Classes, IDs, & Divs

**Maintainable Code**

In this unit, you'll learn about a critical skill beneficial for all web developers: organizing HTML code. Being able to clearly organize HTML code helps in the following ways:

1. Writing clear, maintainable HTML code for yourself and for other developers.
2. Venturing into more advanced CSS styling.

In the last unit, you learned how to style color and fonts. Unfortunately, we had no way of targeting specific HTML elements. Instead, every selector we used targeted all elements of a certain type. By organizing HTML code with labels for elements, you'll learn how to style specific HTML elements at-will.

Let's begin!

**IDs: HTML**

With the proper labels, we can style individual HTML elements! Specifically, we can label HTML elements with a unique identifier, or *ID*. We can then style that specific element in the stylesheet.

To label an element with an ID, we can use the id attribute on an HTML element.

<h1 id="botswana">Botswana</h1>

In the example above, the heading is labeled with an id of botswana.

What purpose do IDs serve? IDs are intended to label unique elements in an HTML file. No two HTML elements should ever share the same ID — that would defeat the purpose of a unique identifier!

# ID Selectors: CSS

Now that you know how to label HTML elements with an ID, we can style that specific element in the stylesheet.

In a previous lesson, you learned how to target HTML elements with element selectors, like so:

p {

font-size: 10px;

font-color: #A091DD;

}

Remember, however, that an element selector like the one in the example above would targets all paragraphs on the web page. In the example above, all paragraphs are styled to have a font size of 10 pixels and a font color of #A091DD.

To style a specific element labeled with an ID, you can use an ID selector in the stylesheet.

#botswana {

background-color: #56ABFF;

}

In the example above, the HTML element with an ID of botswana is targeted with the ID selector #botswana.

All ID selectors begin with the octothorpe character: #. The value of the ID immediately follows the octothorpe. Once you've correctly targeted the element, you can proceed to style it as usual.

# Classes: HTML

IDs are great for labeling unique elements, but IDs have a limitation. Because unique IDs should not be used across multiple HTML elements, they are insufficient for quickly styling elements that should all share a specific style.

CSS offers a solution to this limitation. For multiple elements that should share the same styling, classes can be used to label them.

To label an element with a class, we can use the class attribute on an HTML element.

<h1 class="science">Scientist Discovers Important Cure</h1>

<h1 class="science">New Study Reveals The Importance of Sleep</h1>

In the example above, there are two headings with a class of science. Why?

HTML elements are often labeled with class names that reflect the content they represent on the web page. In the example above, perhaps a news company decided that all news headlines about science should be labeled with a class of science.

Later, in the stylesheet, all science headlines can be quickly styled to have the same styling.

# Class Selectors I: CSS

Now that you know how to label HTML elements with a class, we can style elements belonging to the same class at once. How exactly do you select them in CSS, though?

To style elements of the same class, you can use a class selector in the stylesheet.

.science {

font-family: Georgia, Times, serif;

color: #A3B4C5;

text-transform: uppercase;

}

Class selectors begin with a period (.) and are immediately followed by the name of the class. In the example above, all elements with a class of science will have their typeface, color, and letter case modified.

As you learn more about web development, you'll notice that it's very common to style groups of elements using classes. In fact, you're more likely to see classes more often than you see IDs. While IDs still play a critical role when you move into languages like JavaScript, classes are by far the most commonly used for styling groups of elements.

# Class Selectors II: CSS

The class selector targets all elements of a particular class. It's possible, however, for multiple elements on a web page to share a specific styling, but for one of those elements to differ slightly.

For example, a heading and a paragraph (both with a class of breaking) may need to share the same typeface, but the paragraph may require a styling better suited for paragraphs, as in the following example.

.breaking {

font-family: Georgia, Times, serif;

}

p.breaking {

line-height: 1.3em;

}

The example above uses a new selector: p.breaking. What's the difference between the .breaking and p.breaking selectors?

The .breaking selector targets all elements with a class of breaking. The p.breaking selector targets only <p> elements with a class of breaking. This type of selector allows you to be even more specific about a particular element's styling, even when that element must share some styling with other elements.

Unless otherwise specified, the rest of this course will use the element.class selector syntax.

# Multiple Selectors: CSS

CSS does not limit you to selectors that target a single element or class.

In a previous exercise, you learned how to use a multiple element selector to style multiple elements at once.

h1, p {

font-family: Garamond, serif;

}

The same syntax can be used to style multiple classes at once.

.first, .last {

font-size: 20px;

}

Using a multiple class selector is an efficient way of styling multiple HTML elements.

# Multiple Classes: HTML

It's also possible to label HTML elements with more than one class. How is this functionality useful?

You learned that elements that share a styling (e.g. typeface) are labeled with the same class. When those same elements must also be differentiated, however, labeling with an additional class is helpful.

<h1 class="book domestic">The Way of the Deep</h1>

<h1 class="book foreign">A Night in the Sky</h1>

.book {

font-family: Georgia, serif;

}

.domestic {

font-color: #0902CC;

}

.foreign {

font-color: #B097DD;

}

In the example above, the HTML code uses main headings for two different book titles. In this example, all book titles must share the same typeface, so the headings are labeled with a class of book and are then styled with a typeface of Georgia.

The books, however, must be differentiated based on their country of origin. To differentiate the book titles based on this information, two additional classes, domestic and foreign, are applied to the respective headings, which style the color of the heading in the CSS code.

To label an HTML element with an additional class, simply include the class within the double quotes, immediately after the first class. HTML elements can be labeled with many classes, but whenever possible, it's best to limit an element to four classes at most.

**The Div**

Classes and IDs are useful for labeling elements, but a disorganized, overuse of classes and IDs can result in an HTML file that is difficult to read. We need a way of organizing the contents of the HTML file. This will help:

1. Keep HTML code easy to read.
2. Group elements that belong together.

HTML offers an element that is the backbone of code organization: the *div*, represented by <div> in HTML.

You can think of the div as a box, or container, that groups elements that belong together.

For example, a <div> can be used to group together all of the elements that make up the navigation for a web page, or any other section of a page.

<div>

<h1>Alice In Wonderland</h1>

<p> ... </p>

</div>

In the example above, a heading for "Alice In Wonderland" and a brief paragraph description of the book are contained within a single <div>. If more books were to appear on the page, additional divs could be used to organize them. This would keep the code easier to read, maintain, and style.

# Divs + Classes

Now that you know how to organize code into sections using divs, we can start labeling divs with classes instead, rather than labeling individual elements with classes.

This does not mean that labeling individual elements with classes is no longer useful. Even when divs are labeled with classes, there will be many other times when an individual element will need to be labeled with a class.

<div class="container">

<h1 class="title">Alice In Wonderland</h1>

<p> ... </p>

</div>

div.container {

background-color: rgb(252, 255, 205);

font-family: Roboto, Helvetica, sans-serif;

}

h1.title {

color: #0D1A2F;

}

In the example above, a heading and paragraph are contained within a div that has a class of container. In the stylesheet, the background color and typeface of the div have been styled.

Divs are intended to contain other HTML elements, not raw text. Does it make sense to style a div directly then?

When a div is styled, all elements inside of the div will inherit the styling applied to the div. This example illustrates how easy it is to style sections of a web page using div.

**Note:** The example above uses a class called container. Chances are that you'll frequently see this class name as you learn more about web development.

# Why Divs?

The div is one of the most commonly used elements in all of HTML. Modern web pages make extensive use of the div, and learning how to use divs for organization and styling is a critical skill for all web developers.

Moving forward, this course will make more use of the div so that you can master it and become more familiar with this important element.