To recap the previous video, your goal is to make three elements:

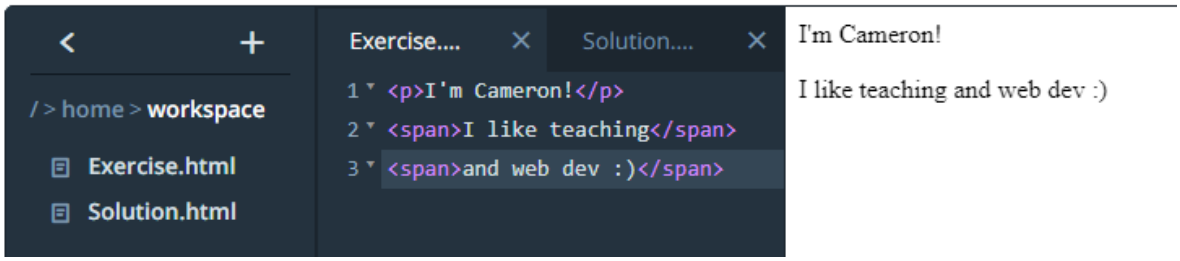
- One paragraph
- Two spans

Make sure you put some text content in every element!

Here's an example element:

```
<tag>content</tag>
```

Feel free to use the workspace below to create your elements!



The HTML Document

Every HTML document you create or load is derived from this basic format:

```
<!DOCTYPE html>
<html>
    <head>
        Meta information goes here!
    </head>
    <body>
        Content goes here!
    </body>
</html>
```

You can think of it as a template. And, following this template will help ensure that the page is displayed as the developer (you) intended. It not only says **what** should be displayed, but also includes relevant information that tells the browser **how** to display it.

This template can be broken down into 3 parts:

1. `DOCTYPE`: Describes the type of HTML. While there are technically different types, for 99.999% of the HTML you'll write, you'll likely be fine with `<!DOCTYPE html>`.
2. `<head>`: Describes meta information about the site, such as the title, and provides links to scripts and stylesheets the site needs to render and behave correctly.
3. `<body>`: Describes the actual content of the site that users will see.

Omitting some of this information doesn't necessarily mean that the page won't be displayed. In fact, your browser will assume certain parts of the template exist even if you accidentally leave them out. Take this line of HTML for example:

```
<h1>This is a heading</h1>
```

If you create an HTML file with only this line, open the file in any modern browser, and inspect the page with developer tools, you'll see that certain parts of the basic HTML document format were assumed:

```
<html>
  <head></head>
▼ <body>
    <h1>This is a test</h1> == $0
  </body>
</html>
```

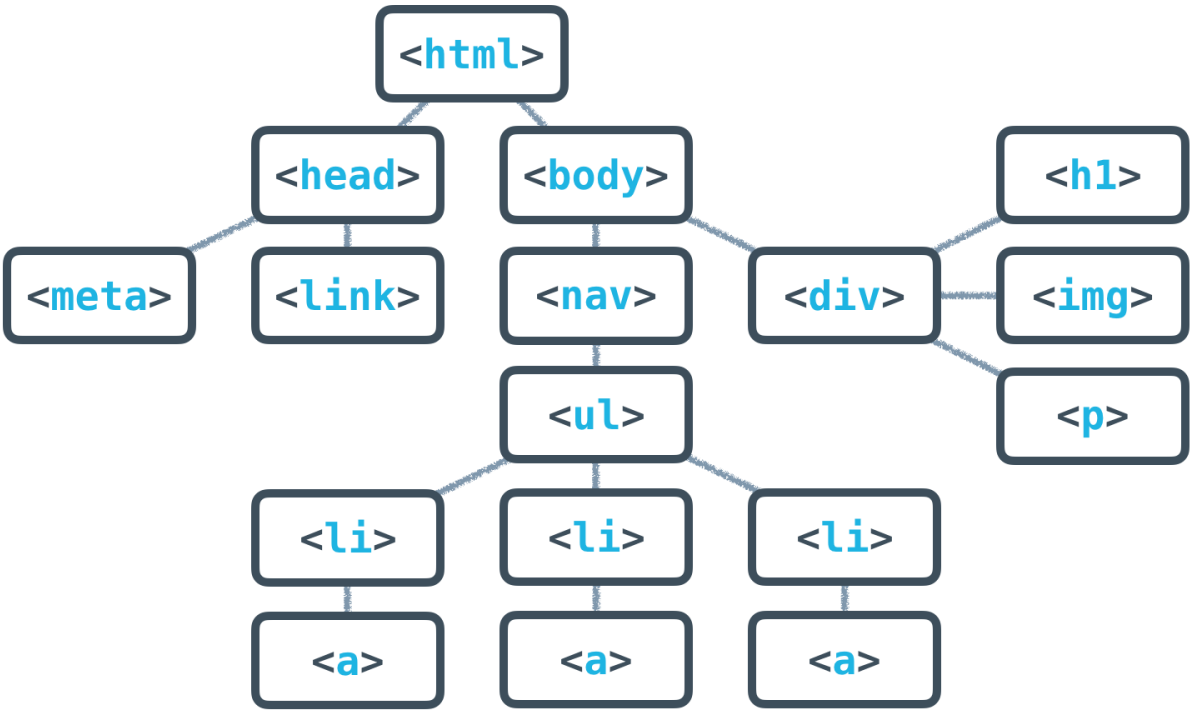
The view from the Elements panel in Developer Tools when you omit all but `<h1>...</h1>`. (Notice that an empty `<head>` has been created for you.)

That being said, this is not guaranteed behavior. Older browsers can be unpredictable, and

Basic HTML Tree Structure

`<head>` and `<body>`

The `<head>` will contain general information and metadata about the page, while the `<body>` will contain the content that will be displayed on the page. Here's an example tree structure for a full HTML document:



Full HTML Tree Structure

All of the HTML syntax that you've learned in this lesson will help you create the **content** of the page, which is always contained inside the `<body>` tags. The `<body>` is always visible.

The `<head>`, on the other hand, is never visible, but the information in it describes the page and links to other files the browser needs to render the website correctly. For instance, the `<head>` is responsible for:

- the document's title (the text that shows up in browser tabs): `<title>About Me</title>`.

- associated CSS files (for style): `<link rel="stylesheet" type="text/css" href="style.css">`.
- associated JavaScript files (multipurpose scripts to change rendering and behavior): `<script src="animations.js"></script>`.
- the charset being used (the text's encoding): `<meta charset="utf-8">`.
- keywords, authors, and descriptions (often useful for SEO): `<meta name="description" content="This is what my website is all about!">`.
- ... and more!

At this point, just focus on these two tags:

- `<title>About Me</title>`
- `<meta charset="utf-8">`

`<meta charset="utf-8">` is pretty standard, and will allow your website to display any Unicode character. (Read more on how UTF-8 works here.) `<title>` will define the title of the document and will be displayed in the tab of the browser window when a user visits the page.

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>This is a title</title>
    </head>
    <body>
        <h1>Hello!</h1>
    </body>
</html>
```

Full HTML Template

HTML Validators

This might seem like a lot to remember, but thankfully, there are tools out there to help you. Much like how the Udacity Feedback Extension tells you when you've met all the requirements for a particular project, HTML validators analyze your website and verify that you're writing valid HTML.

I want you to try one out now!

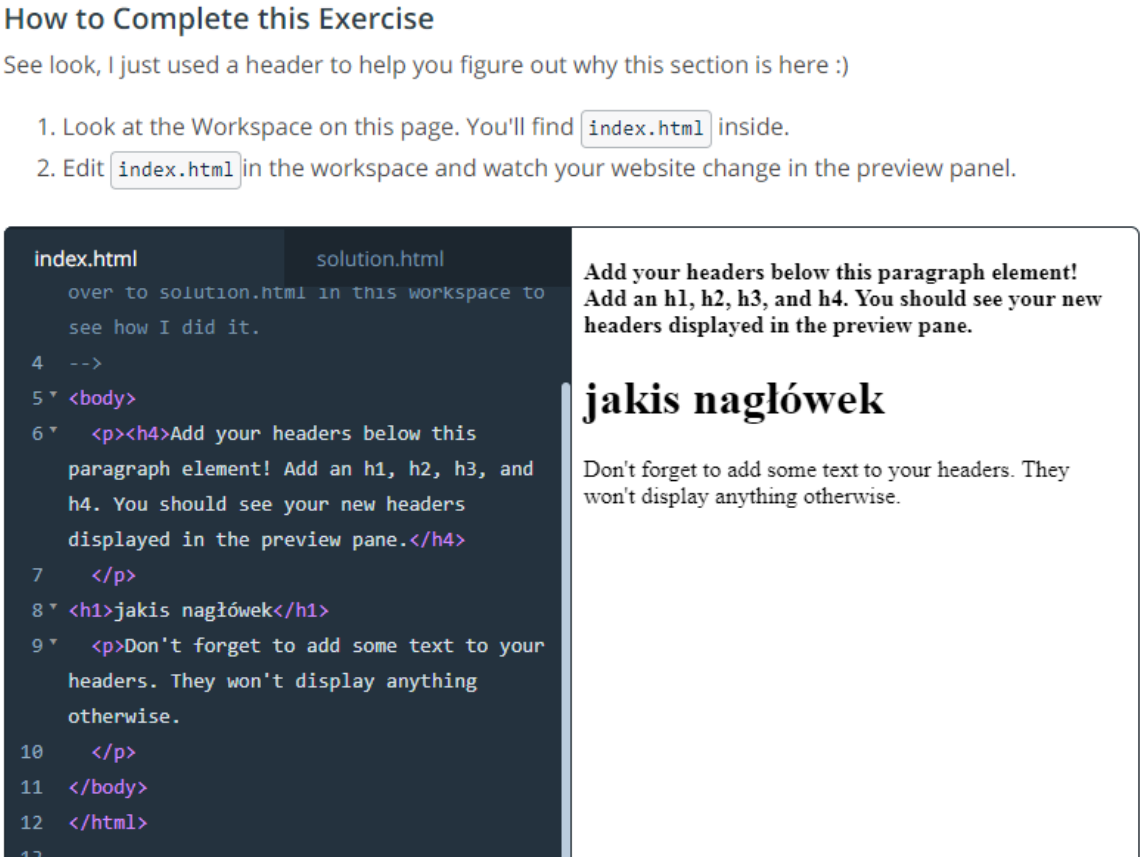
You create button elements the same way you've been creating other elements: `<tag>content</tag>`. In this case, the tag name is `button` and the content that comes between the tags will be the text displayed inside the button. Here's more about buttons on the Mozilla Developer Network (MDN).

Here's my solution:

```
<button>a button!</button>
```

It follows the pattern: `<tag>content</tag>`. Here, the tag is `button` and the content turns into the button's text! If you tried opening this HTML in your browser, you should have made [a button that looks like this]

.



Make a List

Did you know that web developers spend 67.7493% of their time looking things up?

Ok, I made up that number.

But seriously, making sense of documentation and looking up new techniques and technologies is a huge part of any web developer's work. And that's what you're going to do in this exercise.

For this exercise:

- Notice that what you're reading right now is an unordered list :) An unordered list usually displays with bullet points.
- Use the Mozilla Developer Network (MDN) to research **unordered lists** (`<ul>`).
- Make an unordered list with the three web languages:
    - HTML
    - CSS
    - JavaScript

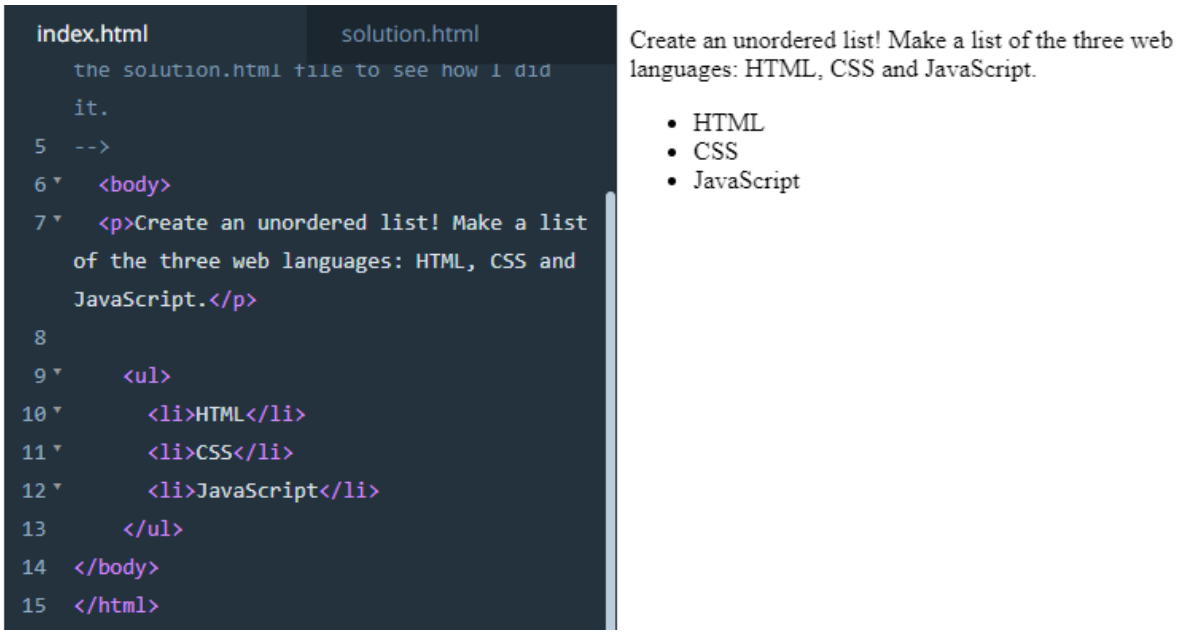
There are two types of lists - ordered lists (with numbers or letters) and unordered lists. Both of them require two kinds of elements. One is going to be nested inside the other (there will be one parent and multiple children).

How to Complete the Exercise

Look at the Workspace on this page. You'll find `index.html` inside. Edit `index.html`in the workspace and watch your website change in the preview panel.

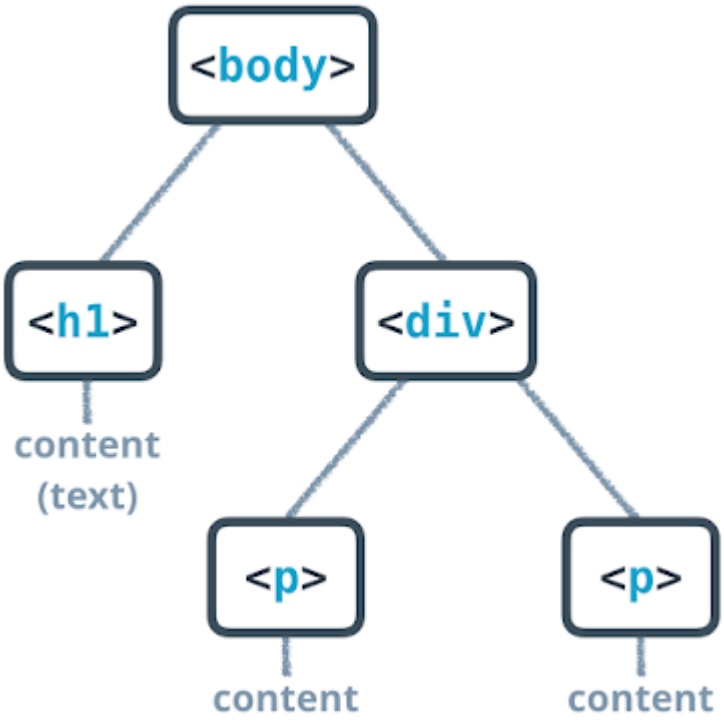
I want you to: (*Look! Another unordered list!*)

- Add an unordered list element with three child elements (one each for HTML, CSS, and JavaScript).



Tree to HTML

You've been learning about the relationship between data trees and HTML. Here's a sample tree for this exercise:



The tree you're going to model

How to Complete this Quiz

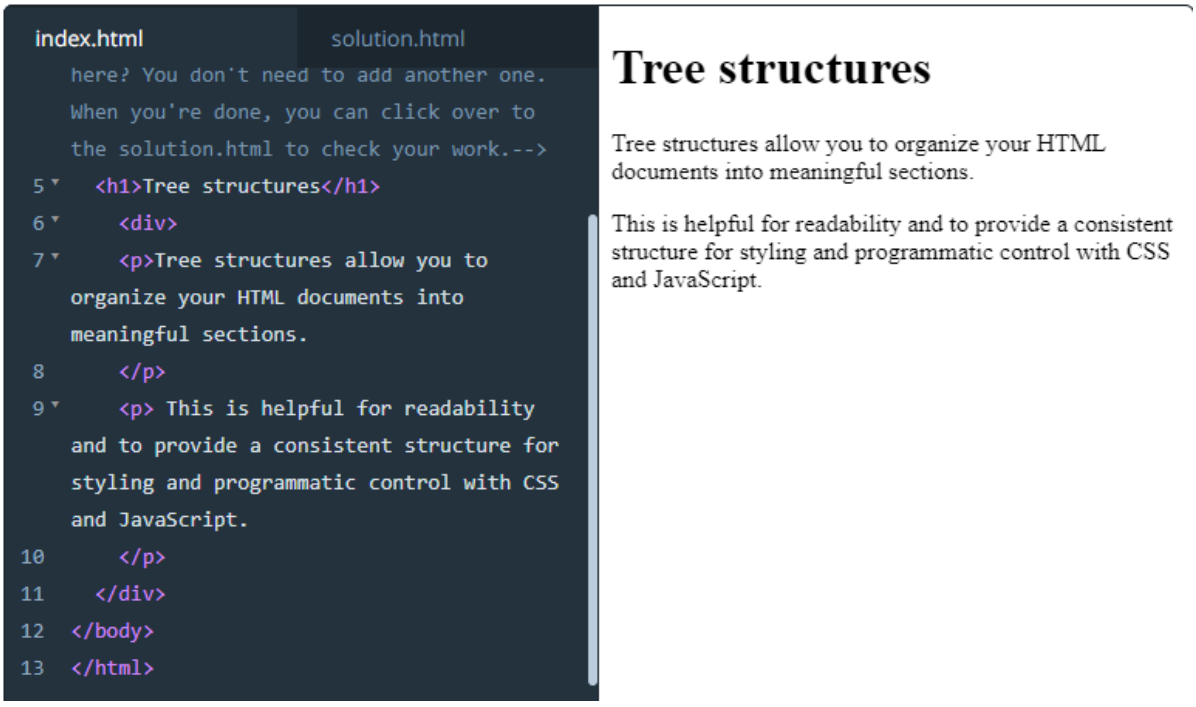
You're going to build the HTML for the image, above. Take a close look at the relationships in the tree. Pay attention to parents, children, and siblings. Notice that some elements have text content.

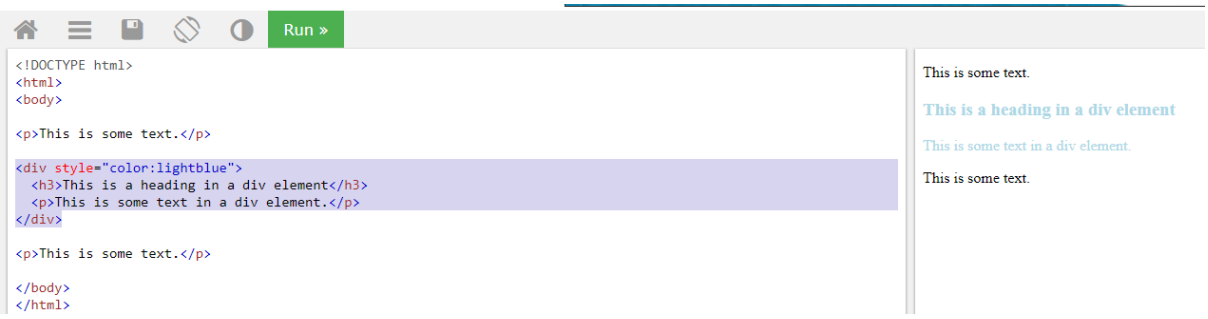
Look at the Workspace, below. You'll find the file `index.html` inside. Edit `index.html` in the workspace and watch your website change in the preview panel.

For this exercise, I want you to:

- Add an `<h1>` tag with some content.

- Add a `<div>` tag.
- Add two paragraph tags as child elements to the`<div>`.
- Add text content to each paragraph element.





Hyperlinks

The power of the web is that pages can lead to other pages. When you click on a link on a web page, it takes you to another page. This link is called a **hyperlink**.

Hyperlinks are created with anchor elements, which generally look like:

```
<a href="https://www.udacity.com">Udacity</a>
```

and render on the page like this: Udacity.

Inside the opening `a` tag there is `href`, which stands for "reference." This is called an **attribute**. Attributes like `href` describe the properties of HTML elements. In this case, the `href` attribute is the target URL that the link will open. The content inside the anchor element is the text that users see displayed on the page.

This is the format that you must use when you make hyperlinks! Note:

- There is a space between `a` and `href`.
- There are no spaces around the `=`.
- The website has two `"` around it.
- There are no spaces between the `href` attribute and the `>` of the opening tag.

MAGES! Images on the web are awesome. Time to make one.

For this exercise, you'll be linking an image in your Workspace. I'll give you the URL of an image and it will be your job to make it display.

An image is made with an `<img>` element. It looks like so:

```
<img src="http://somewebsite.com/image.jpg" alt="short description">
```

The source attribute, `src`, is like the `href` of a link - it is the URL of the image you want to display. For now, your images will need to be hosted online, which means that the URL will need to start with `http://` or `https://`. You'll learn about another way to set image source in the next exercise. The `alt` attribute stands for "alternative description," which is important for people who use screen readers to browse the web. This is text that will show up in lieu of the actual image.

An image element is a little different than the elements you've seen before. Images do not need closing tags! (For the eager, these are called "void elements".)

Here's the MDN reference about images.

How to complete this exercise

Look at the Workspace, below. You'll find the file index.html inside. Edit index.html in the workspace and watch your website change in the preview panel.

For this exercise, you need to create an image tag in the space provided and link it to the URL given in the HTML comment. Here's an example of what the HTML for an image element would look like:

```
<img src="http://somewebsite.com/image.jpg" alt="short description">
```

You'll soon make a website that displays an image that is stored locally on your computer. In order to display a local image, you need to be able to write a **path**.

If there is a file called `index.html` in a directory and there is another directory called `example/` in the same directory, you can access any files in `example/` from `index.html` with the URL (path) `example/filename.html`, e.g. `<a href="example/filename.html">Example Path</a>`.

Paths

A path is a way of describing where a file is stored.

Think of it like this:

Anyone in the world can use the address 1600 Pennsylvania Ave NW, Washington DC, USA 20006 to find the White House. A street address is an absolute path to a location.

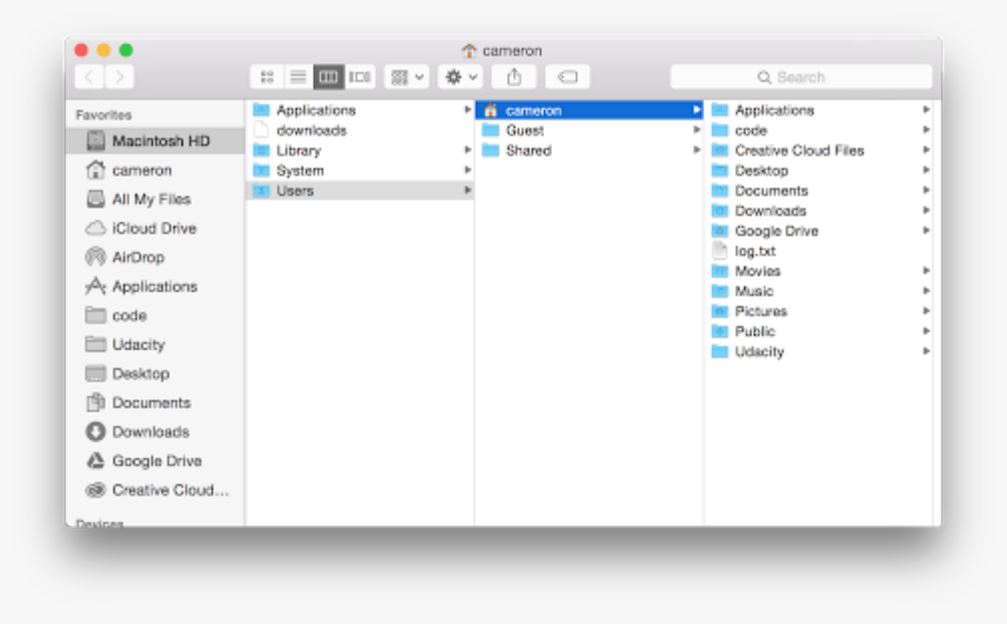
But, if you were at the Eisenhower Executive Office, you could also use the phrase "next door" to find the White House. "Next door" is a relative path because it depends on your current location.

There are essentially two domains for paths that you'll need to consider as a web developer: paths to find files on your computer, **local** files, and paths to find files on other computers, **external** files.

Local Paths

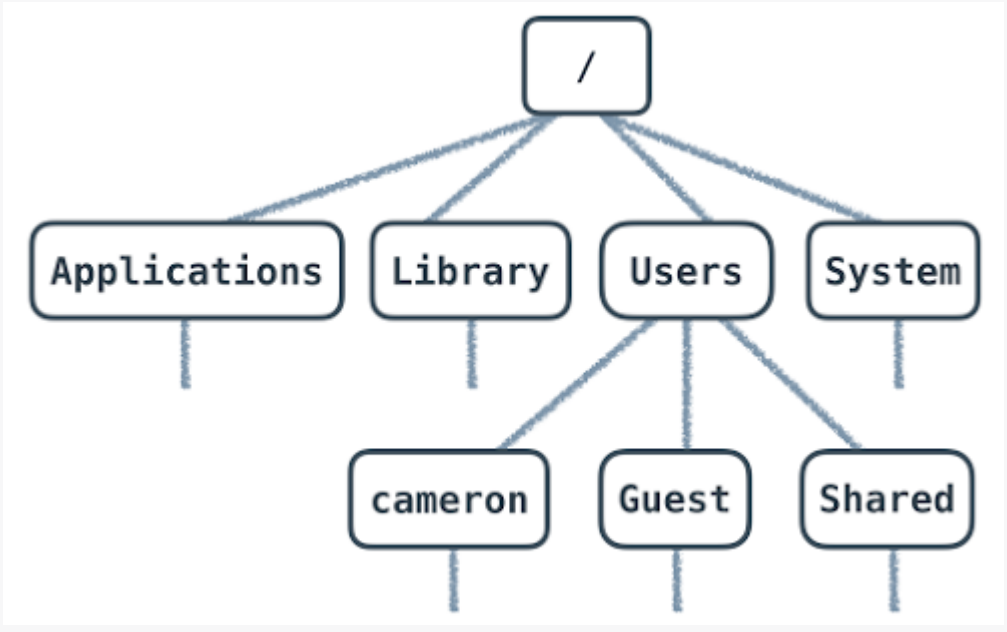
Computers have folders (also called "directories"). Operating systems like Windows, Mac and Linux organize *all* of your files into a tree of directories called a **file system**. There's a top-most directory, often called the **root**, that contains all of the other directories. Within the root, there are files and directories. Within those directories are more files and more directories. And within those directories are even more files and directories, and so on.

Compare this part of the file system on my computer:



Local path directory structure screenshot

to a tree diagram showing the same directory structure:



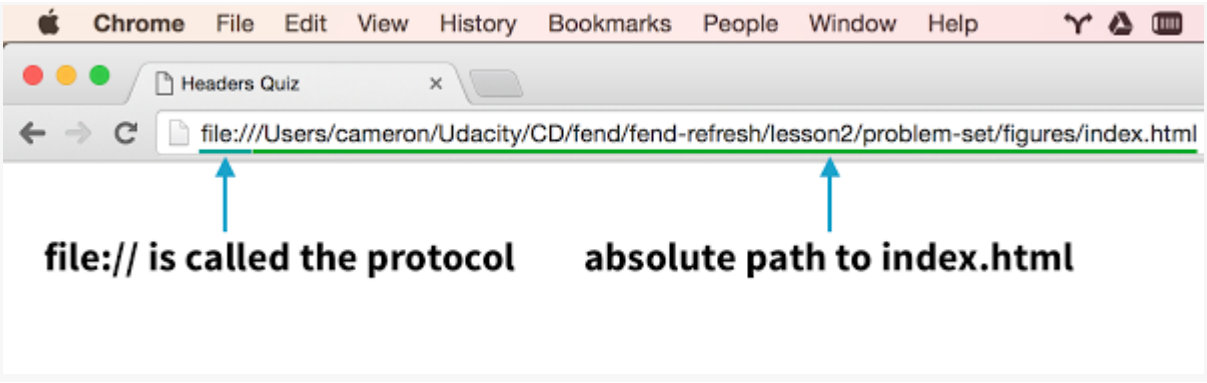
Tree diagram of directory structure

Every file has an address, which we call the "path." An absolute path is written in relation to the computer's root directory. For instance, a file in the Documents folder on a Mac has a path that looks like this:

```
/Users/cameron/Documents/file.txt
```

`file.txt` is stored inside `Documents/`. `Users/`, `cameron/` and `Documents/` are all names of directories. `Documents/` lives inside `cameron/` and `cameron/` lives inside `Users/`. `Users/` is inside the root directory, which is represented by the first `/`. The rest of the `/` are used to separate directories.

When you open an HTML file in your browser, you're seeing the absolute path to the file on your computer.



Absolute Path to index.html

This URL will *only* work for you on your computer. As no one else has your file system, this URL is unique to your computer. If you want other people to be able to access it, then you need an...

External Paths

The process of loading a website from a URL like `https://www.udacity.com` mimics opening an HTML file that you've written and saved to your computer. Every website starts with an HTML file. It just so happens that when you want to visit a website, the HTML file that you want to open lives on a different computer. The computer responsible for giving you a website's files is called a **server**.

Pointing your browser to `https://www.udacity.com` sends a request to Udacity's server for the HTML file (and others) that your computer needs to load the Udacity website. You can think of `udacity.com`as the root path of Udacity's server (computer) that anyone can access (the reality of the situation is actually much more complicated but the general idea is true). Unlike your personal computer (for now!),

Udacity's servers run software that **expose** files to the web, which means that they make them available to anyone who wants them. Servers have an **external path** that anyone can access and is the reason why the web works.

Different websites are just different collections of files. Every website is really just a server (or many servers) with an external address, which we call a URL. Servers store files and send them to computers who request them (the requesting computers are called **clients**).

There are different **protocols** for serving files, the most common of which on the web are HTTP and HTTPS. When you open a file on your own computer, you're using the file protocol. You don't need to know much more about protocols for now, but if you're interested in learning (a lot!) more about HTTP, check out Networking for Web Developers.

Relative Paths

The relative path is similar to the absolute path, but it describes how to find a path to a file from a directory that is not the root directory. Like using the phrase "next door" to tell someone how to find the White House from the Eisenhower Executive Office, a relative path takes advantage of the location of one file to describe where another file can be found.

Relative paths work the same for both local and external paths. Let's break down two examples of absolute paths to see how relative paths work.

External:

```
<a href="http://labs.udacity.com/fend/example/hello-world.html">Hello,
world!</a>
```

Local:

```
<a href="/Users/cameron/Udacity/etc/labs/fend/example/hello-world.html">
Hello, world!</a>
```

`href` is really just a path to a file.

Both examples are links to the same file using absolute paths, but the external example would work for anyone and only my computer can use the link in the local example.

Pay attention to the `fend/example/hello-world.html` portion of both paths - they mean the same thing.

Imagine that you are editing `/Users/cameron/Udacity/etc/labs/fend/test.html`. `test.html` can reference

`hello-world.html` by describing how to get from it's location in `fend/` to `hello-world.html`. The relative path would look like:

```
example/hello-world.html
```

This relative path takes advantage of the fact that `test.html` and `example/` are in the same directory.

But what if I'm editing a file in `/Users/cameron/Udacity/etc/labs/` and I want to write a path to `hello-world.html`? In that case, the relative path would be:

```
fend/example/hello-world.html
```

Now that I'm in `labs/`, not `fend/`, I have to include `fend/` in a relative path to `hello-world.html`.

To finish this off, let's imagine there are two files:

```
http://labs.udacity.com/science/sciences.html
```

and

```
http://labs.udacity.com/science/physics/relativity.html
```

In order to write a relative path from `sciences.html` to `relativity.html`, I only need to include the part of the path that describes how to get from `science/` to `relativity.html`:

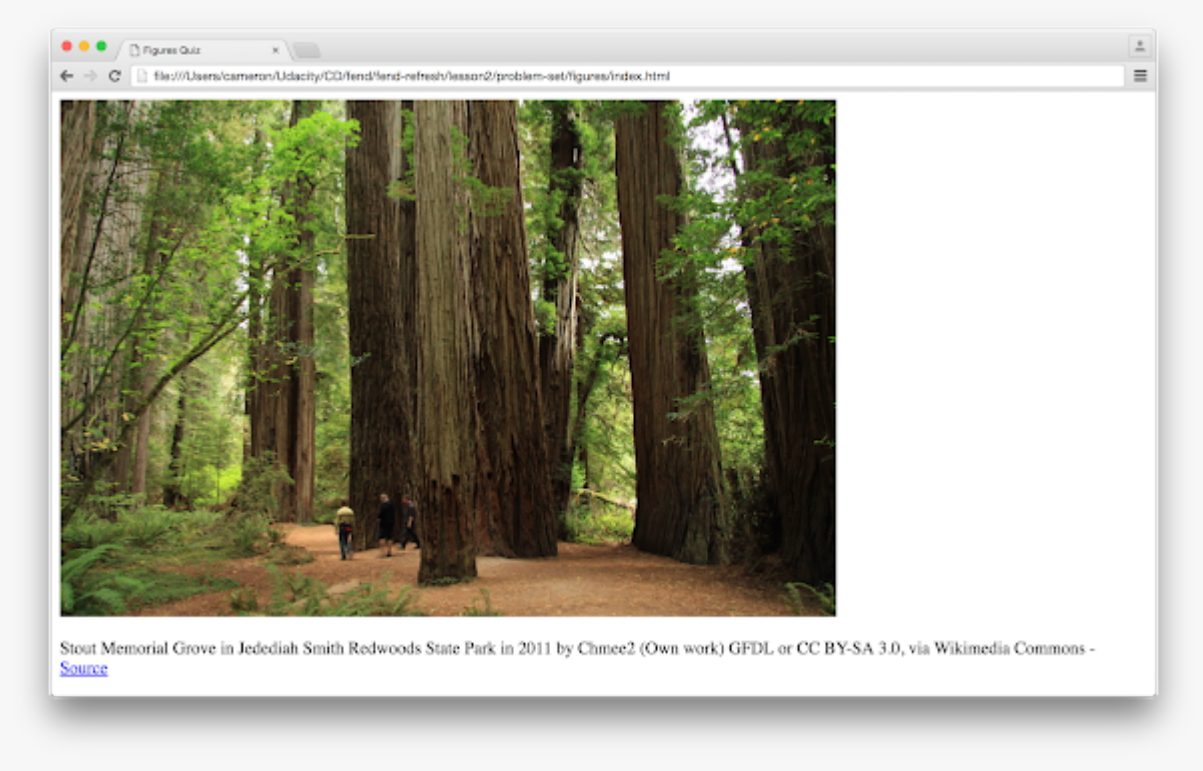
```
<a href="physics/relativity.html">Einstein's Special Relativity</a>
```

And that's it! Now it's time to apply your new skills.

Figures

It's important to respect copyright and attribution on the web. When you use someone else's work, you need to give the author credit. With that in mind, for this exercise you're going to create an image with a caption underneath. I want you to give credit to the image's photographer by attributing them in the image's caption and providing a link to the source material.

This is what it should look like in your browser:



Notice, in the image above, that there is a link to the source in the caption below the image.

How to Complete this Exercise

For this exercise, you need to edit `index.html` so that the caption displays below the image. I want you to use a **relative** path to the image, which is going to be in the same directory as `index.html`. Remember, a relative path points to the file **from the current directory** - it does not start with the root. (Your relative path should *not* start with `/`, `file://`, `C:\`, or `http://`).

I want you to:

1. Create a `<figure>` element

2. Add an `<img>` element whose `src` attribute points to the provided image (look in the Workspace's sidebar for the "redwoods_state_park.jpg" image!)
3. Add an `alt` attribute to the image
4. Add a `<figcaption>` element with the text for the image's caption
5. Change the Markdown code to a real HTML link

Hint: If you get stuck, I recommend checking out the figure element on MDN.

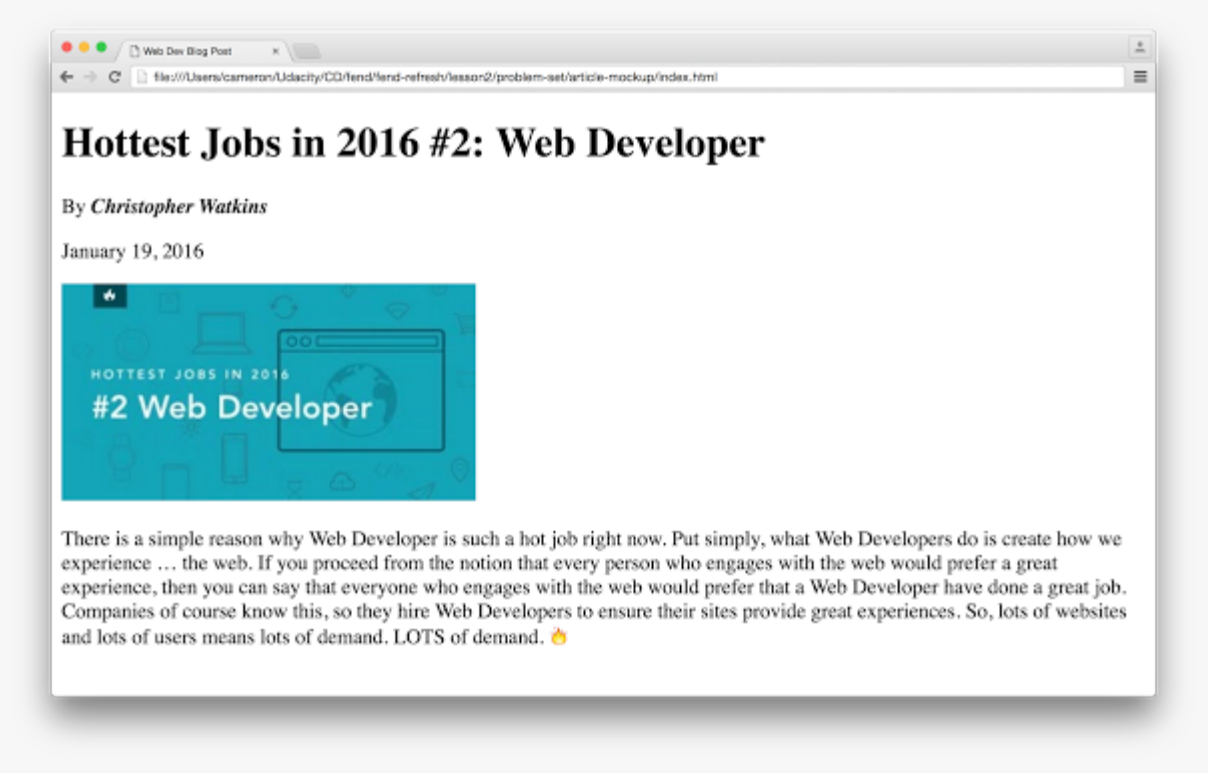
Mockup to Website

It's common for web developers to work with designers who focus on creating user interfaces and user experiences. Designers use software like Adobe Photoshop to mock up - draw - websites. The mockups that they create are usually just images of websites with some annotations and descriptions.

As a web developer, one of the tasks you might be asked to do is take a mockup created by a visual designer and translate it into a live website. I use the word

"translate" because the process of going from mockup to website is similar to the process of translating between natural languages. Just as you can create the same meaning using different words and phrases with a natural language, you can create the same website design using different HTML elements.

I want you to practice the process of going from a mockup to a website now! Here is a website mockup (note: I zoomed in for the screen shot):



There are many ways to turn this mockup into website. The best end result isn't perfect HTML - it's getting your site to look the way it's supposed to. When you're done, compare your website to the mockup. You'll know you've finished this exercise when your site looks the same :)

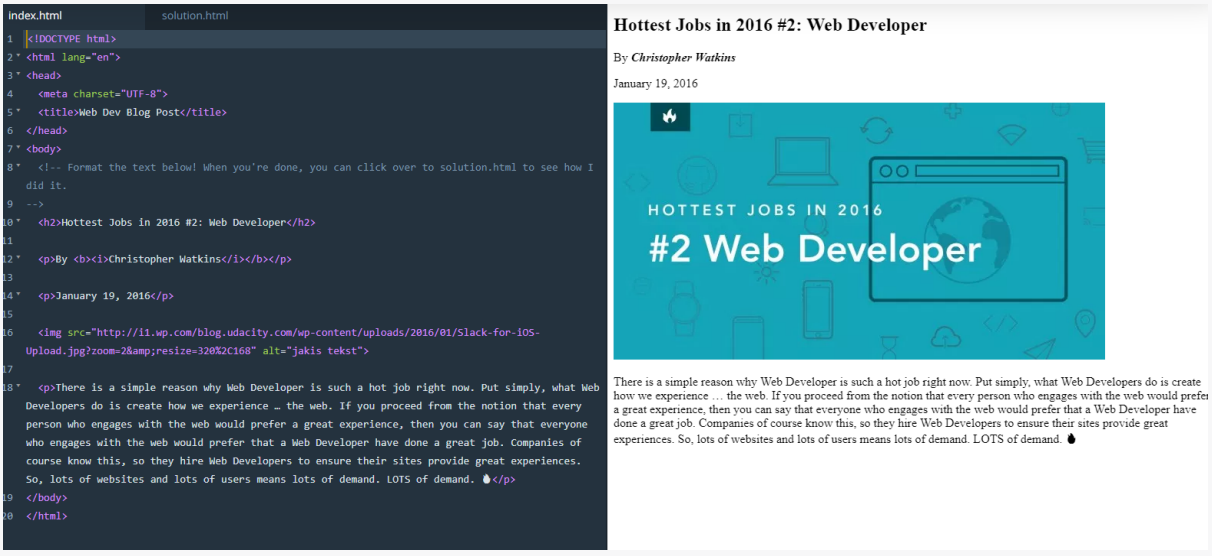
How to Complete this Exercise

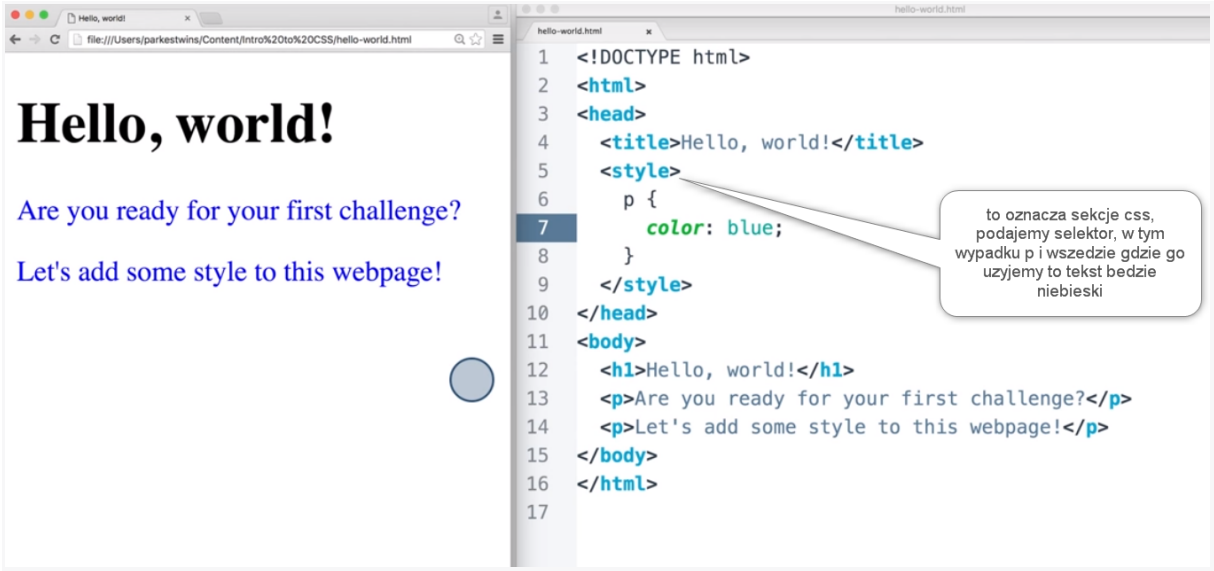
I want you to recreate your website to look identical to the mockup. Start by taking a few minutes to analyze the mockup. What text is changed? What images are there? While doing this, I want you to practice indenting children elements. I'll show you how I indented my HTML in the solution.

This mockup was taken from [this Udacity blog post](#).

I want you to:

1. Position the text content to match the mockup
2. Style the content to match the mockup
3. Add any images that are missing (don't forget to include the `alt` attribute!)

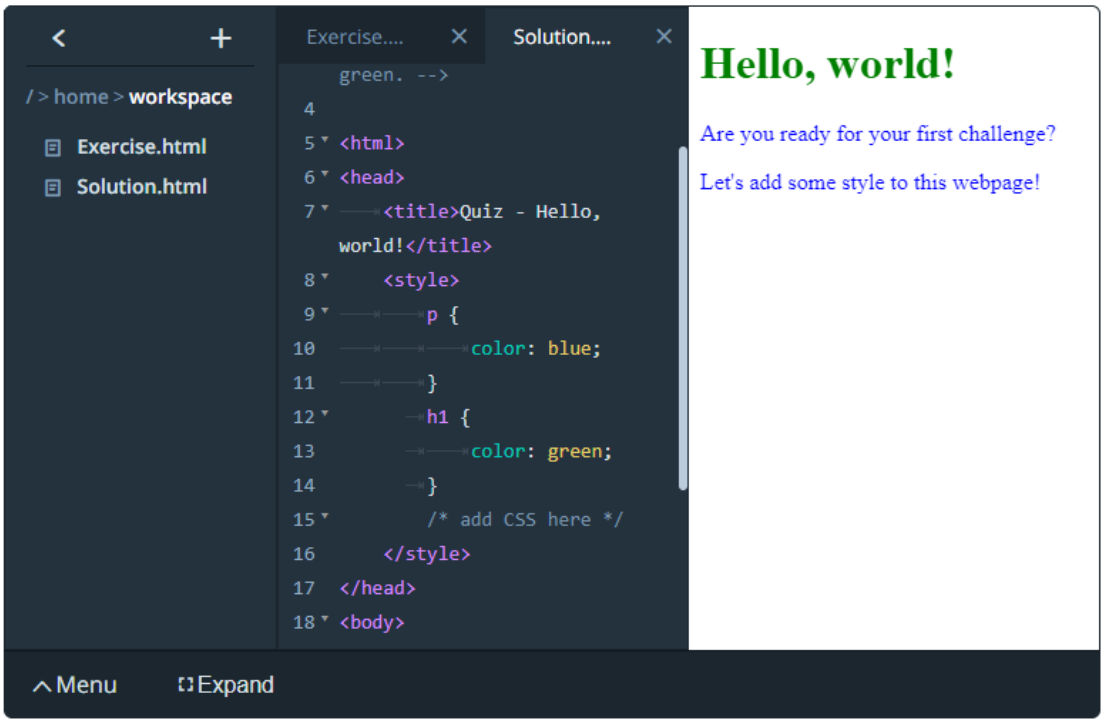



Hello, world!
file:///Users/parkestwins/Content/Intro%20to%20CSS/hello-world.html
Hello, world!
Are you ready for your first challenge?
Let's add some style to this webpage!
hello-world.html
<!DOCTYPE html>
<html>
<head>
<title>Hello, world!</title>
<style>
p {
color: blue;
}
</style>
</head>
<body>
<h1>Hello, world!</h1>
Are you ready for your first challenge?
Let's add some style to this webpage!
</body>
</html>
to oznacza sekcje css, podajemy selektor, w tym wypadku p i wszedzie gdzie go uzyjemy to tekst bedzie niebieski

CSS Syntax

Now, it's your turn! For this exercise, change the color of `Hello, world!` to green. You will need to use a selector to select the header tag, and you'll want to change its `color` property to `green`.

If you need help, use the CSS for the paragraph tags as a guide.



Solution

Feel free to check out the file named `Solution.html` in the workspace above to see how we did it. Here was our thought process:

A comment is a human-readable message inside code. Comments are usually surrounded by or preceded by special characters that instruct computers to completely ignore whatever text is inside the comment. Comments are great because they allow you, the developer, to leave clarifying messages and instructions for other developers (as well as your future self!). Every programming language gives you the ability to write comments.

Comments are also useful when you're testing your code. Rather than deleting potentially useful chunks of code, you can comment them out, getting the same effect without the risk of accidentally losing work!

CSS Comments

You saw a CSS comment in the code from your previous quiz:

```
p {
    color: blue;
}
/* add CSS here */
h1 {
    color: red;
}
```

The line `/* add CSS here */` is a comment. CSS comments are surrounded by an opening `/*` and a closing `*/`. You must use both. The comment made it clear to you where you needed to add your code and it did not affect the style of the page in any way.

HTML Comments

You can write comments in HTML too! Here's how they look.

```
<!-- This is a comment -->
<div class="example">Words, words, words.</div>
```

You must surround your HTML comments with a starting `<!--` and a closing `-->`.

Now, back to CSS.

Tag Selector

```
h1 {
  color: green;
}
```

`class` Attribute Selector

```
.book-summary {
  color: blue;
}
```

`id` attribute selector

```
#site-description {
  color: red;

}
```

Using Selectors

Classes are easily the most useful and versatile type of selectors that you can use. For this exercise, consider the given CSS statement, which uses a class selector for the class `right` and changes the value of the `text-align` property to `right`:

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

QUIZ QUESTION

Which of the following HTML elements match the above CSS statement? Please select all that apply:

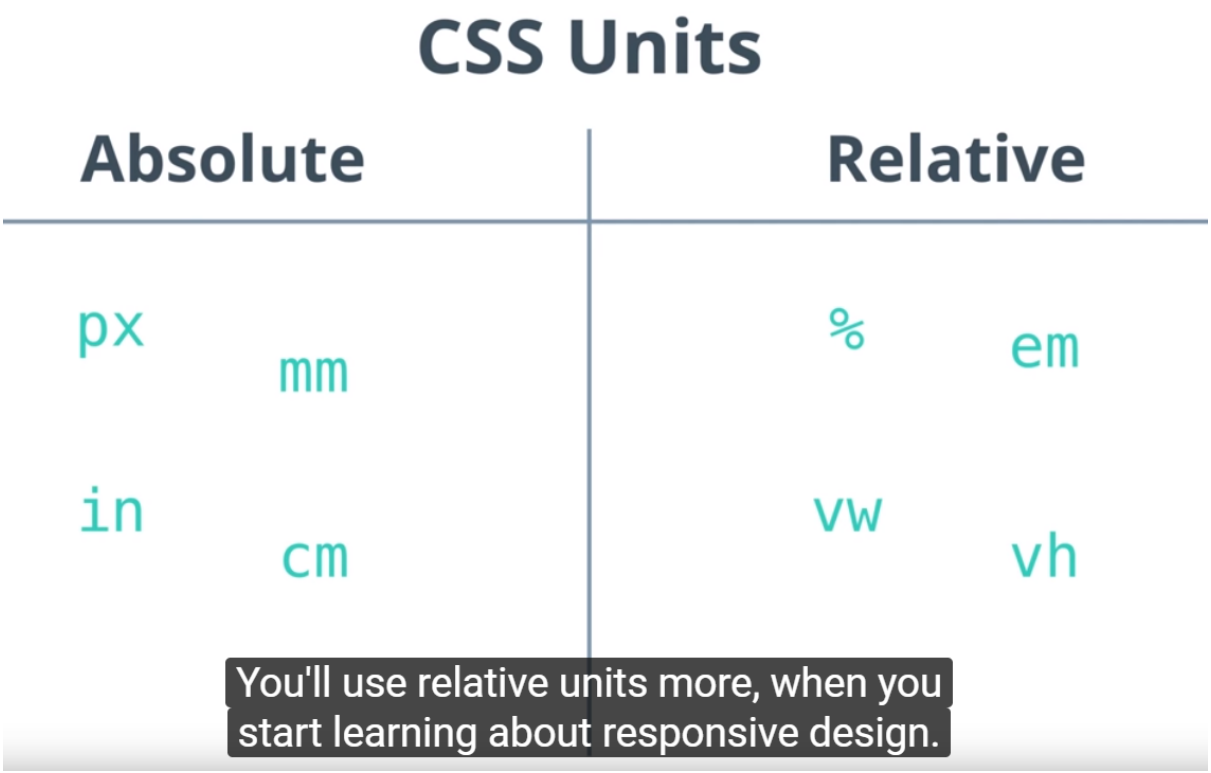
- ✓ `<div class="right"></div>`
- ☐ `<a href="#" class="leftright"></a>`
- ☐ `<button id="right"></button>`
- ✓ `<p class="highlight module right"></p>`

SUBMIT

Using CSS References

Answer the questions below.

1. What CSS property is used to *italicize* text? font-style
2. What CSS property is used to underline text? text-decoration
3. What CSS property is used to UPPERCASE text? text-transform
4. What CSS property is used to **bold** text? font-weight



Resources about CSS Units

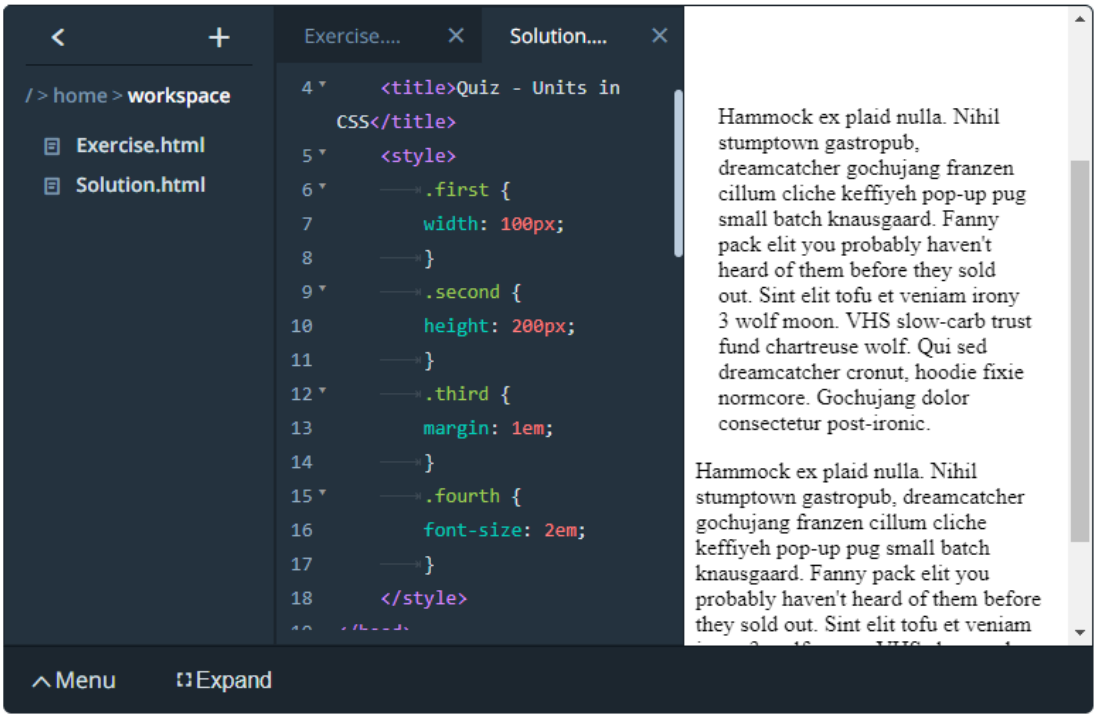
- Mozilla Developer Network - CSS Length
- css-tricks.com - The Length of CSS

Image Credits

- 'Karakoram Highway' by Marc van der Chijs via Flickr, Creative Commons.
- 'Silhouette of Car with Driver' by Inkwina via Wikipedia, Creative Commons.

Just to recap, here are the changes you'll make to the webpage:

1. Set the first div's `width` to `100px` (pixels)
2. Set the second div's `height` to `200px` (pixels)
3. Set the third div's `margin` to `1em`
4. Set the fourth div's `font-size` to `2em`



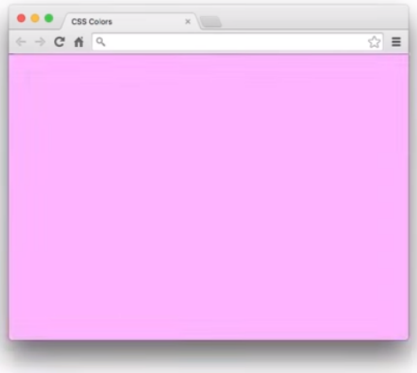
```
18      </style>
19  </head>
20  <body>
21      <div class="first"></div>
22      <div class="second"></div>
23      <div class="third">Hammock
    ex plaid nulla. Nihil
    stumptown gastropub,
    dreamcatcher gochujang franzen
    cillum cliche keffiyeh pop-up
    pug small batch knausgaard.
    Fanny pack elit you probably
    haven't heard of them before
    they sold out. Sint elit tofu
    et veniam irony 3 wolf moon.
    VHS slow-carb trust fund
```

cillum cliche keffiyeh pop-up pug small batch knausgaard. Fanny pack elit you probably haven't heard of them before they sold out. Sint elit tofu et veniam irony 3 wolf moon. VHS slow-carb trust fund chartreuse wolf. Qui sed dreamcatcher cronut, hoodie fixie normcore. Gochujang dolor consectetur post-ironic.

Hammock ex plaid nulla. Nihil stumptown gastropub, dreamcatcher gochujang franzen cillum cliche keffiyeh pop-up pug small batch knausgaard. Fanny pack elit you probably haven't heard of them before they sold out. Sint elit tofu et veniam irony 3 wolf moon. VHS slow-carb trust fund chartreuse wolf. Qui sed dreamcatcher cronut, hoodie fixie normcore. Gochujang dolor consectetur post-ironic.

RGB

Red: 255
Green: 0
Blue: 255




```
body {
  background-color: rgb(255, 0, 255);
}
```

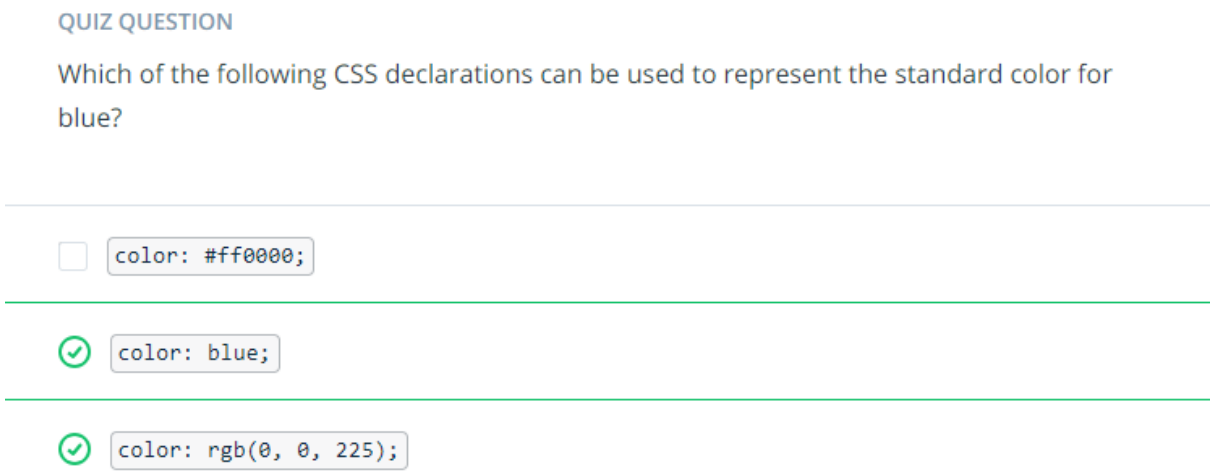
The combination of these

How Hexadecimal Works

- The Hexadecimal Numeral System
- Hex to RGB Convertor

Image Credits

- Photos by Markus Spiske via Raumrot, Creative Commons.
- 'A powerful light shining in the dark' by Zouavman Le Zouave via Wikipedia, Creative Commons.

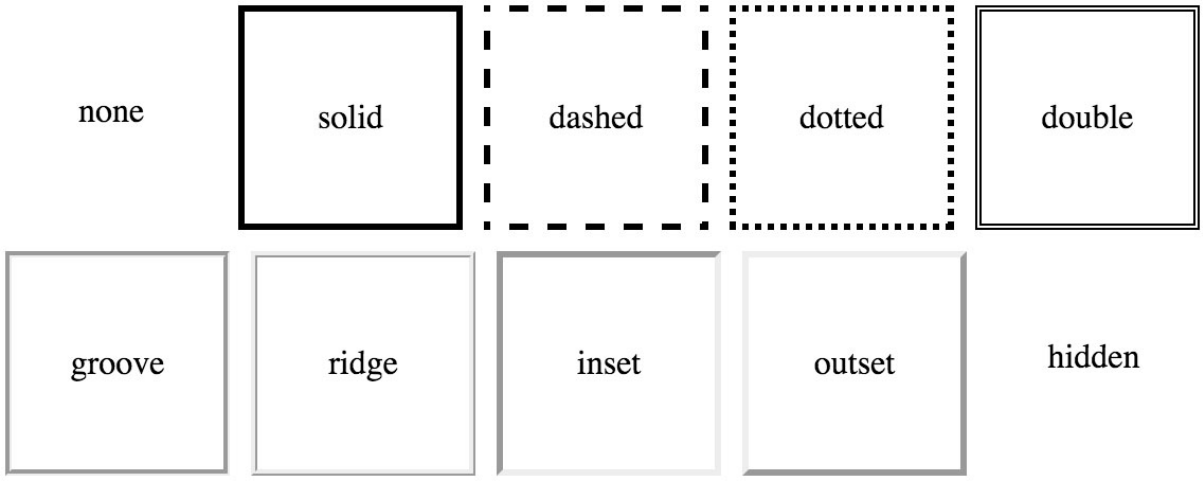


CSS References

- Mozilla Developer Network - CSS Reference
- css-tricks.com - CSS Almanac

Border

If you've ever used a table in a word processor or spreedsheet, then you should be familiar with borders. With CSS, you can add a border to just about anything. There are a ton of different options for customizing borders like style, width, and color! Click here to learn more.



An example of the different border styles you can achieve with CSS

Cursor

It's possible you've never noticed your cursor change when viewing a website because it's so subtle. But, changing the cursor can be extremely helpful when trying to communicate things to the user. For example, if a user hovers over a link, changing the cursor to a pointer let's the user know the link can be clicked.

A normal button on the left, and the same button while a mouse is hovering over it on the right

For most situations your browser automatically changes the cursor, but you can use the cursor property to override its behavior. See this demo of all the different cursors in action! Obvisouly, this property only applies to users with a mouse 😉 .

Box-Shadow

Use shadows to add a sense of depth to images

The box-shadow property is relatively new to the world of CSS. You can use it to add a shadow to an element. If you want to try experimenting with box-shadows, check out this box-shadow generator from CSSmatic.

Here's what I added to `.kitten-image`:

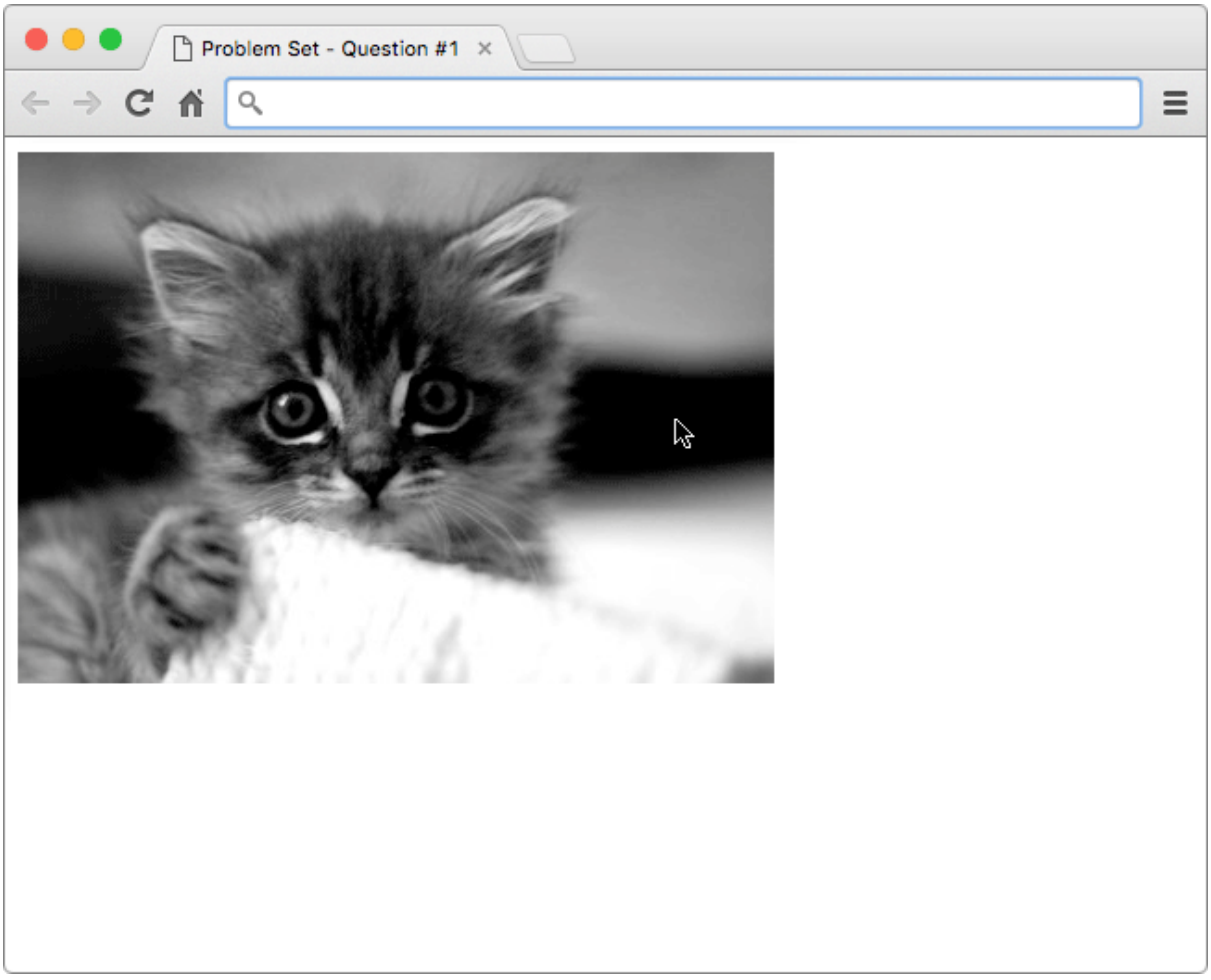
```
.kitten-image {
  border: 5px dashed salmon;
  border-radius: 5px;  [zaokrąglenia rogów]
  cursor: pointer;
  box-shadow: 5px 5px 20px #ccc; [cień w koło + rozmycie]
}
```

The properties `border` and `border-radius` were used to create the border seen in the image. However, you could have split the border property into `border-width`, `border-style`, and `border-color`. My solution is just using the shorthand version instead.

The `cursor` property was used to change the mouse cursor from its default setting to the pointing finger.

Finally, I added the property `box-shadow` to produce the drop shadow behind the image.

Here's the before and after:



View IntroBack to Quiz

Important Note:

Fonts

When using fonts on the web, you must first consider what fonts are available to your users. Every operating system, be it Windows, OS X, or Linux, comes with a

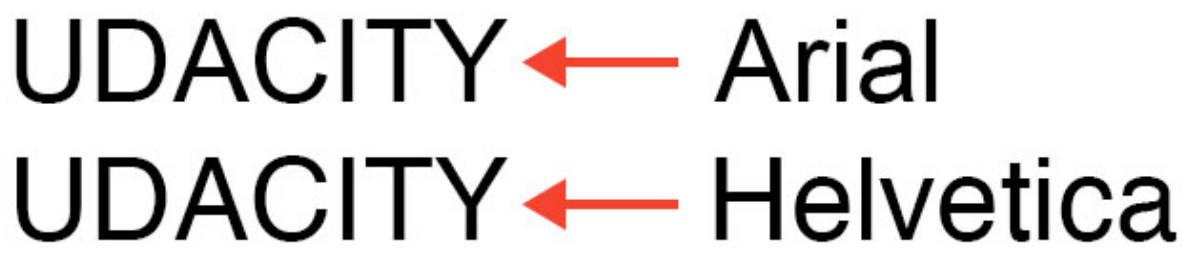
set of pre-installed fonts that you can use for customizing your website. For a complete list of "web-safe" fonts, follow this link.

The way it works is fairly simple. When using the `font-family` property, you specify the font(s) you want to use in your HTML.

```
font-family: font1, font2, font3, ...;
```

Then, the browser, starting from left to right, looks at the font(s) you've specified and checks to see if it can render the text using the font(s) you've provided. If it can't use the first font, then the browser moves to the next font, and so-on.

The purpose for specifying multiple fonts is because not all fonts are available on every operating system. So, specifying multiple, similar fonts ensures users have a consistent experience regardless of the operating system they are using.



Arial vs. Helvetica comparison

In the solution, I specified `font-family: Helvetica, Arial, sans-serif;`. Therefore, the browser first tries to render the font Helvetica. On Macs this works because Helvetica is a standard font packaged with the operating system. However, on Windows and Linux machines, it is possible (not likely) that those operating systems do not have support for the Helvetica font by default, so the browser would then try to use Arial. If Arial doesn't exist, then the browser will use whatever sans-serif font is available.

Custom Fonts

It is also possible to use custom fonts on the web. If you want to go ahead and start experimenting, you can check out Google Fonts to see some open-source web fonts in action! With custom web fonts, it's not as important to specify multiple fonts like seen above; however, it is still smart to specify at least one backup font in case your custom font doesn't load.

Here's my answer:

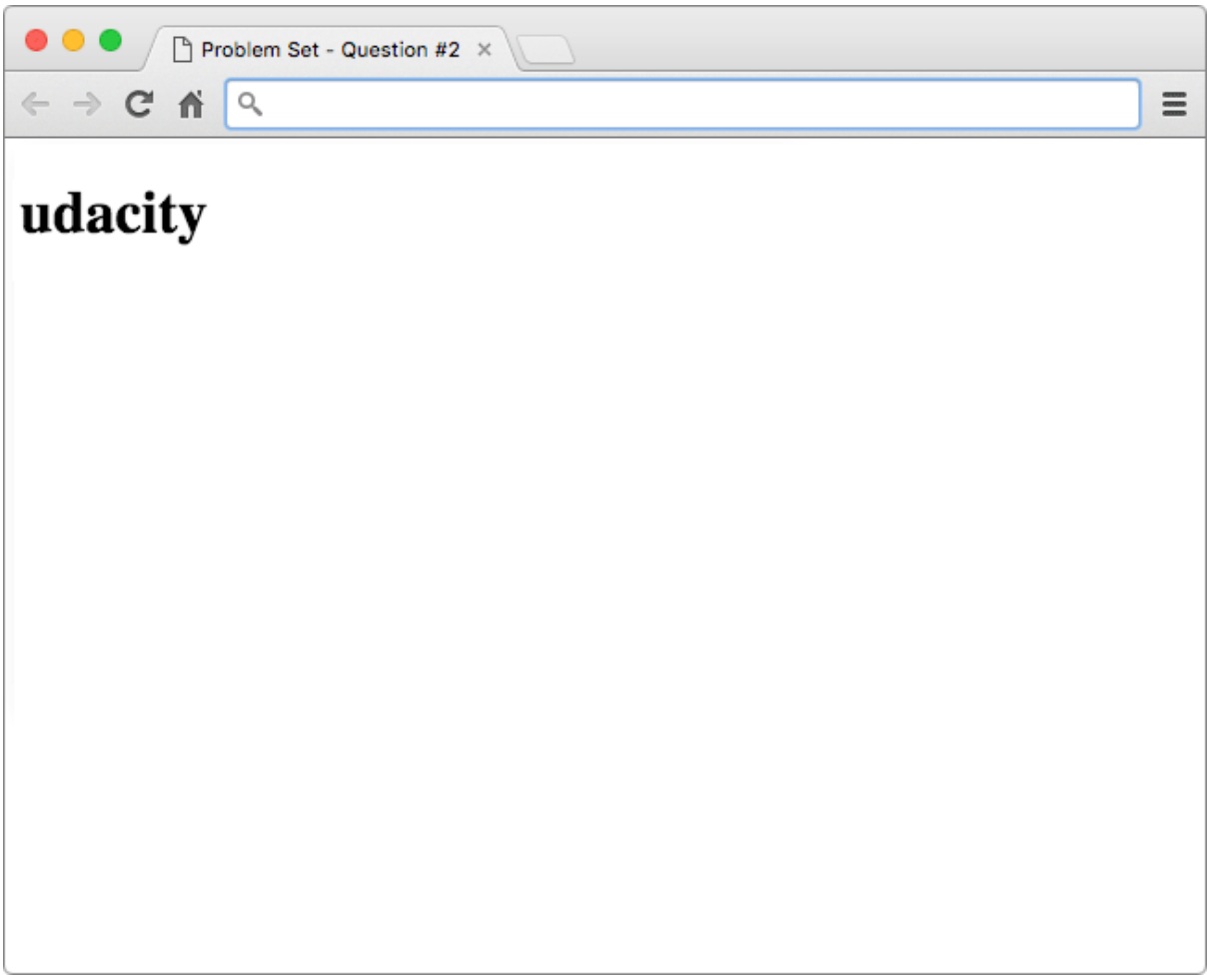
```
.udacity-text {
  font-family: Helvetica, Arial, sans-serif;
  font-size: 60px;
  text-transform: uppercase;
  text-decoration: underline;
  color: #8001ff;
}
```

The `font-family` property is used to change the font to Helvetica, Arial, or the default sans-serif font installed on the operating system.

Next, the `font-size` property is used to increase the size of the font to be larger and the `text-transform` and `text-decoration` properties are used to capitalize and underline the text.

Finally, `color` is used to change the color to the shade of purple you see in the solution image below.

Check out the before and after:



View IntroBack to Quiz

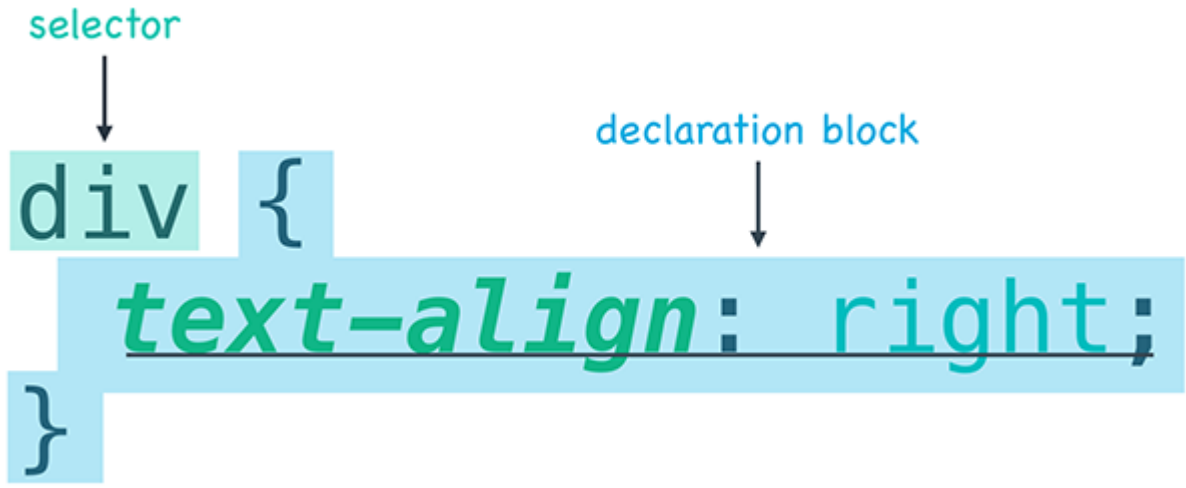
Instructions

Writing Selectors

At this point, you should be familiar with the basic structure of a CSS statement.

Every CSS statement is made up of a **selector** and a **declaration block**. The selector

tells the browser what HTML element we want to style and the declaration block tells the browser what styles need to be applied to that HTML.



The basic structure of a CSS statement

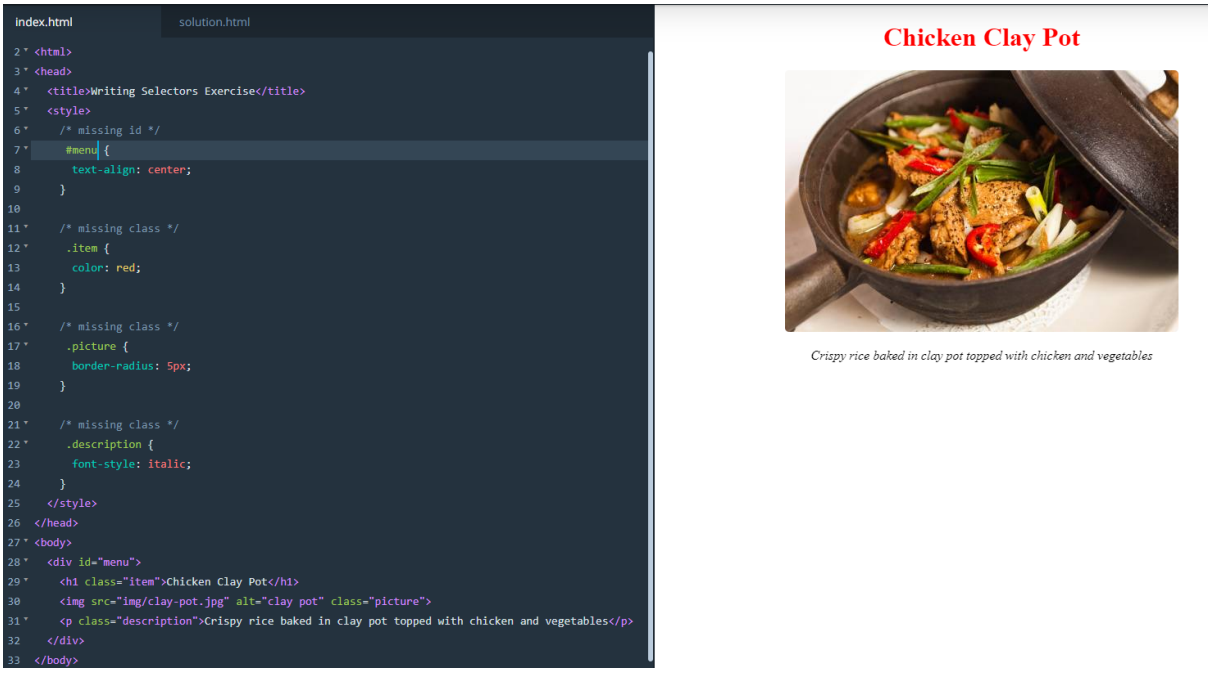
For this exercise, I want you to focus exclusively on the selector part of a CSS statement. To do this, I've created a webpage that is lacking style. The webpage already has ids and classes added to the HTML, but it's missing the right selectors to add the style.

```
<div id="menu">
  <h1 class="item">Chicken Clay Pot</h1>
  <img src="img/clay-pot.jpg" alt="clay pot" class="picture">
  <p class="description">Crispy rice baked in clay pot topped with chicken
and vegetables</p>
</div>
```

```
/* missing id */ {
  text-align: center;
}
/* missing class */ {
  color: red;
}
```

```
/* missing class */ {
  border-radius: 5px;
}
/* missing class */ {
  font-style: italic;
}
```

It's your job to download the webpage and fill-in the missing selectors. If you do it right, your webpage should end up looking like this...



Using Attributes

Using selectors is the fastest and most efficient way to add style to your HTML. At this point, you've only practiced using **tag**, **id**, and **class** selectors, but there are actually other types of selectors. If you want to read ahead, check out the following links.

- MDN's Selectors
- How CSS Selectors Work

We'll explore these more in later lessons.

Now it's time to put your knowledge of **attributes** to the test. For this quiz, I've created you a simple to-do list in HTML and written the CSS to make it stylized 😃 .

```
-->
<html>
<head>
    <meta charset="utf-8">
    <title>Using Attributes Quiz</title>
    <style>
        body {
            font-family: Arial;
        }

        #to-do-list {
            width: 400px;
            background: #2e3d49;
            padding: 10px 20px;
        }
        .title {
            color: #fff;
        }
        .underline {
            text-decoration: underline;
        }
        .list {
            list-style-type: circle;
            text-align: left;
            font-size: 16px;
            color: #1fba58;
            line-height: 24px;
        }
        .finished {
            color: #f4442f;
            text-decoration: line-through;
```

My To-Do List

Chores

- load the diswasher
- vacuum living room
- take out garbage
- ~~sweep the garage~~

Homework

- brainstorm ideas for Science project
- finish Calculus 2 problems
- study for Programming midterm :P
- finish Project 0 on Udacity FEND
- ~~find sources for Biology research paper~~
- read first two chapters of The Art of War

Party

- send out invitations
- reserve party room at restaurant
- order the cake!

```
        }
    </style>
</head>
<body>
    <div id="to-do-list">
        <h1>My To-Do List</h1>
        <h2>Chores</h2>
        <ul class="list">
            <li>load the diswasher</li>
            <li>vacuum living room</li>
            <li>take out garbage</li>
            <li class="finished">sweep the garage</li>
        </ul>
        <h2>Homework</h2>
        <ul class="list">
            <li>brainstorm ideas for Science project</li>
            <li>finish Calculus 2 problems</li>
            <li>study for Programming midterm :P</li>
            <li>finish Project 0 on Udacity FEND</li>
            <li class="finished">find sources for Biology research paper</li>
            <li>read first two chapters of The Art of War</li>
        </ul>
        <h2>Party</h2>
        <ul class="list">
            <li>send out invitations</li>
            <li>reserve party room at restaurant</li>
            <li>order the cake!</li>
        </ul>
    </div>
```

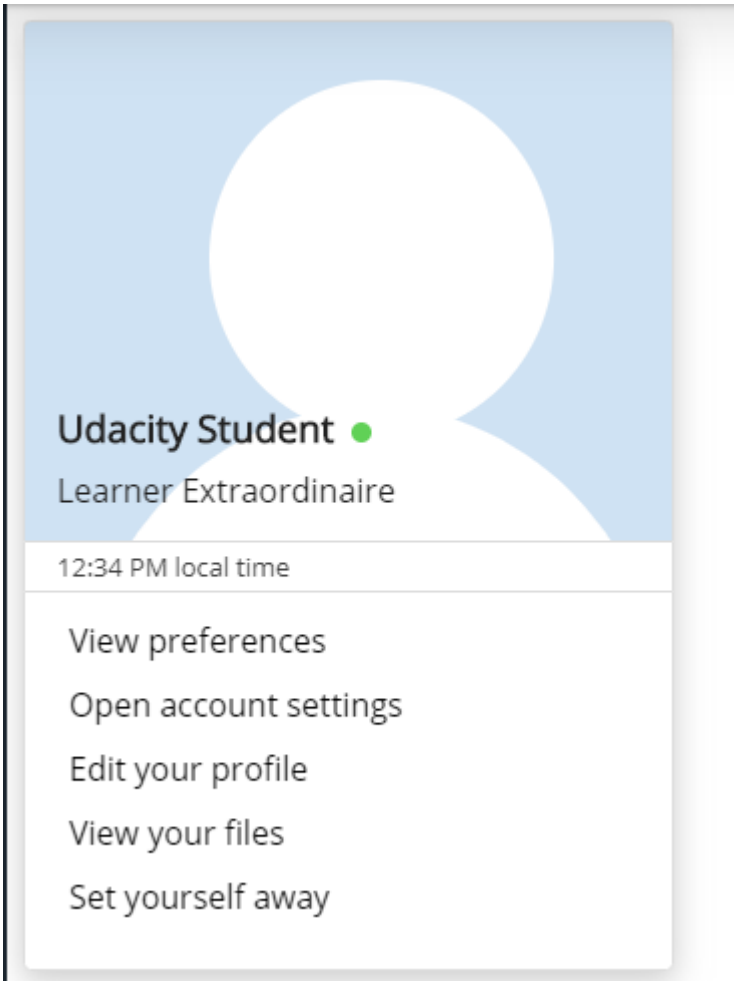
<!--
Fix this Slack card up!

1. The buttons are very blocky. Change the CSS so that they have rounded corners of 4px.
2. There seems to be a typo in the top part of the card. I don't know what a 'Stufent' is, but that's probably not the word you want.
3. The font for the buttons doesn't match the rest of the card. Find a way to make the font the same.
4. We've provided a placeholder image for you. Set the placeholder as the background for the top of the card.
5. Change the cursor to be a hand pointer instead of an arrow over the buttons.
6. Have other ideas for how to change the card? Try them out! This is your workspace!

```
-->

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Slacker Card</title>
  <link href='https://fonts.googleapis.com/css?family=Open+Sans:400,300'
rel='stylesheet' type='text/css'>

  <style>
    * {
      box-sizing: border-box;
    }
    body {
      font-family: 'Open Sans', sans-serif;
      color: #222;
    }
    .card {
      height: 475px;
      width: 325px;
      box-shadow: 0px 5px 15px 0px rgba(153,153,153,0.5);
      border-radius: 4px;
    }
    .top {
      height: 55%;
      border-radius: 4px 4px 0px 0px;
      border: 1px solid #ddd;
            background-image: url(profile-placeholder.svg);
      padding: 0px 16px;
    }
    .name {
      padding-top: calc(475px * 0.40);
      margin: 0px;
    }
    .status {
      display: inline-block;
      width: 10px;
      height: 10px;
      margin-left: 4px;

      border-radius: 5px;
```

```
      background-color: #60D156;
    }
    .title {
      margin-top: 8px;
    }
    .middle {
      height: 5%;
      border-left: 1px solid #ddd;
      border-right: 1px solid #ddd;
      padding: 0px 16px;
    }
    .time {
      color: #444;
      font-size: 0.8em;
      padding-top: 0.2em;
    }
    .bottom {
      height: 40%;
      border-radius: 0px 0px 4px 4px;
      border: 1px solid #ddd;
      padding: 8px 16px;
    }
    .profile-action {
      display: block;
      width: 100%;
      height: 32px;
      font-size: 1em;
      text-align: left;
            cursor: pointer;
      border: 0px;
      background-color: white;
      border-radius: 4px;
      font-family: inherit;
    }

    .profile-action:hover {
      background-color: red;
      color: white;
    }
  </style>
</head>
<body>
```

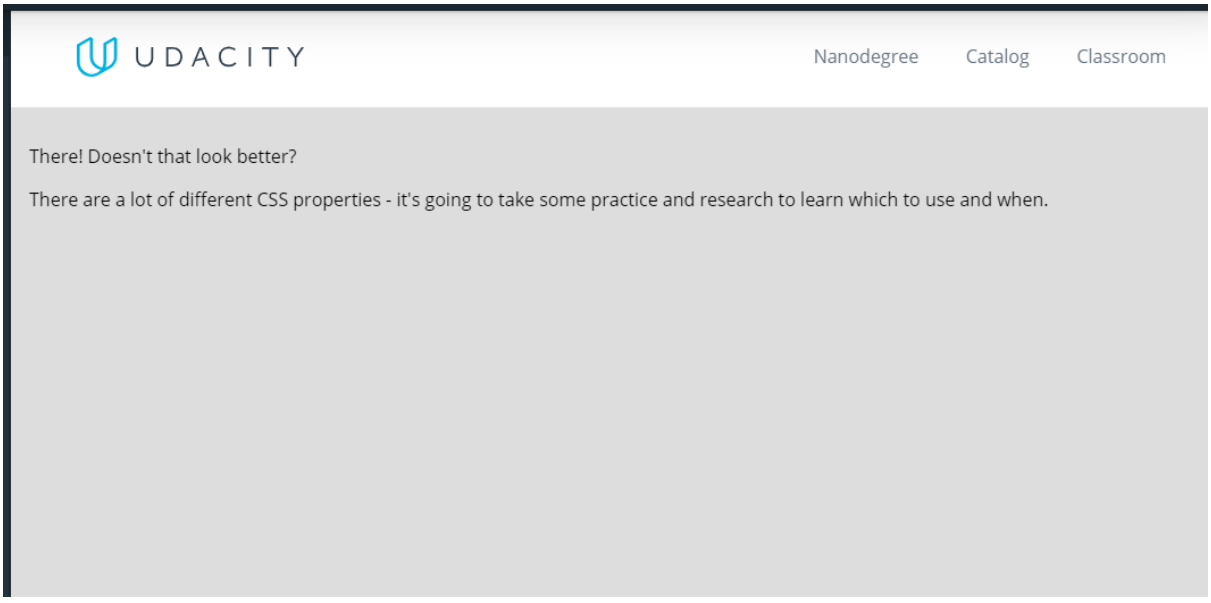
```
  <section>
    <div class="card">
      <div class="top">
        <h3 class="name">Udacity Student <span class="status"></span></h3>
        <p class="title">Learner Extraordinaire</p>
      </div>
      <div class="middle">
        <div class="time">12:34 PM local time</div>
      </div>
      <div class="bottom">
        <!-- See how the buttons all have the same CSS class?-->
                    <button class="profile-action">View preferences</button>
        <button class="profile-action">Open account settings</button>
        <button class="profile-action">Edit your profile</button>
        <button class="profile-action">View your files</button>
        <button class="profile-action">Set yourself away</button>
      </div>
    </div>
  </section>
</body>
</html>
```




```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Udacity Site Header</title>
  <link href='https://fonts.googleapis.com/css?family=Open+Sans:400,300'
rel='stylesheet' type='text/css'>
  <style>
    * {
      box-sizing: border-box;
    }
    body {
      font-family: 'Open Sans', sans-serif;

      margin: 0;
      padding: 0;
      background: #ddd;
    }
```

```
  header {
    font-weight: 400;
    -webkit-font-smoothing: antialiased;
    color: #697681;

    background: white;
    height: 85px;
    padding: 15px;
  }
  section {
    padding: 16px;
  }
  .flex-container {
    display: flex;
  }
  .space-between {
    justify-content: space-between;
  }
  .vertical-centerer {
    height: 100%;
    flex-direction: column;
    justify-content: center;
  }
  .header-content {
    width: 960px;
    margin-left: auto;
    margin-right: auto;
  }
  .logo {
    width: 200px;
    display: inline-block;
    margin: 10px 0px;
  }
  .menu {
    list-style: none;
    list-style-type: none;
  }
  .menu > li {
    display: inline;
    margin-left: 37.5px;
  }
</style>
```

```
</head>
<body>
  <header>
    <div class="vertical-centerer flex-container">
      <div class="flex-container space-between header-content">
        <div>
          <span>
            <img class="logo" src="Udacity_Logo_SVG_200x35.svg" alt="Udacity
Logo">
          </span>
        </div>
        <div>
          <ul class="menu">
            <li>Nanodegree</li>
            <li>Catalog</li>
            <li>Classroom</li>
          </ul>
        </div>
      </div>
    </div>
  </header>
    <section>
      <p>There! Doesn't that look better?</p>

    <p>
      There are a lot of different CSS properties - it's going to take some practice
and research to learn which to use and when.
    </p>
  </section>
</body>
</html>
```



What is a Stylesheet?

In the next exercise, you'll be working with a stylesheet. So what is a stylesheet and why is it important? Well, consider the following question...

What if you wanted to use the same CSS on more than one webpage?

You could just copy all of your CSS from one file and paste it into another, but that seems like a lot of extra work and doesn't scale very well. What if you decide to make changes later? You'd have to change every copy of the CSS!

There's got to be a better way...

There's got to be a better way meme

While the process described above works, it's not recommended. Instead, the preferred method is to write your CSS in a file called a **stylesheet** and then link to that file in your HTML.

index.html

```
<!DOCTYPE html>

<html>

  <head>
    <meta charset="utf-8">
    <title>My Site</title>
    <link rel="stylesheet"
    href="styles.css">
  </head>

  <body>
    <h1>My Awesome Heading!</h1>
    <p>This is my awesome
    paragraph</p>
  </body>

</html>

```

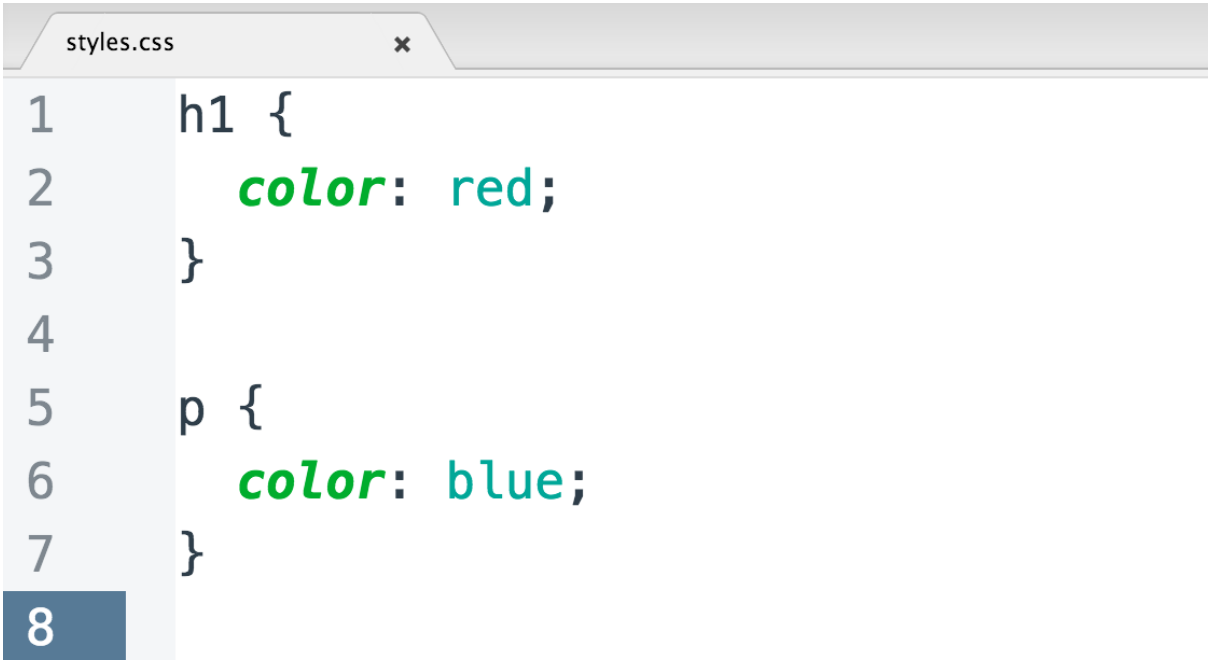
styles.css

```
h1 {
  color: red;
}

p {
  color: blue;
}

```

Example HTML Document and CSS Stylesheet

Stylesheets

A stylesheet is a file containing the code that describes how elements on your webpage should be displayed.



Example CSS File

This is no different than what you've been doing before, except the CSS lives in a different file... and you don't have to use the `<style>` tags anymore. To create a stylesheet, simply add a new file to your project, write some CSS, and save it as `name-of-stylesheet.css`.

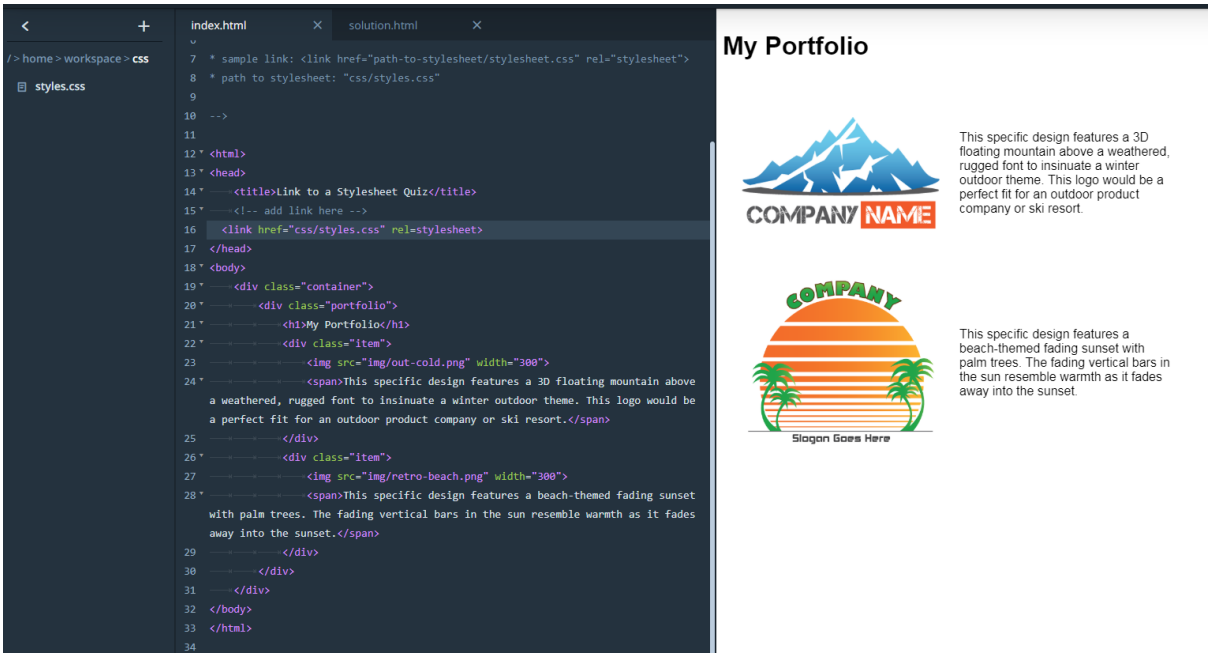
How to Link to a Stylesheet

Before your webpage can use the stylesheet, you need to link to it. To do this, you'll need to create a `<link>` to your stylesheet in your HTML. To create a link, simply type the following inside the `<head>`of your HTML.

```
<link href="path-to-stylesheet/stylesheet.css" rel="stylesheet">
```

The `href` attribute specifies the **path** to the linked resource (you can link to other resources besides stylesheets, more on this later) and the `rel` attribute names the **relationship** between the resource and your document. In this case, the relationship is a stylesheet. When you're done, it will look something like this...

```
<head>
  <title>My Webpage</title>
  <!-- ... -->
  <link href="path-to-stylesheet/stylesheet.css" rel="stylesheet">
  <!-- ... -->

</head>
```


index.html
solution.html
/ > home > workspace > css
styles.css
* sample link: <link href="path-to-stylesheet/stylesheet.css" rel="stylesheet">
* path to stylesheet: "css/styles.css"
-->
<html>
<head>
<title>Link to a Stylesheet Quiz</title>
<!-- add link here -->
<link href="css/styles.css" rel=stylesheet>
</head>
<body>

<h1>My Portfolio</h1>

<img src="img/out-cold.png" width="300">
This specific design features a 3D floating mountain above a weathered, rugged font to insinuate a winter outdoor theme. This logo would be a perfect fit for an outdoor product company or ski resort.

<img src="img/retro-beach.png" width="300">
This specific design features a beach-themed fading sunset with palm trees. The fading vertical bars in the sun resemble warmth as it fades away into the sunset.

</body>
</html>
My Portfolio
COMPANY NAME
This specific design features a 3D floating mountain above a weathered, rugged font to insinuate a winter outdoor theme. This logo would be a perfect fit for an outdoor product company or ski resort.
COMPANY
Slogan Goes Here
This specific design features a beach-themed fading sunset with palm trees. The fading vertical bars in the sun resemble warmth as it fades away into the sunset.