Contents meo mẻo mèo meo =(0w0)=

[ **Turn an Image into a Link** 5](#_Toc45111951)

[ Create a Bulleted Unordered ListPassed 5](#_Toc45111952)

[ Create an Ordered List 6](#_Toc45111953)

[ Create a Text Field 6](#_Toc45111954)

[ Add Placeholder Text to a Text Field 6](#_Toc45111955)

[ **Create a Form Element** 7](#_Toc45111956)

[ Add a Submit Button to a FormPassed 7](#_Toc45111957)

[ Use HTML5 to Require a Field 7](#_Toc45111958)

[ Basic CSS: Use a CSS Class to Style an Element 8](#_Toc45111959)

[ Basic CSS: Change the Font Size of an Element 8](#_Toc45111960)

[ Basic CSS: Set the Font Family of an Element 8](#_Toc45111961)

[ Basic CSS: Import a Google FontPassed 9](#_Toc45111962)

[ Basic CSS: Specify How Fonts Should Degrade 10](#_Toc45111963)

[ Basic CSS: Size Your Images 10](#_Toc45111964)

[ Basic CSS: Add Borders Around Your Elements 11](#_Toc45111965)

[ Basic CSS: Add Rounded Corners with border-radius 12](#_Toc45111966)

[ Basic CSS: Make Circular Images with a border-radius 12](#_Toc45111967)

[ Basic CSS: Give a Background Color to a div Element 12](#_Toc45111968)

[ Basic CSS: Set the id of an Element 12](#_Toc45111969)

[ Basic CSS: Use an id Attribute to Style an Element 13](#_Toc45111970)

[ Basic CSS: Adjust the Padding of an Element 14](#_Toc45111971)

[ Basic CSS: Adjust the Margin of an Element 15](#_Toc45111972)

[ Basic CSS: Add a Negative Margin to an Element 16](#_Toc45111973)

[ Basic CSS: Add Different Padding to Each Side of an Element 16](#_Toc45111974)

[ Basic CSS: Add Different Margins to Each Side of an Element 17](#_Toc45111975)

[ Basic CSS: Use Clockwise Notation to Specify the Padding of an Element 17](#_Toc45111976)

[ Basic CSS: Use Clockwise Notation to Specify the Margin of an Element 17](#_Toc45111977)

[ Basic CSS: Use Attribute Selectors to Style Elements 18](#_Toc45111978)

[ Basic CSS: Understand Absolute versus Relative Units 18](#_Toc45111979)

[ Basic CSS: Style the HTML Body Element 19](#_Toc45111980)

[ **Basic CSS: Inherit Styles from the Body Element** 19](#_Toc45111981)

[ **Basic CSS: Prioritize One Style Over Another** 20](#_Toc45111982)

[**Basic CSS: Override Styles in Subsequent CSS** 21](#_Toc45111983)

[ **Basic CSS: Override Class Declarations by Styling ID Attributes** 22](#_Toc45111984)

[ **Basic CSS: Override Class Declarations with Inline Styles** 23](#_Toc45111985)

[ **Basic CSS: Override All Other Styles by using Important** 23](#_Toc45111986)

[ Basic CSS: Use Hex Code for Specific Colors 24](#_Toc45111987)

[ Basic CSS: Use Hex Code to Mix Colors 24](#_Toc45111988)

**HỌC HTML**

* The h2 element you will be adding in this step will add a level two heading to the web page.

This element tells the browser about the structure of your website. h1 elements are often used for main headings, while h2 elements are generally used for subheadings. There are also h3, h4, h5 and h6 elements to indicate different levels of subheadings.

* p elements are the preferred element for paragraph text on websites. p is short for "paragraph".

You can create a paragraph element like this:

<p>I'm a p tag!</p>

**Note:** As a convention, all HTML tags are written in lowercase, for example <p></p> and not <P></P>.

* These tags give a descriptive structure to your HTML, make your HTML easier to read, and help with Search Engine Optimization (SEO) and accessibility. The main HTML5 tag helps search engines and other developers find the main content of your page.

Example usage, a main element with two child elements nested inside it:

<main>

<h1>Hello World</h1>

<p>Hello Paragraph</p>

</main>

* You can add images to your website by using the img element, and point to a specific image's URL using the src attribute.

An example of this would be:

<img src="https://www.your-image-source.com/your-image.jpg">

Note that img elements are self-closing.

All img elements **must** have an alt attribute. The text inside an alt attribute is used for screen readers to improve accessibility and is displayed if the image fails to load.

**Note:** If the image is purely decorative, using an empty alt attribute is a best practice.

Ideally the alt attribute should not contain special characters unless needed.

Let's add an alt attribute to our img example above:

<img src="https://www.your-image-source.com/your-image.jpg" alt="Author standing on a beach with two thumbs up.">

* You can use a (*anchor*) elements to link to content outside of your web page.

a elements need a destination web address called an href attribute. They also need anchor text. Here's an example:

<a href="https://freecodecamp.org">this links to freecodecamp.org</a>

Then your browser will display the text **"this links to freecodecamp.org"** as a link you can click. And that link will take you to the web address <https://www.freecodecamp.org>.

* a (*anchor*) elements can also be used to create internal links to jump to different sections within a webpage.

To create an internal link, you assign a link's href attribute to a hash symbol # plus the value of the id attribute for the element that you want to internally link to, usually further down the page. You then need to add the same id attribute to the element you are linking to. An id is an attribute that uniquely describes an element.

Below is an example of an internal anchor link and its target element:

<a href="#contacts-header">Contacts</a>

...

<h2 id="contacts-header">Contacts</h2>

When users click the Contacts link, they'll be taken to the section of the webpage with the **Contacts** header element.

* Sometimes you want to add a elements to your website before you know where they will link.

This is also handy when you're changing the behavior of a link using JavaScript, which we'll learn about later.

The current value of the href attribute is a link that points to "http://freecatphotoapp.com". Replace the href attribute value with a #, also known as a hash symbol, to create a dead link.

For example: href="#"

* **Turn an Image into a Link**

You can make elements into links by nesting them within an a element.

Nest your image within an a element. Here's an example:

<a href="#"><img src="https://bit.ly/fcc-running-cats" alt="Three kittens running towards the camera."></a>

Remember to use # as your a element's href property in order to turn it into a dead link.

## Create a Bulleted Unordered ListPassed

HTML has a special element for creating unordered lists, or bullet point style lists.

Unordered lists start with an opening <ul> element, followed by any number of <li> elements. Finally, unordered lists close with a </ul>

For example:

<ul>

<li>milk</li>

<li>cheese</li>

</ul>

would create a bullet point style list of "milk" and "cheese".

## Create an Ordered List

HTML has another special element for creating ordered lists, or numbered lists.

Ordered lists start with an opening <ol> element, followed by any number of <li> elements. Finally, ordered lists are closed with the </ol> tag.

For example:

<ol>

<li>Garfield</li>

<li>Sylvester</li>

</ol>

would create a numbered list of "Garfield" and "Sylvester".

## Create a Text Field

Now let's create a web form.

input elements are a convenient way to get input from your user.

You can create a text input like this:

<input type="text">

Note that input elements are self-closing.

## Add Placeholder Text to a Text Field

Placeholder text is what is displayed in your input element before your user has inputted anything.

You can create placeholder text like so:

<input type="text" placeholder="this is placeholder text">

**Note:** Remember that input elements are self-closing.

* **Create a Form Element**

You can build web forms that actually submit data to a server using nothing more than pure HTML. You can do this by specifying an action on your form element.

For example:

<form action="/url-where-you-want-to-submit-form-data"></form>

Nest your text field inside a form element, and add the action="/submit-cat-photo" attribute to the form element.

## Add a Submit Button to a FormPassed

Let's add a submit button to your form. Clicking this button will send the data from your form to the URL you specified with your form's action attribute.

Here's an example submit button:

<button type="submit">this button submits the form</button>

## Use HTML5 to Require a Field

You can require specific form fields so that your user will not be able to submit your form until he or she has filled them out.

For example, if you wanted to make a text input field required, you can just add the attribute required within your input element, like this: <input type="text" required>

Định dạnh màu cho element:

Ex: <h2 style=color: blue> Cat photos </h2>

Cái này chỉ định dạng cho mỗi dòng này thôi

Giả dụ h2 có nhiều thành phần thì làm cách này cho dễ:

<style>

  h2 {

    color: blue;

  }

</style>

<h2>CatPhotoApp</h2>

Nếu làm cái này thì nó sẽ định dạng hết phần h2 và những element trong đó

## Basic CSS: Use a CSS Class to Style an Element

Classes are reusable styles that can be added to HTML elements.

Here's an example CSS class declaration:

<style>

.blue-text {

color: blue;

}

</style>

You can see that we've created a CSS class called blue-text within the <style> tag. You can apply a class to an HTML element like this: <h2 class="blue-text">CatPhotoApp</h2> Note that in your CSS style element, class names start with a period. In your HTML elements' class attribute, the class name does not include the period.

## Basic CSS: Change the Font Size of an Element

Font size is controlled by the font-size CSS property, like this:

h1 {

font-size: 30px;

}

## Basic CSS: Set the Font Family of an Element

You can set which font an element should use, by using the font-family property.

For example, if you wanted to set your h2 element's font to sans-serif, you would use the following CSS:

h2 {

font-family: sans-serif;

}

## Basic CSS: Import a Google FontPassed

In addition to specifying common fonts that are found on most operating systems, we can also specify non-standard, custom web fonts for use on our website. There are many sources for web fonts on the Internet. For this example we will focus on the Google Fonts library.

[Google Fonts](https://fonts.google.com/) is a free library of web fonts that you can use in your CSS by referencing the font's URL.

So, let's go ahead and import and apply a Google font (note that if Google is blocked in your country, you will need to skip this challenge).

To import a Google Font, you can copy the font(s) URL from the Google Fonts library and then paste it in your HTML. For this challenge, we'll import the Lobster font. To do this, copy the following code snippet and paste it into the top of your code editor (before the opening style element):

<link href="https://fonts.googleapis.com/css?family=Lobster" rel="stylesheet" type="text/css">

Now you can use the Lobster font in your CSS by using Lobster as the FAMILY\_NAME as in the following example:  
font-family: FAMILY\_NAME, GENERIC\_NAME;.

The GENERIC\_NAME is optional, and is a fallback font in case the other specified font is not available. This is covered in the next challenge.

Family names are case-sensitive and need to be wrapped in quotes if there is a space in the name. For example, you need quotes to use the "Open Sans" font, but not to use the Lobster font.

## Basic CSS: Specify How Fonts Should Degrade

There are several default fonts that are available in all browsers. These generic font families include monospace, serif and sans-serif

When one font isn't available, you can tell the browser to "degrade" to another font.

For example, if you wanted an element to use the Helvetica font, but degrade to the sans-serif font when Helvetica isn't available, you will specify it as follows:

p {

font-family: Helvetica, sans-serif;

}

Generic font family names are not case-sensitive. Also, they do not need quotes because they are CSS keywords.

How to comment out:

<!--

<link href="https://fonts.googleapis.com/css?family=Lobster" rel="stylesheet" type="text/css"> -->

## Basic CSS: Size Your Images

CSS has a property called width that controls an element's width. Just like with fonts, we'll use px (pixels) to specify the image's width.

For example, if we wanted to create a CSS class called larger-image that gave HTML elements a width of 500 pixels, we'd use:

<style>

.larger-image {

width: 500px;

}

</style>

EX:

   <style>

}

    .smaller-image {

      width: 100px;

    }

  </style>

<a href="#"><img class="smaller-image" src="https://bit.ly/fcc-relaxing-cat" alt="A cute orange cat lying on its back."></a>

## Basic CSS: Add Borders Around Your Elements

CSS borders have properties like style, color and width.

For example, if we wanted to create a red, 5 pixel border around an HTML element, we could use this class:

<style>

.thin-red-border {

border-color: red;

border-width: 5px;

border-style: solid;

}

</style>

Remember that you can apply multiple classes to an element using its class attribute, by separating each class name with a space. For example:

<img class="class1 class2">

<a href="#"><img class="smaller-image thick-green-border" src="https://bit.ly/fcc-relaxing-cat" alt="A cute orange cat lying on its back."></a>

## Basic CSS: Add Rounded Corners with border-radius

Your cat photo currently has sharp corners. We can round out those corners with a CSS property called border-radius.

## Basic CSS: Make Circular Images with a border-radius

In addition to pixels, you can also specify the border-radius using a percentage.

## Basic CSS: Give a Background Color to a div Element

You can set an element's background color with the background-color property.

For example, if you wanted an element's background color to be green, you'd put this within your style element:

.green-background {

background-color: green;

}

## Basic CSS: Set the id of an Element

In addition to classes, each HTML element can also have an id attribute.

There are several benefits to using id attributes: You can use an id to style a single element and later you'll learn that you can use them to select and modify specific elements with JavaScript.

id attributes should be unique. Browsers won't enforce this, but it is a widely agreed upon best practice. So please don't give more than one element the same id attribute.

Here's an example of how you give your h2 element the id of cat-photo-app:

<h2 id="cat-photo-app">

Ex:

 <form action="/submit-cat-photo"

  id="cat-photo-form">

    <label><input type="radio" name="indoor-outdoor" checked> Indoor</label>

    <label><input type="radio" name="indoor-outdoor"> Outdoor</label><br>

    <label><input type="checkbox" name="personality" checked> Loving</label>

    <label><input type="checkbox" name="personality"> Lazy</label>

    <label><input type="checkbox" name="personality"> Energetic</label><br>

    <input type="text" placeholder="cat photo URL" required>

    <button type="submit">Submit</button>

</form>

One cool thing about id attributes is that, like classes, you can style them using CSS.

However, an id is not reusable and should only be applied to one element. An id also has a higher specificity (importance) than a class so if both are applied to the same element and have conflicting styles, the styles of the id will be applied.

Here's an example of how you can take your element with the id attribute of cat-photo-element and give it the background color of green. In your style element:

#cat-photo-element {

background-color: green;

}

Note that inside your style element, you always reference classes by putting a . in front of their names. You always reference ids by putting a # in front of their names

## Basic CSS: Use an id Attribute to Style an Element

One cool thing about id attributes is that, like classes, you can style them using CSS.

However, an id is not reusable and should only be applied to one element. An id also has a higher specificity (importance) than a class so if both are applied to the same element and have conflicting styles, the styles of the id will be applied.

Here's an example of how you can take your element with the id attribute of cat-photo-element and give it the background color of green. In your style element:

#cat-photo-element {

background-color: green;

}

Note that inside your style element, you always reference classes by putting a . in front of their names. You always reference ids by putting a # in front of their names.

Ex:

  <style>

}

  #cat-photo-form {

    background-color: green;

  }

</style>

## Basic CSS: Adjust the Padding of an Element

Now let's put our Cat Photo App away for a little while and learn more about styling HTML.

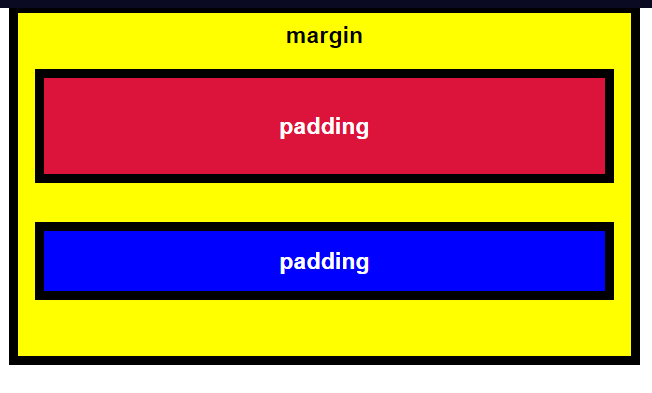
You may have already noticed this, but all HTML elements are essentially little rectangles.

Three important properties control the space that surrounds each HTML element: padding, margin, and border.

An element's padding controls the amount of space between the element's content and its border.

Here, we can see that the blue box and the red box are nested within the yellow box. Note that the red box has more padding than the blue box.

When you increase the blue box's padding, it will increase the distance (padding) between the text and the border around it.

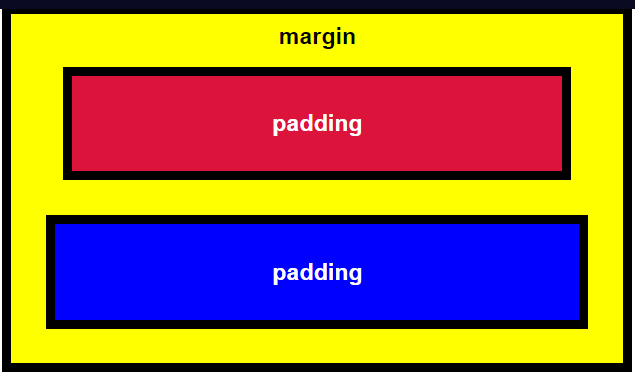


## Basic CSS: Adjust the Margin of an Element

An element's margin controls the amount of space between an element's border and surrounding elements.

Here, we can see that the blue box and the red box are nested within the yellow box. Note that the red box has a bigger margin than the blue box, making it appear smaller.

When you increase the blue box's margin, it will increase the distance between its border and surrounding elements.



## Basic CSS: Add a Negative Margin to an Element

An element's margin controls the amount of space between an element's border and surrounding elements.

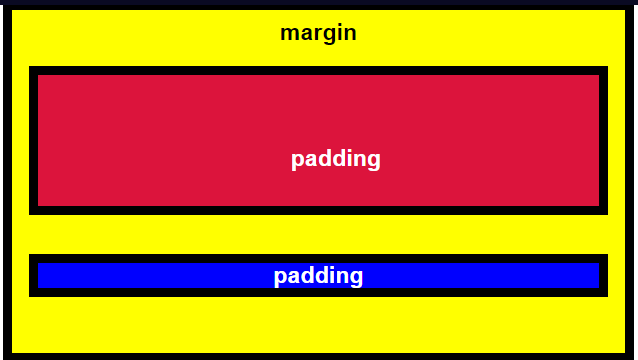
If you set an element's margin to a negative value, the element will grow larger.



## Basic CSS: Add Different Padding to Each Side of an Element

Sometimes you will want to customize an element so that it has different amounts of padding on each of its sides.

CSS allows you to control the padding of all four individual sides of an element with the padding-top, padding-right, padding-bottom, and padding-left properties.



## Basic CSS: Add Different Margins to Each Side of an Element

Sometimes you will want to customize an element so that it has a different margin on each of its sides.

CSS allows you to control the margin of all four individual sides of an element with the margin-top, margin-right, margin-bottom, and margin-left properties.

## Basic CSS: Use Clockwise Notation to Specify the Padding of an Element

Instead of specifying an element's padding-top, padding-right, padding-bottom, and padding-left properties individually, you can specify them all in one line, like this:

padding: 10px 20px 10px 20px;

These four values work like a clock: top, right, bottom, left, and will produce the exact same result as using the side-specific padding instructions.

## Basic CSS: Use Clockwise Notation to Specify the Margin of an Element

Let's try this again, but with margin this time.

Instead of specifying an element's margin-top, margin-right, margin-bottom, and margin-left properties individually, you can specify them all in one line, like this:

margin: 10px 20px 10px 20px;

These four values work like a clock: top, right, bottom, left, and will produce the exact same result as using the side-specific margin instructions.

## Basic CSS: Use Attribute Selectors to Style Elements

You have been adding id or class attributes to elements that you wish to specifically style. These are known as ID and class selectors. There are other CSS Selectors you can use to select custom groups of elements to style.

Let's bring out CatPhotoApp again to practice using CSS Selectors.

For this challenge, you will use the [attr=value] attribute selector to style the checkboxes in CatPhotoApp. This selector matches and styles elements with a specific attribute value. For example, the below code changes the margins of all elements with the attribute type and a corresponding value of radio:

[type='radio'] {

margin: 20px 0px 20px 0px;

}

## Basic CSS: Understand Absolute versus Relative Units

The last several challenges all set an element's margin or padding with pixels (px). Pixels are a type of length unit, which is what tells the browser how to size or space an item. In addition to px, CSS has a number of different length unit options that you can use.

The two main types of length units are absolute and relative. Absolute units tie to physical units of length. For example, in and mm refer to inches and millimeters, respectively. Absolute length units approximate the actual measurement on a screen, but there are some differences depending on a screen's resolution.

Relative units, such as em or rem, are relative to another length value. For example, em is based on the size of an element's font. If you use it to set the font-size property itself, it's relative to the parent's font-size.

**Note:** There are several relative unit options that are tied to the size of the viewport. They are covered in the Responsive Web Design Principles section.

## Basic CSS: Style the HTML Body Element

Now let's start fresh and talk about CSS inheritance.

Every HTML page has a body element.

We can prove that the body element exists here by giving it a background-color of black.

We can do this by adding the following to our style element:

body {

background-color: black;

}

* **Basic CSS: Inherit Styles from the Body Element**

Now we've proven that every HTML page has a body element, and that its body element can also be styled with CSS.

Remember, you can style your body element just like any other HTML element, and all your other elements will inherit your body element's styles.

First, create a h1 element with the text Hello World

Then, let's give all elements on your page the color of green by adding color: green; to your body element's style declaration.

Finally, give your body element the font-family of monospace by adding font-family: monospace; to your body element's style declaration.



* **Basic CSS: Prioritize One Style Over Another**

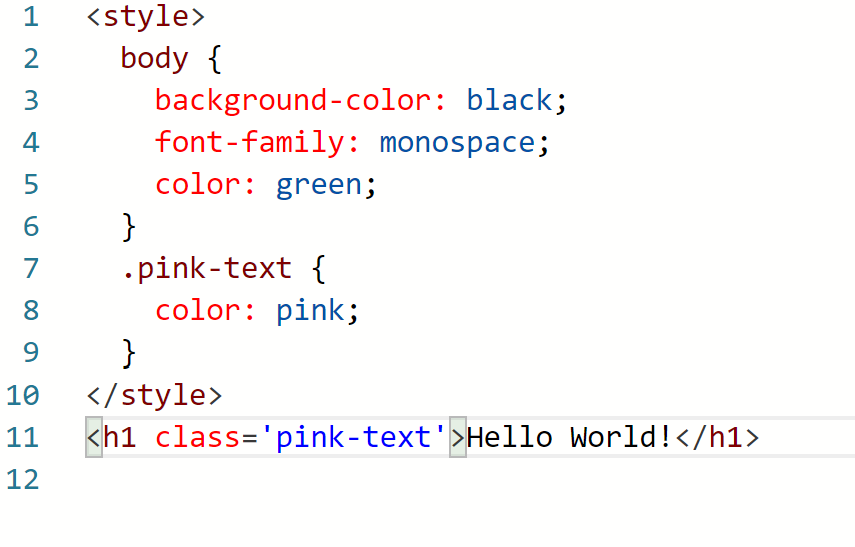
Sometimes your HTML elements will receive multiple styles that conflict with one another.

For example, your h1 element can't be both green and pink at the same time.

Let's see what happens when we create a class that makes text pink, then apply it to an element. Will our class *override* the body element's color: green; CSS property?

Create a CSS class called pink-text that gives an element the color pink.

Give your h1 element the class of pink-text. => class win



**Basic CSS: Override Styles in Subsequent CSS**

Our "pink-text" class overrode our body element's CSS declaration!

We just proved that our classes will override the body element's CSS. So the next logical question is, what can we do to override our pink-text class?

Create an additional CSS class called blue-text that gives an element the color blue. Make sure it's below your pink-text class declaration.

Apply the blue-text class to your h1 element in addition to your pink-text class, and let's see which one wins.

Applying multiple class attributes to a HTML element is done with a space between them like this:

class="class1 class2"

**Note:** It doesn't matter which order the classes are listed in the HTML element.

However, the order of the class declarations in the <style> section is what is important. The second declaration will always take precedence over the first. Because .blue-text is declared second, it overrides the attributes of .pink-text

However, the order of the class declarations in the <style> section is what is important. The second declaration will always take precedence over the first. Because .blue-text is declared second, it overrides the attributes of .pink-text

* **Basic CSS: Override Class Declarations by Styling ID Attributes**

We just proved that browsers read CSS from top to bottom in order of their declaration. That means that, in the event of a conflict, the browser will use whichever CSS declaration came last. Notice that if we even had put blue-text before pink-text in our h1 element's classes, it would still look at the declaration order and not the order of their use!

But we're not done yet. There are other ways that you can override CSS. Do you remember id attributes?

Let's override your pink-text and blue-text classes, and make your h1 element orange, by giving the h1 element an id and then styling that id.

Give your h1 element the id attribute of orange-text. Remember, id styles look like this:

<h1 id="orange-text">

Leave the blue-text and pink-text classes on your h1 element.

Create a CSS declaration for your orange-text id in your style element. Here's an example of what this looks like:

#brown-text {

color: brown;

}

**Note:** It doesn't matter whether you declare this CSS above or below pink-text class, since id attribute will always take precedence.

* **Basic CSS: Override Class Declarations with Inline Styles**

So we've proven that id declarations override class declarations, regardless of where they are declared in your style element CSS.

There are other ways that you can override CSS. Do you remember inline styles?

Use an inline style to try to make our h1 element white. Remember, in line styles look like this:

<h1 style="color: green;">

Leave the blue-text and pink-text classes on your h1 element.

* **Basic CSS: Override All Other Styles by using Important**

Yay! We just proved that inline styles will override all the CSS declarations in your style element.

But wait. There's one last way to override CSS. This is the most powerful method of all. But before we do it, let's talk about why you would ever want to override CSS.

In many situations, you will use CSS libraries. These may accidentally override your own CSS. So when you absolutely need to be sure that an element has specific CSS, you can use !important

Let's go all the way back to our pink-text class declaration. Remember that our pink-text class was overridden by subsequent class declarations, id declarations, and inline styles.

Let's add the keyword !important to your pink-text element's color declaration to make 100% sure that your h1 element will be pink.

An example of how to do this is:

color: red !important;

## Basic CSS: Use Hex Code for Specific Colors

Did you know there are other ways to represent colors in CSS? One of these ways is called hexadecimal code, or hex code for short.

We usually use decimals, or base 10 numbers, which use the symbols 0 to 9 for each digit. Hexadecimals (or hex) are base 16 numbers. This means it uses sixteen distinct symbols. Like decimals, the symbols 0-9 represent the values zero to nine. Then A,B,C,D,E,F represent the values ten to fifteen. Altogether, 0 to F can represent a digit in hexadecimal, giving us 16 total possible values. You can find more information about [hexadecimal numbers here](https://en.wikipedia.org/wiki/Hexadecimal).

In CSS, we can use 6 hexadecimal digits to represent colors, two each for the red (R), green (G), and blue (B) components. For example, #000000 is black and is also the lowest possible value. You can find more information about the [RGB color system here](https://en.wikipedia.org/wiki/RGB_color_model).

body {

color: #000000;

}

## Basic CSS: Use Hex Code to Mix Colors

To review, hex codes use 6 hexadecimal digits to represent colors, two each for red (R), green (G), and blue (B) components.

From these three pure colors (red, green, and blue), we can vary the amounts of each to create over 16 million other colors!

For example, orange is pure red, mixed with some green, and no blue. In hex code, this translates to being #FFA500.

The digit 0 is the lowest number in hex code, and represents a complete absence of color.

The digit F is the highest number in hex code, and represents the maximum possible brightness.

## Basic CSS: Use Abbreviated Hex Code

Many people feel overwhelmed by the possibilities of more than 16 million colors. And it's difficult to remember hex code. Fortunately, you can shorten it.

For example, red's hex code #FF0000 can be shortened to #F00. This shortened form gives one digit for red, one digit for green, and one digit for blue.

This reduces the total number of possible colors to around 4,000. But browsers will interpret #FF0000 and #F00 as exactly the same color.

## Basic CSS: Use RGB values to Color Elements

Another way you can represent colors in CSS is by using RGB values.

The RGB value for black looks like this:

rgb(0, 0, 0)

The RGB value for white looks like this:

rgb(255, 255, 255)

Instead of using six hexadecimal digits like you do with hex code, with RGB you specify the brightness of each color with a number between 0 and 255.

If you do the math, the two digits for one color equal 16 times 16, which gives us 256 total values. So RGB, which starts counting from zero, has the exact same number of possible values as hex code.

Here's an example of how you'd change the body background to orange using its RGB code.

body {

background-color: rgb(255, 165, 0);

}

## Basic CSS: Use RGB to Mix Colors

Just like with hex code, you can mix colors in RGB by using combinations of different values.

## Basic CSS: Use CSS Variables to change several elements at once

CSS Variables are a powerful way to change many CSS style properties at once by changing only one value.

Follow the instructions below to see how changing just three values can change the styling of many elements.

<https://github.com/yukawang27090/HOC-HTML/blob/master/canhcut.html>