for line breaks but that is not the best way to learn. Let me explain. A great programmer is also a great problem solver. A great programmer needs to be able to locate the answer efficiently by themselves. If I told you everything about the HTML language in this book, it would be a very, very long book. Instead, why rewrite what has been stated countless times on the internet? That is correct. I am going to show you how to
solve a problem using the internet. Although for most of the book I am going to teach you the HTML, you should still pick up here on how to solve problems using the internet. In the end, no one can remember everything about programming. Using the internet is a great tool to quickly find what you are looking for to get the job done.

The algorithm to this process is simple: just Google what you want to know. In this case we want to know: “how to create a line break in the HTML programming language.” Note: It is important to specify the programming language when looking online for an answer. Let us type that exactly into Google then:

There we are! Now, let’s click on the first link and learn about the “HTML br tag” (This is what we are looking for by the way). Let’s now look at the usage and definition of this line break:
“The <br> tag inserts a single line break.” Excellent. If you look above to where it says “Example” click on the green button: “Try It Yourself.” You will be brought to a screen with the example code on the left and the actual website rendered on the right. Do not worry about any tags, I will mention those in later chapters. Anyway, notice where the <br> tag is placed and notice to the right how a new line is formed. With this information in mind let us apply the <br> tag to our code:


We still do unfortunately have a problem. We do not have our new paragraphs indented! We have the line breaks, but if we were to really amaze everyone with our website paragraphs, we need to at least indent them. Now I am going to be honest with you and admit I totally forgot how to indent paragraphs. But no fear, we have the internet! If we want to have indents for our paragraphs, let us just plug this into the Google search bar: “how to indent paragraphs in HTML.” Again, this process is very straightforward: just type what you want to do followed by the language.

The first result looks promising so let’s click on that. For me, my first result was the same website we visited earlier (W3 Schools):

![W3 Schools](image)

This seems to be a problem. We have no idea what CSS is so we cannot incorporate this into our HTML code. We will have to work with our problem solving skills by searching the internet again. By the way, CSS is another web programming language that compliments HTML by providing methods for styling...
and formatting HTML documents. I will barely cover CSS in this book. In addition, whenever you need to look something up when programming, the first search might not be the best as you can see, so I am going to walk you through searching again.

Just for a moment, let us look at what the definition of a tab: a tab usually consists of 8 spaces. So, instead of searching how to make a tab, we could just create 8 individual spaces. After looking online how to make a space in HTML, I find that this string of characters creates a space in HTML: &nbsp; . Hence, all I have to do is paste 8 of these consecutively at the start of my paragraph and then I have a tab. Updated Code (including indents and line breaks):

```html
<br>
&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;&lt;br&gt;
<br>
&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;&lt;br&gt;
</p>

This code can be found at 1.2.html.

In summary, in this chapter we learned about the structure of the HTML language also known as tags. We learned how to form tags and write information to the web browser. Also, we talked about the <p> tag and being able to write
sentences to a website. In addition, we talked about creating new paragraphs. Finally, we touched on something we will do frequently in this book: problem solving. In Chapter 2, we will touch on using more tags in HTML.
Chapter 2: More Text Tags

Now of course there is more than just the <p> tag in HTML. When you look at websites, you can see that there is text varying in both size and shape. In this chapter, we look to understand how to incorporate more text-based tags into our websites.

Before we begin, let’s just remind ourselves what kind of websites we are able to currently make. Based on what we have learned so far, the best we can do is a website like this:

As you can see, the website looks very boring and dull. Since our website is filled with only text so far, we should focus on making the text look attractive. The first issue I want to tackle is the lack of headers. A header is defined to be just a bigger, more noticeable piece of text that helps the user sort through the website with ease. To make a header in HTML, we have to use the <h1> tag. Hence, an example header would be:

<h1>Example Header</h1>

Now if we put this in our html file and load it up on the website, we can see that the text inside the header tag is a lot bigger and bolder. With that in mind, let’s try to create an html file using headers to distinguish the different sections of the text. In addition, we should include sample text and use <br> tags to separate all the information out neatly. You can put whatever text you would like in the new sample html file. However, if done correctly, the html file tested on the website should look something like this:
My Website

Welcome to my fantastic website! As you can see here, here is some sample text on this html file. In fact, you can even create another header like this:

Another Header

This is yet another sample sentence. Here are some random words: Hi, Yo, Hot Dog, Couch, Table, Mat, Chair.

Random Words:

Window, Grass, Beach, Tree, Horse, Shelf, Market. You get the point.

And my html code is (can be found as 2.1.html):

```html
<h1>My Website</h1>
<br>
<p>Welcome to my fantastic website! As you can see here, here is some sample text on this html file. In fact, you can even create another header like this:
</p>
<br>
<h1>Another Header</h1>
<br>
<p>This is yet another sample sentence. Here are some random words: Hi, Yo, Hot Dog, Couch, Table, Mat, Chair.</p>
<br>
<h1>Random Words</h1>
<br>
<p>Window, Grass, Beach, Tree, Horse, Shelf, Market. You get the point.</p>
```

Let us break down this code. We first start out with the `<h1>` tag which stands for header one tag. As you can see on the website, the text is bolded and appears much larger. I then add a `<br>` tag and then a `<p>` tag with sample text. I then essentially repeat the code for the rest of the html file.

Moving onto the text tag, or should I say the next **five** tags:

- `<h2>` (heading 2)
- `<h3>` (heading 3)
<h4> (heading 4)
<h5> (heading 5)
<h6> (heading 6)

These tags serve to provide more headings; however, as the number increases in the tag, the emphasis on the content inside decreases. I encourage you to try these tags out in an HTML page to experiment what each does.

Now that we have the idea of tags down, I want to move quickly along with many other text tags. The next tag is the <b> tag. The <b> or bold tag is to make any content inside the tag bold. It is that simple. Let us look an example:

My bold text example

Bold Text!

Code for the website (2.2.html):

<h1>My bold text example</h1>
<br>
<p><b>Bold Text!</b></p>

As you can see, the text is between the b tags and it corresponds to the bolded text on the webpage.

Note about embedded tags: Moving forward, I am going to be including a lot of HTML code examples like the one above: “<p><b>Bold Text!</b></p>.” You may start to see consecutive tags in a row. This is called tag nesting and whatever content is between two tags, acquires the tag’s attributes. Here is an example that will clarify the concept:
Take the line of code: `<p>Hello <b>World!</b></p>`

Since the entire line of code is within the `p` tags, the content is considered to be paragraph content. However, the “World!” is bolded because it is the only world within the `b` tags. The “Hello” is just paragraph and not bolded because it is not surrounded by any other tag. If you are still confused on this topic of nesting tags and content, don’t worry; we will be going over it more as we proceed through the book.

Onto the next text tag: the italics tag. It is denoted by the `<i>` tag. Guess what? It works just like the `<b>` tag but with italics. I won’t include an example this time because you guys get the idea.

The next tags is going to be the `<sup>` and `<sub>` tags. These tags stand for superscript and subscript respectively. Here is a review of both: superscript is when you want to express an exponent for maybe a math website. A subscript is used also in mathematics and sciences for distinguishing between variables. However, as a web developer, unless you are creating a math or science website, I personally would not worry about these; just know that they are there for you to use. For use, just put the tags around any content you want to be an exponent or subscript.

Are you ready for more text tags? Me too. And do not worry, I will be doing a grand review of all of these tags at the end of the chapter. The next tag is a horizontal line break. It is used to separate content in a webpage. The horizontal line break is literally just a horizontal line like so:

```
content
```

```
More Content
```

You use the `<hr>` tag to create this horizontal line. It works just like the break line tag! Remember, you can choose not to close off the horizontal line break tag.
The next tag is more of a placeholder for quotation marks: it is the quotes tag represented as <q>. “Something like this” would be written as <q>Something like this</q>.

The underline tag is a common one. If you wish to underline a selection of text or content, just place that content between <u></u> and you will be good.

The final tag which I will go over is the strikethrough tag. This is especially helpful if you are creating a shopping website in which you will need to display marking down a price to indicate a special sale that is going on. This tag is denoted as <del> and </del>

Well that sure was a lot of tags! Guess what the best part is? You don’t need to burden yourself by memorizing all of these HTML text tags. As a programmer, you have access to the internet! Hence, if you forget how to bold text in HTML, just look it up! The power of Google is amazing and can easily replace this entire chapter with just a few Google searches. With that in mind, I did not include ALL text HTML tags like emphasis, but I did include the main ones you will use. If you need another text tag that was not mentioned, feel free to just look it up! I already said this, but it is so important: being able to find an answer on your own is super valuable!

A quick note on white space:

You have encountered the white space problem in your coding. The white space problem is defined as the following:
<p>Hello, my name is Mike</p>
However, in HTML, this becomes:
Hello, my name is Mike

This is clearly not what we wanted. We wanted a whole lot of spaces between “Mike” and “is”. To counter this, we need to use the space escape character. Essentially, consider an escape character as a series of letters and numbers that act as a space. Let me show you an example:

<p>Hello, my name is &nbsp;&nbsp;&nbsp;Mike</p> becomes
Hello, my name is Mike
Every time we use “&nbsp;”, we generate a space in the website text. You will also encounter the same problem when trying to type out the < and > characters. To type these out just google “less than sign escape character,” you will then get a similar code above to type a less than sign in a webpage. Reminder: you can’t type less than sign because the browser will think it’s the start of a tag. Don’t confuse web browsers, bad things happen.

Instead of giving a summary for each chapter, I am now going to create a chapter website. A chapter website is essentially a website which has HTML code of everything we have learned in the chapter.

Chapter Website:

**Mike's Shopping Website**

**The Best Way To Shop**

Who are we?

*Mike's Shopping Website* is the *best* way to buy everything you need in one click!

What can I buy?

On our website, we offer 100 unique different items

Customer Reviews

"This is the best store ever!" - valued customer

Current Sales

Here: $50

Here is the corresponding source code:

<h1>Mike's Shopping Website</h1>

<h2>The Best Way To Shop</h2>

<br />
<br />
<br />

<h3>Who are we?</h3>

<p><b><u>Mike's Shopping Website</u></b> is the <i>best</i> way to buy everything you need in one click!</p>
What can I buy?

On our website, we offer 10^5 unique different items

Customer Reviews

This is the best store ever! - valued customer

Current Sales

Hat: $5 $2

This code can be downloaded as 2.4.html

Note about IDES: So far, I’m assuming you guys are using like a basic text editor and then launching the saved HTML file to a webpage. This, after a while, becomes tedious and annoying. To save time and improve programming time, you should get an integrated development environment. This is just a software to help you program quickly. These programs have many benefits such as syntax highlighting and error catching. In my honest opinion, there is absolutely no reason to be using the old way of coding HTML with this software out there. Here are a list of free HTML code editors that I recommend to improve your experience:


Sublime Text: http://www.sublimetext.com/

Microsoft Web Matrix (Windows only): http://www.microsoft.com/web/webmatrix/

Aptana Studio: http://www.aptana.com/

Komodo Edit: http://komodoide.com/komodo-edit/

Brackets: http://brackets.io/

Another end of chapter note: So far in the book, we have just been placing HTML code into a file and saving it. However, there is a formal way of writing HTML and an official structure. That structure is given by this template:

```html
<!DOCTYPE HTML>
<html>
<head>
<title>TITLE OF WEBSITE</title>
</head>
<body>

</body>
</html>
```

For every HTML document that you create, you should paste this code first. This code is the template for every HTML website. Let us go through it to understand what it all means and why you don’t have to use it all the time.

`<!DOCTYPE HTML>`: This just tells the browser we are working with HTML. There are other languages and file formats that can be loaded up by a web browser. This just confirms that we are using HTML. However, most of the time, browsers can just tell anyway by the file extension.

`<html>`…`</html>`: This tells the browser that inside these brackets is HTML. You don’t need to include these tags unless you are using some other web programming language which I will not cover in this book.

`<head>`…`</head>`: This section identifies the head section of the HTML page. The head section is a place for you to put all information about the website in it. For example, the name of the website and even a description. More of this will be covered in a later chapter. You should use this if you plan to put information about your website in your HTML document.

`<title>`…`</title>`: This is the only tag that we will put in the head section for now. The title tag just gives the browser the title of the website. You put the title of the website in between the tags.
<body>…</body>: Most of everything on your website will go into the body tag. Everything that you want to show up on the website goes inside the body tag. What we have been writing so far with the tags all goes into the body tag. All of the site’s contents goes in here. All of my h and p tags will go in here because I would want them to show up. Here is an example of how to use the body tag:

<body>

<b><u>Mike's Shopping Website</u></b> is the <i>best</i> way to buy everything you need in one click!</p>
<br />
<h3>What can I buy?</h3>
<p>On our website, we offer 10<sup>5</sup> unique different items</p>
<br />
<hr />
<h3>Customer Reviews</h3>
<p><q>This is the best store ever!</q> - valued customer</p>
<br />
<hr />
<h3>Current &nbsp;&nbsp;&nbsp;Sales</h3>
<br />
<p>Hat: <del>$5</del> $2</p>

</body>

This gets rendered the exact same way as the code earlier in the chapter. You should include body tags because it helps you identify what is being displayed to the browser.

That is a quick crash course of formal HTML structure. You should use it whenever possible to get into the habit of using it. Enough with the end of chapter notes, let’s move to Chapter 3.
Chapter 3: Links

Now if you ever have used the internet, you have come across links. You know, those blue underlined text phrases that look like this: www.google.com. These are also called hyperlinks in case you were wondering. Anyway, these are important in navigating not only websites, but also the entire internet. In this brief chapter, I am going to show you how to create such hyperlinks to not only link you to other websites in your HTML page but also to link to other pages in your website. Because let’s face it, no website has just one page.

To create links in your HTML website, the formula is easy, just follow the template below:

\[ \text{<a href="YOURLINKHERE">TEXT TO DISPLAY ON WEBSITE</a>} \]

Let’s break it down: we first have our \(<a>\) tag but it is not closed right away? What is going on here? Good question. This is a perfect time to introduce what attributes are in HTML. Each tag in HTML can contain attributes. These attributes help to define the function or appearance of the tag. In this case, we are telling the \(<a>\) tag that it will contain the href (short for hypertext reference) of YOURLINKHERE.

And by the way, in case you were wondering, the \(<a>\) tag stands for anchor tag. The anchor was the original name of the hyperlink back in the early days of the internet.

Another example of an attribute would be something like this:

\[ \text{<p id="my paragraph id">}</p> \]

In this case, we giving the paragraph tag an id attribute of “my paragraph id”. The attribute just helps to define the tag. It is that simple. You can of course have multiple attributes like so:

\[ \text{<a href="www.google.com" id="my anchor id">This link goes to Google</a>} \]

That is pretty much all there is for attributes in HTML. If you are still struggling with the concept or need a list of attributes, I suggest you go to this link here: http://www.w3schools.com/html/html_attributes.asp or just do a Google search.
Now that we have covered what attributes are, let’s move to the content between the <a> tags. “TEXT TO DISPLAY ON WEBSITE” is the text that shows up on the webpage. This concept is best illustrated with a sample website:

**THIS LINK TAKES YOU TO A MAGICALLY PLACE**

Source code (3.1.html):
<a href="https://www.google.com">THIS LINK TAKES YOU TO A MAGICALLY PLACE</a>

As you can see, the webpage displays the text in between the <a> tags. However, if I were to click on the blue link, I would be taken to Google. Just go on any business webpage. If you see a button like “About us”, it is really the “About us” in between the <a> tags and the actual URL as the href attribute.

Again, you can find more information on the topic by searching on Google for HTML hyperlinks or anchor tags.

In the last part of the chapter, I would like to go over how to link to other pages in your website. Before we go over anything with linking to other web pages, you need to first make sure all of your HTML files are in the same directory. Now, what do I mean by that? I mean make sure all of your HTML files are in the same folder. Let me show you an example of how we would link to other webpages with the HTML files all in the same directory:
As you can see, all of my HTML files are in one folder. With that in mind, let us link two HTML pages together. To do this, we are going to use the anchor tag like before. However, this time we are going to replace the URL with the name of the file we want. Let me show you a working example so you can best understand this concept.

Document 1 Source Code (3.21.html):

```html
<h1>Document 1</h1>
<a href="document2.html">Go to document 2</a>
```

Document 2 Source Code (3.22.html):

```html
<h1>Document 2</h1>
<a href="document1.html">Go to document 1</a>
```

Pictures:

**Document 1**

Go to document 2

and
Now that we have the fundamentals down, I just want to go over some additional key elements for the `<a>` tag.

Now say we want to incorporate a lot of links on our website. Based off design experience, it would better if the user could have links that opened in new tabs since the user will be opening a lot of tabs. To do this, we are going to add another attribute to our `<a>` tag like so (Again, you can easily look up this information online if you forget it):

```html
<a href="mypage.html" target="_blank">New Link</a>
```

This target attribute is "_blank" meaning that when the hyperlink is clicked, the new content will open up in a new tab.

Let’s talk more about directories. You may encounter a situation where you have folders inside folders. Say the file is inside a folder in the current directory. To access it you would do something like this with you a tag:

```html
<a href="folder/htmlfile.html">SECOND PAGE</a>
```

And if you want to get a file in the folder above your current folder (if the current directory is a subfolder and you want to navigate out of the current subfolder to access the files above):

```html
<a href="../htmlfile.html">SECOND PAGE</a>
```

Again, the two dots indicate that we will navigate to the folder containing the current folder. From there, we then load up the HTML file.

The final random key concept I would like to go over with links is the idea of linking to multiple sections on your HTML page. For example, say we have a really long HTML page, like a Wikipedia page. A Wikipedia page has all of the information split up into different section headings like “history” and “description”
for example. Is there a way that we can use links to navigate to the sections? The answer is of course. This is a two part step:

Step One: Set up the sections in the HTML code:

```html
<h1 id="introduction">Introduction</h1>
<br>
<h1 id="about">About Us</h1>
<br>
<h1 id="more_info">More Info</h1>
```

Notice here how all I did was just put in the id attribute to define the sections. This allows me to create links that will navigate specifically to the given ID. Let me show you how you to navigate to these newly created sections on the webpage:

```html
<h1>Links to sections of the web page:</h1>
<a href="#introduction">Intro</a>
<a href="#about">About</a>
<a href="#more_info">More Information</a>
```

Notice, like the rest of the examples in this chapter, all I did was use the href attribute. This time however, I inserted a hashtag in front of my desired section. This indicates to the webpage that we are dealing with webpage sections.

In addition, when dealing with the id attribute for tags, DO NOT USE spaces. It will create problems and won’t work. Instead, go for underscores to separate words.

That pretty much concludes everything for links in HTML. If you want to read more about links, here is a general page about them: http://www.w3schools.com/html/html_links.asp. Here is the end of the chapter web page along with the source code:

```html
<h1>LINKS!</h1>
```
Welcome to my wonderful page on links. In case you were wondering, you can look up what links are on this website.

My website is expanding each and every day. Soon we will have our About Page done! However, it is not done yet so don't click on it!

Finally, links offer a great way to navigate a really long page like this!

Section links to help your navigation:

Section One
Section Two
Section Three
Section Four

Section One

Section Two

Section Three
Can be downloaded as 3.3.html

**LINKS!**

Welcome to my wonderful page on links. In case you were wondering, you can look up what links are on this [website](#). My website is expanding each and every day. Soon we will have our [About Page](#) done! However, it is not done yet so don't click on it!

Finally, links offer a great way to navigate a really long page like this!

**Section links to help your navigation:**

- [Section One](#)
- [Section Two](#)
- [Section Three](#)
- [Section Four](#)

**Section One**
Chapter 4: Lists

In this chapter, we are going to go over lists in HTML. Now what exactly are lists? I am sure you have seen them before. Here is an example:

Shopping List:
• Bread
• Water
• Milk

Lists can be useful in sites where you may have to list the benefits of a product or service. This chapter will be a simple and straightforward so let’s jump right into it!

The first thing I am going to show you is how to make a bulleted list like I have done above with my shopping list. To do so, follow this template below:

<ul>
<li>ITEM NAME</li>
<li>ITEM NAME</li>
<li>ITEM NAME</li>
</ul>

That’s it! Done! Very simple, but let me go over it anyway. The <ul> tag stands for unordered list meaning it is just a list with no numbers. This means it is a bullet point list. The <ul> tag indicates that we are starting a list and the end tag indicates we are ending that list. In the middle is going to be the <li> tag which just stands for list item. In the list item tag, we put our desired content we want bulleted. We then end the list item and start a new one. Here is the HTML for my shopping list above:

<p>Shopping List:</p>
<ul>
<li>Bread</li>
<li>Water</li>
<li>Milk</li>
</ul>
If you load up the HTML, you will find it to be identical to the list that appeared in the book.

Now you may be wondering how to implement sub-bullets in HTML. Well if you were, you are in luck. If not, well then sorry because I am about to discuss how to create sub-bullets in HTML. To create sub-bullets, just put another ul tag in place of the li tag and put the desired sub-bullet under that new ul tag. Let me show you an example:

```html
<h1>Sub-Bullets in HTML</h1>
<br/>
<h3>My supreme shopping list:</h3>
<br/>
<ul>
  <li>Water</li>
  <li>Children's Stuff</li>
  <ul>
    <li>Toys</li>
    <li>Candy</li>
  </ul>
  <li>Food</li>
  <ul>
    <li>Bread</li>
    <li>Cheese</li>
  </ul>
</ul>

Code is available as 4.1.html
Sub-Bullets in HTML

My supreme shopping list:

- Water
- Children’s Stuff
  - Toys
  - Candy
- Food
  - Bread
  - Cheese

The final thing about lists I would like to go over is an ordered lists. Ordered lists are also numbered lists. An example of a numbered list is as follows:

Top Players
1. Me
2. You
3. Him

The code to implement this in HTML is really simple. Remember how we dealt with unordered lists before? Just replace the “u” in the <ul> tag with an “o” to create the <ol> tag or the ordered list tag. An example of the list above in HTML is:

<p>Top Players</p>
<ol>
  <li>Me</li>
  <li>You</li>
  <li>Him</li>
</ol>

That is all about lists in HTML. You can’t do much with them until you get into CSS which allows you to customize them and make them look good and appealing. Otherwise, that is how you structurally create lists in HTML. More on
lists can be found here: http://www.w3schools.com/html/html_lists.asp Wow, this was a short chapter.

Since this chapter is really short, I thought I would talk about comments in HTML. Now, what exactly are comments? Comments are lines of code that are totally ignored by the website and do NOT result in any change of the webpage or the HTML. Now why would you want to include lines of code that don’t do anything? The answer is readability. You see, when your HTML files start to reach hundreds or even thousands of lines of code, it will get confusing very quickly. It would be great if you had some sort of sub titles to keep you aware of what is going on. That is exactly what comments in HTML do. They describe the code in English or any language for that manner, to help the reader understand the code. Let me show you an example of a comment:

<!— This code below represents a simple paragraph tag —>

<p>Hello!</p>

If you paste the code below in an HTML file and run it on a web browser, you will be surprised to find that only the “Hello!” shows up. This is because the line above is in comments. That content is ignored by the browser.

To form a comment just put your content between the <!— and the —> (that is two dash marks by the way for each arrow). A comment can span multiple lines like so:

<!—
MULTI
LINE
COMMENT
!
!
—>

Comments are again used to describe code in case you or other coders would want to read it. It is not necessary, but it does make it easier to understand.
How to make cereal

Ingredients:

- Milk
- Cereal

Instructions:

1. Get a bowl and spoon
2. Place cereal in bowl
3. Add milk
4. Enjoy!
Chapter 5: Media, Media, Media!

So far, our web pages have been all text and it has been getting so bland. I mean, what web page just has text? Where are all of the pictures? The videos? The audio clips? The animations? Well look no further, because in this chapter we are going to cover all of those media forms and many more! This will be quite a long chapter, but the concept and coding is going to be easy and straightforward. Let’s just jump into it!

I will first start off with images because they are the easiest. In addition, once we know how to implement images, the rest of the media formats follow the exact same pattern. If you want to incorporate an image, follow this template below:

<img src="FILENAME.jpg" alt="DESCRIPTION OF IMAGE" width="WIDTH" height="HEIGHT" />

Okay, so there is a lot going on here; let’s take it a step at a time. First, the img tag stands for image. The “src” stands for source. In this case, the source of the image file.

Note about source of images:
Just like for links and hrefs, the src attribute works exactly the same. You can either link to an image in your current directory or file system OR on the internet. For the file system, remember to adjust the path as necessary depending on where your file is located in your system of folders. For the internet, you need to have one criteria: make sure your URL ends with an image file extension. For example if I go on Google Images (which I don’t recommend because those images are not licensed to you unless otherwise notified or if you are making the content for educational purposes), I need to make sure my URL ends with the extension. Let me show you an example:
Notice in the first picture how I am viewing the images. If I click on one and copy the URL it will not work. On Google Images, I need to click on the picture and click “View Image.” This will bring me to a page where hopefully the URL will look something like this:


Notice how the URL ends with .jpg. Below are the most popular file types for images in HTML.

Note about image types:
There are many image types which can be used in HTML. I am going to go over the most popular:
JPEG - indicated by the .jpeg or the .jpg extension. JPEG files are best with pictures that involve a lot of color

GIF - indicated by .gif. GIF files are used best with pictures with few color or animations in some cases

PNG - indicated by the .png extension. PNG files are used best with images that involve transparency.

Here is a chart of all the other image types which may be used in HTML: https://en.wikipedia.org/wiki/Comparison_of_web_browsers#Image_format_support

Now that we have covered the source of an image, we will move onto the “alt” attribute. The alt attribute stands for description of the image. You should always fill out the alt attribute because in case someone cannot load the image, they will have access to the description.

The last two attributes are width and height. These attributes simply define how wide and how tall an image will be in pixels (A pixel is single dot of color on a computer screen). You can change the units to be percentages if you wish, but that is more CSS than HTML. For this book, I will only discuss images in terms of pixels.

The final component to our image is “/>”. Now you may be thinking “where did the end tag go?” Well, the “/>” is short for an end tag. Since the image tag does not have any content in between the tags, it is useless to write out the entire </img>. Instead of “…><img>” you can just use “/>”.

That was a lot to take in so let me show you an actual example of our image tag in action (5.1.html):

<!-- Example use of an image in HTML -->

<h1>Wikipedia is an awesome site. Here is their logo:</h1>

Wikipedia is an awesome site. Here is their logo:

![Wikipedia logo](wikipedia.png)

I am not done with images just yet. If you are creating images, it is always best to also include captions. To do this, you will need to create a figure tag and inside a `<figcaption>` tag along with the image. An example is below (5.2.html):

```html
<figure>
  <br>
  <figcaption>The Wikipedia logo above is super cool.</figcaption>
</figure>
```

Wikipedia is an awesome site. Here is their logo:

![Wikipedia logo](wikipedia.png)

The Wikipedia logo above is super cool.

In summary, start out by putting a figure tag and inside an image tag, followed by a break line tag, followed by a fig caption tag. There are of course other attributes for images; just look up “images in HTML” on Google or go to this link for more information if you are interested:

Congratulations! You got the difficult part of the chapter over with! It is now time to flow smoothly over the rest of the media forms. I will start out with video!

The code in HTML for videos is so similar to that of images, that I will just start right off the bat with an example video:

<video src="samplevideo.mp4" />

As you can see, this is exactly like our image tag. The only difference is that we now use an MP4 format. MP4, or MPEG-4 or Moving Picture Experts Group 4, is a video format which may be used in websites. Currently, the three main formats for HTML videos are MP4, WebM, and Ogg video file types. If you plan to use a video locally (meaning not from the web and on your machine), it is best to have a MP4 file. All major browsers do support MP4 and the other formats may not work all of the time.

Note: the directory rules still apply for videos. If you have your videos in a sub or super directory, adjust the path accordingly.

Now, if you try to load this up on an HTML document, you will get the following:

Hey, wait a minute, that does not look like a video. What is going on? It just looks like a still image.

You are exactly correct: it is a still image because it is the thumbnail of the video. We have not added any other attributes to the video. In order to actually play a video, we will need controls, like a play and pause button. While we are at it, we may also want to resize the video to be enormous so people who visit the site will be blown away by the massive video. In fact, let me just add some of the most
popular attributes to this video tag. To do so, we just add some attributes to our previous video tag:

```html
<video src="samplevideo.mp4" poster="thumbnail.jpg" width="1000" height="1000" preload controls loop/>
```

This code can be found as 5.3.html. The video comes from Big Buck Bunny.

Let us break down every attribute. As a reminder, the attributes for HTML, as always can be found online. The “poster” attribute is responsible for the thumbnail picture. This picture is what the video will look like before the play button is clicked. The width and height, we covered this. Now the next attributes are some attributes that we have never seen before. These attributes are expressed by simply having the attribute name there. These attributes are not equal to anything, they are just there to express whether or not the HTML element should possess them.

The first of these attributes is the preload attribute. As the name suggests, this attribute is used to tell the browser to load the video before the user would click the play button. This makes the video play without having to load and buffer in the meantime. The “controls” attribute is used to add the controls to the video.
These include the audio, play and pause buttons. The “loop” attribute is used to allow the video to play over and over automatically once the video ends. You may want to use this if your video is a short animation of sorts. This essentially summarizes the video tag. More information can be found here: http://www.w3schools.com/html/html5_video.asp

Now we have been talking about embedded your own local videos into websites. However, what if you want to embed a YouTube video to improve your website? This process is simple and even recommended because with sites like YouTube, you are not hosting the video. By not hosting the video, you are making your website take up less space. To embed a YouTube video into a website do the following:

Go to your desired YouTube video on YouTube:

This is my programming YouTube channel by the way. I highly recommend you check it out.

Next, right click on the video and select “Copy embed code”
Now you have the HTML code to place in your HTML document. Simply paste the code which is already copied into your HTML document to get the following (5.4.html):

Code: <iframe width="854" height="480"
src="https://www.youtube.com/embed/SOJ1S6Q97C8" frameborder="0"
allowfullscreen></iframe>
Other sites like Vimeo also support embedding videos; however, the steps may be different. A simple web search on how to embed videos from a specific content provider will give you an answer.

Finally, the last of our major media: Audio. Audio is just like video and images. Here is the template to use audio in an HTML page:

```
<audio src="audiofile.mp3" controls autoplay />
```

As usual, we have the src attribute which directs to a local audio file. Note: I recommend using mp3 in websites because it is a heavily supported format. The controls and autoplay are also familiar from the video element. Controls, of course, adds the audio controls which are commonly found on any audio player. Autoplay just plays the audio file automatically once one opens the webpage.

Now that we are done talking about all of this media, here is a chapter summary website which incorporates everything we have gone over:

**Media in HTML**

**My excellent trip summed up in one video:**

![Video Player]

**My personal photo album:**

![Photo Album]

* Taken from wikipedia

**My favorite song I listened to while there:**

![Song Player]

<!-- This file summarizes everything learned in the media chapter -->
<html>
<head>
<title>Media Webpage</title>
</head>
<body>
<h1>Media in HTML</h1>
<br>
<br>
<h2>My excellent trip summed up in one video:</h2>
<video src="parisvideo.mp4" width="100" height="100" controls>
</video>
<br>
<br>
<h2>My personal photo album:</h2>
<figure>
<img src="https://upload.wikimedia.org/wikipedia/commons/8/8a/Paris_vue_d'ensemble_tour_Eiffel.jpg" width="100" height="100" />
<img src="https://upload.wikimedia.org/wikipedia/commons/e/e6/Paris_Night.jpg" width="100" height="100" />
<img src="https://upload.wikimedia.org/wikipedia/commons/0/0b/Eiffel_tower-Paris.jpg" width="100" height="100" />
<figcaption>Taken from wikipedia</figcaption>
</figure>
<br>
<br>
<h2>My favorite song I listened to while there:</h2>
<audio src="parisaudio.mp3" controls />
<br>
<br>
</body>
</html>

File may be downloaded as 5.5.html
Chapter 6 Extra HTML

Now that we are done with media, it is time to go over the extra HTML that you need to know but isn’t really worthy of its own chapter. Once we cover this HTML, we will be able to dive into the final two chapters: forms and tables. This chapter will be a hodgepodge of tags that have nothing to do with each other. Let’s start off with the center tag:

The center tag is used to put content in the center of the webpage. For too long, our content has been on the left side of the page and it has looked unattractive. To use this center tag, just place any html content in the tag and it will be moved to the center (6.1.html):

```html
<center>
<h1>Hello World!</h1>
</center>
```

Hello World!

Another tag that is important is the iframe tag. The iframe tag essentially serves as a different webpage for your html page. Think of it as if you want to have a website inside your website. Let me show you an example with two HTML files and how I can combine them into one HTML document.

First HTML document (6.21.html):

```html
<center>
<h1>This is the first HTML document</h1>

<iframe width="500" height="500" src="6.22.html" />
</center>
```

Second HTML document (6.22.html):

```html
<center>
```

As you can see, the second HTML document is embedded in the first HTML document. Another common use of the iframe is Google Maps. If you want to include a Google Map into your site, you would also use an iframe. An example is like so: <iframe src="https://www.google.com/maps/embed?pb=!1m14!1m12!1m3!1d154585.1782437188!2d-73.94203473487148!3d40.74109594634353!2m3!1f0!2f0!3f0!3m2!1i1024!2i768!4f13.1!5e0!3m2!1sen!2sus!4v1446824835780" width="600" height="450" frameborder="0" style="border:0" allowfullscreen></iframe>

You can get this embed code from Google Maps by going to the menu and clicking “Share or embed code”. From there, you will be given an iframe which you just paste in. Other media forms use an iframe so if you ever encounter one, just be sure to simply paste it in your HTML. Also, note the iframe code, you can spot familiar attributes such as a width and height. You are free to edit these according to your site.

The next tag is the <meta> tag. This tag goes in the head section of your HTML document. Remember, the head section of the HTML document describes the HTML page. The main descriptors of the HTML page are <meta> tags. There are many types of <meta> tags, each describing a key attribute of the HTML page. If you plan on developing a professional website, you will definitely need to use these <meta> tags. Here is an example with several common <meta> tags:
Let’s break down the code. For each `<meta>` tag, there is either a name or a http-equiv attribute and a content attribute. The first attribute, either name or http-equiv defines which `<meta>` attribute is being described. For example, name=“description” tells us that the specific `<meta>` tag will describe the website. The content attribute following is used to tell us what the first attribute is being defined as. For description, the content of the description is in the content attribute.

All `<meta>` tags may be found at this link: https://gist.github.com/kevinSuttle/1997924 and you should use only the ones which describe your site which are the first 10 or so.

Now why should we care about these `<meta>` tags? What purpose does describing the site give if we cannot see the contents of the head section anyway? The answer is simple: search engine optimization. People usually find websites by going onto popular search engines like Google or Bing. For example, if someone is trying to find my baking website they would type in “Mike’s baking website” or something like that. These search engines are able to link the user to my site because I would include `<meta>` tags for keywords and description in order to let the search engine know that my website is described as “Mike’s baking website”. For the keywords I would probably include “Mike, Baking, Site, Website, Food.” You want to include the minimum as possible because too many keywords will make your search rank not as good.
Although there were many random elements in this very short chapter, I am going
to combine them all in this end of the chapter summary website:

<!DOCTYPE html>
<html>
<head>
<title>Other elements in HTML</title>
<meta name="description" content="Website summarizing chapter 6" />
<meta name="keywords" content="website, meta, tags, iframe, center" />
<meta http-equiv="author" content="Mike Abelar" />
</head>

<body>
<!-- This website summarizes what we have learned in the random website about
the other stuff in HTML! -->
<center>
<h1>Come Visit NYC!</h1>
<br>
<br>
<iframe src="https://www.google.com/maps/embed?pb=!1m14!1m12!1m3!
1d154585.1782437188!2d-73.94203473487148!3d40.74109594634353!2m3!1f0!
2f0!3f0!3m2!1i1024!2i768!4f13.1!5e0!3m2!1sen!2sus!4v1446824835780"
width="600" height="450" frameborder="0" style="border:0"
allowfullscreen></iframe>
</center>

</body>
</html>

Available as 6.3.html
Come Visit NYC!
Chapter 7: Tables

In this chapter, we are going to learn how to make tables in HTML. No, not the kind you eat on; the kind of tables used to represent data of course. You would want to include tables if you are making a website for experiments or use tables if you want to talk about sales.

Let us just jump right into the HTML of tables. How to create a simple 2x2 table in HTML:

```html
<!DOCTYPE html>
<html>
<body>
<center>
<table>
<tr>
<td>R1C1</td>
<td>R1C2</td>
</tr>
<tr>
<td>R2C1</td>
<td>R2C2</td>
</tr>
</table>
</center>
</body>
</html>
```

Code available as 7.1.html

Every table is started off with a table tag. Everything you make in the table must go in between the tag. The `<tr>` and `<td>` tags stand for “table row” and “table data” respectively. These tags make up the rows and columns of your table.
Every time you put a `<tr>` tag, any content in between that `<tr>` tag will form a row. Hence, when I put two “td” tags, I created a set of two data values inside that row. This allows us to visualize that there were two columns in the data table. The R and C content in the table is used to indicate which row and which column that data represents. In summary, the more `<tr>` tags you add, the more rows your table has. The more `<td>` tags you have, the wider, or the more columns your table will have.

Note: I know you may be wondering how you can add table borders to make it look like a real table. Unfortunately that is CSS which I am not covering in this book. However, an online search will be able to greatly assist you.

In the majority of tables you have seen, you will notice that all tables have some kind of header. This header helps the reader understand what the data represents. If we just plop data in a table without any labels, the reader will be confused. In order to add a header to a table, you have to incorporate the `<th>` or the table header tag like so:

```html
<center>
<table>
<tr>
<th></th>
<th scope="col">H1</th>
<th scope="col">H2</th>
</tr>
<tr>
<th scope="row">H3</th>
<td>R1C1</td>
<td>R1C2</td>
</tr>
<tr>
<th scope="row">H4</th>
<td>R2C1</td>
<td>R2C2</td>
</tr>
</table>
</center>
```

Code available as 7.2.html
Applying table headers is not as intuitive as you think it might be. Let me walk you through what I did. After learning through this example, you should not have a problem creating table headers for any table. I first of all added a row at the top for the column headers. The first element in the row was an empty <th> tag. This is just a blank space so I can move all of my table headers to the right so they are over my columns. For every table header I create, I must specify whether that header will be for the column or for the row. I specify this exactly by using scope="". With the first row down, I then simply added a table header for the rows at the start of each row. That is it. That is how you add table headers to a row. Now people know what they are looking at in a table!

Our table sure does look small. It would be great if there was a way to make the cells bigger. Introducing: the colspan and rowspan attributes for “td” tags! These attributes determine how tall and how wide a given cell will be. Let us look at an example:

```html
<body>
  <center>
    <table>
      <tr>
        <th></th>
        <th scope="col">H1</th>
        <th scope="col">H2</th>
      </tr>
      <tr>
        <th scope="row">H3</th>
        <td colspan="2">R1C1</td>
        <td>R1C2</td>
      </tr>
      <tr>
        <th scope="row">H4</th>
      </tr>
    </table>
  </center>
</body>
```
<table>
<thead>
<tr>
<th></th>
<th>H1</th>
<th>H2</th>
</tr>
</thead>
<tbody>
<tr>
<td>H3</td>
<td>R1C1</td>
<td>R1C2</td>
</tr>
<tr>
<td>H4</td>
<td>R2C1</td>
<td>R2C2</td>
</tr>
</tbody>
</table>

Code available as 7.3.html
In the first table, there is a column span. Notice how the horizontal distance nearly doubles. Note: the functions of the row and col span are inverted. For instance, the rowspan will make something taller while the colspan makes something wider. The second table is an example of a rowspan. Notice how that cell now takes up two rows.

If you are looking to create a data intensive website, you will most likely need long lists of tables. These tables, so far, are just not going to cut it. Introducing the long table:

```
<table>
<thead>
<tr>
<th><u>Date</u></th>
<th><u>Money</u></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>7</td>
</tr>
<tr>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>5</td>
<td>9</td>
</tr>
</tbody>
</table>
```

Code available as 7.4.html
This table introduces another concept. The tags  <thead> and <tbody> are new tags that help us more easily define the header and the body of a table. For the <thead> section, all you now have to do is just insert a row and add elements. The “u” tags just provide an underline which helps the user better identify the header. Now, for the t body, just add the rows and it will align with the header section.

That is it for this chapter. Since I have pretty much summarized tables in multiple examples so far in this chapter, I going to not include a final code website in this chapter because all of the various table structures are provided in depth.
Chapter 8: Forms

If you have ever signed up for a service on a website, you have surely encountered a form. You know, a form you just love to fill out where it prompts you for your username, password, email, address, gender, age, and any other personal information you have. There are many types of forms you can create. In this chapter, I am going to go through all of them right now:

Text Inputs:

```html
<form action="formaction.php">
<p>Email: <input type="text" name="email" size="25" /></p>
</form>
```

Code available as 8.1.html

First off, whenever you create a form, you must encapsulate all of the form’s contents in the form tag. The action attribute of the form is for submitting forms. We will not cover submitting forms in this book because that involves database communication with the PHP programming language. For now, just input random information in the action tag; it will not matter what you input.

Text inputs are great for inputting information such as passwords, usernames, and emails. You can add multiple text inputs in a form for multiple fields of your choice. In order to support other types of inputs, you may edit the type of inputs as shown:
For passwords (This will make all input those black dots you normally do see when inputting a password):

```html
<input type="password" name="password" size="25" />
```

For usernames, emails, and other types of inputs, use type="text". That allows the contents being written to show up so users may see what they type.

The next type of input object you encounter in forms is paragraph answers. For paragraph answers, you will use the `<textarea>` tag like so:

```html
<form action="formaction.php">
<p>Why do you think you would like to sign up?
<br>
<textarea name="why_answer" cols="30" rows="10">Answer goes here</textarea>
</p>
</form>
```

Code available as 8.2.html

The `<textarea>` tag creates a larger input as shown. The cols attribute defines how wide the area will be. The rows attribute defines how tall the text area will be. The text inside the tag serves as the place holder tag for sample input for users.

Radio Button and Checkboxes:
We are using the exact same input tag with checkboxes and radio buttons alike. This time we just adjust the type to checkbox and radio respectively. In addition, when using these form objects, it is important to have the name be equal in the same group. For example, this is how the browser knows which radio buttons are in which group. Remember, you cannot have multiple radio buttons being pressed. The name attribute helps with grouping and is required.

The drop down box is an alternative to the radio and checkboxes and it is created like so:
<form action="formaction.php">
  <p>Favorite Food:</p>
  <br>
  <select name="food">
    <option value="pizza">Pizza</option>
    <option value="bread">Bread</option>
    <option value="ice_cream">Ice Cream</option>
  </select>
</form>

Code available as 8.4.html

The drop down menu is created with the select attribute and has a name for identification. In the select tag are option tags; these serve to be the radio options of the drop down menu, each representing a value.

File Input Boxes: The next type of form object is the file input box. These are used on online application services where you may have to upload a resume for example. The file input box prompts you to search your computer for a file which will then be placed in the input tag, ready for submission. The file input box is similar to other input boxes:
As you can see, we just use the normal input tag and use the “file” attribute to indicate that a file will be uploaded. The “Browse…” button allows the user to then browse his/her local file system for the file.

In this example, I included another input form, the submit button. The submit button submits the form, meaning that this is the last button you will have in your form. The submit button is the button which will send the form to the webserver using the PHP script indicated when the form was first declared. The submit button is just a normal input tag with type=“submit”. The value is just what appears on the button. In a login form for example, the value may be “Log in.” I wanted to include the submit button here because I want us to practice with it for the remaining form objects.

Buttons: The next form object is probably one of the most well-known: a button. Buttons are seldom used in forms compared to the other objects. Rather, buttons are more frequently used to link to forms. For example you would use buttons on the main page for the “Login” and “Register” buttons. A button is straightforward:
There are actually two types of buttons I included in this one. The first button is the normal button which is created by using a simple <button> tag. In between the <button> tag, you will then put the text you want in the button. In this case, the button has a submit type meaning that it, too can complete and submit a form. In addition, it has a formaction attribute. A formaction, or Form Action attribute, just tells the browser where to go once the button is clicked. For example, if this is a login form, after the button is clicked, the user could be brought to the home screen (Indicated by the URL to the home page).

The second button I included in this example is what is called an image button. An image button serves the exact same role as a normal button, except, it just looks better because it has an image. The image for the image button is specified in the src attribute and the height and width attributes are self-explanatory.

Labels: Labels are just another way to put text next to your form object to indicate what it is for. Here is an example so you know exactly what I am talking about:
As you can see, the “Email:” text perfectly aligns with our input box. For creating labels, first surround everything in the <label> tag. Start off with the actual textual label you want to display. Next, include your form object. It’s that simple.

Dates (8.8.html):
The date input is used for dates. Sorry, I have nothing else to say. It is your normal input tag, but this time with type=“date”. I do not think I need to show you guys an example for this one but I will show you guys what it looks like:
There we go! Done! We have finished forms! You can now get creating your wonderful login and registration forms with confidence! Here is the end of the chapter website:

```html
<!DOCTYPE html>
<html>
<head>
<title>Forms</title>
</head>
<body>
<center>
<h1>Sign Up Today!</h1>
<br>
<form action="formaction.php">
<label>Name: <input type="text" name="name"></label>
<br>
<br>
<label>Email: <input type="email" name="email"></label>
<br>
<label>Confirm Email: <input type="email" name="confemail"></label>
<br>
<br>
How did hear about our wonderful service?
<br>
<input type="radio" name="gender" value="male"> Male
<input type="radio" name="gender" value="female"> Female
<br>
<label>Anything We Should Know About You:<br>
<textarea name="info" rows="5" cols="30"></textarea></label>
<br>
<input type="checkbox" name="email"> Sign Me Up For Emails
<br>
</form>
</center>
</body>
</html>
```
<label>Password: <input type="password" name="pass"></label>
<br>
<label>Confirm Password: <input type="password" name="confpass"></label>
<br>
<h4>How did hear about our wonderful service?</h4>
<select name="hearabout">
<option value="web">Web</option>
<option value="tv">TV</option>
<option value="email">Email</option>
</select>
<br>
<h4>Gender:</h4>
<input type="radio" name="gender" value="male" /> Male
<input type="radio" name="gender" value="female" /> Female
<br>
<label>Anything We Should Know About You: <br><textarea name="otherstuff" cols="30" rows="10">Preferences, etc.</textarea></label>
<br>
<input type="checkbox" name="signmeup" value="emails" /> Sign Me Up For Emails
<br>
<button type="submit" name="submitbutton">Submit</button>
</form>
</center>
</body>
</html>

Can be downloaded as 8.9.html
Chapter 9: Stuff You Should Know About Putting Your Website on the Line

This chapter will not involve any coding. The purpose of this chapter is the next step: uploading your site online. You have all of the HTML code, now what?

Getting Domain Names (optional): This step is not at all required if you want to upload your site. However, some individuals like to have their own domain name when uploading their site. For example, myultrasupercoolhtmlwebsite.com. There are many options for buying domain names (Yes, all domain names cost money). I personally use https://www.godaddy.com. Go Daddy has a great tool for searching for domains that is available here: https://www.godaddy.com/domains/domain-name-search.aspx. Whenever I am buying a domain, I always first look it up to make sure if it’s available and what the price is. From there, I go to check out and buy the domain. Detailed tutorials may be easily found on YouTube.

Web Hosting (not optional): This is the part where you actually transfer your files onto the web. For this part, think of the web servers as virtual computers with their file management systems. Meaning you will just drag and drop your web files into the directories of the web servers.

NOTE: Make sure your main page of your website is titled “index.html”. Web servers use the index.html document as the main page for the website; make sure your main HTML is titled index as well.

Before we talk about uploading to web servers, you will need to install a FTP client (a software which will take your files to the web server also known as File Transfer Protocol client). I personally use FileZilla; it is a free software that definitely gets the job done. I will not cover using FileZilla in this book, but there are a plethora of resources online for you to understand it.

There are many hosting websites. You will need to do some research into which are the best and the most reliable. However, each should have FTP support. Some examples are WordPress, GoDaddy, and GitHub Pages. For my websites, I use GoDaddy. However, for my websites in which I don’t need a domain name, I use a service called Byethost: https://byet.host/free-hosting. You just sign up and get free web hosting. They then give you FTP information for FileZilla in which you can then upload your files to their web servers. I do not have a significant amount of experience with Byethost to say whether it is good or not. I am just saying I use it and have not had a problem yet. That is a basic run through of web
hosting. You can find much more about it online with a simple Google Search or two.

That is it for this book. It has been such an amazing experience writing this novel. I really hope I was able to successfully introduce you all to the wonderful world of HTML and web programming. May your successes continue. If you would like to contact me, email at redefiningproductions@gmail.com.

Stay curious,

Mike Abelar