

HTML ESSENTIALS



By Ian Cheung



HTML



ESSENTIALS

by Ian Cheung

HTML ESSENTIALS. Copyright © 2023 by Ian Cheung. All rights reserved.

This work is intended for educational purposes, and I hereby grant permission for it to be copied, photocopied, or utilized in a manner that aligns with educational endeavors. The content within this work may be reproduced for educational use, provided that it is not for commercial purposes, the integrity of the material is maintained, and proper attribution is given to the original source. This permission extends solely to educational contexts and does not imply authorization for any other form of reproduction, distribution, or utilization in any form or by any means, electronic or mechanical, including photocopying, recording, or by any information storage or retrieval system, without the prior written permission of the copyright owner. By using or reproducing this book, you acknowledge and agree to comply with the conditions outlined in this disclaimer.

June 2023 1st edition

The information in this book is distributed on an “As Is” basis, without warranty. While every precaution has been taken in the preparation of this work, the author shall not have any liability to any person or entity with respect to any loss or damage caused or alleged to be caused directly or indirectly by the information contained in it.

Foreword

I extend my sincere gratitude to each and every one of you for choosing to embark on this exciting journey with me through the pages of my book. As an author committed to delivering a polished and flawless reading experience, I am aware that even with rigorous editing and proofreading processes, occasional typos and inconsistencies can evade detection.

Should you encounter any typographical errors, inconsistencies, or other issues within the pages of this book, I humbly request your kind assistance in bringing them to my attention. Your feedback will aid me in rectifying these matters promptly and effectively. You may reach me directly via email at iancheung.goodtime@gmail.com. Your feedback will aid me in rectifying these matters promptly and will be deeply appreciated. Thank you for your support, and I wish you continued enjoyment as you delve further into the chapters that lie ahead.

Now, let's begin our journey in the world of coding!

To Dad,
for the computers you bought.

To Mom,
for your guidance and patience.

Introduction

Could you even imagine a future where the fate of humanity rests upon the lines of code written by a few individuals? Indeed, this scenario might unlikely happen, but learning how to code does empower you to technology that have the potential to revolutionize society. In fact, coding actually emerges as one of the best careers in the 21st century. In terms of job growth, the tech industry is one of the best and the need for coders is still on the rise consistently. Plus, coding jobs offer a high average salary compared to other industries and many of them are remote, meaning you only need a computer and an Internet connection to work from home. Since they require specialized skills and are in high demand, computer coding salaries often range from \$60,000 to over \$100,000 a year.

Fun fact: you actually don't necessarily need to be a coder in the programming sector – half of all programming opportunities are located in industries outside of technology, including finance, manufacturing, or healthcare. Obtaining these specialized skills, unlike other traditional professions, does not even require years of study or an elusive degree – the vast amount of resources both online and offline has made learning coding more accessible than ever before. Even if you don't pursue coding as your professional career, it can still grant you the power to create an online presence, design a portfolio for your professional career, and even generate a lucrative side income by developing groundbreaking applications. The possibilities are simply endless. What are you even waiting for?

One very important sector in computer programming is web design. According to Siteefy, a website hosting company, there are roughly around 1.13 billion websites on the Internet, with an estimation of 10,500 new websites created every hour. Without a doubt, web design is full of creativity, fun, progression, and skill. It is an ever-evolving subject that is hugely satisfying to learn and master. Every day I wake up with a ravenous thirst, driven by a relentless desire to refine my skill set to develop better websites over time. Learning how to develop websites goes beyond understanding how to write code. It develops me as a person and teaches me how to think independently, solve problems, be resilient in the face of seemingly impossible challenges. Steve Jobs once famously said, "I think everybody in this country should learn how to program a computer because it teaches you how to think." That is a final and essential benefit – learning code forces you to approach problems logically and analytically, asking the right questions and testing your solutions to see if they work. And that is a skill that will benefit you in all aspects of your life.

Once you have finished working your way through the long journey of these pages, you will find that you will look at websites in a completely different way, such as "That's easy", or, "I know how to do that!". Web design is immensely satisfying and is an industry full of developers who don't just love what they do, they live and breathe in it. If you work your way through this book, I can guarantee that you too will love programming and all of its quirks.

This book aims to challenge you – possibly in ways you’ve probably never been challenged before. It will also require you to think in ways you definitely haven’t thought about before. At times when it turns out to be difficult or when you are frustrated, don’t feel down! Continue your journey, accept the challenges, and overcome them with patience and resilience. Now, if you are ready, let’s enter the world of code! Welcome!

Who am I?

Before I continue with web design, I would like introduce you to myself. I am currently a high school student. I was born in Hong Kong, a special administrative region in China, and now reside in the San Francisco Bay Area. Like many other people, the onset of the pandemic and the shift to online schooling prompted my interest in coding. Recognizing that my routine had been severely disrupted, I felt compelled to acquire a new skill set, particularly with the abundance of free time available due to online schooling. As such, I began dedicating my evenings and weekends to constructing (very poorly made) websites.

Having had exposure to coding Scratch and Micro:bit at school since a young age, I had never considered coding to be something I would greatly pursue, but instead something on the side as a hobby. However, the aspects of being a developer became more intriguing over time, most notably the ability to create a product I or someone could actually use, and that made me feel passionate about what I was doing. With a solid foundation in computer programming and dedication to developing a career path and expanding my experience in this dynamic sector, I hope to share my passion for technology and coding with other people. And that’s how this book came to be.

What coding is

For individuals who haven't delved into the world of programming, coding can appear as a mysterious art form. The legendary "coder" is often depicted as someone wearing headphones, listening to electronic music, while simultaneously commanding a computer to fulfill their desires. However, in reality, the entire process is way simpler than that.

To put it simply, coding involves writing lines of code that instruct a computer to perform specific tasks. At its core, computers process information by utilizing a binary system, where a series of 1s and 0s represent "on" (true) and "off" (false) states, respectively. In the early days of computing, interacting with a computer meant manually inputting streams of 1s and 0s, which was far from practical nowadays. As a result, computer languages were gradually developed, enabling far more convenient communication between humans and computers.

Similar to human languages like English or Spanish, computer languages have specific commands (words) and syntax (punctuation) so that both humans and computers can comprehend. However, computer languages are absolutely precise and unambiguous. One single misspelled word or a forgotten comma can cause the entire program to fail. But then, this meticulousness and unambiguity ensure that if you write the right code the computer will do exactly what you want as intended.

But I can control a computer without coding

Well, you might be wondering, ‘But I can do everything I need to do with my computer without needing to enter a single line of code.’ Indeed! Over the past few decades or so, highly user-friendly operating systems such as Windows, macOS, Android, and iOS have been developed, and the need to write code to control a computer is virtually eliminated. Thanks to these advanced Graphical User Interfaces (GUI), everyone can approach a computer and start using it right away without any prior knowledge. This has undoubtedly been such a huge step forward in the usability of computing, but it also means that many people may not realize the immense power that they possess when they go beyond everyday software like Microsoft Word and Google Chrome.

Just so you know, every piece of software or hardware you use has been created through coding. Each time you instruct Siri or perform a search on Google, a whopping amount of several thousand or even million lines of code are executed to simply provide you with a short answer or to show you the search results. There is no magic behind this: it is simply the result of the hard work by thousands of developers, and billions of transistors doing what they are being told in those lines of code.

Learning to code grants you complete power over these transistors, such as by creating your personalized website or automating daily and tedious processes to help improve your productivity. Coding is pretty much like a supernatural power, and in that, it enables you to use equipment like laptops and smartphones you already own in a whole new way.

The Key Principles by which the World Wide Web functions

Before we begin delving into the process of building websites, it's very important to comprehend the landscape in which a website functions. In the following section, you will gain a clear understanding of what we mean when we refer to the World Wide Web. We will then investigate what specifically occurs whenever you visit a web page and how the web page is delivered to you. After that, I will teach you about the fundamental tools utilized in modern web design, providing examples of them that are widely embraced by modern web developers. While there is no shortage of tools and software that a web developer can use, only a few resources are actually required to get your website up and running.

The Internet

The Internet is a worldwide network that links devices together. Within this network, numerous other networks connect billions of devices located in different parts of the world. The devices on this network can communicate with each other and share information using various languages called protocols.

The World Wide Web (www)

The World Wide Web, also known as the web, is a way to get information from the Internet using a protocol called HTTP, which stands for Hypertext Transfer Protocol. It is a platform for sharing information such as websites and files in the Internet itself. HTTP is just one of the many protocols used on the Internet. Another example is SMTP, which is used for sending emails.

Web pages

Web pages are files that are created by using HTML (Hypertext Markup Language). They are saved and uploaded to a web server.

Browsers

A browser is a software program that can understand and show HTML files to the user, allowing them to see and potentially interact with them. The most popular browsers nowadays are Firefox, Google Chrome, Internet Explorer, Brave, Opera, and Safari.

Servers

A server is simply a computer connected to a network, designated to solely provide web pages or files to other devices on the network upon request. Servers are discoverable via IP addresses.

Domains

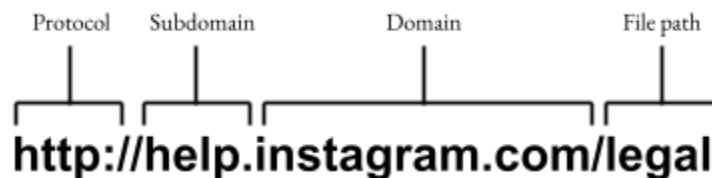
A domain acts similarly to an address or a postcode. It tells the browser where to look for a file. Essentially, a domain name is a name for a server on the network. Anything after the forward slash tells the browser which files on that server you are looking for. You can buy domain names on different platforms such as GoDaddy (<https://godaddy.com/>), but it can be costly.

Subdomains

A subdomain is used to divide your website into smaller sections. They are useful for organizing and structuring a website. It can also potentially contribute to distributing traffic and load balancing when handling high request volumes.

URLs

A Uniform Resource Locator (URL) refers to a specific network location that stores a particular resource. This resource can be in various forms, such as a web page (.html), an image (.png), or a video (.mp4). Generally, a typical URL can be constructed using the following segments:



DNS

A Domain Name System (DNS) functions as a network of servers that are tasked with the responsibility of retrieving the IP address associated with a specified domain. They operate a phone book, where you provide the DNS with a domain name, and it in turn provides you with the corresponding IP address of the server, hence the website you wish to access.

Putting it all together

Let's again walk through the steps involved when you visit a web page through your browser:

1. A user enters a URL into the browser (e.g., www.example.com).
2. The browser queries the DNS server to look up the domain name.
3. The DNS server provides the browser with the IP address associated with the domain.
4. Using the obtained IP address, the browser sends a request to that server for the files.
5. The server processes the request and sends back the requested files to the browser.
6. The browser receives the files from the server.
7. Finally, the browser renders and displays the contents of the files to the user, presenting the web page on the screen.

The tools of web design

Text editor

A website is created using various text-based languages, including HTML, CSS (Cascading Style Sheets), and JavaScript.

To write these languages and have them interpreted by the computer, a text editor is required. While it is technically possible to write a website using a basic text editor like Notepad, it is not ideal for web design because it lacks the advanced features provided by specialized text editors specifically designed for programming.

Throughout this book, we will be using Visual Studio Code as our text editor (you can download it from <https://code.visualstudio.com/download>). While there are countless alternatives available out there, Visual Studio Code is currently the most popular text editor among modern developers. I strongly recommend using it for the duration of this book in order to familiarize yourself with the basic functionality of text editors. After that, you can apply this knowledge to any other plugin-based text editor you choose to use in the future.

Browser

Simply put, web browsers are responsible for presenting websites visually to the user. Without them, all the code we write will remain as code. Indeed, while regular computer users will be happy with any modern browser, web developers have additional considerations when selecting a browser. During the website development process, we especially need effective tools for debugging and inspecting our code.

Each modern web browser offers a mode specifically designed for debugging websites, often referred to as "inspector tools." These tools allow developers to effectively identify and fix issues on the web page within the browser. As a result, the availability and quality of these debugging features greatly influence a web developer's choice of browser.

Among the widely adopted browsers, Google Chrome stands out as the preferred option for both regular users and web developers. Chrome provides an exceptional set of developer tools that prove invaluable in modern web development.

It's worth noting that each browser has its settings and standards for displaying web content. Hence, a website may appear differently across different browsers. To ensure our websites render correctly, it is essential to install and test them on the many popular web browsers currently in use.

Currently, some most popular web browsers, as mentioned before, are:

- Google Chrome
- Safari
- Firefox
- Opera
- Internet Explorer
- Brave

If you are using a Mac, installing Internet Explorer as a native app is not possible because it is designed for Windows operating systems. Although it is not necessary to have Internet Explorer installed to follow the content of this book, in real-world scenarios, you will still need to test your website compatibility with Internet Explorer. To accomplish this, one commonly used approach is to utilize the Remote Desktop application and virtual machines. You can find out more in the following link: <https://docs.microsoft.com/en-gb/archive/blogs/ie/announcing-remoteie-test-the-latest-ie-on-windows-mac-os-x-ios-and-android>.

Image editor (optional)

The modern web is predominantly focused on visual content. In the past, websites had to be cautious with the usage of images due to concerns about slow web page loading speeds. However, thanks to significant advancements in internet speed over the last decade, websites can now incorporate a substantial amount of media without compromising page loading times. Nowadays, it has become essential to possess the ability to create and manipulate media for optimal use on our websites.

One of the key tools for a modern web developer is an image editor. Although you can obviously use default image editors on Windows and macOS to edit, crop, or resize photos, I recommend you install a professional image editor if you are serious about web design. You can gain much more control over the media to create better-looking websites.

While free options for professional image editors are limited, they do exist. GIMP, for instance, is a cross-platform image editor that is more than capable of fulfilling the requirements for image manipulation in web design. Adobe Photoshop, a very popular choice among web developers and professional graphics designers for many years now, has exceptional image optimization and editing capabilities. With Photoshop, it becomes easier to create graphics tailored for your websites. A free trial of Photoshop is available at the time of writing this book, allowing users to test the software for 7 days.

What is HTML?

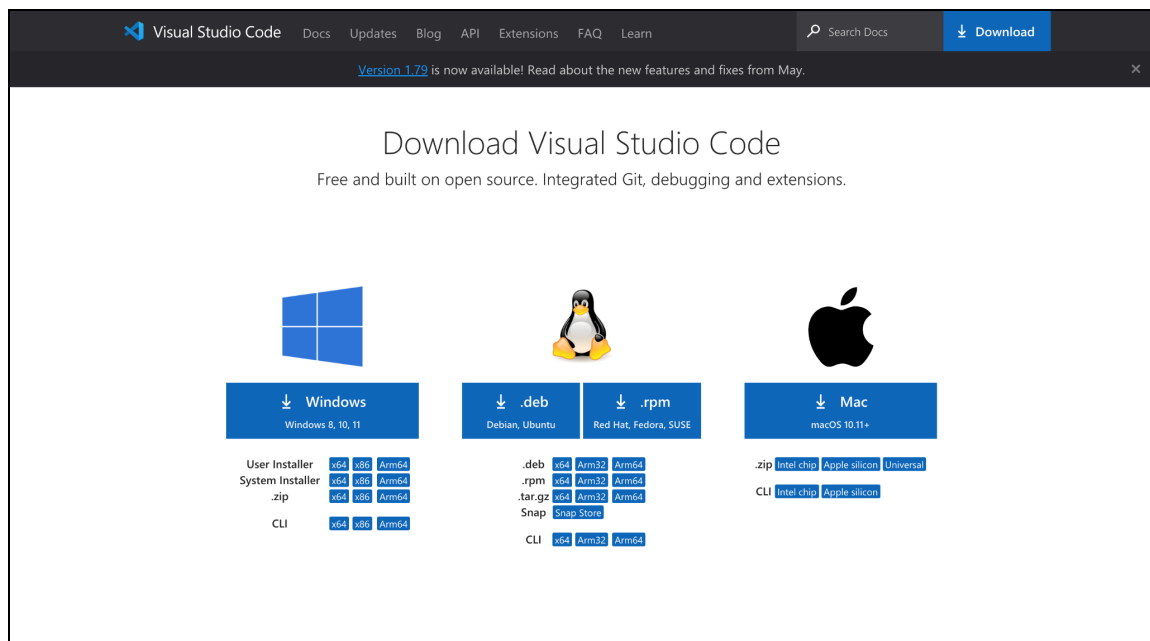
HTML stands for Hypertext Markup Language and serves as the foundation for constructing websites. The term "Hypertext" highlights the key characteristic of HTML, which allows the inclusion of links to other HTML pages, exemplifying the fundamental concept that underlies the web.

HTML originated from the efforts of Tim Berners-Lee in the late 1980s during his work on what we now refer to as the Internet. Initially, Berners-Lee developed HTML as a means of organizing his personal notes but soon recognized the potential for sharing documents with a wider audience. As the web grew, this remarkably effective language led to its adoption by others for implementing text formatting, images, and hyperlinks.

What software do I need?

Throughout the entirety of this book, you only need two software applications ready, namely a text editor and a browser (preferably Chrome). These tools enable you to generate and modify code in any programming language, as well as visualize your creations.

While you have the freedom to choose any text editor, I strongly suggest obtaining Visual Studio Code from <https://code.visualstudio.com/download> as it is free and open-source. It is compatible with Windows, Mac, and Linux operating systems. It will also automatically enhance the readability of your code by employing syntax highlighting and proper indentation, making the process of writing HTML much more enjoyable and relieving you from the burden of manually typing each character.

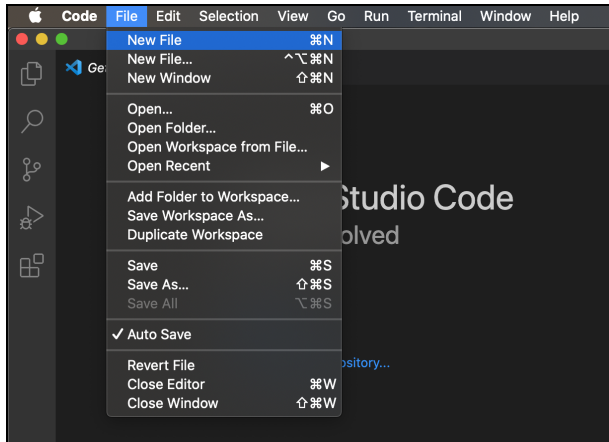


The image shows a screenshot of the Visual Studio Code website's download page. The page is titled "Download Visual Studio Code" and includes the tagline "Free and built on open source. Integrated Git, debugging and extensions." Below the title, there are three main sections for different operating systems: Windows, Linux, and Mac. Each section lists various installation methods and their supported architectures.

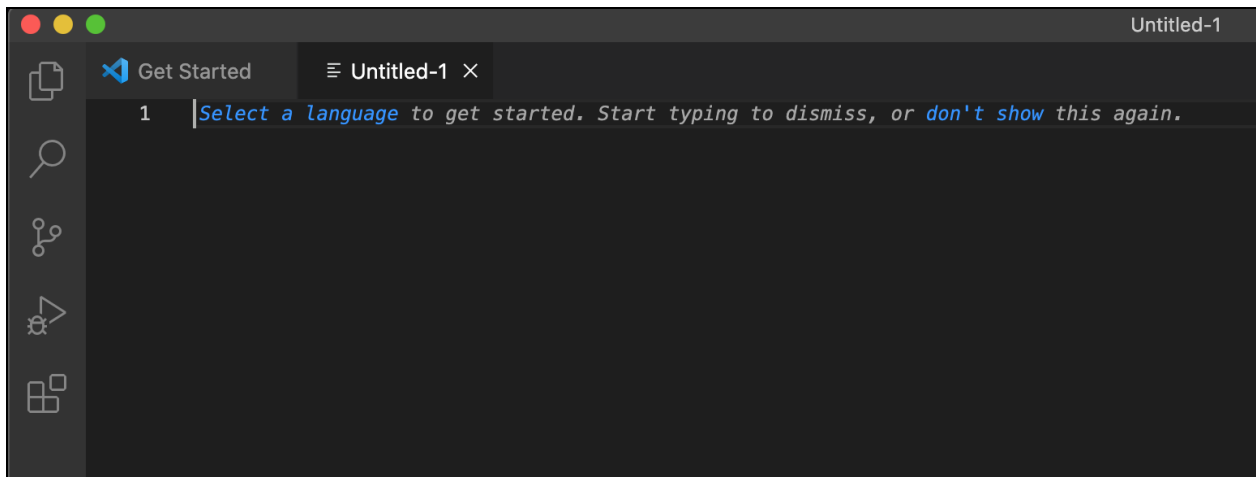
Operating System	Installation Method	Supported Architectures
Windows (Windows 8, 10, 11)	User Installer	x64, x86, Arm64
	System Installer	x64, x86, Arm64
	.zip	x64, x86, Arm64
	CLI	x64, x86, Arm64
	CLI	x64, x86, Arm64
Linux (Debian, Ubuntu, Red Hat, Fedora, SUSE)	.deb	x64, Arm32, Arm64
	.rpm	x64, Arm32, Arm64
	tar.gz	x64, Arm32, Arm64
	Snap	Snap Store
	CLI	x64, Arm32, Arm64
	CLI	x64, Arm32, Arm64
Mac (macOS 10.11+)	.zip	Intel chip, Apple silicon, Universal
	CLI	Intel chip, Apple silicon
	CLI	Intel chip, Apple silicon

I've downloaded "VS Code" - what now?

Upon encountering a series of configurations, you will likely be presented with a welcoming interface. Let's begin by promptly creating a new file.



The beginning of a blank document

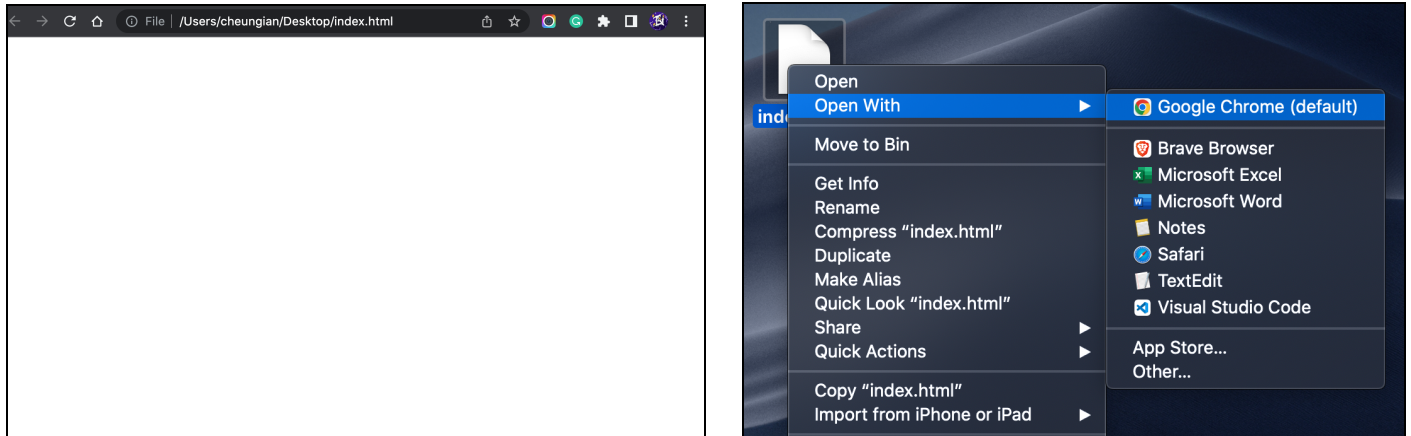


It's very awkward as there is nothing to see yet. Luckily, VS Code provides a convenient solution. Begin by clicking on "Select a language" and search for HTML. Proceed by selecting the HTML option.

Upon completion, it is recommended to save your initial (blank) file. You have the freedom to choose any desired destination, but make sure you can easily locate the file within your File Explorer/Finder.

After that, an HTML document should be generated on your computer. Locate the previously saved document within your File Explorer/Finder and proceed to open it using your web browser.

As expected, the document will appear blank as no code has been written thus far.



Now, let's get right into it. After saving our file as an HTML document, we can dive into writing our HTML code. And what could be a better way to kickstart our HTML code than by using the `<!DOCTYPE html>` declaration and a `<html>` tag?

Example:

```
<!DOCTYPE html>
<html>
</html>
```

Within these `<html>` tags, we will nest all of our HTML code. These tags serve as markers, defining the beginning and end of the HTML code section. In reality, the presence of the `<!DOCTYPE html>` declaration is not 100% mandatory. However, omitting it may occasionally lead to errors, particularly when the browser encounters difficulties interpreting the file correctly. Although the occurrence of such errors is very rare, I strongly advise including this declaration in every HTML document to ensure optimal compatibility and reliability.

The `<head>` tag

First, let's utilize the `<head>` tag. This tag serves the purpose of including the metadata of our web page. Within this section, we can provide essential information that is not directly visible on the web page itself but rather assists the browser in understanding the functionality and structure of our web page.

Example:

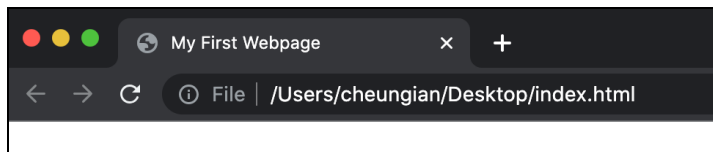
```
<!DOCTYPE html>
<html>
  <head>
  </head>
</html>
```

The Page Title

Now, let's add a `<title>` tag within the `<head>` tag. As you guessed it, the `<title>` tag is used to give the web page a title. This title is what will be displayed on the tab in the browser for your web page.

Example:

```
<!DOCTYPE html>
<html>
  <head>
    <title>My First Webpage</title>
  </head>
</html>
```



In the `<head>` tag, along with the `<title>` element, we also include various **metadata**. Metadata is additional information that helps configure the settings of the HTML document but does not appear visibly within the web page content. Some metadata examples include details such as the author of the web page, the generator used to create it, and other relevant information. Let's discuss one of the most commonly used metadata:

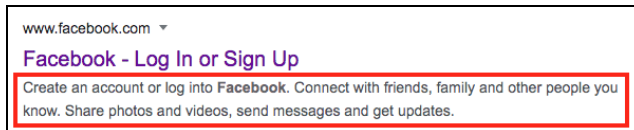
```
<meta charset="utf-8">
```

One metadata that is typically included in most HTML documents is the character set declaration, often set to "UTF-8". "UTF-8" stands for "Unicode Transformation Format 8-bit" and is the most widely employed encoding scheme for representing Unicode characters. This can ensure compatibility for displaying a diverse range of characters and symbols correctly on our web page.


```
<!DOCTYPE html>
<html>
  <head>
    <title>My First Webpage</title>
    <link href="favicon.png" rel="icon">
    <meta name="author" content="John">
    <meta name="description" content="This is a detailed long description of
the web page.">
    <meta name="generator" content="Visual Studio Code">
    <meta name="keywords" content="portfolio, art, graphics, designer">
    <meta http-equiv="content-type" content="text/html">
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
  </head>
</html>
```

Here are what the meta tags mean respectively in order, under the <title> tag.

1. This sets the browser tab's icon to be "favicon.png".
2. This states that the author of this HTML document is "John".
3. The description provides a concise summary or description of your web page's content. Search engines often display this description in the search results, letting users a glimpse of what they can expect when they visit your web page. For example:



4. This states that the text editor you use is "Visual Studio Code".
5. This states that the web page's keywords are "portfolio, art, graphics, designer". This can help you improve your rank in those keywords in Google search results. You may use commas (,) to separate multiple keywords.
6. This states that this document is a text-based HTML document.
7. This states that this document uses UTF-8 encoding.
8. This sets the viewport to ensure that your website looks great on all devices, regardless of their screen size or resolution.
 - a. "width=device-width" sets the width of the page as the screen width of the device.
 - b. "initial-scale=1.0" sets the initial zoom level when the page is first loaded.

While many of these tags only become relevant when your web page is uploaded to a web server and made available online, it's still essential to familiarize yourself with them before doing so.

The <body> tag

Moving on, after the <head> tag, we will now introduce the <body> tag. This tag signifies the beginning and end of the visible content within the HTML document. Inside the <body> tag, we will include the actual content of our web page, such as text, images, and other elements that will be visible to the users.

Example:

```
<!DOCTYPE html>
<html>
  <head>
    <title>My First Webpage</title>
    <meta charset="utf-8">
  </head>
  <body>
    <!-- Page content -->
  </body>
</html>
```

Comments

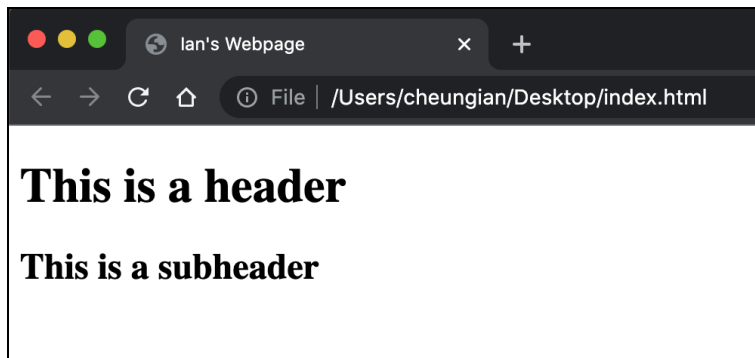
Comments play a very crucial role in programming as they allow developers to explain specific parts of the code using languages that other people can understand. In HTML, comments are enclosed within special tags that distinguish them as non-executable code. This means that they are completely ignored by the browser as if they were not present and have no impact on the web page's functionality. The example <!-- **Page content** --> in the above example is already an illustration of a comment in HTML. To include a comment in HTML code, you can begin the comment with <!-- and conclude it with -->. Comments can also span through multiple lines in HTML with the same syntax. They are particularly valuable when collaborating with other developers, as they provide clarity and understanding when other people are reviewing your code. Now, let's dive deeper into exploring actual HTML elements!

Header Tags

HTML header tags are used to indicate to the browser the hierarchical structure and importance of header text within a web page. These tags enable the browser to understand which text holds greater significance and how it should be treated. Writing header tags is straightforward as they are defined using tags from <h1> to <h6>. The <h1> tag represents the highest level of importance for header text, while the <h6> tag denotes the least important level of header text.

Example:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Ian's Webpage</title>
    <meta charset="utf-8">
  </head>
  <body>
    <h1>This is a header</h1>
    <h2>This is a subheader</h2>
  </body>
</html>
```



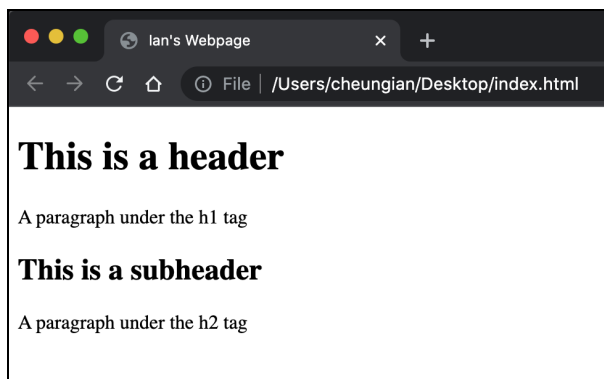
These header tags are particularly useful for displaying larger text to users with emphasis. Although the visual appearance of these tags can vary across different browsers, like any other HTML tags, there is a general convention regarding their font sizes. Typically, the `<h1>` tag is rendered as the largest, while the `<h6>` tag is rendered as the smallest. Occasionally, the `<h5>` and `<h6>` tags may appear with the same font size as the `<p>` tag (which we will discuss later). However, these smaller header tags are commonly utilized for better categorizing and organizing different sections of text within a web page.

Paragraph Tags

For non-header text content, we have the humble yet powerful `<p>` tag, which stands for **paragraph**. Unlike header tags, the paragraph tag has only a single level. Enclosing text within the `<p>` tags signifies a paragraph of text that should appear as a block of continuous content, starting on a new line. You can use multiple `<p>` tags to indicate separate paragraphs, allowing for a clearer and more organized presentation of textual content.

Example:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Ian's Webpage</title>
    <meta charset="utf-8">
  </head>
  <body>
    <h1>This is a header</h1>
    <p>A paragraph under the h1 tag</p>
    <h2>This is a subheader</h2>
    <p>A paragraph under the h2 tag</p>
  </body>
</html>
```



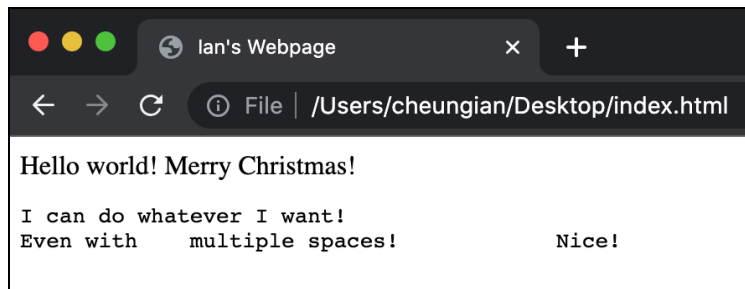
Whitespace collapsing

There is something that you need to be aware of when coding a HTML document, and that is the concept known as "whitespace collapsing." Essentially, when you write multiple lines of code without utilizing tags to separate them, or when the browser encounters consecutive whitespace characters (such as spaces or line breaks in your code), it consolidates them into one single space. As a result, the browser will render the lines of code without any visible line breaks or multiple spaces.

To prevent the collapse of whitespace and preserve the formatting of text, we can utilize the `<pre>` tag. This tag is specifically designed to enclose preformatted text, ensuring that all whitespace, line breaks, and indentation are maintained exactly as they appear in the HTML code. The browser will also render the enclosed text in a monospaced font.

Example:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Ian's Webpage</title>
    <meta charset="utf-8">
  </head>
  <body>
    Hello world!
    Merry Christmas!
    <pre>
I can do whatever I want!
Even with   multiple spaces!           Nice!
    </pre>
  </body>
</html>
```



Inline elements

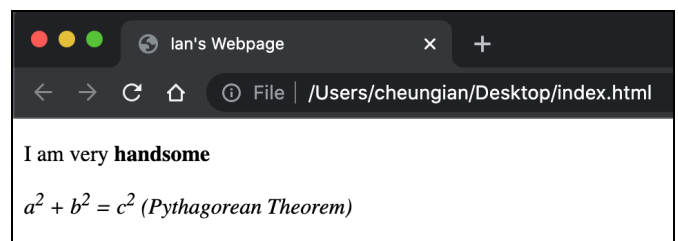
Now that we have successfully printed some texts on our web page, here are some of the tags that can be used to format texts **inline**:

HTML Tag	Effect
 or 	Renders text in bold
 or <i>	Renders text in <i>italics</i>
<u>	<u>Underlines</u> text
<sup>	Renders text as ^{superscript} (appearing above normal text)

<sub>	Renders text as subscript (appearing below normal text)
 or <s>	Strikes through text
<ins>	Displays inserted text, often shown as underlined
<mark>	Highlights text
<q>	Wraps “quotation marks” around the enclosed words
<cite>	Indicates a source, such as an author, book name, or website (usually displayed in italics by the browser)
<small>	Renders text in smaller font size, typically used for remarks or disclaimers
<var>	Displays variables in mathematical formulas (usually displayed in italics)
<bdo dir="rtl">	Overrides the noitcerid txet (text direction) to right-to-left (rtl)
<address>	Represents contact information (usually displayed in italics with a line break before and after the element)
<abbr title="Title">	Represents a short form of a word, with the option to provide a title for the full form of the word that appears when the user hover on the element
<kbd>	Displays user inputs, such as keyboard shortcuts or commands
<samp>	Displays computer output, for example, output from a computer system
<code>	Renders computer code text

The following are some examples on some inline formatting elements and the corresponding output:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Ian's Webpage</title>
    <meta charset="utf-8">
  </head>
  <body>
    <p>I am very <strong>handsome</strong></p>
    <p><var>a<sup>2</sup> + b<sup>2</sup> = c<sup>2</sup></var>
    <i>(Pythagorean Theorem)</i></p>
  </body>
</html>
```

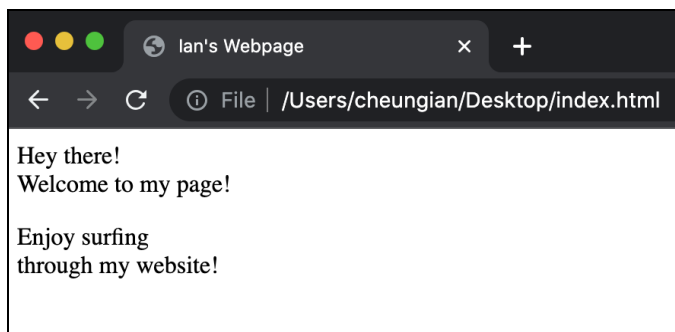


Line break -
 tag

The
 tag is a special tag - it denotes a line break. Practically you can use the
 tag anywhere, such as in between two tags, or even within tags. Its purpose is to create a visual separation within the content, which results in the subsequent content to appear on a new line. Unlike most HTML tags, the
 tag does not require a separate closing tag. It is classified as an "empty HTML element" as it technically does not contain any content.

Example:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Ian's Webpage</title>
    <meta charset="utf-8">
  </head>
  <body>
    Hey there!
    <br>
    Welcome to my page!
    <p>Enjoy surfing <br> through my website!</p>
  </body>
</html>
```



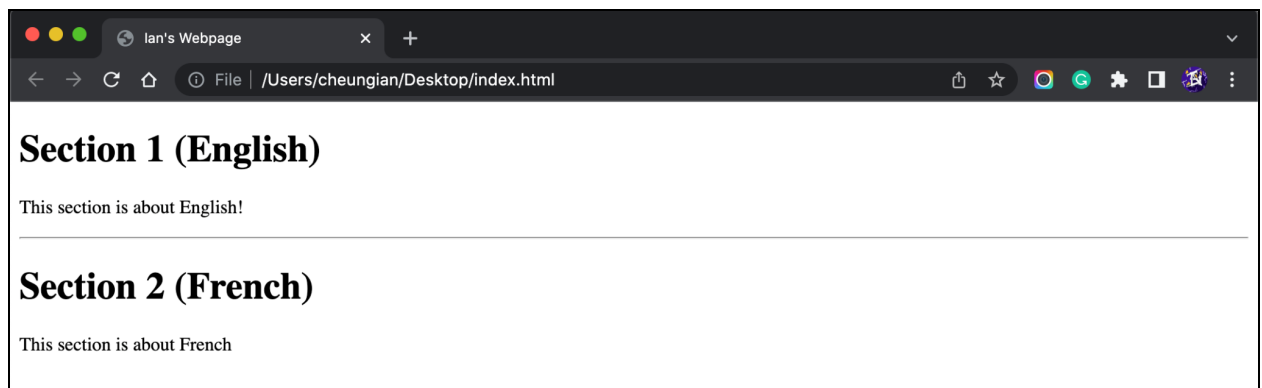
Notice the spacing difference between the
 (between “Hey there!” and “Welcome to my page!”) and <p> (between “Welcome to my page!” and “Enjoy surfing”) tags, both of which facilitate the display of subsequent content on a new line. The
 tag does not create any additional spacing between the lines of text, while the <p> tag automatically creates both a line break and additional spacing before and after the paragraph. This spacing gives a visual distinction between paragraphs and helps improve readability.

The <hr> tag

Similar to the
 tag, the <hr> tag is also really simple. It simply denotes a straight horizontal line across the web page. Let's take a look at this example:

Example:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Ian's Webpage</title>
    <meta charset="utf-8">
  </head>
  <body>
    <h1>Section 1 (English)</h1>
    <p>This section is about English!</p>
    <hr>
    <h1>Section 2 (French)</h1>
    <p>This section is about French</p>
  </body>
</html>
```

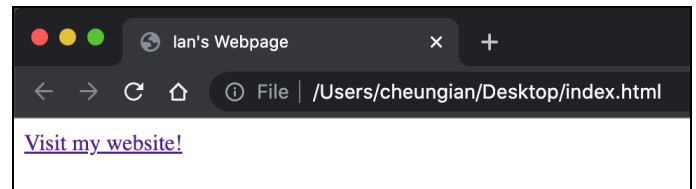


Hyperlinks

A link tag is formatted using a `<a>` tag, with the following syntax:

```
<a href="https://example.com">Visit my website!</a>
```

The **href** attribute specifies the URL of the desired destination. The text enclosed between the opening and closing `<a>` tags represents the visible text that will be displayed as the link. In the given example, clicking on the text "Visit my Website!" would redirect you to the destination URL specified as <https://example.com>.



Link formatting

Absolute URLs

An absolute URL refers to a specific resource that can exist on any location within the web. This type of URL allows you to link to content located on a completely different website. To use an absolute URL, you must format it with the **complete** web address of the desired destination resource. For instance, <https://www.google.com> is an absolute URL that directs you specifically to the Google home page. This allows users to access external web resources and navigate across different websites on the internet.

Relative URL

In contrast to absolute URLs, a relative URL refers to a resource that is on the **same** website or server. It represents a local link within the website. Since relative URLs are referenced from the current document, they require providing a path **relative** to the current document's location. For instance, if the desired file is an HTML file located in the same folder as the current web page, such as when both files are on the Desktop, the URL can be simply formatted as "examplefile.html" without a problem.

To navigate down one directory within the file structure, you can use the forward slash ("/") as a separator. For instance, the URL "images/file.html" refers to the file.html located within the "images" folder, which exists in the same directory as the current document. You can navigate into as many folders as you like, for example: "assets/images/dog/selfie.png".

On the other hand, to move one directory level up in the file structure, you can use the "../" notation. For example, the URL "../file.html" signifies that the desired file.html is located two directory levels above the current document. This notation allows you to access resources in parent directories.

It is generally considered a best practice to utilize relative paths whenever possible. For instance, the BBC website employs relative links for linking between articles since they exist on the same website and server. Relative URLs simplify navigation within a website's internal resources.

Example: (Same website: bbc.com; difference: the file path after "/news/")

<https://bbc.com/news/world-us-canada-53788883>

<https://bbc.com/news/election-us-2020-53782331>

Target attribute

When creating a hyperlink, you can include a "target" attribute to specify how you want the browser to handle the opening of the link. One of the most frequently used options for the target attribute is "_blank". This option instructs the browser to open the linked page or resource in a new browser tab or window, separate from the current page. Here is an example of using the "_blank" target attribute:

```
<a href="https://iancheung.dev" target="_blank">Visit my website!</a>
```

By default, if you do not specify a target attribute for a hyperlink, the browser assumes the target as "_self". This means that when the link is clicked, the linked page or resource will open within the same tab that the link was clicked in. Here are some examples that either explicitly use or imply the usage of the "_self" target attribute, but all perform the same action:

```
<a href="https://iancheung.dev">Visit my website!</a>
```

```
<a href="https://iancheung.dev" target="_self">Visit my website!</a>
```

Mail-to Function

Some websites offer email contact options for their clients. When a user clicks on a hyperlink on the website, it triggers the automatic opening of their computer's email client or the default mail website. This functionality is achieved by adding a "mailto" attribute within the <a> tag.

The "mailto" attribute allows you to create a hyperlink that, when clicked, initiates the email composition process. It specifies the recipient's email address, along with optional queries. Here is an example of using the "mailto" attribute:

```
<a href="mailto:example@example.com?subject=Regarding%20Your%20Inquiry">Contact Us</a>
```

Besides the subject, the "mailto" attribute allows you to include several other parameters or queries to pre-fill additional fields in the user's email composition window. Here are some commonly used queries:

- 1) reply-to: Specifies the email address to which replies should be sent.
Example: `mailto:example@example.com?reply-to=reply@example.com`
- 2) body: Sets the initial content or body of the email.
Example:
`mailto:example@example.com?body=Dear%20Sir/Madam,%0D%0A%0D%0AThank%20you%20for%20your%20interest.`
- 3) cc: Specifies email addresses to be included in the carbon copy (CC) field.
Example: `mailto:example@example.com?cc=cc1@example.com,cc2@example.com`
- 4) bcc: Specifies email addresses to be included in the blind carbon copy (BCC) field.
Example: `mailto:example@example.com?bcc=bcc1@example.com,bcc2@example.com`
- 5) subject: Sets the subject line of the email.
Example: `mailto:example@example.com?subject=Regarding%20Your%20Inquiry`
- 6) attachment: Specifies a file or multiple files to be attached to the email.
Example: `mailto:example@example.com?attachment=file1.pdf,file2.jpg`

In HTML attributes, a blank space usually cannot be directly included. Instead, we use "%20" to represent a space. When rendered by the browser, "%20" is displayed as a blank space.

Download Files

Have you ever encountered a website that initiates the download of files upon clicking a link? Well, achieving that is quite straightforward. You can implement the following code. Users then can simply click on the link on the page to initiate the download of the specified file or folder!

```
<a href="art.png" download>Download Sample Art</a>
```

By default, the file will be downloaded with its original file name. However, you can specify a different name for the downloaded file by setting the download attribute value, in the following case, "portfolio_john.png". Therefore, when downloaded, the file will be saved with the name "portfolio_john.png" in the user's local directory on their computer.

```
<a href="art.png" download="portfolio_john.png">Download Sample Art</a>
```

Since the download attribute is primarily intended for downloading resources that are located within the same domain as your web page, it is generally restricted from downloading resources that are located outside of your web page's domain due to security concerns. This is known as the "same-origin policy" enforced by web browsers to prevent unauthorized access to resources.

HTML bookmark

An HTML bookmark is a hyperlink that references a specific section or element within a web page. To create a bookmark, you assign a unique "id" attribute to the desired destination.

First, let's make our web page very lengthy by adding random text content. This can be achieved by including paragraphs, headings, or any other HTML elements with text inside them. For example:

```
<h2>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vestibulum convallis ipsum vel mi fringilla, id aliquet nisl rhoncus. Suspendisse tristique tincidunt neque, eu pulvinar orci tincidunt at. Ut tristique quam mi, sit amet iaculis elit interdum eget. Donec cursus justo eu leo condimentum consequat. Aenean at fringilla quam. Quisque sed volutpat dolor. Vestibulum maximus felis at sagittis interdum. Vivamus dignissim metus id mi scelerisque, sit amet consectetur turpis venenatis. Maecenas iaculis mauris urna, id tristique eros rutrum eu. Cras pretium mi nec diam elementum, non tincidunt lacus commodo. Sed consequat luctus turpis, ut fermentum neque aliquet vel. In consequat gravida tellus, id aliquam purus volutpat id. Sed congue urna at velit sagittis, nec finibus sem ultricies. Mauris gravida, odio a consectetur viverra, nisl augue bibendum nunc, eu aliquam arcu enim et enim. Quisque quis tempor risus. Sed eu est vel ipsum eleifend ullamcorper. Integer volutpat nulla eu arcu efficitur, vel fringilla tellus auctor. Nunc sed velit ut lectus dapibus interdum. Maecenas auctor quam sed felis volutpat, non interdum elit efficitur. Sed consequat lobortis elementum. Fusce eget enim et lectus egestas iaculis sed et lectus. Donec pellentesque ullamcorper purus. Integer id justo elit. Nulla convallis dui vel tincidunt fringilla. Vivamus quis lacus vitae enim efficitur fringilla. Nunc vitae magna a felis rhoncus facilisis vitae non tortor. Duis vel dolor eget ante tincidunt pretium at id lectus. Morbi malesuada turpis non consectetur finibus. Nullam nec ex sed metus dignissim sollicitudin.</h2>
```

Note: Lorem Ipsum is commonly used as a placeholder text in web design and typesetting. It is used as a temporary filler when the actual content is not yet available or finalized.

Next, we'll create a new paragraph at the bottom of the page **after** the aforementioned long paragraph and assign it an ID. For instance:

```
<p id="lastParagraph">This is the new paragraph at the bottom of the page.</p>
```

Afterward, let's create a hyperlink at the top of the page **before** the lengthy content that jumps to the last paragraph when clicked. The code to achieve this will appear as follows:

```
<a href="#lastParagraph">Jump to the last paragraph</a>
```

By using the href attribute and specifying the value as "#bottom-paragraph", we create a link that points to the paragraph with the corresponding ID. When the hyperlink is clicked, the browser will jump to the paragraph with the "bottom-paragraph" ID.

Hurray! How exciting it is to finally develop something interactive on your web page! Test it out whether it would jump to the bottom of the page when clicking the link.

Images

Images play a crucial role in modern web development. They go beyond just adding visual appeal; they provide context and contribute to an interactive browsing experience. It's crucial to select the right images that align with the style and direction of our website. We can even make our website come alive and create a captivating experience for our visitors by doing so. Now, let's explore how we can enhance our website by adding some vibrant images.

However, before adding images to our web page, we first need to understand the major types of images.

On the web, there are four primary types of images commonly used: JPEG, PNG, GIF, and SVG. Each image type has its own advantages and specific purposes, and the choice of image type should be based on the requirements of the web page.

JPEG

JPEG files are commonly used for photographs due to their ability to achieve high compression ratios, resulting in small file sizes. However, it's important to keep in mind that JPEG is a lossy file format. This means that during compression, some image data is lost, which can lead to a slight decline in image quality. For this reason, JPEGs may not be the best choice for images that require precise details or crisp edges, such as logos.

Additionally, JPEG files do not support an alpha channel, which is responsible for storing information regarding transparency in an image. This limitation means that JPEGs cannot have transparent backgrounds or overlay seamlessly on top of other elements on a web page.

GIF

GIF files are commonly used for logos and animations. They have the advantage of providing small file sizes and the ability to create animated sequences. Unlike JPEG, GIF is a lossless format, which means that no image data is lost during compression. This makes GIFs suitable for images that require precise details and sharp edges, such as logos. GIFs also support animations, which can enhance the dynamic nature of visual content.

Another advantage of GIFs is their support for transparency. They can have areas of the image that are transparent, allowing them to blend seamlessly with different backgrounds. However, one limitation of GIFs is their color palette. They are limited to a maximum of 256 colors, which may not be sufficient for complex images like photographs that require a wide range of colors and shades. GIFs are better suited for images with fewer colors, such as simple line drawings or graphics with limited color variations.

PNG (PNG-24)

PNG files are a relatively newer image file type that offers several advantages over GIFs and JPEGs. Like GIFs, PNGs are a lossless format, meaning they preserve the original quality of the image without any loss of data during compression. PNGs also support varying levels of transparency, whereas GIFs only support binary transparency, meaning each pixel can either be fully transparent or fully opaque. Additionally, PNGs have a wider color range compared to GIFs and can display millions of colors, including full-color images with gradients and shades, making them suitable for photographs and other detailed visuals. However, it's worth noting that PNGs do not support animations as GIFs do.

On the downside, PNG files tend to have larger file sizes compared to JPEGs and GIFs. This can impact website loading times, especially when using multiple or large PNG images. Therefore, it's important to consider the trade-off between image quality and file size when we decide to use PNGs.

SVG

SVG files are gaining popularity among web designers for their unique qualities. Unlike JPEG, PNG, and GIF, which are raster file formats, SVG files are vector-based. This distinction brings several advantages, notably their ability to scale infinitely without losing quality and their remarkably small file sizes. The infinite scalability of SVG files means they can be resized in any ways without sacrificing

sharpness or detail. This makes SVGs ideal for icons, logos, and simple graphics that need to adapt to various screen sizes and resolutions. Moreover, SVG images are created using XML-based markup (literally code), allowing us to manipulate and customize them directly on our websites. This means we can modify the colors, shapes, and other visual elements of an SVG without the need for dedicated photo editing software.

The concern of image file sizes

When incorporating images into your web page, we must be mindful of their impact on the page-loading speed. If your page is taking too long to load, you might consider removing some of the unnecessary images. Another consideration is the size of the images, since large file sizes can significantly slow down the page-loading process. There are various online tools available that can help you compress images effectively, such as <https://imagecompressor.com/>.

Finding the right balance between media-rich content and optimal page-loading times has been a constant challenge for modern web developers. It requires careful analysis and fine-tuning. However, when you manage to strike that sweet spot where your web page loads quickly while still delivering a visually engaging experience, it can add an extra sense of accomplishment to your web design journey.

Image Syntax

To display an image on a website, we utilize the `` tag. The syntax for using this tag is as follows:

```

```

The `` tag is considered an empty tag, which means it doesn't require a closing tag like ``. Instead, it relies on attributes to provide information. The `src` attribute is used to specify the location of the image you want to display. Just like with the link tag, you can use either relative or absolute paths in the `src` attribute. When specifying the image location, it is essential to include the file extension at the end, such as `png`, `jpg`, or `gif`, to indicate the image file type.

The `alt` attribute is used to provide alternative text for an image. This text serves as a fallback when the image cannot be displayed. This is important for cases where the user has a slow internet connection, the browser encounters an issue in loading the image, or when someone relies on a screen reader to access the content. Additionally, search engines also rely on the `alt` text to understand and index the content of the image, which can help improve the image's visibility in search results.

Image sizing

By default, an image is displayed in its original size, which is determined by its native dimensions—dimensions set when the image was originally created. However, we have the ability to customize the image's size by specifying custom dimensions. This allows us to control the width and height of the image being displayed on the web page to suit our specific needs. To do this, we can provide the desired dimensions using the appropriate attributes, as shown below:

```

```

The width and height attributes are used to specify the desired size of an image on a web page. The values for these attributes are typically given in **pixels**, which determine the exact dimensions of the image. However, we can also use **percentage** values to make the image scale proportionally based on the size of the browser window. For example, if we set the width attribute to "50%", it means that the image will always be displayed at half the width of the browser window, regardless of the device or screen size. This can ensure that the image adapts to different viewing environments.

It's important to consider that when both the width and height attributes are specified for an image, it can affect the aspect ratio of the image. This means that if the specified width and height don't match the original image's aspect ratio, the image may appear distorted or stretched. To avoid this, it is common practice to specify either only the width or only the height attribute while leaving the other dimension unspecified. By doing so, the unspecified dimension will automatically scale proportionally based on the image's original aspect ratio.

Image Caption

The <figure> tag allows us to group elements together and create a separate section within our web page. It provides a way to associate a description or caption with the content it wraps. By using the <figcaption> tag inside the <figure> tag, we can specifically target the <figure> element and provide a description for the enclosed content. The appearance of the elements enclosed by the <figure> tag won't be affected. In the following example, we utilize the <figure> tag to wrap two images and use the <figcaption> tag to provide descriptive text for both images.

```
<figure>
  
  
  <figcaption>YouTube Monetization Requirements</figcaption>
</figure>
```

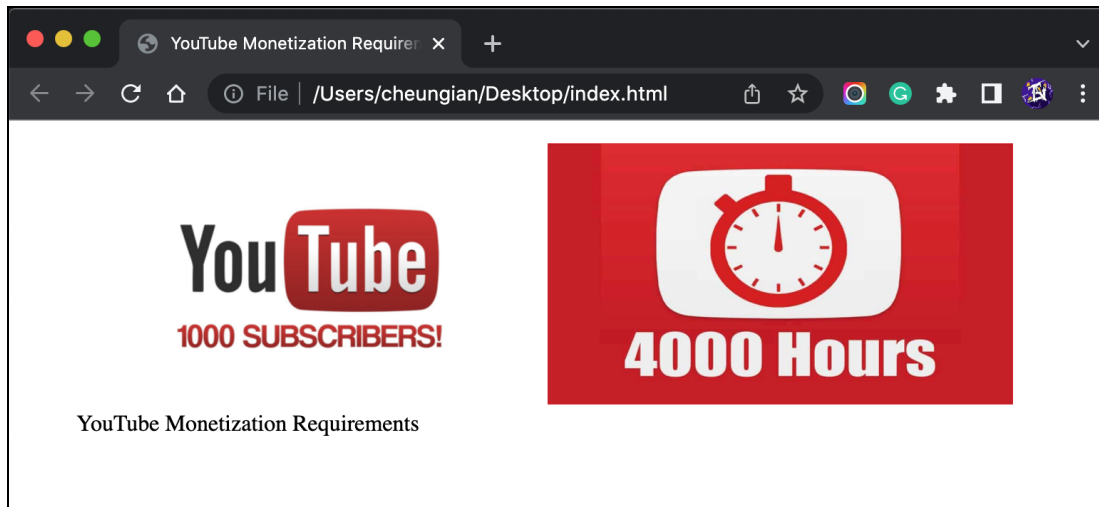



Image Mapping

If you're aiming to redirect users to another page upon clicking on an image, there is a quite straightforward method you can employ. Simply wrap the `` tag with the `<a>` tag, like this:

```
<a href="computer.html"></a>
```

That's no brainer, right? However, what if you want to redirect users to different pages when they click on different parts of an image? Well, an easy solution would be to utilize the `<map>` and `<area>` tags.

The `<area>` element is used in conjunction with the `<map>` element to define clickable areas within an image. The `<map>` element specifies an image map, which is a collection of clickable areas (defined using `<area>`) that correspond to specific regions of an image. Let's take a closer look at the following example to see how to define an image map:

```

```

```
<map name="workmap">
  <area shape="rect" coords="34,44,270,350" alt="Computer"
  href="https://www.w3schools.com/tags/computer.html">
  <area shape="rect" coords="290,172,333,250" alt="Phone"
  href="https://www.w3schools.com/tags/phone.html">
  <area shape="circle" coords="337,300,44" alt="Coffee"
  href="https://www.w3schools.com/tags/coffee.html">
</map>
```

In this example, an image of a workspace is displayed using the `` tag. The `usemap` attribute is set to `"#workmap"` to indicate that it should be associated with the corresponding `<map>` element.

Inside the `<map>` element, we define three clickable areas using `<area>`. The `shape` attribute specifies the shape of the area (in this case, two rectangles and a circle), while the `coords` attribute defines the coordinates that determine the area's boundaries. Additionally, each `<area>` tag includes an `alt` attribute to provide alternative text for accessibility purposes, and an `href` attribute that specifies the URL to which the user should be redirected upon clicking the corresponding area.

The **shape** attribute specifies the shape of an area. Here are the most common ones:

- 1) `rect` – Defines a rectangular region
- 2) `circle` – Defines a circular region
- 3) `poly` – Defines a polygonal region

The **coords** attribute specifies the coordinates of an area in an image map.

- 1) `x1, y1, x2, y2` – Specifies the coordinates of the top-left (x_1, y_1) and bottom-right (x_2, y_2) corner of the rectangle for `shape="rect"`
- 2) `x, y, r` – Specifies the coordinates of the circle center (x, y) and the radius (r) for `shape="circle"`
- 3) `x1, y1, x2, y2, ..., xn, yn` – Specifies the coordinates of the edges of the polygon for `shape="poly"`. If the first and last coordinate pairs are not the same, the browser will add the last coordinate pair to close the polygon

The `<picture>` tag

The `<picture>` element is used to define multiple sources for an `` element. It allows the browser to choose the most appropriate source based on factors like device capabilities, screen size, or resolution. This is particularly useful in responsive web design, where you want to ensure that multiple images can be designed to adapt well to different viewport sizes instead of having one image that is scaled up or down based on the viewport width. To use the `<picture>` element, you need to include two types of tags: one or more `<source>` tags and one `` tag within. For example:

```
<picture>
  <source media="(min-width: 800px)" srcset="large.jpg">
  <source media="(min-width: 600px)" srcset="medium.jpg">
  <source srcset="small.jpg">
  
</picture>
```

The browser will look for the first `<source>` element where the media query matches the current viewport width, and then it will display the proper image (specified in the `srcset` attribute). The `` element is required as the last child of the `<picture>` element, as a fallback option if none of the source tags matches.

- The first `<source>` tag specifies a media query of `"(min-width: 800px)"`, meaning it will be chosen for viewports with a width of 800 pixels or larger. It references the image source `"large.jpg"` using the `srcset` attribute.
- The second one has a media query of `"(min-width: 600px)"`, making it the preferred choice for viewports with a width of 600 pixels or larger. Its corresponding image source is `"medium.jpg"`.
- The third one doesn't have a media query, so it serves as the default option. This image source, `"small.jpg"`, will be used when none of the preceding media queries match the viewport.
- The `` tag is included as the last child of the `<picture>` element to serve as a fallback option in case none of the `<source>` tags are suitable. The `"src"` attribute of the `` tag specifies the source for the fallback image (`"fallback.jpg"` in this example).

Tables

In the earlier days of web design, tables were commonly used as the primary layout element on web pages. They were present on nearly every web page to organize content before the introduction of CSS, as web designers relied heavily on tables to position various elements within cells and columns. While the need for using tables for general web page layout now has diminished, they still hold significance in certain contexts, particularly for displaying structured data. Tables remain the preferred choice when presenting information in a clear and organized structure. Now, let's explore how to create a table.

Table mark-up

Tables in web design are composed of headers, rows, and columns. Each row must have a header (even if it is empty), and must have the same number of columns as the defined headers.

To construct a basic table, we begin by enclosing it within the `<table>` tags. Within these tags, we can define our rows. The first row in our table will serve as the header row, so first we have to create a table row using the `<tr>` tag (`tr` literally means "table row"), as follows:

```
<table>
  <tr>
    <!-- This defines a row of the table -->
  </tr>
</table>
```

Having created a new row within our table, we can now add column headers to this row using the `<th>` tag (th means “table header”). Between the opening and closing `<th>` tags, we specify the text for each header. By default, the header row is styled with bold and centered text to help provide emphasis. While there is no strict limit on the number of columns we can add, let’s consider an example where we add three columns.

```
<table>
  <tr>
    <th>Firstname</th>
    <th>Lastname</th>
    <th>Date of Birth</th>
  </tr>
</table>
```

Now that we have completed the header row, we can proceed to add our data rows. To do this, we simply insert a new row (`<tr>`) immediately after the closing tag of the previous row.

```
<table>
  <tr>
    <th>Firstname</th>
    <th>Lastname</th>
    <th>Date of Birth</th>
  </tr>
  <tr>
    <!-- Next row -->
  </tr>
</table>
```

Within this row, we will be inserting our data. Since we are now working with data cells instead of headers, we will be using the `<td>` tag (td means “table data”). Just like the table header tags, we place our information between the opening and closing tags of the data cell. We can include any content we desire within this cell, even another table if needed. Nevertheless, it’s important to ensure that we have the same number of cells as our header row, so let’s add three cells again.

```
<table>
  <tr>
    <th>Firstname</th>
    <th>Lastname</th>
    <th>Date of Birth</th>
  </tr>
  <tr>
    <td>Elon</td>
    <td>Musk</td>
    <td>06/28/1971</td>
  </tr>
</table>
```

If we wish to include additional data rows, we can easily do so by repeating the process of adding rows beneath existing rows.

```
<table>
  <tr>
    <th>Firstname</th>
    <th>Lastname</th>
    <th>Date of Birth</th>
  </tr>
  <tr>
    <td>Elon</td>
    <td>Musk</td>
    <td>06/28/1971</td>
  </tr>
  <tr>
    <td>Mark</td>
    <td>Zuckerberg</td>
    <td>05/14/1984</td>
  </tr>
  <!-- You can add more rows as needed-->
</table>
```

Sometimes, you may see elements such as `<thead>`, `<tbody>`, and `<tfoot>` being used in conjunction with the `<table>` element to structure the different parts of a table. While they will not affect the layout of the table by default, web designers mostly utilize them to style the enclosed table rows with CSS.

Lists

Lists are widely used in web design for various purposes. In fact, many navigation bars you come across on websites are mostly created using lists in HTML. However, the applications of lists extend beyond navigation. They are also commonly used for image slideshows and, of course, for creating traditional bullet point lists. Depending on our requirements, there are different types of lists we can create. Let's explore them below.

There are three main types of lists that we can use in HTML:

- **Unordered lists:** These are collections of items that don't have a specific order or sequence. Each list item is typically marked with a bullet point by default, although the appearance can be customized later on using CSS.
- **Ordered lists:** These are collections of items that do have a specific order or sequence. Each list item is marked with a number that automatically increments as we add more items to the list.
- **Definition lists:** These are lists where each item consists of a term or label followed by its corresponding definition or description.

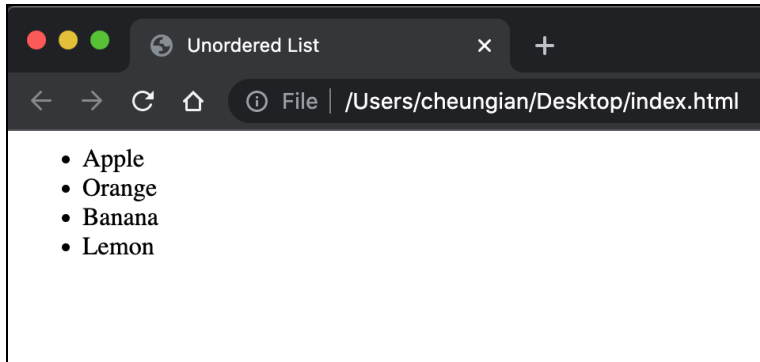
Unordered lists

To create an unordered list in HTML, we use the `` element. The opening tag `` is placed before the list, and the closing tag `` is placed after the list. Each individual item in the list is represented by the `` element, which stands for "list item". The opening tag `` is placed before the item's text, and the closing tag `` is placed after the item's text.

Here's an example of how an unordered list is structured in HTML:

```
<ul>
  <li>Apple</li>
  <li>Orange</li>
  <li>Banana</li>
  <li>Lemon</li>
</ul>
```

By default, when using an unordered list (``) in HTML, the browser automatically adds bullet points as visual markers for each list item (``). The bullet points are part of the default rendering style for unordered lists.



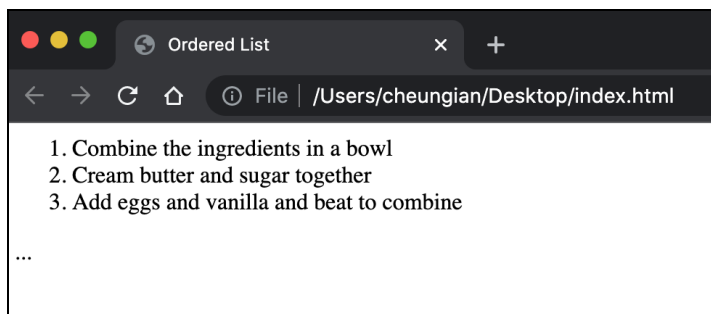
If you prefer to use a different marker or remove the bullet points altogether, you can customize the list style using CSS. With CSS, you can change the bullet point style to different shapes, such as squares or circles, or even remove them completely. However, for the purpose of this book, we will not go into this right now.

Ordered lists

When it comes to ordered lists in HTML, the structure is quite similar to unordered lists. However, instead of using the `` tag to enclose our list items, we use the `` tag instead. The rest of the process remains unchanged.

```
<ol>
  <li>Combine the ingredients in a bowl</li>
  <li>Cream butter and sugar together</li>
  <li>Add eggs and vanilla and beat to combine</li>
</ol>
<p>...</p>
```

Notice how, by default, web browsers automatically display the numbers in ascending order for each item in the list.



OL attributes

Type attribute

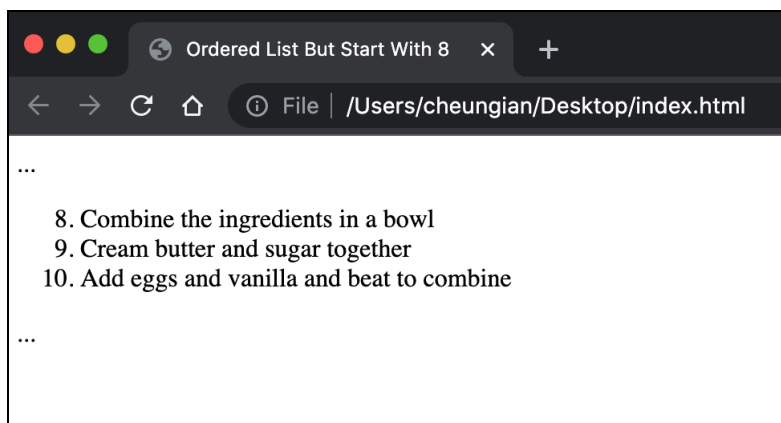
It would be boring if we used numbered lists starting from 1 all day long. In fact, we have the ability to customize how the numbering scheme is displayed in ordered lists by utilizing the type attribute on the opening `` tag. Here are the available values and their corresponding numbering schemes:

- `type="1"`: This is the default value and represents the numbering with Arabic numerals (1, 2, 3, and so on).
- `type="A"`: The list items will be ordered with uppercase letters (A, B, C, and so on).
- `type="a"`: The list items will be ordered with lowercase letters (a, b, c, and so on).
- `type="I"`: The list items will be ordered with uppercase Roman numerals (I, II, III, and so on).
- `type="i"`: The list items will be ordered with lowercase Roman numerals (i, ii, iii, and so on).

Start attribute

In addition to customizing the numbering scheme in ordered lists, we can also specify the starting number using the "start" attribute. By default, the numbering starts from 1, but we can change it to any desired number. Here's an example:

```
<p>...</p>
  <ol start="8">
    <li>Combine the ingredients in a bowl</li>
    <li>Cream butter and sugar together</li>
    <li>Add eggs and vanilla and beat to combine</li>
  </ol>
<p>...</p>
```

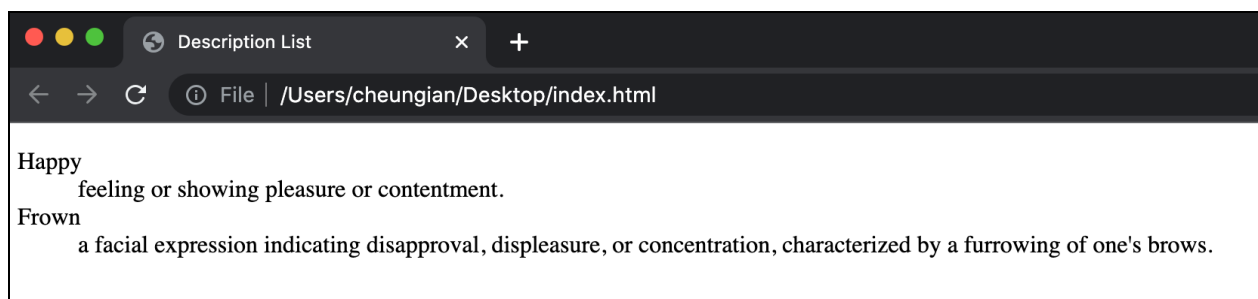


In the above example, if we set the starting number as 8 using the "start" attribute, the list would begin at 8 and continue to the subsequent numbers, such as 9, 10, and so on. Similarly, if we were creating an alphabetical list using uppercase letters, it would start from the eighth letter of the alphabet, which is "H." Hence, the list would begin with "H" and proceed to "I," "J," and beyond, depending on the number of list items.

Description lists

Description lists, also known as "definition lists," are utilized to present terms alongside their corresponding definitions. To construct a definition list, we make use of three tags: the encompassing `<dl>` tag that defines the description list, the `<dt>` tag for the term or name, and finally, the `<dd>` tag for the definition. The structure of the list closely resembles the previous two lists, but for a definition list, we begin with the `<dt>` tag (term), followed by the `<dd>` tag (definition), and then proceed to the next pair of `<dt>` and `<dd>` tags. The code should appear as follows:

```
<dl>
  <dt>Happy</dt>
  <dd>feeling or showing pleasure or contentment.</dd>
  <dt>Frown</dt>
  <dd>a facial expression indicating disapproval, displeasure, or concentration,
  characterized by a furrowing of one's brows.</dd>
</dl>
```



When you view this list in a browser, the definition will be displayed below its corresponding term, indented by one tab space from the left. It's worth mentioning that you are not restricted to using only one definition per term. You can include multiple definitions for the respective term by adding additional `<dd>` elements below the existing `<dd>` element.

iFrames

iFrames provide a way to embed the content of another web page within our web page. This means we can display the contents of another website (whether it be our other pages or completely external websites) on our own web page. Note that certain popular websites like google.com and facebook.com have restrictions in place that prevent their content from being displayed within iFrames.

To achieve this, we can use the following code:

```
<iframe src="http://www.example.com"></iframe>
```

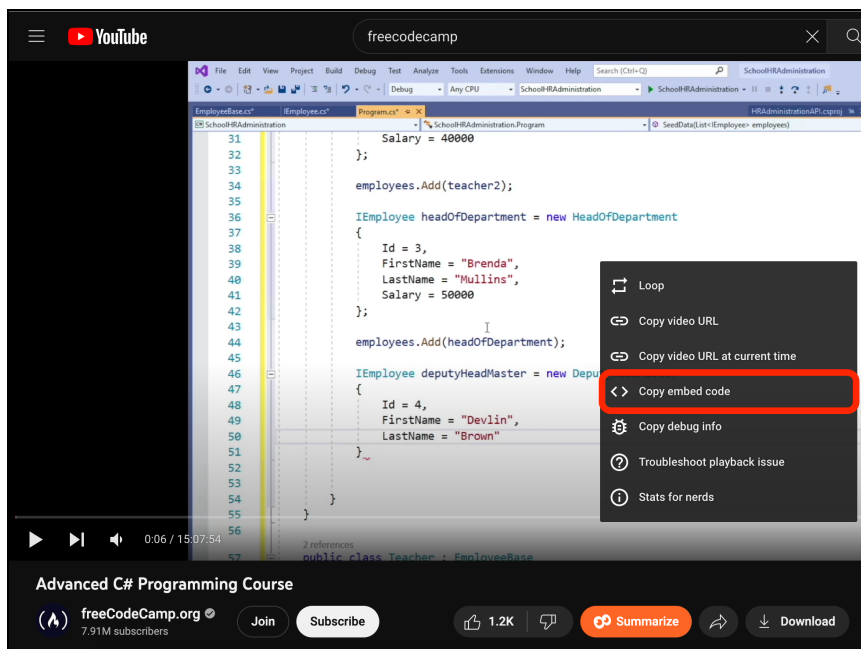
If you'd like to customize the dimensions of the iframe, similar to how we did with images, you can include the width and height attributes within the <iframe> tag. For example:

```
<iframe src="http://www.example.com" width="250px" height="160px"></iframe>
```

Embed

YouTube

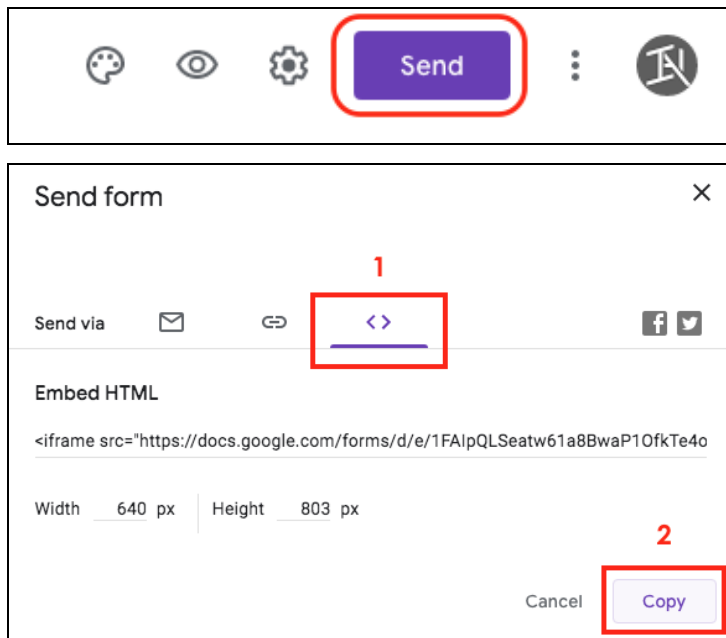
An iFrame can be particularly useful for embedding YouTube videos or Google Forms onto your web pages. To include a YouTube video, go to the YouTube video you want to embed and right-click on the video player. Select the option "Copy embed code" from the context menu.



The HTML code for embedding the video will be automatically copied to your clipboard. You can then paste it into your web page's HTML code, typically within the <body> section. This will allow the YouTube video to be displayed directly on your web page. You can always adjust the parameters and attributes accordingly.

Google Forms

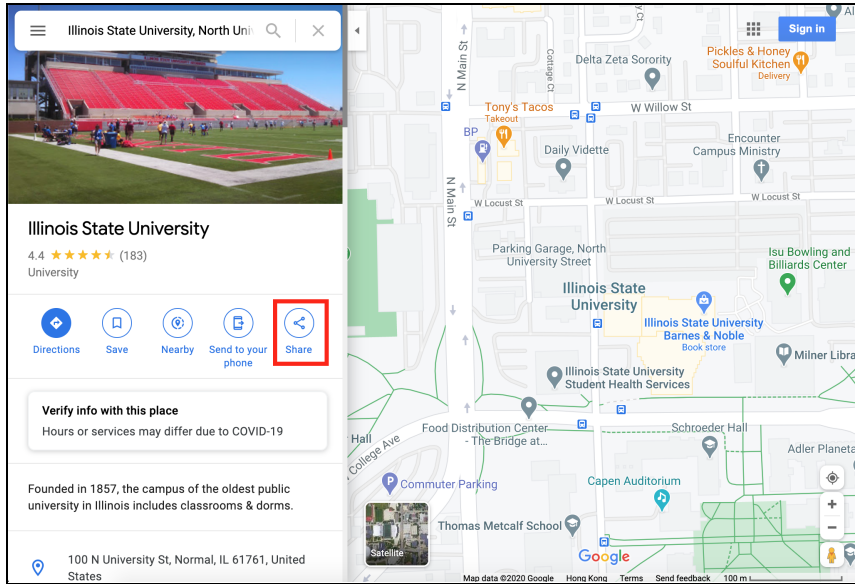
You can also include Google Forms in iFrames. After creating a Google Form, click on the "Send" button in the Google Forms editor. In the options that appear, click on the embed icon (<>). Copy the provided HTML code for embedding the form. Then simply paste the copied code into your HTML document at the desired location.



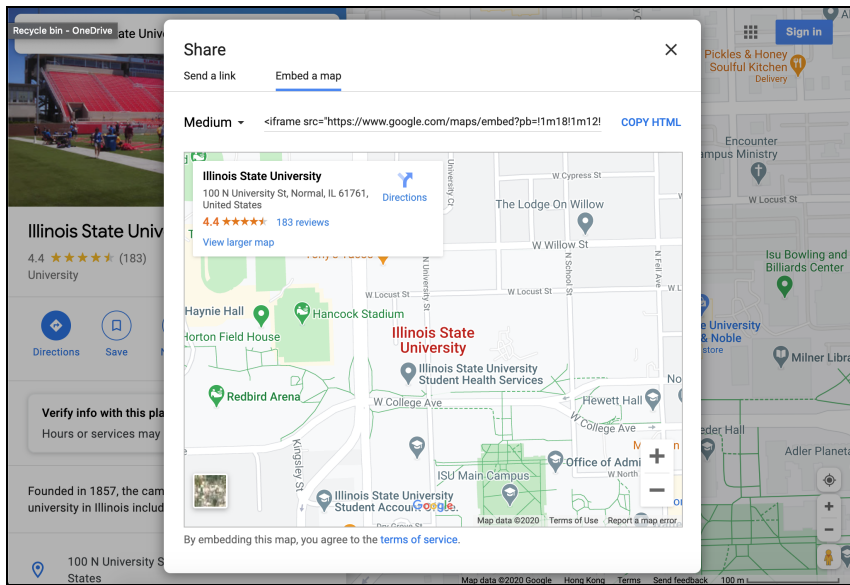
Google Maps

Have you ever wanted to display your location on a map directly on your web page without having to take a screenshot of the map and use it as an image? You can do so by embedding Google Maps. It will provide a much more interactive experience for users as they can explore the map, drag it, and get a better understanding of the surroundings.

Here's how you can embed Google Maps on your web page:



Press the “Share” button and click on “Embed a map”. The following popup will display.



Copy the HTML code and paste it into your HTML document. Simple and nice, isn't it?

Forms

Forms are almost everywhere in the digital world, appearing on websites in various forms such as contact forms, payment information forms, and newsletter subscriptions. They serve as an integral component of web design, facilitating user interaction and engagement. With the ability to gather valuable information from users, forms are particularly effective in uses such as acquiring email addresses for targeted communication, collecting feedback, or enabling users to initiate inquiries and connect with the company. Setting up web forms is a rather straightforward process, making them an ideal choice for establishing communication channels on websites. Now, let's delve deeper into their functionality and implementation.

How forms work

Forms can be divided into two essential components: the form structure and the associated script or logic. The form structure includes various elements such as fields, buttons, dropdowns, text boxes, and radio buttons, which allow users to input and submit information. On the other hand, the script or logic is responsible for processing the submitted data, performing necessary actions, and generating a response. Typically, this script is executed on the server-side and is often implemented using server-side languages like PHP. While the discussion of collecting and handling form data is beyond the scope of this book, keep in mind that PHP forms require the web page to be uploaded on a web server. The following page from W3Schools should likely be able to get you started and guide you in that aspect: https://www.w3schools.com/php/php_forms.asp.

Creating a form

You can create a form using the `<form>` element, which serves as a container for all the form-related content. This content includes various elements like buttons and input fields, although it can also incorporate additional elements such as headings and paragraphs. It is crucial to note that nesting one form inside another is not possible, as it would disrupt the proper functioning of a form.

Form actions

The action attribute is used to specify the desired action to be performed by the browser when the user submits the form. Its value should indicate the location of the script that will be executed upon submission. For instance, it could be something like "process-form.php" or "submit.php".

```
<form action="/scripts/contact.php">
  <!-- Form elements -->
</form>
```

Form method

When creating a form, you specify both the method and action attributes. The method attribute determines how the form data should be transferred to the destination address specified in the action attribute. There are two possible methods: GET and POST.

```
<form action="/scripts/contact.php" method="GET"></form>
```

GET method

The GET method, which is the default method for forms, sends the form data through the URL. The information entered in the form will be appended to the URL as query parameters, which can then be interpreted by the script specified in the action attribute. Here's an example:

Suppose we have a form with the action attribute set to "contact.php" and two input fields, "firstname" and "lastname." If a user enters "Ian" as the firstname and "Cheung" as the lastname, the resulting URL would look something like this:

<https://www.iancheung.dev/contact.php?firstname=Ian&lastname=Cheung>

In the above example, the "contact.php" script will receive the values of the firstname and lastname variables as "Ian" and "Cheung" respectively. The script can then process this information and perform the actions according to the PHP code, returning a result based on the provided data.

POST method

The POST method is an alternative method for sending form data where the information is not visible in the URL. It is commonly used when dealing with sensitive data like passwords and credit card details. Unlike the GET method, the POST method sends the form data privately within the body of the HTTP request rather than appending it to the URL.

The enctype Attribute

The enctype attribute is used to specify the encoding type for sending the form data to the web server. By default, the enctype attribute is set to "application/x-www-form-urlencoded", which is suitable for most form submissions. However, if your form allows users to upload files to the web server, you should set the enctype to "multipart/form-data". Additionally, if you intend to send the form data to an email address, you can set the enctype to "text/plain". This is useful when you want the form data to be sent as plain text in the email.

Target attribute

The target attribute in a form is similar to the target attribute used in <a> tags. By default, when a form is submitted, the server executes the script action within the same tab. However, if you want the form's action to be performed in a new tab, you can set the target attribute to "_blank".

Input fields

1) <input type="text">

This input element creates a single-line text box that allows users to input various information such as names, addresses, comments, and more.

2) <input type="password">

Similar to the text input, this creates a single-line input box. However, the key difference is that the characters entered by the user will be hidden and displayed as dots on the user's screen. This type is commonly used for password fields to enhance security and protect sensitive information from being displayed.

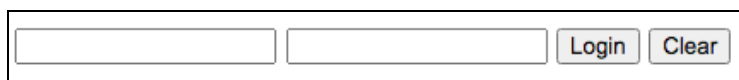
3) <input type="submit">

This input element generates a button specifically designed for submitting a form. When clicked, the browser sends the form's information back to the server by executing the script specified in the form's action attribute. If the "value" attribute is not defined, the button will display the default text set by the browser, which in Chrome's case, is "Submit".

4) <input type="reset">

This input element creates a button that allows users to clear all the inputted data in ALL fields of the form. Similar to the submit button, if the "value" attribute is not defined, the button will display the default text set by the browser, such as "Reset" in Chrome.

```
<form method="POST">
  <input type="text">
  <input type="password">
  <input type="submit" value="Login">
  <input type="reset" value="Clear">
</form>
```



The rendered form consists of a rectangular container with a thin black border. Inside the container, there are two single-line text input fields side-by-side. To the right of these fields are two buttons: one labeled "Login" and one labeled "Clear".

5) `<input type="button">`

This input element creates a plain button that is typically used to execute JavaScript code.

```
<form method="POST">
  <input type="text"><br>
  <input type="button" value="Previous Page" onclick="history.go(-1);">
  <input type="button" value="Next Page" onclick="history.go(1);">
</form>
```

A screenshot of a web form. It features a single-line text input field at the top. Below the input field are two buttons: "Previous Page" on the left and "Next Page" on the right. The buttons have a light gray background and a thin border.

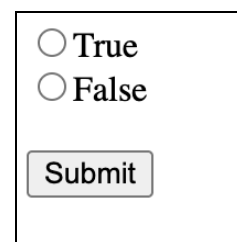
In this example, when users click the "Previous Page" button, the browser executes the JavaScript code (which is out of scope of this book) and navigates the user back to the previous page. Similarly, clicking the "Next Page" button takes the user to the next page.

Please note that if you intend to submit or reset the form, it is important to use the submit or reset button specifically, instead of this regular type of button.

6) `<input type="radio">`

This input type creates a smaller circle that users can click on to select an option. Unlike checkboxes, users can only choose one option at a time. Once an option is selected, it cannot be deselected, meaning the user must choose one and only one option.

```
<form method="POST">
  <input type="radio">True<br>
  <input type="radio">False<br>
  <br><input type="submit">
</form>
```

A screenshot of a web form. It contains two radio buttons stacked vertically. The top one is labeled "True" and the bottom one is labeled "False". Below the radio buttons is a "Submit" button. The radio buttons are unselected, and the submit button has a light gray background and a thin border.

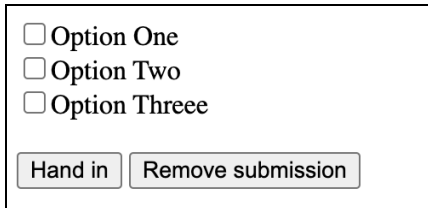
7) `<input type="checkbox">`

This input element creates a small box that users can tick to select one or more options. Unlike radio buttons, checkboxes allow users to choose multiple options if desired.

```
<form method="POST">
  <input type="checkbox">Option One <br>
  <input type="checkbox">Option Two <br>
```



```
<input type="checkbox">Option Three <br>
<br>
<input type="submit" value="Hand in">
<input type="reset" value="Remove submission">
</form>
```

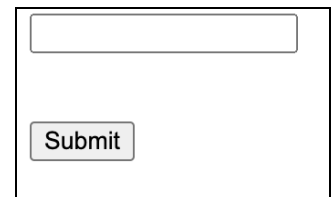


Option One
 Option Two
 Option Three

8) `<input type="hidden">`

Hidden fields are not visible in the form, but their values are still sent to the web server. They are used to transfer information that doesn't require user input but needs to be sent to the back end.

```
<form method="POST">
  <input type="text" name="name"><br>
  <input type="hidden" name="customID" value="69420"><br>
  <br>
  <input type="submit">
</form>
```



9) `<input type="file">`

The file input type allows users to upload files to the web server. When using this field, the form's method attribute must be set to "POST", and the character set (enctype) of the form must be set to "multipart/form-data". Additionally, you can enable multiple file uploads by adding the "multiple" attribute to the tag.

10) `<input type="email">`

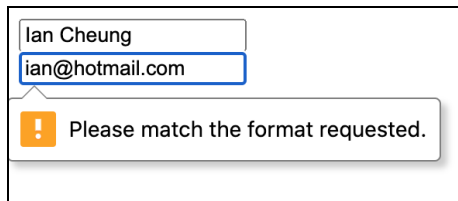
This input type creates a single-line text box specifically designed for inputting email addresses. If the browser detects that the input doesn't match the email address pattern, it will alert the user to re-enter the email address. However, note that the email field only verifies the **format** of the input but doesn't check if the email address actually exists.

If you want to allow users to input multiple email addresses separated by a common symbol (e.g., example@hotmail.com, example@gmail.com), you can also include the "multiple" attribute mentioned earlier.

Additionally, if you wish to set a pattern restriction for email addresses, you can use the "pattern" attribute. For example, pattern=".+@gmail.com" restricts the input to email addresses ending with "@gmail.com".

```
<form method="POST">
  <input type="text" name="name"><br>
  <input type="email" pattern=".+@gmail.com"><br>
  <br>
  <input type="submit">
</form>
```

Upon clicking the submit button, a pop-up message will show up to notify the user regarding the required format.



11) <input type="url">

It creates a single-line text box for the use of inputting an URL. When a user inputs something which is not an URL, the browser will display an alert box to remind the user to re-enter it again. If you wish to only allow website URLs starting with https://, for example, you can use the pattern attribute, like pattern="https://.*".

12) <input type="search">

This input type is technically identical to <input type="text"> as it creates a single-line text box. However, the styling of the text box may vary depending on the browser you are using.

13) <input type="number">

This input type creates a field for users to enter a numeric value. It only allows input of numbers. An additional attribute that can be used with this input type is the "step" attribute. For example, by setting step="3" in the <input> tag, the number value will increment or decrement by 3 when the up or down button is pressed. The "max" and "min" attributes, not maxlength or minlength, are used to define the maximum and minimum allowed values. To set the initial value displayed when the page loads, you can use the value attribute, for example, value="0".

```

<form method="POST">
  <p>Guess my <b>odd</b> number:</p>
  <input type="number" min="0" max="100" step="2" value="1">
  <input type="submit">
</form>

```

14) <input type="tel">

The "tel" value is used for inputting phone numbers. However, it does not validate whether the entered phone number is valid or not, as phone number patterns can vary across different countries or regions. In the following example, we can use the pattern, placeholder, and title attributes to set the phone number format, provide a hint text in the field, and display an alert message when the entered information does not match the requirement. An example is listed below:

```

<form method="POST">
  <p>Enter your telephone number:</p>
  <input type="tel" pattern="[0-9]{4}-[0-9]{6}" placeholder="e.g.
  0932-168167" title="For example: 0932-168167">
  <input type="submit" value="Call">
</form>

```

The image displays two states of a web form. The top state shows a text input field with the placeholder text "e.g. 0932-168167" and a "Call" button. The bottom state shows the same form with the value "12345678" entered in the input field. A validation error message is displayed below the input field, stating "Please match the format requested. For example: 0932-168167".

15) <input type="range">

Contrary to the title, this input type does not input two numbers within a range. Instead, it creates a slider that allows users to select a value from a predefined range. The minimum and maximum values are set using the min and max attributes. The step attribute is also applicable, defining the increment or decrement value when using the slider. By default, the slider is positioned in the middle when the page loads. You can also use the value attribute to set a specific initial position for the slider.

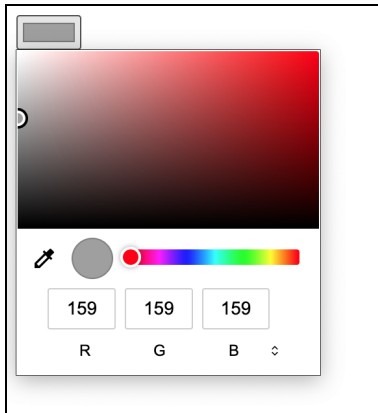
```
<form method="POST">
  <p>Guess my number:</p>
  <input type="range" min="0" max="100" step="5" value="0">
  <input type="submit" value="Guess">
</form>
```

A screenshot of a web form. It contains the text "Guess my number:" followed by a range input field. The range input is a horizontal slider with a blue circular handle positioned at the beginning (0). To the right of the slider is a button labeled "Guess".

16) `<input type="color">`

This input type creates a field that allows users to choose a color. When clicking on the field, a color picker pop-up window will be displayed by the browser or the computer, enabling the user to select a color. By default, the color displayed in the field on the page is black. However, if you want to set the displayed color in the field to red, you can use the value attribute with `value="#ff0000"` (you can visit <https://htmlcolorcodes.com> for more).

```
<form method="POST">
  <input type="color" value="#9f9f9f">
</form>
```



17) `<input type="date">`

This input type creates a field for users to input a date.

18) `<input type="time">`

This input type creates a field for users to input a time.

19) `<input type="datetime-local">`

This input type creates a field for users to input both a date and a time.

20) `<input type="month">`

This input type creates a field for users to input a month.

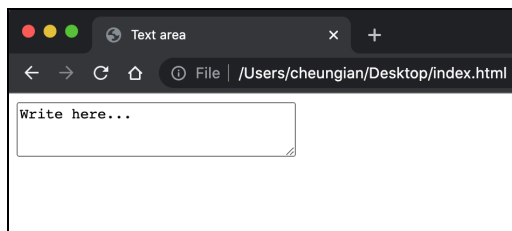
21) `<input type="week">`

This input type creates a field for users to input a week.

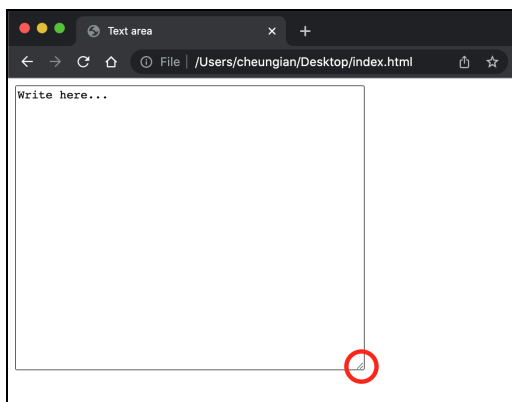
22) The `<textarea>` tag

If you need a large box where users can input a substantial amount of text and customize its size, you can utilize the `<textarea>` tag. When using `<textarea>`, you can specify the width and height of the textbox. Instead of using the width and height attributes like with images, you use the `cols` (columns) and `rows` attributes. Here's an example:

```
<form method="POST">
  <textarea cols="30" rows="3" placeholder="Write here."> Write here...
</textarea>
</form>
```



Users can also drag the edges of the text area to adjust its size.



By default, the field inside the text area is empty. However, if you want to prepopulate the text area with some content, you can place the desired text between the opening and closing `<textarea>` tags, as shown above.

23) Drop-down menus

To create a drop-down list where users can select from a range of options, you can use the `<select>` element. Here's an example:

```
<form method="POST">
  <select name="phone">
    <option value="apple">Apple</option>
    <option value="samsung">Samsung</option>
    <option value="oneplus">Oneplus</option>
  </select>
</form>
```

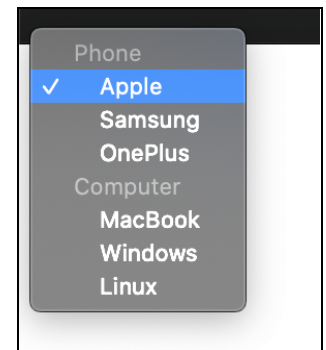
In the above example, the `<form>` element wraps around the `<select>` element, which represents the drop-down menu. The `<select>` element contains multiple `<option>` tags, each representing an available choice for the user. To allow users to select multiple options from the drop-down menu, you can add the "multiple" attribute to the `<select>` tag, like this:

```
<form method="POST">
  <select name="phone[]" multiple>
    <option value="apple">Apple</option>
    <option value="samsung">Samsung</option>
    <option value="oneplus">Oneplus</option>
    <option value="asus">Asus</option>
  </select>
</form>
```

To let the browser process the form accordingly, we will need to set the name of the drop-down menu as a "string", which is widely used in JavaScript, and so, we should add a `[]` (square brackets) after the name of the drop-down menu. Besides, to select multiple options, you need to press the `[Ctrl]` key (Cmd on Mac) to select the option together.

Moreover, we can use the `<optgroup>` to group options together in a group. Look at the following example:

```
<form method="POST">
  <select name="devices">
    <optgroup label="Phone">
      <option value="apple">Apple</option>
      <option value="samsung">Samsung</option>
      <option value="oneplus">Oneplus</option>
    </optgroup>
    <optgroup label="Computer">
      <option value="mac">MacBook</option>
      <option value="windows">Windows</option>
      <option value="linux">Linux</option>
    </optgroup>
  </select>
</form>
```



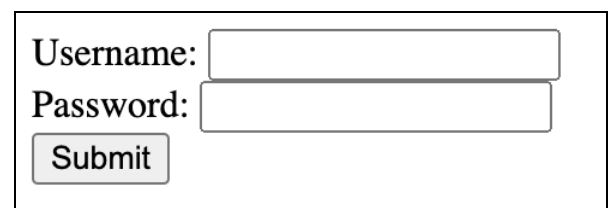
24) The `<button>` tag

In addition to using the `<input>` tag with the `type` attribute set to "submit," "reset," or "button," you can also insert a normal button using the `<button>` tag within the form. Always use the submit or reset button specifically, instead of the `<button>` tag if you intend to submit or reset the form to avoid unintended malfunctioning.

25) The `<label>` tag

Some form elements have default text associated with them, such as the submit and reset buttons mentioned earlier. However, most form elements do not have descriptive text, such as text boxes and password fields. In such cases, you can use the `<label>` tag. Here's an example:

```
<form method="POST">
  <label for="username">Username: </label>
  <input type="text" id="username">
  <br>
  <label for="pwd">Password: </label>
  <input type="password" id="pwd">
  <br>
  <input type="submit">
</form>
```



Input attributes

1) **accept=“...”**

This attribute specifies the acceptable file formats for file upload fields. For example, `<input type="file" accept="image/jpeg, image/gif">` tells the browser that only JPEG and GIF files are acceptable for upload.

2) **autocomplete=“on”**

This attribute defines whether the field should be automatically completed when the page is loaded. Setting it to "on" enables autocomplete, while "off" (the default) disables it.

3) **autofocus**

This attribute tells the browser to automatically focus on that field when the page is loaded.

4) **checked**

This attribute tells the browser to automatically check the radio or checkbox.

5) **disabled**

This attribute disables the field, preventing users from interacting with, clicking on, or typing in it.

6) **name=“example”**

The name attribute specifies the name of the input field. It is used in server-side languages to identify the field and retrieve its value.

7) **maxlength=“10” & minlength=“5”**

These attributes define the maximum and minimum lengths (in characters) allowed for text input fields.

8) **width=“30px” & height=“8px”**

Similar to the attributes used in images, these attributes specify the width and height of the input field.

9) **min=“10” & max=“30”**


These attributes define the minimum and maximum values for number-input fields like "number", "date", or "range".

10) **pattern=“.+@gmail.com”**

The pattern attribute sets a specific input pattern for the field. Users must input something that matches the designated format. The example shown is for email input fields, where users are required to enter email addresses ending with "@gmail.com". Learn more about regular expressions here at https://www.w3schools.com/tags/att_input_pattern.asp.

11) **placeholder=“Write your name here”**

This attribute sets the placeholder or "hint text" for the input field. The placeholder text appears in the field with a gray color, providing a hint to users. When users start typing, the placeholder text disappears.



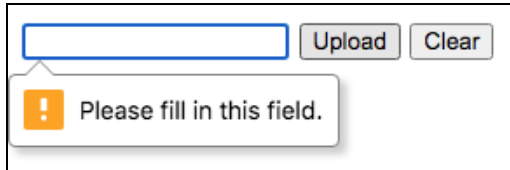
The image shows a rectangular form container. On the left side of the container is a text input field with a light gray border and a light gray background. Inside the field, the text "Write your name here" is displayed in a light gray font. To the right of the input field are two buttons. The first button is labeled "Upload" and the second button is labeled "Clear". Both buttons have a light gray background and a thin border.

12) readonly

This attribute makes the field read-only, preventing users from modifying the inputted information. Unlike with the disabled attribute, users can click on the element but cannot input anything in it.

13) required

This attribute makes the field mandatory, requiring users to fill it in before submitting the form. If a user tries to submit the form without filling in the required field, the browser displays an alert to remind them.



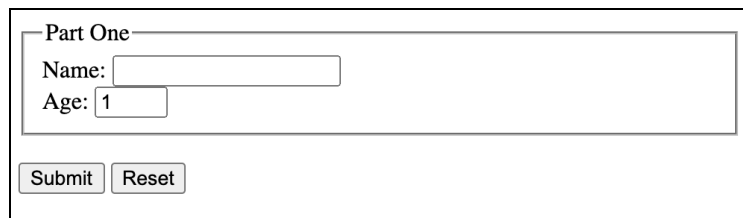
14) size="30"

This attribute is only applicable to text input fields. Setting it to a specific value, such as size="30", adjusts the width of the input field to fit approximately 30 characters. However, users can still type more than the specified size.

Grouping forms elements together

To group form elements together, the <fieldset> element can be utilized. Typically, the browser will enclose all the form elements within the <fieldset> tag using a rectangular border. Additionally, the <legend> tag can be added to define the heading or title of the fieldset.

```
<form method="POST">
  <fieldset>
    <legend>Part One</legend>
    <label for="username">Name: </label>
    <input type="text" id="username">
    <br>
    <label for="age">Age: </label>
    <input type="number" id="age" min="1" max="120" value="1">
  </fieldset>
  <br>
  <input type="submit">
  <input type="reset">
</form>
```

A screenshot of a web form. The form is enclosed in a rectangular border. At the top, there is a legend titled 'Part One'. Below the legend, there are two form elements: a text input field labeled 'Name:' and a number input field labeled 'Age:' with the value '1'. At the bottom of the form, there are two buttons: 'Submit' and 'Reset'.

Videos

In earlier versions of HTML, the absence of `<video>` and `<audio>` tags restricted the direct embedding of videos and audio within web pages. However, with HTML5, modern web browsers gained the capability to handle video and audio playback natively, eliminating the need for external applications such as Windows Media Player, QuickTime Player, or Flash. This enables web developers to seamlessly incorporate multimedia content into their websites without relying on third-party software. The video tag works as follows:

```
<video src="intro.mp4" autoplay controls muted loop preload="metadata"></video>
```

- **src="url"**
This specifies the URL or link to the video that should be displayed.
- **controls**
Including this attribute in the code enables the browser to show the built-in controls for the video, allowing users to play, pause, and adjust the playback.
- **autoplay**
The browser will automatically start playing the video as soon as the page loads if this attribute is included.
- **muted**
This attribute instructs the browser to mute the video's audio when the page is loaded.
- **loop**
The video will continuously replay when this attribute is included.
- **preload="none/metadata/auto"**
The preload attribute offers three options: *none*, *metadata*, and *auto*. When set to *none*, the video will not be downloaded when the page loads. With *metadata*, only essential information about the video, such as its size and duration, will be retrieved without actually downloading the entire video. When set to *auto*, the browser will decide whether to download the video, based on factors such as the user's device and available network resources.
- **height="100px", width="200px"**
These attributes set the width and height of the video player.
- **poster="url"**
This attribute specifies an image to be displayed before the video is downloaded or played. This allows users to have a visual preview of the video content.

The video source tag

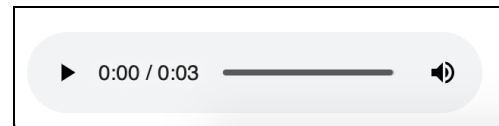
```
<video>
  <source src="intro.mov" type="video/mov">
  <source src="intro.mp4" type="video/mp4">
  <p>The browser does not support the required file format!</p>
</video>
```

In order to ensure compatibility across different browsers that may not support certain video file formats, we may need to provide multiple source options for the video. In the given example, the browser will attempt to play the video file "intro.mov" first. If the browser is unable to play that format, it will then attempt to play "intro.mp4". If both of these formats are not supported by the browser, the <p> element will be displayed by the browser to inform the user that the video cannot be played due to unsupported formats. This can ensure that the video content can be accessed across different browsers.

Audio

Similar to the <video> tag, we can use the <audio> tag to embed audio content in our web pages. Here is an example:

```
<audio src="song.mp3" controls loop></audio>
```



Same to the video source section, we can set different sources for the browser to choose.

```
<audio>
  <source src="song.m4a" type="audio/m4a">
  <source src="song.mp3" type="audio/mp3">
  <p>The browser does not support the required file format!</p>
</audio>
```

Subtitles and captions

Have you ever found yourself watching a show or movie, only to encounter a scene where a character delivers a line that is so incomprehensible that you frantically search for the remote control to rewind? This happened to me a while ago when I was watching the climactic scene of the film "The King of Staten Island" starring Pete Davidson, where his crucial line was nearly impossible to understand. I even had to rewind the scene three times and eventually resort to enabling subtitles in order to finally grasp his words.

Undoubtedly, the utilization of subtitles and captions has become increasingly prevalent in various forms of media. Nearly 57 percent of the respondent in a poll by Vox said that they use subtitles, while just 12 percent said they generally don't. Therefore, we have to consider adding captions to our multimedia content on our web page if we choose to add some.

Luckily, the `<track>` element can be employed to specify external timed text tracks for media elements such as `<video>` or `<audio>`. It is a common practice for incorporating subtitles, captions, or any other textual information associated with the media content on a web page. For example:

```
<video width="320" height="240" controls>
  <source src="forrest_gump.mov" type="video/mov">
  <source src="forrest_gump.mp4" type="video/mp4">
  <track src="subtitles_en.vtt" kind="subtitles" srclang="en" label="English">
  <track src="subtitles_no.vtt" kind="subtitles" srclang="no" label="Norwegian">
</video>
```

Please note that tracks must be formatted in WebVTT format (.vtt files). To learn more about the syntax and formatting requirements of WebVTT subtitle files, I recommend visiting the following resource: https://developer.mozilla.org/en-US/docs/Web/API/WebVTT_API.

The `<details>` tag

Have you ever come across a triangular button on a web page that allows you to toggle the visibility of certain text? If you have, you might have wondered whether you need to use JavaScript or other programming languages to achieve this. Fear not, because with just a straightforward HTML tag called `<details>`, you can achieve this effect with one simple tag.

```
<h1>The details element</h1>
<details>
  <summary>Google Docs</summary>
  <p>Google Docs is an online word processor that lets you create and format documents and work with other people.</p>
</details>
<p><b>Note: </b>The details element is not supported in Edge (prior version 79).</p>
```

The details element

► Google Docs

Note: The details element is not supported in Edge (prior version 79).

Once you click on the triangular button, the hidden content will be revealed.

The details element

▼ Google Docs

Google Docs is an online word processor that lets you create and format documents and work with other people.

Note: The details element is not supported in Edge (prior version 79).

The division tag

The `<div>` tag serves as a versatile tool for grouping and organizing various HTML elements. It acts as a container, allowing you to group a specific set of elements together, making it easier to apply styles and manipulate them using CSS and JavaScript. It is considered a non-semantic HTML tag since it does not inherently convey any meaning about its content, but rather a structural element that aids in organizing and targeting elements efficiently.

```
<!DOCTYPE html>
<html>
  <head>
    <style>
      div {
        border: 5px solid red;
        background-color: lightblue;
        text-align: center;
      }
    </style>
  </head>
  <body>
    <h1>The div element</h1>
    <div>
      <h2>This is a heading in a div element</h2>
      <p>This is some text in a div element</p>
    </div>
    <p>This is some text outside the div element.</p>
  </body>
</html>
```

The div element

This is a heading in a div element

This is some text in a div element.

This is some text outside the div element.

Perhaps this is one of the most complicated and longest chunks of code you've ever seen so far in this book. So let's break it down step by step. First, we have the `<style>` tag, which encompasses all the CSS rules. We have defined the styles for the `<div>` element, setting a 5-pixel red border, a light blue background color, and center-aligned text. Moving on, we encounter a `<div>` element that wraps around the `<h2>` and `<p>` elements, creating a group. Consequently, this div, which contains the two elements within, will adopt the specified style.

The span tag

The `<div>` tag certainly proves useful, doesn't it? However, a minor inconvenience arises—browsers automatically add line breaks before and after the `<div>` element. Hence, to wrap elements inline without adding line breaks before and after them, we need to use the trusty `` tag. Just like the `<div>` tag, the `` tag is also a non-semantic HTML tag. Allow me to demonstrate:

```
<!DOCTYPE html>
<html>
  <head>
    <style>
      div {
        border: 5px solid red;
        background-color: lightblue;
        text-align: center;
      }
    </style>
  </head>
  <body>
    <h1>Here's a title</h1>
    <p>This text will be <span style="text-decoration:
      underline;">overlined</span> for sure!</p>
  </body>
</html>
```

In this given example, you can observe the `` tag enclosing the text "overlined". Within the `` tag, we include a CSS style attribute specifying the text-decoration to be overlined. The `` tag can be used inline throughout the document. However, always remember to employ the `<div>` tag instead of the `` tag when it comes to wrapping a section of content.

Other section wrappers

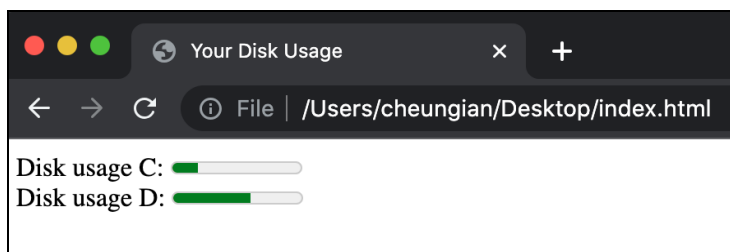
There are many more semantic elements introduced in HTML5 to provide better structure and meaning to the content. They help organize the document and convey the purpose of different sections. Here are some examples:

- `<article>` represents a self-contained article
- `<aside>` is for content tangentially related to the main content
- `<footer>` represents the footer section of a document or section
- `<header>` represents the introductory content or header of a document or section
- `<main>` represents the main content of a document
- `<nav>` is for navigation elements
- `<section>` represents a standalone section or group of content

The `<meter>` and `<progress>` tag

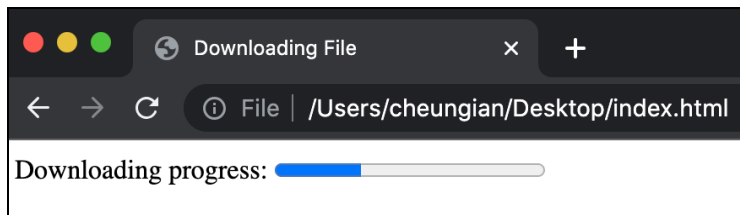
The `<meter>` element represents a scalar measurement within a known range. It's typically used to display values such as disk usage, completion progress, or voting results. It's always a good habit to include the `<label>` tag for each `<meter>` element. The **max** attribute specifies the total amount of work required for the task, with a default value of 1. The **value** attribute indicates the amount of the task that has been completed. If the browser does not support the `<meter>` tag, the text enclosed between the opening and closing `<meter>` tags will be displayed. Here is an example:

```
<label for="disk_c">Disk usage C:</label>
<meter id="disk_c" value="2" max="10">2 out of 10</meter><br>
<label for="disk_d">Disk usage D:</label>
<meter id="disk_d" value="0.6">60%</meter>
```



However, it is important to note that the `<meter>` tag should not be used to indicate progress, such as in a progress bar. For progress bars, the `<progress>` tag should be used instead.

```
<label for="file">Downloading progress:</label>  
<progress id="file" value="32" max="100"> 32% </progress>
```



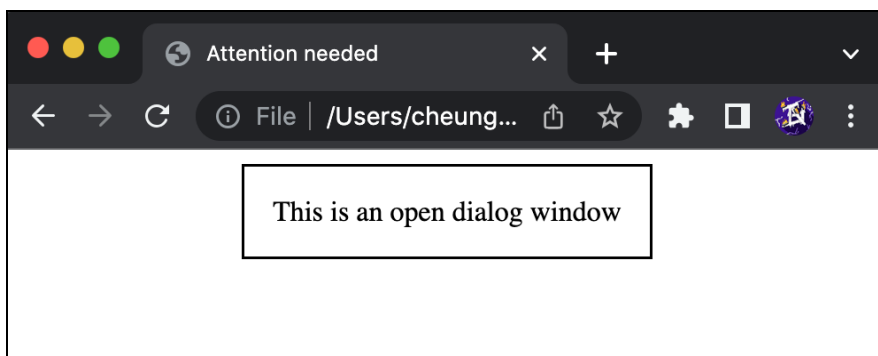
Different browsers may render these tags differently, as with any other HTML tags. You can always customize the appearance later, such as color or size, using CSS.

The `<dialog>` tag

The `<dialog>` tag is used to define a dialog box or subwindow on a web page. It is typically displayed in the center of the page with a border around it, although it varies across different browsers. This element simplifies the creation of popup dialogs and modals, particularly when combined with JavaScript. For now, let's explore how to create a static popup modal that appears when a page is loaded.

```
<dialog open>This is an open dialog window</dialog>
```

The **open** attribute specifies that the dialog element is active and should be displayed.



However, if you are more advanced, have experience with JavaScript, and would like to delve deeper, I highly recommend reading this article by the Mozilla Developer Network (MDN): <https://developer.mozilla.org/en-US/docs/Web/HTML/Element/dialog>. It provides comprehensive information and guidance on utilizing the <dialog> element with JavaScript to create interactive and dynamic dialog boxes and modals.

Special characters/symbols

In HTML, there are special entities that represent various symbols or characters. These entities allow us to display special characters on web pages without conflicting with the HTML syntax. Some of the commonly used entities include:

<u>Symbols</u>	<u>Name entities</u>	<u>Special entities</u>
< (less than symbol)	<	<
> (greater than symbol)	>	>
(blank character)	 	
© (copyright symbol)	©	©

Memorizing all the special symbols and their corresponding entities in HTML can be challenging. Fortunately, there are online resources available to help us easily find and reference these entities when needed. One such resource is the website <https://entitycode.com/>.

Other HTML attributes

Classes

The **class** attribute is used to specify one or more class names for an HTML element. It is primarily used to associate elements with styles defined in a style sheet. However, the class attribute can also be utilized by JavaScript to target and manipulate HTML elements that have a specific class.

An element can have multiple classes assigned to it. To assign multiple classes, you separate them using a space character. For example, the <p class="important active green"></p> element has three classes: "important," "active," and "green." Moreover, a class name can be used by multiple elements, meaning multiple elements can share the same class.

Example:

```
<!DOCTYPE html>
<html>
<head>
  <title>My First Webpage</title>
  <style>
    .intro {
      color: blue;
    }

    .important {
      color: green;
    }
  </style>
</head>
<body>
  <h1 class="intro">Header 1</h1>
  <p>A paragraph.</p>
  <p class="important">Note that this is an important paragraph.</p>
  <p class="important">This is also an important paragraph. :)</p>
</body>
</html>
```



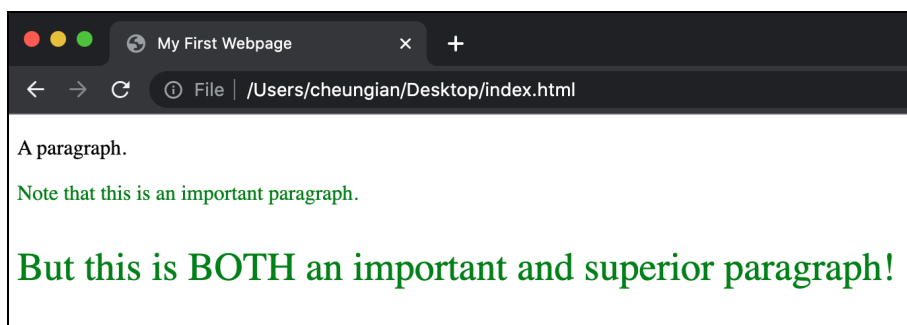
For the purpose of this book, we will go deep into selecting specific elements with classes in CSS and JavaScript.

IDs

In a previous section of this book, we briefly discussed the usage of IDs, specifically in relation to HTML bookmarks. In fact, the **id** attribute serves the purpose of assigning a unique identifier to an HTML element, requiring the assigned value to be unique within the HTML document (only one element can have that ID). Similar to the class attribute, the id attribute primarily functions as a means to reference a specific style defined in a CSS stylesheet and allows JavaScript to manipulate the element associated with the given ID.

Example:

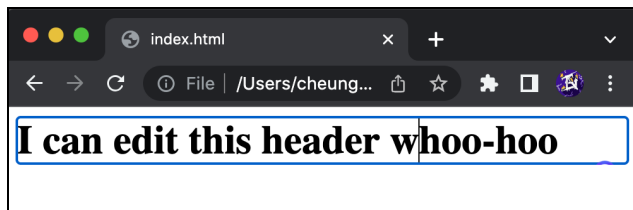
```
<!DOCTYPE html>
<html>
<head>
  <title>My First Webpage</title>
  <style>
    .important {
      color: green;
    }
    #superior {
      font-size: 30px;
    }
  </style>
</head>
<body>
  <p>A paragraph.</p>
  <p class="important">Note that this is an important paragraph.</p>
  <p class="important" id="superior">But this is BOTH an important and superior
  paragraph!</p>
</body>
</html>
```



Contenteditable

The **contenteditable** attribute determines whether the content of an element can be edited by the user. When applied to an element, such as a text element (<h1> or <p>), the user is able to modify its content as if it were a text box. It accepts either "true" or "false" to enable or disable editing, respectively. For instance:

```
<h1 contenteditable="true">This is a totally normal header</h1>
```

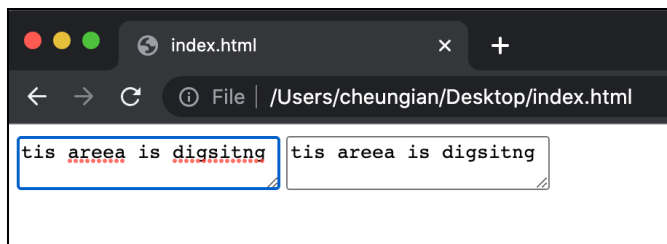


Spellcheck

The **spellcheck** attribute specifies whether the spelling and grammar of an element should be checked. Similar to contenteditable, it can be enabled or disabled by setting its value to either "true" or "false". Only the following can be spellchecked:

- Text values in input elements (not password)
- Text in <textarea> elements
- Text in elements with contenteditable="true"

```
<textarea>tis areea is digsitng</textarea> <!-- With spellcheck -->  
<textarea spellcheck="false">tis areea is digsitng</textarea> <!-- No spellcheck -->
```

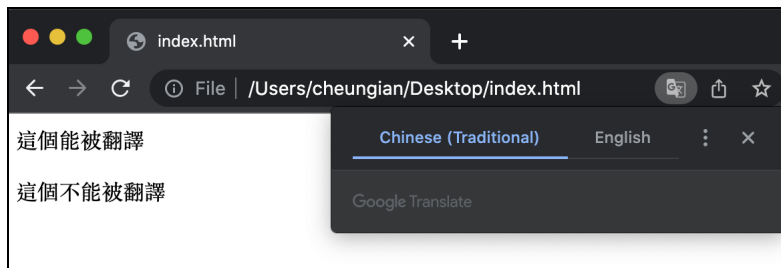


As you can see, when the spellcheck attribute is set to "true", modern web browsers typically provide visual feedback to indicate misspelled words. This feedback is commonly displayed as dotted red lines underneath the incorrectly spelled words.

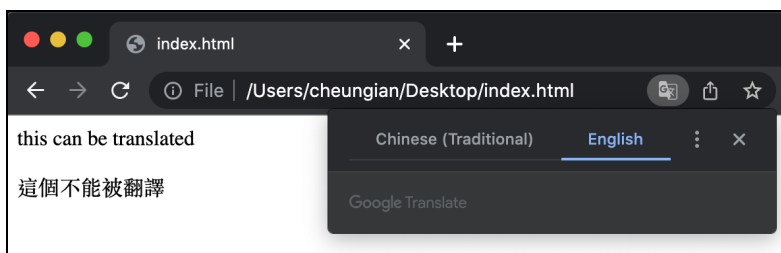
Translate

In many browsers, there are automatic translation functions that activate when a user visits a website containing text in a language different from their default language. However, there may be instances when you want users to view the text in its original language. This is where the **translate** attribute comes in handy. You can determine whether the content within should be subject to browser translation using this attribute. Instead of using "true" or "false," the translate attribute accepts values of "yes" or "no" to enable or disable translation, respectively. For example,

```
<p translate="yes">這個能被翻譯</p> <!-- this can be translated -->
<p translate="no">這個不能被翻譯</p> <!-- this cannot be translated -->
```



When encountering foreign language content, your browser may prompt you to translate it to your default language or even automatically translate it. If a user chooses to translate, the original text will be visible for elements with the translate attribute set to "no", as shown below.



Draggable

The **draggable** attribute allows an element to be draggable by the user. By setting its value to "true" or "false," dragging functionality can be enabled or disabled, respectively. One common use case is to prevent users from downloading images by dragging them to their desktop folder from the web page. This is achieved by setting `draggable="false"` on the `` tag. Here's an example:

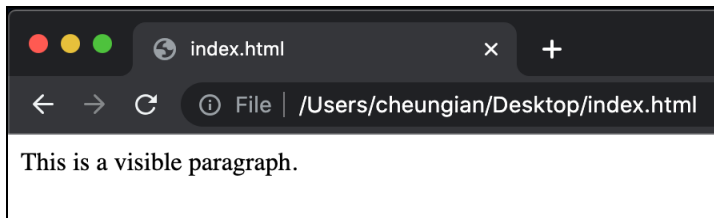
```

```

Hidden

When the **hidden** attribute is present on an element, it indicates that the element is not currently relevant or should not be displayed by browsers. Elements with the hidden attribute specified will not be rendered on the page. This attribute is particularly useful when you want to hide an element until a specific condition is met, such as selecting a checkbox. At that point, a JavaScript program could remove the hidden attribute, making the element visible (JavaScript is, again, out of the scope of this book). For instance:

```
<p hidden>This paragraph should be hidden.</p>  
<p>This is a visible paragraph.</p>
```

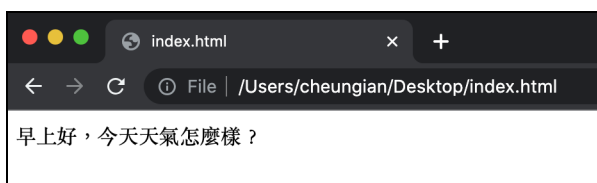


Language

The **lang** attribute specifies the language of the content within an element using a language code, such as "en" for English or "es" for Spanish. You can refer to a list of language codes at https://www.w3schools.com/tags/ref_language_codes.asp. Different languages may have specific character restrictions or require different fonts for proper rendering. Here's an example written in Traditional Chinese that changes font when the lang attribute is applied.

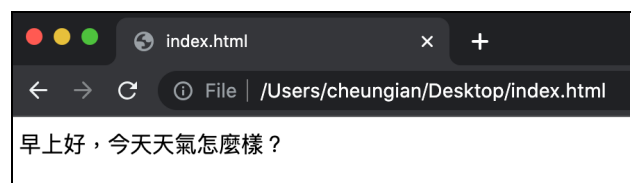
Without the lang attribute:

```
<p>早上好，今天天氣怎麼樣？</p>
```



With the lang attribute:

```
<p lang="zh-HANT">早上好，今天天氣怎麼樣？</p>
```



Conclusion

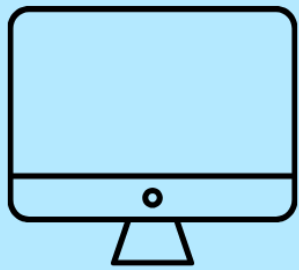
Congratulations! You have successfully mastered the fundamentals of HTML, including some lesser-known yet highly useful tags. Now, it's time to put your knowledge into action and create your own web page, incorporating as many elements as possible. Choose a topic that interests you, and let your creativity flow as you design your page. Start by adding a title to your web page, utilizing header tags for a proper site structure. Enhance your page with images, lists, iframes, and any other elements we have covered throughout this book. Feel free to explore and experiment with different elements!

Up until now, our web pages have been limited to local access only, meaning they can only be viewed on our own computers. But what if we want to share our fabulous creations with the rest of the world? Fortunately, there are numerous free options available for hosting websites online. One such option is **Replit**, a collaborative browser-based Integrated Development Environment (IDE). With Replit, you can easily assemble a simple HTML/CSS website using its awesome editor. Visit <https://repl.it> and create an account if you haven't done so already. Once you're logged in, create a new "repl" by selecting the "HTML, CSS, JS" option as your preferred language. Feel free to name your repl whatever you like.

Now, you can add the code you've written in your local index.html file to the repl. Simply paste your code into the editor. To see your website in action, click the green "Run" button. A popup window will display your website, or if you prefer, you can click the preview button to open it in a separate tab. What better way to share your website than by copying the link and sharing it with your friends? Additionally, if you're interested in hosting your website with a custom domain (a web address of your choice), I've written a blog post that guides you through the process. You can find it at: <https://blog.ianbrawlstars.com/how-to-get-your-websites-online/>.

Now you have the power to share your creations with the world! As we've discovered, HTML forms the foundation of every web page. With your newfound knowledge of HTML, you can now create and modify basic web pages. However, the journey doesn't end here. The next step in our coding adventure involves styling our HTML pages, such as adding colors, adjusting sizes, and fine-tuning text formatting. We accomplish this through CSS (Cascading Style Sheets). If you're interested in diving deeper, stay tuned for my next book, or explore online resources like w3schools.org to expand your skills.

Peace out, and happy coding!



HTML ESSENTIALS

*“Learning to write programs **stretches your mind**, and **helps you think better**, creates a way of thinking about things that I think is **helpful in all domains**.” - Bill Gates*

Computers possess the incredible power to unravel a multitude of challenges once they receive the appropriate instructions. This is where the art of programming comes into play. Embark on an exhilarating journey into the world of HTML and web design with this beginner-friendly guide, as it transports you into a realm of crafting captivating webpages.

By the end of this book, you will not only have attained mastery in HTML but will have also acquired the invaluable ability to analyze problems and confront them head-on with code. While programming languages may rise and fall, this tome will furnish you with an enduring foundation, cultivating the mindset of a seasoned programmer, poised to conquer any challenge that comes your way.

"Ian is a highly skilled coder and developer. His ability to create versatile Discord bots and his proficiency in coding has led to the development of some truly impressive websites and apps. His passion towards computer science is undeniably evident in the quality of his work."

Ian Cheung 

www.iancheung.dev