ERIN GLASS

HOW TO BUILD A WEBSITE WITH HTML

ISBN 978-1-7358317-0-1

How To Build a Website with HTML

Erin Glass

DigitalOcean, New York City, New York, USA

2020-10

How To Build a Website with HTML

About DigitalOcean

DigitalOcean is a cloud services platform delivering the simplicity developers love and businesses trust to run production applications at scale. It provides highly available, secure and scalable compute, storage and networking solutions that help developers build great software faster. Founded in 2012 with offices in New York and Cambridge, MA, DigitalOcean offers transparent and affordable pricing, an elegant user interface, and one of the largest libraries of open source resources available. For more information, please visit https://www.digitalocean.com or follow @digitalocean on Twitter.

Introduction

About this Book

If you are interested in learning how to build and design websites, Hyper Text Markup Language (HTML) is a great place to start. This project-based tutorial series will introduce you to HTML and its methods by building a personal website using our demonstration site (below) as a model. Once you learn the basics, you will know how change the website's design and add personalized content. No prior coding experience is necessary to follow along the chapters in this book.

HTML is the standard markup language used to display documents in a web browser. First developed by Tim Berners Lee in 1990 while working at the European Organization for Nuclear Research (CERN), HTML was one of the key innovative technologies used to publish the world's first website on August 6, 1991. Thanks to a restoration project by CERN, you can now revisit the original website. Since that time, HTML has been significantly updated and expanded but its basic purpose to format and structure web pages remains the same.

Today, HTML is one of many tools used to build the web. Knowing how to write HTML will provide a strong foundation for your career as a web designer and prepare you to learn additional front-end web development skills like CSS and JavaScript.

In this book, you'll learn how to create and customize a website using common HTML tags and techniques. After finishing this book, you'll have a site ready to deploy to the cloud.

Prerequisites

A code editor like Visual Studio Code or Atom. For this book, we will be using Visual Studio Code as our default code editor but you may use any code editor you like. Certain instructions may need to be slightly modified if you use a different editor.

A web browser like Firefox or Chrome. We will be using Firefox as our default browser but you may use any browser you like. Certain instructions may need to be slightly modified if you use a different web browser. Two different profile photos, images, or avatars for personalizing your site (optional).

Once you have your prerequisites ready, you will be ready to set up your HTML project in the next chapter.

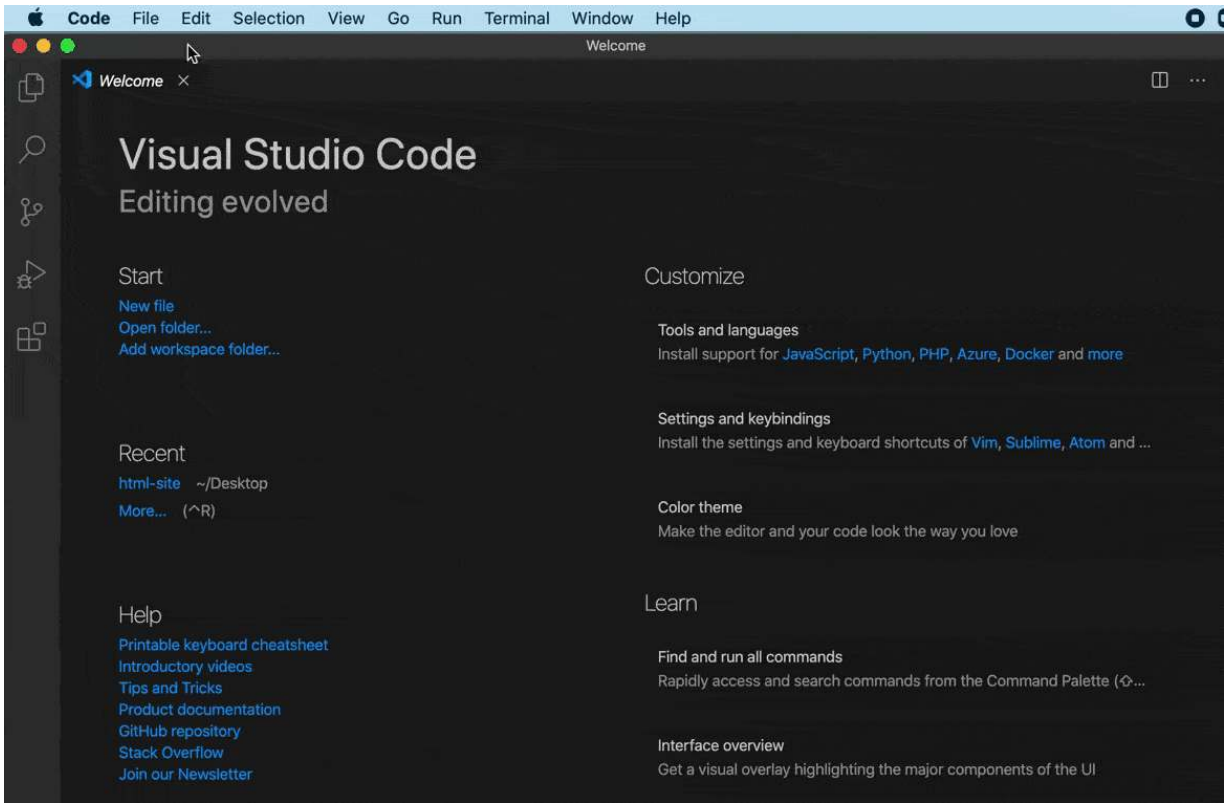
How To Set Up Your HTML Project

Written by Erin Glass

To explore HTML in practice and begin building an HTML website, we'll need to set up a new project using a text editor. This tutorial series uses Visual Studio Code, a free code editor available for Mac, Windows, or Linux, but you may use whichever code editor you prefer.

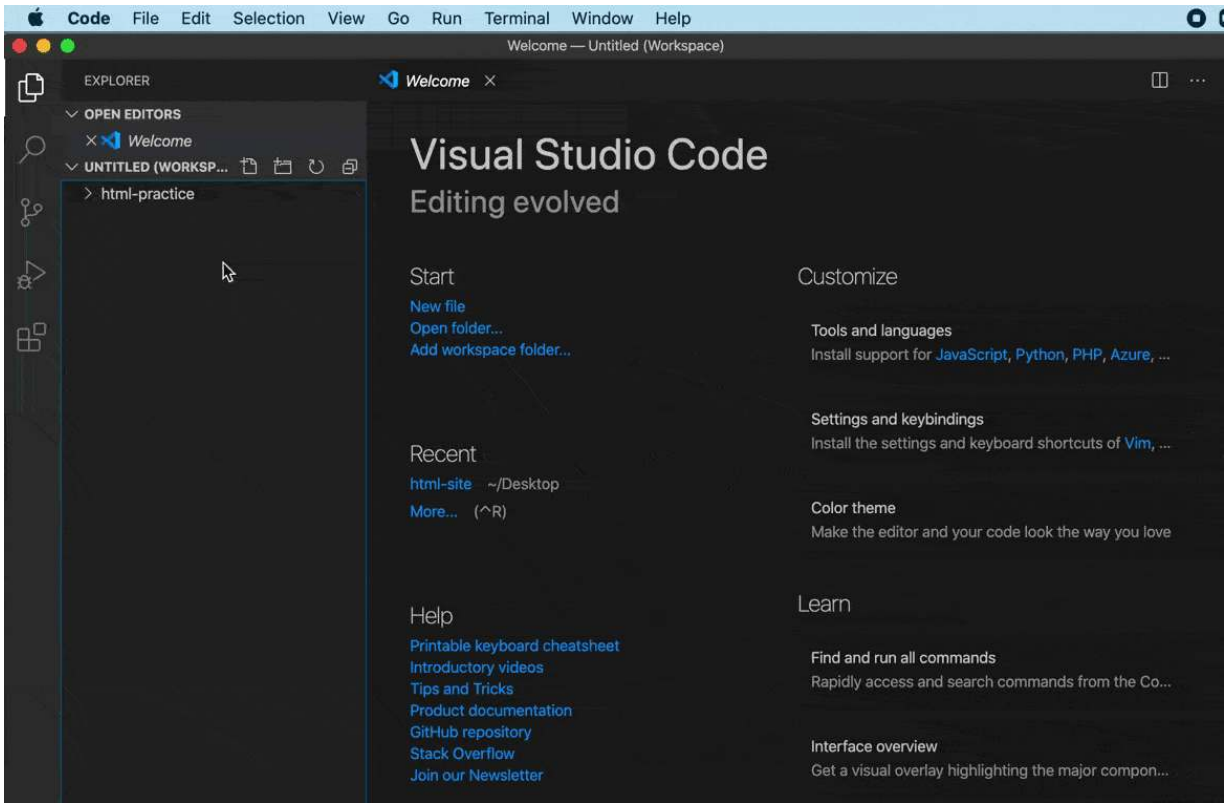
After opening your preferred text editor, open up a new project folder and name it `html-practice`. We'll use this folder to store all the files and folders we create in the course of this tutorial series.

To create a new project folder in Visual Studio Code, navigate to the "File" menu item in the top menu and select "Add Folder to Workspace." In the new window, click the "New Folder" button and create a new folder called `html-practice` as illustrated in the gif below:



Gif of how to add a project folder in Visual Studio Code

Next, create a new file called `index.html` inside the `html-practice` folder. We'll use this file through the tutorial series to experiment with HTML. If you are using Visual Studio Code, you can create a new file by using `Right Click`(on Windows) or `CTRL + Left Click` (on Mac) on the `html-practice` folder, selecting "New File", and creating the file `index.html` as illustrated in the gif below:



Gif of how to add a file in Visual Studio Code

You now have a project folder and file for exploring HTML. We'll return to this file in the tutorials ahead.

Debugging and Troubleshooting CSS and HTML

Before we get started with our HTML exercises, be aware that precision is important when writing HTML. Even an extra space or mistyped character can keep your code from working as expected.

If your HTML code is not rendering in the browser as intended, make sure you have written the code exactly. To troubleshoot errors, check for extra or missing spaces, missing or misspelled tags, and missing or incorrect punctuation or characters. Each time you change your code, make

sure to save your file before reloading it in the browser to check your results.

A Quick Note on Automatic HTML Support Features

Some code editors—such as the Visual Studio Code editor that we're using in this series—provide automatic support for writing HTML code. For Visual Studio Code, that support includes smart suggestions and auto completions. While this support is often useful, be aware that you might generate extra code that will create errors if you're not used to working with these support features. If you find these features distracting, you can turn them off in the code editor's preferences.

We are now ready to begin learning how the CSS language works. In the next tutorial, we'll begin exploring how CSS rules are used to control the style and layout of HTML content on a webpage.

How To View the Source Code of an HTML Document

Written by Erin Glass

This tutorial will introduce you to a basic HTML document and teach you how to view the source code of an HTML document in a browser.

HTML is used to mark up a document with instructions that tell a browser how to display and interpret the document's content. For example, HTML can tell the browser which text content should be interpreted as a heading and which text content should be interpreted as paragraphs. HTML is also used to add images and assign links to text and images. These instructions are communicated through HTML tags, which are written like this: `<tagname>`. Many, though not all tags, use an opening tag and closing tag to wrap around the content that they are used to modify.

To get a sense of how these tags are used, let's inspect a snippet of HTML code. The HTML code below shows how HTML tags are used to structure text and add links and images. Don't worry if you don't understand the tags immediately- we'll study those in the next tutorial.

```
<h1>Sammy's Sample HTML</h1>

<p>This code is an example of how HTML is written.
</p>

<p>It uses HTML tags to structure the text.</p>

<p>It uses HTML to add a <a
href="digitalocean.com/community">link</a>.</p>

<p>And it also uses HTML to add an image:</p>

<img src="https://html.sammy-
codes.com/images/small-profile.jpeg"/>
```

This HTML code is rendered in the browser as follows:

Sammy's Sample HTML

This code is an example of how HTML is written.

It uses HTML tags to structure the text.

It uses HTML to add a link.

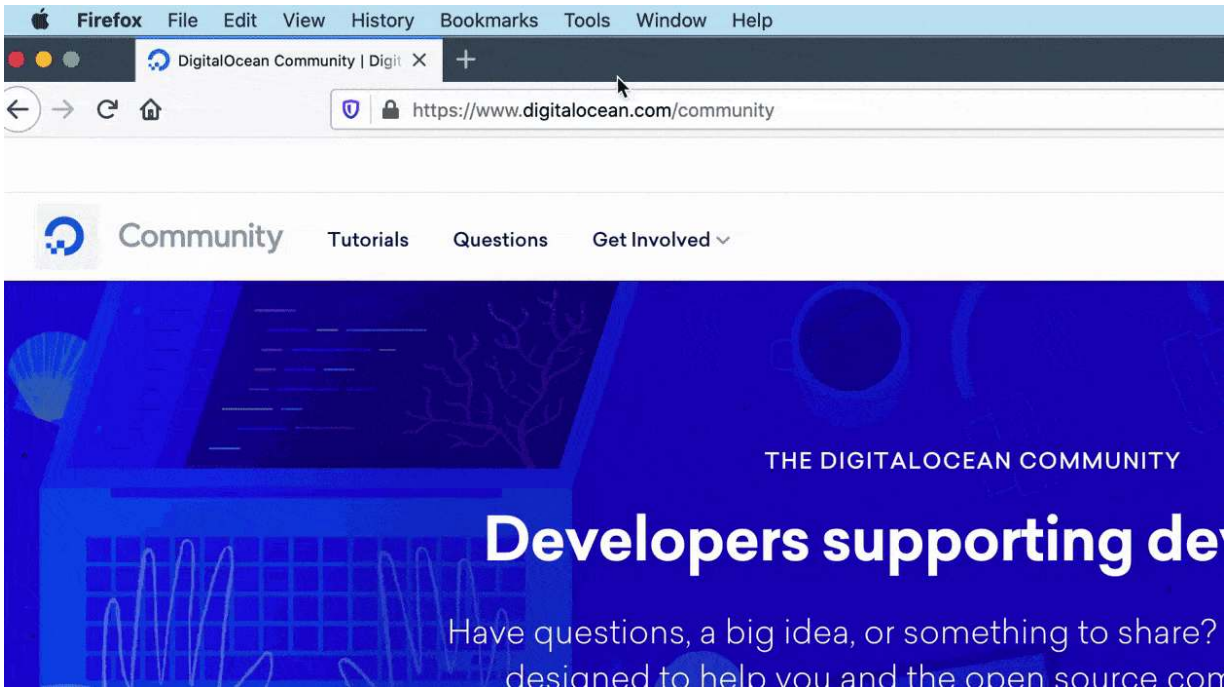
And it also uses HTML to add an image:

HTML Document

You should now have an understanding of how the HTML example code is rendered in a browser. Next, we will learn how to view the source code of any webpage using a browser tool.

Viewing the Source Code of a Webpage

Nearly every webpage you come across uses HTML to structure and display HTML pages. You can inspect the source code of any webpage by using a web browser like Firefox or Chrome. On Firefox, navigate to the "Tools" menu item in the top menu and click on "Web Developer/Page Source" like so:



Gif of how to inspect source code using Firefox

On Firefox, you can also use the keyboard shortcut `Command-U` to view the source code of a webpage.

On Chrome, the process is very similar. Navigate to the top menu item "View" and click on "Developer/View Source." You can also use the keyboard shortcut `Option-Command-U`.

Try inspecting the source code of the demo website that we will build in this tutorial series. You should receive a page with many more HTML tags than our example above. Don't be alarmed if it seems overwhelming. By the end of this tutorial series, you should have a better understanding of how to interpret HTML source code and how to use HTML to build and customize your own websites.

Note: As mentioned above, you can inspect the source code of any webpage using tools from the Firefox or Chrome web browser. Try

inspecting the code of a few of your favorite websites to get a sense of the underlying code that structures web documents. Though the source code of these sites will likely contain more languages than HTML, learning HTML first will help prepare you to learn additional languages and frameworks for creating websites later on if you wish.

You should now have a general understanding of the format of an HTML document and know how to inspect HTML source code using a browser tool. To better understand how HTML works, let's inspect its key components. In the next tutorial, we will learn more about HTML elements, the building blocks that are used to create HTML documents.

How To Use and Understand HTML Elements

Written by Erin Glass

HTML documents are composed of HTML elements. Most HTML elements contain content (such as text or an image) along with HTML tags that tell the browser how to interpret the content (such as a heading or paragraph text). HTML elements can be used to add structure, semantics, and formatting to different parts of an HTML document. An HTML element is often created—but not always—by opening and closing HTML tags, which wrap around a piece of content.

Below is an illustration that labels each of the parts of an HTML element:

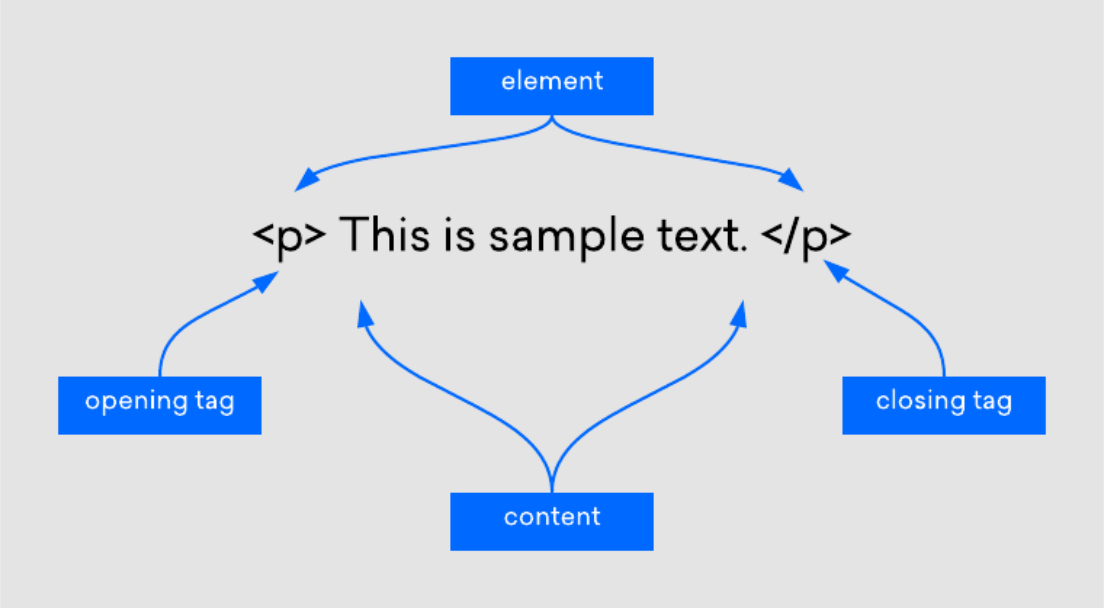


Diagram of an HTML element

Let's try exploring HTML in practice. Try pasting the following line into your "index.html" file that you created when Setting Up Your HTML Project:

```
<strong>My strong text</strong>
```

In this example, the tag adds strong formatting by enclosing the text with a pair of opening and closing tags. Note that closing tags are always denoted by a forward slash (/).

Note: You may note that the tag acts very similar to adding bold styling to the text. You can achieve the same styling effect by using the bold <b> tag, however the tag adds bold styling and semantic meaning that indicates the text is of strong importance. If you are using the bold styling because you want to note the importance of the text, make sure to use the tag, which will enable screen readers to announce their importance to the user.

Similarly, the emphasis tag adds italic styling and semantic meaning that indicates the text is emphasized. The italics tag <i> only adds the italic styling to the text. If you are using the italic styling because you want to emphasize the text, make sure to use the tag, which will enable screen readers to announce their emphasis to the user.

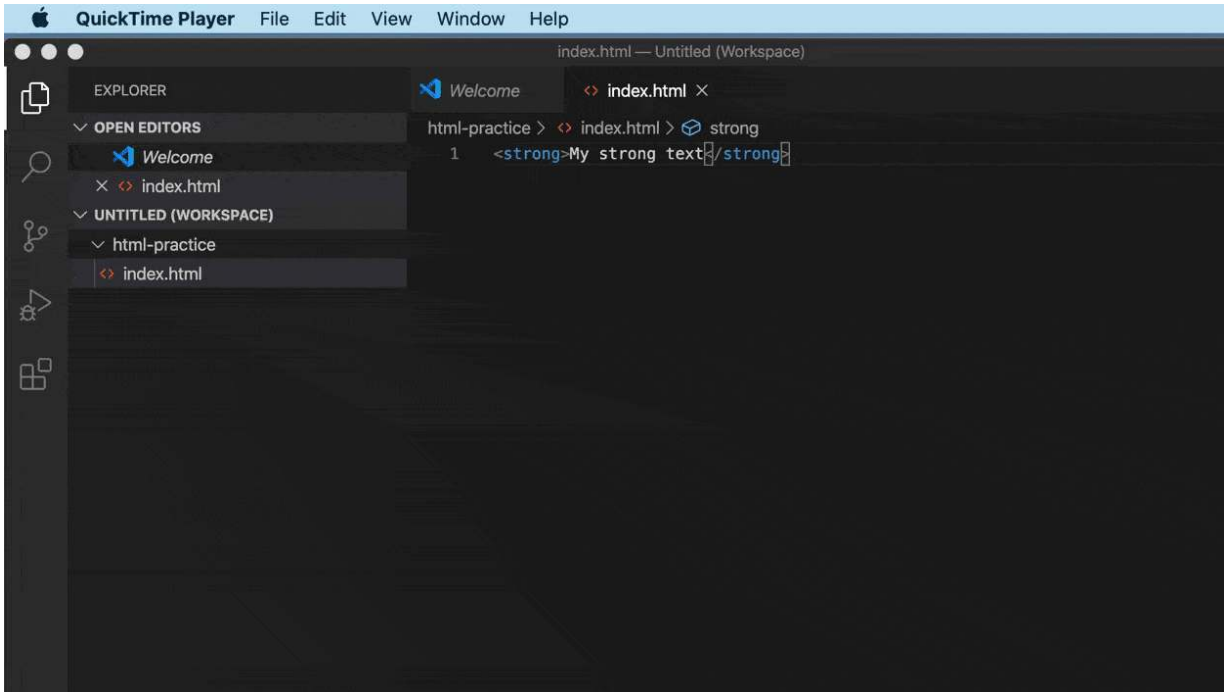
To check the results of this HTML code, we can load the "index.html" file in our browser. Though your file is not online, the browser will still be able to interpret the HTML file as if it were a web page document. Make sure to save your "index.html" file before loading it in the browser as only saved updates will be rendered.

How To View an Offline HTML File In Your Browser

You can view an offline HTML file in the browser in several ways:

- Drag and drop the file into your browser
- `CTRL + Left Click` (on Macs) or `Right Click` (on Windows) to open the file with a browser
- Copy the full path of your file and paste the file in your browser bar

If you are using the Visual Studio Code text editor, you can copy the file path using `CTRL + Left Click` (on Macs) or `Right Click` (on Windows) on the file "index.html" in the left hand panel and selecting "Copy Path." Then paste the path in your web browser as illustrated in the gif below:



Gif of how to copy a file path and load it in your browser

Note: To load the file in the browser, it's important that you copy the absolute path, (which refers to the file location relative to the root directory), rather than the relative path, (which refers to the file relative to the current working directory). In Visual Studio Code, "Copy Path" refers to the full file path. Be aware, however, that we will use relative paths of files in other parts of this tutorial.

After loading the file in your browser, you should receive a page that contains the following:

My bold text

Let's try experimenting with other HTML elements. On the next line of your "practice.html" file, try adding the `<em>` element, which adds emphasis.

```
<strong>My bold text</strong>
<em>My emphasized text</em>
```

Save the file and reload the file in the browser. You should receive something like this:

My bold text My emphasized text

The first phrase should be styled with strong formatting and the second phrase should be styled with emphasis. However, you may be surprised by the side-by-side placement of the two phrases. If you added `<em>My emphasized text</em>` to the second line of the text editor, you may have expected "My emphasized text" to show up on the line below "My bold text" in the web browser. However, certain HTML elements, such as `<strong>` and `<em>`, require you to specify line breaks with additional

HTML elements (if you desire lines breaks). We'll explain why in the next tutorial.

How To Use Inline-level and Block-level Elements in HTML

Written by Erin Glass

This tutorial will teach you the difference between inline-level and block-level elements in HTML and how they affect a piece of content's position on the page.

When arranging elements in an HTML document, it's important to understand how these elements take up space on a webpage. Certain elements can take up much more space on the webpage than the content they contain. Understanding the behavior of different element types will help you anticipate how they affect the position of other elements on the page.

In general, there are two different types of elements—inline-level elements and block-level elements—whose spacing defaults behave differently from one another. Below, we'll describe how the default settings of these elements determine their position on the webpage and relative to the position of nearby elements.

Inline-level Elements

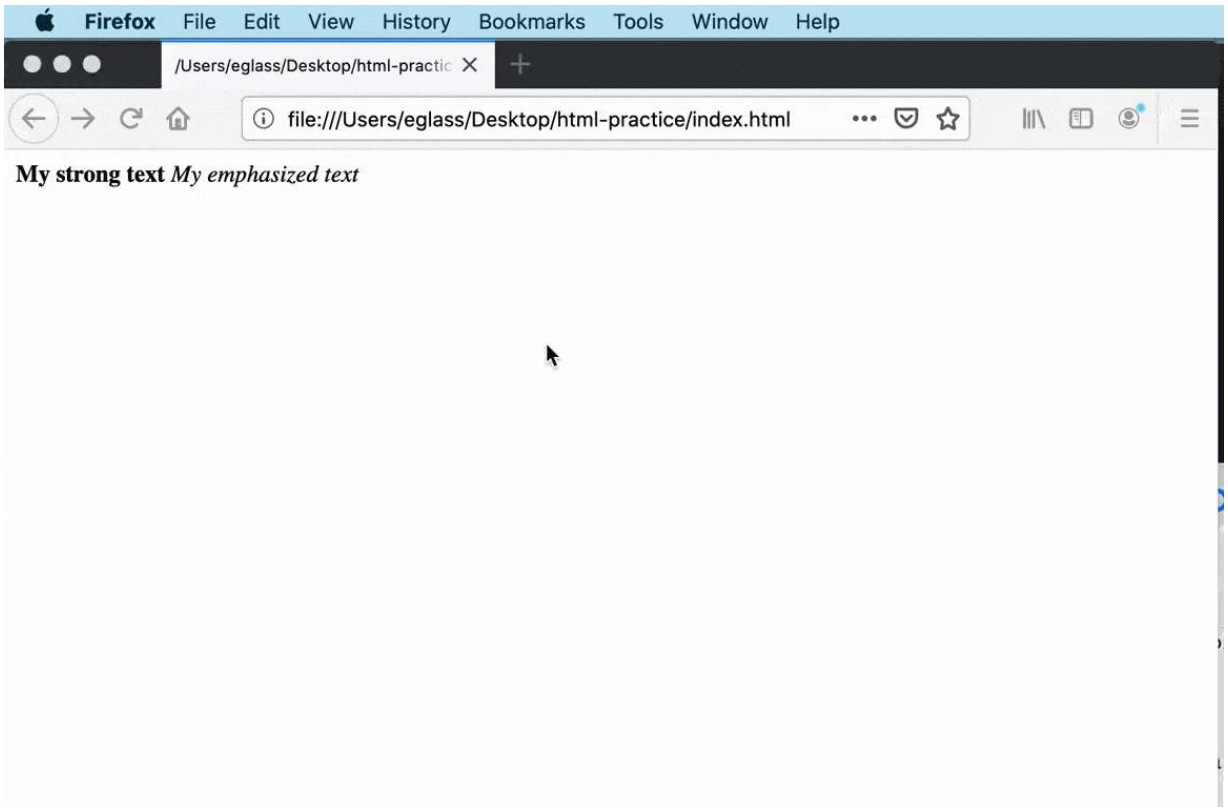
Inline elements are elements whose horizontal width is determined by the width of the content they contain. The `<strong>` element and the `<em>` element we covered in the last tutorial are both examples of inline elements.

We can use Firefox's Web Developer Inspector to inspect the size of HTML elements on a webpage. (If you are using Chrome, you can use the

Developer Inspect Elements tool instead but this tutorial will give instructions for Firefox's Web Developer tool.)

Return to the `index.html` file that you loaded in your browser. If you need to reload it and don't remember how, refer to the instructions Loading an HTML File in your Browser from the last tutorial.

Then navigate to the Tools menu item in the top menu bar and select "Web Developer/Inspector." Selecting this menu item will open up the Inspector interface that allows you to inspect the HTML and CSS of a webpage. Next, hover your cursor over the text `My strong text`, which should highlight the text in light blue. This highlighting shows the full extent of the space occupied by the element that your cursor is hovering over. As you may have expected, the element's occupied space is just large enough to contain its text content:



This gif illustrates the process of using the Inspector tool as described in the paragraph above.

Unlike block-level elements, which we'll cover in the next section, inline elements do not take up their own line of horizontal space. Thus, inline elements will rest side by side on a webpage if you do not specify a break with an additional HTML element, such as the line break `<br>` element. This sizing default is often convenient, as you may want to mark up single words in a paragraph with an inline element such as `<strong>` or `<em>` without pushing subsequent text to the next line.

Try adding the `<br>` tag in between your two lines of code in the `index.html` file. (You will need to return to your file in the text editor.) Note that the `<br>` element only requires an opening tag and does not wrap around any text:

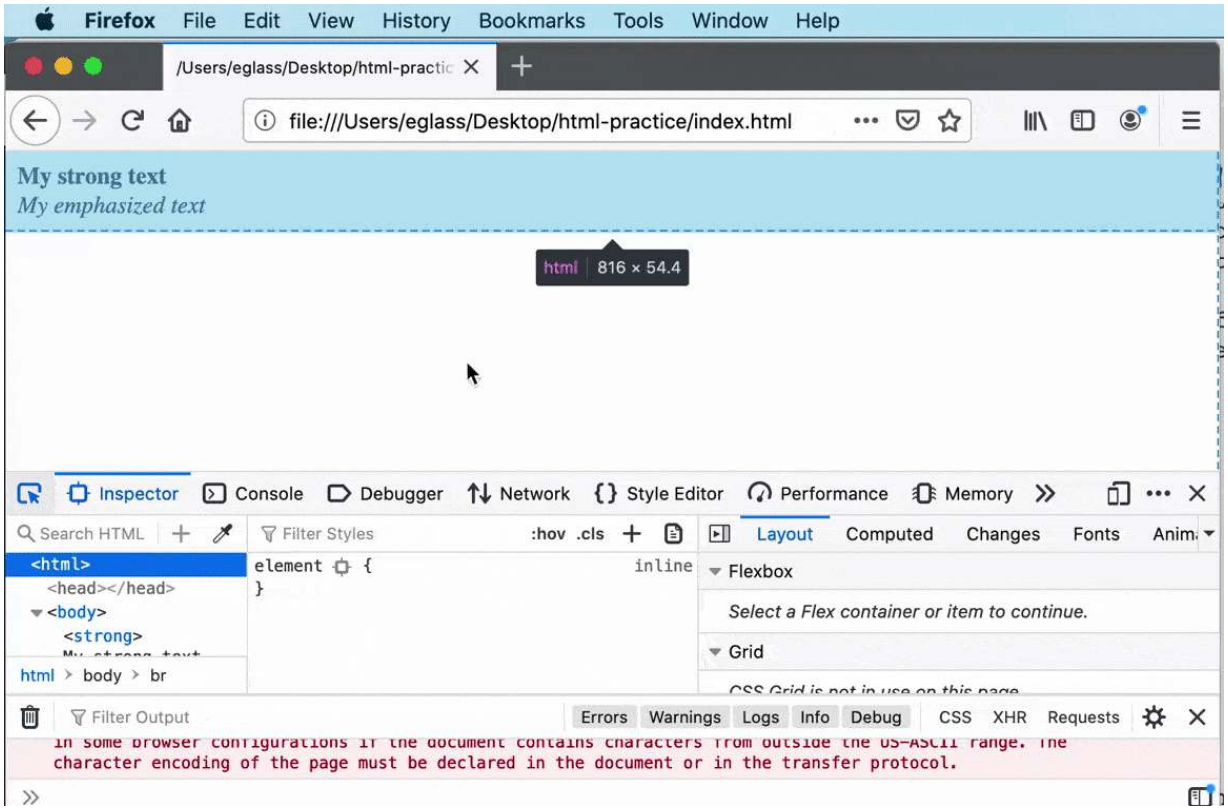
```
<strong>My strong text</strong>
<br>
<em>My emphasized text</em>
```

Save and reload the document in your browser to check your results. You should receive something like this:

My strong text My emphasized text

Your two phrases should now be on separate lines as they are now separated by the line break element `<br>`.

If you use the Firefox Web Developer Inspector to check the size of the elements, you can confirm that the width of each of the text elements is still determined by the width of the text content:



Block-level Elements

Block-level elements behave differently than inline-level elements in that they take up an entire line of horizontal space on a webpage. This means that they automatically start on a new line and that they automatically push subsequent elements onto a new line as well.

For example, the HTML heading elements (`<h1>` through `<h6>`) are block-level elements that automatically place their content onto a new line and push any content that follows onto a new line. Each of these six heading elements represent a different heading size.

Let's study how this works in practice. At the bottom of your `index.html` file, try pasting in the highlighted code snippet:

```
<strong>My strong text</strong>
<br>
<em>My emphasized text</em>

<h1>Heading 1</h1>
<h2>Heading 2</h2>
<h3gt;Heading 3</h3>
<h4>Heading 4</h4>
<h5>Heading 5</h5>
<h6>Heading 6</h6>
```

Save your file and reload it in the browser. You should receive something like this:

My strong text
My emphasized text

Heading 1

Heading 2

Heading 3

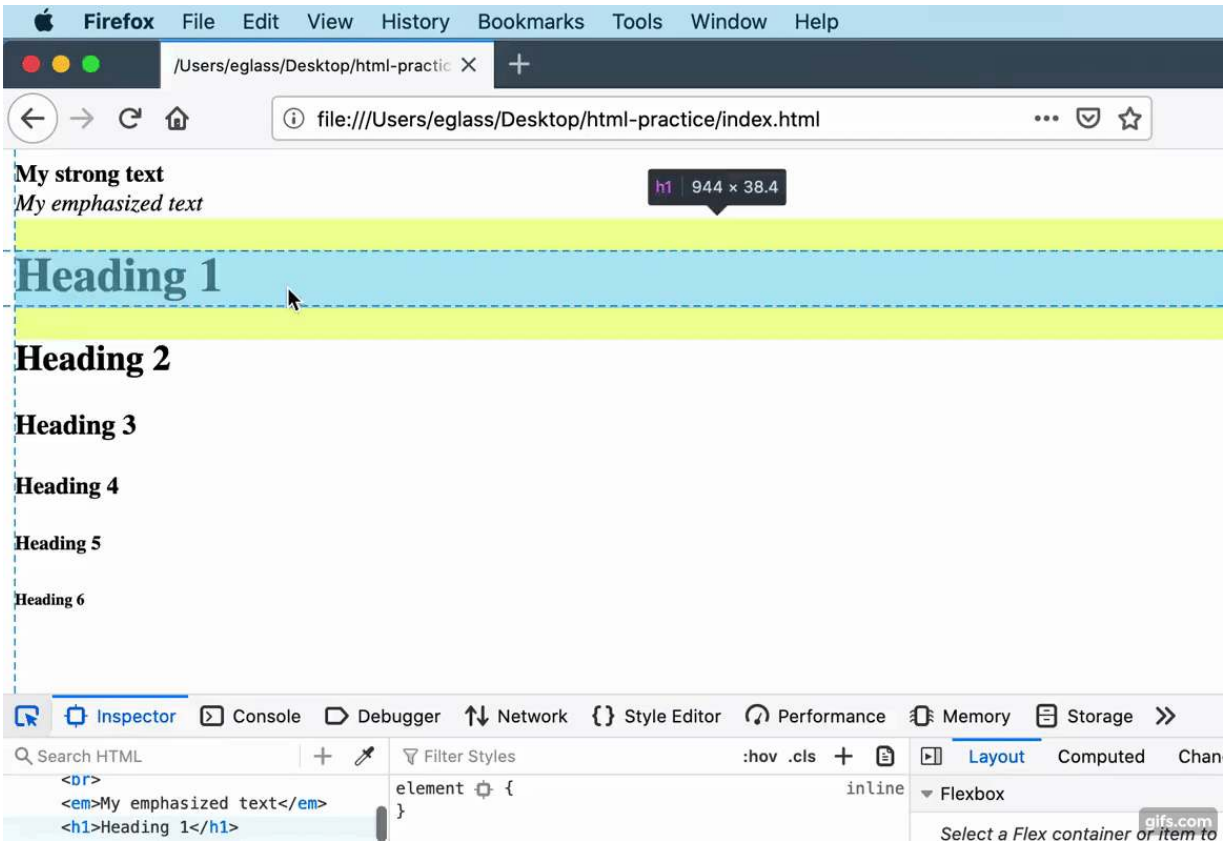
Heading 4

Heading 5

Heading 6

HTML Headings

Let's now inspect the block-level heading elements to study how they differ from the inline-level text elements above. Open up the Firefox Web Developer Inspector and hover over each of the elements to inspect their occupied space as indicated by the blue highlighting. You should be able to confirm that the occupied horizontal space of each of the inline-level elements is determined by its text content, while the occupied horizontal space of each of the block-level elements stretches across the entire web page:



This gif illustrates the process of using the Inspector tool as described in the paragraph above.

Block-level elements will always push inline-level elements down to the next line, even if you write those HTML elements on the same line of the HTML document. To confirm this for yourself, try writing a block-level element and an inline-level element on the same line of your `index.html` file. Erase the existing content from your file and add the following code snippet:

```
<strong>My strong text</strong><h1>My heading</h1>
<em>My emphasized text</a>
```

Can you guess how this HTML code will render in the browser? Save your file and reload it in the browser. You should receive something like this:

My strong text

My heading

My emphasized text

Inline and block level elements

Notice that the heading element `<h1>` has started on a new line and pushed the subsequent text element to a new line even though the elements were written on the same line in the HTML document.

You should now have an understanding of how inline-level and block-level elements are positioned and how they affect the position of nearby elements. Remembering their distinct behaviors can be useful when arranging HTML elements in the future.

We'll continue learning about new inline and block elements in the tutorials ahead.

How To Nest HTML Elements

Written by Erin Glass

This tutorial will teach you how to nest HTML elements in order to apply multiple HTML tags to a single piece of content.

HTML elements can be nested, meaning that one element can be placed inside another element. Nesting allows you to apply multiple HTML tags to a single piece of content. For example, try pasting the following code snippet inside your `index.html` file:

```
<strong>My bold text and <em>my bold and
emphasized text</em></strong>
```

Save your file and reload it in the browser. (For instructions on creating an `index.html` file, please see our tutorial here or for loading the file in your browser, see our tutorial here.) You should receive something like this:

My bold text and my bold and emphasized text

Nesting Best Practices

Note that it is recommended to always close nested tags in the reverse order that they were opened. For example, in the example below, the `<em>` tag closes first as it was the last tag to open. The `<strong>` tag closes last as it was the first to open.

```
This sentence contains HTML elements that are
<strong><em>nested according to best
practices</em></strong>.
```

As a counter example, the following HTML code contains tags that are not nested according to best practices, as the `<strong>` tag closes before the `<em>` tag:

```
This sentence contains HTML elements that are
<strong><em>not nested according to best
practices</strong></em>.
```

While not technically necessary for rendering your HTML in the browser, nesting your tags in the proper order can help improve the readability of your HTML code for you or other developers.

How To Use HTML Attributes

Written by Erin Glass

HTML attributes can be used to change the color, size, and other features of HTML elements. For example, you can use an attribute to change the color or size of a font for a text element or the width and height for an image element. In this tutorial, we'll learn how to use HTML attributes to set values for the size and color properties of HTML text elements.

An HTML attribute is placed in the opening tag like this:

```
<element attribute="property:value;">
```

One common attribute is `style`, which allows you to add style properties to an HTML element. While it's more common to use a separate stylesheet to determine the style of an HTML document, we will use the style attribute in our HTML document for this tutorial.

Using the Style Attribute

We can change multiple properties of an `<h1>` element with the style attribute. Clear your "index.html" file and paste the code below into your browser. (If you have not been following the tutorial series, you can review instructions for setting up an `index.html` file in our tutorial Setting Up Your HTML Project.

```
<h1 style="font-size:40px;color:green;"> This text is 40 pixels and green.</h1>
```

Before we load the file in our browser, let's review each of the parts of this HTML element:

- `h1` is the name of the tag. It refers to the largest-sized Heading element.
- `style` is the attribute. This attribute can take a variety of different properties.
- `font-size` is the first property we're setting for the `style` attribute.
- `40px;` is the value for the property `font-size`, which sets the text content of the element to 40 pixels.
- `color` is the second property we're setting for the `style` attribute.
- `green` is the value for the property `color`, which sets the text content color to green

Note that the properties and values are contained by quotation marks, and that each `property:value` pair ends with a semicolon `;`.

Save the file and reload it in your browser. (For instructions on loading the file in your browser, see our tutorial here.) You should receive something like this:

This text is 40 pixels and green.

You have now learned how to use the `style` attribute to change the font size and font color of a text element. Note that certain elements require attributes, such as the `<a>` element which allows you to add a link to a text or image, and the `<img>` element, which allows you to add an image to the document. We'll cover those two elements in the next tutorials and learn about additional attribute usage for `<div>` containers and other HTML elements later on in the tutorial series.

How To Add Images To Your Webpage Using HTML

Written by Erin Glass

In this tutorial, we'll learn how to use HTML to add images on a website. We'll also learn how to add alternative text to images to improve accessibility for site visitors who use screen readers.

Adding an image with HTML

Images are added to an HTML document using the `<img>` element. The `<img>` element requires the attribute `src` which allows you to set the location of the file where the image is stored. An image element is written like this:

```
<img src="Image_Location">
```

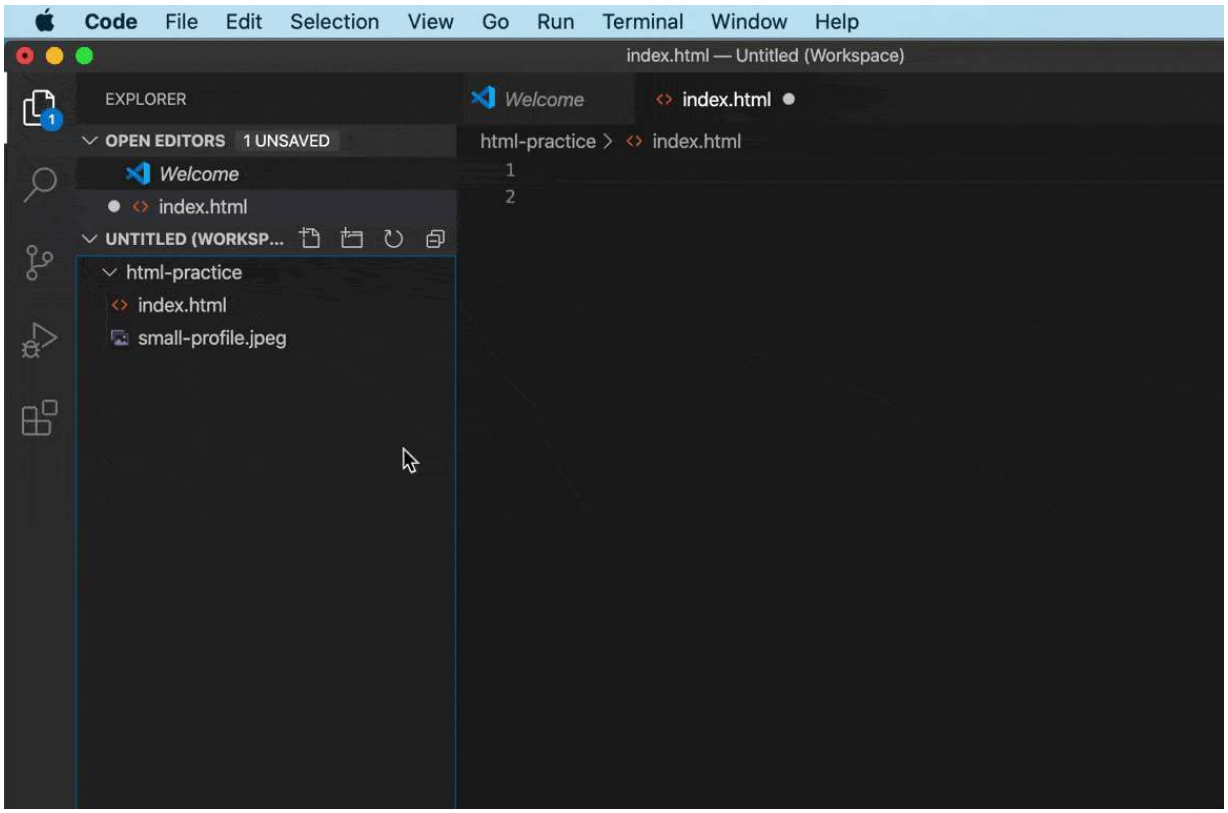
Note that the `<img>` element does not use a closing `</img>` tag. To try using the `<img>` element, download our image of Sammy the Shark and place it in your project directory `html-practice`.

Note: To download the image of Sammy the Shark, visit the link and `CTRL + Left Click` (on Macs) or `Right Click` (on Windows) on the image and select "Save Image As" and save it as `small-profile.jpeg` to your project directory.

Next, erase the content of your `index.html` file and paste `<img src="Image_Location">` into the file. (If you have not been

following the tutorial series, you can review instructions for setting up an `index.html` file in our tutorial Setting Up Your HTML Project.

Then, copy the file path of the image and replace `Image_Location` with the location of your saved image. If you are using the Visual Studio Code text editor, you can copy the file path by using `CTRL + Left Click` (on Macs) or `Right Click` (on Windows) on the image file `small-profile.jpeg` in the left-hand panel and selecting "Copy Path." For an illustration of the process, please see the gif below:



Gif of how to copy an image file path

Note: Make sure to copy the relative or project file path of the image rather than the absolute or full file path of the image. The relative path

refers to the file location relative to the current working directory (as opposed to the absolute path, which refers to the file location relative to the root directory.) While both paths will work in this instance, only the relative path would work if we decided to publish our website online. Since our end goal is to create a publishable website, we will start using relative paths now when adding `<img>` elements to our document.

Save your `index.html` file and reload it in your browser. You should receive something like this:

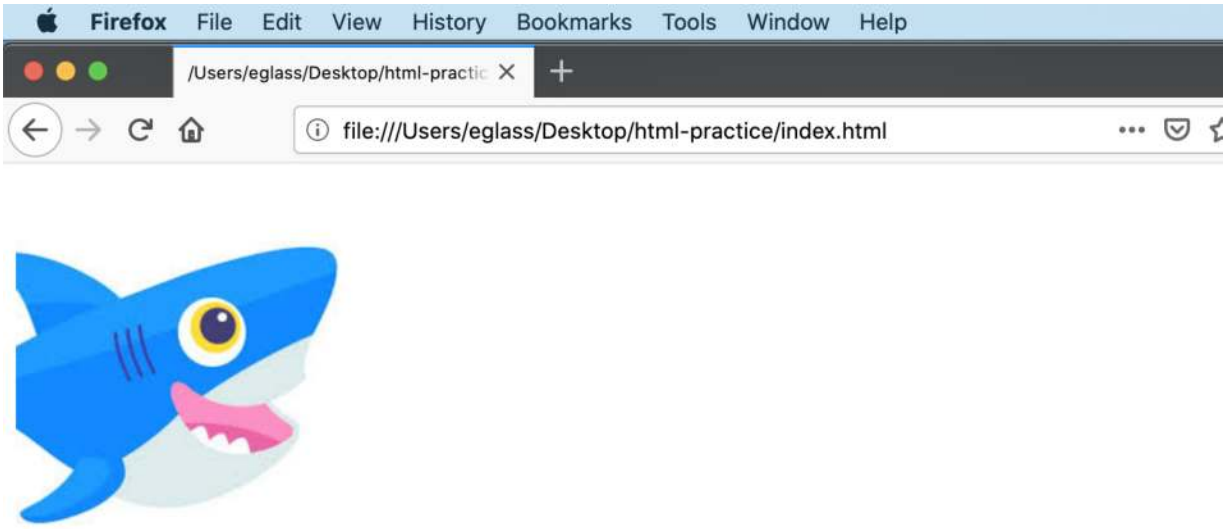


Image in browser

Technically, you can also use links to images hosted online as file paths. To understand how this works, try replacing the image location with a link to our image of Sammy the Shark like this:

```
<img src="https://html.sammy-codes.com/images/small-profile.jpeg">
```

Save your file and reload it in the browser. The image should still load in your web document, but this time the image is being sourced from its online location rather than your local project directory. You can experiment with adding other online images by using their location links as the `src` attribute in the `<img>` tag.

However, when building a website it is generally better to host your images in your project directory to ensure the sustainability of the site. If the image is taken down by its host or if its address changes, it will no longer render on your site.

Alternative Text for Accessibility

When adding an image, you should always include alternative text describing its content using the `alt` attribute. This text is typically not displayed on the webpage but is used by screen readers to communicate content to visually-impaired site visitors.

```
<img src="https://html.sammy-codes.com/images/small-profile.jpeg" alt="Digital Ocean's mascot, a blue smiling shark.">
```

When adding alternative text, keep the following best practices in mind:

- For informative images, alternative text should clearly and concisely describe the subject matter of the image, without referring to the image

itself. For example, do not write "Image of Sammy the Shark, DigitalOcean's mascot" but "Sammy the Shark, DigitalOcean's mascot."

- For decorative images, the `alt` attribute should still be used but with a null value, as this improves the screen reader experience: `<img src="images/decorative_image.jpeg" alt="">`.
- For a useful guide on determining whether an image is informative or decorative, visit https://www.w3.org/WAI/tutorials/images/decision-tree/

You should now have familiarity with how to add images to your HTML document and how to add alternative text to aid with accessibility. We'll learn how to change the image size and style in the tutorial How To Add a Profile Image To Your Webpage later on in the series. In the next tutorial, we'll learn how to add links to an HTML document.

How To Add Hyperlinks in HTML

Written by Erin Glass

This tutorial will walk you through the steps of adding hyperlinks to your webpage.

Hyperlinks can be added to text or images with the anchor link element `<a>`. The `<a>` tag requires the attribute `href`, which is used to specify the destination link. The `<a>` element is used like this:

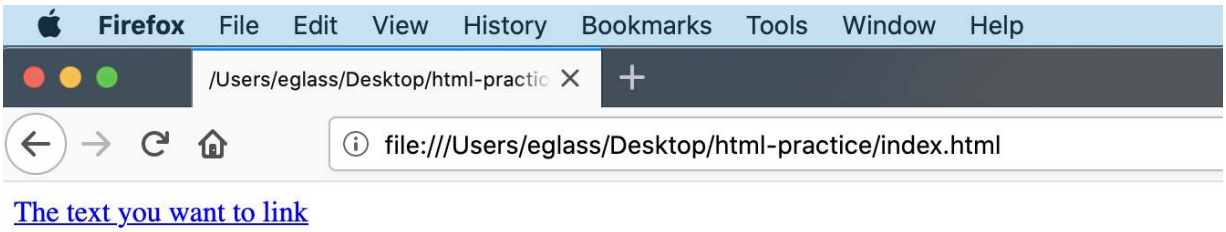
```
<a href="www.DestinationLink.com">The text you want to link</a>
```

Try adding the code snippet below to your "index.html" file and changing the sample highlighted text with a real link, such as `https://digitalocean.com/community`:

```
<a href="https://www.digitalocean.com/community">The text you want to link</a>
```

(If you have not been following the tutorial series, you can review instructions for setting up an `index.html` file in our tutorial Setting Up Your HTML Project. Save the file and load it in your browser.

You should receive an output like this:



HTML Link

You can also link images by wrapping an image element with an <a> element like so:

```
<a
href="https://www.digitalocean.com/community">The
text you want to link
<img src="https://html.sammy-
codes.com/images/small-profile.jpeg">
</a>
```

Try testing the code snippet in your browser to check that it works. You should now understand how to add hyperlinks to text and images on your webpage.

How To Use a , the HTML Content Division Element

Written by Erin Glass

The HTML Content Division element (`<div>`) acts as a container to structure a webpage into separate components for individual styling. This tutorial will teach you how to create and style `<div>` containers on your webpage.

On its own, the `<div>` element has little effect on the layout of the page and is usually given attributes to adjust its size, color, position, or other features. Typically, developers style `<div>` elements with CSS, but this tutorial will give you an understanding of how `<div>` elements work by styling them directly in an HTML document.

The `<div>` element is styled with the HTML `style` attribute. As demonstrated in the example below, a `<div>` element can contain multiple style properties:

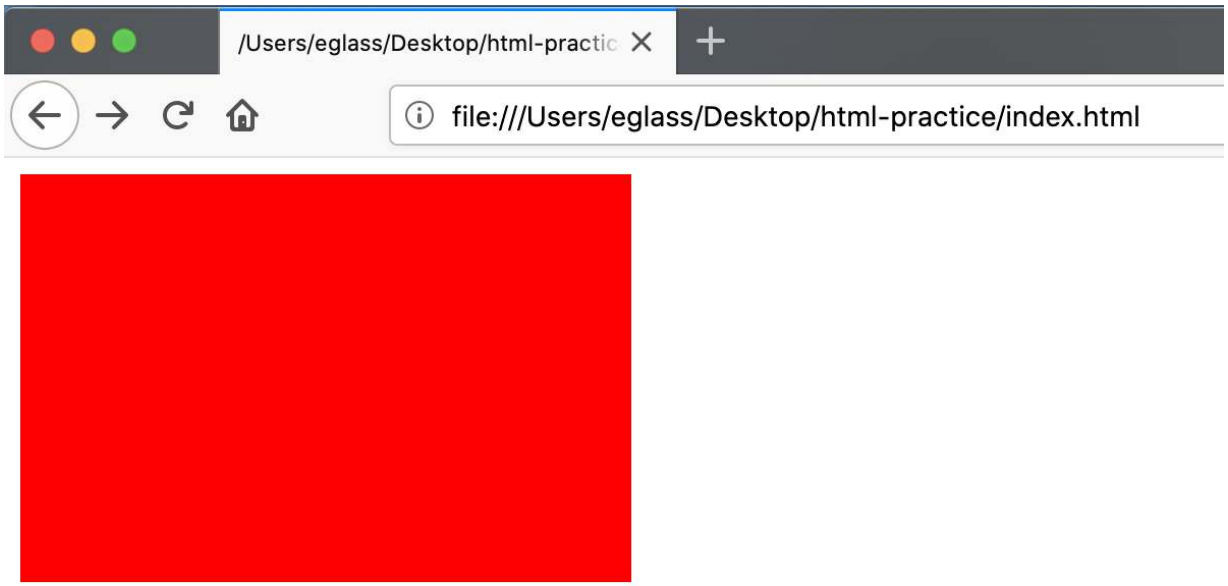
```
<div style="property: value; property: value;">
</div>
```

Notice that the `<div>` element has opening and closing tags but does not require any content. To understand how the `<div>` element works in practice, clear your `index.html` file and paste the code below inside. (If you have not been following the tutorial series, you can review instructions

for setting up an `index.html` file in our tutorial Setting Up Your HTML Project.

```
<div style="width:300px;height:200px; background-color:red;">
</div>
```

Save the file and reload it in the browser. (For instructions on loading the file in your browser, see our tutorial here.) You should receive a red box with a width of 300 pixels and a height of 200 pixels like this:

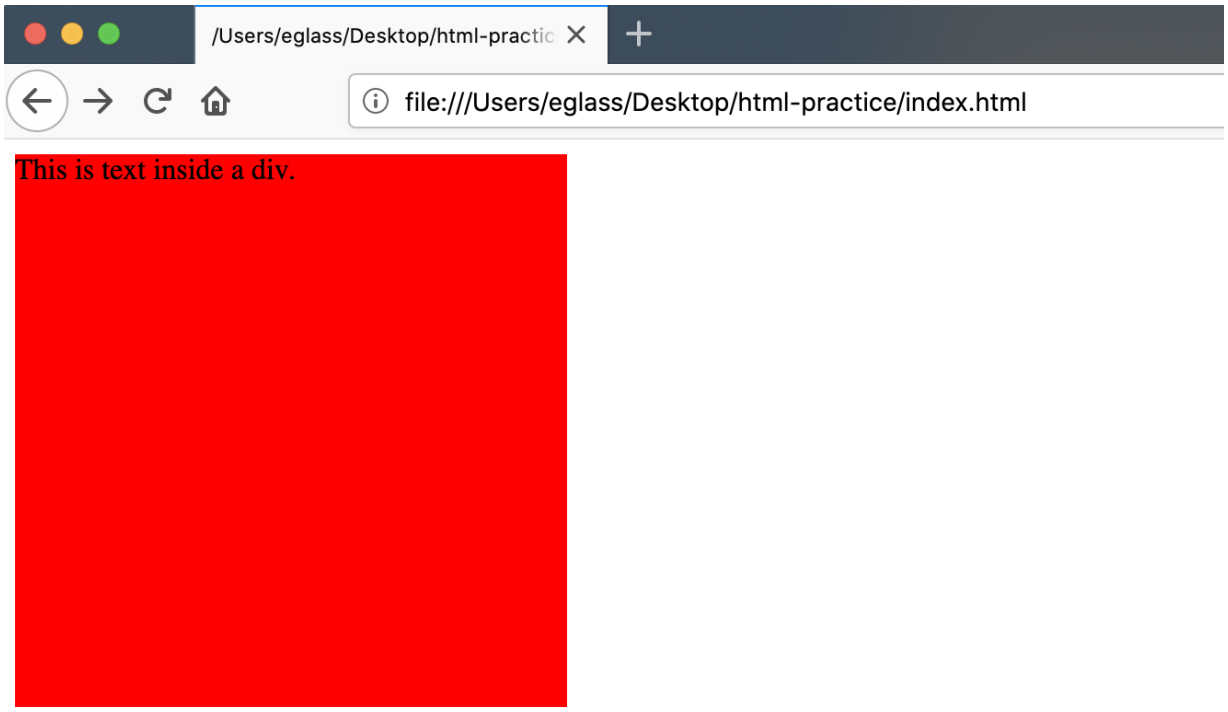


Red div

You can also add content to a element by adding content in between the opening and closing tags. Try adding text in between your tags like so:

```
<div style="width:300px;height:300px; background-color:red;">
  This is text inside a div.
</div>
```

Save the file and reload it in the browser. You should receive something like this:

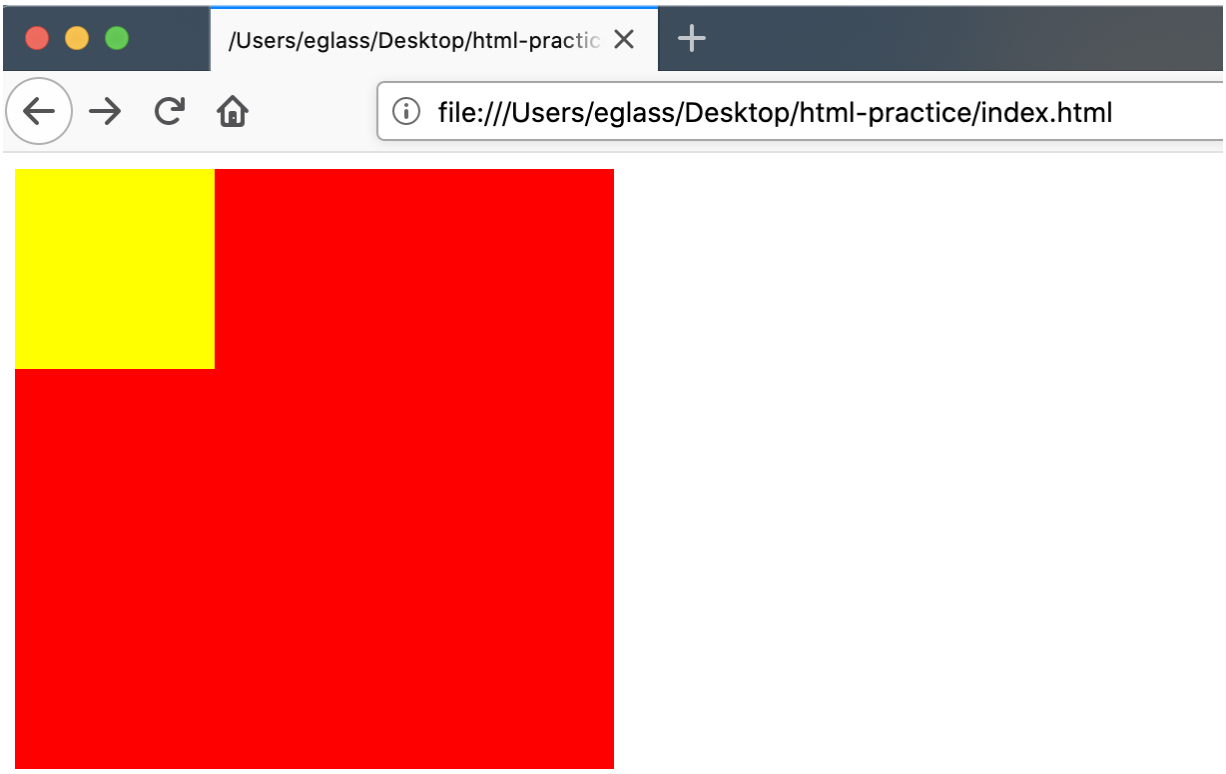


Div with text

You can also put containers inside containers. Try adding a yellow container inside the red container like so:

```
<div style="width:300px;height:300px; background-color:red;">
  <div style="width:100px;height:100px; background-color:yellow;">
  </div>
</div>
```

Save the file and reload it in the browser. You should receive something like this:



Div inside div

Note that `<div>` elements are block-level elements and will start on their own line and push subsequent elements down to the next line. Try adding another `<div>` element to your file to understand how it is pushed down to the next line (even though there appears to be enough space for it to fit next to the second `<div>` element:

```
<div style="width:300px;height:300px;background-color:red;">
  <div style="width:100px;height:100px; background-color:yellow;">
  </div>
</div>
<div style="width:100px;height:100px; background-color:blue;">
</div>
```

Save the file and reload it in the browser. You should receive something like this:

Three divs

You should now have a basic understanding of how `<div>` elements work. We will return to `<div>` elements when we begin building our website in the third section of this tutorial series.

How To Modify the Color of HTML Elements

Written by Erin Glass

You can use HTML to modify the color of some elements and pieces of content of a webpage. For example, you can change the color of text, a border, or—as demonstrated in the tutorial on HTML Content Division—of a `<div>` element. The method for changing the color values of these pieces of content varies from element to element.

In this tutorial, you will learn how to change the color of text, image borders, and `<div>` elements using HTML color names.

The color of text elements, such as `<p>` or `<h1>`, is modified by using the `style` attribute and the `color` property like so:

```
<p style="color:blue;">This is blue text.</p>
```

Try writing this code in your `index.html` file and loading it in the browser. (If you have not been following the tutorial series, you can review instructions for setting up an `index.html` file in our tutorial Setting Up Your HTML Project. For instructions on loading the file in your browser, see our tutorial here.)

You should receive something like this:

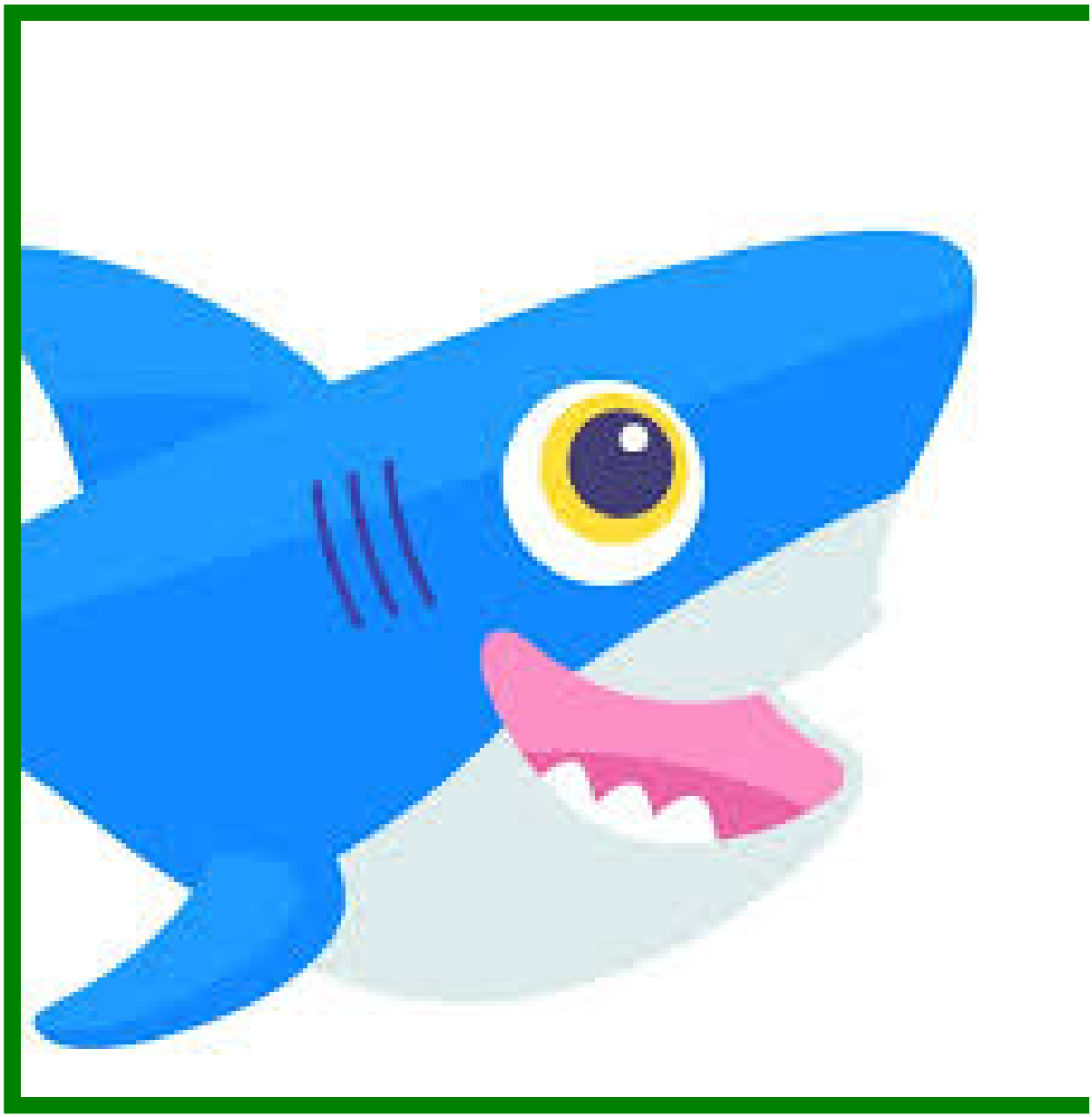
This is blue text.

The color of a border is modified by using the `style` attribute and the `border` property:

```
<img src="https://html.sammy-codes.com/images/small-profile.jpeg" style="border: 10px solid green"/>
```

Try writing this code in your `index.html` file and loading it in the browser. (Note that we are using an image that we are hosting online in this example. We have also specified that the border should be 10 pixels wide and solid (as opposed to dashed)).

You should receive something like this:



The color of a `<div>` container is modified by using the `style` attribute and the `background-color` property like so:

```
<div style="width:200px;height:200px;background-color:yellow;"></div>
```

Try writing this code in your `index.html` file and loading it in the browser. You should receive something like this:

In these examples, the color value is defined by color names. Try changing the color of text, image borders and `<div>` elements using the following color names: `black`, `white`, `gray`, `silver`, `purple`, `red`, `fuchsia`, `lime`, `olive`, `green`, `yellow`, `teal`, `navy`, `blue`, `maroon`, and `aqua`.

Note that colors can also be specified by hexadecimal values. A hexadecimal color consists of six alphanumeric digits preceded by a pound symbol, such as `#0000FF` (blue), `#40E0D0` (turquoise), or `#C0C0C0` (silver). For this tutorial series, however, we will continue using color names.

You should now have a basic familiarity with how to change the color of text, image borders, and `<div>` elements using color names. We will return to colors later on the tutorial series when we begin building our website.

How To Set Up Your HTML Website Project

Written by Erin Glass

The first thing to do when creating a new website project is to create a project directory (or folder) to store all the files that will be created in the process. This tutorial will walk you through the steps of setting up folders and files necessary for building your website with HTML.

For this website project, we can continue using the `html-practice` project directory and `index.html` file we created earlier in the tutorial series. (If you have not been following along this tutorial series and need instructions for setting up a project directory and `index.html` file, please see our earlier tutorial in this series Setting Up Your HTML Project.

Note: If you decide to choose your own name for the directory, make sure to avoid character spaces, special characters (such as !, #, %, or others), and capital letters, as these can cause problems later on.

Next, we'll format the `index.html` file to serve as the website's homepage. The first step in formatting a web document is to add the `<!DOCTYPE html>` declaration to the first line. Make sure your `index.html` file is clear and then add `<!DOCTYPE html>`to the first line of the document.

This declaration tells the browser which type of HTML is being used and is important to declare as there are multiple versions of HTML. In this declaration, `html` specifies the current web standard of HTML, which is HTML5.

Next, we'll add the `<html>` element by adding opening and closing `<html>` tags. The `<html>` element tells the browser that all content it contains is intended to be read as HTML. Inside this tag, we will also add the `lang` attribute, which specifies the language of the webpage. In this example, we will specify that our site is in English using the `en` language tag. For a full list of language tags, visit the IANA Language Subtag Registry.

Your document should now look like this:

```
<!DOCTYPE html>
<html lang="en">
</html>
```

From this point forward, all content that we add to our website will be added between the opening and closing `<html>` tags.

We will begin adding content to our site in the next tutorial.

How To Add an HTML <head> Element To Your Webpage

Written by Erin Glass

This tutorial will walk you through the steps of adding a `<head>` element to your webpage, which creates a section for us to include machine-readable information about our web document. This information is primarily used by browsers and search engines to interpret the content of the page. Content placed inside the `<head>` element will not be visible on the web page.

Note: The HTML `<head>` element is a semantic element in that it tells the browser and the developer the meaning or purpose of its content. Semantic elements are used to aid human readability of an HTML document, provide the browser further information for interpreting the content, improve site accessibility (screen readers use semantic tags), and can assist with SEO positioning.

Add the opening and closing `<head>` tags inside of the `<html>` tags. Next, add two additional lines of HTML code inside the `<head>` tags like this:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>Sammy's First Website</title>
  </head>
</html>
```

Note that you have nested a variety of HTML elements inside one another. The `<title>` and `<meta>` elements are nested inside the <head> element, and the `<head>` element is nested inside the `<html>` element. We will nest elements frequently as the tutorial proceeds.

Let's now pause briefly to understand the purpose of the code we've just added. The line of code after the opening `<head>` tag — `<meta charset="utf-8">` — specifies the document's character set to UTF-8, a unicode format that supports a majority of characters from a wide variety of written languages.

The next line of code sets the HTML document's title using the `<title>` element. The content you insert into this element will be displayed on the browser tab and as the website's title in search results, but it will not show up on the web page itself. Make sure to replace "Sammy's First Website" with your name or the name of the website you're building.

Though developers often add additional information in the `<head>` section, we now have sufficient information for creating a basic HTML webpage. Save your file before moving onto the next section. If you try loading the file in your browser, you should receive a blank page.

You should now know the purpose of HTML <head> elements and how to add one to an HTML file.

How To Add a Favicon to Your Website with HTML

Written by Erin Glass

In this tutorial, we will walk through the steps involved in adding a favicon to your webpage using HTML. You can use any image you like for your favicon, but keep in mind that simple, high-contrast images often work best given the favicon's small size. You can also generate a custom favicon through sites like favicon.cc.

A favicon is a small image that is located in the browser tab to the left of a webpage's title. The image below illustrates the location of a favicon.

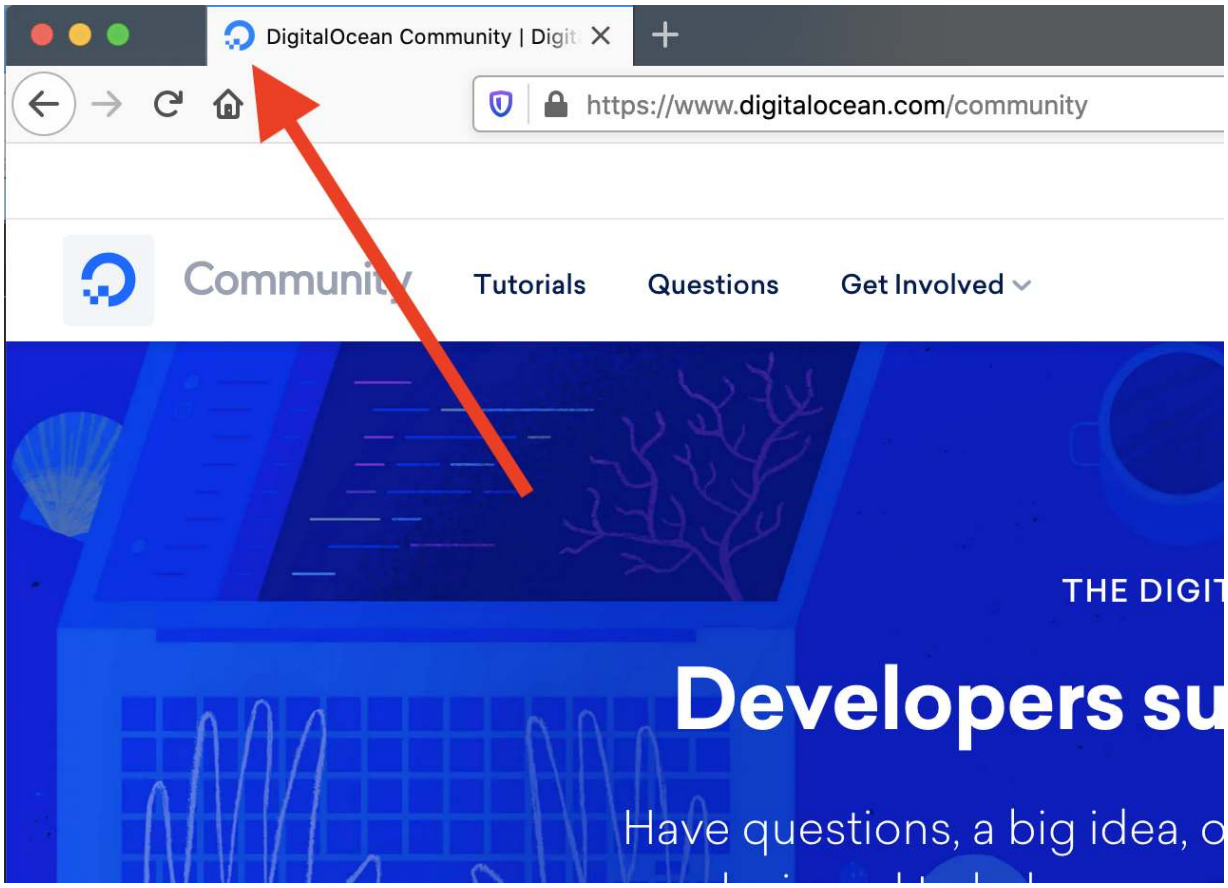


Image of Favicon

To add a favicon to your site, create a folder in your project directory called `images` (if you don't already have one) and save your desired favicon image in this folder. If you don't have an image, you download this Sammy the Shark image that we have hosted on our demonstration website. (For a refresher on how to add images to webpages using HTML, please visit our tutorial HTML Images from earlier in this tutorial series.

Next, add the `<link>` element (highlighted below) to your `index.html` file right below the `<title>` element. Your code should now be like this:

```
...
<title> Sammy's First Website </title>
<link rel="shortcut icon" type="image/jpg"
href="Favicon_Image_Location"/>
...
```

Make sure to replace `Favicon_Image_Location` with the relative file path of your favicon image. Save the `index.html` file and reload it in your web browser. Your browser tab should now contain a favicon image.

You should now know how to add favicon images to websites using HTML.

How To Style the HTML <body> Element

Written by Erin Glass

The HTML `<body>` element is a semantic element that tells the browser that its content is part of the body of the webpage and intended for display. In this tutorial, we will add a `<body>` element to our web document where we can add content later on.

To add a `<body>` element to your document, insert opening and closing `<body>` tags after the closing `</head>` tag but before the closing `</html>` tag. Your document should now have the following code:

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="utf-8">
  <title>Sammy's First Website</title>
  <link rel="shortcut icon" type="image/jpg"
href="Favicon_Image_Location"/>
</head>

<body>
</body>
</html>
```

You should now understand the purpose of an HTML `<body>` element and understand how to add one to your HTML file.

In the next step, we'll begin to add our website content in between the `<body>` tags.

Creating the Top Section of Your Homepage With HTML

Written by Erin Glass

We will now begin adding content by replicating the top section of the demonstration website.

Top section of demonstration website

This top section is composed of a large background image, a small profile image, a text header, a text subheader, and a link. Each of these pieces of content are styled and positioned with HTML.

In the remaining tutorials of this series, we'll learn how to use HTML to recreate this content on a new webpage.

How To Add a Background Image to the Top Section of Your Webpage With HTML

Written by Erin Glass

In this tutorial we'll learn how to use a `<div>` container to structure the top section of the webpage. We will use the `style` attribute to specify the height of our `<div>` container, apply a background image, and specify that the background image should cover the entire area of the `<div>` container.

Before we get started, we'll need a background image. You may download and use our demonstration site's background image for the purpose of the tutorial, or you can choose a new image. (For a refresher on how to add images to webpages using HTML, please visit our tutorial HTML Images from earlier in this tutorial series).

Once you've chosen your background image, save the image in your `images` folder as `background-image.jpg`.

Next, paste the highlighted code snippet into your `index.html` file below the opening `<body>` tag and above the closing `</body>` tag:

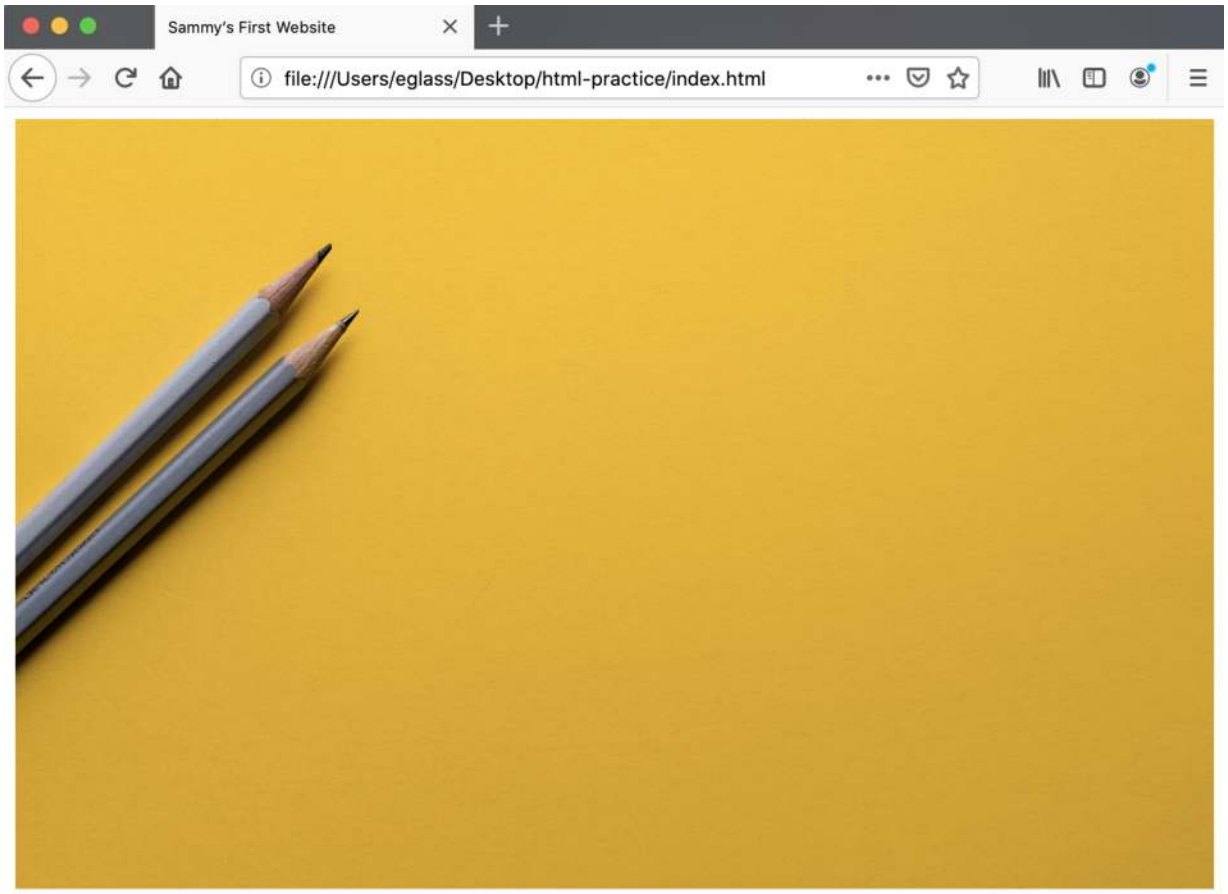
```
. . .
<body>
<!--First section-->
  <div style="background-image:
url('Image_Location');
  background-size: cover; height:480px; padding-
top:80px;">
  </div>
</body>
...
```

Make sure to switch the text that says `Image_Location` with the file path of your image and don't forget to add the closing `</div>` tag.

Note that we have added the comment `<!--First section-->` to help organize our HTML code. A comment is a piece of code that is ignored by the browser. Comments are used to help explain or organize code to developers. They are created with the opening tag `<!--` and the closing tag `-->`.

We have also specified the `height` to 480 pixels and `padding-top` to 80 pixels, which will create 80 pixels of space between the top of the `<div>` element and any content we place inside. Note that you will not be able to see the effects of the `padding-top` value until we place content inside in the next step.

Save the file and reload it in the browser. You should receive something like this:



Background image in top section

Alternately, you can use a background color instead of a background image. To use a background color, replace the `<div>` element code snippet you just created with the following highlighted `<div>` element code snippet like this:

```
. . .
<body>
  <!--First section-->
  <div style="background-color: #f4bc01;
height:480px; padding-top:80px;">
  </div>
</body>
...
```

Save the file and reload it in the browser to check your results. The background image should now be replaced with a container that is the same size but has a solid yellow color.

If you compare the `<div>` container on your site with the same `<div>` container on the demonstration site, you may notice that your webpage's `<div>` container is surrounded by a small margin of white space. This margin is due to the fact that all HTML pages are automatically set to have a small margin by default.

To remove this margin, we need to add a style attribute to the opening `<body>` tag that sets the margin of the `<body>` element of the HTML page to 0 pixels. Locate the opening `<body>` in your `index.html` file and modify it with the highlighted code:

```
<body style="margin:0;">
```

Save and reload the file in your browser. There should now be no white margin surrounding the top `<div>` container.

You should now know how to add a `<div>` container with a background image to structure the top section of a webpage.

How To Add a Styled Profile Image To Your Webpage With HTML

Written by Erin Glass

In this tutorial, we'll walk through the steps of adding and styling the top profile image as displayed in the demonstration site.

Top section of demonstration website

Before we get started, you may want to pick a personal profile photo for including on your site. If you don't have a profile photo, you can use any image for demonstration purposes or create an avatar through a site like Getavataaars.com. Otherwise, you can use the image from our demonstration site by downloading the image here. (For a refresher on how to add images to webpages using HTML, please visit our tutorial HTML Images from earlier in this tutorial series.)

Once you've selected an image, save it as `small-profile.jpg` in your `image` folder.

Paste the following `<img>` element in between the opening and closing `<div>` tags you created in the last tutorial like so:

```
...
<div style="background-image:
url('ImageLocation');background-size: cover;
height:480px; padding-top:80px;">
  <img src="ImageFilePath;" style="height:150px;">
</div>
...
```

Make sure to switch out the highlighted `src` address with the file path of your profile image. Note that we are also using the style attribute to specify the height of the image to 150 pixels. The `<img>` element does not require a closing tag.

Save and reload the page in the browser to check your results. You should receive the following:

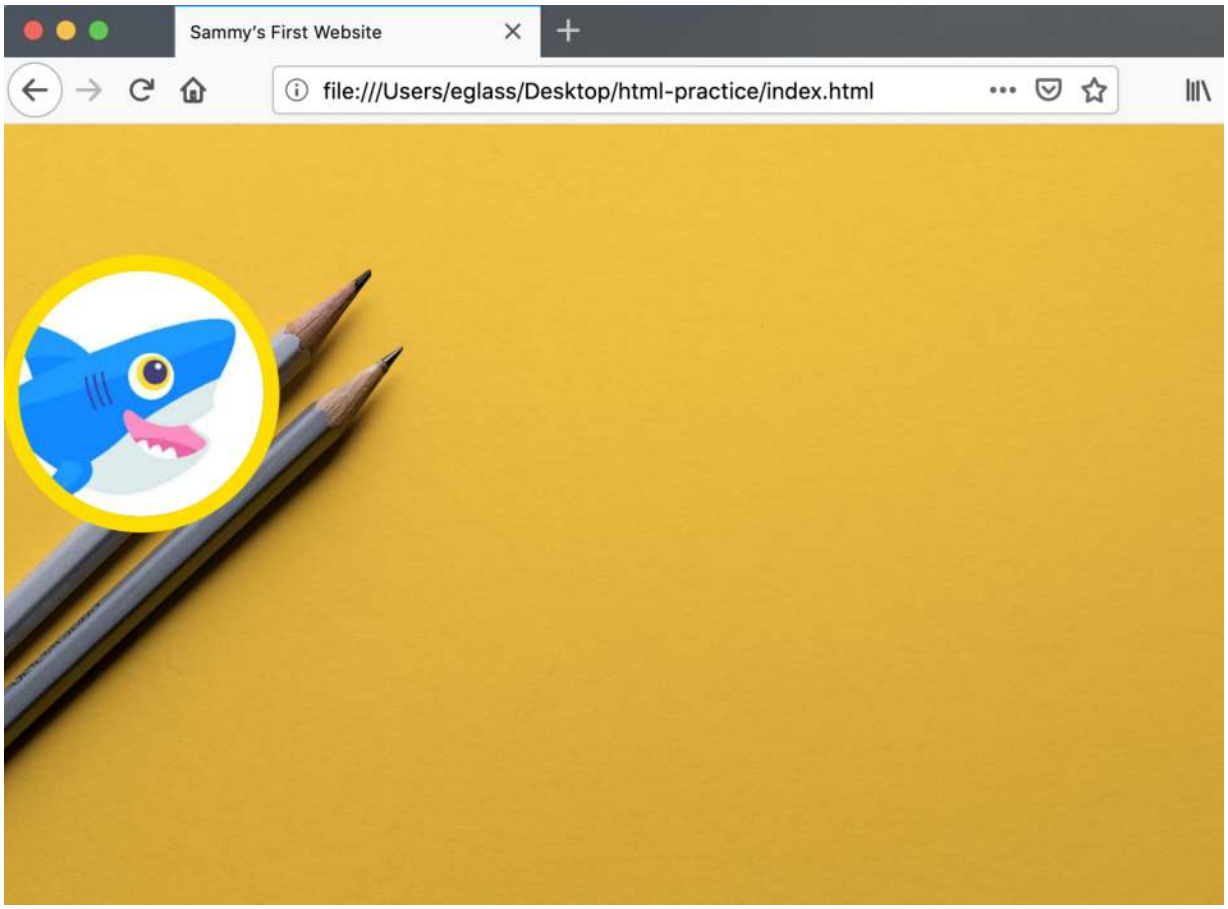
Profile image on background image

Your profile image should appear as 150 pixels tall given the height we specified with the `style` attribute. It should also located 80 pixels below the top of the `<div>` container, given the `top-padding` property we specified for the `<div>` container in the previous tutorial. Next, let's add properties to our `style` attribute that will give our image a circular shape and a yellow border. We will also add alternative text to improve accessibility for site visitors who use screen readers.

Add the highlighted properties to your `<img>` element:

```
<img src="ImageFilePath" style="height:150px;
border-radius: 50%; border: 10px solid #FEDE00;"
alt="This is a small profile image of
DigitalOcean's mascot, a blue smiling shark.">
```

Make sure you still have the correct file path of your image listed as the `src` address. Save the file and reload it in the browser. You should receive something like this:



Styled profile image

Before moving on, let's briefly pause to study the code modifications we just made. Setting the `border-radius` value to 50% gives the image a circular shape. Setting the border value to `10px solid #FEDE00` gives the image a solid border that is 10 pixels wide and has the hexadecimal color value `#FEDE00`.

You should now know how to add and style a profile image to your website with HTML. We are now ready to add a title and subtitle to the webpage in the next tutorial.

How To Add and Style a Title To Your Webpage With HTML

Written by Erin Glass

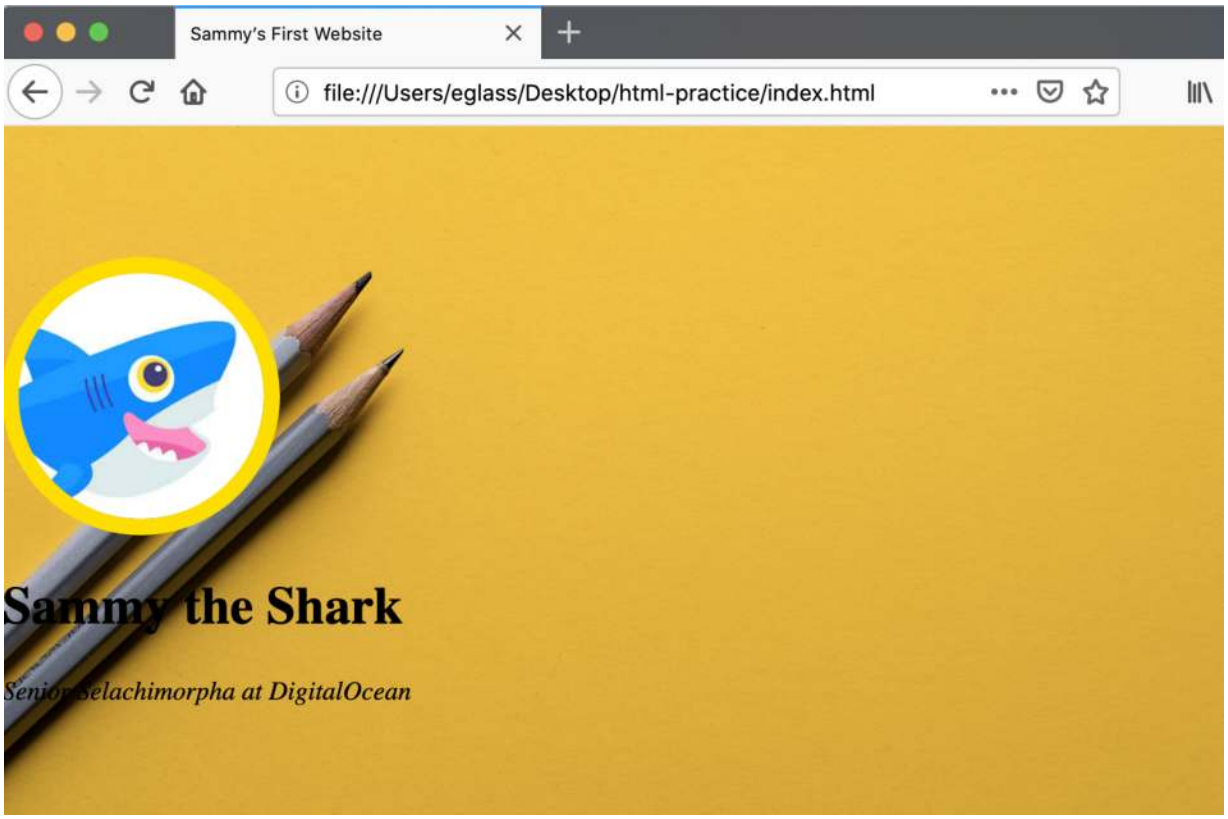
In this tutorial, we will add and style a title and subtitle to our homepage. For the demonstration site, we're using Sammy's name and Sammy's professional title, but you can add any content here that you like. For this content, we'll use the `<h1>` heading element, the `<p>` paragraph element, and the `<em>` emphasis element.

Paste the following highlighted code snippet after your profile `<img>` element and before the closing `</div>` tag:

```
...
<img src="https://html.sammy-codes.com/images/small-profile.jpeg"
style="height:150px; border-radius: 50%; border: 10px solid #FEDE00; padding-top:80px;">
<h1>Sammy the Shark</h1>
<p><em>Senior Selachimorpha at DigitalOcean</em>
</p>
</div>
```

Make sure to change the text with your own information.

Save the file and reload it in the browser. You should receive something like this:



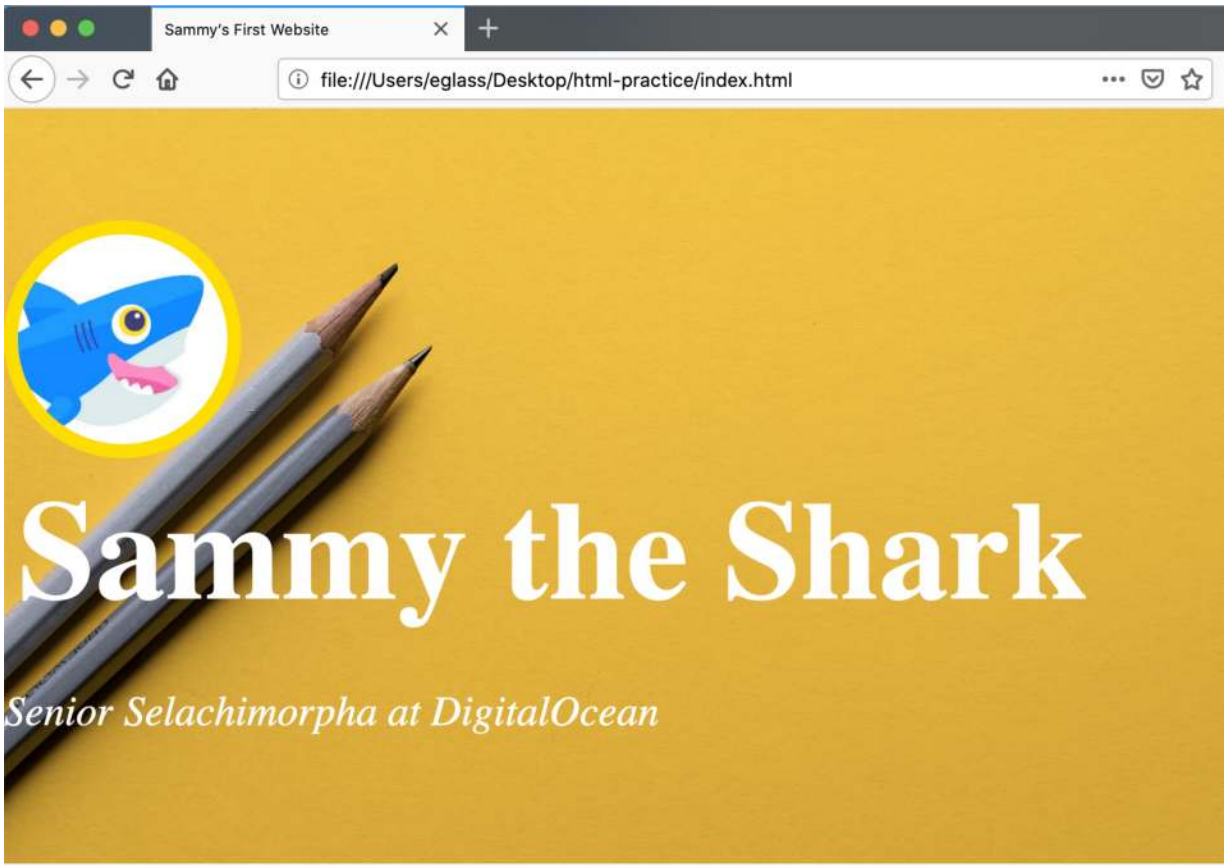
Title and subtitle

The elements used in this code snippet apply some light styling to our title and subtitle. However, we'll need to add additional style values if we want the style of our title and subtitle to match the style of the demonstration site.

To make these modifications, we'll add the `style` attribute to these elements to set additional properties. Add the highlighted attributes to your `<h1>` and `<p>` elements as demonstrated in the following code snippet:

```
<h1 style="font-size:100px; color:white;
margin:10px;">Sammy the Shark</h1>
<p style="font-size:30px; color: white;">
<em>Senior Selachimorpha at DigitalOcean</em></p>
```

Save your file and reload it in the browser. You should receive something like this:



Styled title

These style properties adjust the font size to 30 pixels and change the font color to white. We have also added a margin of 10 pixels to the `<h1>` element.

You should now know how to add and style a title and subtitle to your webpage with HTML. In the next tutorial, we'll learn how to create and link to an additional webpage on your website.

How To Create and Link To Additional Website Pages With HTML

Written by Erin Glass

When building a website, you may want to have more than one webpage. If you want to add and link to additional pages, you'll need to first create a new `html` file in your website project directory. In this tutorial, we'll learn how to create and link to an additional webpage on your website

Our demonstration website includes an "About" webpage. In this tutorial, we'll walk you through the process of creating and linking to an "About" webpage, but you may change the title and the content of this page to fit your needs.

To add a new page to your website, create a new file named `about.html` and save it in your project directory `html-practice`. (If you have not been following the tutorial series, you can review instructions for setting up a new `html` file in our tutorial Setting Up Your HTML Project.)

Note: If you decide to choose your own name for the file, make sure to avoid character spaces, special characters (such as !, #, %, or others), and capital letters as these can cause problems later on. You must also include the `.html` extension.

Next, you'll need to format the file by adding information that will help the browser interpret the file content. To format the file, add following code snippet at the top of the document:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>About</title>
  </head>
</html>
```

Make sure to change the highlighted text with a title you want for you page. For an explanation of each of the HTML tags, please visit the earlier tutorial in this series Adding an HTML <head> Element To Your Webpage. Save the file before you continue.

Before adding any content to this page, let's walk through the steps of adding a link to this page on your homepage.

First, return to your `index.html` file and add the following snippet below the subtitle of your site and above the closing `</div>` tag:

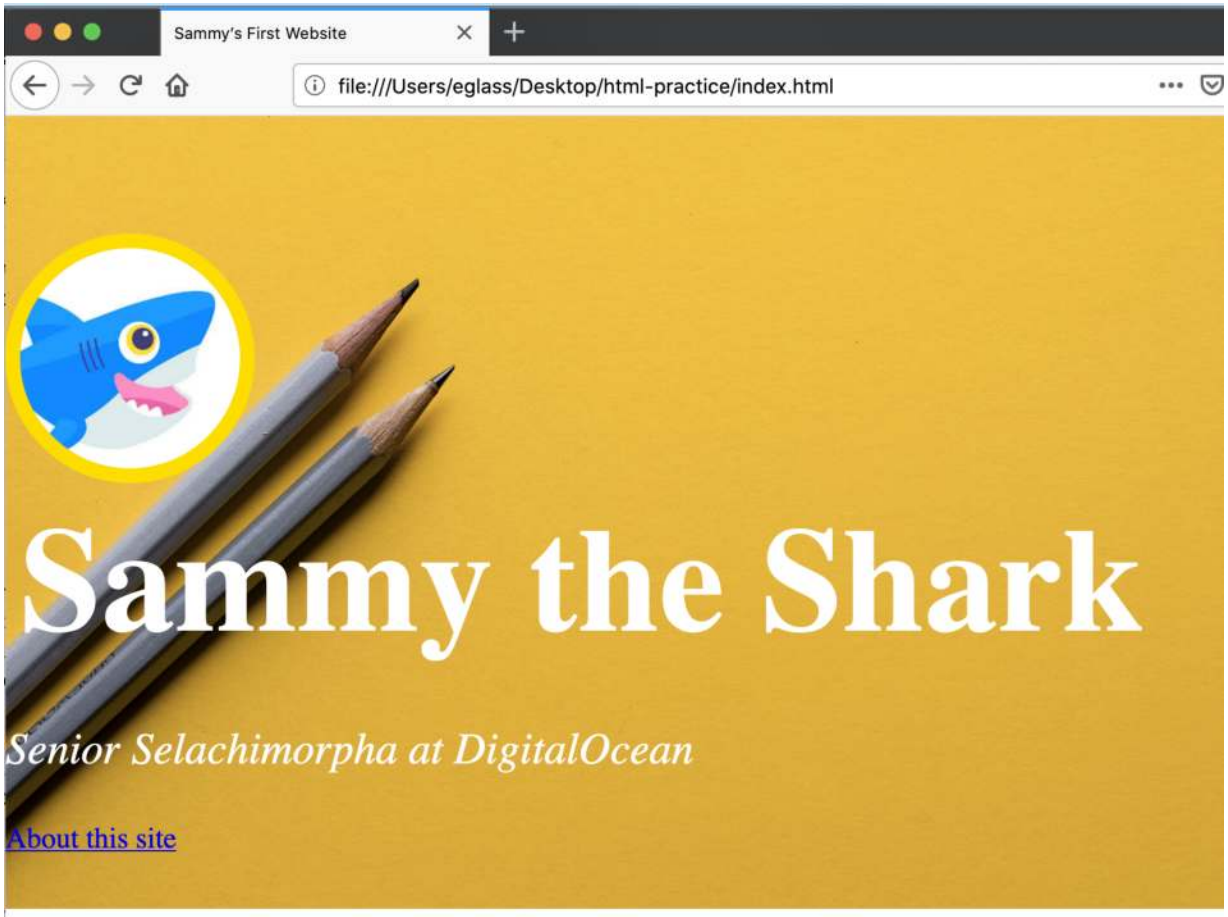
```
...
<p style="font-size: 20px; color:#1F9AFE;">
<a href="Webpage_FilePath">About this site</a>
</p>
...
```

Change the highlighted file path to the relative file path of your "about.html" file. The relative path refers to the file location relative to the current working directory (as opposed to the absolute path, which refers to

the file location relative to the root directory.) If you are using the Visual Studio Code text editor, you can copy the relative file path by `CTRL + Left Click` (on Macs) or `right-clicking' (on Windows) on the file icon and selecting`Copy Relative Path.`

Note that we have also specified a font-size and color using the `style` attribute. Save your `index.html` file and reload it in the browser.

You should now have a link that directs to your `about.html` web page like this:



Webpage with link

If you receive an error, make sure that your file is in the same project directory as your `index.html` file and that there are no errors in your project path.

You should now know how to create and link to a new webpage on your website. You can use these same steps to create and link to additional webpages on your website. You can also add content to any new webpage in the same way you are learning to add content to your homepage.

How To Center or Align Text and Images on Your Webpage with HTML

Written by Erin Glass

Aligning content to the center, left, or right can be useful for arranging content on your page. In this tutorial, we'll learn how to align text using HTML.

To align text on a webage, we can use the `style` attribute and the property `text-align`.

For example, the following code snippet would center the text "Sample text":

```
<p style="text-align:center;">Sample text</p>
```

To align your HTML content to the left or right, you would replace `center` with `left` or `right`.

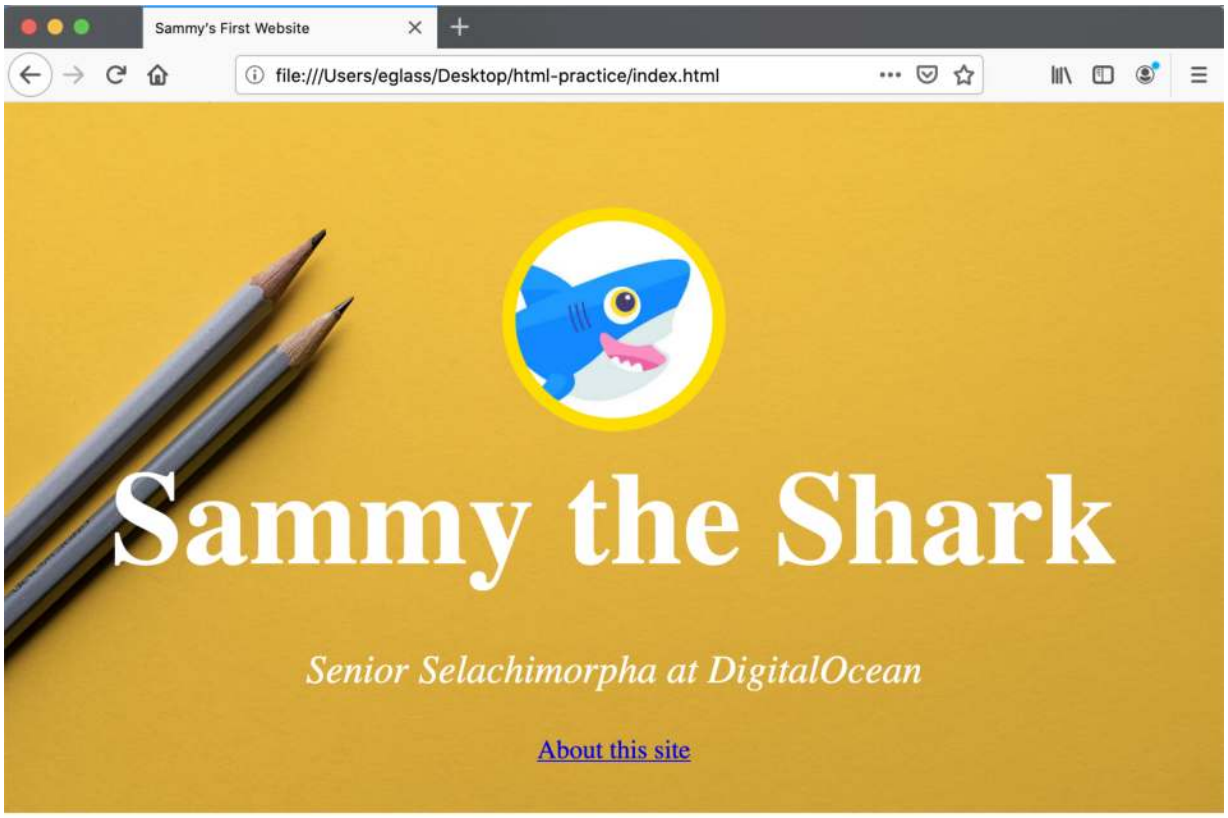
In this tutorial, we'll go through the process of using the `text-align` property to center the images and text in the top section of our webpage as illustrated in our demonstration website.

To center this content, we'll add the `text-align` property to the `<div>` element that contains the background image, profile image, title, subtitle, and link in the top section of the homepage.

Locate this `<div>` element in your `index.html` file and add the highlighted text like so:

```
...
<!--First section-->
<div style="background-image:
url('https://html.sammy-
codes.com/images/background.jpg');
background-size: cover; height:480px; padding-top:
80px; text-align: center;">
  <img src="https://html.sammy-
codes.com/images/small-profile.jpeg"
style="height:150px; border-radius: 50%; border:
10px solid #FEDE00;">
  <h1 style="font-size:100px; color:white;
margin:10px;">Sammy the Shark</h1>
  <p style="font-size:30px; color: white;">
<em>Senior Selachimorpha at DigitalOcean</em></p>
  <p style="font-size: 20px; color:#1F9AFE;"><a
href="Webpage FilePath";>About this site</a></p>
</div>
...
```

Only copy and add the highlighted `text-align` attribute as other parts of this HTML code will not be specific to your project. Save your file and reload it in the browser. You should receive something like this:



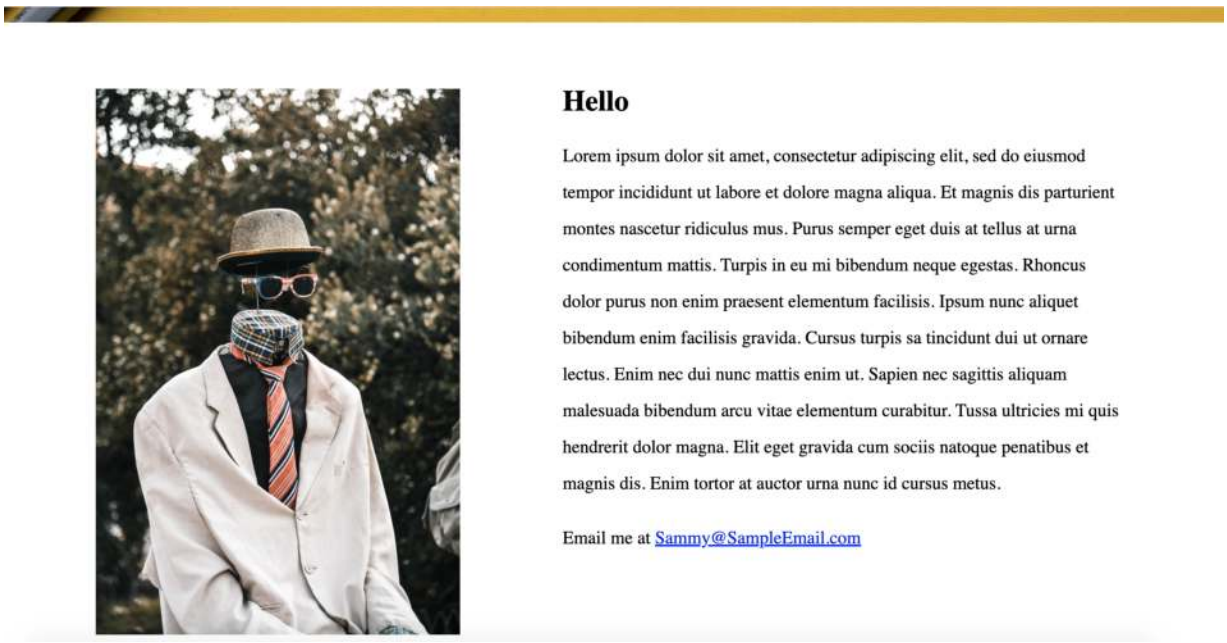
Centered content

You should now understand how to center and align text and have a section that looks like the top section of the demonstration site. In the next tutorial, we will recreate the middle section of the demonstration site.

How To Create the Body of Your Homepage With HTML

Written by Erin Glass

In this tutorial, we will recreate the body or middle section of our demonstration website using HTML `<div>` elements and HTML `style` properties.



Middle section

The middle section of our demonstration website contains a large profile image and a short paragraph of text displayed side by side. We can achieve this layout by using `<div>` containers that we learned about in a previous tutorial in this series. Note that if you continue learning front-end skills

such as CSS, there are improved methods for arranging content on a webpage that build upon the methods we'll use in this tutorial.

How To Add a Large Profile Image to Your Webpage

First, we'll add a large profile image as displayed in the demonstration site. Before we start, make sure you have selected a large profile image or other image to use. We'll be displaying our image at 400 by 600 pixels, so make sure your image size will work with those dimensions. If you do not have an image, you can download the image from our demo site. Once you have your image, save it in your images folder. (For a refresher on how to add images to webpages using HTML, please visit our tutorial HTML Images from earlier in this tutorial series).

Next, copy the following code snippet after the last closing `</div>` and before the closing `<body>` tag in your "index.html" file:

```
...
<!--Second section-->
<img src="images/large-profile.jpg"
style="height:600px; margin:100px; float: left;"
alt="A pretend invisible person wearing a hat,
glasses, and coat.">
...
```

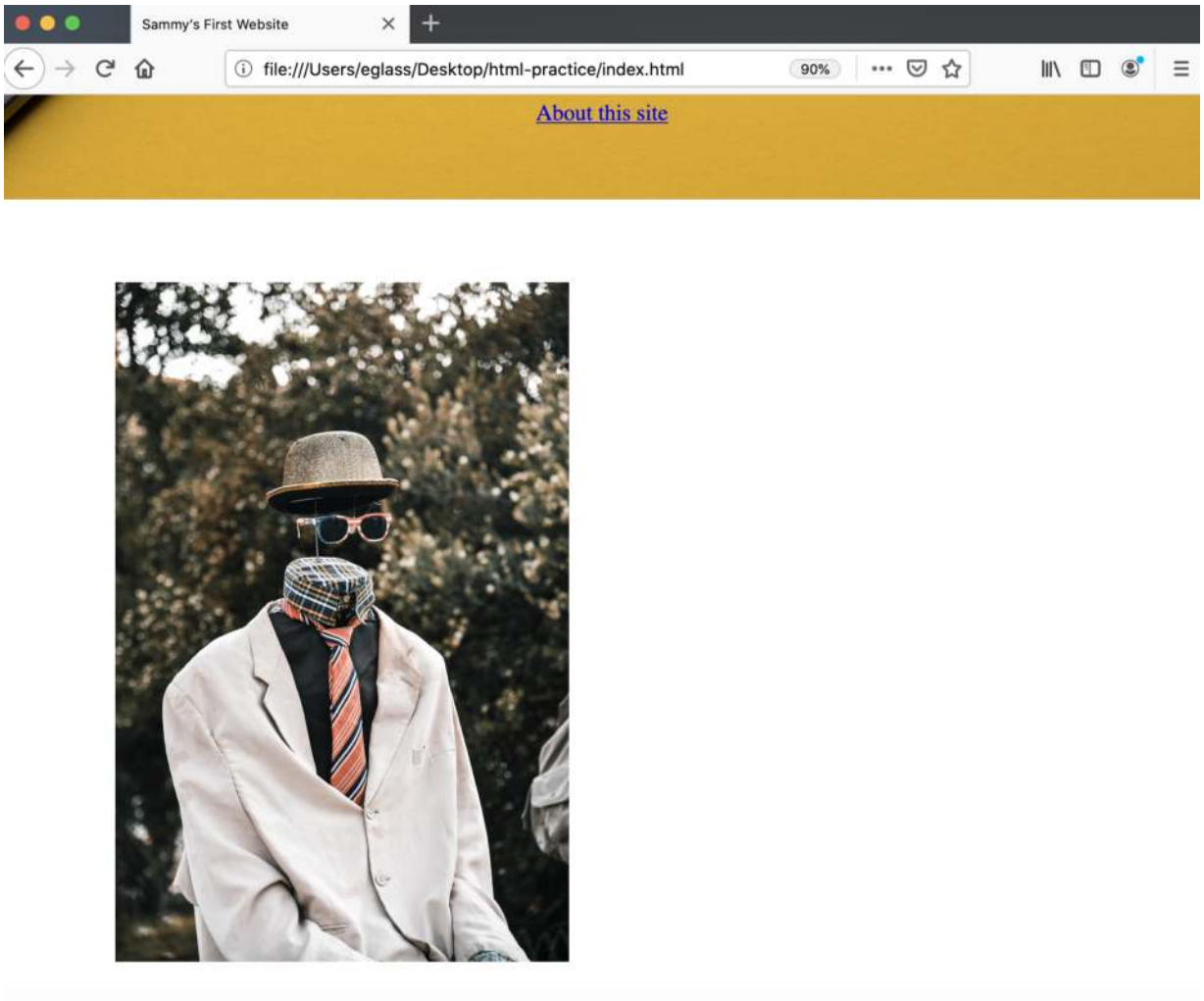
Let's pause briefly to review each part of this code snippet:

- The `<!--Second section-->` is a comment that will not be read by the browser and is used to help organize our `html` file for the

purpose of human readability

- The `<img>` tag tells the browser we are inserting an image into the webpage.
- The `src="images/large-profile.jpg"` tells the browser where to find the image that is being displayed.
- The `style` attribute allows us to define the `height`, `margin`, and `float` properties. The `margin` property allows you to specify the size of blank space surrounding an HTML element. The `float` property allows you to "float" the image to the right and left side of the display while allowing text to flow around its side.
- The `alt` attribute allows you to add alternative text to your image to improve site accessibility to visitors who use screen readers. Don't forget to change the alternative text in this code snippet to a description that matches your image.

Save your "index.html" file and reload it in the browser. The section below the top section of your webpage should now look like this:



Middle section of webpage with large profile image

If you have errors, check to make sure that you have added all the HTML code in the correct area of the `index.html` file and that your image is located in the file path you specified with the `src` attribute.

How To Add an "About Me" Section to Your Webpage

Next, we will add a paragraph of text to the right of the image. Feel free to substitute the dummy text in this example with text of your own choosing.

We will create this text section by creating a `<div>` container and inserting text content inside.

In your "index.html" file, add the following code snippet after the image you added in the step above and before the closing `</body>` tag:

```
...
<div style="height:600px; margin:100px;">
  <h1>Hello </h1>
  <p style="line-height: 2.0; font-
size:20px;">Lorem ipsum dolor sit amet,
consectetur adipiscing elit,
  sed do eiusmod tempor incididunt ut labore et
dolore magna aliqua. Et magnis dis parturient
montes
  nascetur ridiculus mus. Purus semper eget duis
at tellus at urna condimentum mattis. Turpis in eu
mi
  bibendum neque egestas. Rhoncus dolor purus non
enim praesent elementum facilisis. Ipsum nunc
aliquet
  bibendum enim facilisis gravida. Cursus turpis
sa tincidunt dui ut ornare lectus. Enim nec dui
nunc
  mattis enim ut. Sapien nec sagittis aliquam
malesuada bibendum arcu vitae elementum curabitur.
Tussa
  ultricies mi quis hendrerit dolor magna. Elit
eget gravida cum sociis natoque penatibus et
magnis
  dis. Enim tortor at auctor urna nunc id cursus
metus.</p>
  <p style="line-height: 2.0; font-
```

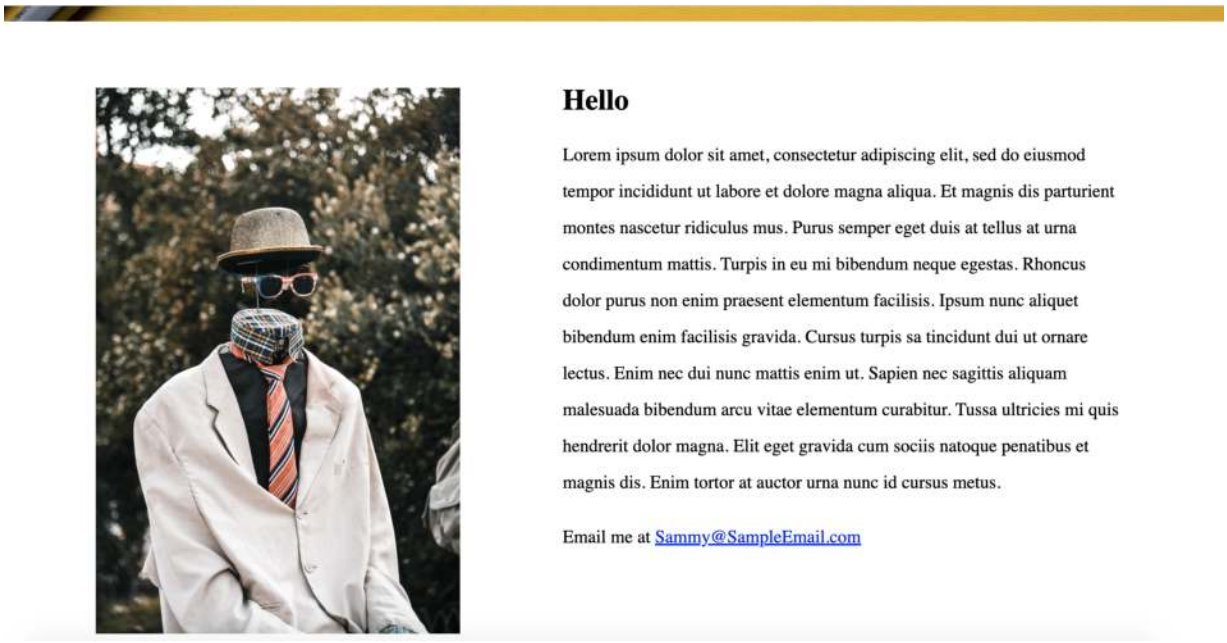
```
size:20px;">Email me at <a

href="mailto:Sammy@SampleEmail.com">Sammy@SampleEm
ail.com </a></p>
</div>
```

Let's pause briefly to review each part of this code snippet:

- The `<div style="height:600px; margin:100px;">` element creates a container that has a `height` of 600 pixels and `margin` of 100 pixels.
- The `<h1>` element adds a text header to our content.
- The two `<p style="line-height: 2.0; font-size:20px;">`tags create two paragraphs whose line height is expanded to 2.0 and whose font is 20 pixels.
- The 'Sammy@SampleEmail.com ` adds an email link to the email address.
- The closing `</div>` tag closes our `<div>` container.

Save your "index.html" file and reload it in the browser. The section below the top section of your webpage should now look like this:



Final styling for middle section

Your image and text should now be displayed as they are in the demonstration website. You may adjust the style properties in the code snippets to change the height, margins, font size or other style properties of your content.

Note that if your browser viewport is shrunk extensively, your text will eventually flow over into other elements on your page. To create layouts that are responsive to a range of devices, you'll need to learn additional front end skills such as CSS (tutorial series coming soon) and Flexbox.

You should now have an understanding of how to arrange images and text side by side using `<div>` containers, the `style` attribute, and style properties. In the next and final tutorial of the series, we'll learn how to create a website footer with the HTML `<footer>` element.

How To Add a Footer To Your Webpage With HTML

Written by Erin Glass

A website footer is the final block of content at the bottom of a webpage. Footers can contain any type of HTML content, including text, images, and links. In this final tutorial of the series, we'll create the following basic footer for our webpage using a `<footer>` element:

Footer

To get started, paste the following code snippet after your closing `</div>` tag and before your closing `</body>` tag:

```
. . .
<!--Footer-->
<footer style="height:auto; background-
color:#F7C201;">
  <h1><Made with ♡ at DigitalOcean</h1>
</footer>
. . .
```

In this snippet, `<!--Footer-->` is a comment that will not be read by the browser and is used to help organize our `html` file for the purpose of human readability. Below this comment, we have added a `<footer>` element, specified its background color, and used the style attribute to set its height to automatically adjust to the content inside. A `<footer>` element is a semantic element in that its name tells the developer the purpose of the content. This is helpful for developers as well as for site visitors who use screen readers. Otherwise, the `<footer>` element works just like a `<div>` element.

We have also added text content and an emoji inside the `<h1>` element. Feel free to change this text content with a message and emoji of your choosing.

Save your file and reload it in the browser to check the results. You should receive something like this:

Unstyled footer

Notice that our footer content is not quite like the one in the demonstration site. Our footer content has a white bottom margin and the text has different styling. To remove the bottom margin and style the text, add the highlighted attributes to your `<h1>` element like so:

```
<h1 style="color:white; padding:40px; margin:0;
text-align:center;">Made with ♡ at
DigitalOcean</h1>
```

Save the file and reload it in the browser. You should now have a footer styled in the same manner as the demonstration site pictured at the top of this tutorial.

In this tutorial you have learned how to create and style a footer. You can now explore adding different types of content and styling to your footer using elements from this tutorial series.