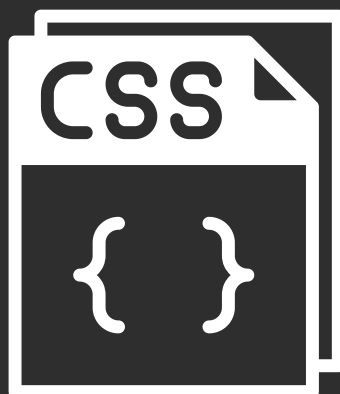
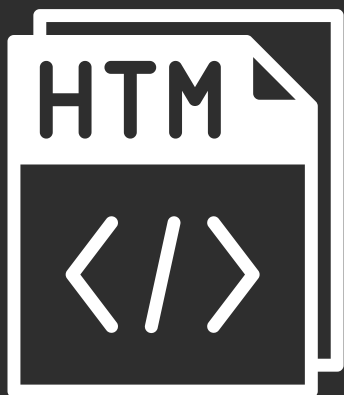


HTML & CSS

A Practical Guide



By Eric Hu

www.ericsdevblog.com

Introduction

HTML Basics

What is HTML?

Prepare your computer for web development

The structure of an HTML document

Elements and attributes

Headings and paragraphs

Formatting elements

Styling HTML elements

Links

Lists

Tables

Images and file paths

Forms

Layout elements

Block elements vs. inline elements

The head section

Conclusion

CSS Basics

What is CSS

How to select HTML elements

The `class` and `id` selectors

The combinator selectors

The pseudo-selectors

Other selectors

Defining colors in CSS

RGB color

HEX color

HSL color

Working with typography

Text alignment

Text decoration

Text spacing

Using a font

Customizing font

Conclusion

Advanced HTML and CSS

How are the elements displayed

Border, margin, and padding

Resizing elements

Deal with content overflow

Box sizing

Functions and variables

Applying filters

Z-index and backdrop filters

CSS transitions

CSS animations

Styling backgrounds

Styling lists

Styling tables

Conclusion

How to Position and Arrange Content Using CSS

Display types

Inline

Block

Inline block

display vs. visibility

How to position elements

Relative position

Fixed position

Absolute position

Sticky position

Transforming elements

How to align elements

Horizontally center an element

Vertically center an element with padding

Vertically center an element using position and transform

Left and right align elements

Creating grids

Grid columns and rows

Grid gaps

Grid items

Grid alignment

Vertical alignment

Horizontal alignment

Flexbox layout

A practical example

Conclusion

Responsive Design

Setting the viewport

Media queries and breakpoints

Creating responsive page layouts

Using flexbox

CSS grids

Legacy layout method

Responsive layouts without media queries

Responsive images

Responsive typography

Conclusion

Recreating YouTube Using HTML and CSS

Creating the page layout

The HTML document

Navbar layout

Sidebars

The content section

Building the navigation bar

Building the sidebar

Building the video card component

Conclusion

Some CSS Tools and Frameworks

CSS preprocessor

PostCSS

Popular CSS frameworks

HTML & CSS Best Practices

1. Use semantic HTML tags
2. Maintain clear indentation and formatting
3. Comment your code
4. Ensure accessibility and responsiveness
5. The `<head>` section is important
6. Separate your files
7. Choose meaningful class and ID names
8. Keep selectors simple
9. Keep a consistent style
10. Pay attention to browser compatibility

Conclusion

Introduction

Welcome to “HTML & CSS: A Practical Guide”, a comprehensive course designed to equip you with the essential skills needed to create adaptive and functional web pages.

In this course, we embark on an exciting journey through the fundamental building blocks of web development: HTML (HyperText Markup Language) and CSS (Cascading Style Sheets). Whether you're a complete beginner or someone looking to refine their existing knowledge, the practical examples in this course will provide you with hands-on experience, enabling you to design and structure web content effectively, creating seamless and responsive designs that captivate users across an array of devices.

Let's unlock the world of web development and lay the groundwork for your successful journey in creating stunning and user-friendly websites.

HTML Basics

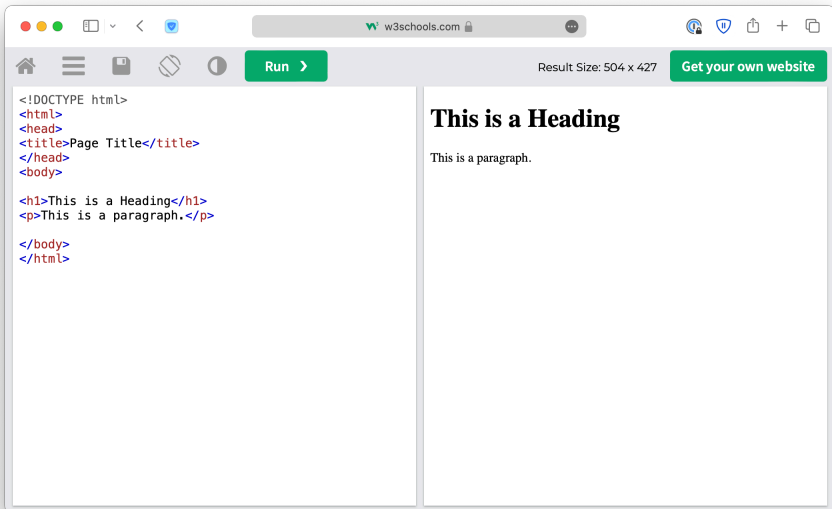
HTML and CSS are the most fundamental building blocks of a webpage. It is your first step towards becoming a web developer. HTML (HyperText Markup Language) defines the structure and content of the webpage, while CSS (Cascading Style Sheets) defines the webpage's presentation and appearance. Together, they form the colorful webpages you see today.

This course on HTML and CSS will cover everything you need to know about these technologies. By the end of this course, you will be able to create webpages that are visually appealing on devices of all sizes. It doesn't matter if you're a beginner or have some experience since this course is designed to help you learn and grow. Don't worry if you don't have any prior knowledge, as we'll start from the basics and work our way up together.

What is HTML?

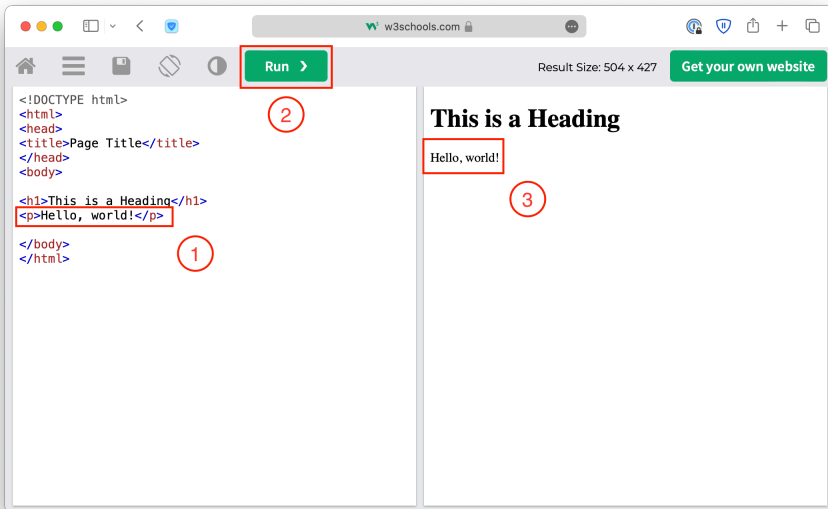
HTML, short for HyperText Markup Language, is the standard markup language used to create webpages. It defines the structure and content of webpages using elements and tags, such as headings, paragraphs, images, links, forms, and more. These elements instruct web browsers on how to display the content of a webpage.

To start writing HTML code, you can head over to [W3Schools' online HTML editor](#).



On the left side, you will find the HTML source code, which is essentially the blueprint for what will be displayed. The browser takes this blueprint and transforms it into the webpage you see on the right side.

You can modify the source code directly to see how it affects the displayed webpage. Once you've made your desired changes, simply click the **Run** button, and the right panel will reflect the alterations.



Congratulations! Now you are officially a programmer capable of building webpages. However, this is only the beginning, there is still a lot more to be done before you can create fully functional and visually pleasing webpages.

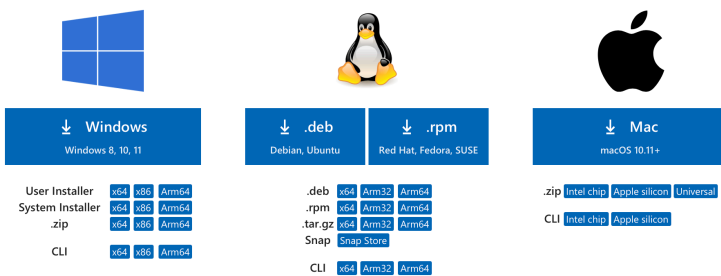
Prepare your computer for web development

First of all, you must ensure your computer is ready for web development. A basic online editor is not going to be enough this time. To get started, make sure you have a web browser installed. Any popular web browser on the market, such as Google Chrome, Microsoft Edge, Safari, or Firefox, should be sufficient for this course. You may download and install the browser of your choice from the linked websites.

In addition, you'll need a code editor to write and edit your code. Visual Studio Code is a great option for beginners, and it's the most popular code editor out there. Simply download the appropriate installer for your operating system from their official website.

Download Visual Studio Code

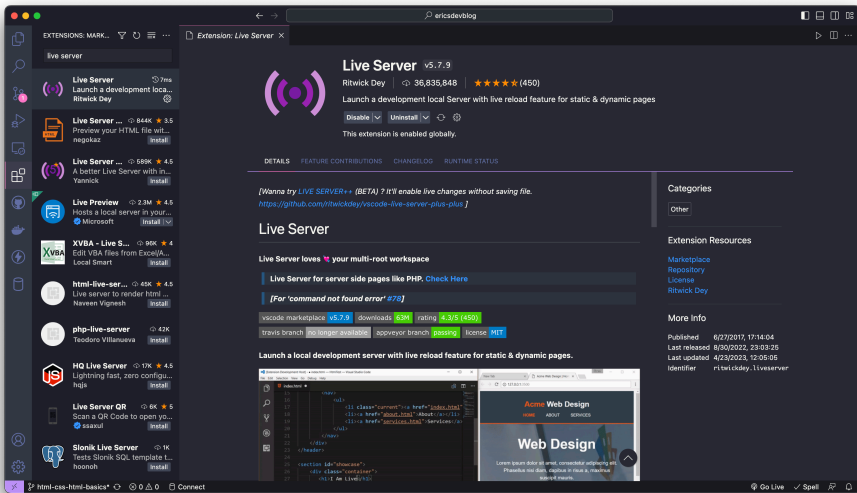
Free and built on open source. Integrated Git, debugging and extensions.



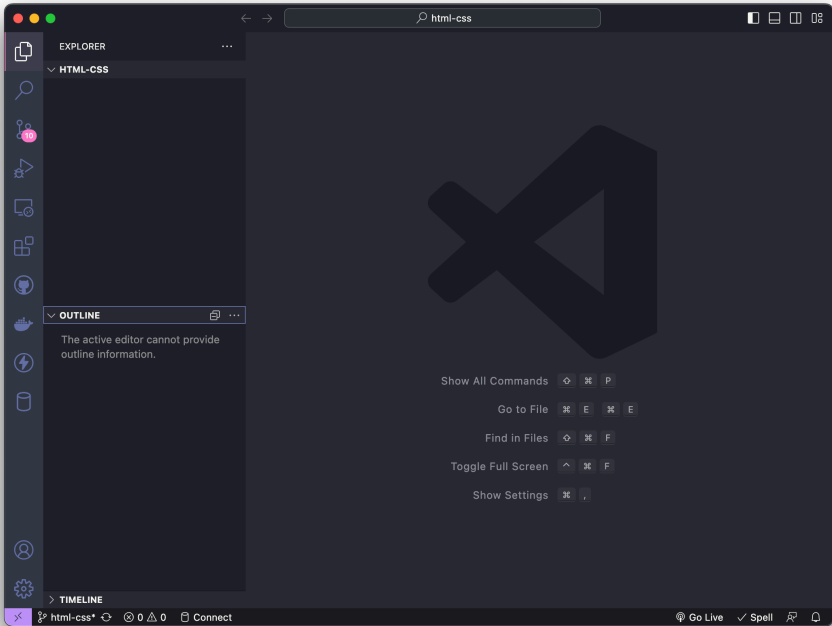
The image displays the download options for Visual Studio Code, organized into three main sections: Windows, Linux, and Mac. Each section features a platform icon (Windows logo, Tux penguin, and Apple logo) above a blue button with a download arrow and the platform name. Below these buttons are lists of installation methods and their corresponding architecture options.

| Platform | Installation Method | Architecture Options | |
|----------|---------------------|--------------------------------------|-------------------|
| Windows | User Installer | x64, x86, Arm64 | |
| | System Installer | x64, x86, Arm64 | |
| | zip | x64, x86, Arm64 | |
| | CLI | x64, x86, Arm64 | |
| Linux | .deb | Debian, Ubuntu | x64, Arm32, Arm64 |
| | | Red Hat, Fedora, SUSE | x64, Arm32, Arm64 |
| | .rpm | Red Hat, Fedora, SUSE | x64, Arm32, Arm64 |
| | | Snap Store | x64, Arm32, Arm64 |
| | .tar.gz | x64, Arm32, Arm64 | |
| | CLI | x64, Arm32, Arm64 | |
| Mac | .zip | Intel chip, Apple silicon, Universal | |
| | CLI | Intel chip, Apple silicon | |

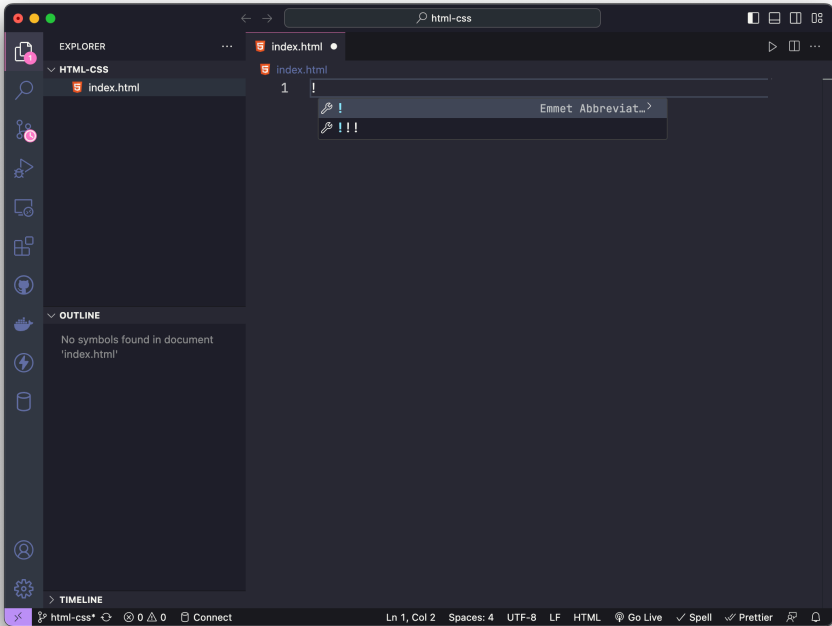
After you've installed VS Code, make sure to install the **Live Server** extension as well. Navigate to the **Extensions** tab on the left sidebar, and type in **Live Server** in the search box. From there, you'll be able to download and install the extension with ease.



This extension will create a local development server with the auto-reload feature. For example, create a new work directory and open it using VS Code.



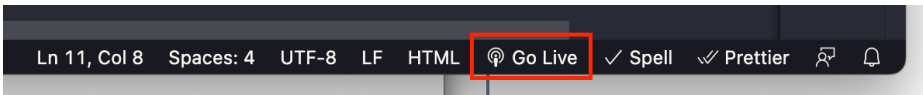
Create a new file named `index.html` under this directory. The `.html` extension indicates that this is an HTML document. Type in `!` in the VS Code editor, and you will see suggestions like this:



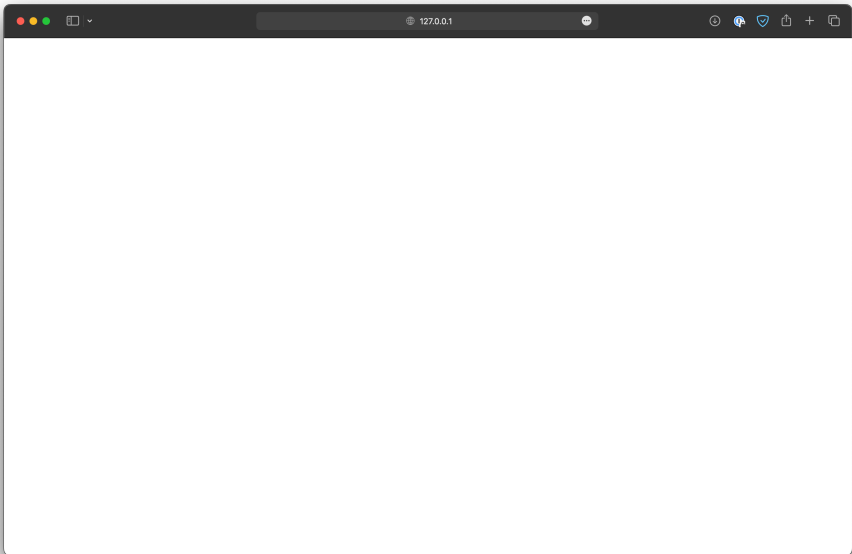
This is a shortcut that allows you to create HTML documents quickly. Select the first option, and the following code should be created.

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <meta charset="UTF-8">
5     <meta name="viewport" content="width=device-width,
6     initial-scale=1.0">
7     <title>Document</title>
8 </head>
9 <body>
10 </body>
11 </html>
```

Notice that at the bottom right corner of the VS Code window, there is a **Go Live** button.



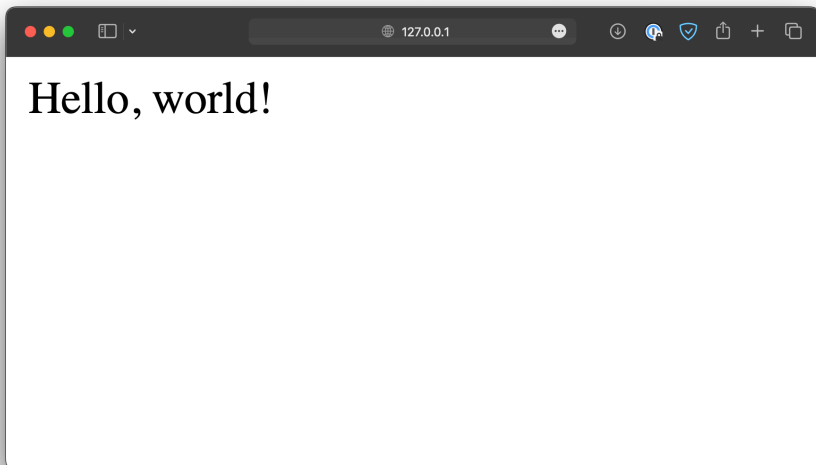
Clicking this button will activate the **Live Server** extension. A dev server will be started, hosting the `index.html` file you created.



Of course, right now, the file is still empty. Add something between the `<body>` and `</body>` tags.

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <meta charset="UTF-8">
5     <meta name="viewport" content="width=device-width,
6     initial-scale=1.0">
7     <title>Document</title>
8 </head>
9 <body>
10     Hello, world!
11 </body>
12 </html>
```

The webpage will be refreshed with the new content.



The structure of an HTML document

A typical HTML document always has the following structure:

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      . . .
5  </head>
6  <body>
7      . . .
8  </body>
9  </html>
```

The `<!DOCTYPE html>` tag defines the document type, and when a web browser encounters `<!DOCTYPE html>`, it understands that the page should be parsed and displayed according to the rules and specifications of HTML5, the latest version of HTML. This ensures that modern browsers interpret the webpage's content and layout correctly.

Everything else in the file should be enclosed inside an `<html>` element, defined by an opening tag (`<html>`) and a closing tag (`</html>`). `lang="en"` is called an attribute, which tells the browser and search engine that English is the primary language used for the content of this webpage.

Inside the `<html>` element, there are two child elements, `<head>` and `<body>`. `<head>` contains metadata and other information about the HTML document. This information will not be displayed in the browser but is often used by search engines for SEO (Search Engine Optimization) purposes. `<body>`, on the other hand, contains the main content of the webpage that is visible to the users, and for that reason, it is also the part of the HTML file we are going to focus on in this course.

Elements and attributes

Let's take a closer look at the example and notice that the HTML document is made up of different elements in a nested structure. In HTML, most elements have both an opening tag and a closing tag:

```
1 <tag>. . .</tag>
```

In this example, `<tag>` is called the opening tag, and `</tag>` is the closing tag, and together, they form an HTML element. The element could hold content between the opening and closing tags.

```
1 <tag>Hello, world!</tag>
```

The element can also contain other elements, forming a nested structure.

```
1 <tag>
2   <tag>. . .</tag>
3   <tag>
4     <tag>. . .</tag>
5   </tag>
6 </tag>
```

Inside the opening tag, you can define attributes, which are used to specify additional information about the element, such as its `class`, `id`, and so on.

```
1 <tag attribute="value">. . .</tag>
```

The attribute is usually in a key/value pair format, and the value must always be enclosed inside matching quotes (double or single).

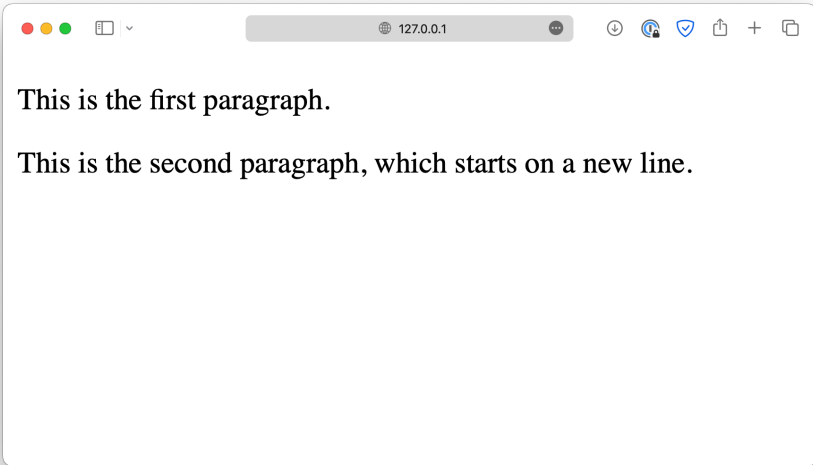
There are some exceptions to these general formats. For example, the `
` element, which creates a line break, does not need a closing tag. Some attributes, such as `multiple`, do not require a value. We will discuss these exceptions later in this course as we encounter specific examples.

However, you should remember that if an element does require a closing tag, then it should never be left out. In most cases, the webpage could still render correctly, but as the structure of your HTML document grows more complex, unexpected errors may occur.

Headings and paragraphs

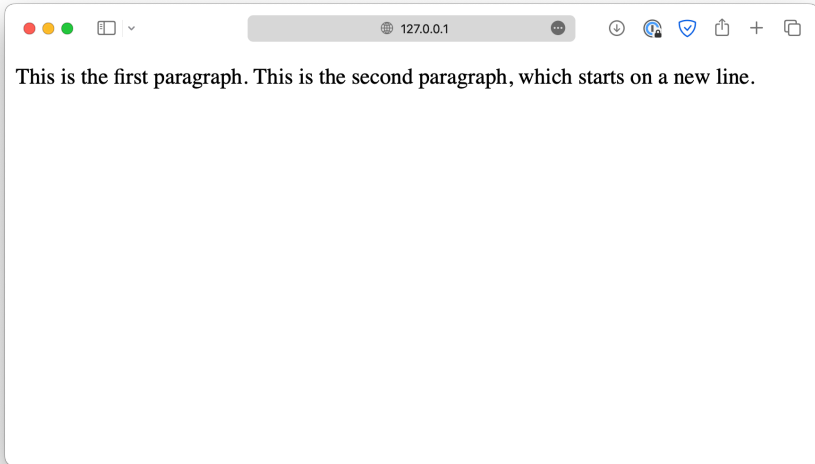
The paragraph is probably the most commonly used HTML element, defined by `<p></p>`. It is a block-level element, meaning each paragraph will start on a new line.

```
1 <body>
2   <p>This is the first paragraph.</p>
3   <p>This is the second paragraph, which starts on a new
   line.</p>
4 </body>
```



Without the `<p>` element, your browser will automatically ignore the extra white spaces and render the text in a single line.

```
1 <body>
2     This is the first paragraph.
3     This is the second paragraph, which starts on a new
   line.
4 </body>
```



You'll need to use the `
` element if you want a line break inside one paragraph. This is one of those HTML elements that does not require a closing tag.

```
1 <body>
2   <p>This is the first paragraph.<br>
3     This is the second paragraph, which starts on a new
4     line.</p>
5 </body>
```