Agenda:

1. Introduction to CSS
2. CSS Anatomy
3. Ways to add CSS to a HTML file
4. CSS Selectors
5. CSS Colors
6. CSS Typography

Introduction to CSS:

We have learned about the HTML in last section that puts the layout and fundamental structure of a Web Page but with HTML alone it can b unappealing.

So, in order to make our web page more appealing we here going to start learning about CSS.

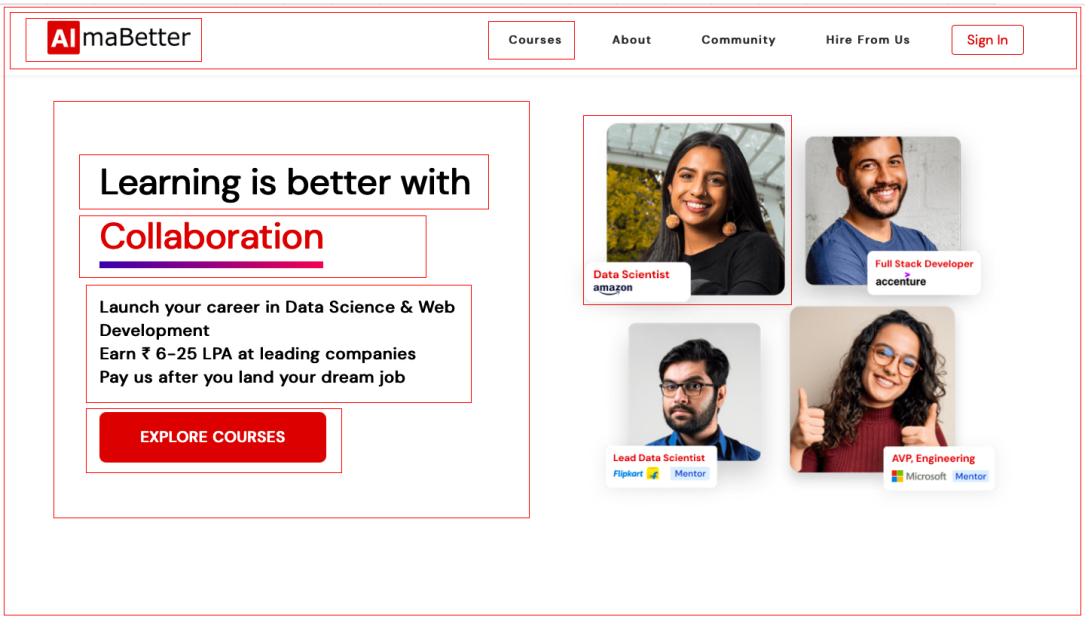
In this section, we will look at how to make your web pages more attractive, controlling the design of them using CSS.

Cascading Style Sheets or *CSS* is a language web developers use to style the HTML content on a web page. If you're interested in modifying colors, fo types, font sizes, images, element positioning, and more, CSS is the tool for the job!

U**nderstanding CSS: Thinking Inside The Box**

The key to understanding how CSS works is to imagine that there is an invisible box around every HTML element.

Consider the HTML Page shown below:



In the above AlmaBetter webpage, we have drawn box around the HTML elements, so we can think of every HTML element as a box which we will goir to style with CSS.

CSS allows you to create rules that control the way that each individual box (and the contents of that box) is presented.

In this example, block level elements are shown with red borders, and inline elements have green borders.

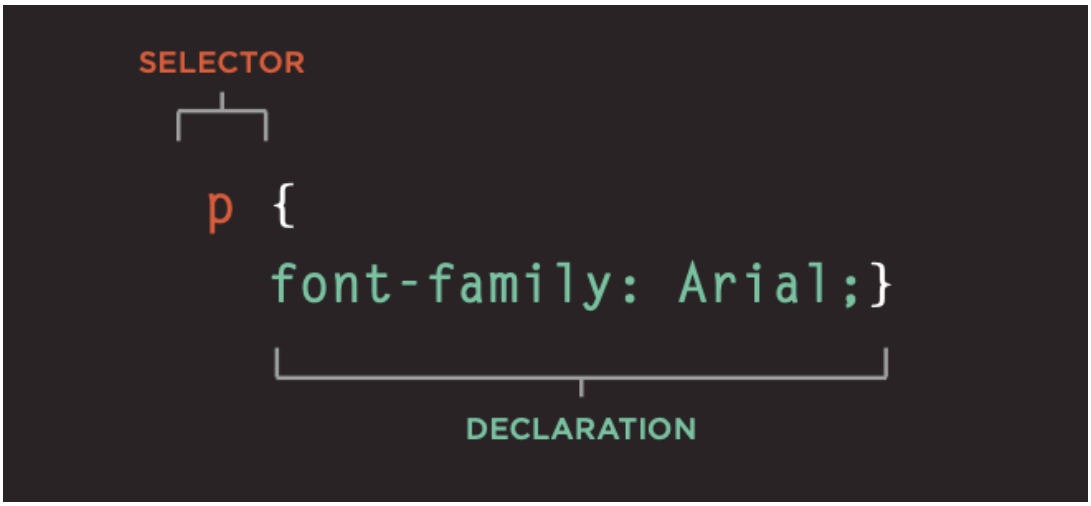
The `<body>` element creates the first box, then the `<h1>`, `<h2>`, `<p>`, `<b>`, and `<a>` elements each create their own boxes within it.

Using CSS, you could add a border around any of the boxes, specify its width and height, or add a background color. You could also control text inside box — for example, its color, size, and the typeface used.

CSS Anatomy

CSS associates styles rules with the HTML Elements:

CSS works by associating rules with HTML elements. These rules govern how the content of specified elements should be displayed. A CS rule contains two parts: a selector and a declaration.



This rule indicates that all

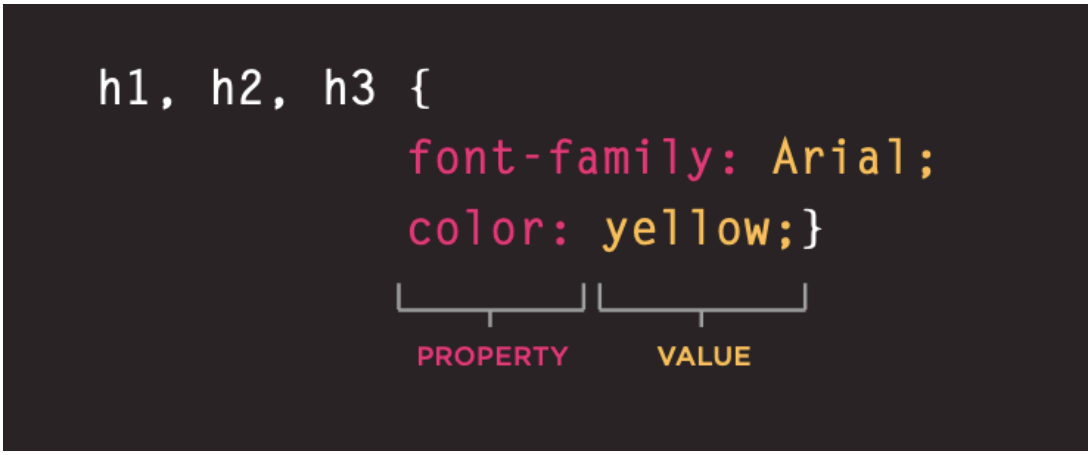
elements should be shown in the Arial typeface.

Selectors indicate which element the rule applies to. The same rule can apply to more than one element if you separate the element name with commas.

Declarations indicate how the elements referred to in the selector should be styled. Declarations are split into two parts (a property and value), and are separated by a colon.

CSS properties affect how elements are displayed:

CSS declarations sit inside curly brackets and each is made up of two parts: a property and a value, separated by a colon. You can speci several properties in one declaration, each separated by a semi-colon.



This rule indicates that all

,

and

elements should be shown in the Arial typeface, in a yellow color.

Properties indicate the aspects of the element you want to change. For example, color, font, width, height and border.

Values specify the settings you want to use for the chosen properties. For example, if you want to specify a color property then the value is the color you want the text in these elements to be.

Ways to add CSS to a HTML File:

We can add style to our HTML Element in any of the three ways defined below:

INLINE STYLING:

CSS styles can be directly added to HTML elements by using the `style` attribute in the element's opening tag. Each style declaration is ended with a semicolon. Styles added in this manner are known as *inline styles*.

```
<h2 style="text-align: center;">Centered text</h2>
```

```
<p style="color: blue; font-size: 18px;">Blue, 18-point text</p>
```

In the above example, we added a style to our h2 and p tag using inline styling in which a style attribute is added and css property we want to add to the element is specified within quotes.

For h2 tag we added a text-align property to center the text.

For p tag we added a color property to change color of text to blue and font-size property to change the font size of the text.

EXTERNAL STYLING:

In external styling the css is written in a different file from the HTML File and then that file is added to our HTML file through element defined in the element of the HTML file.

element:

The element can be used in an HTML document to tell the browser where to find the CSS file used to style the page. It is an empty element (meaning it does not need a closing tag), and it lives inside the element. It should use three attributes:

href: This specifies the path to the CSS file (which is often placed in a folder called **css** or **styles)**.

pe: This attribute specifies the type of document being linked to. The value should be **text/css**.

rel:

This specifies the relationship between the HTML page and the file it is linked to. The value should be **stylesheet** when linking to a CSS file.

HTML File:

```
<!DOCTYPE html>
<html>
<head>
    <title>Using External CSS</title>
    <link href="css/styles.css" type="text/css"
          rel="stylesheet" />
  </head>
<body>
    <h1>Externally linked css stylesheet.</h1>
</body>
</html>
```

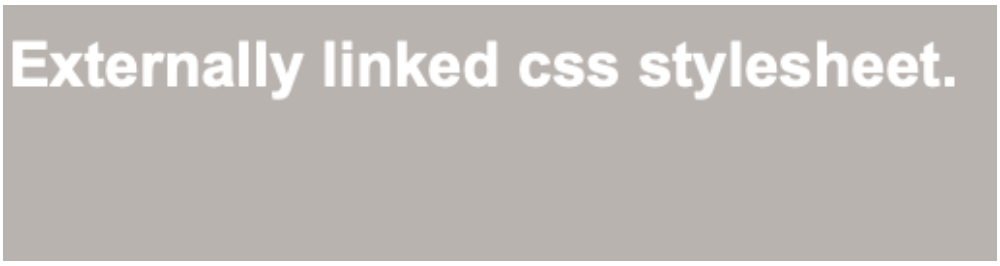
CSS File:

```
body {
    font-family: arial;
    background-color: rgb(185,179,175);}
h1 {
    color: rgb(255,255,255);}
```

In the above example , we are linking a external css file with the path specified in href attribute and type as “text/css” and rel as “stylesheet”.

And in the css file we added the code to convert background color of the complete body to the color specified and changed the color of the h1 tag to white.

Output:



An HTML page can use more than one CSS style sheet. To do this it could have a element for every CSS file it uses. For example, some developers use one CSS file to control the presentation (such as fonts and colors) and a second to control the layout.

INTERNAL STYLING:

We can add the internal styles to every HTML file within

You can also include CSS rules within an HTML page by placing them inside a element, which usually sits inside the element of the page.

The element should use the type attribute to indicate that the styles are specified in CSS. The value should be text/css.

```
<!DOCTYPE html>
<html>
<head>
    <title>Using Internal CSS</title>
    <style type="text/css">
      body {
          font-family: arial;
          background-color: rgb(185,179,175);}
      h1 {
          color: rgb(255,255,255);}
    </style>
  </head>
<body>
    <h1>Styling done by Internal styles</h1>
</body>
</html>
```

In the above example, we added the internal styles to our HTML file in which we changed the background color of the body and changed the color of heading to white.

****Output:**

Note:

When building a website with more than one page, you should use an external CSS style sheet. This:

- **Allows all pages to use the same style rules (rather than repeating them in each page).**
- **Keeps the content separate from how the page looks.**
- **Means you can change the styles used across all pages by altering just one file (rather than each individual page).**

CSS Selectors:

There are many different types of CSS selector that allow you to target rules to specific elements in an HTML document.

Let's start on the types of selectors through which we can select specific elements in a HTML File:

Universal Selector:

The *universal selector* selects all elements of *any* type on a HTML file.

Targeting all of the elements on the page has a few specific use cases, such as resetting default browser styling, or selecting all children of a parent element.

The universal selector uses the `*` character in the same place where you specified the type selector in a ruleset, like so:

```
* {
  font-family: Verdana;
```

```
}
```

In the code above, every text element on the page will have its font changed to `Verdana` .

Type Selector:

A selector is used to target the specific HTML element(s) to be styled by the declaration. Just like its name suggests, the type selector matches the *type* of the element in the HTML document.

It targets all the designated tag of the whole HTML file.

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

In the above code, we have selected the p tag that specifies the

paragraph tag of the HTML file , through this code all the

tags in HTML file will change their text color to red.

Some important notes on the type selector:

- **The type selector does not include the angle brackets.**
- **Since element types are often referred to by their opening tag name, the type selector is sometimes referred to as the *tag name* or *element* selector.**

Class Selector:

CSS is not limited to selecting elements by their type. As you know, HTML elements can also have attributes. When working with HTML and CSS a *class* attribute is one of the most common ways to select an element.

To select an HTML element by its class using CSS, a period (`.`) must be prepended to the class's name.

Let's understand this with an example:

HTML File:

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

The paragraph element in the example above has a `class` attribute within the opening tag of the `<p>` element. The `class` attribute is set to `'brand'` . To select this element using CSS, we can create a ruleset with a class selector of `.brand` .

Css :

```
.brand {
    color: red;
    }
```

In the above example, we. have selected the elements to which we have specified class as brand through this we can change the text color of every element to which we specified the class brand to red.

We can specify a class of similar type to any number of elements on a HTML File, this helps us to style a particular amount of elements to a specific style.

Id Selector:

Oftentimes it's important to select a single element with CSS to give it its own unique style. If an HTML element needs to be styled uniquely, we can give it an ID using the `id` attribute.

In contrast to `class` which accepts multiple values, and can be used broadly throughout an HTML document, an element's `id` can only have a single value, and only be used once per page.

To select an element's ID with CSS, we prepend the `id` name with a number sign (`#`).

Let's understand this through an example:

HTML:

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

Here, we have an h1 HTML tag to which we have specified a id attribute with the value 'large-title'.

CSS:

```
#large-title {
     color: red;
}
```

In the above example, we have selected the HTML tag specified with id attribute of value 'large-title' and specified a style to it that it's text color to be red.

Attribute Selector:

You may remember that some HTML elements use attributes to add extra detail or functionality to the element. Some familiar attributes may be `href` and `src` , but there are many more —including `class` and `id` !

The *attribute selector* can be used to target HTML elements that already contain attributes. Elements of the same type can be targeted differently by their attribute or attribute value. This alleviates the need to add new code, like the `class` or `id` attributes.

The most basic syntax is an attribute surrounded by square brackets.

Let's see this through an example:

```
[href]{
   color: magneta;
}
```

In the above example: `[href]` would target all elements with an `href` attribute and set the `color` to `magenta` .

There is one another way to select the attribute element that is by using `type[attribute*=value]` .In short, this code selects an element where the attribute contains any instance of the specified value.

Let's consider the example shown below:

HTML :

```
<img src='/images/seasons/cold/winter.jpg'>
<img src='/images/seasons/warm/summer.jpg'>
```

The HTML code above renders two `<img>` elements, each containing a `src` attribute with a value equaling a link to an image file.

Css:

```
img[src*='winter'] {
  height: 50px;
}

img[src*='summer'] {
  height: 100px;
}
```

Now take a look at the above CSS code. The *attribute selector* is used to target each image individually.

- The first ruleset looks for an `img` element with an attribute of `src` that contains the string `'winter'`, and sets the `height` to `50px`.
- The second ruleset looks for an `img` element with an attribute of `src` that contains the string `'summer'`, and sets the `height` to `100px`.

Pseudo-Class Selector:

You may have observed how the appearance of certain elements can change, or be in a different state, after certain user interactions. For instance:

- When you click on an `<input>` element, and a blue border is added showing that it is in *focus*.
- When you click on a blue `<a>` link to *visit* to another page, but when you return the link's text is purple.
- When you're filling out a form and the submit button is grayed out and *disabled*. But when all of the fields have been filled out, the button has color showing that it's *active*.

These are all examples of pseudo-class selectors in action! In fact, `:focus`, `:visited`, `:disabled`, and `:active` are all pseudo-classes. Factors such as user interaction, site navigation, and position in the document tree can all give elements a different state with pseudo-class.

A pseudo-class can be attached to any selector. It is always written as a colon `:` followed by a name.

Let's consider an example shown below:

```
p:hover {
  background-color: lime;
}
```

In the above code, whenever the mouse hovers over a paragraph element, that paragraph will have a lime-colored background.

These are the some important and mostly used selectors through which you can select an element to style it and let's look at the specificity of these selectors:

Specificity of css selectors:

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, type.

Let's take an example:

HTML:

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

CSS:

```
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 type 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.

Note:

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

Chaining of css 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.

Let's take an example where there was a `special` class for `<h1>` elements.

```
h1.special {

}
```

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

Descendant Combinator:

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

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

The nested `<li>` elements are descendants of the `<ul>` element and can be selected with the *descendant combinator* like: ``

```
.main-list li {

}
```

In the example above, `.main-list` selects the element with the `.main-list` class (the `<ul>` element). The descendant `<li>`'s are selected by adding `li` to the selector, separated by a space. This results in `.main-list li` as the final selector.

Chaining and Specificity:

Adding more than one tag, class, or ID to a CSS selector increases the specificity of the CSS selector.

Let's understand this through an example:

```
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` type 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`.

Adding styles to 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.

Consider the following code:

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

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

The above code has repetitive style attributes as we are specifying the same style to both h1 and menu. So, there is an another way through which we can do this in an effective manner,

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 consider the code below:

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

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

Till now we have become quite familiar with the working of css and the ways to select an element in order to style it.

So, now we are going to start with css colors:

CSS Colors:

CSS supports a wide variety of colors. These include *named colors*, like `blue` , `black` , and `limegreen` , along with colors described by a numeric value. Using a numeric system allows us to take advantage of the whole spectrum of colors that browsers support. In this lesson, we're going to explore all the color options CSS offers.

Colors in CSS can be described in three different ways:

- ***Named colors* — English words that describe colors, also called *keyword colors***
- ***RGB* — numeric values that describe a mix of red, green, and blue**
- ***HSL* — numeric values that describe a mix of hue, saturation, and lightness**

Before we start to look at the colors deeply, firstly let's learn about the foreground and background colors concept in css:

Foreground and Background:

Color can affect the following design aspects:

- **The foreground color**
- **The 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.**

Let's see this through an example:

```
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.

Now after understanding this let's take a deep look at css colors :

Hexadecimal Color scheme:

One syntax that we can use to specify colors is called *hexadecimal*. Colors specified using this system are called *hex colors*. A hex color begins with a hash character (#) which is followed by three or six characters. The characters represent values for red, blue and green.

Consider the example shown below:

```
darkseagreen: #8FBC8F
sienna:       #A0522D
saddlebrown:  #8B4513
brown:        #A52A2A
black:        #000000 or #000
white:        #FFFFFF or #FFF
aqua:         #00FFFF or #0FF
```

In the example above, you may notice that there are both letters and numbers in the values. This is because the hexadecimal number system has 16 digits (0-15) instead of 10 (0-9) like in the standard decimal system. To represent 10-15, we use A-F.

To see all the colors with their hex code click Here is a list of many different colors and their hex values.

RGB Color scheme:

There is another syntax for representing RGB values, commonly referred to as "RGB value" or just "RGB", that uses decimal numbers rather than hexadecimal numbers, and it looks like this:

```
h1 {
  color: rgb(23, 45, 23);
}
```

Each of the three values represents a color component, and each can have a decimal number value from 0 to 255. The first number represents the amount of red, the second is green, and the third is blue. These colors are exactly the same as hex, but with a different syntax and a different number system.

Hue, Saturation and Lightness color scheme:

The syntax for HSL is similar to the decimal form of RGB, though it differs in important ways. The first number represents the degree of the hue, and can be between 0 and 360. The second and third numbers are percentages representing saturation and lightness respectively.

Consider the example shown below:

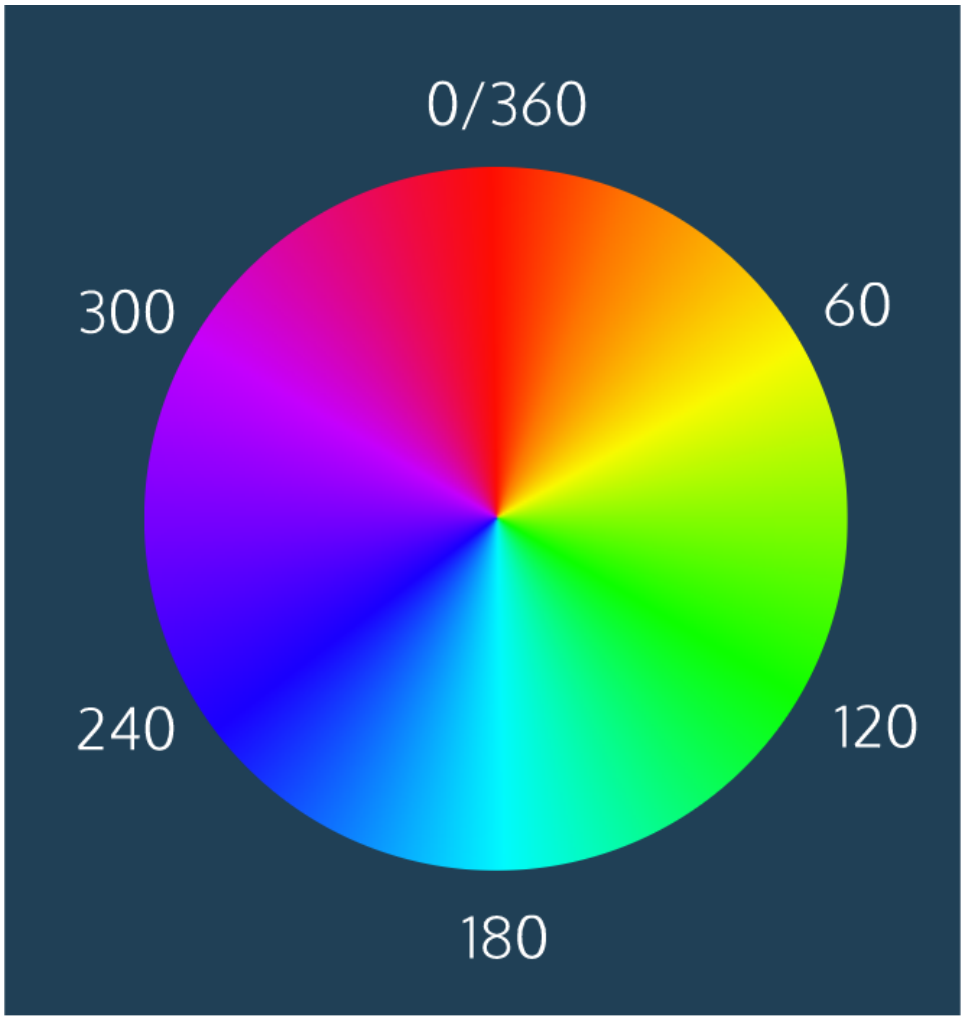
```
color: hsl(120, 60%, 70%);
```

Let's look one by one on hue, saturation and lightness:

Hue:

Hue is the first number. It refers to an angle on a color wheel. Red is 0 degrees, Green is 120 degrees, Blue is 240 degrees, and then back to Red at 360.

You can see an example of color wheel below:



Saturation:

Saturation **refers to the intensity or purity of the color. The saturation increases towards 100% as the color becomes richer. The saturation decreases towards 0% as the color becomes grayer.**

Lightness:

Lightness **refers to how light or dark the color is. Halfway, or 50%, is normal lightness. Imagine a sliding dimmer on a light switch that starts halfway. Sliding the dimmer up towards 100% makes the color lighter, closer to white. Sliding the dimmer down towards 0% makes the color darker, closer to black.**

HSL is convenient for adjusting colors. In RGB, making the color a little darker may affect all three color components. In HSL, that's as easy as changing the lightness value. HSL is also useful for making a set of colors that work well together by selecting various colors that have the same lightness and saturation but different hues.

So, till now we have seen about the opaque colors , but now let's take a look at the opactiy and alpha properties of css colors:

Opacity and Alpha:

All of the colors we've seen so far have been opaque, or non-transparent. When we overlap two opaque elements, nothing from the bottom element shows through the top element.

Now, we will change the opacity, or amount of transparency, of some colors so that some or all of the bottom elements are visible through a covering element.

To use opacity in the HSL color scheme, use `hsla` **instead of** `hsl` **, and four values instead of three.**

Refer to the example shown below:

```
color: hsla(34, 100%, 50%, 0.1);
```

The first three values work the same as `hsl`. The fourth value (which we have not seen before) is the *alpha*. This last value is sometimes called opacity.

Alpha is a decimal number from zero to one. If alpha is zero, the color will be completely transparent. If alpha is one, the color will be opaque. The value for half-transparent would be `0.5`.

You can think of the alpha value as, "the amount of the background to mix with the foreground". When a color's alpha is below one, any color behind it will be blended in. The blending happens for each pixel; no blurring occurs.

The RGB color scheme has a similar syntax for opacity, `rgba`. Again, the first three values work the same as `rgb` and the last value is the alpha.

Take a look at the example shown below:

```
color: rgba(234, 45, 98, 0.33);
```

A little unconventional, but still worth mentioning is how hex colors can also have an alpha value. By adding a two-digit hexadecimal value to the end of the six-digit representation (`#52BC8280`), or a one-digit hexadecimal value to the end of the three-digit representation (#F003), you can change the opacity of a hexadecimal color. Hex opacity ranges from `00` (transparent) to `FF` (opaque).

Alpha can only be used with HSL, RGB, and hex colors; we cannot add the alpha value to name colors like `green`.

There is, however, a named color keyword for zero opacity, `transparent`. It's equivalent to `rgba(0, 0, 0, 0)`, and it's used like any other color keyword:``

```
color: transparent;
```

So, this is all about css colors , now let's move on with the flow and starting looking into the css typography:

Css Typography:

So from now on, we will be focusing on typography, the art of arranging text on a page. We'll look at:

- How to style and transform fonts.
- How to lay text out on a page.
- and how to add external fonts to your web pages.

In typography , we are going to start with the font-family:

Font family:

We can change the font of an element using font-family property in css.

For example:

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

In the example above, the font family for all `<h1>` heading elements have been set to Arial.

Using Multi-word value:

When specifying a typeface with multiple words, like Times New Roman, it is recommended to use quotation marks (`' '`) to group the words together, for example:

```
h1 {
  font-family: 'Times New Roman';
}
```

Font Weight:

In CSS, the `font-weight` property controls how bold or thin text appears. It can be specified with keywords or numerical values.

Keyword Values

The `font-weight` property can take any one of these keyword values:

- `bold` : Bold font weight.
- `normal` : Normal font weight. This is the default value.
- `lighter` : One font weight lighter than the element's parent value.
- `bolder` : One font weight bolder than the element's parent value

Numerical Values

Numerical values can range from 1 (lightest) to 1000 (boldest), but it is common practice to use increments of 100. A font weight of `400` is equal to the keyword value `normal` , and a font weight of `700` is equal to `bold` .

Let's take a look at a example to understand it better:

```
.left-section {
  font-weight: 700;
}

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

In the example above, text in elements of both `.left-section` and `.right-section` classes will appear bold.

Font Style

The `font-style` property specifies the font style for a text.

It can take values: normal, italic, oblique, initial and inherit.

For example:

```
h3 {
  font-style: italic;
}
```

The `italic` value causes text to appear in italics. The `font-style` property also has a `normal` value which is the default.

Text Transformation

Text can also be styled to appear in either all uppercase or lowercase with the `text-transform` property.

For example:

```
h1 {
  text-transform: uppercase;
}
```

he code in the example above formats all `<h1>` elements to appear in `uppercase` , regardless of the case used for the heading within the HTML code. Alternatively, the `lowercase` value could be used to format text in all lowercase.

Depending on the type of content a web page displays, it may make sense to always style a specific element in all uppercase or lowercase letters. For example, a website that reports breaking news may decide to format all `<h1>` heading elements such that they always appear in all uppercase, as in the example above. It would also avoid uppercase text in the HTML file, which could make code difficult to read.

Text Layout

You've learned how text can be defined by font family, weight, style, and transformations. Now you'll learn about some ways text can be displayed or laid out within the element's container.

Letter Spacing

The `letter-spacing` property is used to set the horizontal spacing between the individual characters in an element. It's not common to set the spacing between letters, but it can sometimes help the readability of certain fonts or styles. The `letter-spacing` property takes length values in units, such as `2px` or `0.5em` .

For example:

```
p {
  letter-spacing: 2px;
}
```

In the example above, each character in the paragraph element will be separated by 2 pixels.

Word Spacing

You can set the space between words with the `word-spacing` property. It's also not common to increase the spacing between words, but it may help enhance the readability of bolded or enlarged text. The `word-spacing` property also takes length values in units, such as `3px` or `0.2em` .

```
h1 {
  word-spacing: 0.3em;
}
```

In the example above, the word spacing is set to `0.3em` . For word spacing, using `em` values are recommended because the spacing can be set based on the size of the font.

Line Height

We can use the `line-height` property to set how tall we want each line containing our text to be. Line height values can be a unitless number, such as `1.2` , or a length value, such as `12px` , `5%` or `2em` .

```
p {
  line-height: 1.4;
}
```

In the example above, the height between lines is set to `1.4` . Generally, the unitless value is preferred since it is responsive based on the current font size. In other words, if the `line-height` is specified by a unitless number, changing the font size will automatically readjust the line height.

Text Alignment

The `text-align` property, aligns text to its parent element.

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

In the example above, the `<h1>` element is aligned to the right side, instead of the default left.

Now, let's take an complete example of the css topics we have learned so far:

so consider the code below:

```
<!DOCTYPE html>
<html>
```

```
<head>
    <title>Text</title>
    <style type="text/css">
          body {
            padding: 20px;}
          h1, h2, h3, a {
            font-weight: normal;
            color: #0088dd;
            margin: 0px;}
          h1 {
            font-family: Georgia, Times, serif;
            font-size: 250%;
            padding-bottom: 10px;}
          h2 {
            font-family: "Gill Sans", Arial, sans-serif;
            font-size: 90%;
            text-transform: uppercase;
            letter-spacing: 0.2em;}
          h3 {
            font-size: 150%;}
              p{
              font-family: Arial, Verdana, sans-serif; line-height: 1.4em;
              color: #665544;}
                    p.intro:first-line {
                      font-weight: bold;}
                    .credits {
                      font-style: italic;
                      text-align: right;}
              .history {
          background-color: lightgray;
        }
                a{
                text-decoration: none;}
                      a:hover {
                        text-decoration: underline;}
    </style>
  </head>
    <body>
        <h1>Briards</h1>
        <h2>A Heart wrapped in fur</h2>
        <p class="intro">The <a class="breed"
                 href="http://en.wikipedia.org/wikiBriard">briard</a>,
                or berger de brie, is a large breed of dog traditionally used as a
                herder and guardian of sheep.</p>
        <h3>Breed History</h3>
        <p>The briard, which is believed to have originated in France,
             has been bred for centuries to herd and to protect sheep.
                The breed was used by the French Army as sentries,
                messengers and to search for wounded soldiers because of its
                fine sense of hearing. Briards were used in the First World War
                almost to the point of extinction. Currently the population of briards
                slowly recovering. Charlemagne, Napoleon, Thomas Jefferson
                all owned briards.</p>
        <p class="credits">by Ivy Duckett</p>
    </body>
    </html>
```

Output:

So, in this module we have looked and learned about the basics of the CSS that can be applied to a HTML file for basic styling. In the next module we are going to look at the deeper properties and concepts of the CSS through which we can style our web page as we needed.

Interview Questions

> **How can CSS be integrated into an HTML page?**

There are three ways of integrating CSS into HTML: using style tags in the head section, using inline-styling, writing CSS in a separate file, and linking into the HTML page by the link tag.

> **What is meant by RGB stream?**

RGB represents colors in CSS. The three streams are namely Red, Green, and Blue. The intensity of colors is represented using numbers 0 to 256. This allows CSS to have a spectrum of visible colors.

> **What is the difference between a class and an ID?**

Class is a way of using HTML elements for styling. They are not unique and have multiple elements. Whereas ID is unique and it can be assigned to a single element.

Thank You !

Thank You !