To style an element add the style tag and can be styled by the code as:

<style>

      h1{

        text-align : center

      }

    </style>

We gave this style in the head area as we did not want to show the data.

Text-align will align the text the way we want to do we pu center means it puts text within the center area of the screen

Refer to the following code

<!DOCTYPE html>

<html lang="en">

  <head>

    <meta charset="utf-8" />

    <title>Cafe Menu</title>

    <style>

      h1 {

        text-align: center;

      }

      h2{

        text-align: center;

      }

      p{

        text-align: center;

      }

    </style>

  </head>

  <body>

    <main>

      <h1>CAMPER CAFE</h1>

      <p>Est. 2020</p>

      <section>

        <h2>Coffee</h2>

      </section>

    </main>

  </body>

</html>

Here I have made multiple text align by just giving the tag detail as the tag name.

We also group the same styling texts by separating by comma as:

<style>

      h1,h2,p {

        text-align: center;

      }

    </style>

WE make a separate file for css usually as we can have various attributes but the style tag also does the job but makes code long.

To link another css file into the html file use self closing link tag as code:

    <link rel=stylesheet href=styles.css>

While styling the style may restrict to the device so to generalize the view to all devices use the following meta attribute in the head

    <meta name="viewport" content="width=device-width, initial-scale=1.0"/>

This makes view more generalized.

To change the background color we use the normal background-color attribute

As:

body {

  background-color: brown;

}

Div tag is used to basically design layout purposes

<div>

      <main>

        <h1>CAMPER CAFE</h1>

        <p>Est. 2020</p>

        <section>

          <h2>Coffee</h2>

        </section>

      </main>

    </div>

We don’t want the div to take whole space so we can set the width also as:

div {

  width: 300px;

}

If we move background from body to div then only the div will be colored not the whole body,

If we change width from 300 px to 80% we observer that 300 px is fixed by by mentioning 80% we set the width to be 80% of body width. So 80% makes it dynamic

Unlike referring in px.

Now to center align the whole div use the margin-left and margin-right and set them to auto,

Think of the margin as invisible space around an element. Using these two margin properties, center the div element within the body element.

As:

div {

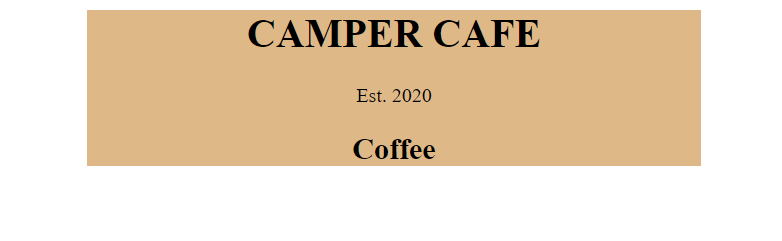
  margin-left: auto;

  margin-right: auto;

  width: 80%;

  background-color: burlywood;

}



To refer many tags we use class name as

.class-name{}

As:

<div class="menu">

This change is to be made in html file

Then we can refer the class name as:

.menu {

  width: 80%;

  background-color: burlywood;

  margin-left: auto;

  margin-right: auto;

}

A background image can be added by background-image attribute as:

body {

  background-image: url(https://cdn.freecodecamp.org/curriculum/css-cafe/beans.jpg)

}

We use article to refer the items and can be styled accordingly but we observer that the price and the flavor do not come in the same line.

We do this to the article

<article class="item">

            <p class="flavor">French Vanilla</p>

            <p class="price">3.00</p>

Then in css following changes are made:

.item p {

  display: inline-block

}

But using the inline block takes only the width of the content so we have to fix it up by with attribute as:

.flavor {

  text-align: left;

  width: 49%;

}

.price {

  text-align: right;

  width: 49%;

}

We use padding left and right to give some space b/w the border and the text.

.menu {

  padding-left: 20px;

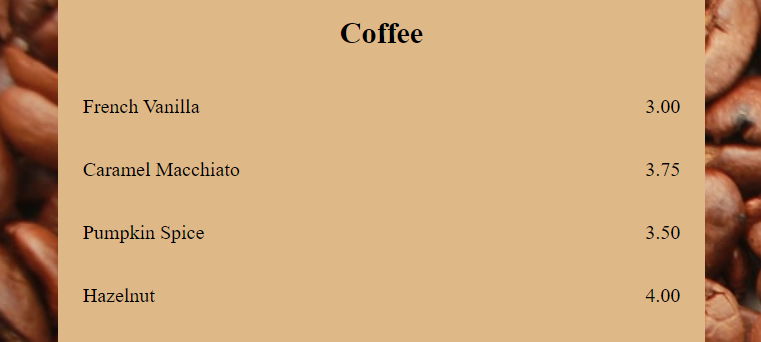
  padding-right: 20px;

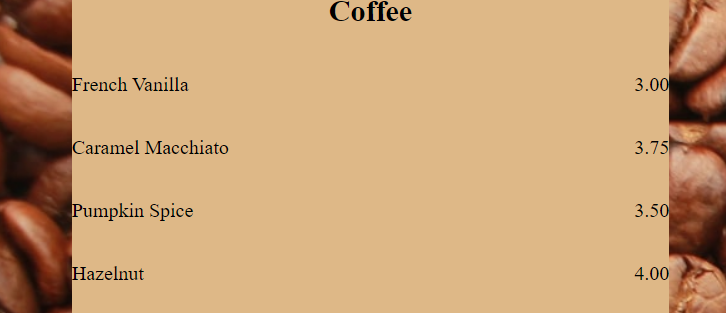
  width: 80%;

  background-color: burlywood;

  margin-left: auto;

  marg

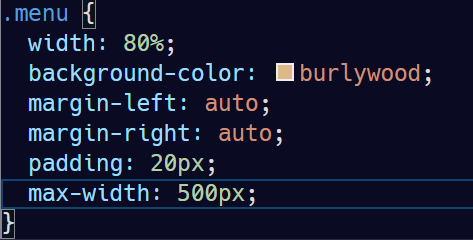




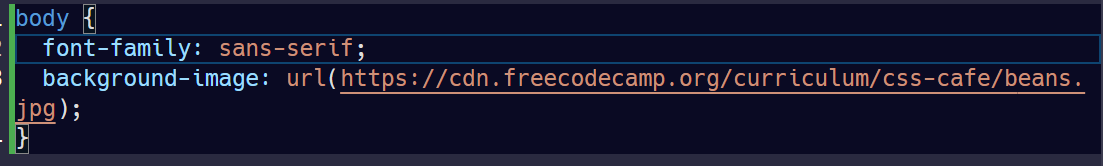
The first shows after padding and the second shows before adding attribute.

We can have padding-left, padding-right, padding-top, padding-bottom and if all have same value we can just use the padding attribute.

We added a max-width property to the style as when we increase size of screen it will always cover up 80% so we set a max size after which it does not extend.



We can also change the font by using font-family attribute as :



You can add a fallback value for the font-family by adding another font name separated by a comma. Fallbacks are used in instances where the initial is not found/available.

h1, h2 {

  font-family: Impact,serif;

}

We can also style font by using the font-style property.

.established {

  font-style: italic;

}

We can also give font size:

h1{

  font-size: 40px;

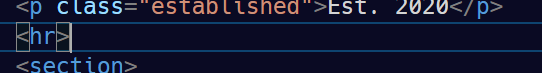
}

h2{

  font-size: 30px;

}

To display a divider line b/w section use <hr> hr tags are self closing.





We can change height and other styling features over the hr tag

As:

hr {

  height: 3px;

  background-color: brown;

  border-color: brown;

}

We also see that the white grey lines are border and we can change their color by border-color feature.

We can also change the border-width using border width property.

Flexbox has a main and cross axis. The main axis is defined by the flex-direction property, which has four possible values:

* row (default): horizontal axis with flex items from left to right
* row-reverse: horizontal axis with flex items from right to left
* column: vertical axis with flex items from top to bottom
* column-reverse: vertical axis with flex items from bottom to top

**Note**: The axes and directions will be different depending on the text direction. The values shown are for a left-to-right text direction.

Try the different values to see how they affect the layout.

When you are done, set an explicit flex-direction of row on the .gallery element.

Which is basically achieved by putting up display as flex

.gallery {

  display: flex;

}

The flex-wrap property determines how your flex items behave when the flex container is too small. Setting it to wrap will allow the items to wrap to the next row or column. nowrap (default) will prevent your items from wrapping and shrink them if needed.

Make it so your flex items wrap to the next row when they run out of space.

The justify-content property determines how the items inside a flex container are positioned along the main axis, affecting their position and the space around them.

Give your .gallery selector a justify-content property with center as the value.

The align-items property positions the flex content along the cross axis. In this case, with your flex-direction set to row, your cross axis would be vertical.

To vertically center your images, give your .gallery selector an align-items property with center as the value.

Notice how some of your images have become distorted. This is because the images have different aspect ratios. Rather than setting each aspect ratio individually, you can use the object-fit property to determine how images should behave.

Give your .gallery img selector the object-fit property and set it to cover. This will tell the image to fill the img container while maintaining aspect ratio, resulting in cropping to fit.

Your images need some space between them.

The gap CSS shorthand property sets the gaps, also knowns as gutters, between rows and columns. The gap property and its row-gap and column-gap sub-properties provide this functionality for flex, grid, and multi-column layout. You apply the property to the container element.

Give your .gallery flex container a gap property with 16px as the value.

Smooth out your images a bit by giving the .gallery img selector a border-radius property with 10px set as the value.

The ::after pseudo-element creates an element that is the last child of the selected element. You can use it to add an empty element after the last image. If you give it the same width as the images it will push the last image to the left when the gallery is in a two-column layout. Right now, it is in the center because you set justify-content: center on the flex container.

Example:

.container::after {

content: "";

width: 860px;

}

Create a new selector using an ::after pseudo-element on the .gallery element. Add a content property set to an empty string "" and 350px set for the width property.