1.

Unit 2 CSS (6 days)

Welcome to the second unit of Build Websites from Scratch!
In this unit, you will learn how to style your websites using CSS! This will allow you to begin making your projects unique and beautiful.
You'll start by learning how to create CSS files and hook them up to your HTML pages.
You'll then learn about the syntax of CSS, diving deep into how to select specific elements using CSS selectors and utilizing some of the most important CSS visual rules.
Finally, you'll build and style two websites from scratch on your computer. You should notice by the end of this unit that your sites are beginning to look more and more professional. You've already covered a lot of ground in a short amount of time!
Good luck!

Day 1

Today you will explore the power of CSS! You will learn what CSS is and how it can be used to make your plain HTML look much more exciting.

You will start with a lesson on how to create CSS files and link them to HTML documents.

You will then follow that up with a lesson on the many different ways you can select HTML elements to be styled.

Good luck!

Lesson

CSS Setup

The content on your web page won't change unless you style it with CSS. In this lesson you'll learn how to incorporate CSS in your HTML code.

What is CSS?

So far, you've learned about the fundamentals of HTML, including the basic structure required to set up HTML files and the HTML elements commonly used to add content to a web page.
The HTML elements that we've used to add content to a web page have resulted in fairly bland results in the browser. For example, most of the content we've created has been the same color, and the same font — and we've had no direct control over the size of the text (apart from the six different heading options). How can we make our HTML more visually appealing?
CSS, or Cascading Style Sheets, is a language that web developers use to *style* the HTML content on a web page. If you're interested in modifying colors, font types, font sizes, shadows, images, element positioning, and more, CSS is the tool for the job!
Let's begin!

==========

style.css (original code)
==========

```
html, body {
  margin: 0;
  padding: 0;
}
body {
  font-family: 'Roboto', sans-serif;
  font-weight: 100;
}

.container {
  margin: 0 auto;
  max-width: 940px;
  padding: 0 10px;
}

.header {
  background: url(http://s3.amazonaws.com/codecademy-content/projects/innovation-cloud/bg.jpg) no-repeat
center center;
  background-size: cover;
  height: 800px;
  text-align: center;
}

.header .container {
  position: relative;
  top: 200px;
}

.header h1 {
  color: #fff;
  line-height: 100px;
  font-size: 80px;
  margin-top: 0;
  margin-bottom: 80px;
  text-transform: uppercase;
}

@media (min-width:850px) {
  .header h1 {
    font-size: 120px;
  }
}

.header p {
  color: #fff;
  font-weight: 500;
  letter-spacing: 8px;
  margin-bottom: 40px;
  margin-top: 0;
  text-transform: uppercase;
}
```

```
.btn {
  color: #fff;
  background: #000;
  padding: 10px 40px;
  text-decoration: none;
  transition: background .5s;
}

.nav {
  background: #000;
  height: 80px;
  width: 100%;
}

.nav ul {
  height: 80px;
  list-style: none;
  margin: 0 auto;
  padding: 0;
}

.nav ul li {
  color: #fff;
  display: inline-block;
  height: 80px;
  line-height: 80px;
  list-style: none;
  padding: 0 10px;
  transition: background .5s;
}

.btn:hover, .nav ul li:hover {
  background: #117bff;
  cursor: pointer;
  transition: background .5s;
}

.main .container {
  margin: 80px auto;
}

.main img {
  float: left;
  margin: 50px 80px 50px 0;
}

.jumbotron {
  background: url(http://s3.amazonaws.com/codecademy-content/projects/innovation-cloud/jumbotron_bg.jpg)
center center;
  background-size: cover;
  height: 600px;
}
```

```
.jumbotron .container {
  position: relative;
  top: 220px;
}

.jumbotron h2 {
  color: #fff;
  text-align: right;
}

.jumbotron p {
  color: #fff;
  text-align: right;
}

.jumbotron .btn {
  margin: 10px 0 0;
  float: right;
}

.footer {
  background: #000;
  height: 80px;
  padding-bottom: 50px;
}

.footer p {
  color: #fff;
  font-size: 14px;
  height: 80px;
  line-height: 80px;
  margin: 0;
}

@media (max-width: 500px) {
  .header h1 {
    font-size: 50px;
    line-height: 64px;
  }

  .main, .jumbotron {
    padding: 0 30px;
  }

  .main img {
    width: 100%;
  }
}
```

==========

end of code

Inline Styles

Although CSS is a different language than HTML, it's possible to write CSS code directly within HTML code using *inline styles*.

To style an HTML element, you can add the `style` attribute directly to the opening tag. After you add the attribute, you can set it equal to the CSS style(s) you'd like applied to that element.

```
<p style="color: red;">I'm learning to code!</p>
```

The code in the example above demonstrates how to use inline styling. The paragraph element has a `style` attribute within its opening tag. Next, the `style` attribute is set equal to `color: red;`, which will set the color of the paragraph text to red within the browser.

You might be wondering about the syntax of the following snippet of code: `color: red;`. At the moment, the details of the syntax are not important; you'll learn more about CSS syntax in other exercises. For now, it's important to know that inline styles are a quick way of directly styling an HTML element.

If you'd like to add *more* than one style with inline styles, simply keep adding to the `style` attribute. Make sure to end the styles with a semicolon (`;`).

```
<p style="color: red; font-size: 20px;">I'm learning to code!</p>
```

The <style> Tag

Inline styles are a fast way of styling HTML, but they also have limitations. If you wanted to style, for example, multiple `<h1>`elements, you would have to add inline styling to each element manually. In addition, you would also have to maintain the HTML code when additional `<h1>` elements are added.

Fortunately, HTML allows you to write CSS code in its own dedicated section with the `<style>` element. CSS can be written between opening and closing `<style>` tags. To use the `<style>` element, it must be placed inside of the head.

```
<head>
  <style>

  </style>
</head>
```

After adding a `<style>` tag in the head section, you can begin writing CSS code.

```
<head>
  <style>
    p {
      color: red;
      font-size: 20px;
```

```
    }
  </style>
</head>
```

The CSS code in the example above changes the color of all paragraph text to red and also changes the size of the text to 20 pixels. Note how the syntax of the CSS code matches (for the most part) the syntax you used for inline styling. The main difference is that you can specify which elements to apply the styling to.

Again, the details of the CSS syntax in the example above aren't important at the moment. You will learn more about the details of CSS syntax in later lessons.

```
<!DOCTYPE html>
<html>
<head>
 <style>
 p {
  font-family: Arial;
 }
 </style>
 <style>
  p {
   color: red;
   font-size: 20px;
  }
 </style>
 <title>Animals Around the World</title>

</head>
```

THE CSS SETUP

Structure vs Style

Although the `<style>` element allows you to write CSS code within HTML files, this mixture of HTML and CSS can result in code that is difficult to read and maintain.

It's common for developers to add substantial amounts of custom CSS styling to a web page. When that CSS code is placed within a `<style>` element in an HTML file, you risk the following two things:

1. Creating a large HTML file that is difficult to read and maintain (by you and other developers). This can result in an inefficient workflow.
2. Maintaining a clear distinction between web page structure (HTML) and web page styling (CSS).

THE CSS SETUP

The .css file

Developers avoid mixing code by storing HTML and CSS code in separate files (HTML files contain only HTML code, and CSS files contain only CSS code).

You can create a CSS file by using the **.css** file name extension, like so: **style.css**

With a CSS file, you can write all the CSS code needed to style a page without sacrificing the readability and maintainability of your HTML file.

Linking the CSS File

Perfect! We successfully separated structure (HTML) from styling (CSS), but the web page still looks bland. Why?

When HTML and CSS code are in separate files, the files must be linked. Otherwise, the HTML file won't be able to locate the CSS code, and the styling will not be applied.

You can use the `<link>` element to link HTML and CSS files together. The `<link>` element must be placed within the head of the HTML file. It is a self-closing tag and requires the following three attributes:

1. `href` — like the anchor element, the value of this attribute must be the address, or path, to the CSS file.
2. `type` — this attribute describes the type of document that you are linking to (in this case, a CSS file). The value of this attribute should be set to `text/css`.
3. `rel` — this attribute describes the relationship between the HTML file and the CSS file. Because you are linking to a stylesheet, the value should be set to `stylesheet`.

When linking an HTML file and a CSS file together, the `<link>`element will look like the following:

```
<link href="https://www.codecademy.com/stylesheets/style.css" type="text/css" rel="stylesheet">
```

Note that in the example above the path to the stylesheet is a URL: `https://www.codecademy.com/stylesheets/style.css`

Specifying the path to the stylesheet using a URL is one way of linking a stylesheet.

If the CSS file is stored in the same directory as your HTML file, then you can specify a relative path instead of a URL, like so:

```
<link href="./style.css" type="text/css" rel="stylesheet">
```

Using a relative path is very common way of linking a stylesheet.

Review: CSS Setup

Great job! You learned how to link an HTML file and a CSS file together.

Let's review what you've learned so far:

1. HTML and CSS are kept in separate files to keep code maintainable and readable, as well as keep structure separate from styling.
2. The `<style>` element allows you to write CSS code within an HTML file.
3. A CSS stylesheet can be linked to an HTML file using the `<link>` element, which requires three attributes:

- `href` - set equal to the path of the CSS file.
- `type` - set equal to `text/css`.
- `rel` - set equal to `stylesheet`.

Introduction to CSS Selectors

CSS is a styling language that can add color and structure to an HTML website. CSS works by selecting HTML elements, then applying styles to them. In this lesson, we'll focus on how CSS selects HTML elements.

To style an HTML element, CSS first has to select it, using something called a CSS selector.

CSS always follows this two part process:

1. Select HTML elements.
2. Apply styles to the elements.

In this lesson, we'll focus on how CSS selects HTML elements. There are lots of ways to select elements, and we'll show you a number of the most common.

Let's get started.

2/13

Tag Name

CSS can select HTML elements by using an element's tag name. A tag name is the word (or character) between HTML angle brackets.

For example, in HTML, the tag for a paragraph element is `<p>`. The CSS syntax for selecting `<p>` elements is:

```
p {

}
```

In the example above, all paragraph elements will be selected using a CSS *selector*. The selector in the example above is `p`. Note that the CSS selector matches the HTML tag for that element, but without the angle brackets.

In addition, two curly braces follow immediately after the selector (an opening and closing brace, respectively). Any CSS properties will go inside of the curly braces to style the selected elements.

```
h1 {
color: maroon;
}
```

3/13

Class Name

CSS is not limited to selecting elements by tag name. HTML elements can have more than just a tag name; they can also have *attributes*. One common attribute is the `class` attribute. It's also possible to select an element by its `class` attribute.

For example, consider the following HTML:

```
<p class="brand">Sole Shoe Company</p>
```

The paragraph element in the example above has a `class`attribute within the `<p>` tag.
The `class` attribute is set to `"brand"`. To select this element using CSS, we could use the following CSS selector:

```
.brand {

}
```

To select an HTML element by its class using CSS, a period (`.`) must be prepended to the class's name. In the example above case, the class is `brand`, so the CSS selector for it is `.brand`.

```
h1 {
color: maroon;
}
.title {
color: teal;
}
```

4/15
Multiple Classes
We can use CSS to select an HTML element's `class` attribute by name.
So far, we've selected elements using only one class name per element. If every HTML element had a single class, all the style information for each element would require a new class.
Luckily, it's possible to add more than one class name to an HTML element's `class` attribute.
For instance, perhaps there's a heading element that needs to be green and bold. You could write two CSS rules like so:

```
.green {
  color: green;
}

.bold {
  font-weight: bold;
}
```

Then, you could include both of these classes on one HTML element like this:

```
<h1 class="green bold"> ... </h1>
```

We can add multiple classes to an HTML element's `class`attribute by separating them with a space.
This enables us to mix and match CSS classes to create many unique styles without writing a custom class for every style combination needed.
<h1 class="title uppercase">Top Vacation Spots</h1>

5/13

ID Name

If an HTML element needs to be styled uniquely (no matter what classes are applied to the element), we can add an ID to the element. To add an ID to an element, the element needs an `id` attribute:

```
<h1 id="large-title"> ... </h1>
```

Then, CSS can select HTML elements by their `id` attribute. To select an `id` element, CSS prepends the `id` name with a hashtag (`#`). For instance, if we wanted to select the HTML element in the example above, it would look like this:

```
#large-title {

}
```

The `id` name is `large-title`, therefore the CSS selector for it is `#large-title`.

Supposed to work but has flags....

```
<h1 class="title uppercase id=" article-title""="">Top Vacation Spots </h1>
```

========

6/13

Classes and IDs

CSS can select HTML elements by their tag, class, and ID. CSS classes and IDs have different purposes, which can affect which one you use to style HTML elements.

CSS classes are meant to be reused over many elements. By writing CSS classes, you can style elements in a variety of ways by mixing classes on HTML elements.

For instance, imagine a page with two headlines. One headline needs to be bold and blue, and the other needs to be bold and green. Instead of writing separate CSS rules for each headline that repeat each other's code, it's better to write a `.bold` CSS rule, a `.green` CSS rule, and a `.blue` CSS rule. Then you can give one headline the `bold green` classes, and the other the `bold blue` classes.

While classes are meant to be used many times, an ID is meant to style only one element. As we'll learn in the next exercise, IDs override the styles of tags and classes. Since IDs override class and tag styles, they should be used sparingly and only on elements that need to always appear the same.

```
h1 {
  color: maroon;
}
```

```
.title {
  color: teal;
}

.uppercase {
  text-transform: uppercase;
}
#article-title {
font-family: cursive;
text-transform: capitalize;
}
```

---

line 11: <h1 class="title uppercase" id="article-title">Top Vacation Spots</h1>

line 13: <h6 class="publish-time">Published: 2 Days Ago</h6>

==========
css code

```
h1 {
  color: maroon;
}

.title {
  color: teal;
}

.uppercase {
  text-transform: uppercase;
}

#article-title {
  font-family: cursive;
          text-transform: capitalize;
}

.publish-time {
  color: gray;
}
```

==========
7/13

Specificity

Specificity is the order by which the browser decides which CSS styles will be displayed. A best practice in CSS is to style elements while using the lowest degree of specificity, so that if an element needs a new style, it is easy to override.

IDs are the most specific selector in CSS, followed by classes, and finally, tags. For example, consider the following HTML and CSS:

```
<h1 class="headline">Breaking News</h1>
```

```
h1 {
  color: red;
}

.headline {
  color: firebrick;
}
```

In the example code above, the color of the heading would be set to `firebrick`, as the class selector is more specific than the tag selector. If an ID attribute (and selector) were added to the code above, the styles within the ID selector's body would override all other styles for the heading. The only way to override an ID is to add *another* ID with additional styling.

Over time, as files grow with code, many elements may have IDs, which can make CSS difficult to edit, since a new, more specific style must be created to change the style of an element.

To make styles easy to edit, it's best to style with a tag selector, if possible. If not, add a class selector. If that is not specific enough, then consider using an ID selector.

Line 11: <h1 class="title uppercase article-title">Top Vacation Spots</h1>

in CSS delete:

```
#article-title {
  font-family: cursive;
        text-transform: capitalize;
}
```

add;

```
.cursive {
  font-family: cursive;
}

.capitalize {
  text-transform: capitalize;
}
```

line 11 change:

<h1 class="title cursive capitalize article-title">Top Vacation Spots</h1>

Chaining Selectors

When writing CSS rules, it's possible to require an HTML element to have two or more CSS selectors at the same time.

This is done by combining multiple selectors, which we will refer to as chaining. For instance, if there was a `.special`class for `h1` elements, the CSS would look like:

```
h1.special {

}
```

The code above would select only the `h1` elements that have a class of `special`. If a `p` element also had a class of `special`, the rule in the example would not style the paragraph.

Let's use chaining to select the destinations to add a style to them.

In **style.css**, write a CSS selector for `h2` elements with a class of `.destination`. Inside the selector's curly braces, write this:

```
font-family: cursive;
```

This will make the destinations cursive, like the title of the article.

Instructions seem to indicate the following:

```
h2 {
}
.destination {
 font-family: cursive;
}
```

and when I checked the code that is what they had.
But the first curly braces should not be there. h2 . destination
And they want it on a different line, right under the h1 code line before the title line.

```
h1 {
  color: maroon;
}

h2.destination {
  font-family: cursive;
}
```

.title { blah blah what is already there, etc.

9/13

Nested Elements

In addition to chaining selectors to select elements, CSS also supports selecting elements that are nested within other HTML elements. For instance, consider the following HTML:

```
<ul class='main-list'>
  <li> ... </li>
  <li> ... </li>
  <li> ... </li>
</ul>
```

The nested `<li>` elements are selected with the following CSS:

```
.main-list li {

}
```

In the example above, `.main-list` selects the `.main-list` element (the unordered list element). The nested `<li>` are selected by adding `li` to the selector, separated by a space, resulting in `.main-list li` as the final selector (note the space in the selector).
Selecting elements in this way can make our selectors even more specific by making sure they appear in the context we expect.

<h5 class="description">Top Attractions</h5>

css what I thought was:
h5.description { ….note: .description should not have been here; on next section see code below.
color: teal;
but not what they wanted:

.description h5 {
color: teal;
}

On the next section, I wanted their code to show so I could see this:

h1 {
color: maroon;
}

h2.destination {
font-family: cursive;
}

h5 {
color: rebeccapurple;
}

.description h5 {
color: teal;
}

.title {

Chaining and Specificity

In the last exercise, instead of selecting all `h5` elements, you selected only the `h5` elements nested inside the `.description` elements. This CSS selector was more specific than writing only `h5`. Adding more than one tag, class, or ID to a CSS selector increases the specificity of the CSS selector.
For instance, consider the following CSS:

```
p {
  color: blue;
```

```
}
```

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

Both of these CSS rules define what a `p` element should look like. Since `.main p` has a class and a `p` tag as its selector, only the `p` elements inside the `.main` element will appear `red`. This occurs despite there being another more general rule that states `p` elements should be `blue`.

I wrote

```
h5 {
  color: rebeccapurple;
}
.description h5 {
  color: teal;
}
```

and it was supposed to stay teal but it turned to rebeccapurple.

I would check the frame code but it is displaying what I wrote.

So I will have to figure out what they wanted later.

I tried inverting the lines

```
.description h5 {
  color: teal;
}
h5 {
  color: rebeccapurple;
}
```

nope...no difference

but it lets me move on so I will have figure out later what they wanted.

11/13

Important

There is one thing that is even more specific than IDs: `!important`. `!important` can be applied to specific attributes instead of full rules. It will override *any* style no matter how specific it is. As a result, it should almost never be used. Once `!important` is used, it is very hard to override.
The syntax of `!important` in CSS looks like this:

```
p {
  color: blue !important;
}
```

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

Since `!important` is used on the `p` selector's `color`attribute, all `p` elements will appear `blue`, even though there is a more specific `.main p` selector that sets the `color`attribute to `red`.

The `!important` flag is only useful when an element appears the same way 100% of the time. Since it's almost impossible to guarantee that this will be true throughout a project and over time, it's best to avoid `!important` altogether. If you ever see `!important` used (or are ever tempted to use it yourself) we strongly recommend reorganizing your CSS. Making your CSS more flexible will typically fix the immediate problem and make your code more maintainable in the long run.

Multiple Selectors

In order to make CSS more concise, it's possible to add CSS styles to multiple CSS selectors all at once. This prevents writing repetitive code.

For instance, the following code has repetitive style attributes:

```
h1 {
  font-family: Georgia;
}

.menu {
  font-family: Georgia;
}
```

Instead of writing `font-family: Georgia` twice for two selectors, we can separate the selectors by a comma to apply the same style to both, like this:

```
h1,
.menu {
  font-family: Georgia;
}
```

By separating the CSS selectors with a comma, both the `h1`and the `.menu` elements will receive the `font-family: Georgia` styling.

```
h5,
p {
  font-family: Georgia;
}
```

Review CSS Selectors

Throughout this lesson, you learned how to select HTML elements with CSS and apply styles to them. Let's review what you learned:

- CSS can change the look of HTML elements. In order to do this, CSS must select HTML elements, then apply styles to them.
- CSS can select HTML elements by tag, class, or ID.
- Multiple CSS classes can be applied to one HTML element.
- Classes can be reusable, while IDs can only be used once.
- IDs are more specific than classes, and classes are more specific than tags. That means IDs will override any styles from a class, and classes will override any styles from a tag selector.
- Multiple selectors can be chained together to select an element. This raises the specificity, but can be necessary.
- Nested elements can be selected by separating selectors with a space.
- The `!important` flag will override any style, however it should almost never be used, as it is extremely difficult to override.
- Multiple unrelated selectors can receive the same styles by separating the selector names with commas.

Great work this lesson. With this knowledge, you'll be able to use CSS to change the look and feel of websites to make them look great.

Final css code:

```
h1 {
  color: maroon;
}

h2.destination {
  font-family: cursive;
}
h5,
p {
  font-family: Georgia;
}
h5 {
  color: rebeccapurple !important;
}

.description h5 {
  color: teal;
}

.title {
  color: teal;
}
```

```
.uppercase {
  text-transform: uppercase;
}

.publish-time {
  color: gray;
}

.cursive {
  font-family: cursive;
```

Day 2

another 2 quiz

side-bar is never discussed.

CSS Visual Rules

The purpose of CSS is to modify the appearance of web pages. In this lesson you'll learn the structure and syntax of CSS as you create and modify web pages.

By the end of this lesson you'll be able to change the appearance of a web page using CSS selectors, properties, and values.

1/10

Introduction To Visual Rules

In this lesson, you'll learn the basic structure and syntax of CSS so that you can start styling web page elements.

2/10

CSS Structure

To style an HTML element using CSS, you need to write a CSS declaration inside the body of a CSS selector.

```
h1 {
  color: blue;
}
```

The example above selects the `<h1>` element. Inside of the selector's body, we typed `color: blue`.

This line is referred to as a CSS declaration. CSS declarations consist of a *property*and a *value*.

Property — the property you'd like to style of that element (i.e., size, color, etc.).

Value — the value of the property (i.e., 18px for size, blue for color, etc.).

In the example above, the property is `color` and the value is `blue`. Note that a semicolon (`;`) is always used at the end of a declaration.

Finally, the entire snippet of code in the example above is known as a *CSS rule*. A CSS rule consists of the selector (here, `h1`) and all declarations inside of the selector.

3/10

Font Family

If you've ever used a formatted word processor, chances are that you probably also used a feature that allowed you change the font you were typing in. Font refers to the technical term typeface, or *font family*.

To change the typeface of text on your web page, you can use the `font-family` property.

```
h1 {
  font-family: Garamond;
}
```

In the example above, the font family for all main heading elements has been set to `Garamond`.

When setting typefaces on a web page, keep the following points in mind:

1. The font specified in a stylesheet must be installed on a user's computer in order for that font to display when a user visits the web page.
2. The default typeface for all HTML elements is `Times New Roman`. You may be familiar with this typeface if you have ever used a formatted word processor. If no `font-family` attribute is defined, the page will appear in `Times New Roman`.
3. It's a good practice to limit the number of typefaces used on a web page to 2 or 3. This helps the page load faster in some cases and is usually a good design decision.
4. When the name of a typeface consists of more than one word, it's a best practice to enclose the typeface's name in quotes, like so:

```
h1 {
  font-family: "Courier New";
}
```

You can find a reference of web safe fonts here.

font-family: "Georgia";

font-family: Helvetica;

4/10

Font Size

Changing the typeface isn't the only way to customize text. Often times, different sections of a web page are highlighted by modifying the *font size*.

To change the size of text on your web page, you can use the `font-size` property.

```
p {
```

```
  font-size: 18px;
}
```

In the example above, the `font-size` of all paragraphs was set to `18px`. `px` means pixels and is a way to measure font size.

`font-size 18px`

5/10

Font Weight

In CSS, the `font-weight` property controls how bold or thin text appears.

```
p {
  font-weight: bold;
}
```

In the example above, all paragraphs on the web page would appear bolded.

The `font-weight` property has a another value: `normal`. Why does it exist?

If we wanted *all* text on a web page to appear bolded, we could select all text elements and change their font weight to `bold`. If a certain section of text was required to appear normal, however, we could set the font weight of that particular element to `normal`, essentially shutting off bold for that element.

`font-weight bold`

6/10

Text Align

No matter how much styling is applied to text (typeface, size, weight, etc.), text always appears on the left side of the browser.

To align text we can use the `text-align` property. The `text-align` property will align text to the element that holds it, otherwise known as its *parent*.

```
h1 {
  text-align: right;
}
```

The `text-align` property can be set to one of the following three values:

1. `left` — aligns text to the left hand side of its parent element, which in this case is the browser.
2. `center` — centers text inside of its parent element.
3. `right` — aligns text to the right hand side of its parent element.

text-align: center;

7/10

Color

Before discussing the specifics of color, it's important to make two distinctions about color. Color can affect the following design aspects:

- Foreground color
- Background color

Foreground color is the color that an element appears in. For example, when a heading is styled to appear green, the *foreground color* of the heading has been styled.

Conversely, when a heading is styled so that its background appears yellow, the *background color* of the heading has been styled.

In CSS, these two design aspects can be styled with the following two properties:

- `color`: this property styles an element's foreground color
- `background-color`: this property styles an element's background color

```
h1 {
  color: red;
  background-color: blue;
}
```

In the example above, the text of the heading will appear in red, and the background of the heading will appear blue.

background-color: white;

color: black

8/10

Opacity

Opacity is the measure of how transparent an element is. It's measured from 0 to 1, with 1 representing 100%, or fully visible and opaque, and 0 representing 0%, or fully invisible.

Opacity can be used to make elements fade into others for a nice overlay effect. To adjust the opacity of an element, the syntax looks like this:

```
.overlay {
  opacity: 0.5;
}
```

In the example above, the `.overlay` element would be 50% visible, letting whatever is positioned behind it show through.

```
.caption {
  display: block;
  font-family: 'Playfair Display', serif;
  font-size: 14px;
```

```
  font-style: italic;
  line-height: 14px;
  margin-left: 20px;
  padding: 10px;
  position: relative;
  top: 80%;
  width: 60%;
  background-color: white;
  color: black;
  opacity: 0.75;
}
```

9/10

Background Image

CSS has the ability to change the background of an element. One option is to make the background of an element an image. This is done through the CSS property `background-image`. Its syntax looks like this:

```
.main-banner {
  background-image: url("https://www.example.com/image.jpg");
}
```

1. The `background-image` property will set the element's background to display an image.
2. The value provided to `background-image` is a `url`. The `url` should be a url to an image. The `url` can be a file within your project, or it can be a link to an external site. To link to an image inside an existing project, you must provide a relative file path. If there was an image folder in the project, with an image named `mountains.jpg`, the relative file path would look like:

```
.main-banner {
  background-image: url("images/mountains.jpg");
}
```

background-image: url("https://s3.amazonaws.com/codecademy-content/courses/freelance-1/unit-2/soccer.jpeg");

day 3

Today you will start with a quiz on the CSS visual rules you learned yesterday.
You will then use your knowledge of CSS selectors to finish styling a recipe website on Codecademy.
Finally, you will learn how to incorporate CSS in web pages on your local computer.

Quiz css visual rules

Project: Healthy Recipes

not exactly sure how to start:

```
.header {
  height: 150px;
}
```

on html page

```
 <p id="cook-time" #font-weight: bold;>Total time: 45 minutes</p>
```

I did this but it may be wrong:

```
img {
    height: 150px;
}
.description h5 {
  font-size: 20px;
  font-weight: bold;
}
.ingredients {
list-style: square;
}
p, .time {
color: gray;
  font-family: Helvetica;
}
.citation, .external-link {
color: SeaGreen;
}
```

==========

Create Your First HTML/CSS Project

Reading Time: About 5 minutes

Mark Complete

Requirements:

- Text editor
- An Internet browser

INTRODUCTION

In this article, we'll cover how to take the skills you've learned on Codecademy and use them to create a basic web page built entirely from the tools you have on your computer.

STEP 1: CREATE A FOLDER STRUCTURE FOR YOUR WEB PAGE

Let's create a folder structure to support your web page. A well-designed folder structure will help you quickly navigate to the HTML or CSS files that contain your code.

First, open Finder (in Mac) or Explorer (in Windows). Next, create a folder (also known as a directory) called **DevProject**. This folder will contain all of the files for your HTML and CSS project.

Open the **DevProject** folder. Inside, create the following items:

1. A new file called **index.html** (use your preferred text editor)
2. A new folder called **resources**

The **index.html** file will contain the HTML code for your web page, while the **resources** folder will contain all of the necessary resources needed by the HTML files (CSS files, images, etc.).

Next, open the newly created **resources** folder. Inside of this folder, create the following:

1. An additional folder named **css**

The **css** folder will contain the CSS files needed to style your web page.

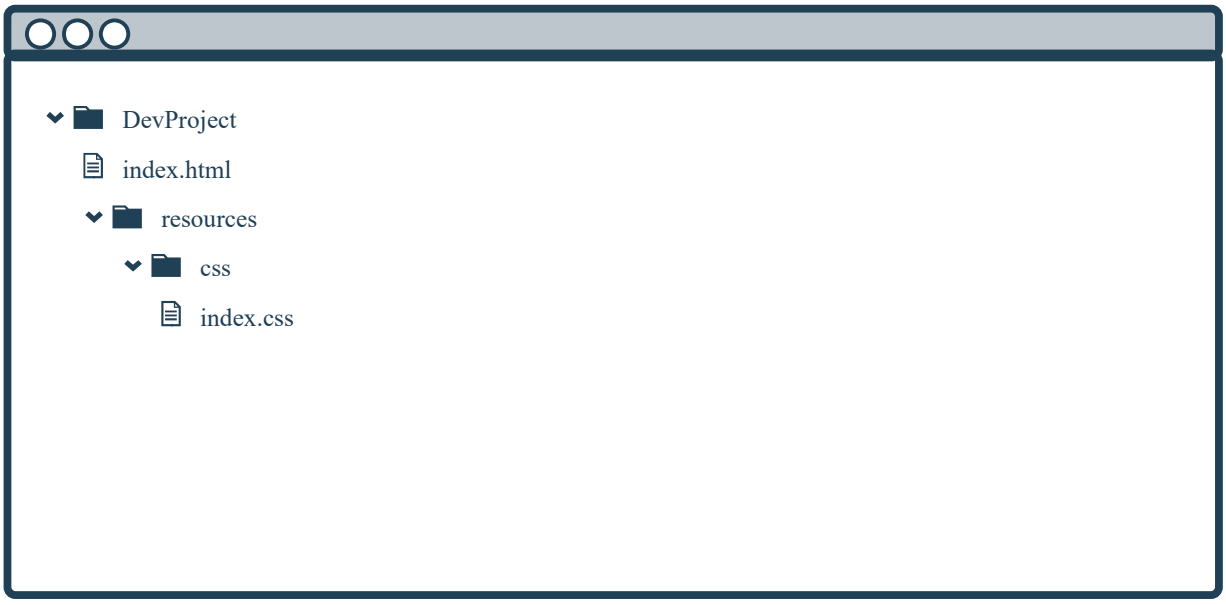
Finally, open the **css** folder you just created. Inside of this folder, create the following:

1. A new file named **index.css** (use your preferred text editor)

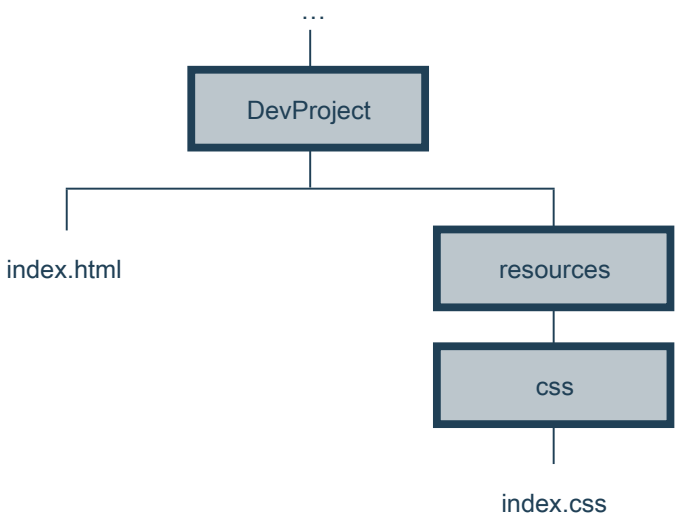
The **index.css** file will contain all of the CSS styling rules that will be applied to your web page.

This overall folder structure will help support your workflow as you add files or resources. At a high-level, here's what it should look like:

File Manager
Graphical Interface



Filesystem Tree



STEP 2: ADD CONTENT TO YOUR WEB PAGE

Great! With your folder structure, HTML, and CSS files all in the right place, we can add content to the web page.

First, open the **index.html** file in your preferred text editor. Next, add the required boilerplate HTML code:

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

==

After you add the boilerplate HTML code, feel free to also add the following items:

1.Your own title between the `<title>` tags

2.Your own content within the `<body>` tags. If you need some quick, pre-written content, feel free to use the following and modify as you wish:

```
<h1>FirstName LastName</h1>
<p>I'm learning to code on Codecademy! My goals include:</p>
<ul>
  <li>Learn to code</li>
  <li>Start a company</li>
  <li>Help the world</li>
</ul>
<p>This is one of my favorite websites:
  <a href="https://www.google.com/" target="_blank">Google</a>
</p>
```

Finally, open the **index.css** file in your preferred text editor. Add the following pre-written CSS rules to the file (feel free to modify as you wish):

```
{
  font-family: Helvetica, Arial, sans-serif;
}
h1 {
  color: SeaGreen; }
p,
li {
  font-size: 18px;
}
a {
  text-decoration: none;
}
```

Be sure to save your changes for both files!

STEP 3: LINK YOUR HTML FILE AND CSS FILE

As it turns out, the HTML content you added will not be styled by the CSS rules unless **index.html** and **index.css** are linked together. In the `<head>` section, link the stylesheet to the HTML file.

```
<link href="./resources/css/index.css" type="text/css" rel="stylesheet">
```

You might be wondering why the `href` attribute is set to `./resources/css/index.css`. This is because you must specify exactly where the **index.css** file lives within your folder(s) *relative* to where **index.html** lives (otherwise, the two files won't link).

Again, make sure to save your changes!

STEP 4: VIEW YOUR PROJECT

Great work - let's take a look at your web page in the browser.

Open your preferred web browser. In the menu bar, click on "File" and then click on "Open File..." (or equivalent). Navigate to your **index.html** file and then click "Open" (or equivalent). The browser should load your web page. What do you see?

At this point, feel free to make changes to your HTML or CSS code. Keep in mind that in order to view any *new* changes you make, you'll have to refresh your browser.

REVIEW

Congrats! In this short time, you learned how to take what you've been learning on Codecademy and apply it using the tools you have available on your own personal computer. You successfully learned how to:

1.Create a folder structure to support both your workflow and your web page

2.Add HTML content and CSS styling to respective files

3.Link the HTML and CSS files together

4.View your web page in a browser (and refresh the browser to view new changes)

In general, the four points above are a strong starting point for your own developer workflow. As you learn more, you may modify the workflow to fit your specific needs. At this point, feel free to modify the content of the web page or create an entirely new project. Happy coding!

OK, this is what I wrote in the index.html file:

```
<!DOCTYPE html>
<html>
  <head>
    <link href="./resources/css/index.css" type="text/css" rel="stylesheet">
    <title>The CSS Day Three Project</title>
  </head>
  <body>
    <h1>Drop Dead Fred </h1>
<p>I'm learning to code on Codecademy! My goals include:</p>
<ul>
  <li>Learn to code</li>
  <li>Start a company</li>
  <li>Help the world</li>
</ul>
<p>This is one of my favorite websites:
  <a href="https://www.google.com/" target="_blank">Google</a>
</p>
  </body>
```

</html>

and I wrote this inside the index.css file

```
{
  font-family: Helvetica, Arial, sans-serif;
}
h1 {
  color: SeaGreen; }
p,
li {
  font-size: 18px;
  color: blue;
}
a {
  text-decoration: none;
}
```

Day 4

Reading Time: About 1 minute

Mark Complete

Today you will use your knowledge of CSS visual rules to style a portfolio site on

Codecademy.

You will then learn about an incredibly powerful developer tool called Chrome

DevTools, which will allow you to debug your CSS and even look at the CSS of

every website on the Internet!

====

Olivia Woodruff Portfolio
Olivia Woodruff, a fictional software developer, has built a portfolio using HTML. It's up to you to customize the portfolio to make it more visually appealing. Use your knowledge of CSS visual rules to transform Olivia's portfolio.

This is the index.html that I wrote:

```
<!DOCTYPE html><html><head>
  <title>Olivia Woodruff Webmaster</title>
  <link href="index.css" type="text/css" rel="stylesheet"/>
</head>

<body>
  <div class="header">
    <img src="https://s3.amazonaws.com/codecademy-content/courses/freelance-
1/unit-2/travel.jpeg"/>
    <h1>Olivia Woodruff</h1>
    <h2>WEB MASTER</h2>
    <p class="title">Building Competitive Websites</p>
    <h3>Just some of the necessary skills:
<ul>
  <li>HTML - this is the basic building block of webpages</li>
  <li>CSS - cascading style sheets dictate the look of a site</li>
   <li>JavaScript, Ajax, PHP, ASP, Java, Perl, C++</li>
   <li>MySQL, SQL Database: web gathered data storage</li>
   <li>Project management: insures the finished project matches the stake holders
vision.</li>
  <li>Customer communication: vital part of project management</li>
  <li>Documentation: technical concepts and code that is being custom created</li>
</ul>
</h3>
<h2>Projects</h2>
  <p class="title">Web Development projects</p>
  <ul>
    <li>Coffee Brur</li>
    <li>Taco Finder</li>
    <li>CSS Selector Finder</li>
    <li>HTML Formatter</li>
  </ul>

  <p class="title">Design projects</p>
  <ul>
    <li>Yum Yum Fudge Inc.</li>
    <li>University of Marimont Dance Marathon</li>
  </ul>
  <h2>Contact</h2>
  <p>Find me on Twitter, Dribbble, and GitHub.</p>
  <h6>© Copyright. All Rights Reserved.</h6>
```

I wrote a simple min. but this is the index.css code that I wrote:

```
H1 {
  font-family: 'Times New Roman';
  font-size: 30px;
  background: #ffffff;
  color: maroon;
}

H2 {
  font-family: arial,verdana,sans-serif;
  font-size: 20px;
  background: #ffffff;
  color: black;
}
H3 {
  font-family: arial,verdana,sans-serif;
  font-size: 14px;
  background: #ffffff;
  color: black;
  font-weight: normal;
}
P {
  font-family: arial,verdana,sans-serif;
  font-size: 16px;
  background: #ffffff;
  color: navy;
}
```

==========
Copy my code onto Atom with folder WebmasterExample
Then create the pages index.html and index.css
Copy my code into the correct page.
Then open it with your browser.
As I said, my is very simple.

CSS Visual Rules in Chrome Inspector

Reading Time: About 8 minutes

USING CHROME DEVTOOLS FOR CSS VISUAL RULES

Requirements:

•An Internet browser

INTRODUCTION

Browser developer tools allow web developers to quickly collect important information on most websites. These tools are available within most major web browsers, like Chrome, Safari, and Firefox, to name a few. Because Google Chrome is the preferred browser for many professional developers, we'll learn how to use the browser developer tools that Google Chrome provides, known as Chrome DevTools.

STEP 1: ACCESSING DEVTOOLS

The quickest way of accessing DevTools in Chrome is to navigate to any website (like this one) and *right click* (press `Ctrl` and click for a single button mouse) anywhere on the page. Upon doing so, a menu will appear directly beside the area you clicked on. In the menu, select "Inspect." This will automatically launch DevTools within your browser. DevTools will appear as a window on either the bottom or right hand side of your screen. It should look something like this:

You should see the following tabs: `Elements`, `Console`, `Sources`, `Network`, `Timeline`, `Profiles`, `Application`, `Security` and `Audits`.

This rest of this article will focus exclusively on the `Elements` tab.

STEP 2: USING DEVTOOLS TO VIEW CSS STYLES

DevTools can provide you with a lot of information about a website, but it's particularly exceptional at examining a page's HTML elements, along with the CSS styles for those respective elements. Let's try it out!

1. Open an incognito Chrome browser (in the browser's menu, click on "File" then "New Incognito Window"). This will allow you to read this article while completing the following steps.

2. Navigate to Codecademy's homepage (make sure you are logged out).

3. Right click (or `Ctrl` and click simultaneously) on the heading that says "Learn to code interactively, for free."

4. Select "Inspect" in the menu that appears.

5. DevTools should appear at the bottom of your page (it's normal if its appears in another location, as its location can be changed).

6. Click on the "Elements" tab of DevTools (if you're not already on it).

7. In the left pane, notice the interactive DOM (HTML elements) that contains the current content of the web page.

8. Mouse over the HTML code – as you

8. mouse over, notice that DevTools will highlight the corresponding HTML element on the web page.

9. Note that you can expand closed elements by clicking the arrow directly to the left of them.

10. Alternatively, click the "Select element" icon (shown in the image below) in the top-left corner of the console and then click on an element within the web page – this is a much quicker way of accessing a specific element on the web page that you want to inspect.

11. On the right hand side of DevTools, click on the tab named `Styles` (if you're not already on it) — this tab displays *all* of the CSS styles associated with the element highlighted in the left side of DevTools.
12. Scroll down in the `Styles` tab, notice that some CSS styles are crossed out with a horizontal black line.

13. Remember, the `Styles` tab shows *all* styles applied to that element (rules can often be overwritten by more specific rules, which causes the horizontal black line through some CSS rules, denoting that that rule is not being used).
14. To instead view *only* the styles applied to that specific element, click on the `Computed` tab directly next to the `Styles` tab in the right pane. In this pane, you will see only the styles that are being applied to that element, also known as the *computed styles*. (If the `Computed` tab is not appearing for you, your browser may be sized too small. Expand the width of the browser until it appears.)

STEP 3: MODIFYING CSS STYLES WITH DEVTOOLS

DevTools is also useful for modifying *existing* CSS rules and previewing those changes directly on the page you're viewing.

To try it out, click again on the `Styles` tab in the right pane of DevTools (feel free to use the Codecademy website again). Scroll down to a CSS rule (one that is *not*crossed out with a black line), click on the value of any applied CSS rule, change the value, and press "Enter" (or "return") on your . You should see the change automatically update on the page.

There are a few things to keep in mind when using DevTools to modify a web page:

1.When you modify or change a CSS rule, you may be affecting more than one element.

2.DevTools provides easy-to-use tools when you modify certain CSS rules. (For example, when modifying color values, DevTools will provide you with a color picker to help you select a color.)

3.DevTools is only a *sandbox* tool, meaning that any changes you make to the web page will *not be saved*, so make sure to write down any changes you'd like to make when using DevTools for your own web page!

STEP 4: ADD CSS STYLES WITH DEVTOOLS

In addition to modifying existing CSS rules, you can add *new* CSS rules as well. Let's continue using DevTools on the Codecademy homepage.

1.Notice that the homepage has a large heading in the middle: "Learn to code interactively, for free."

2.Right click on the heading and click "Inspect" in the menu that appears. DevTools will highlight the corresponding HTML element in its left pane.

3.Take a look at the `Styles` tab and click on the `+` icon in the top-right corner of the right pane – notice that this creates a new, empty CSS rule for that element.

4.Within the element's selector, click and add a new CSS declaration. The following is an example:

```
background-color: red;
```

You should see the background color of the heading change to red. You can also continue to add your own CSS styling as you wish.

In the future, feel free to try this on your own website(s) as you build them from the ground up. Building with DevTools can result in a more efficient workflow, as it can help you avoid repeatedly saving and viewing changes manually.

STEP 5: MODIFY HTML WITH DEVTOOLS

DevTools also lets you directly *modify* the HTML content of a web page. Let's try this out one more time on the Codecademy homepage.

1.Again, right click on the heading in the center of the homepage ("Learn to code interactively, for free") and click "Inspect" in the menu that appears.

2.DevTools will automatically highlight the HTML code in the left pane associated with the content that you inspected on the web page.

3.Double click on the heading text between the `<h1>` opening and closing tags in the left pane.

4.Change the heading to say something else, like your name, or "Codecademy", and press Enter.

At this point, you should see the web page's heading change and say something entirely different.

You can also add HTML of your own as well. Let's add an `<h2>` element directly below the heading you just modified.

1.Right click on the `<h1>` element you just modified, a menu should appear. Click on "Edit as HTML." (You can also delete elements using this menu.)

2.A large text field should appear. Directly edit the HTML by adding

an `<h2>`element below the `<h1>` with the text of your choice.

3.To complete/view your changes, click on any other element in the left window

pane or press `Command` and `Enter` at the same time (on a Mac keyboard).

What happens to the web page? Remember, these are sandboxed changes, so your changes will not be saved, nor permanently affect the website you are applying changes to.

REVIEW

The Chrome web browser provides you with robust web developer tools known as DevTools. With DevTools, you can view a web page's existing DOM elements and associated styles, as well as modify and preview changes you make to the web page, resulting in an efficient workflow. If you're interested in learning more about DevTools, visit the official documentation at

https://developer.chrome.com/devtools

(come back later and go thru the dev tools)

Day 5

come back later and go thru the dev tools

Beach Paradise

A travel agency is in need of a website that showcases the top beach destinations preferred by their customers. Use your HTML and CSS skills to create the website and style it according to their needs.

Beach Paradise

In this project, you'll follow step-by-step instructions to build a simple travel website using the tools on your own computer.

At a high-level, the website will include the following:

1. A heading and a subheading.

2. An ordered list of four beach locations.

3. Four sections that describe a different beach, each using:

- An image
- A smaller subheading
- A brief description (paragraph) of that beach

You'll also style the website to make it more visually appealing.

As you build, we will provide you with the resources you'll need to complete the website, including:

- The images
- The colors for the website

To prepare for this project, we recommend that you read the following resources (if you haven't already done so):

1. Create Your First HTML/CSS Project

[CSS Visual Rules in Chrome Inspector]

The browser to the right shows the website you will build. Check your progress as your

complete the tasks to ensure you're on track.

```
h1, h2, h3, p, li {
  font-family: Helvetica, sans-serif;
}

h1, h2 {
  color: DarkGreen;
}

h3 {
  color: SeaGreen;
}
```

```
<!DOCTYPE
html><html><head
>
                <title>Beach Paradise</title>
                <link href=".\Resources\css\style.css" type="text/css" rel="stylesheet"/>
                </head>
                <body>

                <h1>Beach Paradise</h1>
                <h2>The Top Beach Destinations in the World</h2>
                <div class="beaches">
                <ol>
                <li><a href="#bahia">Bay of Bahia</a></li>
                <li><a href="#selange">Selange Beach</a></li>
                <li><a href="#hunt">Hunt Lagoon</a></li>
                <li><a href="#moat">Moat Lagoon</a></li>
                </ol>
                </div>
                <div id="bahia">
                <img src="https://s3.amazonaws.com/codecademy-content/courses/freelance-1/unit-2/bahia.jpe
                bay"/>
```

```
<h3>Bay of Bahia</h3>
<p>Nestled on the northern end of the Isle of Caludge, this beach is cited as the most popular g
vacationers during peak summer months. Popular activities include scuba diving, snorkeling, ar
watching.</p>
</div>

<div id="selange">
<img src="https://s3.amazonaws.com/codecademy-content/courses/freelance-1/unit-2/selange.j
alt="Selange beach"/>
<h3>Selange Beach</h3>
<p>Selange Beach is known for its temperate sand. Made up of ground coral due to thousands
tectonic activity, the sand is always cool to the touch, despite the heat. The western portion incl
flora and fauna and many pre-Columbian ruins.</p>
</div>

<div id="hunt">
<img src="https://s3.amazonaws.com/codecademy-content/courses/freelance-1/unit-2/hunt.jpeg
lagoon"/>
<h3>Hunt Lagoon</h3>
<p>Known primarily for it&#39;s hidden location on Kamar Island (which includes swimming
underground cave), Hunt Lagoon offers a peaceful respite from the tourist-laden locations foun
Kamar. If possible, make a visit at night – it&#39;s bioluminescent lagoon is like viewing firew
water!</p>
</div>
<br/>

<div id="moat">
<img src="https://s3.amazonaws.com/codecademy-content/courses/freelance-1/unit-2/moat.jpe
lagoon"/>
<h3>Moat Lagoon</h3>
<p>Last, but certainly not least, is Moat Lagoon. This is one of the most difficult beach locatio
The lagoon is located inside of a prehistoric cenote, or underground aquifer. Accessing it involv
the moat surrounding the lagoon (hence the name) with locals by kayak. Once there, marvel at t
phenomenal (and endangered) quetzal bird and its golden feathers.</p>
</div>
```

Day 6

Today you will create a fully-functional website with HTML and CSS. You may get stuck or become frustrated at certain points, and you may need to review material from prior lessons. However, don't give up! Pushing yourself to overcome difficulties is one of the most important parts of your learning experience.

Once you finish this project, you'll be another step closer to being able to call yourself a website developer!

Reviewable Project

Dasmoto's Arts & Crafts (To stay on track, submit by **OCT 18)**

An overseas art store has contacted you to design a website for the items they sell. They've provided you with all of the documents needed for you to start. Showcase your HTML and CSS skills by creating a website for them.

Dasmoto's Arts & Crafts

In this project, you'll build a simple website for a fictional arts and crafts store using the tools on your own computer. This project will provide you with less guidance than previous projects.

You should expect to use the Internet, Codecademy, and other resources when you encounter a problem that you cannot easily solve.

At a high-level, this project will require the following:

1. A folder structure that makes sense for the project
2. An HTML file
3. A CSS file

To successfully complete the project, you'll require the following images:

1. Image 1
2. Image 2
3. Image 3
4. Image 4

The rest of the page's styling (font sizes, colors, etc.) is outlined in the following design spec,

which is a standard document you'd expect to receive as a freelance web developer:

- Spec

Use the website to the right and the resources above to guide you through the project.

==========

```
<!DOCTYPE html>
<html>

<head>
  <title>Dasmoto's Arts and Crafts</title>
  <link href="./style.css" type="text/css" rel="stylesheet"/>
</head>

<body>
  <h1>Dasmoto's Arts and Crafts</h1>

  <div class="section">
    <h2 id="brushes">Brushes</h2>
    <img src="./pic\shacksaw.jpeg"/>
    <h3>Hacksaw Brushes</h3>
    <p>Made of the highest quality oak, Hacksaw brushes are known for their
weight and ability to hold paint in large amounts. Available in different sizes. <span
class="price">Starting at $3.00 / brush.</span></p>
  </div>

  <div
   class="section">
     <h2 id="frames">Frames</h2>
     <img src="./pics\frames.jpeg"/>
     <h3>Art Frames (assorted)</h3>
     <p>Assorted frames made of different material, including MDF, birchwood, and
PDE. Select frames can be sanded and painted according to your needs. <span
class="price">Starting at $2.00 / frame.</span></p>
  </div>
```

```
  <div class="section">
    <h2 id="paint">Paint</h2>
    <img src="./pics\finnish.jpeg"/>
    <h3>Clean Finnish Paint</h3>
    <p>Imported paint from Finland. Over 256 colors available in-store, varying in quantity (1 oz. to 8 oz.). Clean Finnish paint microbinds to canvas, increasing the finish and longevity of any artwork. <span class="prices">Starting at $5.00 / tube.</span>
    </p>
  </div>

</body>
```

submitted oct 17
https://github.com/rxr2090/prj-rev-bwfs-dasmoto

==========