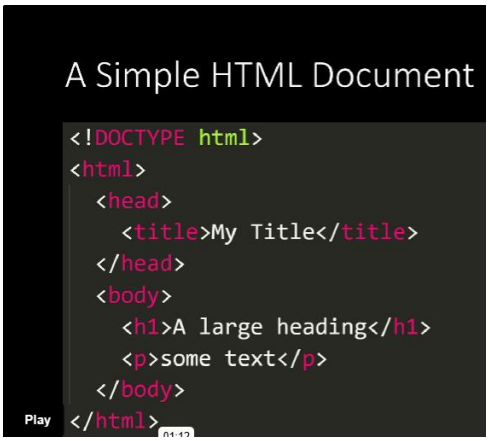
HTML CRASH COURSE

HTML INTRODUCTION

What is HTML

A Simple HTML Document



A tag is what is inside the angled brackets. The first tag describes the type of document. It's a special tag at the beginning of the document to tell the browser what type of code it can be expecting and what the document is going to be. In this case, the type is HTML.

The second tag is what is known as an HTML tag. It is an opening tag, which has the double brackets and is located at the beginning of the document. The closing tag will be at the end of the document and will contain not only the brackets but also the forward slash (/) . In this case </html>

Typically, all HTML documents have an opening and closing tag but on rare occasions, the document will contain a single tag acting as an opening and closing tag.

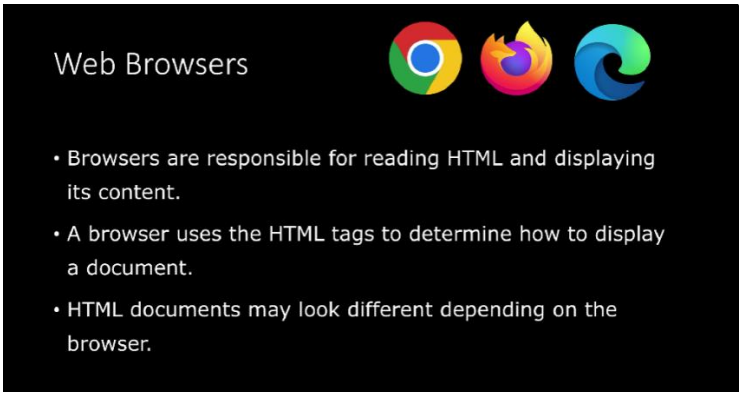
Inside the HTML tag is the HTML document. It can be called file in some cases.

After the opening tag, we have the head tag where we include everything that's not going to be displayed on the web page. It's not the main primary content. It is typically metadata or information about the web page. In this case we have our head tag and then title tag. Inside the title tag is going to be My Title because that's what we want the title to be.

After the title tag, we have the body tag. The body is what is going to include all the main content for our webpage. In this case, h1 means header 1, which is a large heading. Then we have the paragraph.

<h1>A large heading </h1> This whole line contains an opening tag, a content and a closing tag. The whole line is what we call HTML element.

Web Browsers



HTML Files/ Editors

HTML Files end with the extension: .html

You can create .html documents using any text editor you'd like, however, there are special editors that make this process easier.

Visiting a Website

When you are visiting a website, you are viewing rendered HTML. This HTML is downloaded by your local browser from the server where the website is hosted. The website is located on a server that is hosting it and is connected to the internet. The Browser then requests to view the website and the request is granted by the server. The browser downloads the HTML document and reads it to allow you to view it.

HTML ENVIRONMENT SETUP

Code editor / IDE Integrated development Environment

VS Code = Visual Studio Code

File – open folder – select folder (or create one)

Create new file

How does visual studio code work?

This program is popular because it has a lot of customization features available (different shortcuts, different themes etc)

Extensions:

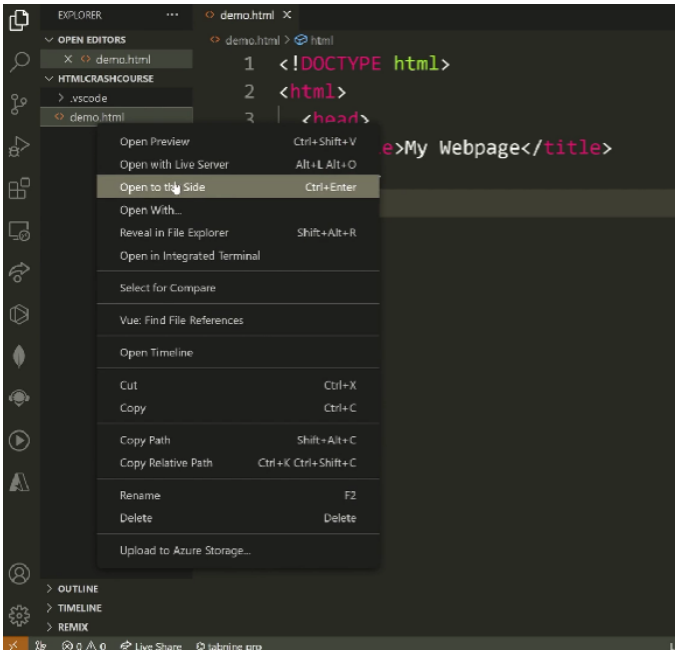
Html CSS Support

Prettier: Code formator. It will automatically format the code once you save the file.

Live Server

Shortcut to open the command pallet. Allows you to have different visuals. CTRL + Shift + P

To view the HTML document we're working on, click reveal in File explorer

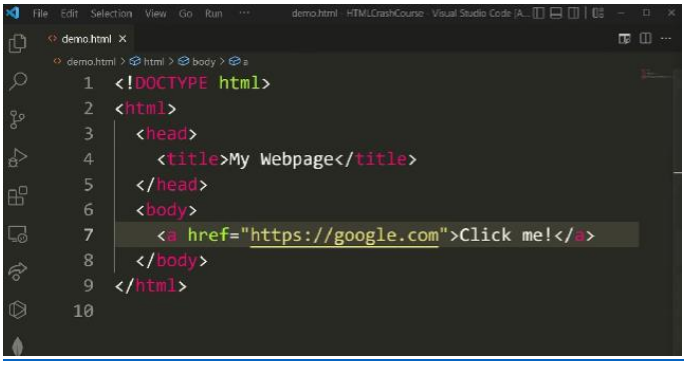


MOST COMMON HTML TAGS

  paragraph

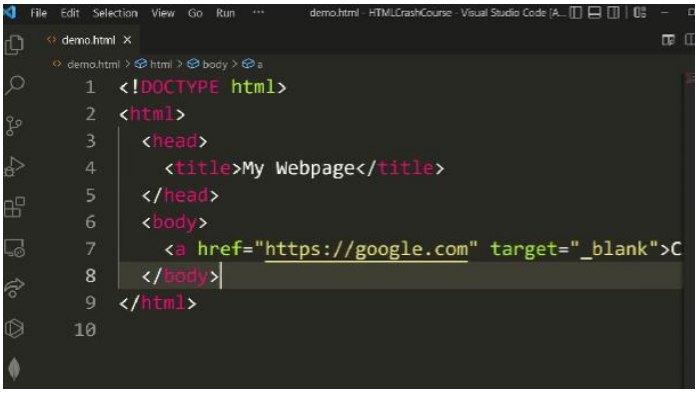
<h1></h1> header = large header

<a> Link tag allows you to link to another part of your webpage. To be able to click on another page we must include the term href. For example: <a href=https://google.com

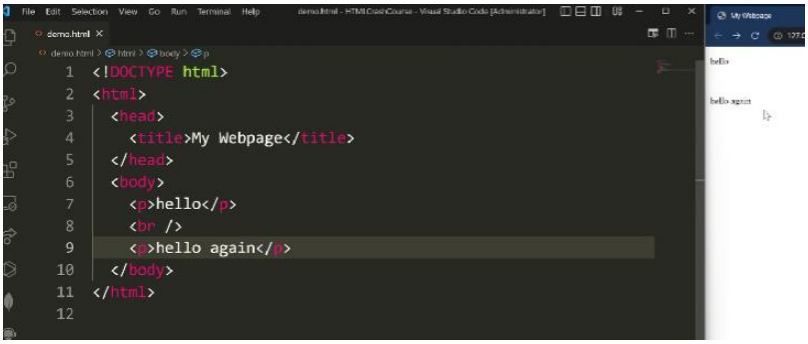


If we want to open a new tab for that link instead of the actual root page being changed, we'll add target.

Example: <a href=https://google.com target="blank"



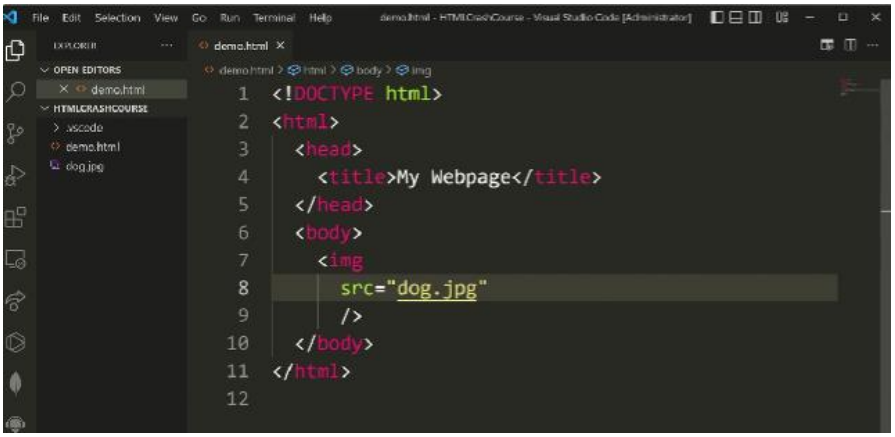
 breakline. Special tag because we don't have any closing tab. The tag breaks the line. Give us some space between the lines, paragraphs.



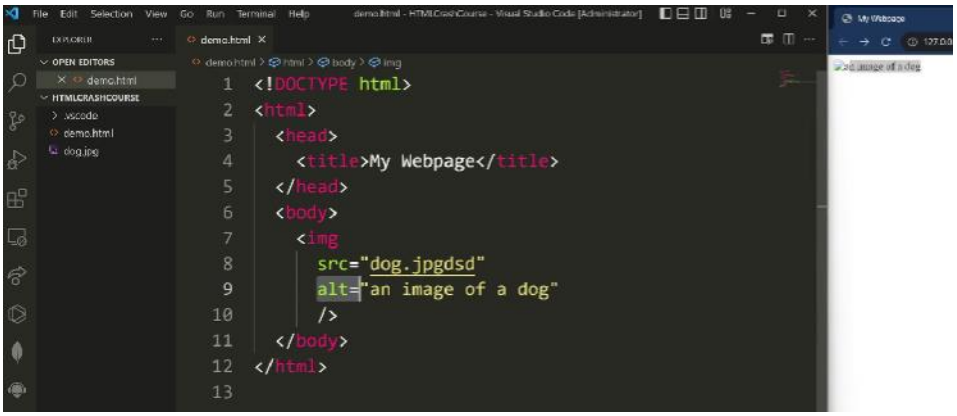
<img /> Image tag. Another special tag. For this tag, we need to specify the size of the image displayed. We're gonna put a link to the image online or download the image.

To link to the image, right click on the image and click on copy image address then go the image tag and type in the source (src)= then paste the source address within the quotes. Example

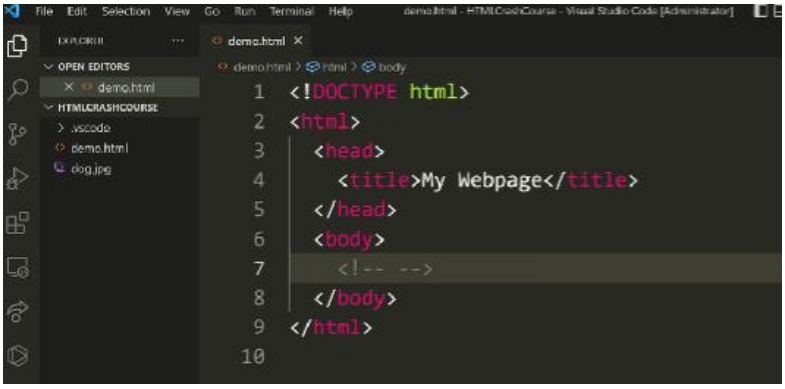
I can also download the image onto the directory that Im working on. Save the image into the file I'm working on, and it will appear into visual code.



Alt= alternative text. In case the link doesn't work, or the image doesn't work.



Comments on HTML

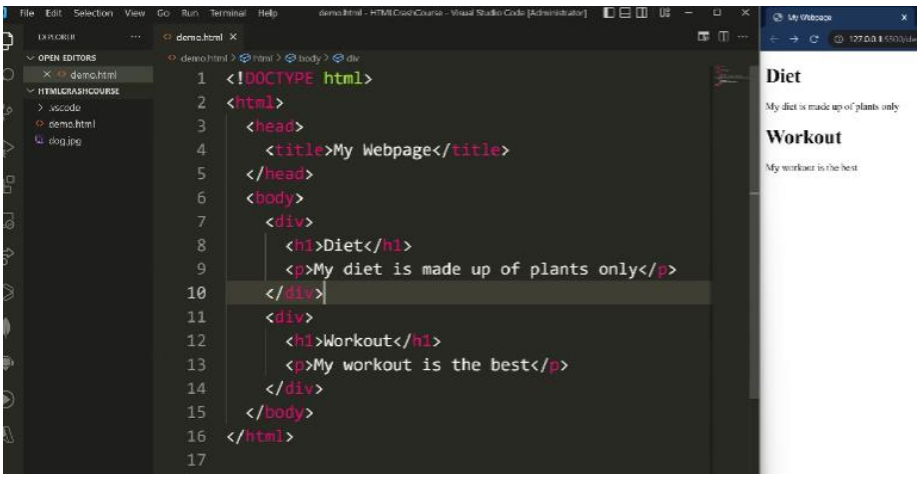


These won't appear on the page. They're only shown in the code for developers.

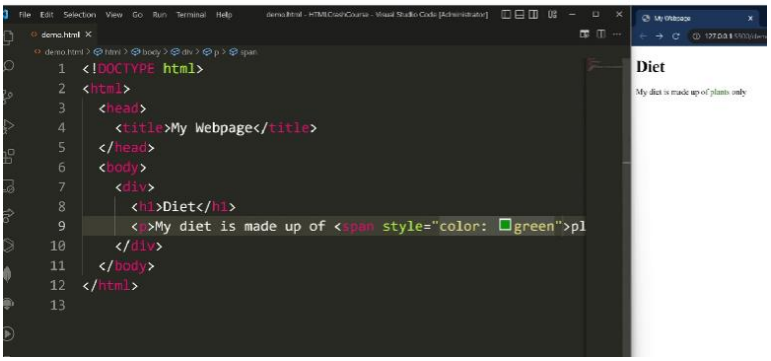
  This tag allows the text to be in bold.

 This tag puts the text in italics.

  division. It's a section of your html code. This tag is for more elaborate coding and is used to separate different sections. It allows you to work on different sections individually without changing the entire document.



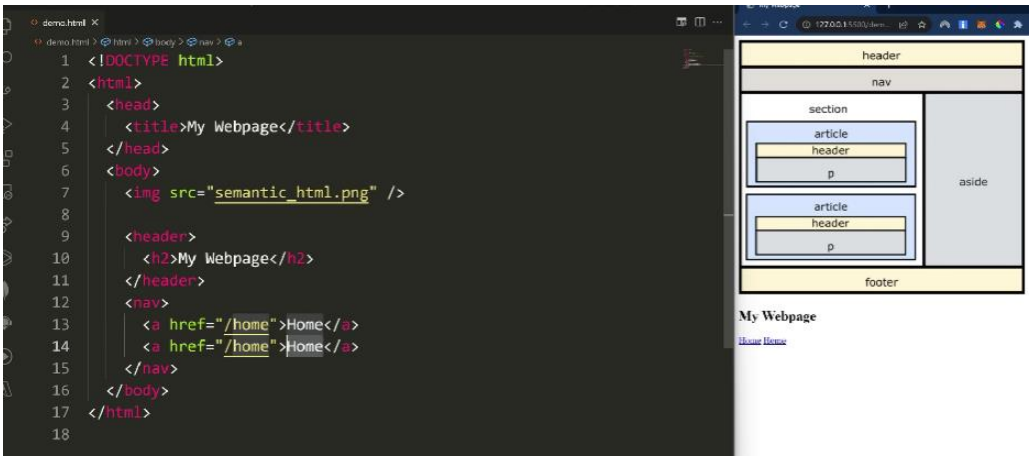
 This tag is used for in line distinction, to style individual words, give them a different color etc



SEMANTIC ELEMENTS

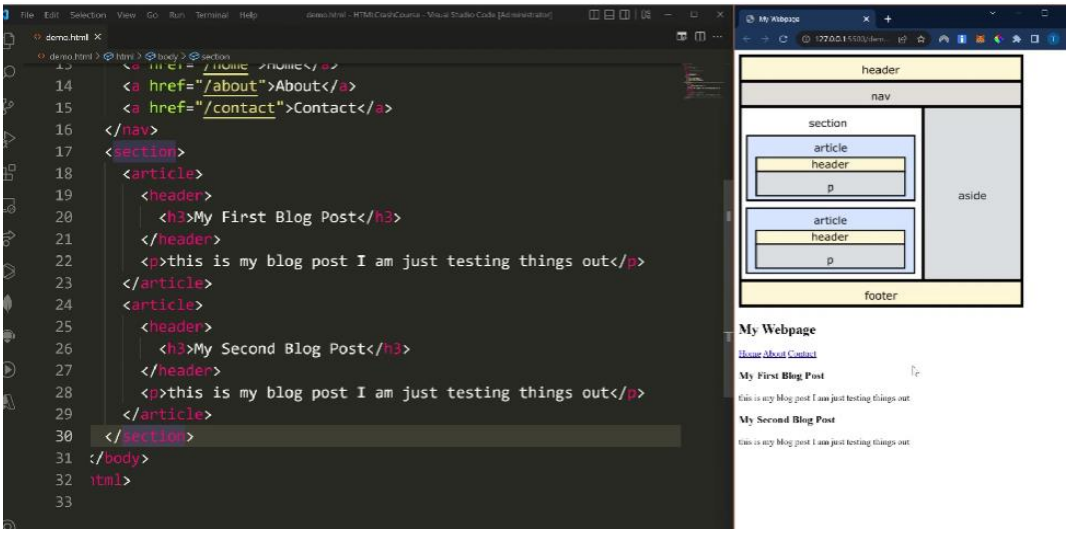
They describe the type of structure of the html document to developers. With those, we don't have to use <div tag> or tags like that. They make code reading easier. Especially if you're using a code reader or have someone else read the code.

<nav> <nav/> Navigation bar will be used instead of



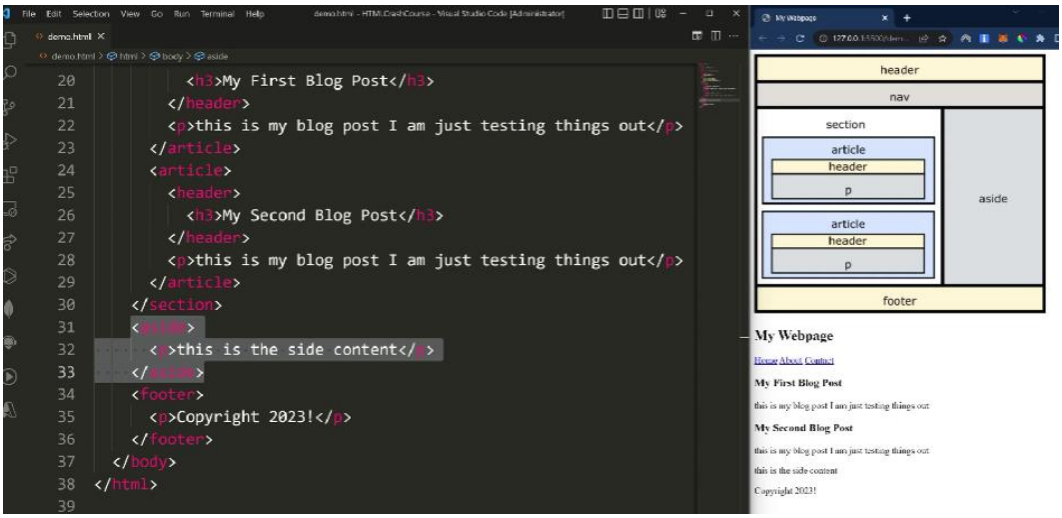
<section><section/>

<article><article/> is used for a website like a blog, a website that hosts comments, newspapers articles. What is inside an article tag should be independent of everything else on the webpage. They shouldn't rely on anything else on the page to make sense.

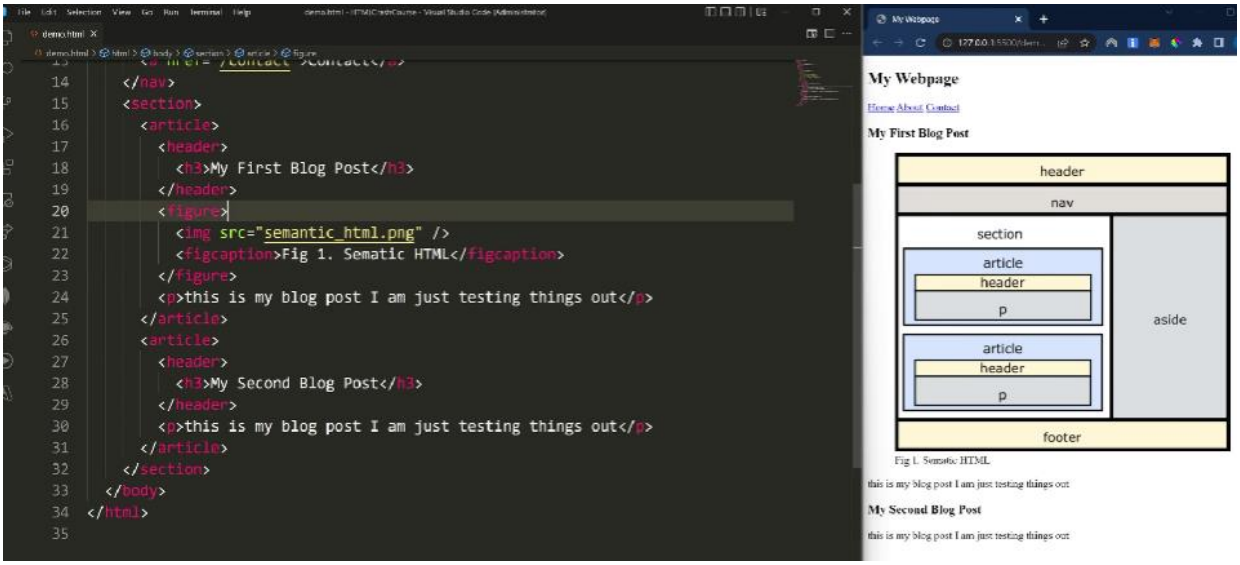


<footer><footer/> Anything that exists at the end of the webpage. This can also be at the end of a specific section.

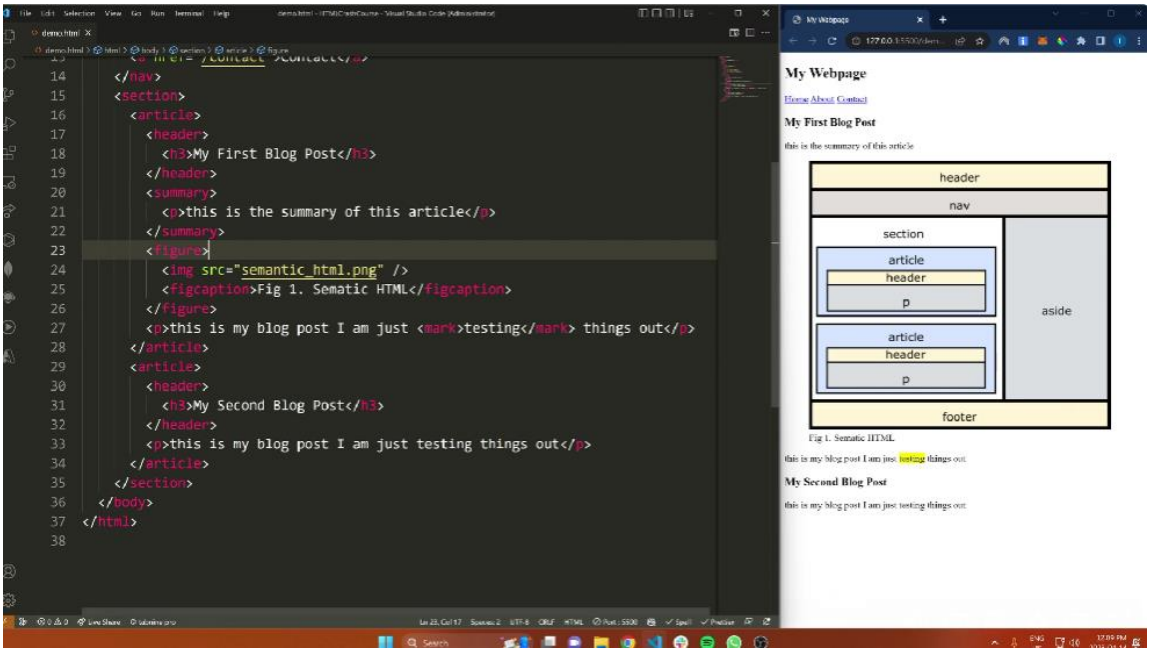
<aside><aside/> Tag that's going to be situated outside of the section. Side content. It still appears below the rest of the content unless we specify inside the tag where we want it put.



If we want to insert a picture inside an article tag, we're going to use the tag <figure><figure/>. Within the tag, we're going to write the image tag. Example



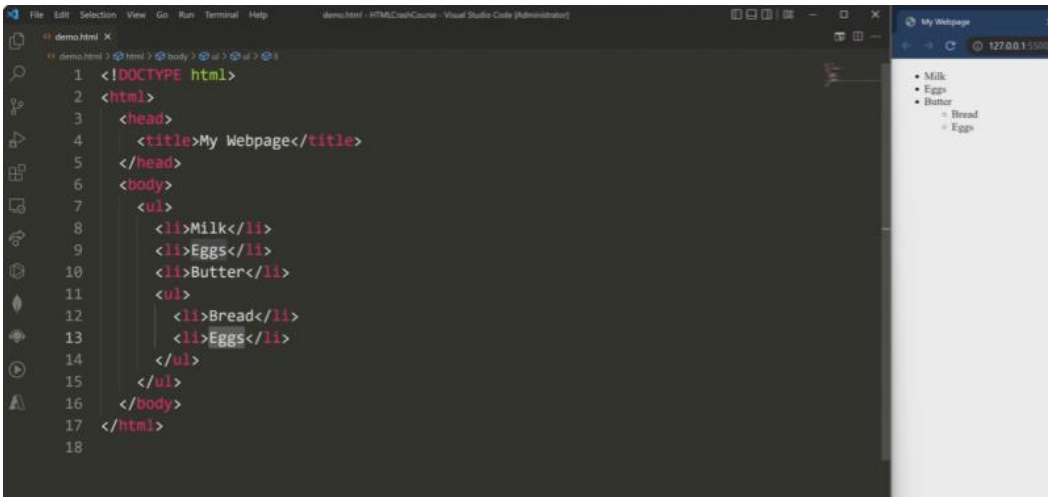
<summary> </summary> tag



LISTS

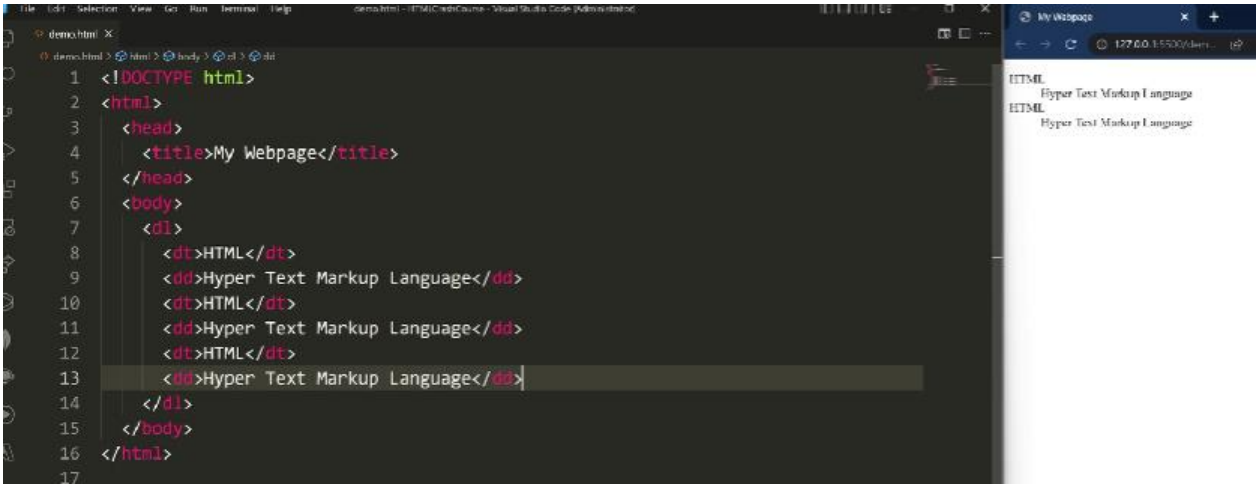
 unordered list

 List item



 ordered list. The difference with the ul is instead of having random dots for each point, it will be 1) 2) etc

<dl></dl> Description list is used for terms that have a definition after them. <dt> specifies the term and <dd> specifies the definition of said term.

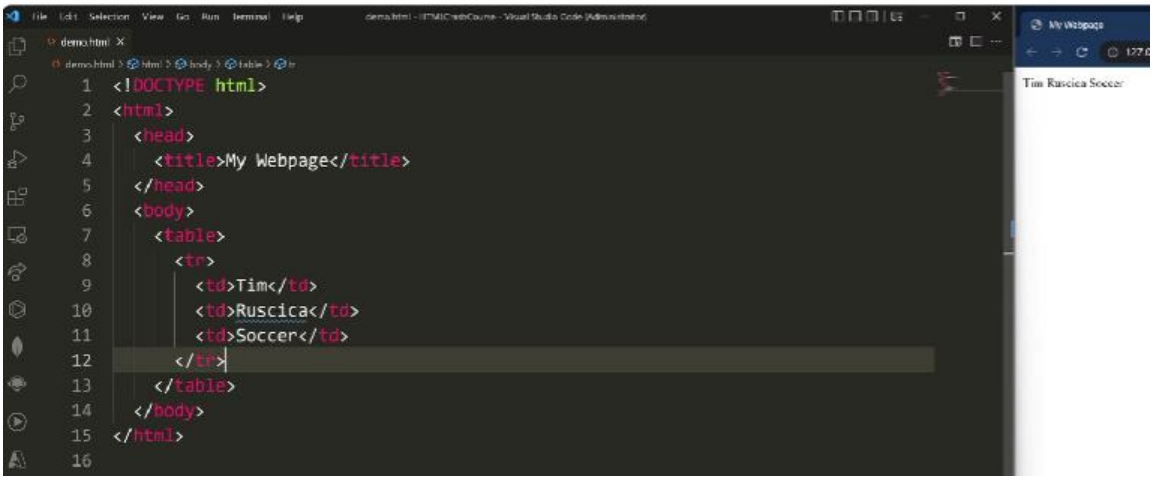


TABLES

<table></table>

<tr> table row

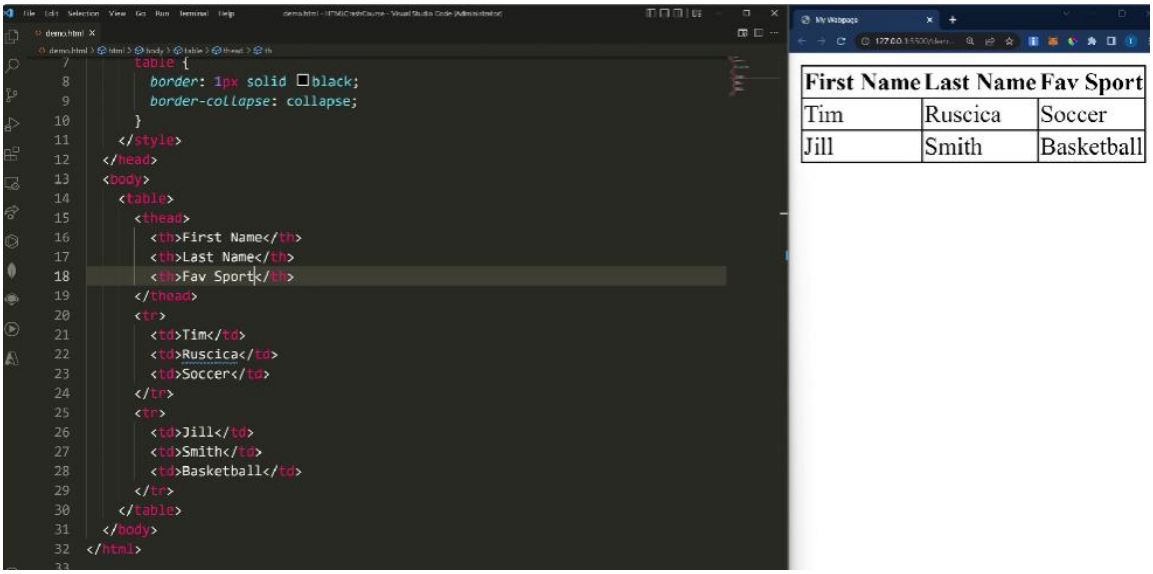
<td></td>



All these elements are inside of a table, even though the result doesn't show it.

<thead> table header.

<th></th> Bolded version of a table data (<td>)



To specify what is the body of the table, we'll insert the tag <tbody>. Underneath this tag, we'll put the details of the table data. To structure the table better, we'll have a header, a body, and a footer.

<thead>

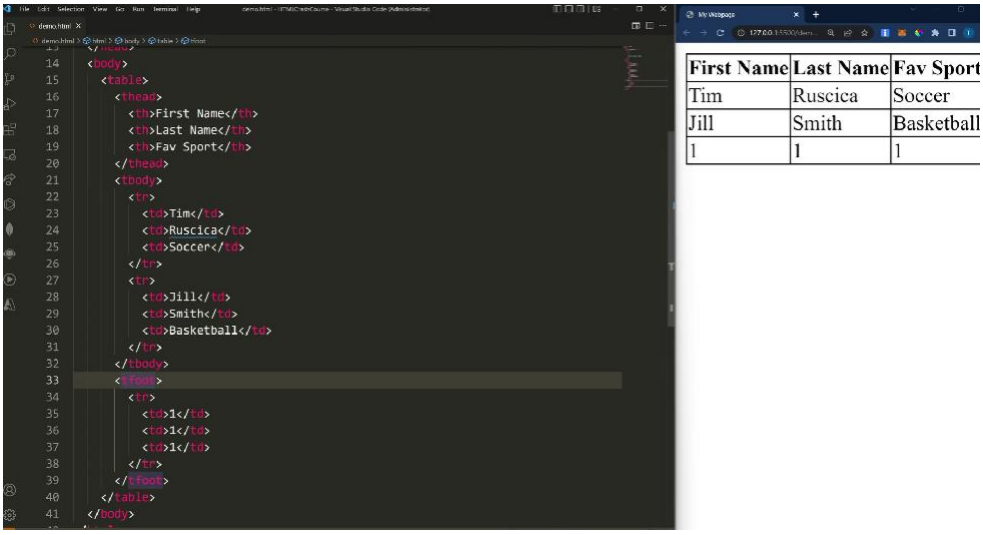
</thead>

<tbody>

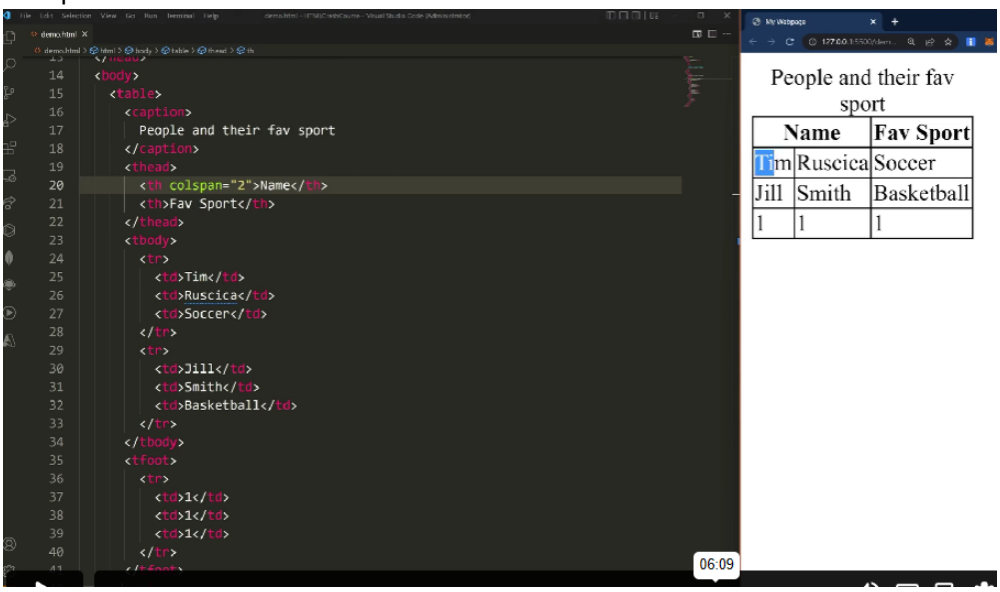
</tbody>

<tfoot>

</tfoot>



To separate a column in 2 inside the header.

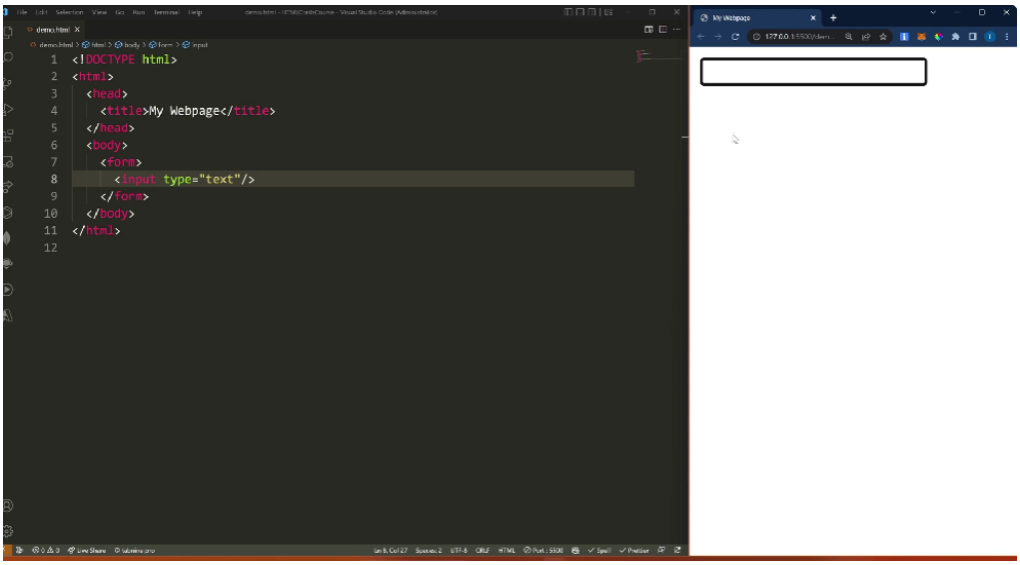


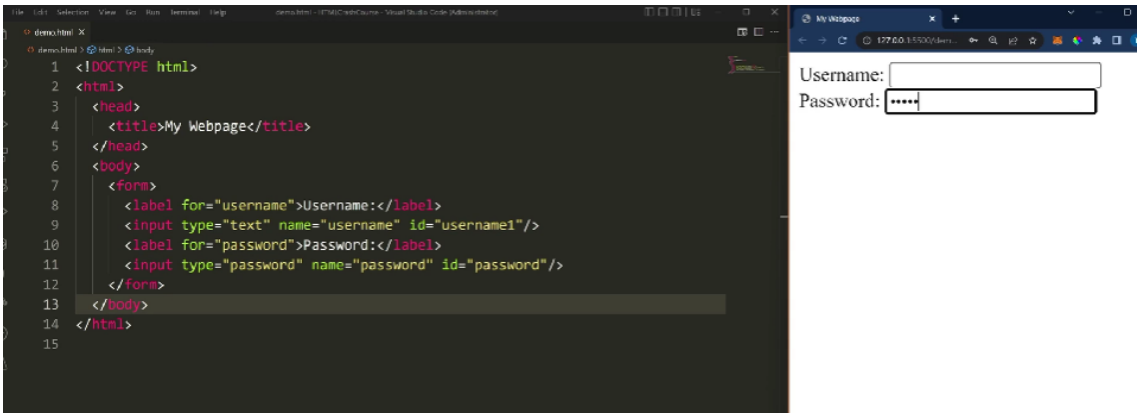
The same can be done inside the table, using rowspan.

FORMS

Section of html that allows user input (Ex: tweet). Allows communication between user and developer.

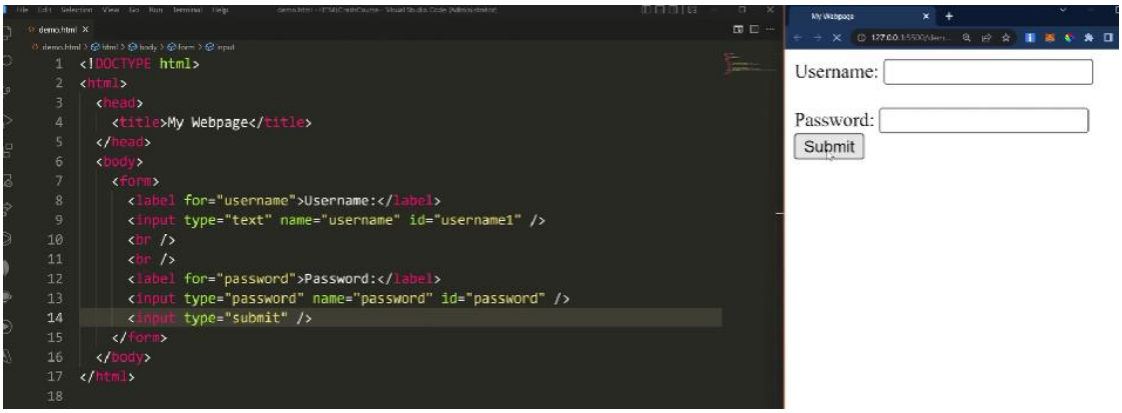
We'll use the tag <form> to specify it's a form. Then <input> for the space allowed for the user to put their input. <text> to specify, it's in text form.





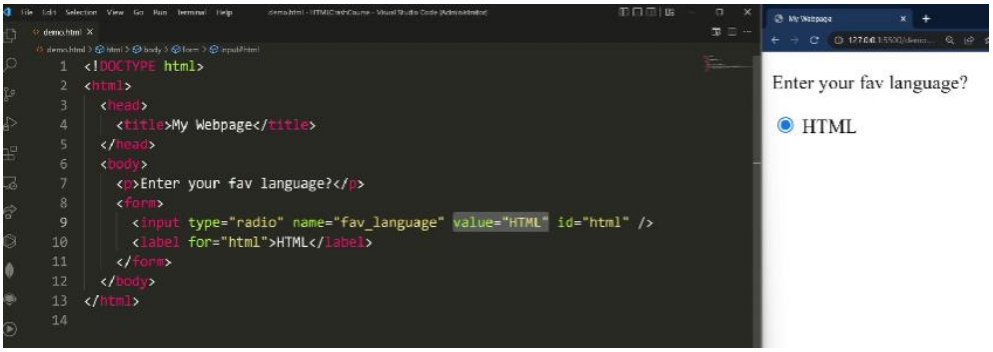
The password is blocked out with dots because by default when you write password inside the code, the result will be hidden.

Submit tag means we're done with the form and whatever data we collected is going to be submitted to the developer.



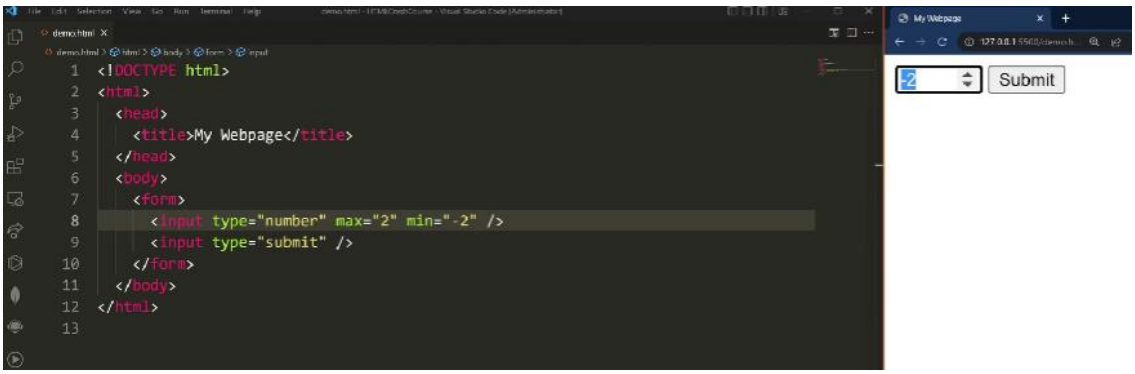
Quary perimeter is whatever comes after the ? inside the url and It can change the url after you click submit after putting your username and password.

Radio button allows you to select only one of the buttons.



Checkbox works like radio button except we can select multiple results.

If we want to create a tab where we can only type in numbers, we need to take out the <p> tag from the code. And if we want the number not to go higher or lower than a specific number, we'll add the details to the code with the max/ min.



BLOCK AND INLINES

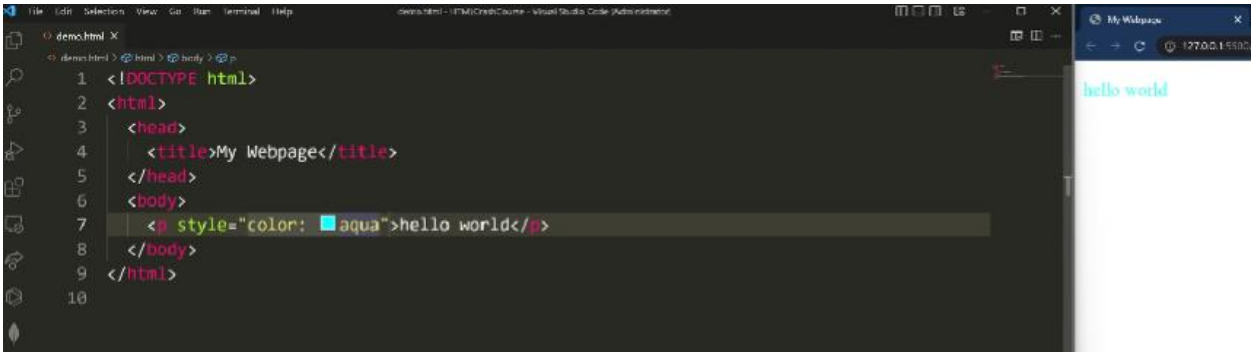
In html we have 2 different types of elements. Block elements and inline elements

Block will take up an entire space and send us to the next line. <Nav> and <head> are examples of block elements. They're going to fill the width of their parent automatically, which is the element they're nestled inside of. Whereas the inline element, they're going to take the minimum amount of width possible.

CSS FUNDAMENTALS

CSS stands for "Cascading styling sheet" and is a language used to style and design a website.

We can style the document in various ways. One option is what we call in line style which is going to be located like this. Those specificities are called columns or properties.



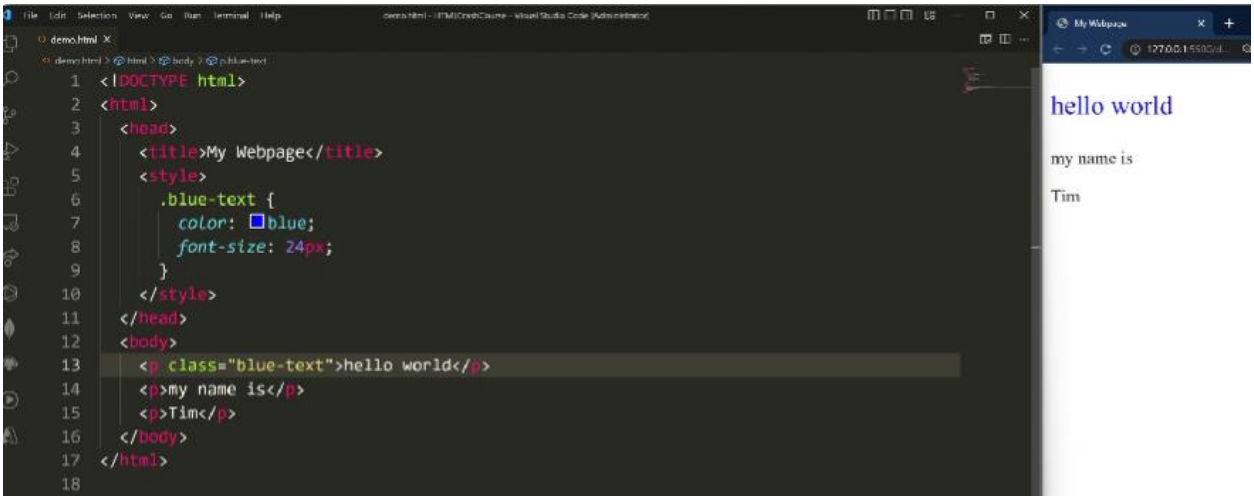
Another way of styling a document is to have a dedicated styling sheet that we're going to create outside of the element. If we put them in a separate location, they will be easier to spot and modify. To do that, we need to create a style tag that will be in the head of the document. We're going to use what we call CSS select.

In the head of the document, if the entire section we want to style in a similar way is a paragraph, we'll use the p and then follow it up with curly braces. Within the curly braces we'll put the specificities we want, all separated by a semi colon.

Example

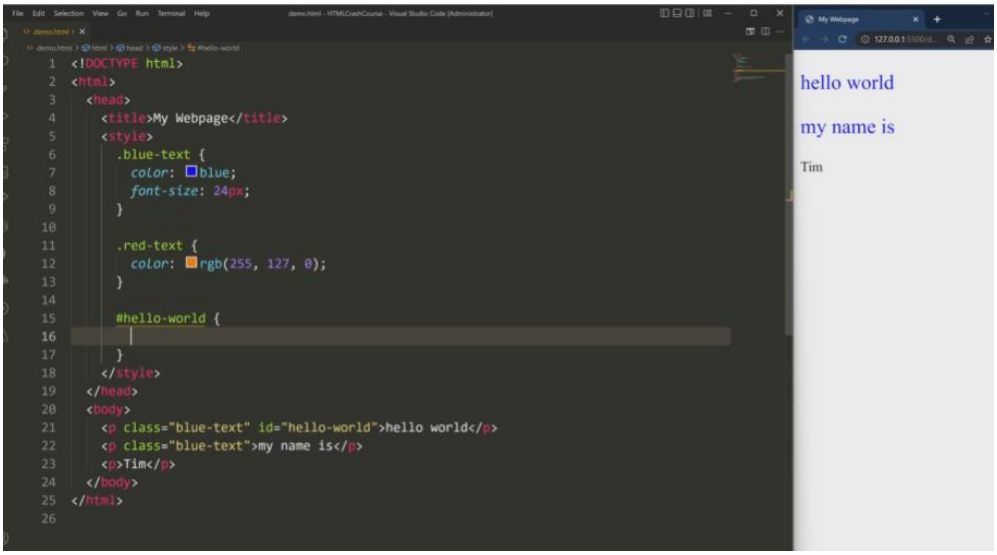
The way to style multiple paragraphs without doing them all is to use the specification “class” inside the tag. All the elements that contain that “class” will then be styled the same way. Whenever we name the class, we can only use regular alphabet and – or _ but anything different won’t work. Like when we name a file on a computer.

Here the style specificities are written inside the head tag, and we use the dot before the details to tell us that what’s following is the styling. In the rest of the document, every element that is going to contain the word class will follow the same styling rules as what the . was saying inside the head tag.



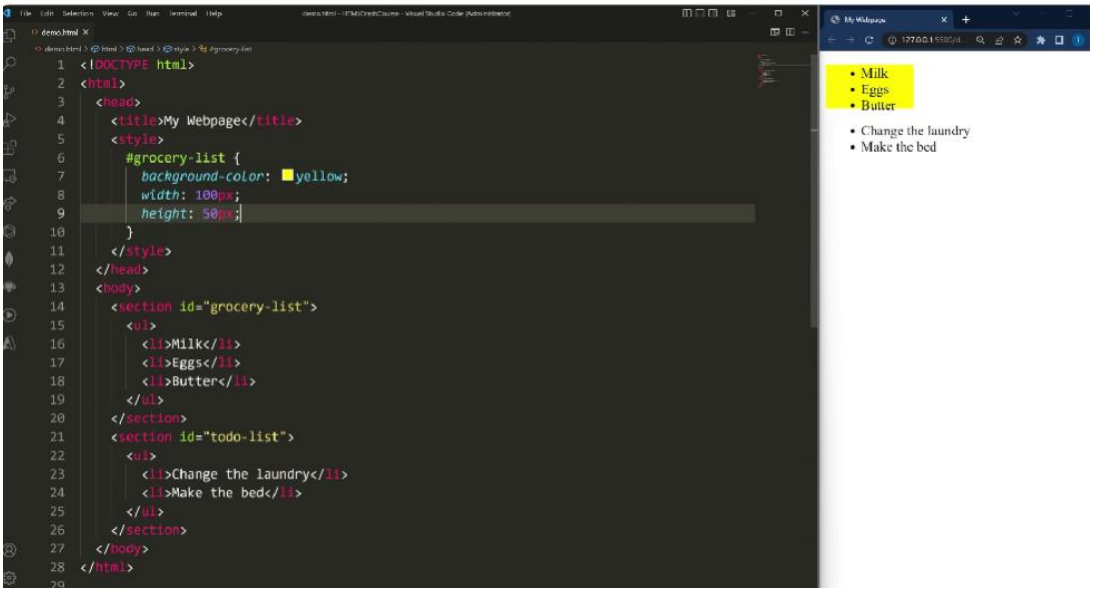
To style a specific element, we can use the term id to give it a unique style rule. For the id, to reference it inside the had tag, we will use the pound sign # It's different from the class that will use the .

We then type the id name. In this case it's hello world Example



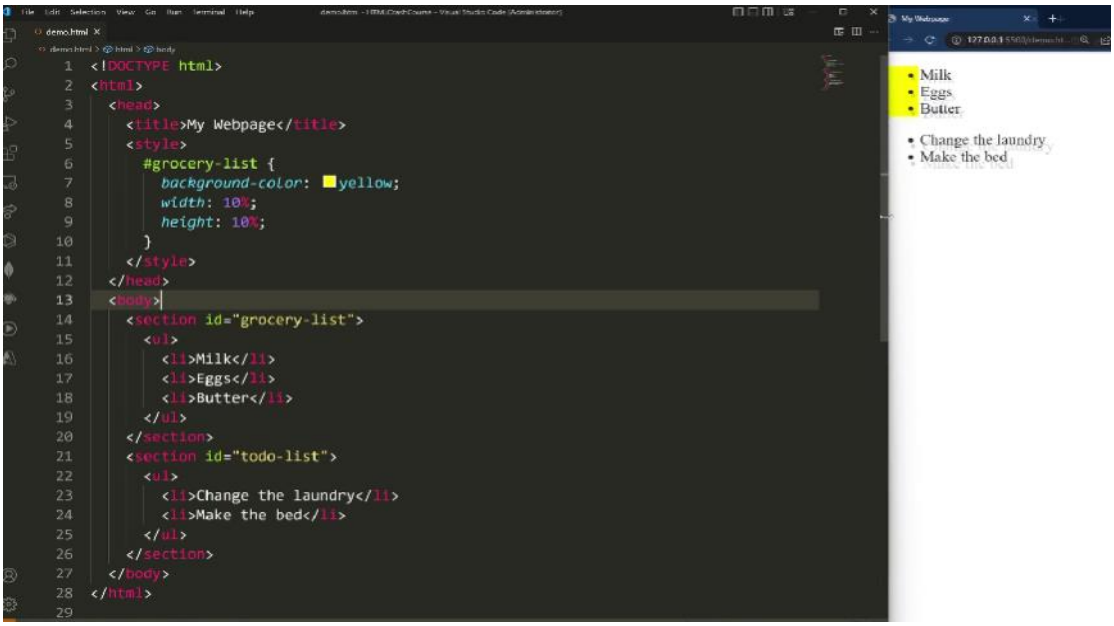
We can modify the width and the height of a section by accessing the id of said section and using the units' elements. There is a width and height property for all the css elements.

The first type of unit we can use is the pixels (px)

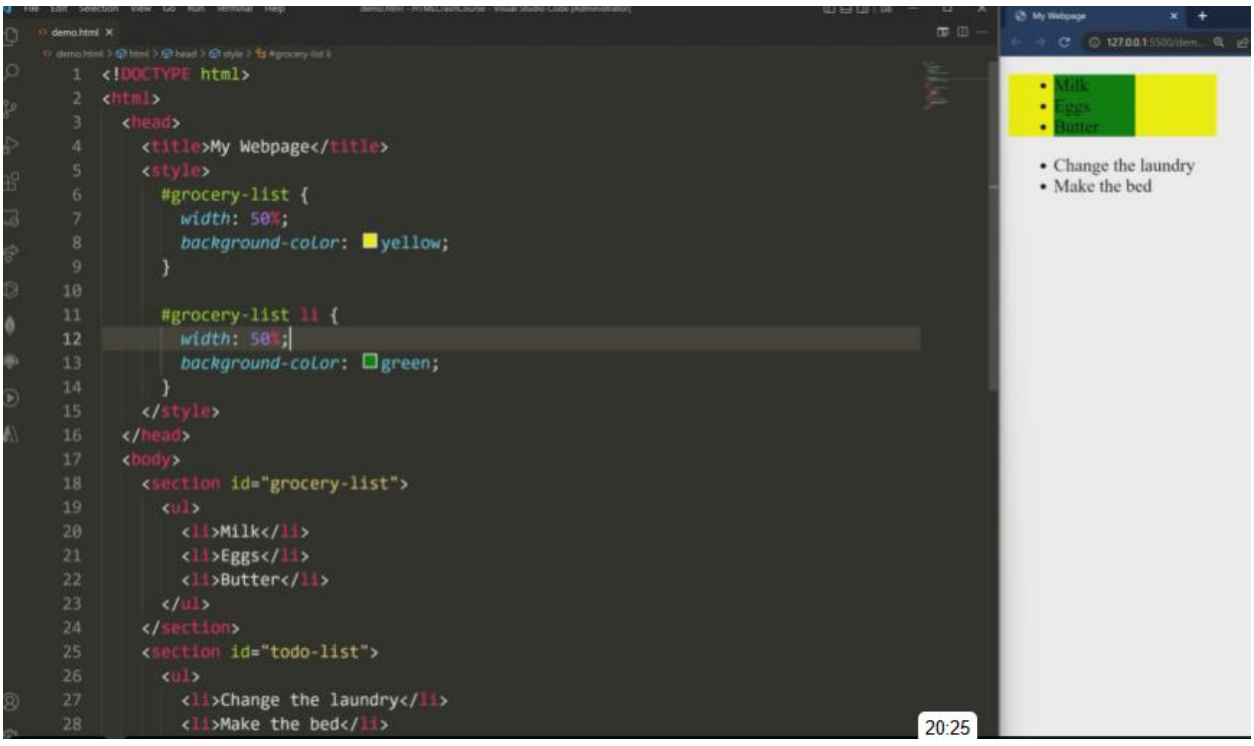


Another element is em, which is the font size of the parent element we are inside of. If we have 100em, that means we multiply the default font size by 100. We also have rem which is the default font size of the browser we are using. It is possible to use fractional numbers like0.1 for example to make the font smaller. The typical default font size for a browser is 14 or 16.

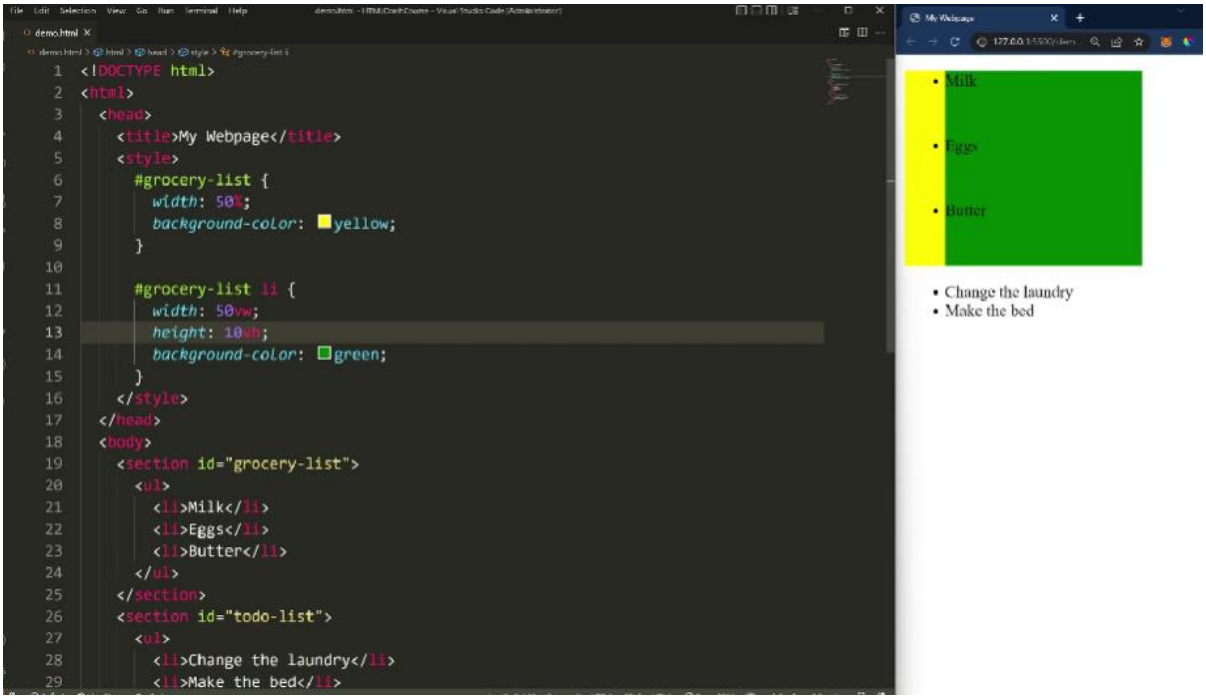
Another unit you can use is the percentage and it's going to be relative to the size of the parent element. In this case the parent element is the body.



Here we have the font size of the element reduced to half of its size. 50%. Then we have the li tag reduced to half of its size as well.

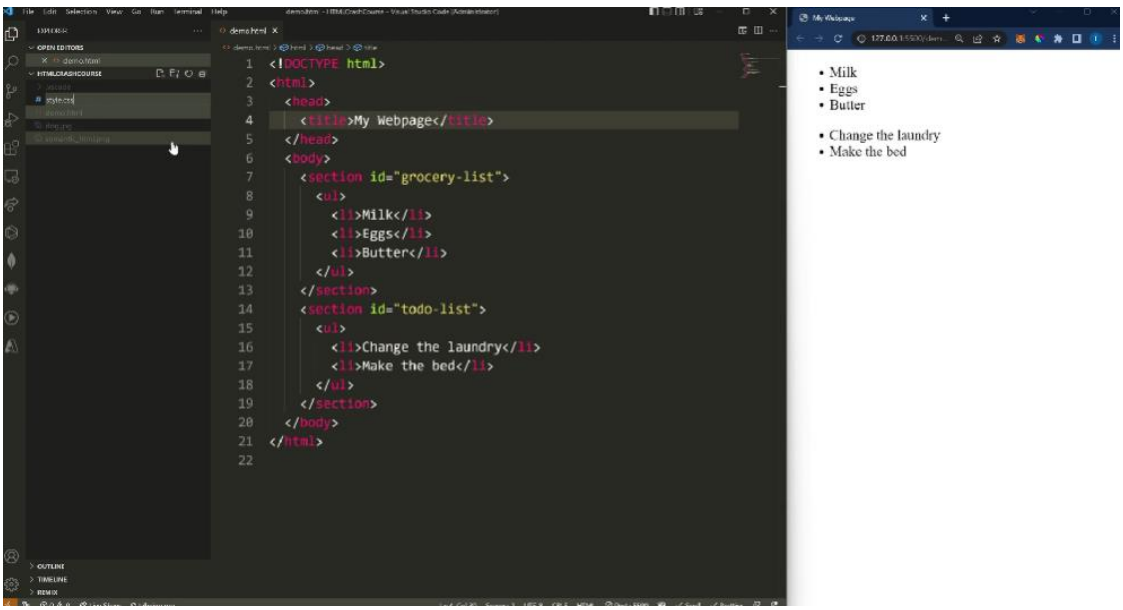


To modify the size of the li element without it being linked/ dependent to the parent element, we can use vw, which stands for viewport width. So, we take 50 (%) multiplying by the total width of the viewport, so the total width of the element. We do the same thing for the height and use vh.



To create a dedicated style sheet, we can move all the styling specificities into a separate style sheet that we reference inside our original sheet. To do so we create a new document called style.css

The name can change but the format has to be css



```css
#grocery-list {
    width: 50%;
    background-color: yellow;
}

#grocery-list li {
    width: 50vw;
    height: 10vh;
    background-color: green;
}
```

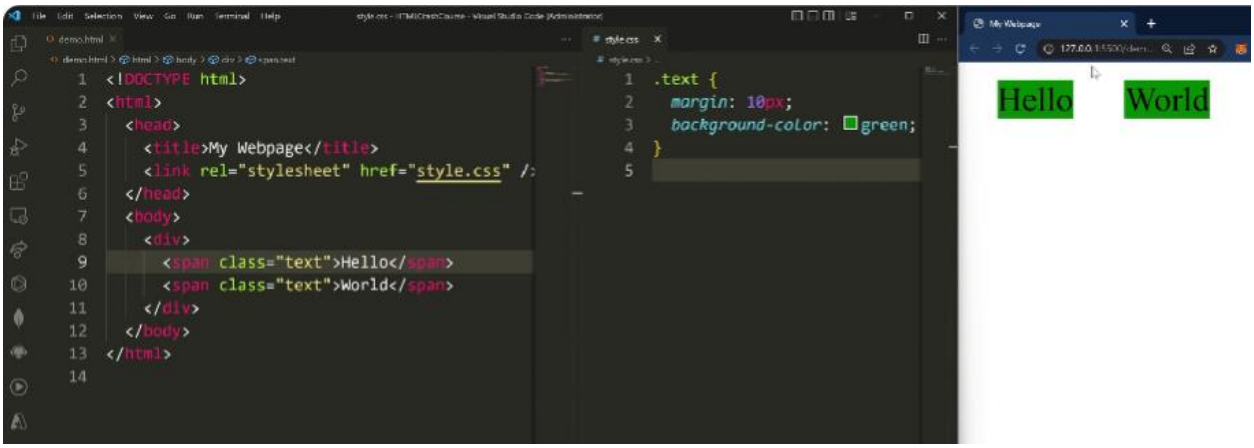
To reference the style sheet inside the html document, we'll use a link tag.

```html
<!DOCTYPE html>
<html>
  <head>
    <title>My Webpage</title>
    <link rel="stylesheet" href="style.css" />
  </head>
  <body>
    <section id="grocery-list">
      <ul>
        <li>Milk</li>
        <li>Eggs</li>
        <li>Butter</li>
      </ul>
    </section>
    <section id="todo-list">
      <ul>
        <li>Change the laundry</li>
        <li>Make the bed</li>
      </ul>
    </section>
  </body>
</html>
```

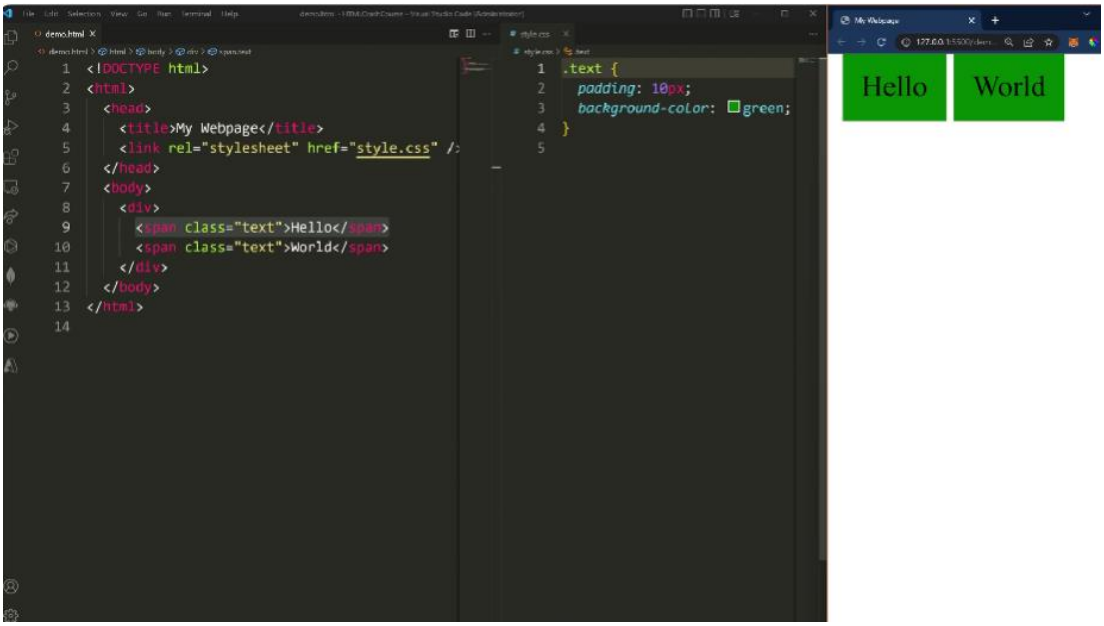
CSS BOX MODELS

On the left-hand side of the screen, we have a div and two span class. These represent the elements Hello and world which are different from one another but located next to each other. To style these elements, we have different possibilities.

Margin: Space outside of the element and between different elements. In this case, we write 10px for 10 pixels, which is going to be the space outside, on all sides of the element. We also add a background color of green. The margin in the middle is larger because it is doubled. We have the 10px from the element “hello” and the 10px from the element "world”, making it a 20px. The background color is only on the actual text element.



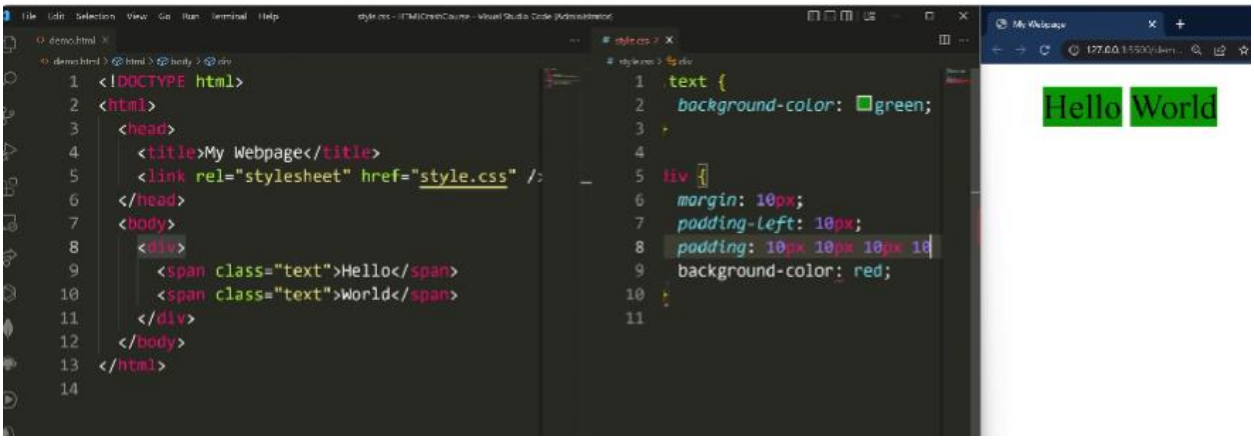
Padding refers to the space within the element we are working on. So, to get to the edge of the text element, we have a space of 10px. That’s why “hello” doesn’t take up the entire green colored box.



Margin is outside of the element; padding is outside of the element.

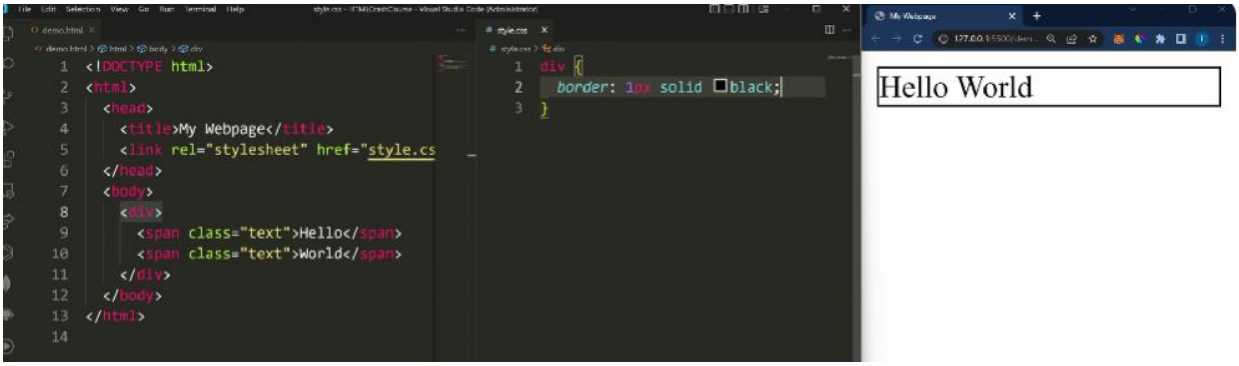
To apply the padding on only 1 side, we have different options. We either write the tag padding-left, padding-right, padding-top, padding-bottom, or we write padding: 10px 10px 10px 10px, referencing to 10px to the top, the right, the bottom, the left, in that order. The same applies to margin.

When you use padding you can also use %.

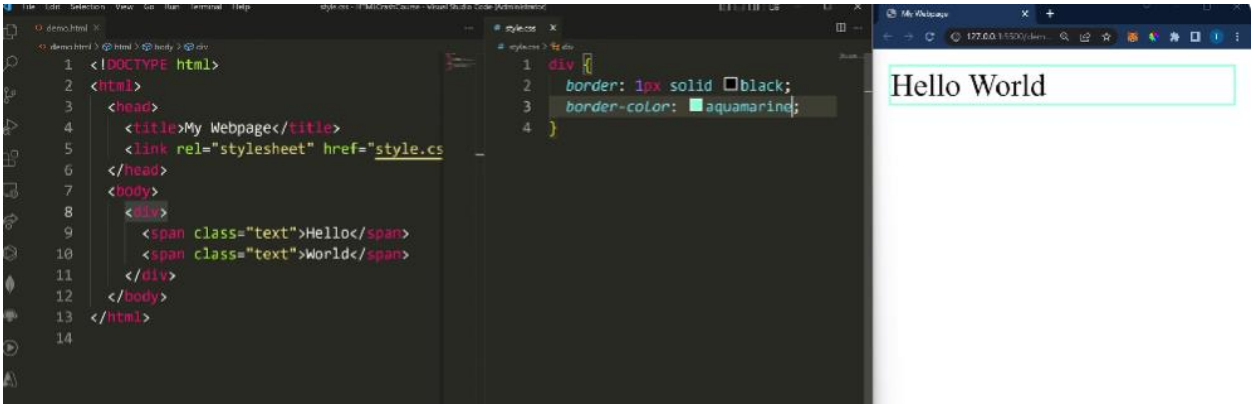


Border: Every CSS model is bounded by rectangles. This rectangle is usually the size of the content or in the case of a block level element, the entire width of the container.

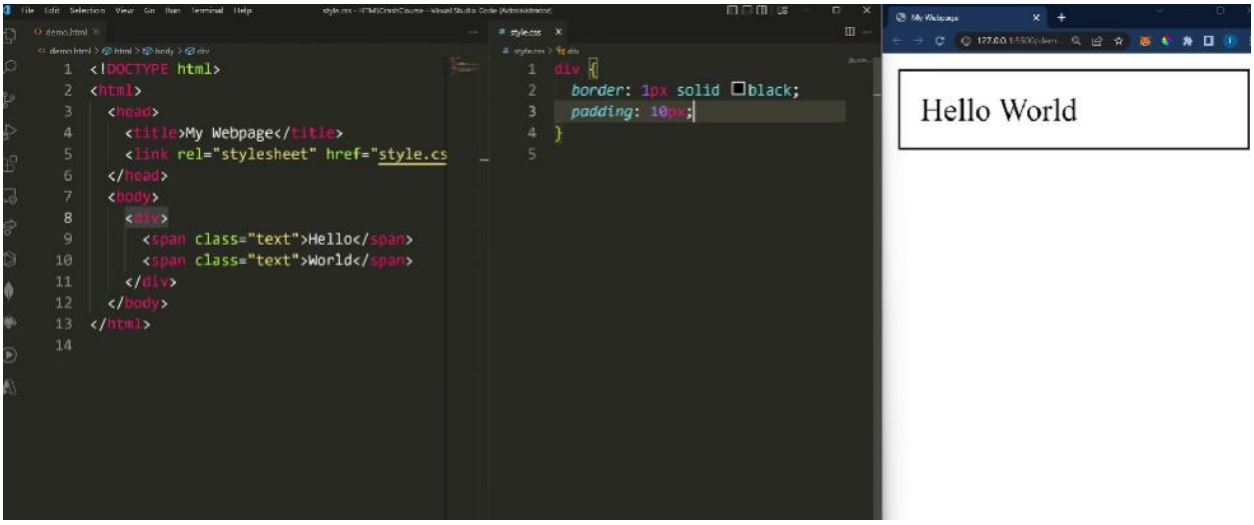
This rectangle that we can call box is understanding all margin, padding, border within this rectangle. When we write our border, we have three properties. The size of the border, the style, and the color



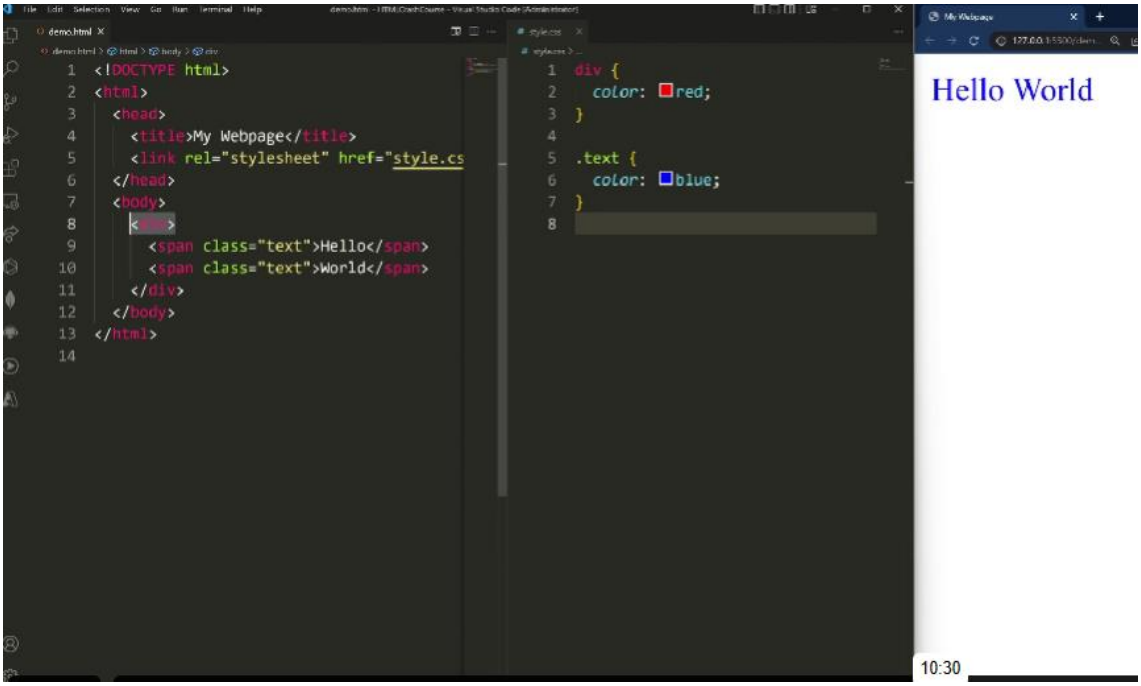
You can also specify individually like this:



If we want to add some padding, we'll write it like this:



If we have a parent element that has been styled, all the element within the parent are going to be affected and styled the same. Even if they have the class name attached to them. To make a distinction, we will use the . because the . has more power over the element than the parent element.

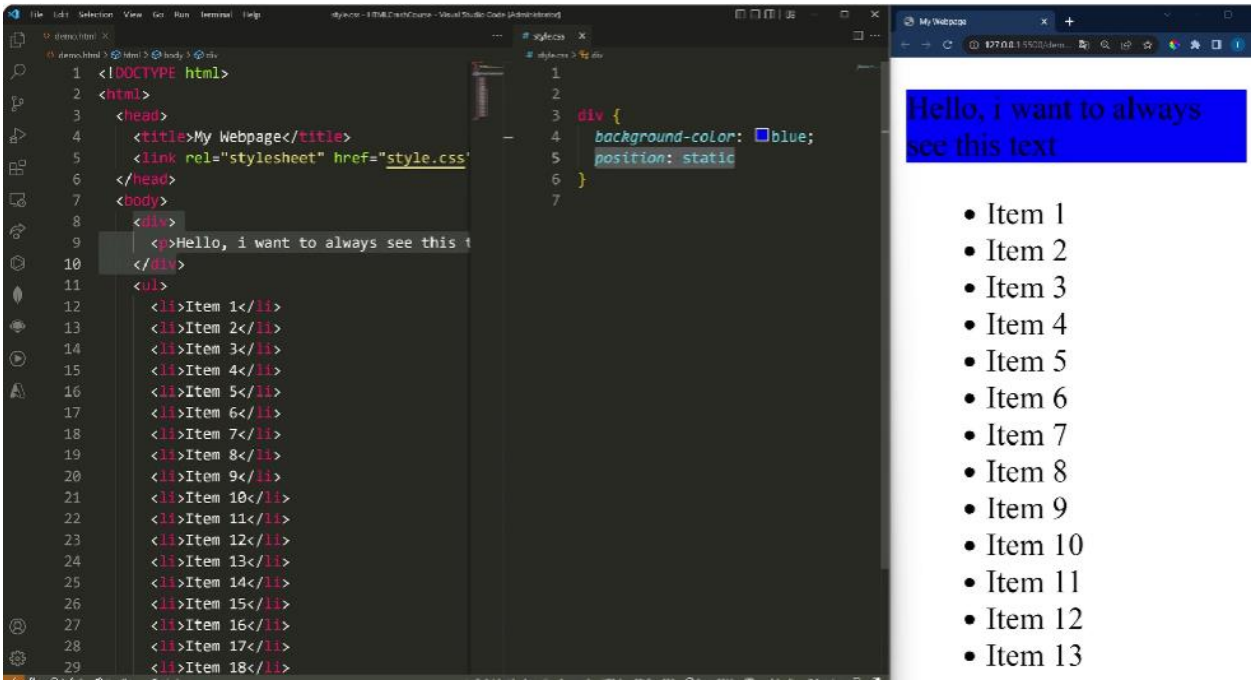


If we hadn't typed in the .text with the specifications, the elements would be styled the same as the parent despite being a class element.

CSS POSITIONING

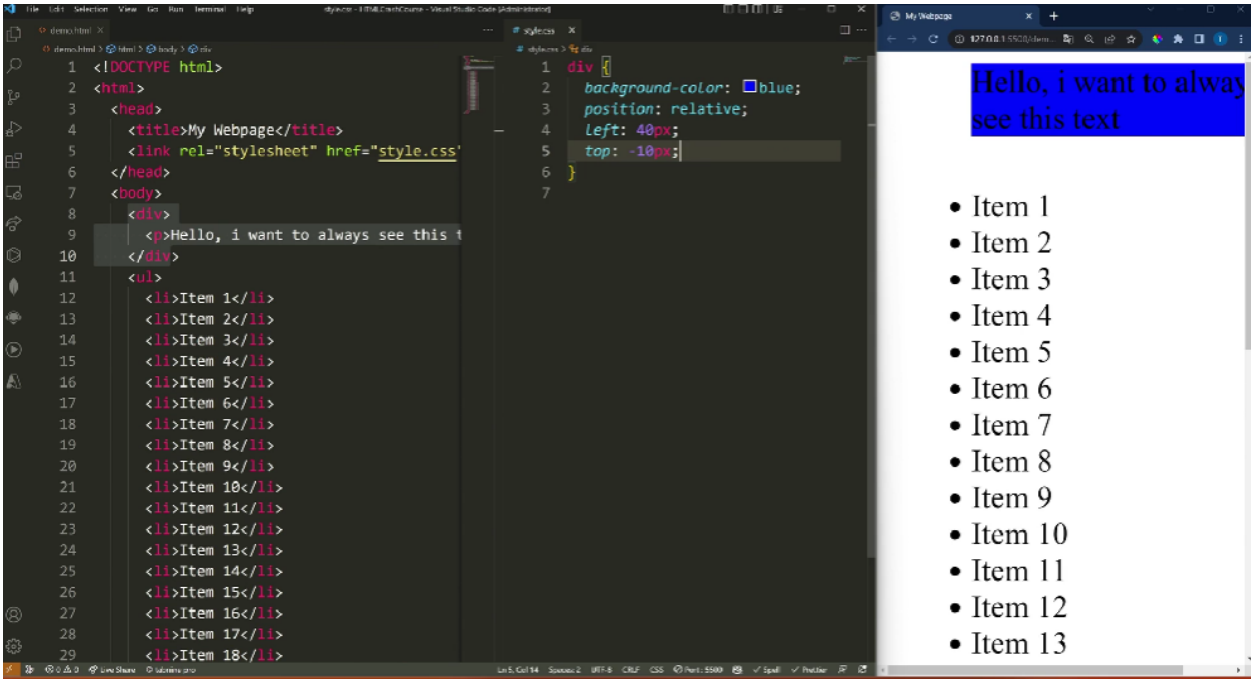
There are 5 different types of positioning and the reason why we do this is we want to be able to know how the elements are positioned on the page.

Static: default position. The elements follow the regular flow of the document.

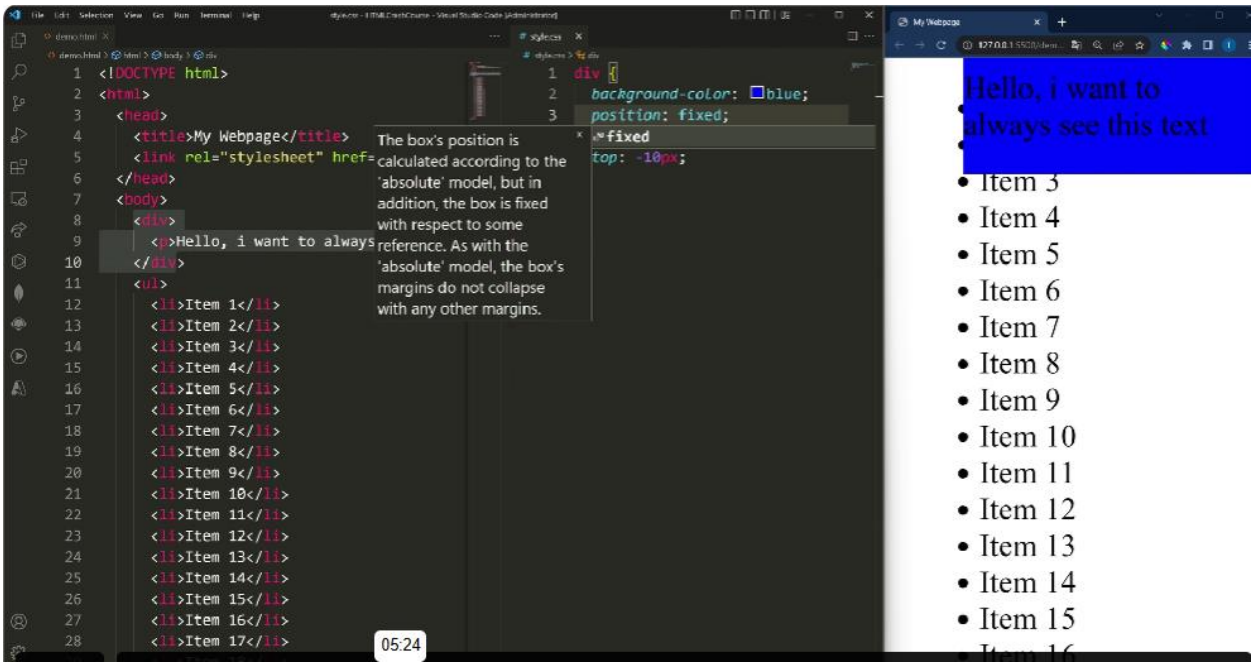


If we use the static positioning, we don't need to use the left, top, right and bottom specifications since it's default.

Relative: The element is positioned relative to where it would normally exist inside of the document. In this case, we want the element to be 40px away from the left-hand side. The element is then pushed off to the right. If we wanted to move the element to the side, we could also use a margin. The difference though is that it's not moving the element over. It's adding a gap or spacing between the element and the left-hand side. Here we also want the element to be closer to the top, so we'll use negative value. -10px.



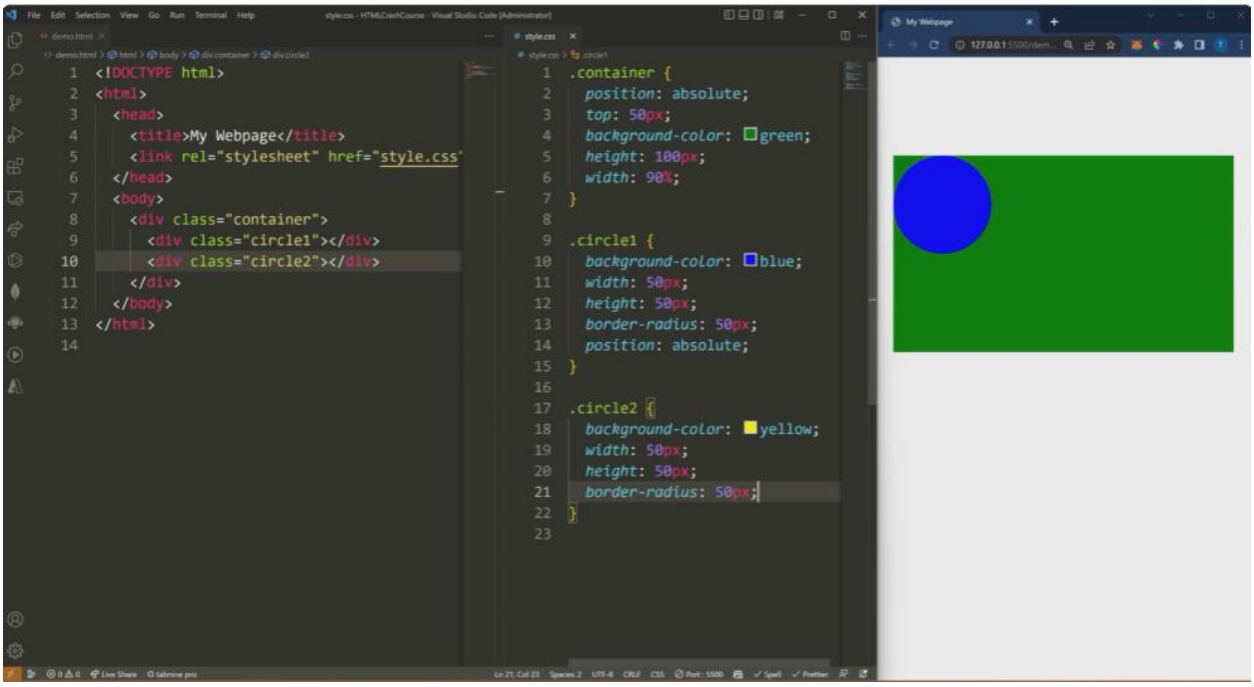
Fixed: This positions the element relative to the viewport. So, the element will move as we scroll up and down the document and will overlap with other elements. This is useful for a top navigation bar or a footer since we always want these to stay on a specific spot of the viewport.



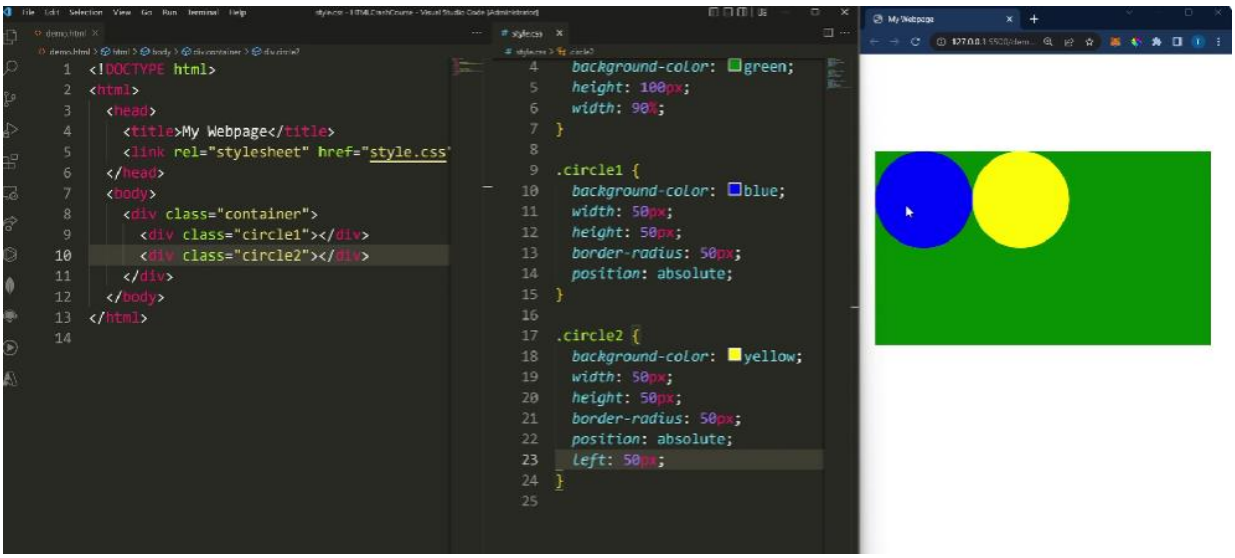
Absolute: The element is positioned the closest to the parent tag it's located inside of. We will use the specificities left, top, right, bottom again but the position will be relative to the parent. So 40px away from the parent not the edge for example. Absolute also allows the element to overlap other elements

but the difference between absolute and fixed is that absolute will be sticking to its position, not moving as we scroll up and down the viewport.

If we don't specify the l, t, r, b details, the element will by default go to the top left corner of the container that it's in. In this case the two circles are on top of each other because both absolute and both on the top left part of the container. Note that when we want to add a circle, we'll use the border-radius tag and use the same exact amount of px as the width and height.

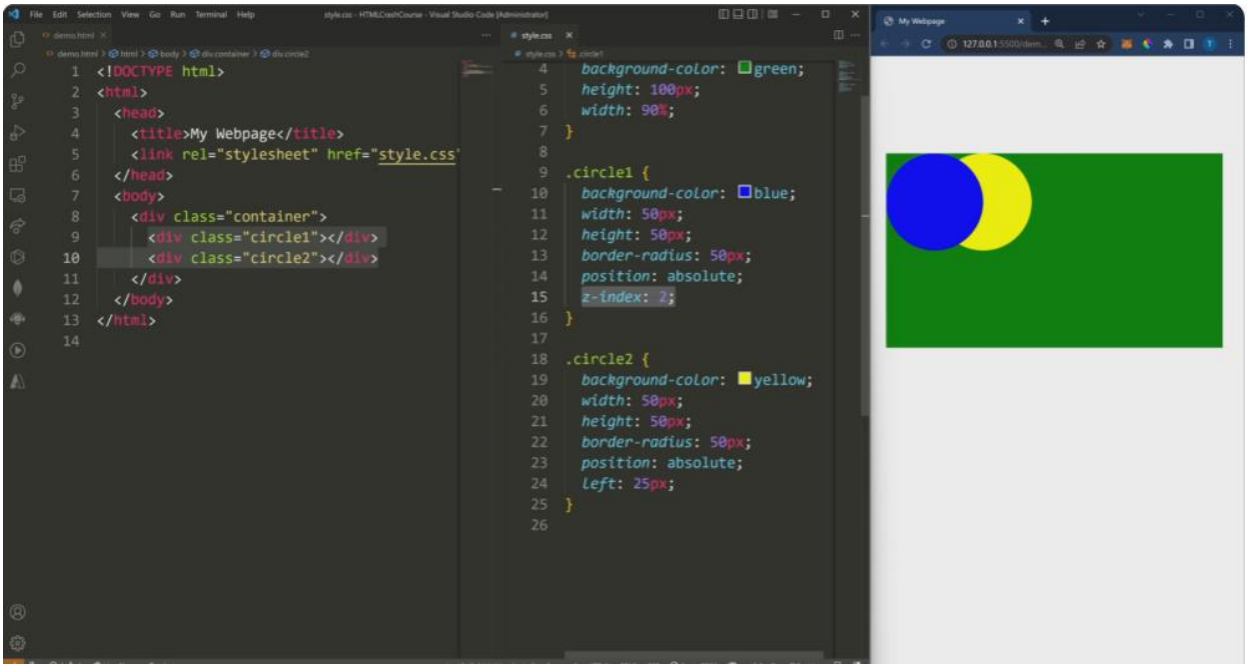


In this case, we changed the yellow circle, circle 2 and specified we wanted the radius to be 50px away from the parent element.



The z-index

The z-index allows you to specify which elements are on top of each other. For example, with those two circles the first one is the blue one because the tag is written first in order in the html code. Sometimes we don't have control about which order to write which elements. The z-index allows us to overrule that order and decide ourselves which element has the priority. In this case, we want the blue circle to have priority. We can write for example 2 for the blue one and 1 for the yellow one, knowing the 2 has a higher value than 1 does. We consider the z index as being a 3D notion compared to the left and right for the other notions. The z index only works when the elements are positioned. If we don't specify the exact positions for everything it won't work



Sticky: It won't do anything unless we specify the left, right, top, and bottom properties. Sticky is going to work based on the user's scroll position. Sticky is kind of an in between, between the fixed and absolute position.

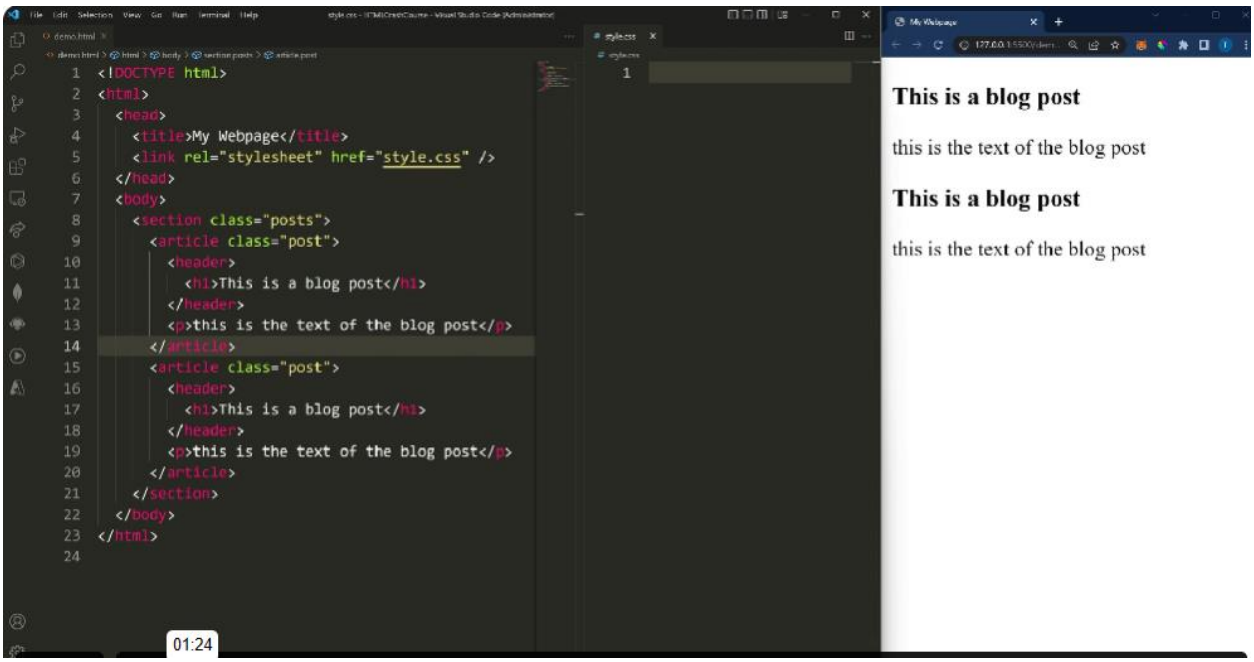
Relative to where it should be on the document and fixed is fixed to the screen.

The element will be moving as we scroll down but once we reach the spot where the element should be, it's going to "stick" to that spot and stop moving as we keep scrolling.

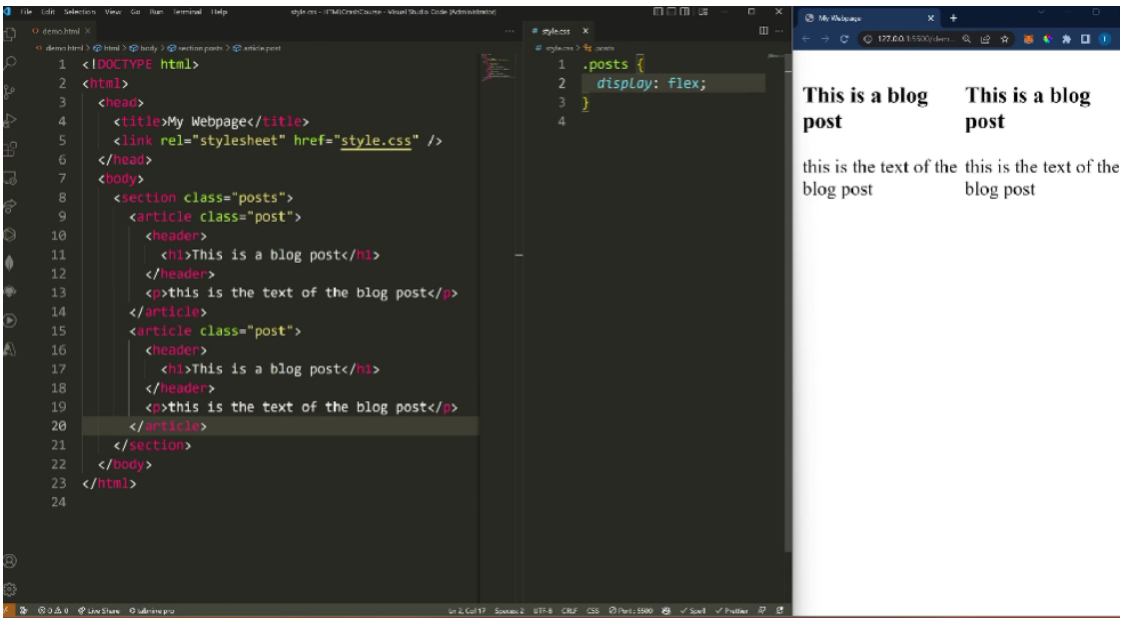
CSS LAYOUT

Flexbox: Popular way of designing a layout without using the techniques learned previously.

Here we have two different articles that we want to try and put side by side instead of following one another. We could use the absolute positioning but the problem with that technique is that they might end up overlapping or not taking the correct amount of space we want them to take.

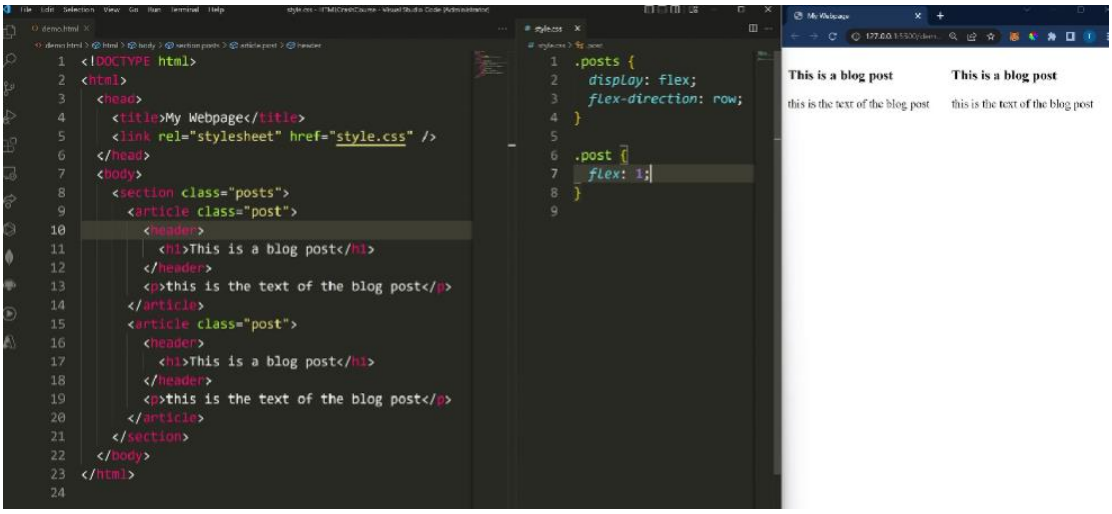


The two articles are in the class called “posts” so we’re going to reference them and use the flex tag.



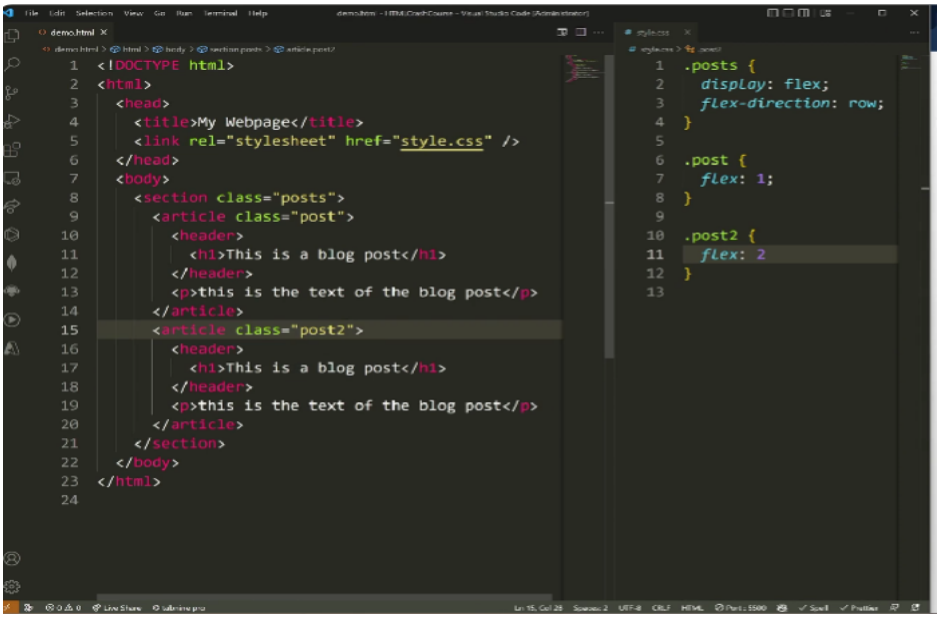
They are automatically put next to each other. Automatically it aligns the items across the main axis. The main axis by default is a row (we can change that and make a column if we want).

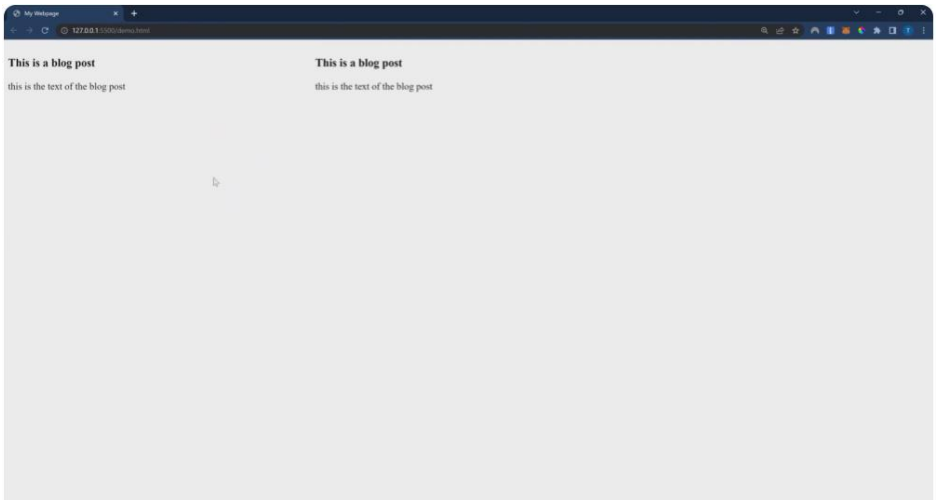
If we add the number 1 in the properties, the two articles now occupy 50% of the screen. Because as soon as we have a flex property inside the parent tag, the inside tags are gonna take up a certain percentage of the layout, depending on the value we attribute to the flex. Here we have two articles taking the entire layout so we divide 1 by 2 and obtain 50%. If we had 3 elements and divided by 1 by 3 they would occupy 33% of the layout.



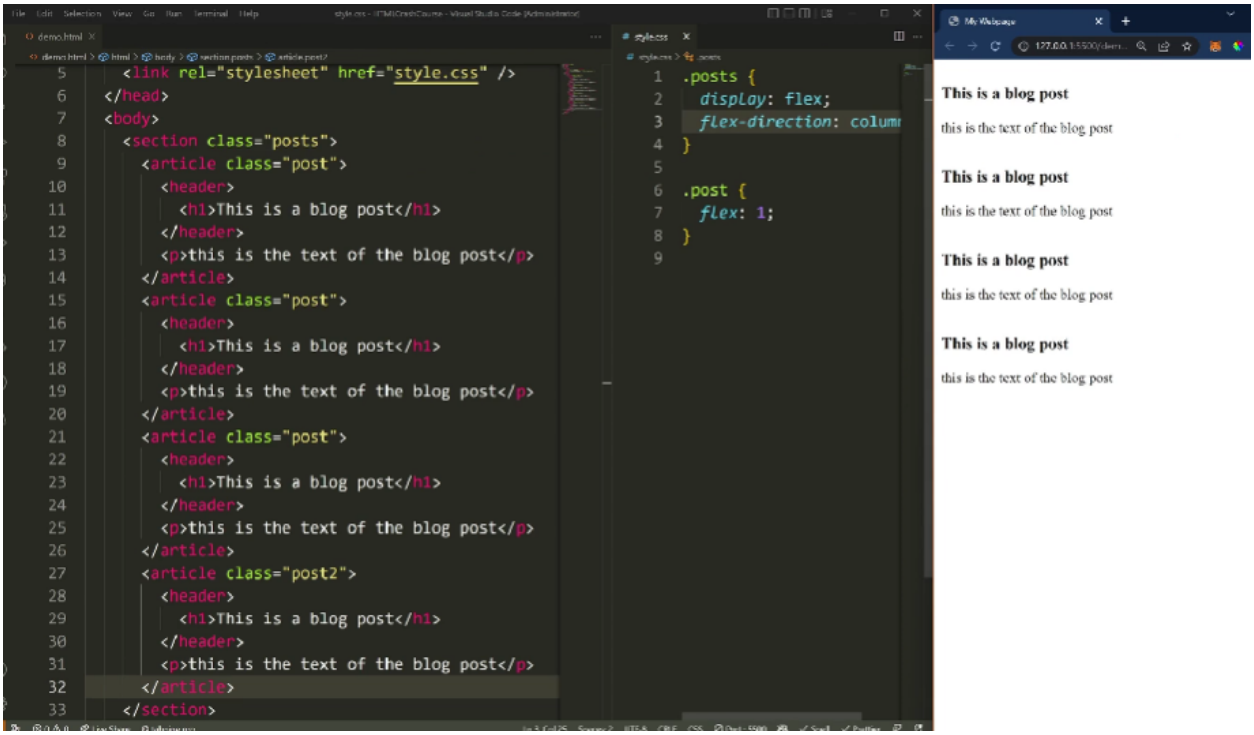
We can get more complex by having two elements, one occupying a third of the layout and the second one two thirds. For that we would have to separate them inside the class. We would have class1 where the element would occupy 100% of the container because we put a flex 1.

Then a class2 where the element would occupy only half of the layout. The flex would be 2. We would then divide 1 element by 2 and the element would occupy 50% of its container. This would be the general result.

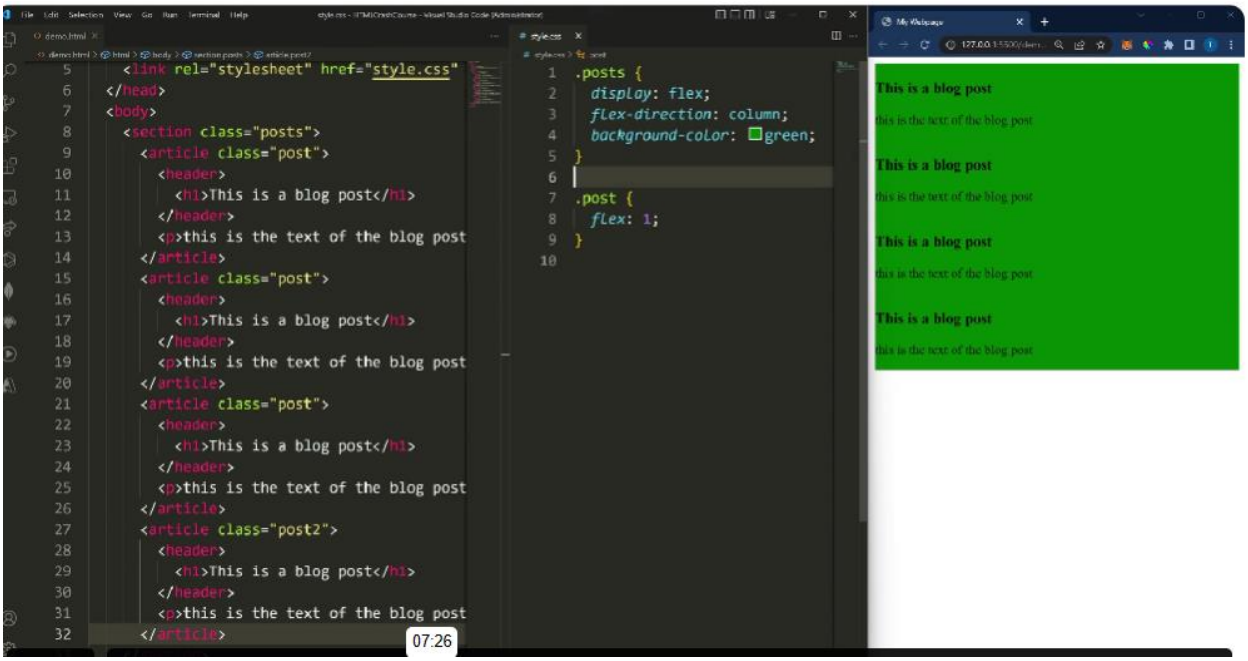




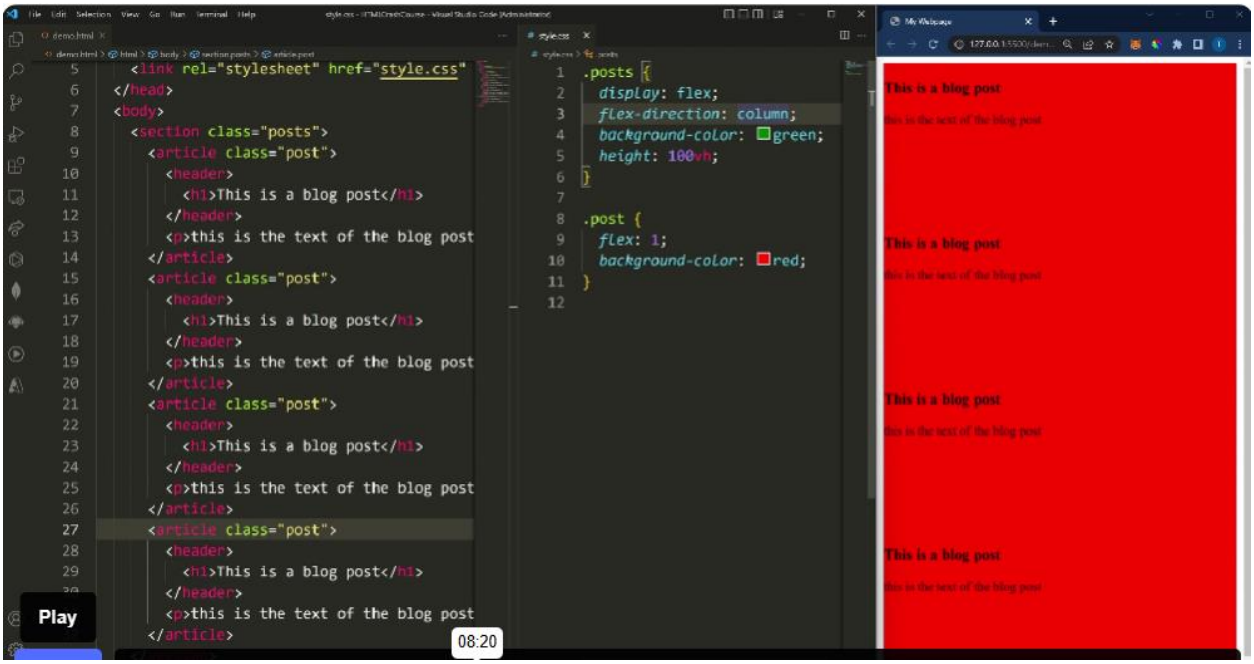
If we want to use columns



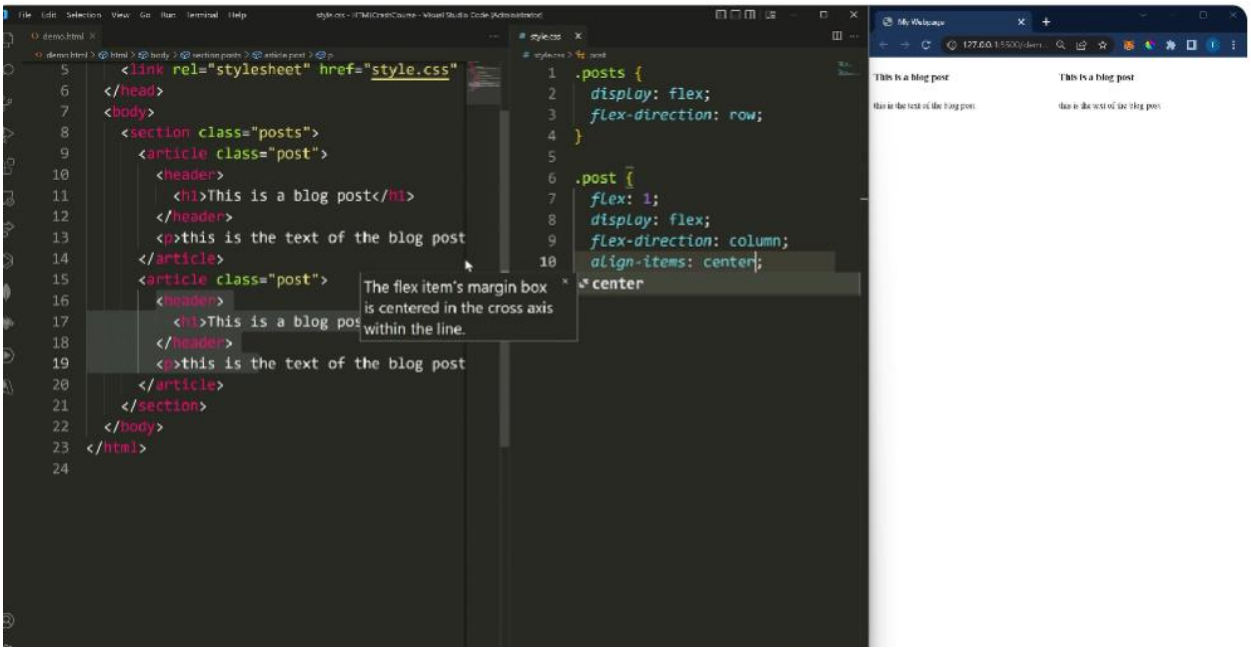
We have 4 articles and the flex used is 1, so the container will be 1/4 so 25% for each element. With the background color set in green.



We can change the height of the container by adding a vh property.



To adjust the element to the center of the document we can use the tag align-items and specify its center.

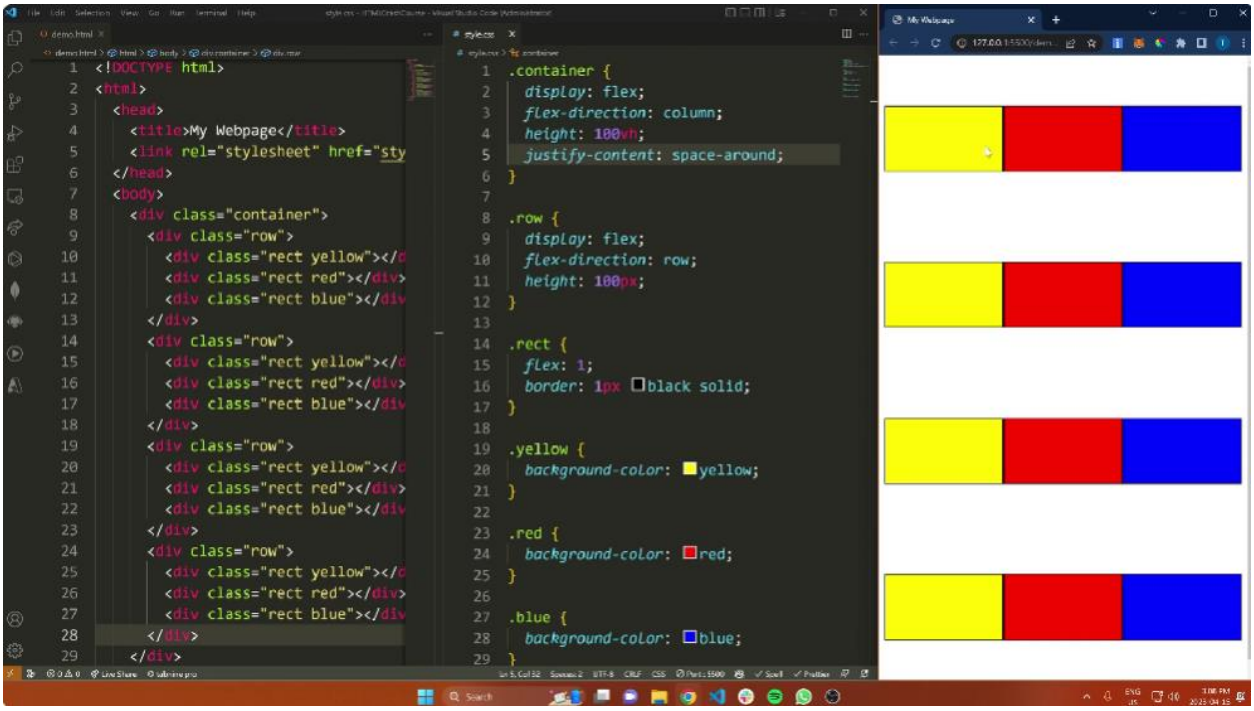


If we want the alignment to be made according to the main axis, we could use the property “justify-content: center”

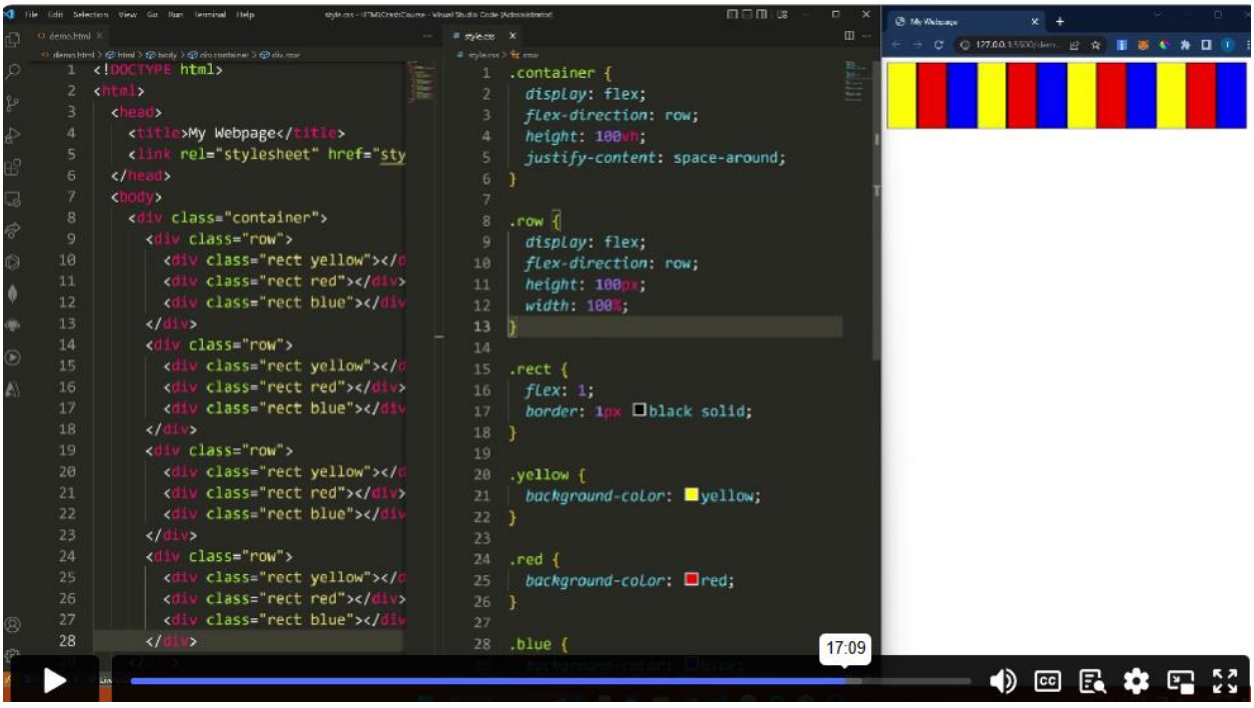
If we want to push the element to the very bottom of our container we can use flex-end.

The main axis by default is a row, so the direction is going to be horizontal. When we want the elements to align, they are going to follow that logic. The cross axis is by default vertical. If we want the main axis to be vertical, we will specify it to be column instead of row.

If we create a container that we divided into 3 rows and 3 columns, but we want the row to be separated, we can use the property space-between. We can also use the property space-around which is the same except it will also include the space around the container not just within the container.



If we change the flex-direction from column to row, the result will show up as 4 vertical black lines in a row, with space between them. Because since it's a row, they're by default going to take up the minimum amount of space and the visual will be messed up. To avoid that issue we need to specify the width of those elements, so we'll use the vw property. The result will be this:



Space-around and space-between can only be used in the main axis