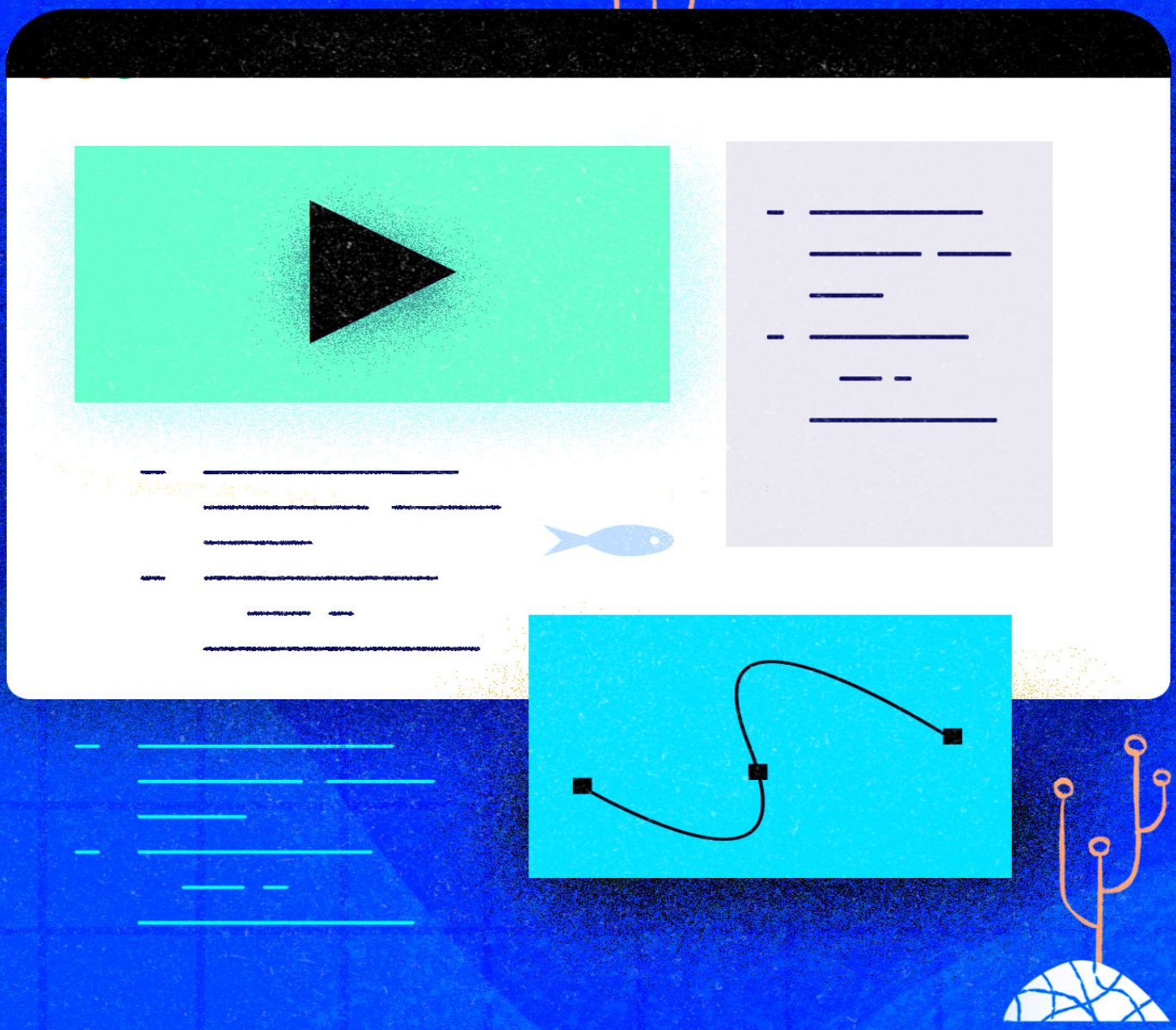
ERIN GLASS

HOW TO BUILD A WEBSITE WITH CSS AND HTML



This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0

International License.

ISBN 978-1-7358317-1-8

How To Build a Website with CSS and HTML

Erin Glass

DigitalOcean, New York City, New York, USA

2020-11

How To Build a Website with CSS and HTML

19. How To Add a Resume or Employment History Section To Your Website With CSS (Section 4)
20. How To Add Your Educational History and Skills To Your Website Using CSS (Section 5)
21. How To Create a Featured Quote Box On Your Website Using CSS (Section 6)
22. How To Create a Static Footer With HTML and CSS (Section 7)

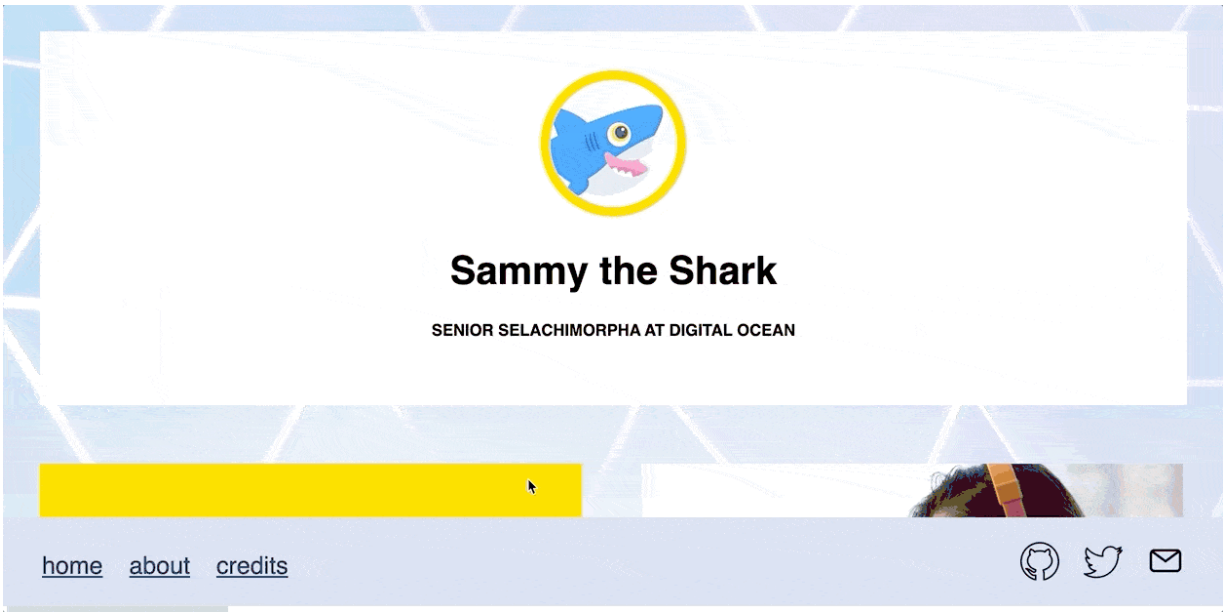
About DigitalOcean

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

This project-based book will introduce you to Cascading Style Sheets (CSS), a stylesheet language used to control the presentation of websites, by building a personal website using our demonstration site as a model. Though our demonstration site features Sammy the Shark, you can switch out Sammy's information with your own if you wish to personalize your site.



Gif of CSS demonstration site

Alongside HTML and JavaScript, CSS is one of the core technologies of the World Wide Web. If you have some understanding of HTML and are

looking to grow your front-end development skills, learning CSS is a great next step.

The first half of this book will introduce CSS through hands-on exercises and the second half of the book will provide steps for recreating the demonstration website. If you want to start building the demonstration website right away, you can start with the chapter How To Set Up Your CSS and HTML Website Project and proceed from there.

By the end of this CSS book, you will have files ready for deploying a website to the cloud, as well as an understanding of how to continue modifying the site's design with HTML and CSS. You will also have a foundation for learning additional front-end web development skills (such as JavaScript) and frameworks (like Tailwind).

Prerequisites

- A code editor like Visual Studio Code or Atom. This series will use 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. This book will use 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).
- Familiarity with HTML. If you aren't familiar with HTML or would like a refresher, you can follow the first ten tutorials of our series How

To Build a Website With HTML before starting this series.

Once you have your prerequisites ready, you will be ready to start your CSS website project in the chapters ahead.

A Brief Introduction To CSS

Written by Erin Glass

This tutorial will briefly introduce the historical origins of CSS and give a high-level overview of how CSS works with HTML. This tutorial will prepare you to follow the hands-on exercises and website building project in the tutorials ahead.

History of CSS

CSS was first introduced by Håkon Wium Lie in 1994 while working at the European Organization for Nuclear Research (CERN) alongside Tim Berners-Lee, the inventor of the World Wide Web. At the time, webpages were typically created exclusively with HTML, the Hypertext Markup Language that Berners-Lee had developed in the 1990s. However, HTML had been developed to describe the semantics of a web document's components (such as its headings and paragraphs) rather than provide style instructions. As the growing use of HTML to style webpages became increasingly unwieldy, CSS was introduced to provide a more efficient method for styling the display and layout of a website in conjunction with HTML.

How CSS Works With HTML

Websites that are built with HTML and CSS will typically consist of an HTML file that contains content such as text, image links, and HTML tags, and a CSS file that contains style rules that are applied to the HTML

content. For example, an HTML file might have header text (marked up with the HTML tag `<h1>`) and paragraph text (marked up with the HTML tag `<p>`). Its corresponding CSS file might have rules instructing the browser to make all header text 20 pixels in size and all paragraph text the color blue. These CSS style rules would then apply to all header and paragraph text wherever they appeared in the HTML document, without you having to add style instructions in the HTML document each time.

CSS is also a powerful tool for arranging website content. By giving size, color, and other properties to HTML elements, you can use CSS to create content boxes that structure and style the layout of a webpage.

Conclusion

In the tutorials ahead, you will use CSS to style text, image, and other HTML elements as well as style and control the layout of a webpage. To get started, you'll first need to create a few files and folders where you'll practice writing HTML and CSS code. In the next tutorial, you will be guided through the steps of setting up your CSS and HTML project using the freely-available code editor Visual Studio Code.

How To Set Up Your CSS and HTML Practice Project With a Code Editor

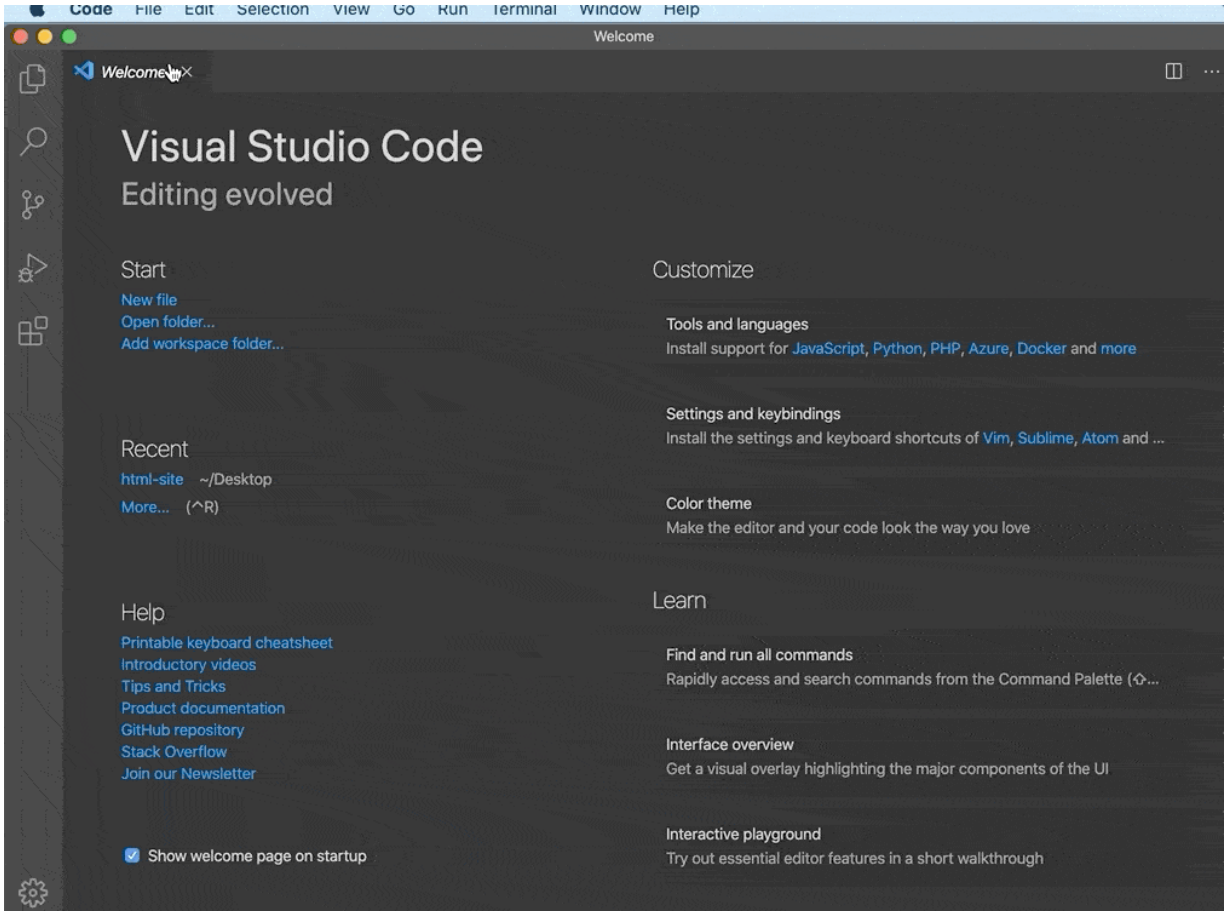
Written by Erin Glass

In this tutorial, you will set up the folders and files necessary for exploring CSS and building a website. Using a code editor, you will create a project directory for our website, a folder and file for our CSS code, a file for our HTML code, and a folder for images. This tutorial series uses Visual Studio Code, a code editor freely-available for Mac, Windows, or Linux, but you may use whichever code editor you prefer. Note that certain instructions may need to be slightly modified if you use a different editor.

How To Create HTML and CSS Files and Folders

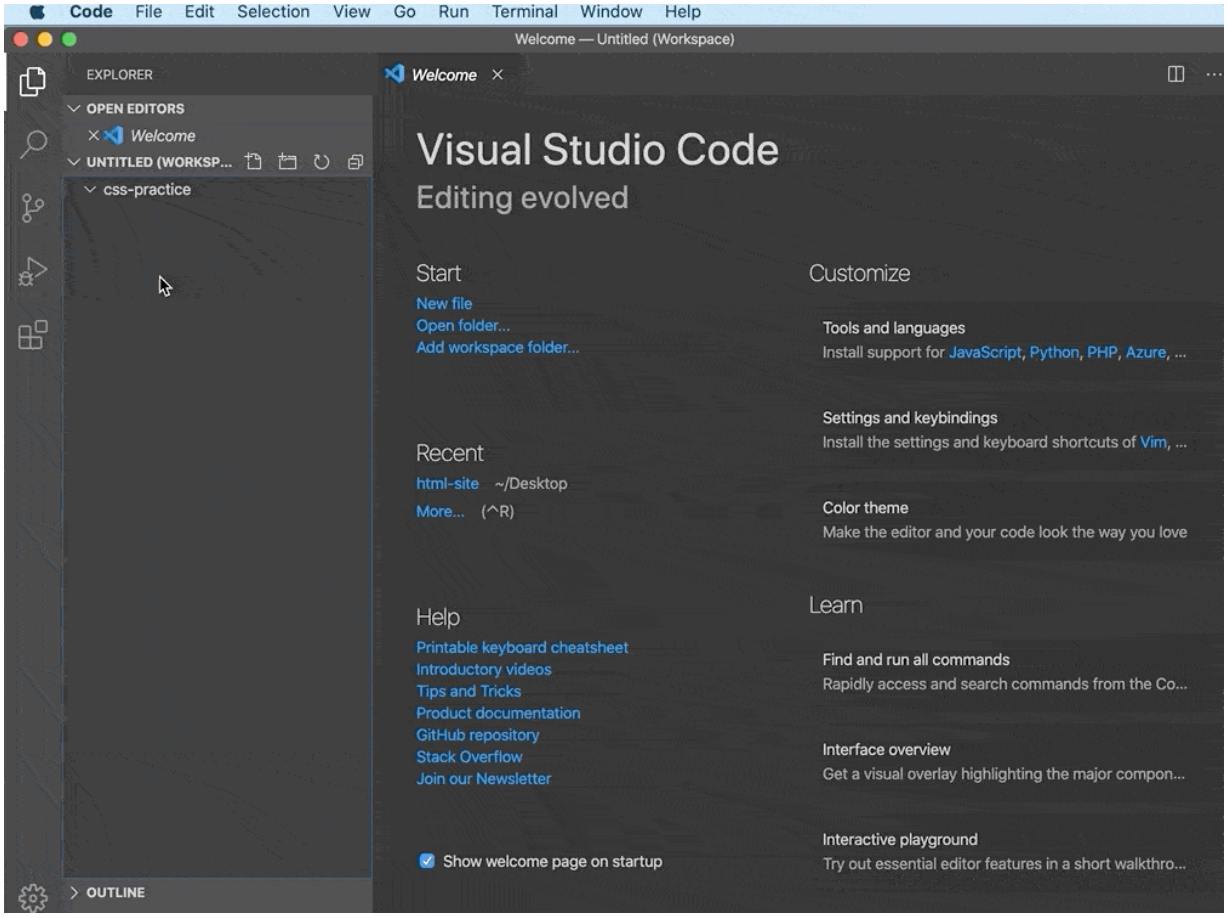
After opening your preferred text editor, open up a new project folder and name it `css-practice`. You'll use this folder to store all of the files and folders you create in the course of this tutorial series.

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 `css-practice`:



Gif of process of adding a project folder in Visual Studio Code

Next, create a new folder inside `css-practice` and name it `css`. Inside this folder, open up a new file in your project directory and save it as `styles.css`—this is the file you'll use to store our CSS style rules. If you are using Visual Studio Code, you can create a new folder by using `Right Click`(on Windows) or `CTRL + Left Click` (on Mac) on the `css-practice` folder, selecting "New Folder" and creating the `css` folder. Then, click `Right Click`(on Windows) or `CTRL + Left Click` (on Mac) on the new `css` folder, select "New File", and create the file `styles.css` as illustrated in the gif below:



Gif of how to add CSS folder and file

Save the file and leave it open.

You also need to create a file to add our HTML content—the text, images, and HTML elements that will be rendered in the browser. In the project directory `css-practice`, open an additional new file and save it as `index.html` in the same way you created the `styles.css` file above. Make sure to save this `index.html` file in the `css-practice` folder and not in the `css` folder.

Next, you need to add a line of code in the `index.html` document that instructs the browser to use the `styles.css` file as our style sheet. To do

this, you'll use the HTML `<link>` tag and link to the `styles.css` file. Add the following code snippet to your HTML document:

index.html

```
<link rel="stylesheet" href="css/styles.css">
```

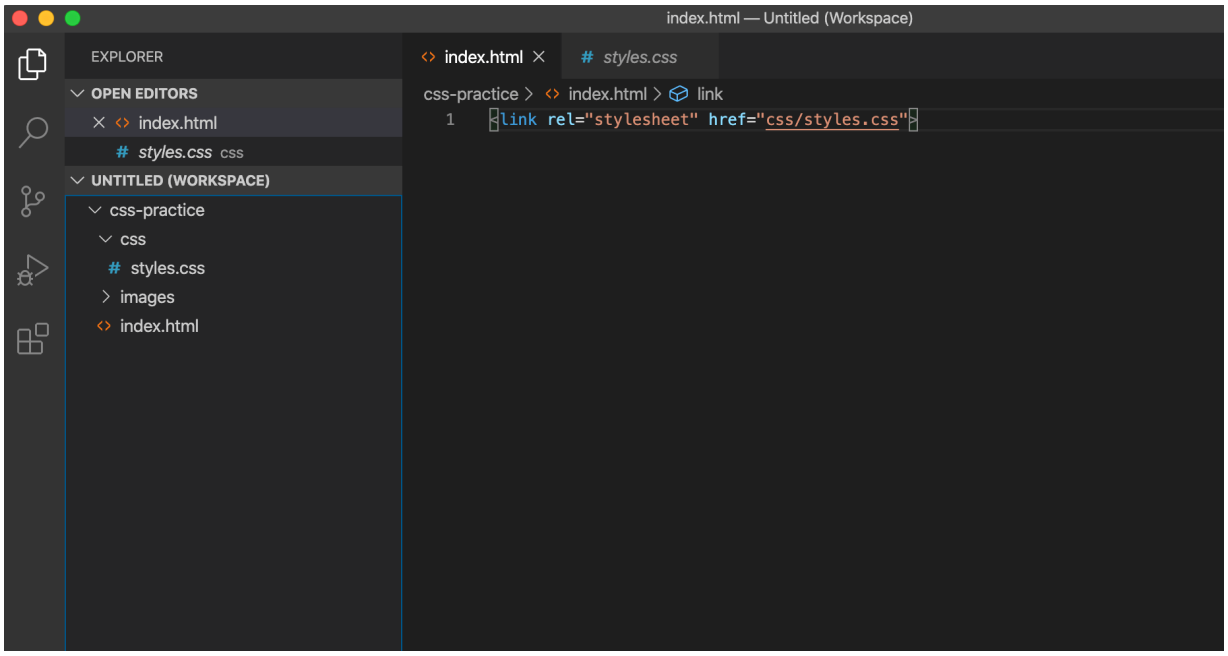
This code snippet tells the browser to interpret the HTML code according to the stylesheet located at `css/styles.css`. Make sure you don't erase this line while adding or deleting content to your `index.html` file throughout this tutorial series. Save your `index.html` file and keep it open.

Finally, create an additional folder inside `css-practice` and name it `images` in the same way that you created the `css` folder above. This folder will be where you save any images you use in this tutorial series.

You should now have a project folder named `css-practice` that contains the folders and files necessary for exploring CSS in this tutorial series:

- A folder named `css` that contains the file `styles.css`
- An empty folder named `images`
- A file named `index.html`

If you are using Visual Studio Code, your editor should now display the following file tree and the open files:



Visual Studio Code Editor with file tree displayed

Notice that the file names include extensions (`.html` and `.css`) that refer to the type of content they contain. You will add content to these files in our hands-on exercises in the tutorials that follow.

Debugging and Troubleshooting CSS and HTML

Precision is important when working with HTML and CSS. Even an extra space or mistyped character can keep your code from working as expected.

If your HTML or CSS code is not rendering in the browser as intended, make sure you have written the code exactly as written in the tutorial. Though we encourage you to manually write out the code for the purpose of learning, copy and pasting can be helpful at times to ensure that your code matches the examples.

HTML and CSS errors can be caused by a number of things. Check your markup and CSS rules for extra or missing spaces, missing or misspelled

tags, and missing or incorrect punctuation or characters. You should also make sure that you don't accidentally use "curly" or "smart" quotation marks like ‟ and ″ that are often used by word processors. Curly quotes are designed for human-readable text and will cause an error in your code as they are not recognized as quotation marks by browsers. By typing quotation marks directly into your code editor, you can make sure you are using the right kind.

Also, 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 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.

Conclusion

You are now ready to proceed with the tutorial series. In the next tutorial, you'll begin exploring how CSS rules are used to control the style and layout of HTML content on a webpage.

How To Understand and Create CSS Rules

Written by Erin Glass

In this tutorial, you will learn how to understand and create CSS rules (also known as rulesets) for styling and controlling the layout of HTML content. The tutorial will begin with an example CSS rule that makes `<h1>` HTML elements blue to study how CSS rules work in action before explaining each of the components of a CSS rule.

Prerequisites

To follow this tutorial, make sure you have set up the necessary files and folders as instructed in the previous tutorial How To Set Up You CSS and HTML Practice Project.

Exploring an Example CSS Rule

Below is an example of a CSS rule. Write the following rule into you `styles.css` file:

```
[label styles.css]
h1 {
  color: blue;
}
```

Save your `styles.css` file. Note that you have indented `color: blue` two spaces to the right. This indentation is a recommended best practice for writing CSS style rules as it makes the code more easily read by developers.

The rule you have just added instructs the browser to give any HTML text content tagged with the HTML element `<h1>` a blue color. (For a refresher on how HTML elements work, please visit our tutorial How To Use and Understand HTML elements).

Next, add a piece of HTML content that is tagged with the `<h1>` element to the `index.html` file (right below the `<link rel="stylesheet" href="css/styles.css">` line at the top of the document) :

index.html

```
<h1>A Sample Title</h1>
```

Save the file and load the HTML file in your browser to check your results. (For instructions on viewing an HTML file in your browser, please visit our tutorial step How To View An Offline HTML File In Your Browser).

In your browser, you should receive the following results:

[Webpage results] (https://assets.digitalocean.com/articles/how-to-build-a-website-with-css/a-simple-title.png)

If you don't have the same results, make sure you have saved both your `index.html` file and your `styles.css` file and that there are no errors in your code.

How To Understand the Components of a CSS Rule

Let's now examine the example CSS rule to understand each of its different components. In general, a CSS rule is composed of a selector, a declaration block, properties, and values. The diagram below illustrates how each of these parts are represented in a rule:

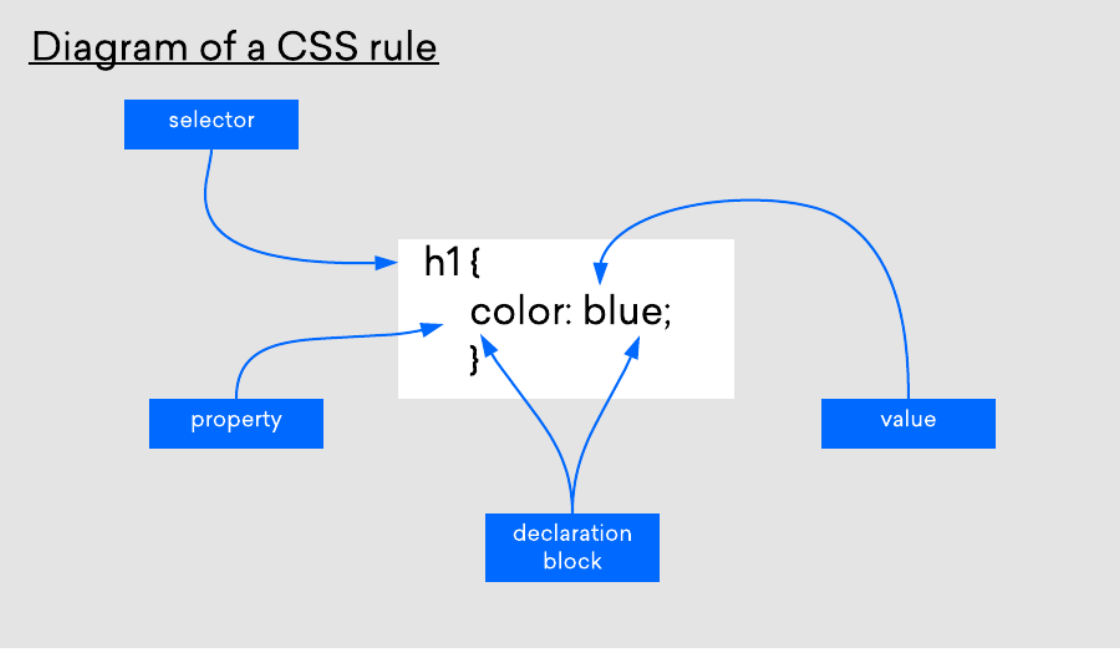


Diagram of a CSS rule

Let's now study each of these parts and how they relate to the example CSS rule.

- The selector indicates which type of content is to be styled by the CSS rule. The selector is placed at the beginning of the CSS rule and outside of the opening curly bracket. In the CSS example, the selector

is the `<h1>` HTML element, which is a tag selector. We'll learn about other types of selectors later on in the tutorial series.

- The declaration block is the part of the CSS rule that declares a style rule for the selector. The declaration block is placed inside of the curly brackets. In the CSS example, the declaration block is `color:blue;`.
- The property refers to the property of the HTML content that the CSS rule will modify, such as `font-size` or `color`. In the CSS example, the property is `color`. Note that a colon is appended after the property.
- The value refers to the specific value assigned to the property, such as `16px` or `blue`. In the example CSS rule, the value is `blue`. Note that a semicolon is appended after the value.

Once you declare a rule for a selector, every piece of content in your HTML document marked with that selector will be displayed according to the rule. Exceptions will occur, however, if a conflicting CSS rule is given precedence.

Conclusion

In this tutorial you examined all the components that are needed to write a complete CSS rule, including the selector, declaration block, properties, and values.

In the next tutorial, you will add multiple properties to a CSS rule and create different CSS rules for a single HTML document.

How To Declare Values For Multiple Properties In a CSS Rule

Written by Erin Glass

In this tutorial, you will learn how to declare values for multiple properties in a CSS rule. Declaring multiple properties in a single rule allows you to apply many style instructions—such as size, color, and alignment—to an element all at once. We'll also explore creating a variety of CSS rules that allow us to apply different styles to different pieces of content in a single HTML document.

Prerequisites

To follow this tutorial, make sure you have set up the necessary files and folders as instructed in a previous tutorial in this series How To Set Up You CSS and HTML Practice Project.

Creating a CSS Rule With Multiple Declarations

To add more than one declaration to a CSS rule, try modifying your `<h1>` rule in your `styles.css` file (or adding the entire code snippet if you haven't been following the tutorial series) so that it includes the additional highlighted declarations:

```
h1 {
  color: blue;
  font-size: 100px;
  font-family: Courier;
  text-align: center;
}
```

Save the file and reload your HTML document in your browser. (For instructions on loading an HTML file, please visit our tutorial step How To View An Offline HTML File In Your Browser). Your text should now be located in the center of the page, have a size of 100 pixels, and render with the `Courier` font:

Styled header text

In the next section, we'll add more CSS rules to extend the styling possibilities for the webpage's content.

Creating Multiple CSS Rules To Style HTML Content

In this section we'll add some more text to the `index.html` file using an HTML `<p>` element. We'll experiment with modifying its properties using

a new CSS ruleset that only applies to tags.

In the index.html file, add a line containing Some paragraph text below the existing <h1>A sample title<h1> line that you added in the How To Understand and Create CSS Rules tutorial:

index.html

```
<h1>A sample title</h1>
<p>Some paragraph text</p>
```

Save the index.html file and reload it in the browser window to check how the file is displayed. Your browser should render a blue heading that says "A sample title" and an unstyled paragraph below it that says "Some paragraph text" like the following example:

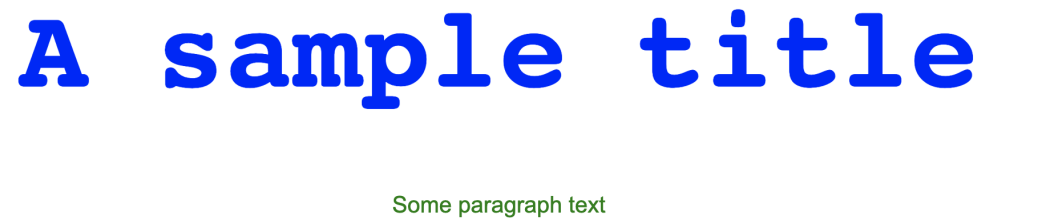
Webpage output with a blue <h1> heading and an unstyled element

Next, let's add a CSS rule to style the element. Return to your styles.css file and add the following ruleset at the bottom of the file:

styles.css

```
. . .
p {
  color: green;
  font-size: 20px;
  font-family: Arial, Helvetica, sans-serif;
  text-align: center;
}
```

Save the file and reload it in the browser window to check how the file is displayed. Your `<p>` text should now have the style you declared in the CSS rule you just created:



Webpage output with styled <p> text

Now that you have CSS rules for the `<h1>` and `<p>` elements, any text you mark up with these tags in your HTML document will take on the style rules that you declared for these elements in your `styles.css` file.

Further Practice

If you want to continue experimenting with CSS rules, try creating CSS rulesets for different HTML text elements—such as `<h2>`, `<h3>`, and `<h4>`—and using them to modify text in your `index.html` file. If you're stuck, you can copy the CSS rules in the following example and add them to your `styles.css` file:

styles.css

```
  . . .
h2 {
  color: red;
  font-size: 40px;
}

h3 {
  color: purple;
  font-size: 50px;
}

h4 {
  color: green;
  font-size: 60px;
}
```

Save your file and then add the following HTML content to your `index.html` file:

index.html

```
<h2> This is red text with a size of 40 pixels.
</h2>
<h3> This is purple text with a size of 50 pixels.
</h3>
<h4> This is green text with a size 60 pixels.
</h4>
```

Save the file and load `index.html` in your browser. You should receive the following results:

A Sample Title

Some paragraph text

This is red text with a size of 40 pixels.

This is purple text with a size of 50 pixels.

This is green text with a size 60 pixels.

Webpage content styled with multiple CSS rules

Conclusion

In this tutorial you experimented with specifying values for multiple properties using CSS. You also created multiple CSS rules for styling text content in an HTML document. You will expand upon both these skills when you begin building the demonstration website later on in the tutorial series. In the next tutorial, you will begin exploring how to style images with CSS.

How To Style Images With CSS

Written by Erin Glass

In this tutorial, you will learn how to style images with CSS to add a border, and change the shape, and size of the image. Using CSS to style images allows you to uniformly specify how images should appear across your website with only a few rulesets.

Prerequisites

To follow this tutorial, make sure you have set up the necessary files and folders as instructed in a previous tutorial in this series How To Set Up You CSS and HTML Practice Project.

Adding Images To `index.html`

First, you need to add an image to the `images` folder. You can download the image from the demonstration website or use any image in a JPEG/JPG or PNG format. This exercise will also work better if the dimensions of your image are around 150-200 pixels by 150-200 pixels.

Note: To download the image of Sammy the Shark, visit this 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 `images` folder.

Once you have selected an image, save it to your `images` folder as `small-profile.jpeg`. If you save it as a different file name, you'll need to modify the image file path in the step below.

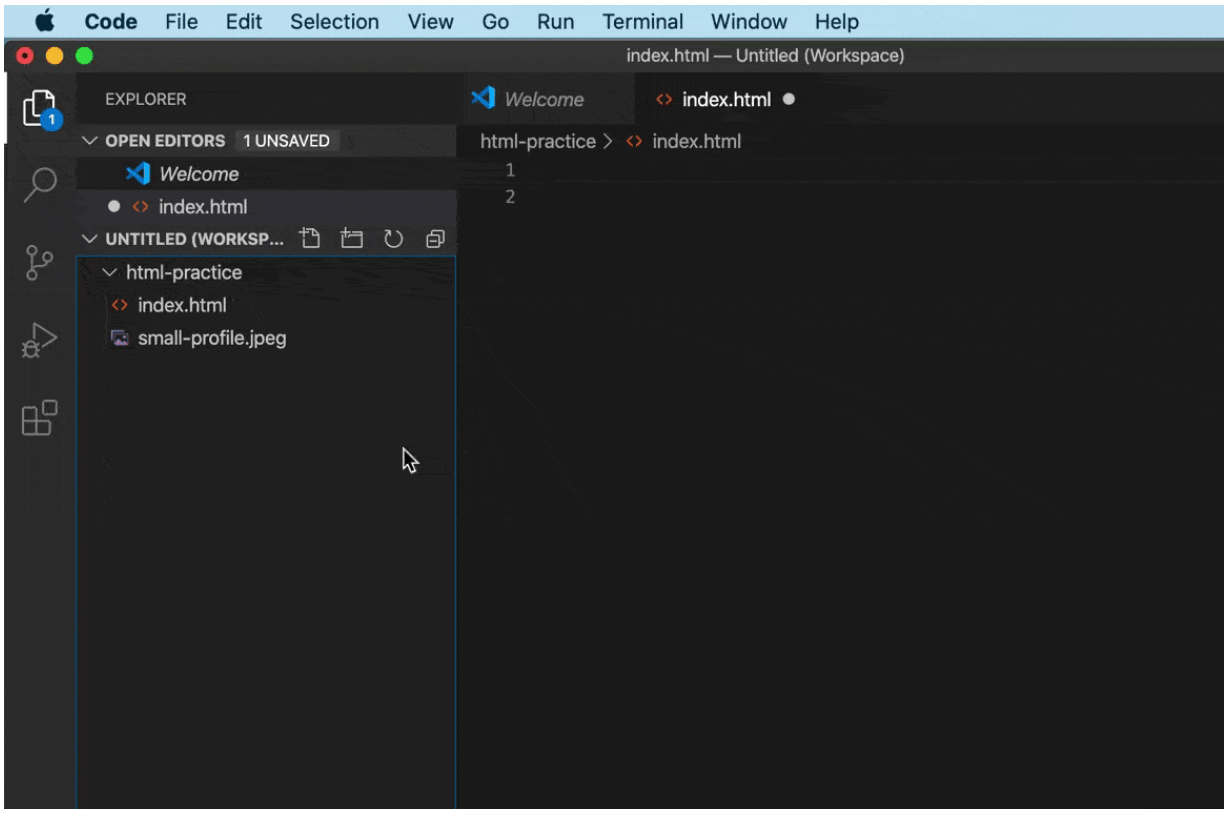
Next, erase all the content in your `index.html` file (except for the first line of code: `<link rel="stylesheet" href="css/styles.css">`) and add the following code snippet:

index.html

```
<img src="images/small-profile.jpeg" alt="Sammy the Shark, DigitalOcean's mascot">
```

This code snippet uses the `<img>` tag to insert an image and gives the browser the location of the image file (`images/small-profile.jpeg`). Make sure the highlighted file path is correct if you have changed the file name of your image.

Note: To copy the file path of your image using Visual Studio Code, hover over the icon of the image file in the left-hand panel, click `CTRL + Left Click` (on Macs) or `Right Click` (on Windows), and select "Copy Path." For an illustration of the process, please see the gif below:



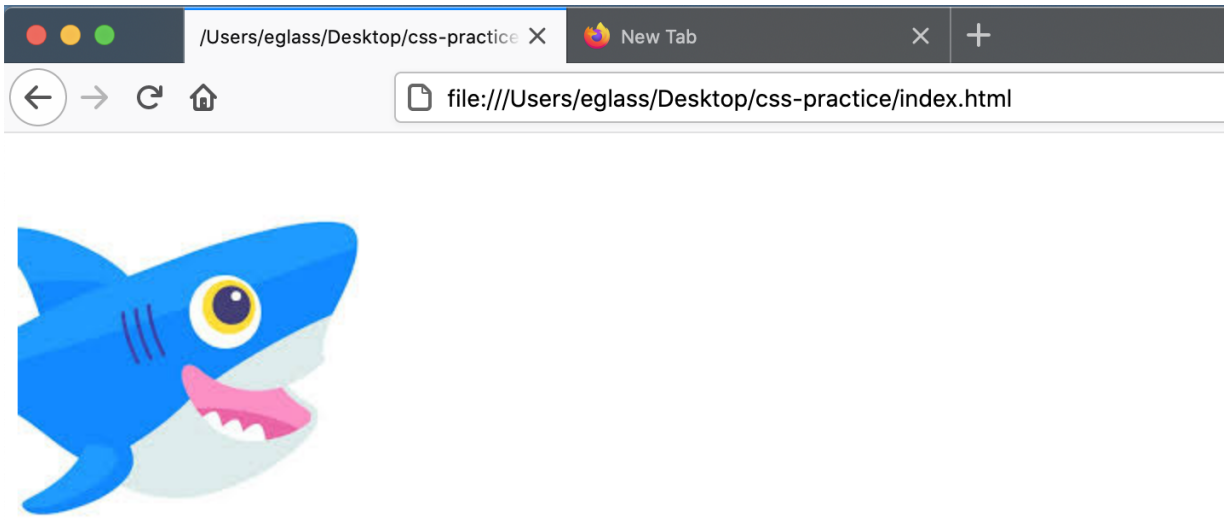
Gif of how to copy an image file path

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 you decided to publish the website online. Since the end goal is to create a publishable website, you will start using relative paths now when adding `<img>` elements to the document.

You have also added the alternative text `Sammy the Shark, DigitalOcean's mascot` using the `alt` attribute. When creating websites, alternative text should be added to all images to support site

accessibility for individuals who use screen readers. To read more about using alternative text with HTML, please visit the section on alternative text in our guide How To Add Images To Your Webpage Using HTML.

Save your `index.html` file and reload it in your browser. (For instructions on loading an HTML file, please visit our tutorial step How To View An Offline HTML File In Your Browser). You should receive a blank page with your image displayed:



Small profile image displayed in browser

If your image doesn't display, check your code for errors and confirm that you have the correct file path for the image.

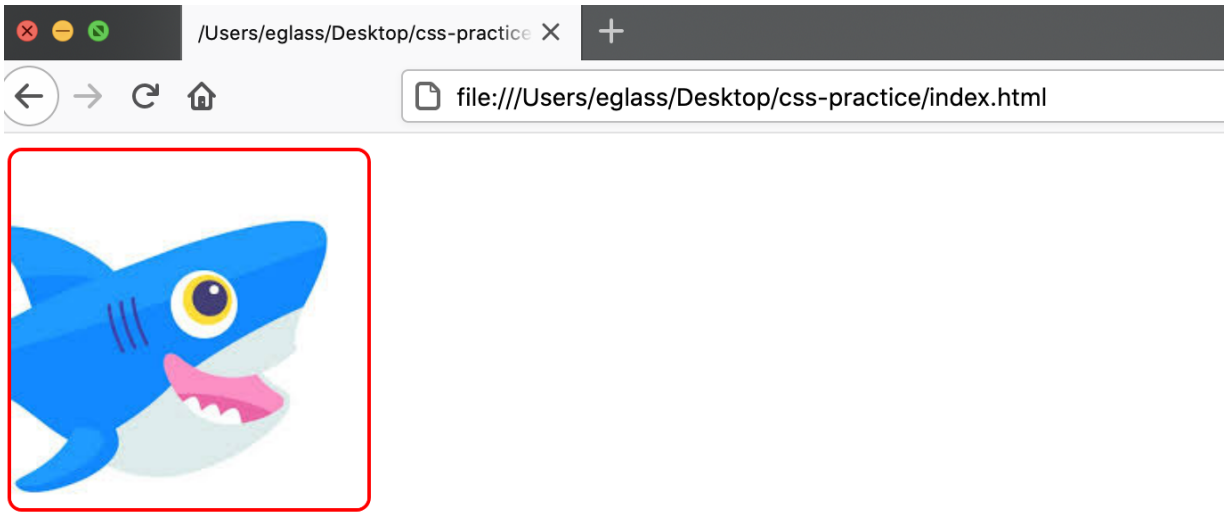
Adding Styles To Images

Now that `index.html` displays an image of Sammy the Shark (or the image of your choice), you'll add a CSS rule to style the image. In your `styles.css` file, erase everything (if you've been following along the tutorial series) and add the following ruleset at the bottom of the document:

styles.css

```
. . .
img {
  border: 2px solid red;
  border-radius: 8px;
  width: 200px;
}
```

Save your `styles.css` file and reload your `index.html` file in your browser. You should now receive an image with new style properties:



Webpage with styled small profile image

In this CSS rule, you have specified values for three different properties of the HTML `<img>` element. Let's pause to examine each of the different properties and values:

- The `border` property allows you to add a border to your image and specify the size, style, and color of the border. Notice that you can add multiple values for this CSS property. In this rule, you have specified a `solid`, `red` border with a width of `2px`.
- The `border-radius` property defines the radius of an element's corners, allowing you to round the edges of an element. In this rule,

you have specified 8 pixels as the size of the radius. Try changing this number to see how it affects the display of the image.

- The `width` property defines the width of the image. In this rule, you have specified the width to be 200 pixels wide. Note that if you leave the height undefined, the height of the image will automatically adjust to maintain the original proportions of the image. Try changing both the height and width at the same time to check what happens.

Exploring How Style is Applied To All Images

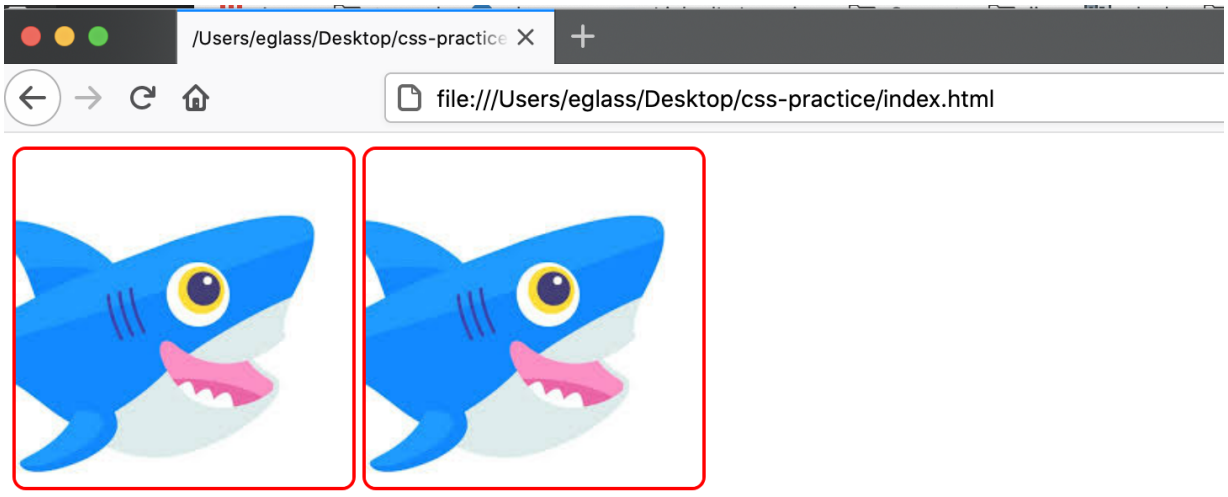
Note that if you add any additional images to your HTML document, they will also have the same styling. To study how this works, add a second image to your `index.html` file using the HTML `<img>` element. (You can copy and paste the first `<img>` element if you don't have a second image handy):

index.html

```
<img src="images/small-profile.jpeg" alt="Sammy
the Shark, DigitalOcean's mascot">
<img src="images/small-profile.jpeg" alt="Sammy
the Shark, DigitalOcean's mascot">
```

Make sure to change the highlighted section with your image file path. Save your `index.html` file and load it in the browser. Your webpage

should display two images styled with the same CSS ruleset for the `<img>` tag:



Webpage displaying two images with the same style

To continue exploring style possibilities for images, try changing the values in the CSS rule you just created in the `styles.css` file, saving the file, and reloading the `index.html` to check the results.

Conclusion

In this tutorial you explored how to style an image's border size, color, appearance, height, width, and border radius. You will return to image

styling when you begin building the demonstration website in the second half of this tutorial series.

Now that you are familiar with how to apply a set of style rules to all `<img>` elements, you may be curious how to apply different style rules to individual or groups of `<img>` elements. In the next tutorial, you will create CSS classes, which allow developers to sort HTML elements into different classes for different CSS styling.

How To Create Classes With CSS

Written by Erin Glass

In this tutorial, you will create a CSS class selector, which will allow you to apply CSS rules only to HTML elements that are assigned the class. CSS class selectors are useful when you want to apply different style rules for different instances of the same HTML element.

Prerequisites

To follow this tutorial, make sure you have set up the necessary files and folders as instructed in a previous tutorial in this series How To Set Up You CSS and HTML Practice Project.

How CSS Class Selectors Work

A CSS class selector allows you to assign style rules to HTML elements that you designate with that class rather than all instances of a certain element. Unlike HTML elements (such as `<p>`, `<h1>` or `<img>`), whose names are predetermined, class names are chosen by the developer when they create the class. Class names are always preceded by a `.`, which can help you distinguish between tag selectors and class selectors in CSS files.

A CSS rule for a class selector is written in the same way as a rule for a tag selector, with the exception of the `.` prepended to the class name:

```
.red-text {
  color: red;
}
```

To use a class when adding HTML content to your webpage, you must specify it in the opening tag of an HTML element using the class attribute in your HTML document like so:

```
<h1 class="red-text">Content.</element>
```

Creating a CSS Class Using a Class Selector

Let's begin exploring CSS classes in practice. Erase everything in your `styles.css` file and add the following code snippet to specify a rule for the class `red-text`:

styles.css

```
.red-text {
  color: red;
}
```

After adding the code snippet to your `styles.css` file, save the file.

Return to your `index.html` and erase everything but the first line of code `<link rel="stylesheet" href="css/styles.css">` that links to your CSS stylesheet. Then add the following HTML code snippet:

index.html

```
<p class="red-text">Here is the first sample of paragraph text.</p>
```

Note that the class name is not prepended here with a `.` as it is when being used as a selector for a CSS rule. Your entire `index.html` file should have the following contents:

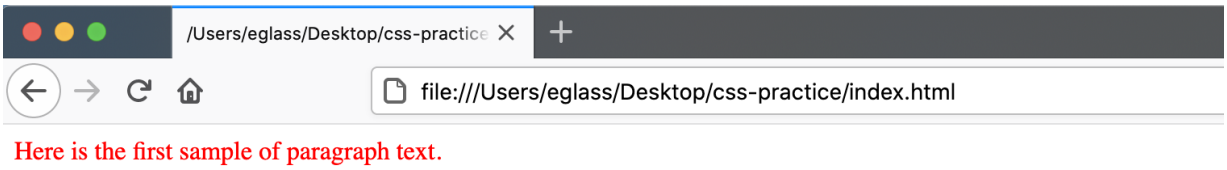
index.html

```
. . .
<link rel="stylesheet" href="css/styles.css">
<p class="red-text" Here is the first sample of paragraph text.</p>
```

In this code snippet you have added text using the HTML `<p>` tag. But you have also specified the `red-text` class by adding the highlighted class attribute `class="red-text"` inside the opening HTML tag.

Save your `index.html` file and load it in the browser. (For instructions on loading an HTML file, please visit our tutorial step How To View An Offline HTML File In Your Browser).

You should receive a webpage with red text:



Webpage output with red paragraph text

Let's add an additional CSS class to explore styling different pieces of `<p>` text content with different classes. Add the following code snippet to your `styles.css` file (after your CSS rule for "red-text"):

styles.css

```
.yellow-background-text {
  background-color: yellow;
}
```

This CSS rule declares that the class `yellow-background-text` is assigned the `yellow` value for the `background-color` property. Any HTML text element assigned this class will have a yellow background.

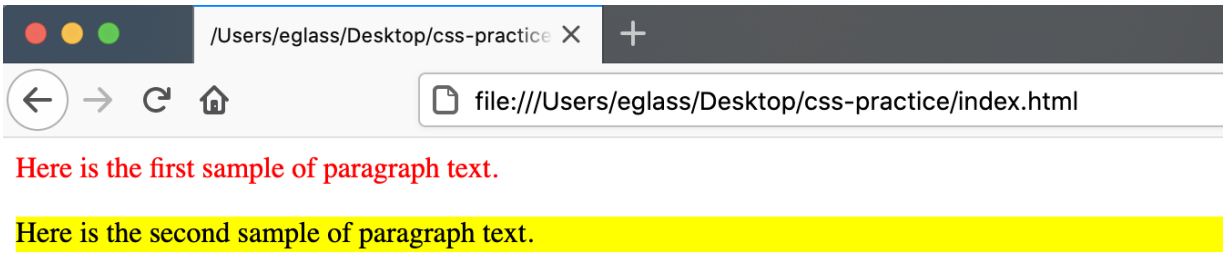
Note that the use of the word `text`in the class `yellow-background-*text*` is for human readability purposes only. You do not need to include the word `text` in your class names for classes assigned to HTML text.

To apply this new CSS class, return to your `index.html` file and add the following line of code to the bottom:

index.html

```
<p class="yellow-background-text"> Here is the second sample of paragraph text.<p>
```

In this code snippet, you have added some text content with the `<p>` element and specified the `yellow-background-text` class. Save the file and reload it in your browser. You should have a webpage with two different sentences, the first one red and the second one with a yellow background:



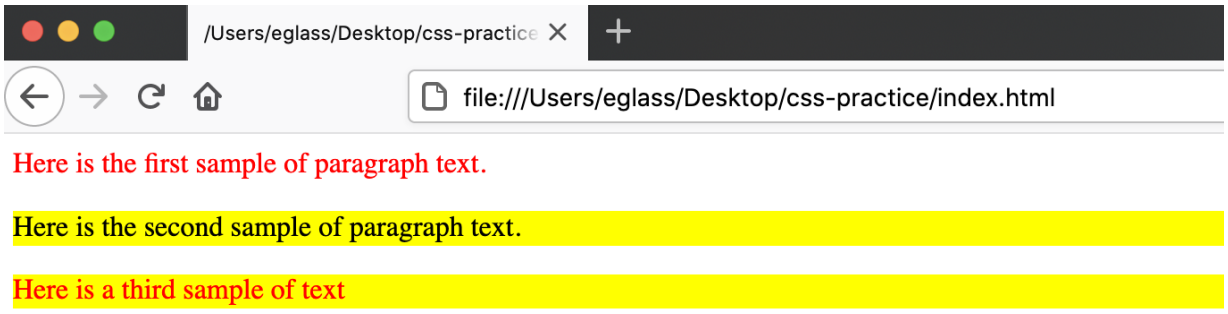
Webpage with text styled by two classes

Note that you can add more than one class to an HTML tag. Try adding both classes to a single text element by adding the following line to your `index.html` file:

index.html

```
<p class="red-text yellow-background-text">Here is a third sample of text.</p>
```

Note that the class names are only separated by a space. Save the file and reload it in the browser. You should receive something like this:



IWebpage with text styled by three classes

Your third line of text should now be styled according to the property values set in the `red-text` class and the `yellow-background-text` class and have a red font and yellow background.

Adding CSS Classes to Images

CSS classes can also be applied to other HTML elements, such as images. To explore using CSS classes for images, erase the content in your `styles.css` file and add the following code snippet:

```
.black-img {
  border: 5px dotted black;
  border-radius: 10%;
}
.yellow-img {
  border: 25px solid yellow;
  border-radius: 50%;
}
.red-img {
  border: 15px double red;
}
```

styles.css

Here you have created CSS rules for three different classes that can be applied to the HTML `<img>` tag. Before you move on, let's briefly study what we've declared in each ruleset:

- The first CSS rule declares that the class `black-img` should have a `black`, `dotted` border five pixels wide and a `border-radius` sized at 10%, which gives the element rounded corners.
- The second CSS rule declares that the class `yellow-img` should have a `yellow`, `solid` border 25 pixels wide and a `border-radius` sized at 50%, which gives the element a circular shape.
- The third CSS rule declares that the class `red-img` should have a `red`, `double` border 15 pixels wide. You have not set a border-radius,

so the border will conform to the element's shape.

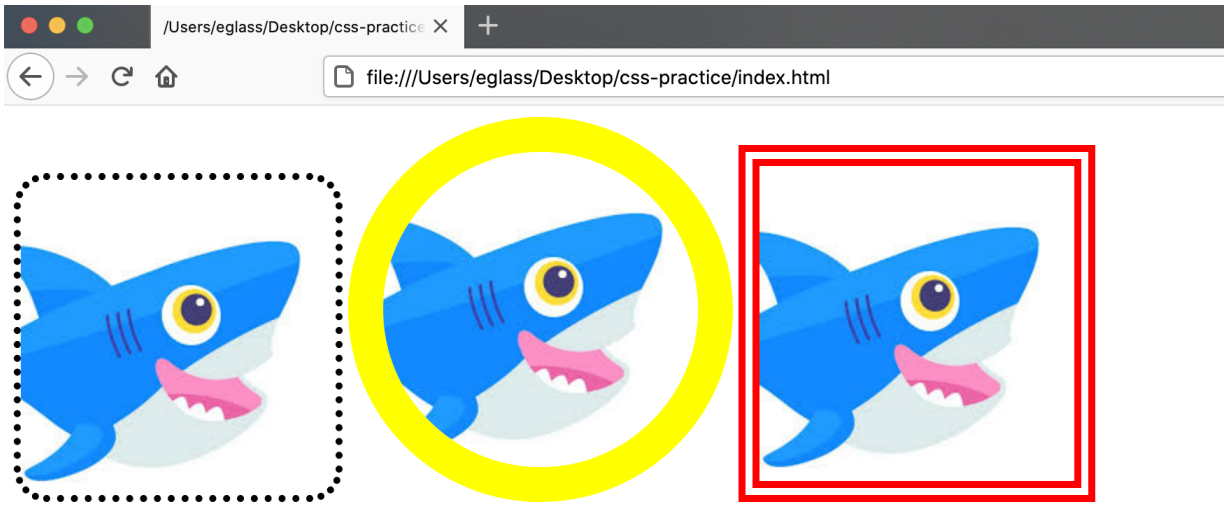
Save the `styles.css` file. Then erase everything from your `index.html` file (except for the first line of code: `<link rel="stylesheet" href="css/styles.css">`) and add the following code snippet:

index.html

```
<img src="https://css.sammy-codes.com/images/small-profile.jpeg" class="black-img">
<img src="https://css.sammy-codes.com/images/small-profile.jpeg" class="yellow-img">
<img src="https://css.sammy-codes.com/images/small-profile.jpeg" class="red-img">
```

Each of these three lines of HTML code add an image to the HTML document and assign it one of the three classes you just added to the `styles.css` file. Note that you are sourcing the image from an online location. You can also use your own image by specifying the file path as instructed in our tutorial How To Add Images To Your Webpage With HTML.

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



Webpage with images styled with three classes

Your webpage should now display three images, each styled with the different specifications of their assigned class.

To continue exploring CSS classes, trying creating new classes with different rulesets and applying them to different types of HTML content. Note that properties and values specified in class declaration blocks will only work on elements that they are intended for. For example, a `font-color` declaration will not change the color of an image border. Likewise, a `height` declaration will not change the size of the font.

Conclusion

You have now explored how to create classes, assign them specific property values, and apply them to text and image content. You will return to using classes when you begin building the website in the second half of this tutorial series.

In the next tutorial, you will create CSS ID selectors, which work similarly as class selectors with the exception of some unique features.

How To Create IDs with CSS

Written by Erin Glass

In this tutorial, you will create CSS ID selectors and learn how and why to use them when building a website with CSS and HTML.

CSS ID selectors function similarly to CSS class selectors. They allow you to create CSS rules that you can apply to HTML elements that have a unique ID attribute. Like classes, ID names are chosen by the developer when they create a CSS rule using the ID selector. However, IDs are different from classes in that you can only use an individual ID once in an HTML document. Thus, you would only define IDs for items that appear on a page once like a top logo, a site title, or a navigation bar. In general, CSS IDs are used sparingly.

Prerequisites

To follow this tutorial, make sure you have set up the necessary files and folders as instructed in a previous tutorial in this series How To Set Up You CSS and HTML Practice Project.

Creating a CSS ID Selector

When creating a rule for an ID, a # is prepended to the ID's name:

```
#my-first-id {
  color: blue;
}
```

This CSS rule creates an ID named “my-first-id” and declares that any HTML text element assigned this ID will be blue.

Let’s now explore how IDs work in practice. First, make sure you have set up the necessary files and folders as instructed in a previous tutorial in this series How To Set Up You CSS and HTML Practice Project.

Erase everything in your `styles.css` file (if you have been following along with this series) and add the CSS rule above for “#my-first-id” to your `styles.css` file and save the file.

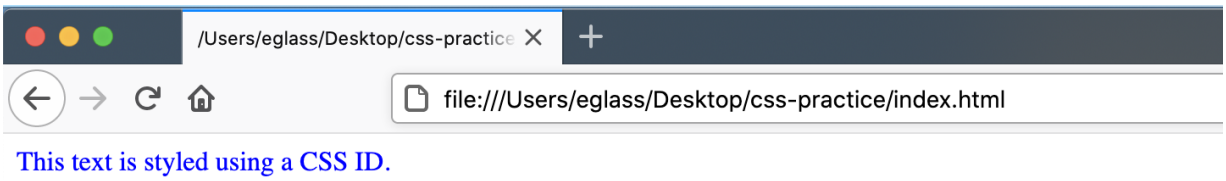
Next, return to your `index.html` file and erase everything (except for the first line of code: `<link rel="stylesheet" href="css/styles.css">`). Then add the following code snippet:

index.html

```
<p id="my-first-id">This text is styled using a CSS ID.</p>
```

Save the file and reload it in the browser. (For instructions on loading an HTML file, please visit our tutorial section How To View An Offline HTML File In Your Browser).

You should receive something like this:



Webpage with text styled with a CSS ID

In this exercise, you have created the CSS ID “my-first-id” in your `styles.css` file and then applied it to text content in your `index.html` file using the highlighted ID attribute. Note that you can create and apply IDs for any type of HTML content, such as images and `<div>` elements.

Conclusion

You explored how to create and use IDs for styling elements that only appear once on your webpage. In the next tutorial, you’ll learn about CSS pseudo-classes, a special type of class activated by certain states that can be triggered by user behavior.

How To Create Pseudo-classes With CSS

Written by Erin Glass

In this tutorial, you will create CSS pseudo-classes and learn how and why to use them. You will also practice using the `:hover` pseudo-class that allows us to change the style of an element when the user's cursor is hovering over it.

Pseudo-classes are CSS classes that are activated only during certain states. For example, the pseudo-class `:hover` can be used to change the appearance of an image or text element when the user's cursor hovers over the element. The pseudo-class `:visited` is often used to change the color of a link after a user has clicked on it.

Pseudo-classes are declared in CSS by appending a `:` and the name of the pseudo-class to a tag, class, or ID selector. This pseudo-class will then be automatically applied to any HTML content assigned the tag, class, or ID of the pseudo-class. You do not need to add any extra code to an HTML element to make a pseudo-class work.

Prerequisites

To follow this tutorial, make sure you have set up the necessary files and folders as instructed in a previous tutorial in this series How To Set Up You CSS and HTML Practice Project.

Creating a Pseudo-class with CSS

Let's now try a practical exercise to explore how pseudo-classes work. First, make sure you have set up the necessary files and folders as instructed in a previous tutorial in this series How To Set Up You CSS and HTML Practice Project.

Erase everything in your `styles.css` file (if you added content from previous tutorials) and add the pseudo-class below to your document:

styles.css

```
img:hover {
 border: 10px solid red;
}
```

In this code snippet, you have added the highlighted pseudo-class `:hover` to the `<img>` tag selector. Save the file and return to the `index.html` file and erase everything (except for the first line of code: `<link rel="stylesheet" href="css/styles.css">`). Then add the following snippet of HTML code to your `index.html` file:

index.html

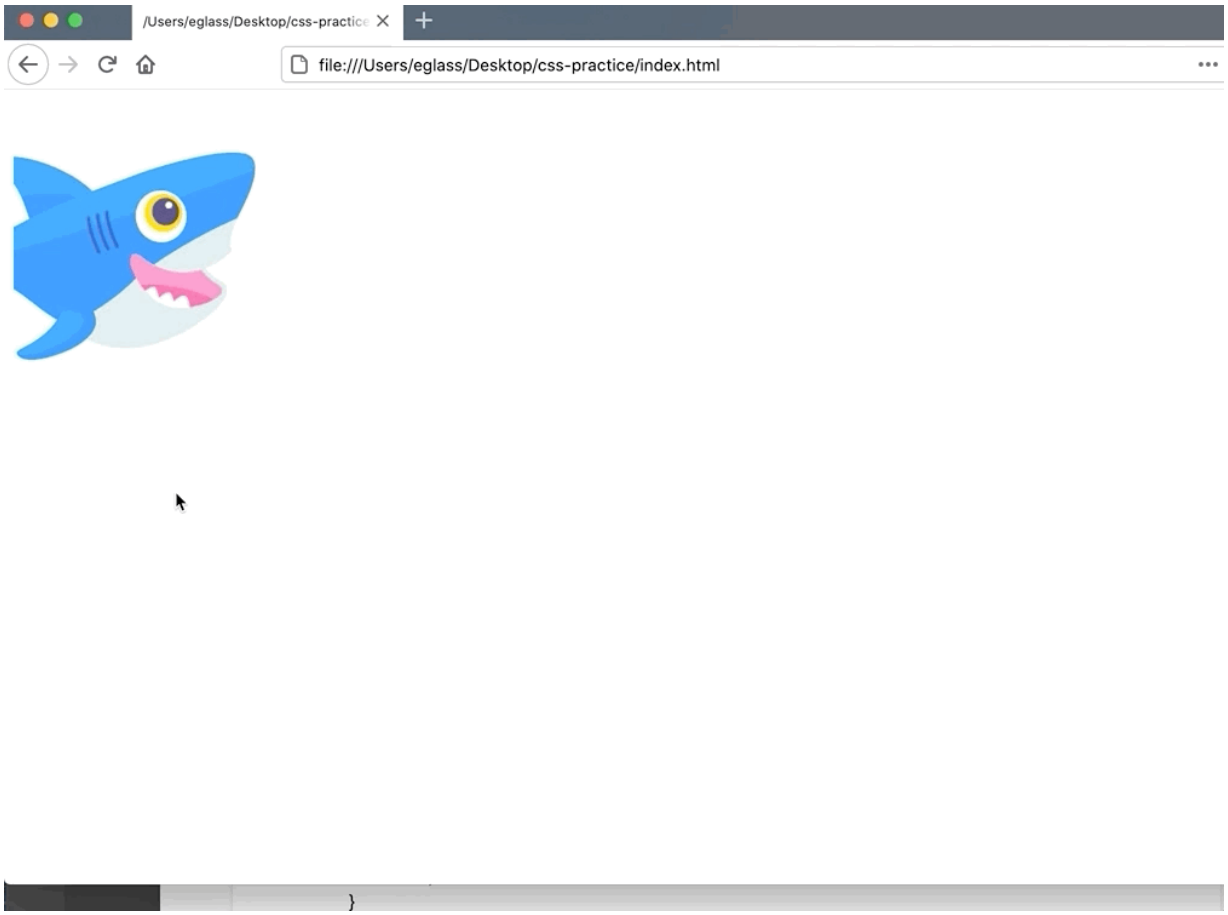
```
<img src="https://css.sammy-
codes.com/images/small-profile.jpeg">
```

Note that you are sourcing the image from an online location for convenience. You can also use your own image by specifying the file path

as instructed in our tutorial How To Add Images To Your Webpage With HTML.

Save your `index.html` file and load it in the browser. (For instructions on loading an HTML file, please visit our tutorial step How To View An Offline HTML File In Your Browser).

You should receive something like this:



Gif of cursor hovering over image to make red border appear

The webpage should now display an image of Sammy the Shark. Try hovering your cursor over the image. A solid red border 10 pixels wide should appear around the image when your cursor moves over the image.

Your browser automatically applies the pseudo-class `:hover` when your cursor interacts with an `img` element based on the rule that you added to `styles.css`.

You can use the `:hover` pseudo-class with text elements as well. If you'd like to try applying `:hover` to a text element, erase everything in your `styles.css` file and add the pseudo-class below to the document:

styles.css

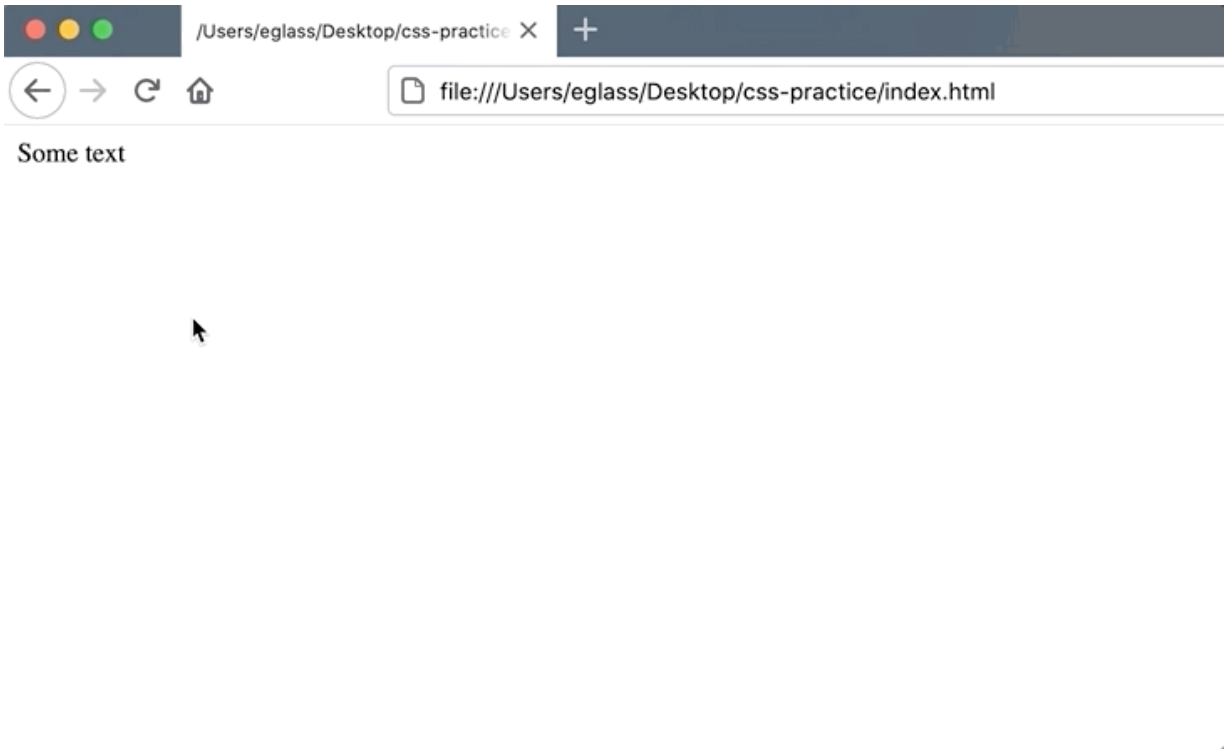
```
p:hover {
  font-size:100px;
  color:red;
}
```

Save the `styles.css` file. Return to the `index.html` file, erase everything (except for the first line of code: `<link rel="stylesheet" href="css/styles.css">`), and add the following code snippet:

index.html

```
<p>Some text</p>
```

Save your file and load it in the browser to check your results. You should receive a page with the text "Some text" that changes color and size when you hover your cursor over it:



Gif of cursor hovering over text to make it have larger size and red color

Conclusion

In this tutorial you explored how and why to use pseudo-classes. You also experimented with applying them to text and image based HTML elements. You will use pseudo-classes to build the footer of the demonstration website if you follow the second half of this tutorial series.

In the next tutorial, you'll learn about creating and styling the HTML `<div>` element, which can be used to structure the layout of a webpage.

How To Style the HTML element with CSS

Written by Erin Glass

This tutorial will introduce you to styling the HTML Content Division element—or `<div>` element—using CSS. The `<div>` element can be used to structure the layout of a page and break up a webpage into separate components for individual styling. In this tutorial, you will create and style `<div>` elements, as well as learn how to add and style other elements inside a `<div>` container. These skills will prepare you to use `<div>` elements as layout tools later on in the series when you begin recreating the demonstration website.

The `<div>` element is used by adding opening and closing `</div>` tags to an HTML document. On its own, the `<div>` element typically has little visual effect on the presentation of a webpage. To specify the size, color, and other properties of a `<div>` element, you can assign it style rules using CSS.

Prerequisites

To follow this tutorial, make sure you have set up the necessary files and folders as instructed in a previous tutorial in this series How To Set Up You CSS and HTML Practice Project.

Exploring the `<div>` Element in Practice

Let's try a hands-on exercise to study how the `<div>` element works. Erase everything in your `styles.css` file (if you added content from previous tutorials). Next, add the following CSS rule for the `<div>` tag selector:

styles.css

```
div {
 background-color: green;
 height: 100px;
 width: 100px;
}
```

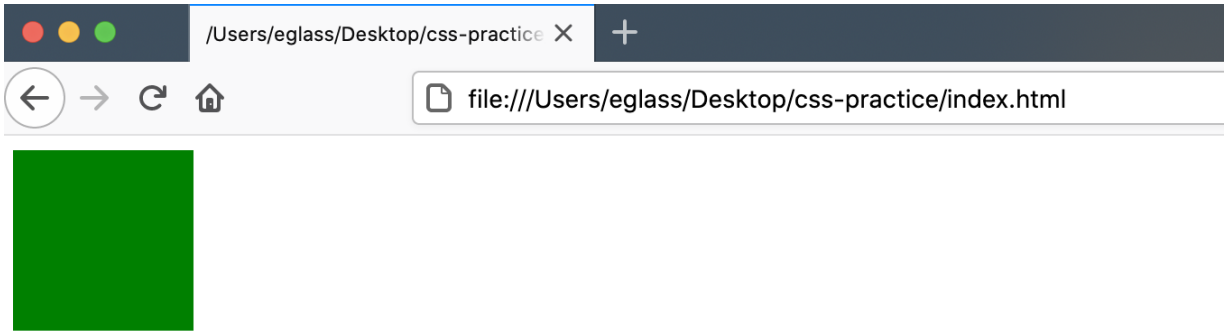
Save the `styles.css` file. Next, return to your `index.html` file, erase everything that's there (except for the first line of code: `<link rel="stylesheet" href="css/styles.css">`) and add the following code snippet:

index.html

```
<div></div>
```

Notice that the `<div>` element has opening and closing tags but does not require any content. Save the `index.html` file and reload it in your browser. (For instructions on loading an HTML file, please visit our tutorial step How To View An Offline HTML File In Your Browser).

Your webpage should display a green box 100 pixels wide and 100 pixels tall as specified by the CSS rule:



Now that you have a styling rule for your `<div>` element, every `<div>` element you add to your page will be styled in the precisely the same manner. However, when creating a website, it is unlikely that you will want all of your HTML `<div>` elements to be styled in the same way. For this reason, developers often create classes that they can use to style `<div>` elements in different ways.

To practice creating classes for `<div>` elements, erase the CSS rule you just created and add the following new three CSS rulesets to the `styles.css` file:

```
.div-1 {
  background-color: blue;
  height: 50px;
  width: 50px;
}

.div-2 {
  background-color: red;
  height: 100px;
  width: 100px;
}

.div-3 {
  background-color: yellow;
  height: 200px;
  width: 200px;
}
```

In this code snippet, you have created styling rules for three different classes: `div-1`, `div-2`, and `div-3`. Note that you have added a `.` before the class selector as required when declaring CSS rules for classes.

Save the `styles.css` file and return to your `index.html` file. Erase the `<div>` element you just created and, add the three `<div>` elements to your `index.html` file, applying a class to each one that corresponds to the CSS class selectors that you defined in `styles.css`:

index.html

```
<div class="div-1"></div>
<div class="div-2"></div>
<div class="div-3"></div>
```

Note that you have added the class as an attribute to the `<div>` tag by adding the class attribute and class name to each opening `<div>` tag. Save the file and reload it in your browser. You should receive something like this:

Your webpage should display three `<div>` elements, each styled with a different color and size according to their assigned CSS style rules. Note

that each element starts on its own new line as elements are block-level elements and have this default behavior.

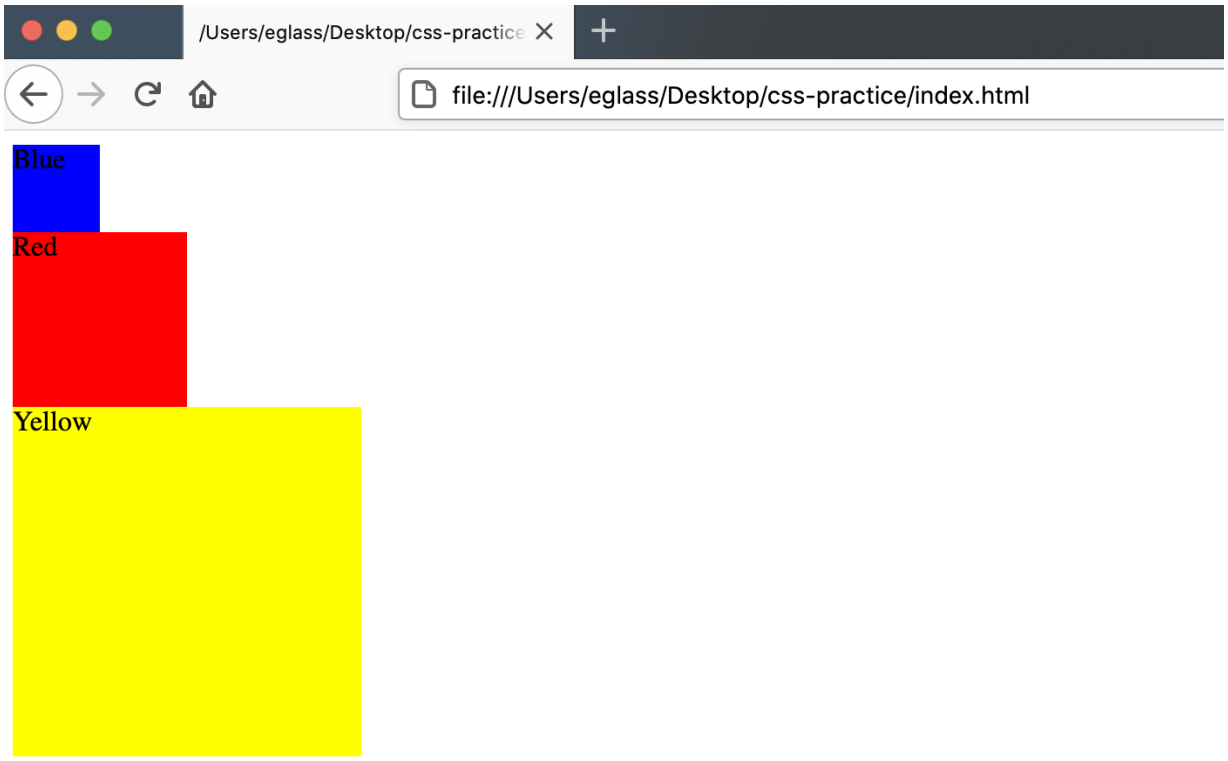
Adding and Styling Text in a Container

You can put text inside a container by inserting text in between the opening and closing tags. Try adding text inside each of the elements in your index.html file:

index.html

```
<div class="div-1">Blue</div>
<div class="div-2">Red</div>
<div class="div-3">Yellow</div>
```

Save the file and reload it in your browser. You should now have text displayed in each of your containers:



Webpage with elements containing text

You can add additional HTML elements to your text inside the elements. For example, try adding the HTML heading tags (<h2> to <h4>) to your text inside the tags in your index.html file:

```
<div class="div-1"><h2>Blue</h2></div>
<div class="div-2"><h3>Red</h3></div>
<div class="div-3"><h4>Yellow</h4></div>
```

Save the file and reload it in your browser. The text inside the containers should now be styled according to the default properties of the <h1> to <h4> tags:

Webpage with header text inside `<div>` containers

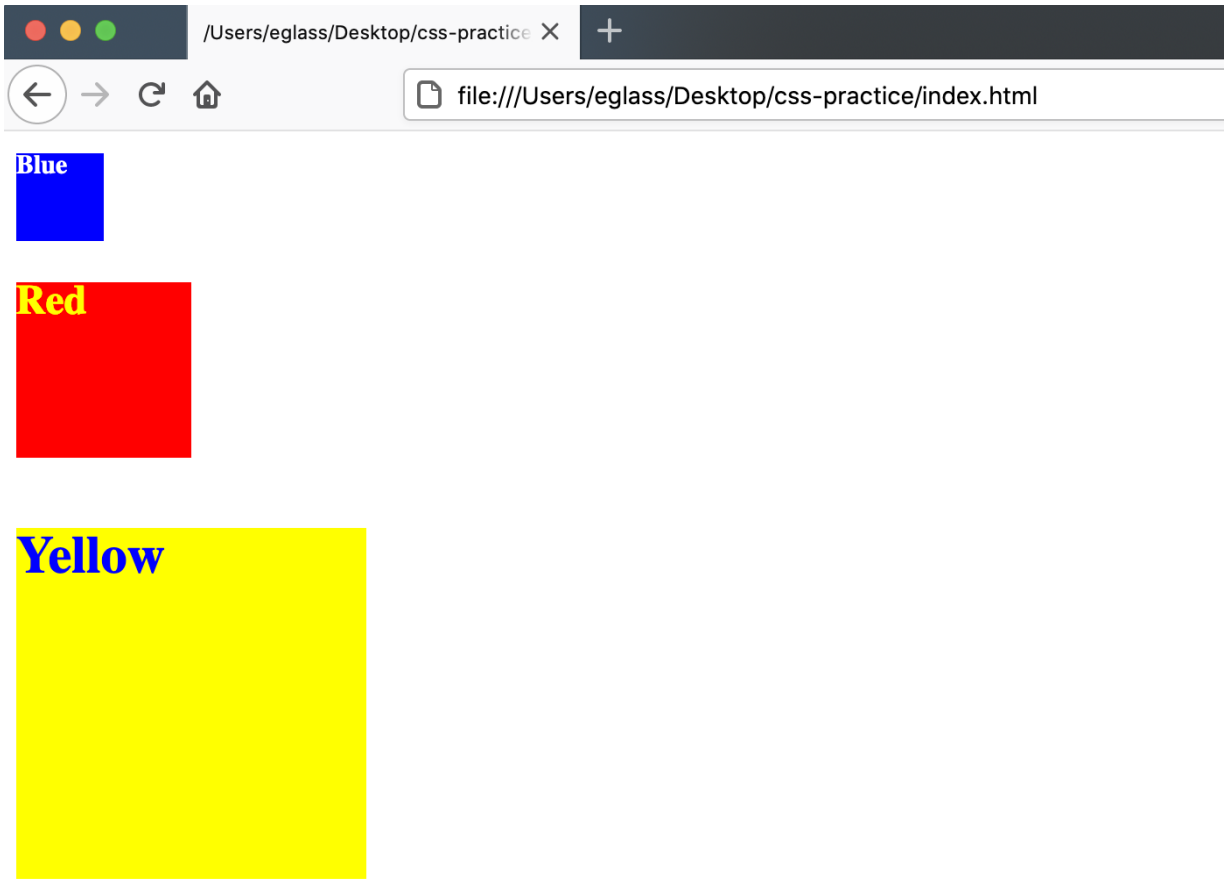
Note that the `<div>` elements have also adjusted their positions slightly. This repositioning is caused by the default margin properties of the `<h2>` through `<h4>` elements. You'll learn more about margins in the next tutorial on the CSS Box Model, but for now it is fine to ignore them

To style text inside the `<div>` containers, you can specify text property values in the rulesets for your `<div>` classes. Try adding the properties and values to your rulesets in your `styles.css` file as highlighted in the in the following code snippet:

styles.css

```
.div-1 {
  background-color: blue;
  height: 50px;
  width: 50px;
  font-size: 10px;
  color: white;
}

.div-2 {
  background-color: red;
  height: 100px;
  width: 100px;
  font-size: 20px;
  color: yellow;
}

.div-3 {
  background-color: yellow;
  height: 200px;
  width: 200px;
  font-size:30px;
  color: blue;
}
```

Save your `styles.css` file and reload the `index.html` file in your browser. The text inside the `<div>` containers should now be styled according to the CSS rules in your `styles.css` file:



Conclusion

In this tutorial you explored how to style the color and size of a `<div>` element and how to add and style text inside a `<div>` element. You will use the `<div>` element to control the layout of a page when you begin building the website. In the next tutorial, you will learn about the CSS Box Model, and how to use it to adjust the size of an element's content, padding, borders, and margin.

How To Adjust the Content, Padding, Border, and Margins of an HTML Element With CSS

Written by Erin Glass

In this tutorial, you will learn about the CSS Box Model, a model used to refer to the content, padding, border, and margins of an HTML element. Understanding the CSS Box Model is helpful for adjusting the size of any of these parts of an HTML element and understanding how the size and position of elements is determined. This tutorial will begin by explaining each of the boxes of the CSS Box Model and then move on to a practical exercise on adjusting their values using CSS style rules.

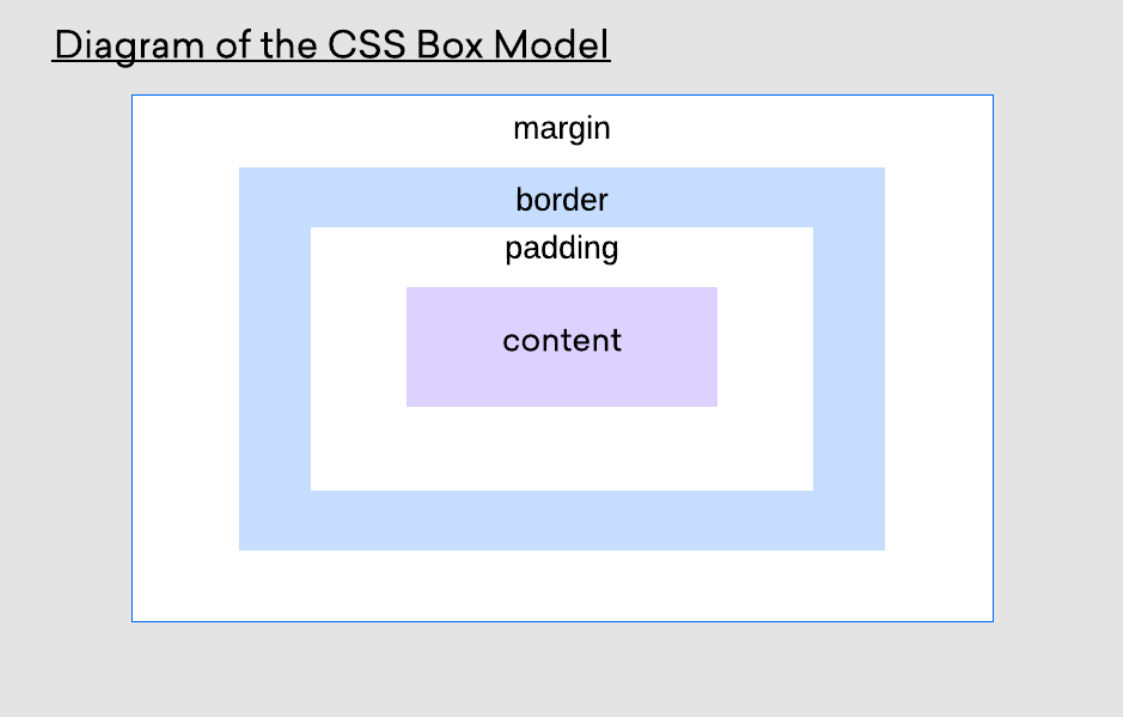


Diagram of CSS Box Model

Prerequisites

To follow this tutorial, make sure you have set up the necessary files and folders as instructed in a previous tutorial in this series How To Set Up You CSS and HTML Practice Project.

The CSS Box Model

An HTML element can be understood as a series of four overlapping boxes:

- The content box is the innermost box where the text or image content is placed. By default, its size is frequently set by the size of the content it contains. It is also the only box in the box model whose value is

typically not zero by default (if it contains content); in contrast, the padding, border, and margin of an element default to zero for many HTML elements (such as `<p>`, `<h1>`, and `<img>` elements) unless you specify otherwise. When you set values for the width and height of an element, you are typically changing the width and height of the content box.

- The padding box is the second overlapping box, which consists of a transparent space that surrounds the content box. By default, the padding of many HTML elements is set to zero. Increasing the size of an element's padding increases the distance between the content box and the border box.
- The border box is the third overlapping box that surrounds the padding box. By default, the border value of most HTML elements is set to zero. Increasing the size of an element's border increases the distance between the padding box and the margin box. Note that the color, thickness, and style of the border can be adjusted.
- The margin box is the fourth and final overlapping box that consists of transparent space outside of the border of an element. By default, the margin value of some HTML elements is set to zero, though some elements have specified margin values as their default, such as the `<h1>` through `<h6>` heading tags. Margins of two different elements are also allowed to overlap sometimes in a behavior called margin collapse. When this happens, the margin size defaults to the size of whichever element's margin is the largest.

Now that you are familiar with the components of the CSS Box Model, you can practice styling these different boxes to explore how they work

together to lay out and style an HTML element. You'll start by creating a `<div>` element that contains text content and then adjust the values of each of these boxes to help demonstrate their location in an element.

Adjusting The Content Size of an HTML Element With CSS

First, make sure you have set up the necessary files and folders as instructed in a previous tutorial in this series How To Set Up You CSS and HTML Practice Project.

Erase everything in your `styles.css` file (if the file contains content from previous tutorials) and add the following CSS rule to your `styles.css` file:

styles.css

```
.yellow-div {
  background-color:yellow;
}
```

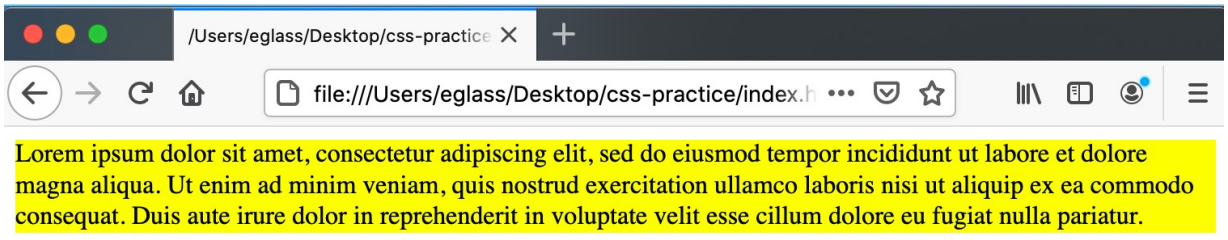
Save the `styles.css` file. You have just created a class using the class selector `yellow-div`. Any `<div>` element you assign this class will have a yellow background color.

Next, erase all the content in your `index.html` file (except for the first line of code: `<link rel="stylesheet" href="css/styles.css">`) and add the following code snippet:

index.html

```
<div class="yellow-div">
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.
</div>
```

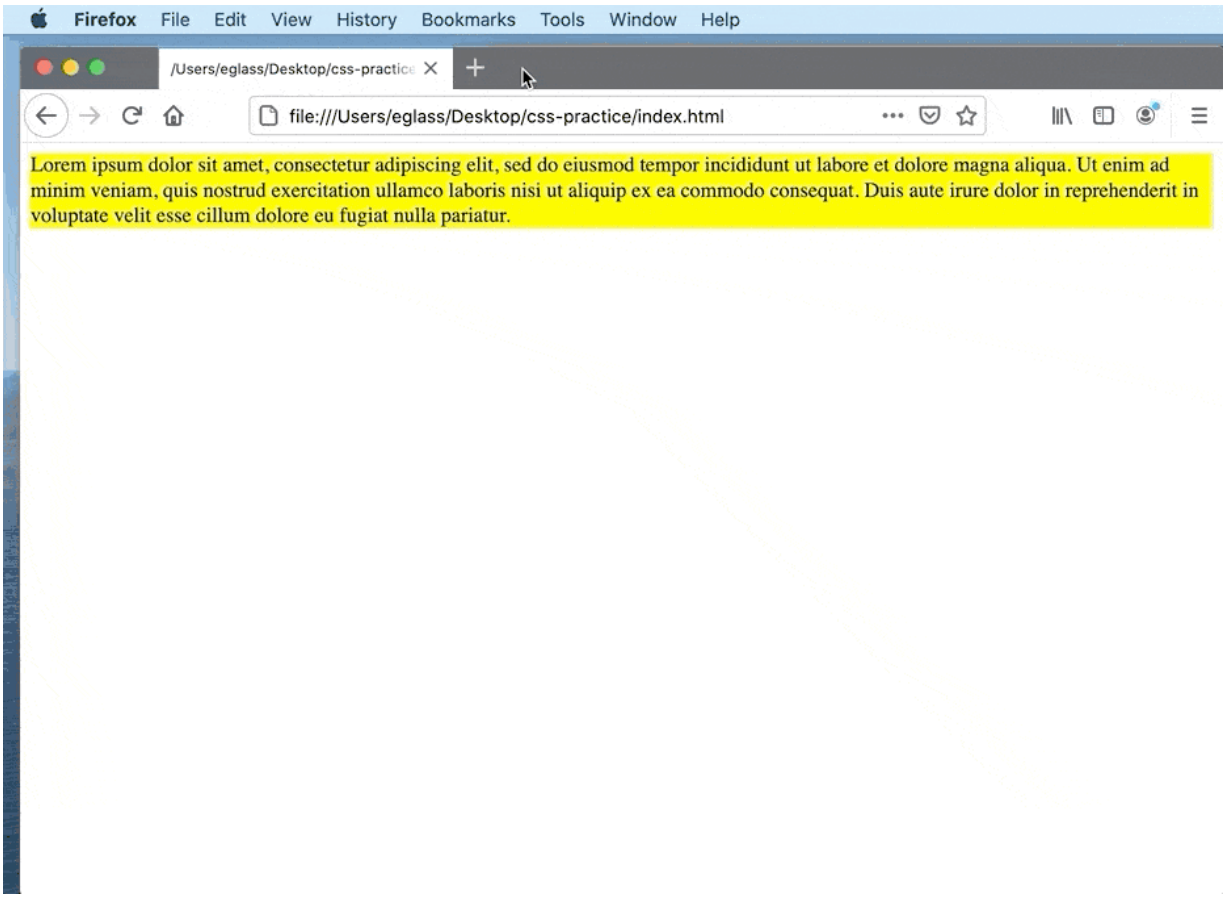
Save the file and load it in the browser. You should receive the following results:



Webpage with text in yellow containers

Your webpage should display a yellow box that contains the text content you added to the HTML file. Currently, only the innermost box—the content box—has a size and value; the padding, border, and margin are all set to zero. Notice also that the width and height of the yellow box is automatically determined by the size of the text content inside the `<div>` container. Try adding or subtracting text content to experiment with how the size of the `<div>` container changes accordingly.

Note: You can use Firefox's Web Developer tools to view the box model of an HTML element and the values set for each box. Navigate to the Tools menu item in the top menu bar and select "Web Developer/Toggle Tools" from the dropdown menu. The Developer Tools should appear in the bottom of your window. Click the the arrow icon on the far left of the tool kit menu and then click on the element that you wish to inspect. The box model of the selected element will show up in the bottom right of the Developer Tools window pane. You may need to expand the window to view it.



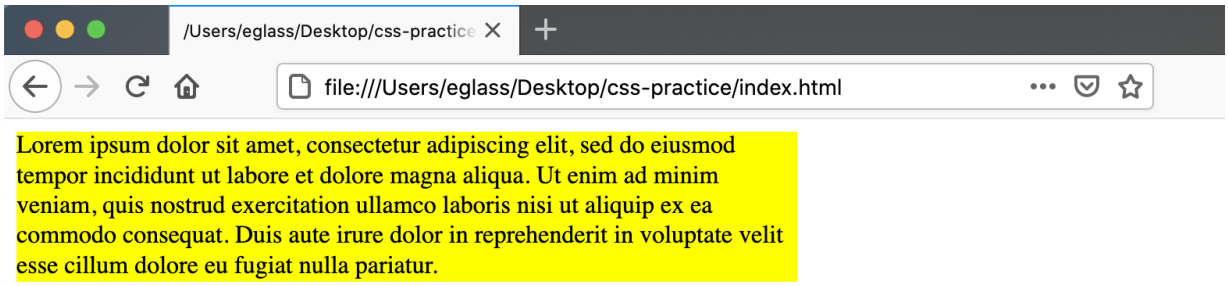
Gif of using Firefox Web Developer tools to view the box model of an element

Next, let's specify the width of the `<div>` container to study how that changes the presentation of the element in the browser. Add the following highlighted line to your CSS rule in your `styles.css` file to set the `width` to 500 pixels:

styles.css

```
.yellow-div {
  background-color:yellow;
  width: 500px;
}
```

Save the file and load it in your browser. Your `<div>` container should now be 500 pixels wide, with its height automatically adjusting to allow the text content to fit inside:



Webpage with text `div` container that is 500 pixels wide

Note that you can also specify the height of a `<div>` element instead and allow for the width to adjust automatically. Or you can specify both the height and width, but be aware that the content will spill over the `<div>` container if the `<div>` element is too small.

How To Adjust the Padding Size of an HTML Element With CSS

Next, let's increase the padding size to study how it changes the display of the `<div>` element. Add the following highlighted line to your CSS rule in your `styles.css` file to set the `padding` to 25 pixels:

styles.css

```
.yellow-div {
  background-color:yellow;
  width: 500px;
  padding:25px;
}
```

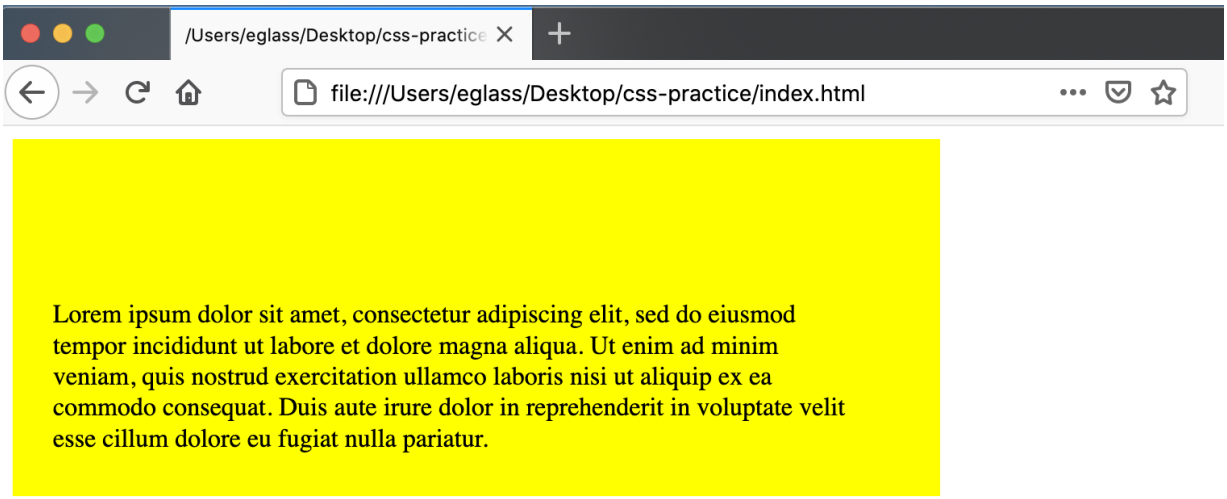
Save the `styles.css` file and reload the `index.html` file in your browser. The size of the yellow box should have expanded to allow for 25 pixels of space between the text content and the perimeter of the box: ! [Webpage with a yellow `<div>` container with width and padding specified](https://assets.digitalocean.com/articles/how-to-build-a-website-with-css/div-with-padding.png) You can change the size of the padding by adjusting the padding value size. You can also change the padding size of

specific sides of the element by using the following properties: `padding-left`, `padding-right`, `padding-top`, `padding-bottom`. For example, try replacing the declaration `padding:25px;` in your `styles.css` file with the highlighted snippet below:

styles.css

```
.yellow-div {
  background-color:yellow;
  width: 500px;
  padding-left:25px;
  padding-right: 50px;
  padding-top: 100px;
  padding-bottom: 25px;
}
```

Save the `styles.css` file and load the `index.html` file in your browser. You should receive something like this:



Knowing how to specify padding sizes for individual sides of an element can be useful when arranging content on a webpage.

Adjusting the Border Size, Color, and Style of an HTML Element With CSS

Let's now practice setting values for the border of an HTML element. The border property lets you set the size, the color, and the style (such as `solid`, `dashed`, `dotted`, `inset`, and `outset`) of an HTML element. These three values are set by assigning them to the border property like so:

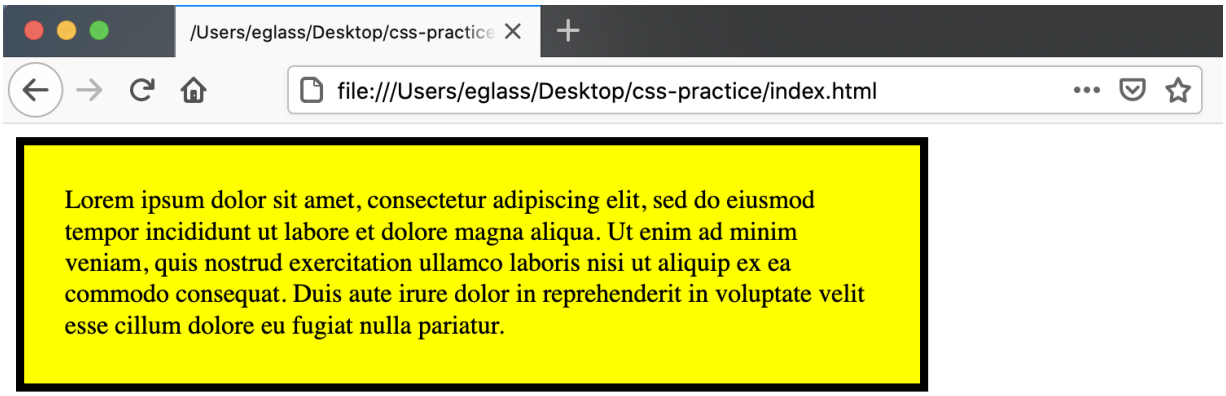
```
selector {
  border: size style color;
}
```

Try adding the following highlighted declaration to add a solid black border that is five pixels wide:

styles.css

```
.yellow-div {
  background-color:yellow;
  width: 500px;
  padding: 25px;
  border: 5px solid black;
}
```

(You may want to erase your different padding declarations from the previous tutorial section and replace them with the single `padding:25px;` declaration to keep the ruleset manageable). Save the `styles.css` file and reload `index.html` in your browser to inspect the changes. Your yellow box should now have a border with the values you set in the CSS rule:



Webpage with yellow `<div>`, padding, and border

You can try changing the values to study how they change the display of the element in the browser. Like with padding, you can also specify the border side you'd like to adjust with the properties `border-right`, `border-left`, `border-top`, `border-bottom`.

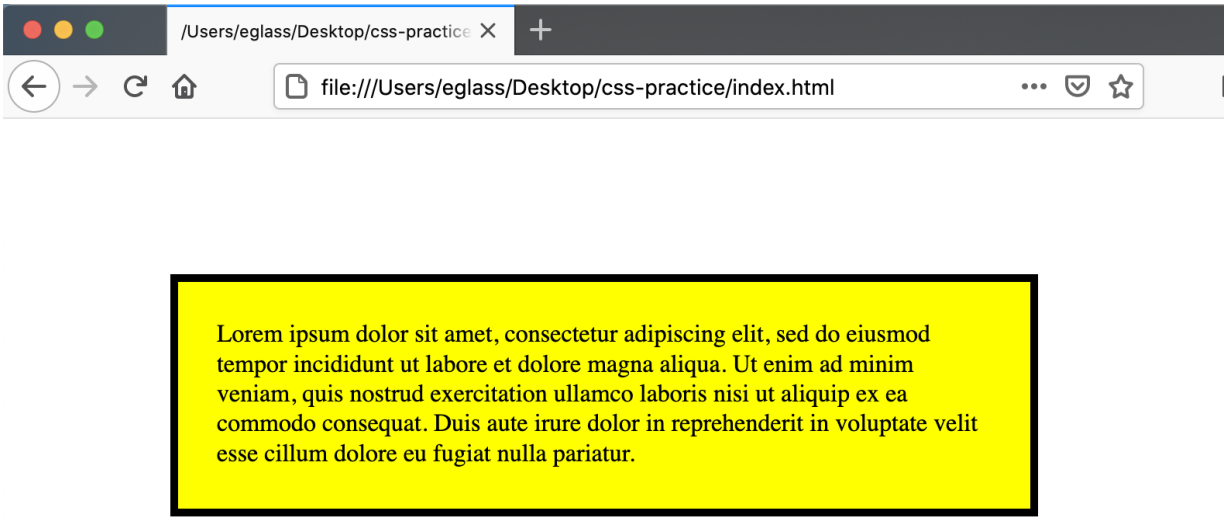
Adjusting the Margin Size of an HTML Element With CSS

Next, let's try adjusting the size of the margins of an element with CSS. In this exercise, we'll give the margins a very large value so that it is easy to see how margin size is displayed in the browser. Add the following highlighted declaration to your ruleset in your `styles.css` file to set the margin to 100 pixels:

styles.css

```
  .yellow-div {
  background-color:yellow;
  width: 500px;
  padding: 25px;
  border: 5px solid black;
  <^>margin:100px;<^>
}
```

Save the `styles.css` file and reload `index.html` in your browser to inspect the changes. The yellow box should have moved 100 pixels down and 100 pixels to the right to allow for the 100 pixels of margin space between its border and the edges of the viewport:



Webpage with `<div>` with padding, border, margins specified

Note: You may have noticed that the yellow box originally had a small margin of white space between its top and left side and the edges of the viewport. This margin is automatically created by some browsers to allow for space between the edges of the viewport and the website content. You can remove this margin by setting the top and left margin to zero.

Like the padding and border, the sizes of specific sides of the margin can be set using `margin-left`, `margin-right`, `margin-top`, and `margin-bottom`.

Before moving on, add another `<div>` container to the page to study how the margin affects the position of nearby content. Without erasing anything, add the additional CSS ruleset to your `styles.css` file:

styles.css

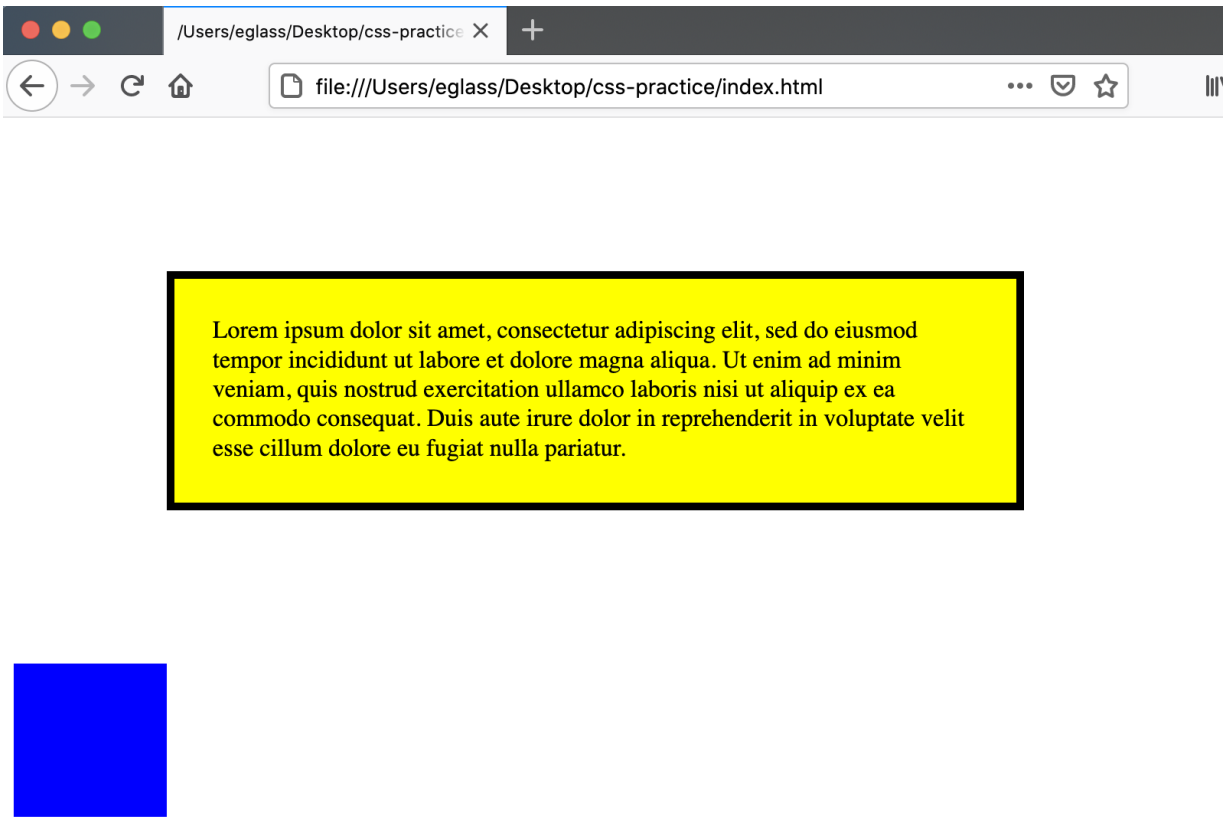
```
. . .
.blue-div {
  height:100px;
  width:100px;
  background-color: blue;
}
```

Save the file and return to your `index.html` file. Without erasing anything, add the following `<div>` element to your file and assign it the `blue-div` class:

index.html

```
. . .
<div class="blue-div"></div>
```

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



Two `<divs>` containers with margin space between them

The browser should now display a blue box that is 100 pixels wide and 1000 pixels high. This blue box should be 100 pixels below the yellow box on account of the yellow box's margin. In general, surrounding elements will by default be pushed away from an element on account of its margin. Be aware, however, that the margins of adjacent elements will often overlap due to margin collapse. The size of the overlapping margin is determined by the size of the largest margin between the two elements.

Conclusion

In this tutorial you learned about the CSS box model and how to adjust the size of each of its content, padding, border, and margin properties. Understanding the behavior of these properties and how to set values for them is useful when organizing and styling content on a webpage. This knowledge will be useful when building the demonstration website in the remaining tutorials. In the next tutorial, you will set up an `index.html` file to serve as the website's homepage.

How To Set Up Your CSS and HTML Website Project

Written by Erin Glass

Introduction

In this tutorial, you will set up the folders and files necessary for building a website with HTML and CSS. You will also prepare an `index.html` file so that it is ready to receive HTML content in the tutorials ahead.

Prerequisites

If you have been following along with this tutorial series, you can continue using the `css-practice` project directory, `index.html` file, `images` folder, `css` folder, and `styles.css` file that you created earlier in the series. If you have not been following along this tutorial series and need instructions for setting up these folders and files, please see our earlier tutorial in this series How To Set Up Your CSS and HTML Practice Project.

Note: If you decide to create your own names for the folders or files, make sure to avoid character spaces, special characters (such as !, #, %, or others), and capital letters, as these can cause problems later on. Be aware also that you will need to modify your file paths in some of the steps throughout the remainder of this tutorial series to ensure that they correspond with the names of your files.

You should have a project folder named `css-practice` that contains the following folders and files that are necessary to explore CSS in this

tutorial series:

- A folder named `css` that contains the file `styles.css`
- An empty folder named `images`
- A file named `index.html`

In the first step of this tutorial, you will prepare the `index.html` file so that it is ready to receive content in the tutorials ahead.

How To Prepare Your `index.html` File For HTML Content

To prepare your `index.html` file to serve as your website's homepage, we'll need to add a few important lines of HTML. These lines of HTML will serve as instructions for the browser and will not be displayed on the webpage itself. Make sure that your `index.html` file is empty (if you have content from previous tutorials) and add the following code snippet to the document:

index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
   <meta charset="utf-8">
   <title><^>Sammy the Shark<^></title>
   <link rel="stylesheet" href="css/styles.css">
  </head>
  <body>
  </body>
</html>
```

Make sure to change the highlighted site title with a title of your own choosing. Then save the `index.html` file. Before continuing, let's briefly review the code that you just added to understand its purpose:

- The `<!DOCTYPE html>` declaration tells the browser which type of HTML is being used in this document. It is important to declare this value as there are multiple versions of the HTML standard, and browsers need to know which to use. In this declaration, `html` specifies the current web standard of HTML, which is HTML5.
- The opening and closing `<html>` tags tell the browser that all content inserted between these two tags should be interpreted as HTML. Inside this tag, you also added the `lang` attribute, which specifies the language of the webpage. In this example, the language is set to

English using the `en` language tag. For a full list of language tags, visit the IANA Language Subtag Registry.

- The opening and closing `<head>` tags creates a section in the HTML document that typically contains information about the page, rather than page content itself. Browsers do not display the information in a `<head>` section.
- The `<meta charset="utf-8">` tag specifies the document's character set should be UTF-8, a unicode format that supports a majority of characters from a wide variety of written languages.
- The `<title>` tag tells the browser the name of the webpage. This title appears on the browser tab and when the site is listed in search results but it does not show up on the web page itself. Make sure to replace **"Sammy the Shark"** with your name or a title of your choosing if you want to personalize the site.
- The `<link rel="stylesheet" href="css/styles.css">` tells the browser where to find the stylesheet that contains the style rules. If you followed the instructions earlier in this series How To Set Up Your CSS and HTML Practice Project, your stylesheet should be located at this file path.
- The opening and closing `<body>` tags will contain the main content of the webpage. You'll add the HTML content between these tags in the tutorials ahead.

Conclusion

You have now created all of the folders and files necessary for creating a website with HTML and CSS. You should also have an `index.html` file

prepared with the necessary HTML content for serving as your website's homepage. In the next tutorial, you'll explore how the demonstration site is constructed and the steps you will take to recreate it.

An Overview of Our Demonstration HTML and CSS Website

Written by Erin Glass

In this tutorial, you will explore the structure of the demonstration website and a plan for recreating it in the tutorials ahead.

An Overview of the Demonstration Website

Visually, the site can be broken up into seven horizontal sections:

Header section

"About me" section

"Projects" section

"Experience" section

"Education" and "Skills" section

Featured quote section

Static footer section

Illustration of the demonstration website's sections

In the previous overview image, each of the seven sections is labeled accordingly:

- The "Header" section (at the top). Instructions for this section are detailed in our tutorial How To Build the Header Section of Your Website With CSS (Section 1)
- The "About me" section (second from the top). Instructions for this section are detailed in our tutorial [How To Build the About Me Section of Your Website With CSS (Section 2)] (https://www.digitalocean.com/community/tutorials/how-to-build-the-about-me-section-of-your-website-with-css-section-2)
- The "Projects" section (third from the top). Instructions for this section are detailed in our tutorial How To Build a Tiled Layout With CSS (Section 3)
- The "Experience" section (fourth from the top). Instructions for this section are detailed in our tutorial [How To Add a Resume or Work History Section To Your Website With CSS (Section 4)]. (https://www.digitalocean.com/community/tutorials/how-to-add-a-resume-or-work-history-section-to-your-website-with-css-section-4)
- The "Education" and "Skills" section (fifth from the top). Instructions for this section are detailed in our tutorial How To Add Your Educational History and Skills To Your Website With CSS (Section 5)
- The featured quote section (sixth from the top). Instructions for this section are detailed in our tutorial How To Create a Featured Quote

Box on Your Website With CSS (Section 6)

- The static footer, which “sticks” to the bottom of the page. Instructions for this section are detailed in our tutorial How To Create a Static Footer With CSS (Section 7)

Each of these sections are created with the CSS properties for HTML elements that you explored in the first half of the tutorial series. In the remainder of this tutorial series, you will reconstruct each of these sections in their own separate tutorial. If you are just beginning to learn CSS, we recommend that you replicate the style choices in the tutorials including size, color, and background images to keep things consistent with the examples as you work through each tutorial.

At the end of this tutorial series there are suggestions for experimenting with the style and layout of your website. These suggestions will demonstrate how to personalize the content and remix these tutorials to create new style and arrangement possibilities for your site.

Conclusion

In this tutorial, you explored the structure of the demonstration website and an overview of the plan for recreating it. In the next tutorial, you’ll create a CSS rule to style the entire body of the webpage and learn why this rule is an important first step.

How To Style the Body of a Website With CSS

Written by Erin Glass

In this tutorial, you will style the body of a webpage with a CSS rule. You will use this rule to apply and style a background image and set the font family for the webpage. You will also create a style rule that changes the color of all hyperlinked text to a color that better matches the demonstration website's color palette.

This exercise will be used to recreate the style of the demonstration site but you can apply and modify the same rules used here for other HTML/CSS website projects.

Prerequisites

To follow this tutorial, make sure you have set up the necessary files and folders as instructed in a previous tutorial in this series How To Set Up You CSS and HTML Practice Project.

For this tutorial, we suggest you use the background image from the demonstration site which you can download from this link. You may use another image as your background, but make that sure that the image is large enough to fill the screen.

Note: To download the background image of the demonstration site, visit this link and click `CTRL + Left Click` (on Macs) or `Right Click` (on Windows) on the image and select "Save Image As" and save it as `background-image.jpeg` to your "image' folder.

Once you have selected an image, make sure it's saved as "background-image.jpeg" in your `images` folder. You are now ready to proceed to the next step.

Adding a Background Image To Your Website With CSS

To declare style rules for the body of a webpage, you will need to create a CSS rule for the `body` tag selector. These rules will then be applied to all elements that are placed inside the opening and closing `<html>` tags that you added to the `index.html` file in the earlier tutorial How To Set Up Your CSS and HTML Website Project.

To add a background image to your site, create a CSS rule using the `<body>` tag selector. Erase everything in your `styles.css` file (if you have been following along with this series) and add the following ruleset:

styles.css

```
/* General Website Style rules */
body {
  font-family: "Helvetica", Sans-Serif;
  background-image: url("../images/background-image.jpeg");
}
```

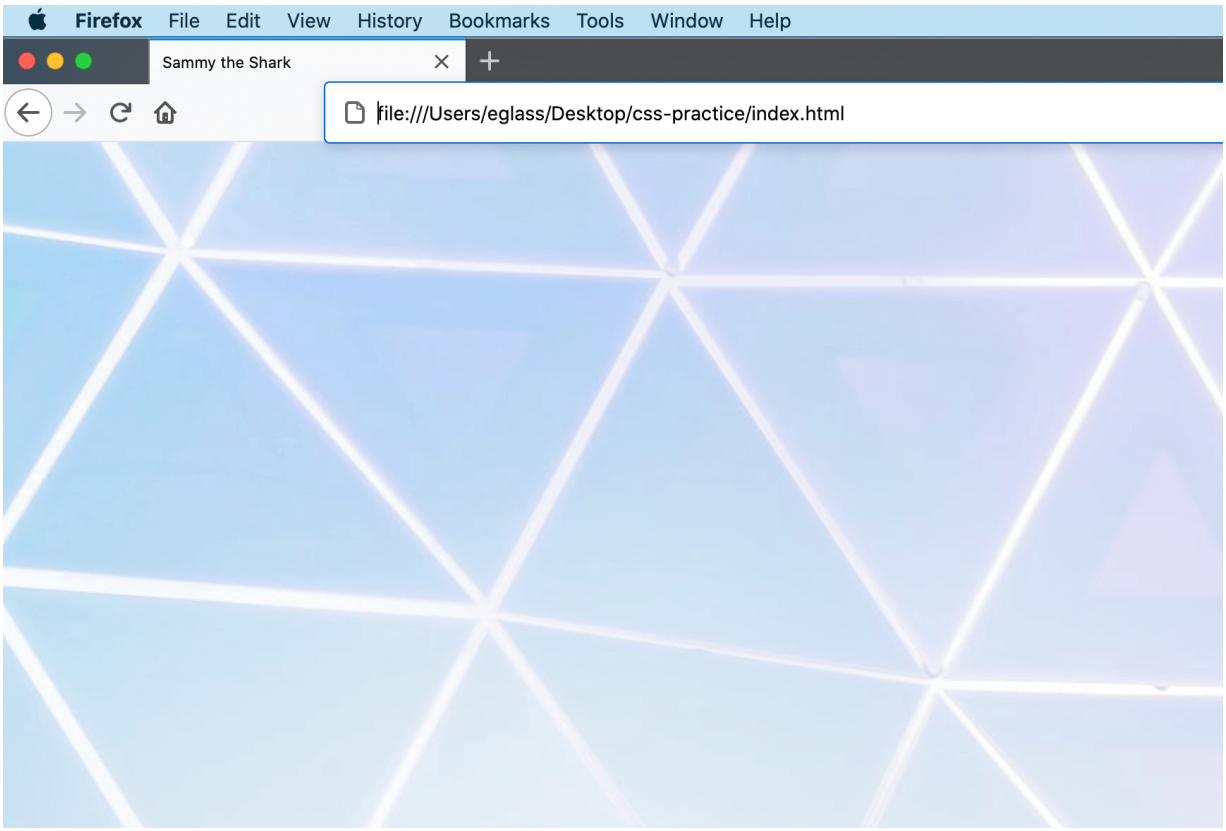
Take note of the highlighted file path, which tells the browser where to locate the background image. If you have changed the name or location of the image then you will need to adjust the file path here accordingly.

Let's pause briefly to understand each of the declarations in this ruleset:

- `/* General Website Style rules */` is a CSS comment, which is not displayed by the browser. Like HTML comments, CSS comments are useful for explaining and organizing your code for future reference. Notice that CSS comments open and close with `/*` and `*/` tags instead of `<!--` and `-->` tags used for HTML comments.
- The `font-family: "Helvetica", Sans-Serif;` declaration sets the font family (Helvetica) and generic font family (Sans-Serif) for all the text on the webpage. (Note that you can specify different font families for text content on the same webpage by adding CSS rules later on). The generic font family is given as a backup in case the first font family isn't available and the browser needs to pick a back up font. You can explore other fonts by replacing "Helvetica" with other font names, such as `Times`, `Courier`, or `Palatino`.
- The `background-image: url("../images/background-image.jpeg;")` declaration tells the browser to add a background image to the webpage using the file found with the specified file path. Note that you have prepended `../` to the file path name to tell the browser to locate the `images` folder in the directory above the directory that contains the file you are working in (`styles.css`).

Save your `styles.css` file and load the `index.html` page in your browser. For instructions on loading an HTML file, please visit our tutorial step How To View An Offline HTML File In Your Browser.

You should receive a page with no content except for the background image:



Webpage with background image only

If you don't receive an image, check to make sure your file path is correct and that there are no errors in your `index.html` file and `styles.css` file.

Changing the Color of Hyperlinked Text

Next, we'll add a CSS rule that changes the color of all hyperlinked text to a color that better matches the website color palette.

At the bottom of your `styles.css` file, add the following ruleset:

styles.css

```
a {
  color: #112d4e;
}
```

This ruleset will style any text marked up with an `<a>` tag with the HTML color code `#112d4e`. The style will not be apparent until you add `<a>` elements to your `index.html` page (which you will do in the last tutorial How To Create a Static Footer With HTML and CSS. You can change the style color by changing the HTML color code in this CSS rule.

Conclusion

You should now have a webpage with a large background image. In addition, you declared a font family that will be applied when you begin to add text content. Using rulesets like these allow you to change the font and background image of a webpage by creating a ruleset for the `body` tag selector. Finally, you created a style rule that specifies the color of any hyperlinked text you add to the page.

In the next tutorial, you'll recreate the header section of the demonstration website.

How To Build the Header Section of Your Website With CSS (Section 1)

Written by Erin Glass

In this tutorial, you will recreate the top header section of the demonstration website using HTML and CSS. You can switch out Sammy's information with your own if you wish to experiment or personalize the size. The methods that you use here can be applied to other CSS/HTML website projects.

Screenshot of header section of demonstration website

Prerequisites

To follow this tutorial, make sure you have set up the necessary files and folders as instructed in a previous tutorial in this series How To Set Up You CSS and HTML Practice Project.

Adding the Title and Subtitle To Your Webpage Header

Our website header includes the title (“Sammy the Shark”), a subtitle (“SENIOR SELACHIMORPHA AT DIGITALOCEAN”), and a small profile image. These elements are wrapped inside a `<div>` container that is styled with a class defined in the CSS stylesheet. You will recreate this section by adding the text and image content, creating a class for the `<div>` container, and then wrapping the text and image content in a `<div>` container that is assigned the newly-created class.

To add a title and subtitle to your site, add the following highlighted code snippet in between the opening and closing `<body>` tags in the `index.html` file. Switch out Sammy’s information with your own if you would like to personalize your site:

index.html

```
. . .

<body>

<^><!--Header content--><^>
    <^><h1>Sammy the Shark<h1><^>
    <^><h5>SENIOR SELACHIMORPHA AT DIGITALOCEAN<h5>
<^>
</body>
```

In this code snippet, you have added the title `Sammy the Shark` and assigned it the `<h1>` heading tag as it is the most important heading of this

webpage. You have also added the subtitle `SENIOR SELACHIMORPHA AT DIGITALOCEAN` and assigned it the `<h5>` heading tag, as it is a less important heading.

Note that you have also added the comment `<!--Header content-->` just before the title. A comment is used to save explanatory notes on your code for future reference and is not displayed by the browser to site visitors (unless they view the source code of the webpage). In HTML, comments are written between `<!--` and `-->` as demonstrated in the code snippet above. Make sure to close your comment with the ending comment tag (`-->`) or all of your content will be commented out.

Adding and Styling a Small Profile Image To Your Webpage Header

Next, you'll add a small profile image to the header section. Pick a profile photo that you want to include on your site. If you don't have a profile photo, you can use any alternative image (such as the profile image of Sammy) or create an avatar through a site like Getavataaars.com.

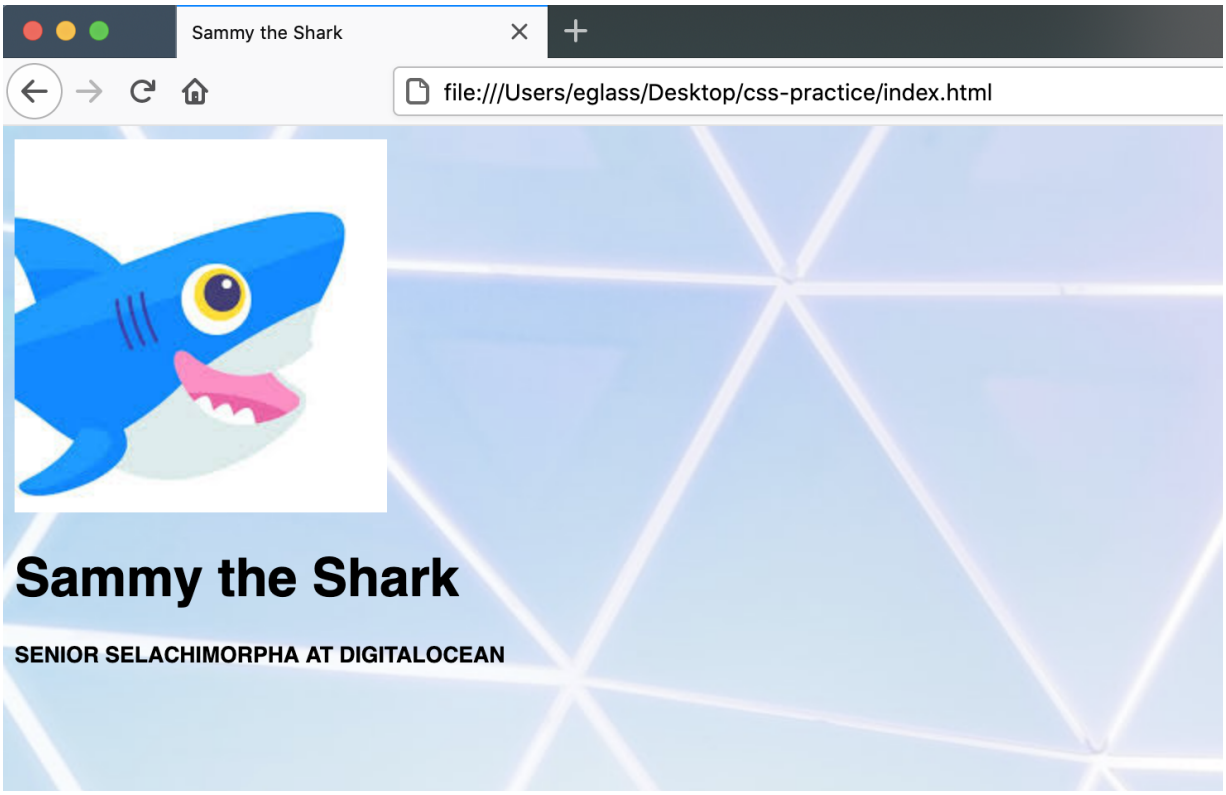
Once you have selected an image, save it to your `images` folder as `small-profile.jpeg`.

Now add the profile image to the webpage by using an `<img>` tag and the `src` attribute assigned the file path of your profile image. Add the following highlighted code snippet to your `index.html` file just after the `<!--Header content-->` line and before the `<h1>Sammy the Shark<h1>` line:

index.html

```
. . .

  <body>

  <!--Header content-->
        <^><img src="images/small-profile.jpeg"
alt="Sammy the Shark, DigitalOcean's mascot"><^>
        <h1>Sammy the Shark<h1>
        <h5>SENIOR SELACHIMORPHA AT
DIGITALOCEAN<h5>
   </body>
</html>
```

Save the file and load it in the browser. Your webpage should now have a title, subtitle, profile image, and background image:



Webpage with profile image, title, and subtitle

Notice that the image does not have the same styling as the profile image in the demonstration site. To recreate the shape, size, and border of the profile image in the demonstration site, add the following ruleset to your `styles.css` file:

styles.css

```
. . .
/*Top header profile image*/
.profile-small {
   height:150px;
   border-radius: 50%;
   border: 10px solid #FEDE00;
}
```

Before moving on, let's review each line of code you just added:

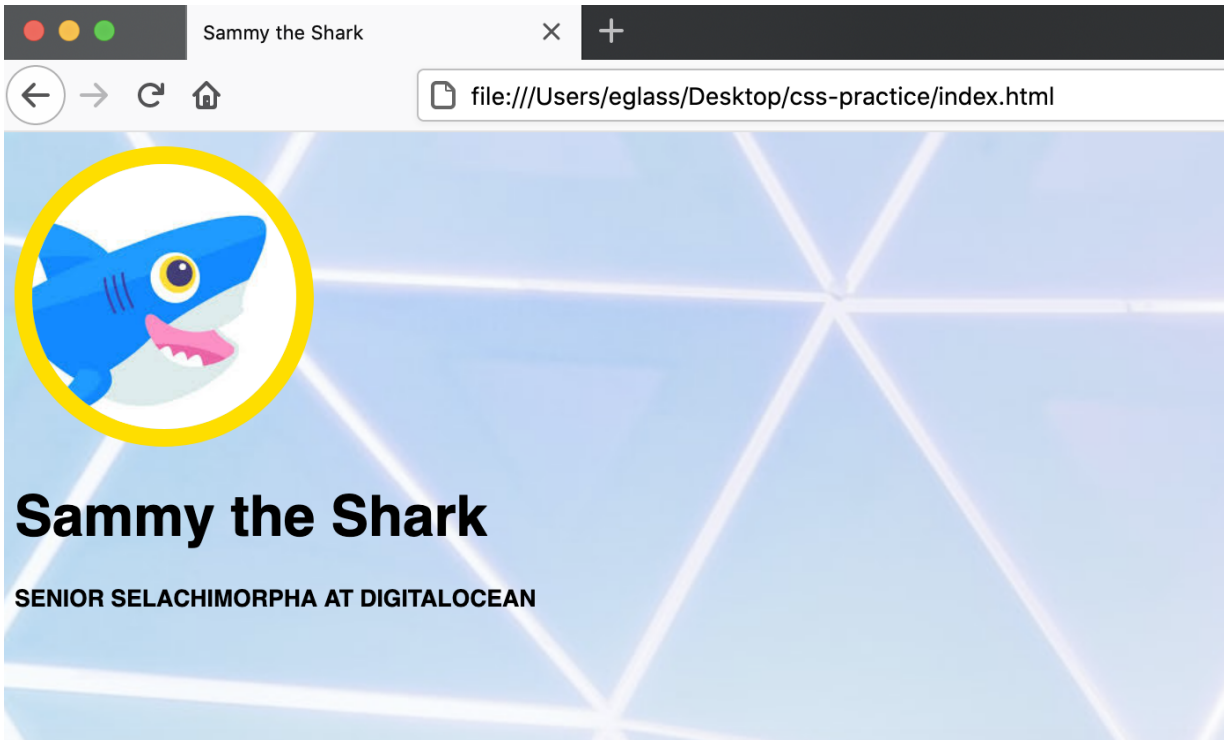
- `/*Top header profile image*/` is a CSS comment for labeling the code.
- The text `.profile-small` refers to the name of the class we're defining with the ruleset. This class will be applied to the profile image in the next step.
- The declaration `height:150px;` sets the height of the image to 150 pixels and automatically adjusts the width to maintain the image size proportions.
- The declaration `border-radius: 50%;` rounds the edges of the image into a circular shape.
- The declaration `border: 10px solid #FEDE00;` gives the image a solid border that is 10 pixels wide and has the HTML color code `#FEDE00`.

Save the file and return to your `index.html` file to add the `profile-small` class to your `<img>` tag like so:

index.html

```
. . .
        <img src="images/small-profile.jpeg"
<^>class="profile-small"<^> alt="Sammy the Shark,
DigitalOcean's mascot">
. . .
```

Save the file and reload it in your browser. Your profile image should now have a height of 150 pixels, a circular shape, and a yellow border:



Header with styled profile image

In the next step, you'll apply additional styling to the title, subtitle, and profile image as a whole.

Styling and Positioning the Header Content With CSS

You will now define a class with CSS to style and position the header content. Return to the `styles.css` file and create the `header` class by adding the following CSS ruleset:

styles.css

```
. . .
/* Header Title */
.header {
  padding: 40px;
  text-align: center;
  background: #f9f7f7;
  margin:30px;
  font-size:20px;
}
```

Let's pause briefly to understand each line of the code that you just added:

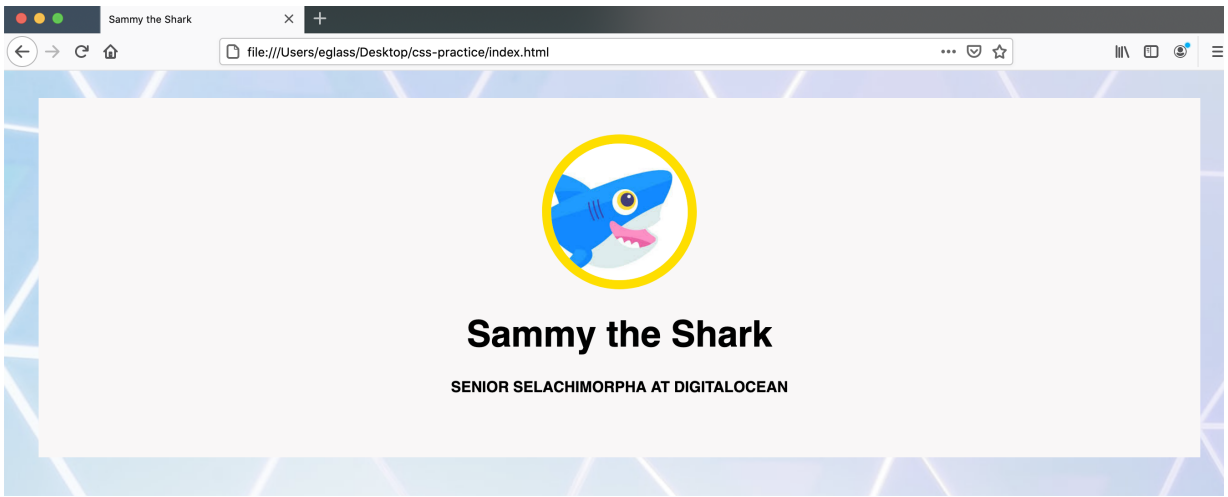
- The `/* Header Title */` is a comment, which is not displayed by the browser.
- The text `.header` is the name of the class selector we're creating and defining with this ruleset.

- The `padding: 40px;` declaration creates 40 pixels of padding between the content and the border of the element.
- The `text-align: center;` declaration moves the content to the center of the element. You can also adjust the value to `left` or `right` to align the text accordingly.
- The `background: #f9f7f7;` declaration sets the color to the specific HTML color code used in the demonstration website. This tutorial will not cover HTML color codes in this tutorial series, but you can also use HTML color names (`black`, `white`, `gray`, `silver`, `purple`, `red`, `fuchsia`, `lime`, `olive`, `green`, `yellow`, `teal`, `navy`, `blue`, `maroon`, and `aqua`) to change the color value of this property.
- The `margin:30px;` declaration creates a margin of 30 pixels between the perimeter of the element and the perimeter of the viewport or any surrounding elements.
- The `font-size:20px;` declaration increases the size of both the title and subtitle.

Save your `styles.css` file. Next, you will apply this `header` class to your header content. Return to the `index.html` page and wrap the header content (that you already added to your file) in a `<div>` tag that is assigned the `header` class:

```
. . .
<!--Section 1: Header content-->
    <^><div class="header"><^>
      <img src="images/small-profile.jpeg"
class="small-profile.jpeg" alt="Sammy the Shark,
DigitalOcean's mascot">
      <h1>Sammy the Shark<h1>
      <h5>SENIOR SELACHIMORPHA AT DIGITALOCEAN<h5>
    <^></div><^>
  </body>
</html>
```

Save the `index.html` file and reload it in your browser. Your title, subtitle, and profile image should now be styled inside a `<div>` container according to the rules you declared with the `header` class:



Header content now centered and styled

Conclusion

You have now recreated the header section of the demonstration website on your webpage using HTML and CSS. You added and styled a title, subtitle, and profile image using `<div>` containers and CSS classes. If you are interested, you can continue to explore design possibilities by modifying your CSS rules for your header content.

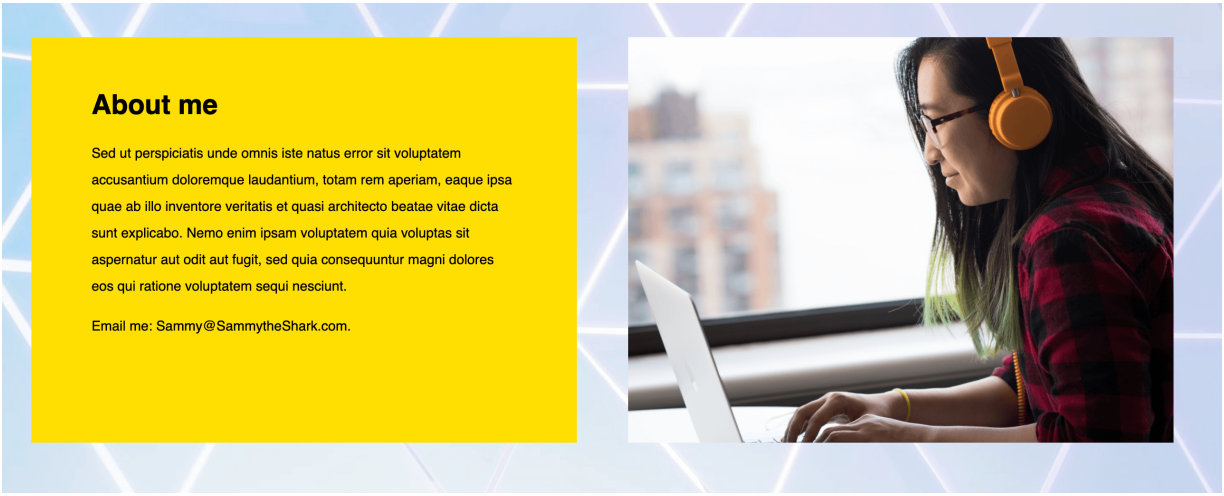
When you are ready, you can continue to the next tutorial where you will recreate the second section of the demonstration site.

How To Build the About Me Section of Your Website With CSS (Section 2)

Written by Erin Glass

In this tutorial, you will recreate the second section of the demonstration website using CSS. Feel free to switch out Sammy's information with your own if you wish to personalize the size. The methods you learn here can be applied to other CSS/HTML website projects.

The second section of the site contains two content boxes, one that contains text and one that contains a large profile photo:



Screenshot of the second section of the website

Prerequisites

To follow this tutorial, make sure that you have set up the necessary files and folders as instructed in a previous tutorial in this series How To Set Up You CSS and HTML Practice Project.

You will need a profile image to place in the content box on the right. If you don't have a profile image, you can use this image for demonstration purposes.

Note: To download the large profile image, visit this link and click `CTRL + Left Click` (on Macs) or `Right Click` (on Windows) on the image and select "Save Image As" and save it as `large-profile.jpeg` to your `images` folder.

Before proceeding, make sure your selected image is saved in your `images` folder as `large-profile.jpeg`.

Creating Style Rules For Text and Image Content Boxes

To create these two content boxes, you will first define a column class in the `styles.css` file that styles the boxes for this purpose. Then you will add the text and image content to the HTML document.

Return to the `styles.css` file and copy and paste the following rulesets at the bottom of the file:

styles.css

```
. . .
/* Include padding and border in total box size */
* {
  box-sizing: border-box;
}
/* Create two equal columns that float next to
each other */
.column-2 {
  float: left;
  width: 45%;
  padding: 40px;
  padding-left:70px;
  padding-right: 70px;
  height: 475px;
  margin:30px;
  margin-right:30px;
  margin-bottom: 70px;
  background-color: #FEDE00;
  line-height:2;
}
```

Before moving on, let's pause to understand each of the rulesets we've just added.

The first ruleset uses the "`*`" selector to indicate that the ruleset should be applied to all HTML elements and classes. This ruleset declares the `box-sizing` property's value as `border-box`, which adjusts the total calculated width and height of a CSS element to include its padding and border size. By default, width and height sizes of an element refer only to the content of an element. Setting the `box-sizing` property to `border-box` makes it easier to adjust the total width and height of an element and can be helpful when laying out content on a page. To read more about the CSS box model, please visit our tutorial How To Adjust the Content, Padding, Border, and Margins of an HTML Element With CSS.

The second ruleset defines a class named "column-2" with sizing specifications that allow for two columns to be displayed side by side on the page. This class is named `column-2` to differentiate it from columns with other sizes that you will create classes for later on in the tutorial.

Some of the values and properties in this ruleset have not yet been covered in this tutorial series:

- The `float:left;` declaration instructs the element to float to the left side of the container it's inside (in this case the viewport itself) while allowing surrounding content to flow around its right side. You can also set the `float` property value to `right` or `none`, though this tutorial uses the `left` value to recreate the demonstration website.
- The `width: 45%;` declaration sets the element's width to 45% of the width of its container, which in this case is the viewport itself. Setting sizes (such as width) in percentages instead of pixels can be useful when you want the element to resize according to the size of the

container in which it's situated. Note, however, that dynamic sizing can be a tricky process—there are multiple methods for creating responsive elements which can be implemented after establishing a foundation in CSS.

- The `background-color: #FEDE00;` sets the element's background color to the HTML color code "#FEDE00".
- The `line-height:2;` increases the spacing between lines.
- If you want to learn more about the other declarations, please review the previous sections in this tutorial series on setting the sizes of content, padding, and margins.

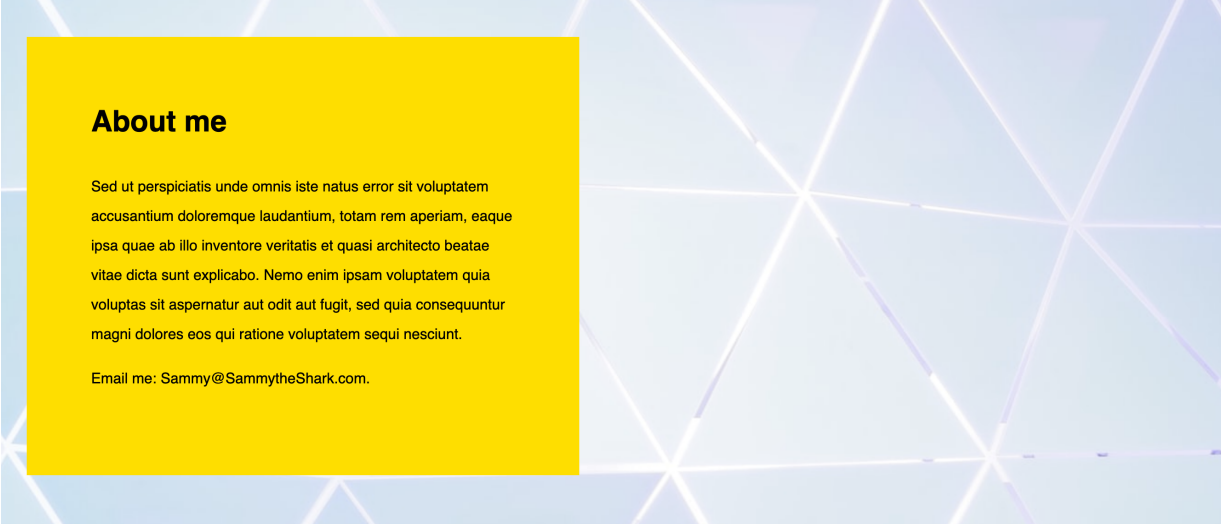
Adding the "About me" Content Box

Next, you will add the "About me" content box to the webpage using the `column-2` class that you just created. Save your `styles.css` file and return to the `index.html` file. Add the following code after the closing `</div>` tag in your header section, before the closing `</body>` tag :

```
. . .
<!--Section 2: About me-->

        <div class="column-2">
            <h1>About me</h1>
            <p>Hi! I'm Sammy the Shark, Senior Selachimorpha at DigitalOcean by day, dabbler in all things dev by night. This site is a demonstration website for the tutorial series "<a href="https://www.digitalocean.com/community/tutorial_series/how-to-build-a-website-with-css">How To Build a Website With CSS</a>," which walks you through building and customizing this website from start to finish.</p>
            <p>If you're following this tutorial series, you can replace this text with your own "About Me" content.</p>
        </div>

. . .
```

Save the file and load it in the browser. For instructions on loading an HTML file, please visit our tutorial step How To View An Offline HTML File In Your Browser.

You should now have a yellow box on the left side of the webpage that contains text:



Webpage with yellow div box with un-styled text

Note that your webpage should still contain the header content you added in the previous tutorial of this series How To Build the Header Section of Your Website With CSS.

Let's briefly review how this HTML code is functioning:

- The first line in this code snippet (`<!--Section 2: About me-->`) is a comment that helps organize the HTML content. It will not display in the browser and is included here for reference later.
- The next line of code (`<div class="column-2" style="background-color:#FEDE00;">`) creates a `<div>` container, assigns it the style of the `column-2` class you defined in the `styles.css` file, and uses the HTML inline `style` attribute to assign it the background color `#FEDE00`.
- The `<h1>` and `<p>` tags that follow contain the text you are inserting into the "About me" text box. Notice that you have closed the `<div>`

container at the end of this text. You can switch out Sammy's text with your own text if you plan on personalizing your website.

Adding the Profile Image Content Box

Next, you will add the second content box that contains the large profile image. There are a number of ways you can add an image box, but in this tutorial you'll add the large profile image by making it the background image of another `<div>` container that is assigned the `column-2` class.

Return to the `styles.css` file and add the following code snippet to the bottom of the document:

styles.css

```
. . .
/* Large profile image */
.large-profile {
  background: url('../images/large-profile.jpeg');
  background-size: cover;
  background-position: center;
}
```

In this code snippet you have added a comment to organize the CSS rules and created and defined the new class `large-profile` that you'll use to style the box that holds the large profile image. In this ruleset, the `background: url('images/large-profile.jpeg');` declaration tells the browser to use the image found at the specified file path

as the background image of the element. The `background-size: cover` declaration fits the image to cover the space of the container in which it is situated, the `background-position:center` declaration centers the image inside the container.

Next you will add a `<div>` container that is assigned both the `column-2` class and the `large-profile` class to recreate the box with the large profile image.

Save your `styles.css` file and return to the `index.html` file. Add the following code snippet below the closing `</div>` tag of your first column and above the closing `</body>` tag:

```
. . .
  <div class="column-2 large-profile">
  </div>
```

This code snippet creates a `<div>` container styled according to the rules declared in the `column-2` class and the `profile-picture` class.

Save both files and reload `index.html` in your browser. Your webpage should now have the text box and image box as styled in the demonstration website (and pictured in the first image of this tutorial). Note that your webpage should also still include the header content you created in the previous tutorial. You can continue experimenting with the declared values in the `column-2` and `profile-large` classes to explore different design possibilities.

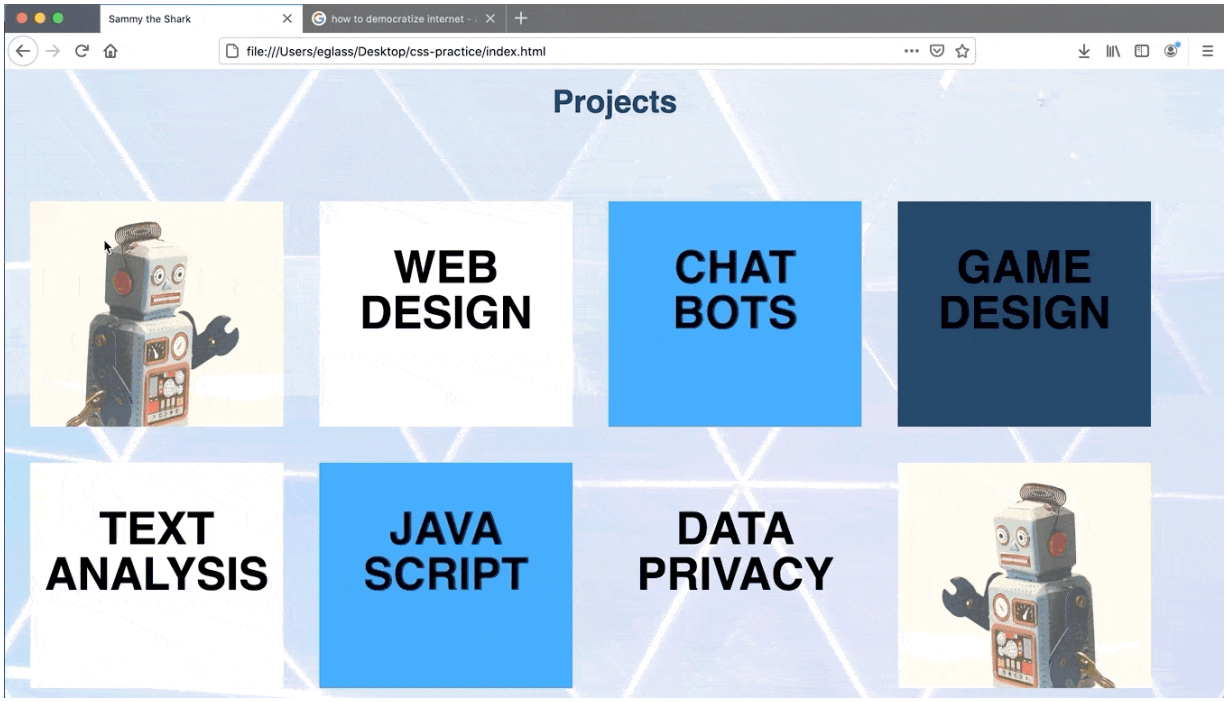
Conclusion

You have now created and styled content boxes for text and images using CSS. In the next tutorial, you will recreate the third section of the website. In the process, you will arrange content into two rows of four boxes and apply a pseudo-class that will cause the boxes to change color when the user hovers over them with their cursor.

How To Build a Tiled Layout With CSS (Section 3)

Written by Erin Glass

In this tutorial, you will recreate the tiled layout of the "Projects" section of the demonstration website by styling eight HTML `<div>` containers with CSS classes. You will also add the `hover` pseudo-class to these elements so that they change color when a user hovers over them. Feel free to switch out Sammy's information with your own if you wish to personalize the size. The methods you use here can be applied to other CSS/HTML website projects.



Gif of "projects" section of the demonstration website

Prerequisites

To follow this tutorial, make sure you have set up the necessary files and folders as instructed in a previous tutorial in this series How To Set Up You CSS and HTML Practice Project.

Adding and Styling a Header Title

To get started, add the heading title “Projects” to a new section on the webpage, making sure not to delete any content you’ve added from the previous tutorials. Add the following code snippet after the last closing `</div>` tag in the `index.html` file:

index.html

```
. . .
<!--Section 3: Projects-->

  <h2 >Projects</h2>
. . .
```

The first line of this code snippet is a comment to label the code you will add to create the third section of the website. A comment is used to save explanatory notes on your code for future reference and is not displayed by the browser to site visitors (unless they view the source code of the webpage). The second line adds the title text “Projects” and assigns it the heading element `<h2>`.

Next, you will style the heading title by creating a `section-heading` class. Return to the `styles.css` file and copy and paste the following code snippet at the bottom of the file:

styles.css

```
. . .
/*  Section 3  */

.section-heading {
  text-align:center;
  color:#102C4E;
  margin-top: 150px;
  margin-bottom:70px;
  font-size: 35px;
}
```

This code snippet defines the style for the `section-heading` class. Please review the previous sections in this tutorial series on setting the sizes of content, padding, and margins.

Next you will add the "section-heading" class to the header title "Projects" in the HTML file. Return to the `index.html` file and add the class to the HTML element like so:

index.html

```
<!--Section 3: Projects—>

   <h2 class="section-heading">Projects</h2>
```

Save both files and load your web page in the browser. For instructions on loading an HTML file, please visit our tutorial step How To View An Offline HTML File In Your Browser. The header should now be centered on the page and its size, positioning, and color should be adjusted like in the following image:

Styled project heading on webpage

Adding and Styling Tiled Project Boxes

Now you will add the eight project boxes below the section header. To get started, you'll create a CSS class that allows us to style `<div>` containers in a way that will allow four of them to fit side by side on the webpage.

Return to the `styles.css` file and add the following code snippet at the bottom of the document:

styles.css

```
. . .
/* Sizing for Project Containers */
.column-4 {
  float: left;
  width: 21%;
  padding: 10px;
  margin:20px;
  height: 250px;
}
```

In this code snippet you have defined the class `column-4` and specified values that allow for four columns to be displayed side by side on the page:

- The `float: left;` declaration instructs the element to float to the left side of the container it's inside (in this case the webpage) while allowing surrounding content (in this case the other project boxes) to rest on its right side.
- The `width: 21%;` declaration sets the element's width to 21% of the width of its container, which in this case is the webpage. Setting sizes (such as width) in percentages instead of pixels can be useful when you want the element to resize according to the size of the container it's inside. Note, however, that dynamic sizing can be a tricky process—there are multiple methods for creating responsive elements which can be implemented after establishing a foundation in CSS.

- If you want to learn more about the other declarations, please review the previous sections in this tutorial series on setting the sizes of content, padding, and margins.

Next, you will create a class for each of the eight boxes so that you can style them differently, as well as add a featured image to the first and last box to match the demonstration site. To get started, save one or two images to use as a featured image in your images folder. If you don't have an image, you can download the image of a right-facing robot and left-facing robot that you used in the demonstration site.

Note: To download the robot images, visit links and click `CTRL + Left Click` (on Macs) or `Right Click` (on Windows) on the image and select "Save Image As" and save it as `project-left.jpeg` and 'project-right.jpeg' to your images folder.

To create a class for each project box, add the following code snippet to the bottom of your `styles.css` file:

styles.css . . .

```
/* Color and Images for Project Containers  */

.project-1 {
  background: url('../images/project-left.jpeg');
  background-size: cover;
}

.project-2 {
  background-color:white;
}

.project-3 {
  background-color:#209BFF;
}

.project-4 {
  background-color:#112d4e;
}

.project-5 {
  background-color:#F9F7F7;
}

.project-6 {
  background-color:#209BFF;
```

```
}

.project-7 {
    background-color:#ffffff00;
}

.project-8 {
  background: url('../images/project-right.jpeg');
  background-size: cover;
}
```

If you are using your own images, make sure you have saved them to your images folder and that you have specified the correct file path in the highlighted area in the ruleset for class `project-1` and class `project-8`.

Let's pause briefly to review the code we've just written. In the rulesets for class `project-1` and `project-8`, you have added a background image, specified its file path location and declared that the background image should be fitted to "cover" the entire element.

In the rulesets for `project-2` through `project-7`, you have specified different background colors using HTML color codes. Note that you have made the background color transparent for `project-7` as a design choice, but you can change this as you wish. You can also explore different background images and colors for each of these classes by experimenting with their values.

Next you will add a ruleset that changes the font size and positioning of the text that will be added to these project boxes. Add the following ruleset to the bottom of the `styles.css` file:

styles.css

```
. . .
.project-text {
  text-align:center;
  font-size:50px;
}
```

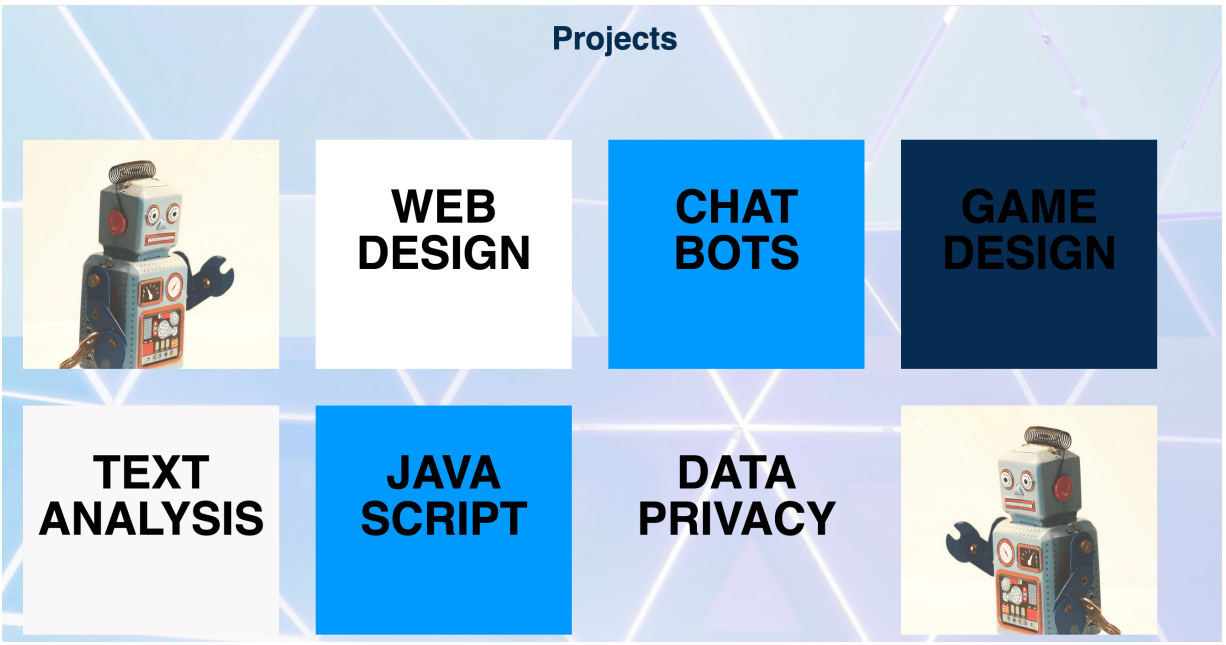
Save your `styles.css` file. Now you will add `<div>` containers to the HTML document and style them with the CSS classes you just defined. Return to the `index.html` file and add the following code snippet below this line: `<h2 class="section-heading">Projects</h2>`:

index.html

```
. . .
<div class="column-4 project-1">
</div>

<div class="column-4 project-2">
  <h2 class="project-text">WEB <br>DESIGN</h2>
</div>

<div class="column-4 project-3">
  <h2 class="project-text">CHAT <br>BOTS</h2>
</div>

<div class="column-4 project-4">
  <h2 class="project-text">GAME <br> DESIGN</h2>
</div>

<div class="column-4 project-5">
  <h2 class="project-text">TEXT <br> ANALYSIS</h2>
</div>

<div class="column-4 project-6">
  <h2 class="project-text">JAVA <br> SCRIPT</h2>
</div>

<div class="column-4 project-7">
  <h2 class="project-text">DATA <br> PRIVACY</h2>
</div>
```

```
<div class="column-4 project-8">
</div>
```

Save your `index.html` file and reload it in the browser. You should receive output like the following image:



Styled project containers on webpage

You should have two rows of four boxes, each styled according to the `column-4` and `project-x` classes they've been assigned with the class attribute. In the HTML code, you have also added text content (such as "CHAT BOTS") and assigned all text content the class `project-text`.

You have also added the HTML line break element (`<br>`) to create a line break between the two words in each box. Feel free to change this text now or later on if you wish to personalize your website. You can also use

the HTML hyperlink element `<a>` to link this text to new pages you create for your website. You'll explore this option in more detail at the end of the tutorial series.

Next, you will add a pseudo-class to make the boxes change color when the user hovers their cursor over them.

Changing Content Color With User Interaction

If you return to the demonstration website and hover your cursor over the boxes in the "Projects" section, you'll notice that they change color. This color change is achieved by adding the `hover` pseudo-class to each of project classes.

As you may recall if you followed the tutorial on pseudo-classes earlier in this series, pseudo-classes are created by appending a colon and the pseudo-class name to the name of the class you are trying to modify. To add the `:hover` pseudo-class to the project classes, add the following rulesets at the bottom of your `styles.css` file:

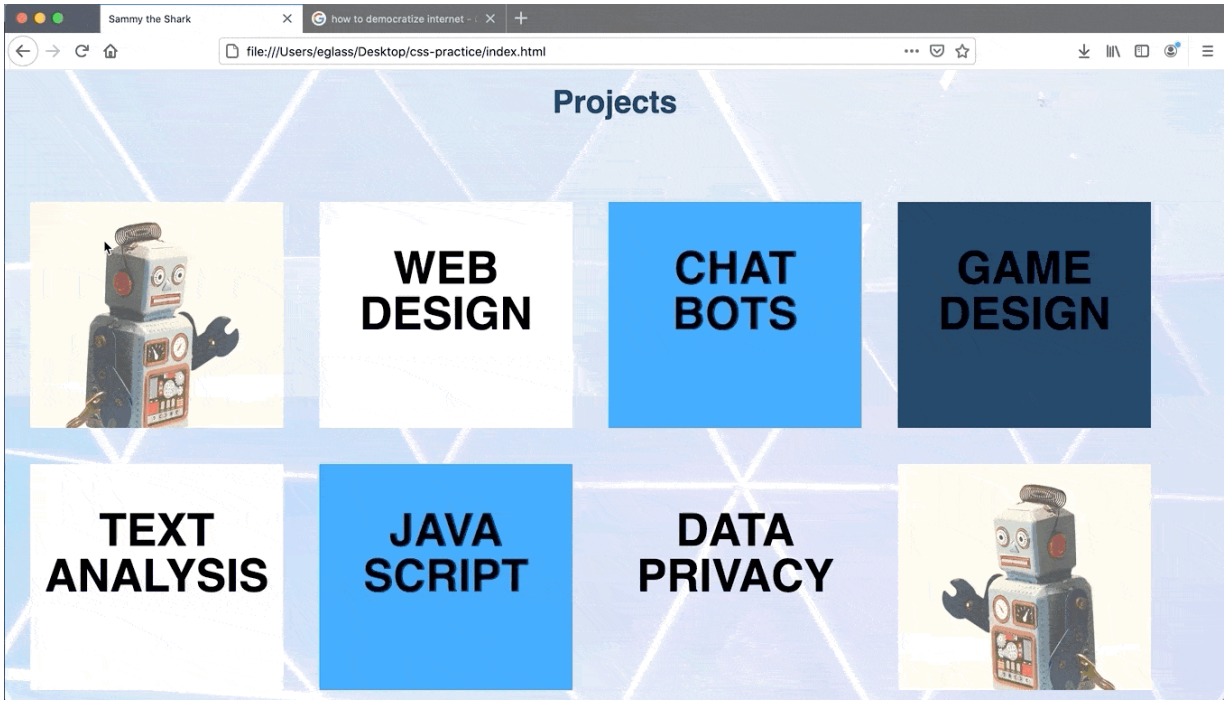
styles.css .project-2:hover { background-color:#FEDE00; }

```
.project-3:hover {
  background-color: #FEDE00;
}

.project-4:hover {
  background-color: #FEDE00;
}

.project-5:hover {
  background-color: #FEDE00;
}

.project-6:hover {
  background-color: #FEDE00;
}

.project-7:hover {
  background-color: #FEDE00;
}
```

In this code snippet you created `hover` classes for six of the eight project classes. This `hover` class instructs the element to change its color to the HTML color code `#FEDE00` when the user hovers the cursor over the box. Note that you have only added the hover class to the project boxes

that contain text (and not to the project boxes that contain background images).

Save the `styles.css` file and reload `index.html` in the browser. Check to make sure that the `hover` pseudo-class is working by hovering your cursor over the project boxes. They should change color when your cursor passes over them:



Gif demonstrating hover style on project boxes

Conclusion

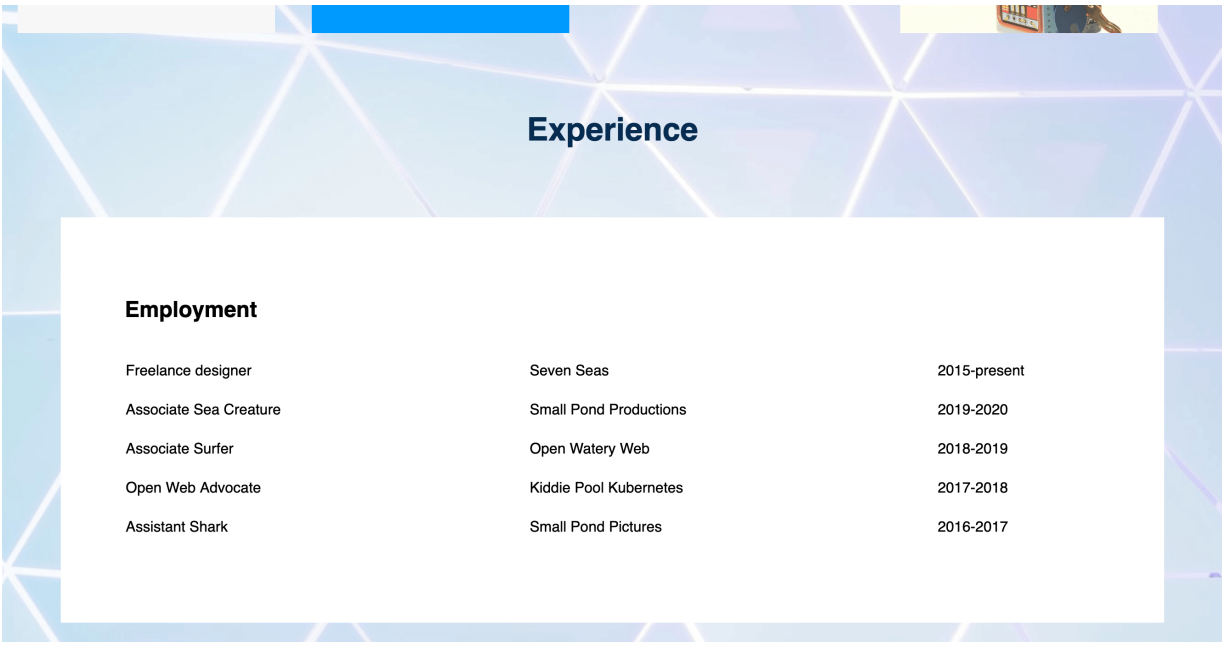
You have now laid out HTML content in boxes using CSS classes and added a `hover` pseudo-class to change their appearance when a user's cursor hovers over them. You can continue to experiment with these methods by changing the style declarations in these classes or changing the size and quantity of boxes you use to organize the layout of your page.

In the next tutorial, you will add an “Employment” section to a website using HTML tables.

How To Add a Resume or Employment History Section To Your Website With CSS (Section 4)

Written by Erin Glass

In this tutorial, you will recreate the “Employment” section of the demonstration website (or fourth section) using HTML tables and CSS classes. Feel free to switch out Sammy’s information with your own if you wish to personalize the size. The methods you use here can be applied to other CSS/HTML website projects.



Employment section of demonstration website

To build this section, you will add and style a section heading, add and style a wide column, and add and style an HTML table inside of the

column.

Prerequisites

To follow this tutorial, make sure you have set up the necessary files and folders as instructed in a previous tutorial in this series How To Set Up You CSS and HTML Practice Project.

Creating a Section Break and Section Title

To get started, create a class that will add space between the content in the prior “Projects” section and this “Employment” section. Add the following CSS comments and CSS ruleset to the bottom of your `styles.css` file:

styles.css

```
. . .
/* Section 4 */

/* Add space between sections */
.section-break {
  margin:50px;
  height:500px;
}
```

In this code snippet you have added a CSS comment labeling the CSS rulesets for “Section 4” and a CSS comment explaining the purpose of the `section-break` class. You will assign this class to an empty `<div>` in the `index.html` file, which will give it a height of 500 pixels and a margin of 50 pixels. Though the `<div>` will be invisible, it’s height will act as a section break by pushing subsequent content 500 pixels down the page.

Return to your `index.html` file and add the following code snippet:

index.html . . .

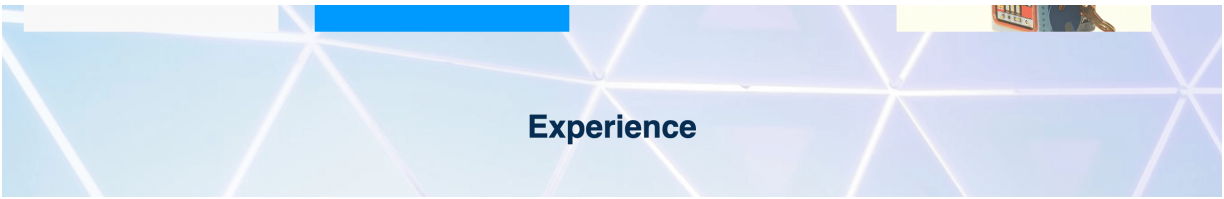
```
<!--Section 4: Employment—>

<div class="section-break"> </div>
<h2 class="section-heading">Experience</h2>
```

This code snippet adds an HTML comment to label the HTML code used for the fourth section of the website, and adds a `<div>` container assigned the `section-break` class that you just created. The code snippet also adds the "Experience" section heading and styles it using the class `section-heading` that you created in the previous tutorial [How To Build a Tiled Layout With CSS] (https://www.digitalocean.com/community/tutorials/how-to-build-a-tiled-layout-with-css-section-3).

Note: If you have not been following along with this tutorial series, you can add the `section-heading` class to your `styles.css` file by adding the following code snippet to the bottom of the file:

styles.css . . . .section-heading { text-align:center; color:#102C4E; margin-top: 150px; margin-bottom:70px; font-size: 35px; }

Save your `index.html` file and load it in the browser. You should now have a section heading named "Experience" following a section break:



Screenshot of “Experience” heading on demonstration website

Styling a Wide Column and Table

Next, you will create classes that will allow you to style the wide white column and the table you will place inside it. Add the following code snippet at the bottom of the `styles.css` file:

styles.css . . .

```
/* Wide column */
.column-1 {
  width: 90%;
  height: auto;
  padding-top:70px;
  padding-left:70px;
  padding-bottom:70px;
  margin:auto;
  margin-bottom:50px;
  margin-top: 75px;
  background-color:white;
}

/* Table formatting */
.table-style {
  width:100%;
  border-spacing: 24px;
}
```

In the first ruleset, you have declared a number of style rules for the class `column-1`. Note that you have specified the `width` in a percentage so that the column will change size according to the width of the viewport. You have specified the `height` to `auto`, which will allow the table to adjust its height according to the height needs of the HTML content you

place inside. You have also created a rule to make the background color of a `<div>` assigned this class `white`.

If you want to learn more about the other declarations in this ruleset, please review the previous sections in this tutorial series on setting the sizes of content, padding, and margins.

In the second ruleset, you have defined the class `table-style` and declared a number of rules. The `width:100%` declaration makes the table's width take up the entire width of the container in which it's situated, which will be the wide column you're creating. The `border-spacing:24px;` declaration puts 24 pixels of space between the cells of the table, allowing the content of the table to take up the width of the column. If you didn't include this rule, each of the table cells would be much closer together.

Adding the Column and Table

Now you will add the column and table to the HTML file. Save your `styles.css` file, return to the `index.html` file and add the following code snippet just below the HTML line of code `<h2 class="section-heading">Experience</h2>`:

index.html

```
. . .

<div class="column-1">
  <h2>Employment</h2>
  <table class="table-style">
    <tr>
      <td>Freelance designer</td>
      <td>Seven Seas</td>
      <td>2015-present</td>
   </tr>
    <tr>
      <td>Associate Sea Creature</td>
      <td>Small Pond Productions</td>
      <td>2019-2020</td>
   </tr>
   <tr>
      <td>Associate Surfer</td>
      <td>Open Watery Web</td>
      <td>2018-2019</td>
   </tr>
   <tr>
      <td>Open Web Advocate</td>
      <td>Kiddie Pool Kubernetes</td>
      <td>2017-2018</td>
   </tr>
   <tr>
      <td>Assistant Shark</td>
```

```
        <td>Small Pond Pictures</td>
        <td>2016-2017</td>
    </tr>
  </table>
 </div>
</div>
```

In this code snippet, you have added a `<div>` container styled according to the `column-1` class and placed an HTML table inside styled with the `table-style` class. Inside the table, you have placed the Employment history content. The `<tr>` tag opens up a table row where the following three sets of table data (marked up with the `<td>` tag) are inserted. To read more about how HTML tables work, please visit our tutorial How To Create Tables With HTML

Save both files and reload your web page in the browser. Your webpage should now have a single wide column that contains a table with four rows and three columns as pictured at the beginning of this tutorial.

Note that the first three `<td>` elements are inserted between the first opening and closing set of `<tr>` tags. You can continue to add rows by using the same table row and data format and the column's height will adjust accordingly because you have set the `height` to `auto` for the `column-1` class. Or, you can add additional columns by adding `<td>` elements inside the `<tr>` rows.

Conclusion

You have now created and styled a table with CSS to display employment history content in a structured layout. Experiment with sizing and adding rows and columns to customize tables for different purposes. In the next tutorial, you will continue exploring table layout possibilities by creating a table for “Education” and “Skills”.

How To Add Your Educational History and Skills To Your Website Using CSS (Section 5)

Written by Erin Glass

In this tutorial, you will recreate the “Education” section and “Skills” section (or fifth section) of the demonstration website using HTML tables and CSS classes. Feel free to switch out Sammy’s information with your own if you wish to personalize your website. The methods you use here can be applied to other CSS/HTML website projects.

Education and Skills section of demonstration website

To build these sections, you’ll create a CSS class that styles two equal-sized content boxes that can fit side by side on the webpage. You’ll then add a table inside each box where you will add text content.

Prerequisites

To follow this tutorial, make sure you have set up the necessary files and folders as instructed in a previous tutorial in this series How To Set Up You CSS and HTML Practice Project.

Creating and Styling Two Equal-Sized Tables

First, copy and paste the following code snippet at the bottom of your `styles.css` file:

styles.css

```
. . .
/* Fifth section */

.column-2a {
  float: left;
  width: 45%;
  padding: 40px;
  padding-left:70px;
  padding-right: 70px;
  padding-bottom:60px;
  height:450px;
  margin:30px;
  margin-right:30px;
  margin-bottom: 40px;
  background-color:white;
}
```

This code snippet creates the class `column-2a`, which is like the `column-2` class you created to style the "About" section in a previous tutorial in this series, except that its height property is set to `450px`. If you change the amount of content in this box, you may need to adjust the height accordingly, otherwise it may overflow or be cut off. If you want to learn more about the other declarations, please review the previous sections in this tutorial series on setting the sizes of content, padding, and margins.

Save the `styles.css` file before you proceed.

Next, return to the `index.html` file and paste the following code snippet after the last closing `</div>` tag:

index.html

```
. . .

<!--Section 5: Education and Skills-->

<div class="column-2a">
  <h2>Education</h2>
    <table class="table-style">
      <tr>
        <td>Barnacle Bootcamp</td>
        <td>2020</td>
      </tr>
      <tr>
        <td>Seaweed University</td>
        <td>2019-2020</td>
      </tr>
      <tr>
        <td>Highwater High School</td>
        <td>2018-2019</td>
      </tr>
      <tr>
        <td>Middle-Sized Pond Middle School</td>
        <td>2017-2018</td>
      </tr>
      <tr>
        <td>Minnow Elementary School</td>
        <td>2016-2017</td>
      </tr>
```

```
        <tr>
          <td>Phytoplankton Preschool</td>
          <td>2015-2016</td>
        </tr>
      </table>
</div>
```

This code snippet creates a column using the “column-2a” class and inserts a table styled with the `table-style` class created in the previous tutorial. Inside the table, you have placed your Educational history content. The `<tr>` tag opens up a table row where the following three sets of table data (marked up with the `<td>` tag) are inserted. To read more about how HTML tables work, please visit our tutorial How To Create Tables With HTML

Save the file and reload your browser to check that the table is showing up correctly. You should have table like the following screenshot:

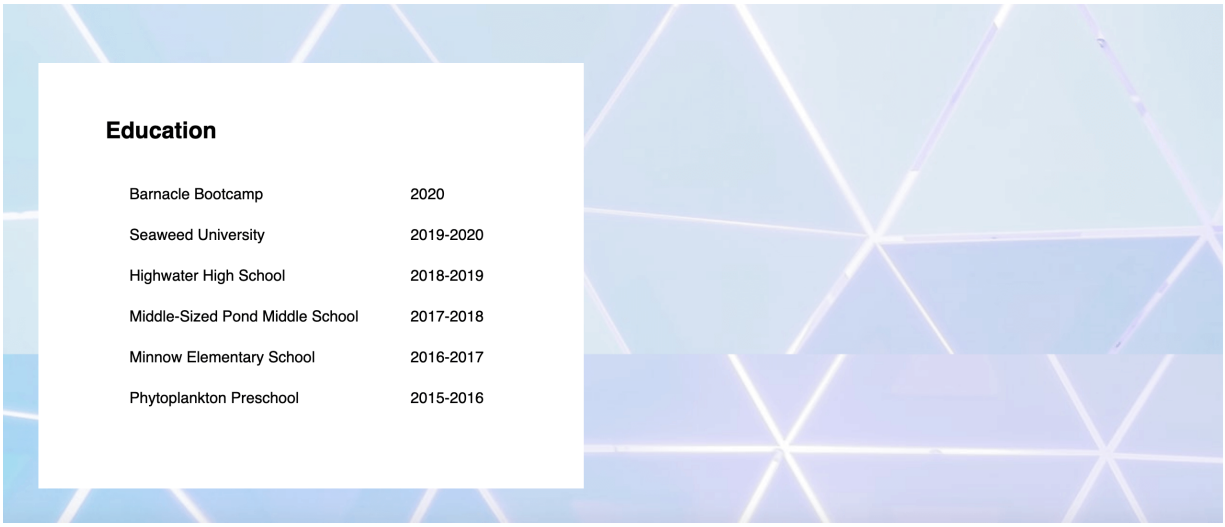


Table with educational content

Next, you will add the second table that lists Sammy's skills. Return to the `index.html` file and paste the following code snippet after the last closing `</div>` tag:

index.html

. . .

```
  <div class="column-2a">
     <h2>Skills</h2>
    <table class="table-style">
      <tr>
        <td>Social Media</td>
        <td>⭑ ⭑ ⭑ ⭑ ⭑</td>
      </tr>
      <tr>
        <td>Public Speaking</td>
        <td>⭑ ⭑ ⭑ ⭑ ⭑</td>
      </tr>
      <tr>
        <td>Internet Ethics Ambassador</td>
        <td>⭑ ⭑ ⭑ ⭑</td>
      </tr>
      <tr>
        <td>Content production</td>
        <td>⭑ ⭑ ⭑ ⭑ ⭑</td>
      </tr>
      <tr>
        <td> HTML</td>
        <td>⭑ ⭑ ⭑</td>
      </tr>
      <tr>
        <td> CSS</td>
```

```
        <td>⭐⭐⭐</td>
      </tr>
    </table>
  </div>
```

This code snippet works exactly like the previous code snippet: it creates a column using the `column-2a` class and inserts a table styled with the `table-style` class. Note that you are using emojis to create the star image. You can use any emoji as HTML text content.

Save the file and reload your browser to check that the table is showing up correctly. You should now have two tables displayed side by side as shown in the image at the beginning of this tutorial.

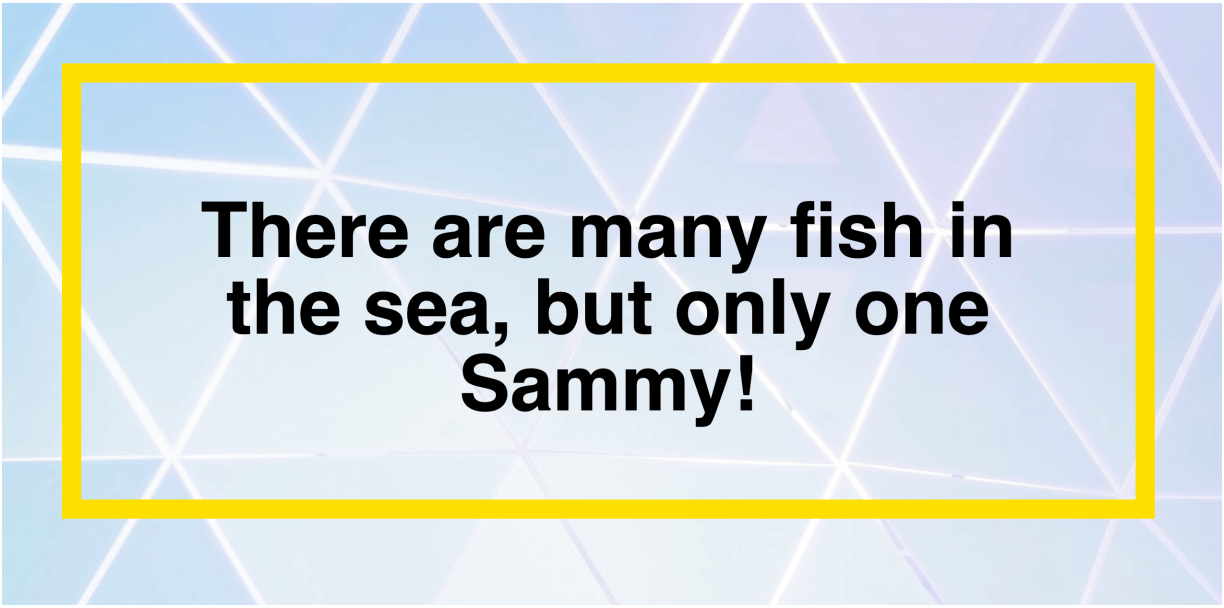
Conclusion

You have now added text content using styled tables. You can experiment with sizing and adding rows and columns to customize tables for different purposes. In the next tutorial, you will create a content box with a large featured quote on your website.

How To Create a Featured Quote Box On Your Website Using CSS (Section 6)

Written by Erin Glass

In this tutorial, you will add a featured quote to your website using CSS as displayed in the sixth section of the demonstration website. You might use this section to feature a favorite quote, a testimony about your work, or a message to your site visitors. You can also hyperlink this quote to another webpage if you wish. The methods you use here can be applied to other CSS/HTML website projects.



Featured quote section on demonstration website

Prerequisites

To follow this tutorial, make sure you have set up the necessary files and folders as instructed in a previous tutorial in this series How To Set Up You CSS and HTML Practice Project.

Creating Style Rules For the Featured Quote Section

To create the featured quote section, you will create a class to style the container and a class to style the featured text. In your `styles.css` file, add the following code snippets:

styles.css

```
. . .
/* Section 6: Featured Quote  */

.column-quote {
  width: 90%;
  height: 475px;
  padding: 40px;
  padding-left:70px;
  padding-right: 70px;
  padding-bottom:100px;
  margin:auto;
  margin-bottom:150px;
  border: 20px solid #FEDE00;
}

.quote {
  font-size:80px;
  font-weight:bold;
  line-height: 1;
  text-align: center;
}
```

In this code snippet, you have added the CSS comment `/* Section 6: Featured Quote */` to label this section of the CSS code. Then,

you have defined the class `column-quote`, which you will use to style the quote box, and specified the size, padding, margins, and border of the container.

Note that the margin is set to `auto`, which horizontally centers the container in the middle of the page. In addition, the bottom margin is set to 200 pixels to give some space to the bottom of the page. If you want to learn more about the other declarations, please review the previous sections in this tutorial series on setting the sizes of content, padding, borders, and margins.

You have also created the `quote` class, which you will use to style the text of the featured quote. Note that you have set the `line-height` property to `1`, which shrinks the space between text lines from the default setting of 1.6. Experiment with changing this value to determine what line spacing you prefer.

Save the `styles.css` file.

Adding the Featured Quote Section

Return to the `index.html` file. After the last closing `</div>` tag, add the following code snippet:

index.html

```
. . .

<!--Section 6: Featured Quote-->

<div class="section-break"> </div>
<div class="column-quote">
  <p class="quote">There are many fish in the sea,
but only one Sammy!</p>
</div>
```

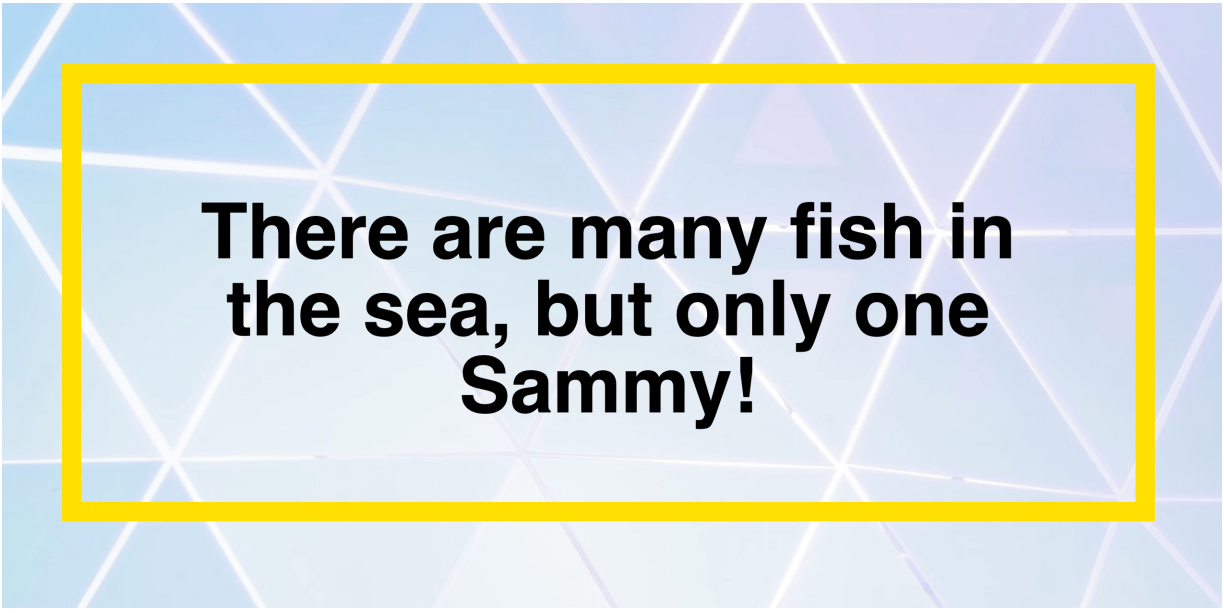
Before moving on, let's pause to examine each line of code:

- The HTML comment `<!--Section 6: Featured Quote-->` labels this section of code in the `index.html` file and will not be displayed by the browser.
- The `<div class="section-break"> </div>` element creates a section break using a class you may have defined in a previous tutorial. If you did not follow that tutorial, you can add that class by adding the CSS rule `.section-break {margin:50px; height:500px;}` to your `styles.css` file. This element creates space between the content in the previous section and the featured quote section.
- The `<div class="column-quote">` tag and its closing `</div>` tag create a container for the featured quote with the style

rules you declared for the `column-quote` class in your `styles.css` file.

- The `<p class="quote">There are many fish in the sea, but only one Sammy! </p>` inserts the text content into the `<div>` container you opened in the line above and styles it according to the rules you declared for the `quote` class selector in your `styles.css` file. If you change the length of your text content, you may want to modify one of the classes in this section to change the size of the font or the size of the container to make sure the text still fits.

Save the `index.html` file and reload it in your browser. Your webpage should now display a large featured quote in a transparent container with a solid background:



Featured quote section on demonstration website

Conclusion

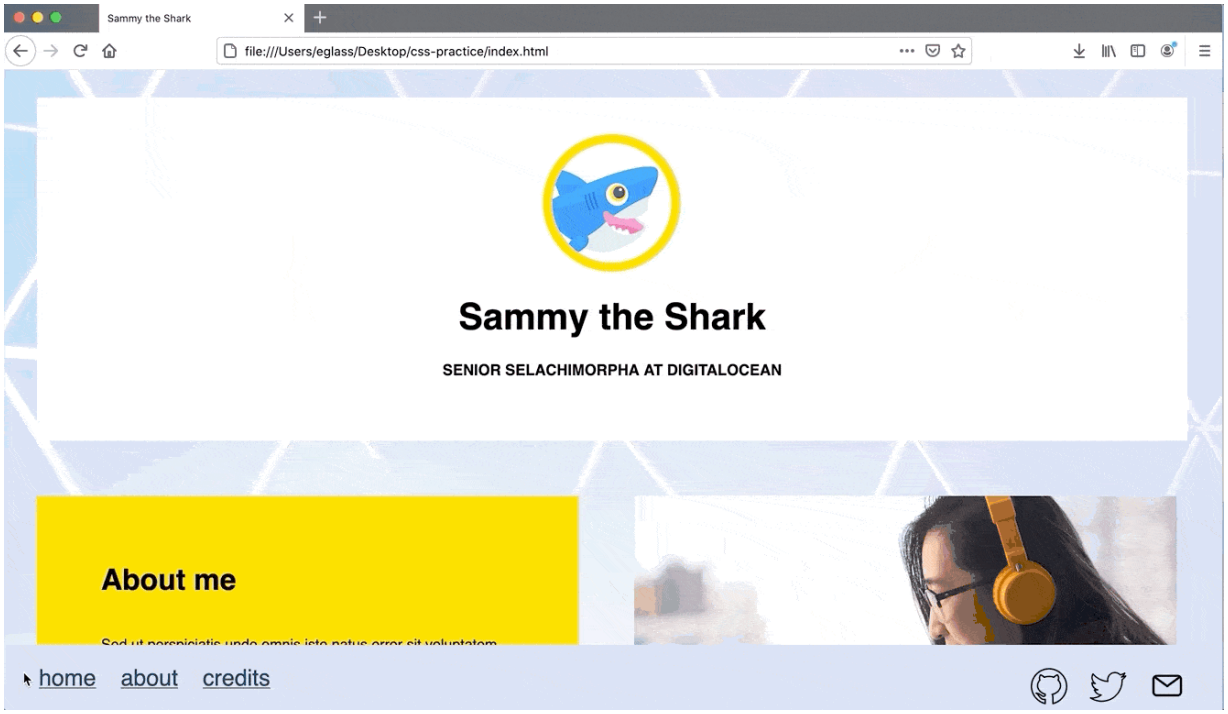
In this tutorial, you created a featured text box on your website and modified its style for different website layouts. If you wish to hyperlink your quote to a new website page, please visit our tutorial How To Create and Link To Additional Website Pages With HTML.

In the next and final tutorial of the tutorial series, you will create a static footer, which “sticks” in a fixed position at the bottom of the viewport as the visitor scrolls down the page.

How To Create a Static Footer With HTML and CSS (Section 7)

Written by Erin Glass

In the final tutorial of the CSS series, you will create a static footer that stays in a fixed position at the bottom of the viewport as the visitor scrolls down the page. This tutorial will recreate the footer in the demonstration website but you can use these methods for other website projects as well.



Gif of static footer on demonstration website

Prerequisites

To follow this tutorial, make sure you have set up the necessary files and folders as instructed in a previous tutorial in this series How To Set Up You

CSS and HTML Practice Project.

This tutorial uses several social media icons as content in the footer. If you'd like to use these icons, download them now from our demonstration site and save them to your images folder as:

- "twitter.jpeg"
- "github.jpeg"
- "email.jpeg."

To download these images, click on the linked filename above and then click `CTRL + Left Click` (on Macs) or `Right Click` (on Windows) while hovering on the image and select "Save Image As". Save the images with the instructed file names to your `images` folder.

Once you have your icons saved, you can proceed to the next section.

Adding a Class To Style Your Footer

First you will define a "footer" class by adding the following code snippet to the bottom of the `styles.css` file:

styles.css

```
. . .
/* Footer */

.footer {
  position:fixed;
  bottom:0;
  left:0;
  width:100%;
  height: 90px;
  background-color: #D0DAEE;
}
```

Save the `styles.css` file. In this code snippet you have added a comment to label the CSS code for the Footer section. You then defined a class named `footer` and declared several style rules. The first rule declares its `position` as `fixed`, which means the element will not move from the location you specify as the user scrolls down the page. This location is specified by the next two declarations: `bottom:0` and `left:0`, which specifies a location zero pixels from the left and zero pixels from the bottom of the browser's viewport.

By changing these values, you can change the location of the element on the page. Note, however, that any value aside from zero needs to include the `px` suffix after the number. The ruleset also specified the width, height, and background color of the `footer` class.

You are now ready to add the footer content in the next section of this tutorial.

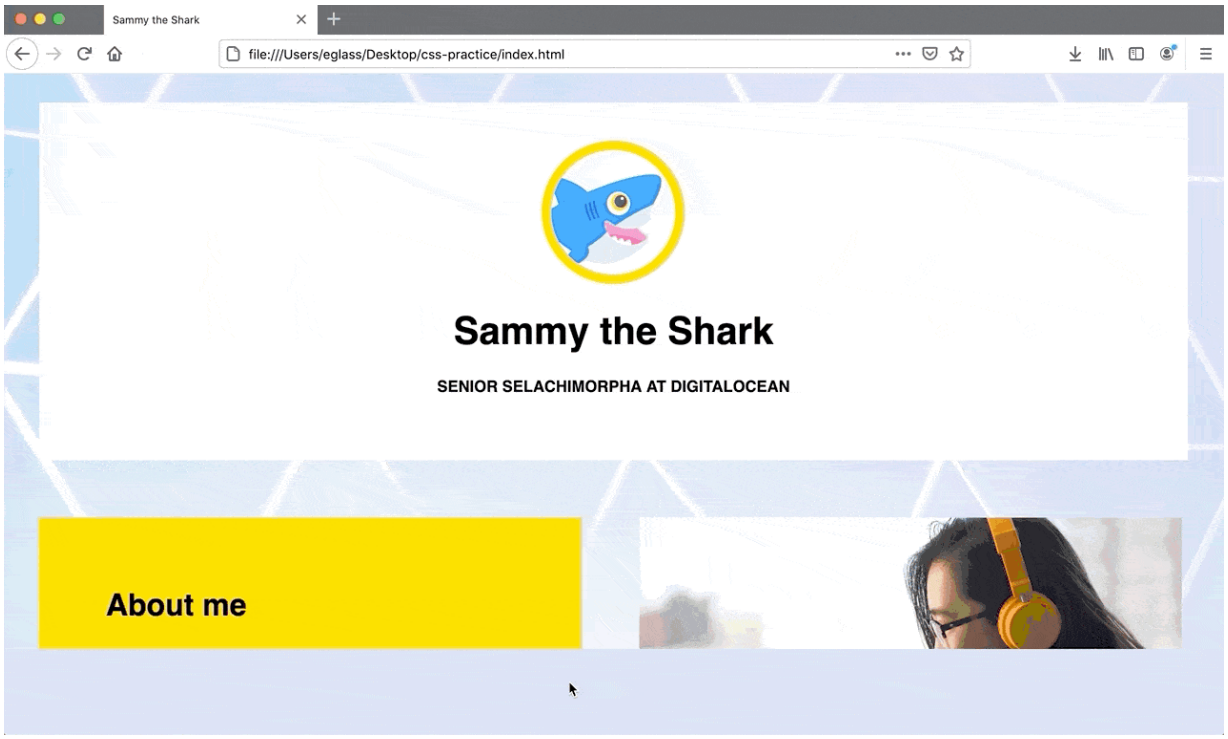
Adding a Footer Styled With Your Footer Class

To add the footer content, you will add a `<div>` container to the webpage and assign the footer class that you just created. Return to your `index.html` file and paste the following code snippet after the end of the last closing `</div>` tag:

index.html

```
. . .

<!--Section 7: Footer-->

<div class="footer">
</div>
```

Save your `index.html` file and reload it in the browser. (For instructions on loading an HTML file, please visit our tutorial step How To View An Offline HTML File In Your Browser). You should now have an empty footer section at the bottom of your webpage that stays in place as you scroll up and down the page:



Gif of blank fixed footer

Next you will add content to the newly created footer.

How To Add and Style Menu Items To Your Footer

In this step, you will add and style the menu items to the left side of the footer. These menu items can be used to link to other pages on your site. Currently, there is only one webpage on your site, so you can use the links we provide for demonstration purposes. Later on, if you add additional pages to your website you can create menu items here and add the correct links. You can learn how to create and link to new webpages with this tutorial on How to Build a Website with HTML.

Return to your `styles.css` file and add the following code snippet to the bottom of the file:

styles.css

```
. . .
.footer-text-left {
  font-size:25px;
  padding-left:40px;
  float:left;
  word-spacing:20px;
}

a.menu:hover {
  background-color:yellow;
  font-size:20px;
}
```

Let's pause briefly to review each of the rulesets we've created:

- The first ruleset defines a class named `footer-text-left` that will be used to style the menu item text. Note that you are setting the `float` property to `left` so that the text assigned to this class will float to the left of the page. You are also using the `word-spacing` property to grant extra space between the menu items. If any of your menu items are more than one word, you'll need to create a class for styling the menu items (instead of just changing the word spacing value).
- The second ruleset uses the `hover` pseudo-class to add a yellow background color to the text when the user hovers their cursor over the

text.

Now you will add the menu items to the webpage. Return to your `index.html` file and add the following highlighted code snippet inside the footer container that you've already created:

index.html . . .

```
<div class="footer">
  <p class="footer-text-left">
    <a href="index.html" class="menu">home</a>
    <a href="https://css.sammy-
codes.com/about.html" class="menu">about</a>
    <a href="https://css.sammy-
codes.com/credits.html" class="menu">credits</a>
  </p>
</div>
```

This code snippet adds two menu items ("about" and "credits"), links these menu items, and styles the text with the `footer-text-left` and `a.menu` classes you just created.

Save both files and reload your webpage in the browser. You should receive something like this:

Gif of footer with menu items

Adding Social Media Icons

Next you will add the social icons to the footer, which you can use to link to your social media accounts. If you want to use icons for different social media platforms, you can search for free icons on the web and download them to your `images` folder. Return to your `styles.css` file and add the following three rulesets to the bottom of your file:

styles.css . . .

```
.footer-content-right {
  padding-right:40px;
  margin-top:20px;
  float:right;
}

.icon-style {
  height:40px;
  margin-left:20px;
  margin-top:5px;
}

.icon-style:hover {
  background-color:yellow;
  padding:5px;
}
```

Let's pause to review each ruleset:

- The first ruleset defines the class `footer-content-right` and assigns it specific padding, margin, and float values. You will use this ruleset to style a `<div>` element that will hold the social media icons.
- The second ruleset creates the class `icon-style` that will provide height and margin values to the size and position of the social media icons.

- The third ruleset uses the `hover` pseudo-class to add a yellow background to the icon when the user hovers their cursor over the text.

Save your `styles.css` file. You will now add the social media icons to the footer. Return to your `index.html` file and add the following code snippet after the last closing `</a>` tag of the menu items:

index.html

```
. . .

<div class="footer-content-right">
  <a href="https://github.com/digitalocean"><img
src="images/github.jpeg" class="icon-style"
alt="Github icon"></a>
  <a href="https://www.twitter.com/DigitalOcean">
<img src="images/twitter.jpeg" class="icon-style"
alt="Twitter icon"></a>
  <a href="https://www.twitter.com"><img
src="images/email.jpeg" class="icon-style"
alt="Emailicon"></a>
</div>
```

Make sure that you change the file paths and links with your own social media information.

This code snippet creates a `<div>` container, which is assigned the style of `footer-content-right` the class. Inside this `div` container, you

have added three social media icons using the HTML `<img>` tag, and linked each image using the HTML `<a>` tag.

You have also added the alternative text that describes each icon using the `alt` attribute. When creating websites, alternative text should be added to all images to support site accessibility for individuals who use screen readers. To read more about using alternative text with HTML, please visit the section on alternative text in our guide How To Add Images To Your Webpage Using HTML.

Save your `index.html` file and reload it in the browser. You should now have a fixed footer with three social media icons on the right that link to your accounts. The links should change color when the user hovers their cursor over them. To confirm your results, you can compare them to the gif at the beginning of this tutorial.

Conclusion

You have now created a static footer that stays in a fixed position at the bottom of the viewport as the visitor scrolls down the page. You can continue exploring footer design and content possibilities by changing values in the CSS classes that you created, or add different types of content to your `index.html` file. For more ideas on exploring design and layout possibilities for your website, the conclusion of this tutorial series has more suggestions for things to try like rearranging content sections, adding links to other pages, and changing layout styles using the box model.