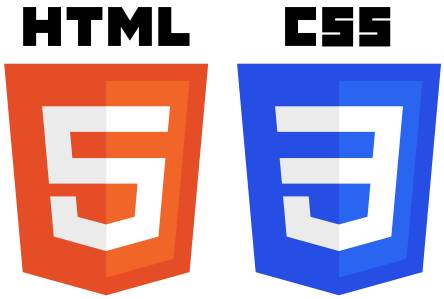
HTML & CSS

FREE HANDBOOK



Aliff Aiman
Adam Hafiz

Introduction

If you're reading this book, chances are you're interested in learning HTML and CSS. But before we dive into the details of how to use these tools to create websites, let's take a moment to define what HTML and CSS actually are, and why they're important.

HTML stands for Hypertext Markup Language. It's the standard markup language used to create web pages. HTML provides the structure and content of a web page, defining elements such as headings, paragraphs, and lists. When you visit a website, your web browser reads the HTML code and uses it to display the page.

CSS, on the other hand, stands for Cascading Style Sheets. CSS is used to style web pages, providing the visual design elements such as fonts, colors, and layouts. With CSS, you can control the appearance of every aspect of a web page, from the font size and color of text to the position and size of images.

Together, HTML and CSS are the foundation of every website on the internet. They allow web developers to create rich, dynamic, and engaging experiences for users.

But why should you learn HTML and CSS? Here are just a few reasons:

1. In-demand skill: Web development is a rapidly growing field, and HTML and CSS are essential skills for anyone looking to work in this industry. From building basic websites to creating complex web applications, HTML and CSS are used in every aspect of web development.
2. Control and customization: By learning HTML and CSS, you gain complete control over the look and feel of your website. You can create unique designs and layouts that stand out from the crowd, and customize every aspect of the user experience.
3. Creative expression: HTML and CSS offer endless opportunities for creative expression. Whether you're building a personal website, creating an online portfolio, or developing a new web app, HTML and CSS allow you to bring your ideas to life in ways that are only limited by your imagination.

In the following chapters, we'll dive deeper into the details of how to use HTML and CSS to create websites. But before we get started, let's take a quick look at what you'll need to get started with web development.

Getting Started with Web Development

Before you can start creating web pages with HTML and CSS, you'll need to set up your development environment. Here are the basic tools you'll need to get started:

1. Text Editor: A text editor is a program that allows you to write and edit code. There are many text editors to choose from, including free and paid options. Some popular text editors for web development include Sublime Text, Visual Studio Code, and Atom.
2. Web Browser: You'll need a web browser to view and test your web pages. Most modern web browsers such as Google Chrome, Mozilla Firefox, and Microsoft Edge are capable of rendering HTML and CSS.
3. Local Web Server: A local web server allows you to test your web pages locally on your computer. There are many options for setting up a local web server, including Apache, Nginx, and Node.js.

Once you have your development environment set up, you're ready to start creating web pages with HTML and CSS.

Let's take a quick look at the basic structure of an HTML document. Every HTML document begins with a document type declaration, which tells the browser what version of HTML the document is written in. For example, the document type declaration for HTML5 is:

```
<!DOCTYPE html>
```

After the document type declaration, the HTML document consists of two main sections: the head and the body. The head section contains metadata about the document, such as the title of the page and links to CSS stylesheets. The body section contains the content of the page, including headings, paragraphs, images, and other elements.

Here's a basic example of an HTML document:

```
<!DOCTYPE html>
<html>
  <head>
    <title>My First Web Page</title>
    <link rel="stylesheet" href="styles.css">
  </head>
  <body>
    <h1>Hello, World!</h1>
    <p>Welcome to my first web page.</p>
    <img src="image.jpg" alt="An image">
  </body>
</html>
```

In this example, we've included a title for the page and a link to a CSS stylesheet. We've also added a heading, a paragraph of text, and an image to the body of the page.

In the next chapter, we'll dive deeper into the basics of HTML, including how to create headings, paragraphs, and lists.

Getting Started with HTML

Now that you've set up your development environment and learned the basics of the HTML document structure, it's time to start creating some HTML content.

HTML elements are the building blocks of web pages. An HTML element consists of a start tag, content, and an end tag. The start tag is enclosed in angle brackets (< >) and includes the name of the element. The end tag is also enclosed in angle brackets, but includes a forward slash (/) before the element name.

Here's an example of a basic HTML element:

```
<p>This is a paragraph element.</p>
```

In this example, the start tag is ** ** and the end tag is ** **. The content between the start and end tags is "This is a paragraph element."

Here are some of the most commonly used HTML elements:

1. Headings: Headings are used to create headings for sections of your page. There are six levels of headings, with **<h1>** being the largest and **<h6>** being the smallest.

```
<h1>This is a level 1 heading</h1>
<h2>This is a level 2 heading</h2>
<h3>This is a level 3 heading</h3>
```

2. Paragraphs: Paragraphs are used to create blocks of text.

```
<p>This is a paragraph element.</p>
```

3. Lists: Lists are used to create ordered and unordered lists.

```
<ul>
  <li>Item 1</li>
  <li>Item 2</li>
  <li>Item 3</li>
</ul>

<ol>
  <li>Item 1</li>
  <li>Item 2</li>
  <li>Item 3</li>
</ol>
```

4. Links: Links are used to create hyperlinks to other web pages or resources.

```
<a href="<https://www.example.com>">Click here to visit Example.com</a>
```

5. Images: Images are used to display images on your web page.

```
<img src="image.jpg" alt="An image">
```

These are just a few examples of the many HTML elements available. In the next chapter, we'll learn how to add links and images to your web pages, and how to use HTML5 semantic elements to improve the structure of your content.

Links and Images in HTML

Now that you've learned the basics of HTML elements, it's time to add some more content to your web page. In this chapter, we'll learn how to add links and images to your HTML documents.

Links

Links are used to create hyperlinks to other web pages or resources. To create a link, you'll use the **<a>** element, which stands for "anchor." The href attribute is used to specify the URL that the link should point to.

Here's an example of a basic link:

```
<a href="<https://www.example.com>">Click here to visit Example.com</a>
```

In this example, the link text is "Click here to visit Example.com" and the URL is "**https://www.example.com**."

You can also create links to other pages on your own website. For example:

```
<a href="about.html">About Us</a>
```

In this example, the link text is "About Us" and the URL is "about.html." This assumes that you have a file named "about.html" in the same directory as your current HTML file.

Images

Images are used to display images on your web page. To add an image, you'll use the **<img>** element. The src attribute is used to specify the URL of the image, and the alt attribute is used to provide a description of the image.

Here's an example of an image element:

```
<img src="image.jpg" alt="An image">
```

In this example, the URL of the image is "image.jpg" and the alt text is "An image."

You can also specify the width and height of the image using the width and height attributes:

```
<img src="image.jpg" alt="An image" width="200" height="150">
```

In this example, the image will be displayed at a width of 200 pixels and a height of 150 pixels.

HTML5 Semantic Elements

HTML5 introduced several new semantic elements that are designed to improve the structure and readability of your HTML code. Here are a few examples:

1. **<header>**: The header element is used to define a header for a document or a section of a document.
2. **<nav>**: The nav element is used to define a section of the page that contains navigation links.
3. **<main>**: The main element is used to define the main content of a document.
4. **<article>**: The article element is used to define an independent piece of content within a document.
5. **<footer>**: The footer element is used to define a footer for a document or a section of a document.

By using these semantic elements, you can make your HTML code more descriptive and easier to understand.

That's it for this chapter! In the next chapter, we'll dive deeper into CSS and learn how to style your web pages.

Introduction to CSS

Now that you've learned the basics of HTML, it's time to move on to CSS. In this chapter, we'll introduce you to the basics of CSS and show you how to use it to style your web pages.

CSS stands for Cascading Style Sheets. CSS is used to style web pages, providing the visual design elements such as fonts, colors, and layouts. With CSS, you can control the appearance of every aspect of a web page, from the font size and color of text to the position and size of images.

CSS works by selecting HTML elements and applying styles to them. To apply a style to an element, you'll use a CSS rule. A CSS rule consists of a selector and one or more declarations.

Here's an example of a basic CSS rule:

```
p {
  color: red;
}
```

In this example, the selector is "p," which selects all paragraph elements on the page. The declaration is "color: red," which sets the text color of the selected elements to red.

CSS Selectors

CSS selectors are used to select HTML elements that you want to style. There are several types of selectors, including:

1. Element selectors: Selects elements based on their element type.

```
p {
  /* styles applied to all paragraph elements */
}
```

2. Class selectors: Selects elements based on their class attribute.

```
.my-class {
  /* styles applied to all elements with class="my-class" */
}
```

3. ID selectors: Selects elements based on their ID attribute.

```
#my-id {
  /* styles applied to the element with id="my-id" */
}
```

CSS Properties

CSS properties are used to set the visual styles of selected elements. There are many CSS properties available, including:

1. Font properties: Set the font family, size, weight, and style of text.

```
font-family: "Helvetica Neue", Arial, sans-serif;
font-size: 16px;
font-weight: bold;
font-style: italic;
```

2. Color properties: Set the color of text, backgrounds, and borders.

```
color: #333;
background-color: #fff;
border: 1px solid #ccc;
```

3. Layout properties: Set the position, size, and spacing of elements.

```
width: 50%;
height: 200px;
margin: 10px;
padding: 20px;
position: absolute;
top: 0;
left: 0;
```

These are just a few examples of the many CSS properties available.

In the next chapter, we'll dive deeper into CSS and learn how to use CSS selectors, properties, and values to create more advanced styles for your web pages.

Styling with CSS

In the previous chapter, we introduced you to the basics of CSS, including selectors and properties. In this chapter, we'll dive deeper into CSS and show you how to use CSS to create more advanced styles for your web pages.

CSS Selectors

CSS selectors are used to select HTML elements that you want to style. There are several types of selectors, including:

1. Descendant selectors: Selects elements that are descendants of other elements.

```
ul li {
  /* styles applied to all li that are descendants of a ul element */
}
```

2. Child selectors: Selects elements that are direct children of other elements.

```
ul > li {
  /* styles applied to all li that are direct children of ul element */
}
```

3. Attribute selectors: Selects elements based on their attributes.

```
a[href="<https://www.example.com>"] {
  /* styles applied to all a elements with
href="<https://www.example.com>" */
}
```

CSS Properties

CSS properties are used to set the visual styles of selected elements. There are many CSS properties available, including:

1. Background properties: Set the background color, image, and position of elements.

```
background-color: #fff;
background-image: url('image.jpg');
background-position: center;
```

2. Text properties: Set the font, size, color, and alignment of text.

```
font-family: "Helvetica Neue", Arial, sans-serif;
font-size: 16px;
color: #333;
text-align: center;
```

3. Border properties: Set the width, style, and color of borders.

```
border: 1px solid #ccc;
border-radius: 5px;
```

4. Layout properties: Set the position, size, and spacing of elements.

```
width: 50%;
height: 200px;
margin: 10px;
padding: 20px;
position: absolute;
top: 0;
left: 0;
```

CSS Units

CSS units are used to specify sizes and positions in CSS. There are several types of units, including:

1. Pixels (px): A pixel is a unit of measurement based on the screen resolution.

```
width: 200px;
```

2. Percentages (%): A percentage is a unit of measurement based on the size of the parent element.

```
width: 50%;
```

3. Em (em): An em is a unit of measurement based on the size of the current font.

```
font-size: 1.5em;
```

4. Rem (rem): A rem is a unit of measurement based on the size of the root font.

```
font-size: 1.5rem;
```

These are just a few examples of the many CSS units available.

In the next chapter, we'll learn how to create CSS layouts using the box model and positioning properties.

CSS Layouts

In the previous chapters, we introduced you to the basics of HTML and CSS, including selectors and properties. In this chapter, we'll dive deeper into CSS layouts and show you how to use CSS to create page layouts.

The Box Model

The box model is a fundamental concept in CSS layout. Every element in a web page is a rectangular box, and the box model defines how the size and position of each box are calculated.

The box model consists of four parts:

1. Content: The actual content of the box, such as text, images, or other elements.
2. Padding: The space between the content and the edge of the box.
3. Border: The line around the edge of the box.
4. Margin: The space between the border of the box and other elements on the page.

Here's an example of how the box model works:

```
.box {
  width: 200px;
  height: 100px;
  padding: 20px;
  border: 1px solid #ccc;
  margin: 10px;
}
```

In this example, the width and height of the box are set to 200px and 100px, respectively. The padding is set to 20px, the border is set to 1px solid #ccc, and the margin is set to 10px.

Positioning

CSS positioning is used to set the position of elements on the page. There are several types of positioning, including:

1. Static: The default positioning of elements. Elements are positioned according to the normal flow of the page.
2. Relative: Elements are positioned relative to their normal position in the page flow.

```
.box {
  position: relative;
  top: 10px;
  left: 10px;
}
```

In this example, the box is positioned 10 pixels down and 10 pixels to the right of its normal position.

3. Absolute: Elements are positioned relative to their nearest positioned ancestor.

```
.box {
  position: absolute;
  top: 50px;
  left: 50px;
}
```

In this example, the box is positioned 50 pixels down and 50 pixels to the right of its nearest positioned ancestor.

4. Fixed: Elements are positioned relative to the browser window.

```
.box {
  position: fixed;
  top: 0;
  left: 0;
}
```

In this example, the box is positioned at the top left corner of the browser window.

CSS Grid

CSS Grid is a powerful layout tool that allows you to create complex, multi-column layouts with ease. With CSS Grid, you can create rows and columns and place elements anywhere within them.

Here's an example of a basic CSS Grid layout:

```
.container {
  display: grid;
  grid-template-columns: 1fr 1fr;
  grid-template-rows: auto;
  grid-gap: 10px;
}

.box {
  grid-column: 1 / 3;
  grid-row: 1;
}
```

In this example, we've created a grid with two columns and one row. The grid-gap property sets the spacing between the columns. The box element is placed in the first and second columns of the first row.

These are just a few examples of the many CSS layout techniques available.

In the next chapter, we'll learn how to use CSS to create responsive designs that work on different screen sizes and devices.

Responsive Design with CSS

In the previous chapters, we introduced you to the basics of HTML and CSS, including selectors, properties, and layouts. In this chapter, we'll dive deeper into CSS and show you how to use it to create responsive designs that work on different screen sizes and devices.

Responsive Design

Responsive design is an approach to web design that focuses on creating websites that look good and function well on a wide range of devices, including desktops, laptops, tablets, and smartphones. With responsive design, the layout and design of a website adapt to the screen size and resolution of the device on which it is being viewed.

To create a responsive design, you'll use CSS media queries. Media queries are used to apply CSS styles based on the characteristics of the device being used to view the website, such as screen size, resolution, and orientation.

Here's an example of a basic media query:

```
@media (max-width: 600px) {
  .box {
    width: 100%;
  }
}
```

In this example, we're using a media query to set the width of the .box element to 100% when the screen width is less than or equal to 600px. This ensures that the .box element will take up the full width of the screen on smaller devices.

Viewport Units

Viewport units are a type of CSS unit that are based on the size of the viewport, rather than the size of the element being styled. There are two types of viewport units:

1. vw: One viewport width unit is equal to 1% of the width of the viewport.

   ```
   .box {
     width: 50vw;
   }
   ```

   In this example, the width of the .box element will be set to 50% of the width of the viewport.

2. vh: One viewport height unit is equal to 1% of the height of the viewport.

   ```
   .box {
     height: 50vh;
   }
   ```

   In this example, the height of the .box element will be set to 50% of the height of the viewport.

Flexible Box Layout

Flexible Box Layout, also known as Flexbox, is a layout mode in CSS that is designed to make it easier to create responsive designs. With Flexbox, you can create flexible and dynamic layouts that adapt to the size of the screen and the content within them.

Here's an example of a basic Flexbox layout:

```
.container {
  display: flex;
  flex-wrap: wrap;
  justify-content: space-between;
  align-items: center;
}

.box {
  flex-basis: 200px;
}
```

In this example, we're using Flexbox to create a row of boxes. The flex-wrap property is set to wrap, which allows the boxes to wrap onto a new row when the screen size is too small to fit them all on one row. The justify-content property is set to space-between, which evenly distributes the boxes along the row. The align-items property is set to center, which centers the boxes vertically within the row. The flex-basis property is set to 200px, which sets the initial size of each box.

These are just a few examples of the many responsive design techniques available in CSS.

In the next chapter, we'll learn how to use CSS frameworks and libraries to streamline the process of creating responsive designs.

CSS Frameworks and Libraries

In the previous chapters, we introduced you to the basics of HTML and CSS, including selectors, properties, and layouts. We also covered responsive design techniques using media queries, viewport units, and Flexbox. In this chapter, we'll introduce you to CSS frameworks and libraries that can help streamline the process of creating responsive designs.

CSS Frameworks

CSS frameworks are pre-written CSS styles and code snippets that you can use to quickly create the layout and design of a website. CSS frameworks can save you time and effort by providing a starting point for your design, as well as built-in responsiveness and cross-browser compatibility.

Here are some popular CSS frameworks:

1. Bootstrap: Bootstrap is a popular CSS framework that includes a wide range of pre-designed components and styles, including navigation bars, forms, buttons, and typography.
2. TailwindCSS: TailwindCSS is another popular CSS framework that includes a grid system, responsive typography, and pre-designed components for things like navigation menus, forms, and buttons.
3. Bulma: Bulma is a lightweight CSS framework that emphasizes simplicity and modularity. It includes a responsive grid system and pre-designed components like navigation menus, forms, and buttons.

CSS Libraries

CSS libraries are collections of pre-written CSS styles and code snippets that you can use to add specific functionality to your website. CSS libraries can save you time and effort by providing pre-built styles and functionality for things like animations, sliders, and tooltips.

Here are some popular CSS libraries:

1. Animate.css: Animate.css is a library of pre-written CSS styles that you can use to add animations to your website.
2. Slick: Slick is a library of pre-written CSS styles and JavaScript code that you can use to create responsive sliders and carousels.
3. Tooltipster: Tooltipster is a library of pre-written CSS styles and JavaScript code that you can use to create tooltips and popovers.

Using CSS frameworks and libraries can help you create responsive designs quickly and easily. However, it's important to remember that these tools should be used as a starting point, and you should still customize the design to meet the specific needs of your website.

In the next chapter, we'll wrap up our introduction to HTML and CSS and discuss best practices for writing clean and maintainable code.

Best Practices for Writing Clean and Maintainable Code

In the previous chapters, we introduced you to the basics of HTML and CSS, including selectors, properties, layouts, and responsive design techniques. In this final chapter, we'll discuss best practices for writing clean and maintainable code.

Clean Code

Clean code is code that is easy to read, easy to understand, and easy to maintain. Here are some tips for writing clean code:

1. Use descriptive names: Use descriptive names for variables, classes, and functions. This makes the code easier to read and understand.
2. Keep it simple: Keep your code as simple as possible. Avoid complex code and unnecessary complexity.
3. Comment your code: Use comments to explain your code and document how it works.
4. Use consistent formatting: Use consistent formatting for your code, including indentation, spacing, and line breaks.
5. Avoid redundancy: Avoid duplicating code or using unnecessary repetition.

Maintainable Code

Maintainable code is code that is easy to maintain and update over time. Here are some tips for writing maintainable code:

1. Use a consistent coding style: Use a consistent coding style to make it easier to read and understand your code.
2. Organize your code: Organize your code into logical sections and separate different parts of your code with comments or blank lines.
3. Use version control: Use a version control system like Git to keep track of changes to your code and collaborate with others.
4. Test your code: Test your code thoroughly to catch errors and ensure that it works as expected.
5. Use documentation: Use documentation to explain how your code works, how to use it, and how to modify it.

Conclusion

HTML and CSS are essential tools for building modern, responsive websites. By following best practices for writing clean and maintainable code, you can create websites that are easy to

read, easy to understand, and easy to maintain over time. We hope this introduction to HTML and CSS has been helpful, and we wish you the best of luck in your web development journey.

Thank you for reading this book on HTML and CSS. We hope that you found it helpful and informative. This book has covered a wide range of topics related to HTML and CSS, including the basics of HTML, CSS selectors and properties, layouts, responsive design, and best practices for writing clean and maintainable code.

While this book is a great starting point, there is always more to learn when it comes to web development. We encourage you to continue learning and exploring new techniques and technologies.

If you have any questions or feedback on this book, please feel free to reach out to us. We're always happy to hear from our readers.

Thank you again for reading, and we wish you the best of luck in your web development journey!