Now that you have a basic foundation of HTML, it's now time to move on to the next step. The next step is to now learn CSS.

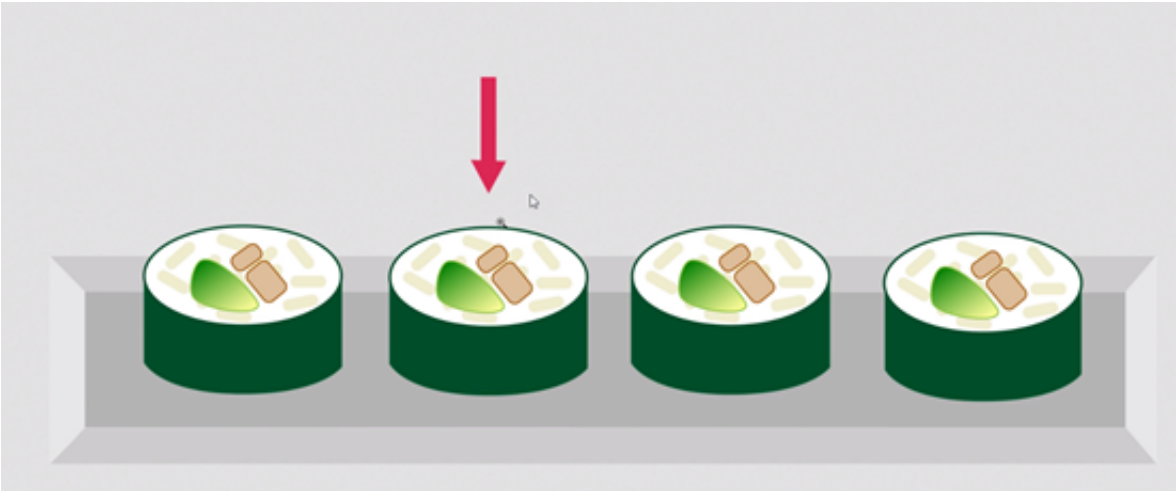
HTML allows you to **represent** the **content** on your page, it allows you to tell the browser, give me a title, give me a paragraph, show a table here, show a form here, let's add these fields to the form, let's show a list. It basically allows you to create all the content. HTML doesn't tell the browser how things should look. What sort of font should be used, the size of the font, the background color of the web page, how elements are positioned to each other, how much spacing to give elements, and how much border we assign to a certain container.

All these questions above can be answered by using CSS. **CSS** allows you to **make** your web **pages** aesthetically **pleasing**.

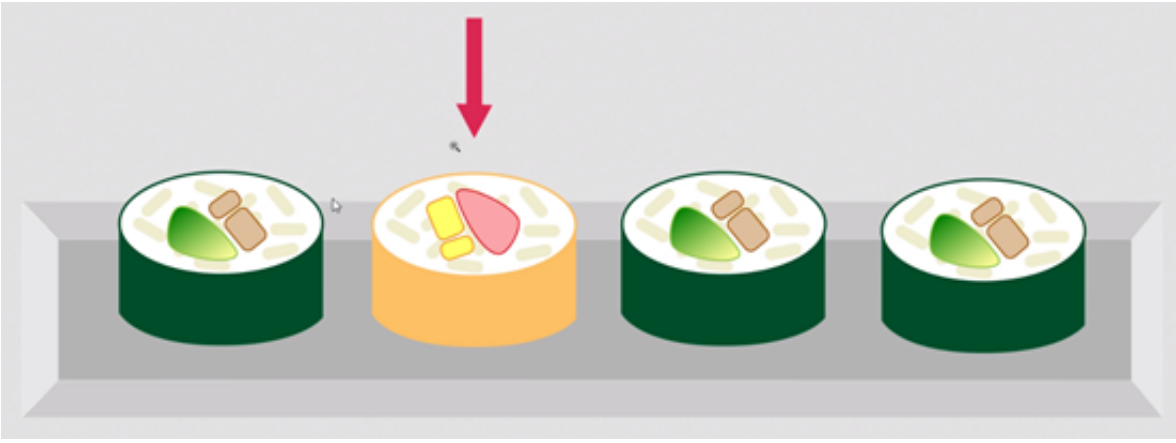
Working With CSS

When working with CSS it's always a two step process. The **first step** is to **select** an **element**. Then once you select it you **modify it**. You apply the CSS rules to change the way it looks.

A good analogy would be this, CSS has a stage where you select an element, the element that you want to modify.



Stage two is that you modify the element, you change its color, you change its font, you change its dimensions. You use all sorts of magic CSS properties to make it look pretty.



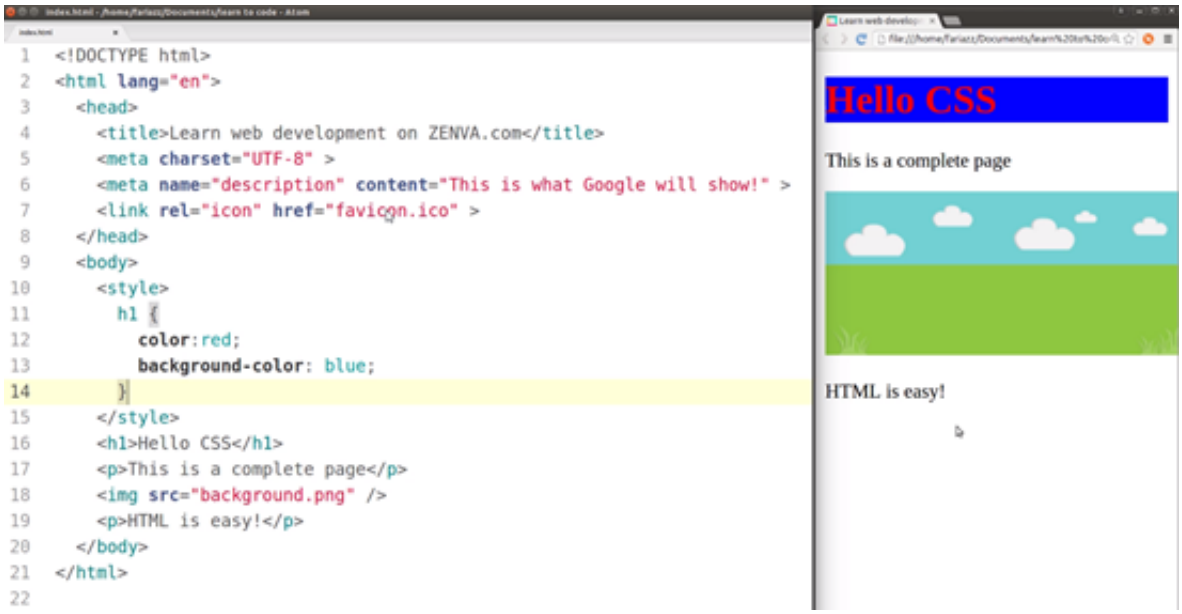
Hello World CSS

Let's now look at some code and do a hello world CSS.

We have a basic page, let's change the title. There are all sorts of ways to add CSS to a page, but we will look at just one of them for this lesson.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Learn web development on ZENVA.com</title>
    <meta charset="UTF-8" >
    <meta name="description" content="This is what Google will show!" >
    <link rel="icon" href="favicon.ico" >
  </head>
  <body>
    <style>
      h1 {
        color:red;
        background-color: blue;
      }
    </style>
    <h1>Hello CSS</h1>
    <p>This is a complete page</p>
    <img src="background.png" />
    <p>HTML is easy!</p>
  </body>
</html>
```

Save this and **refresh** the page.



CSS is **not** used for **content**, it's **used** to style content, to make content **look pretty**.

One way to include CSS on a page is by using **style tag**, which you need to **open and close**, and that needs to go **above** the **HTML** content.

Inside your **style tag** you can **add CSS** rules.

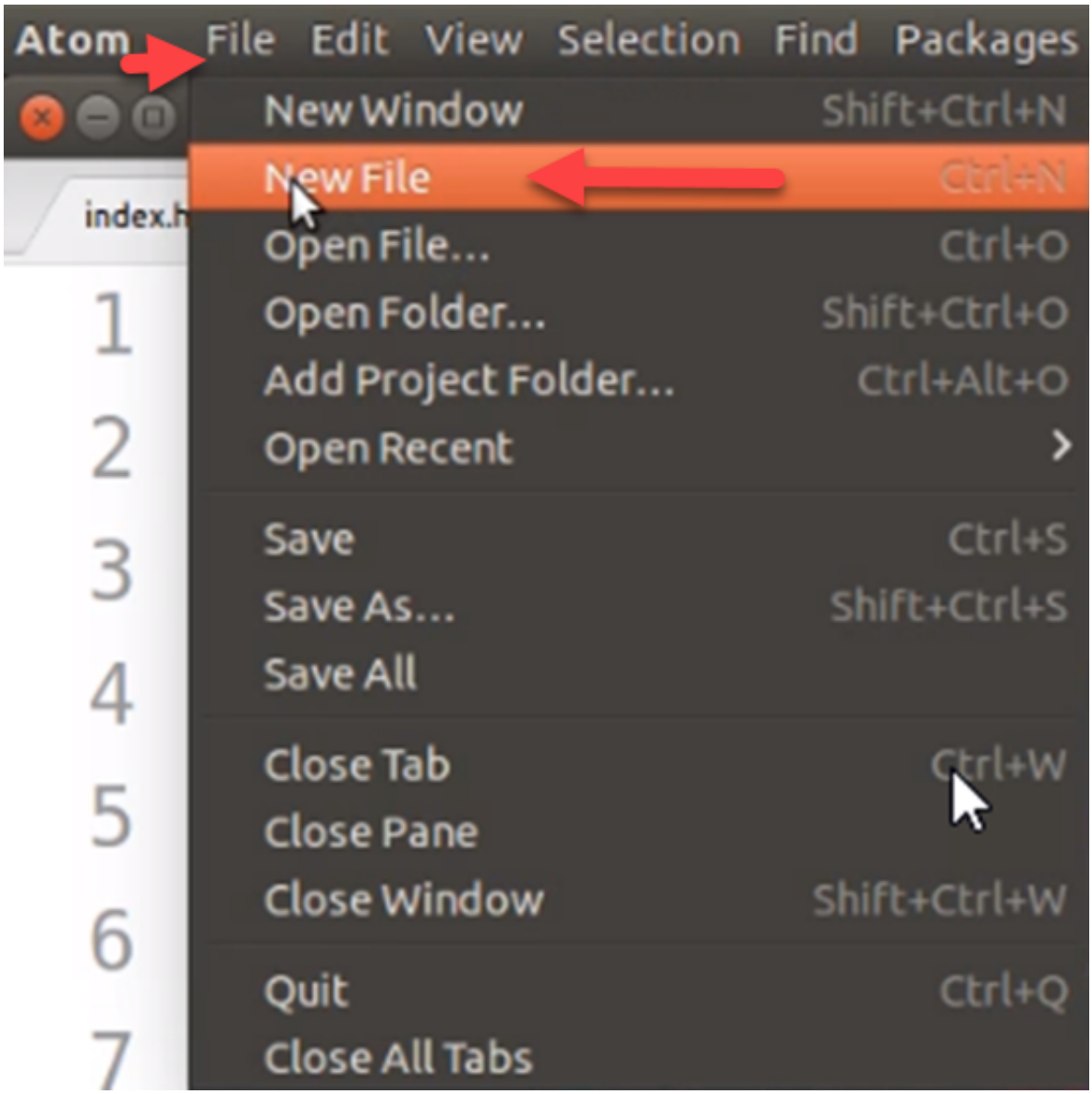
You always need to select an element and then you need to modify it, that is how CSS works.

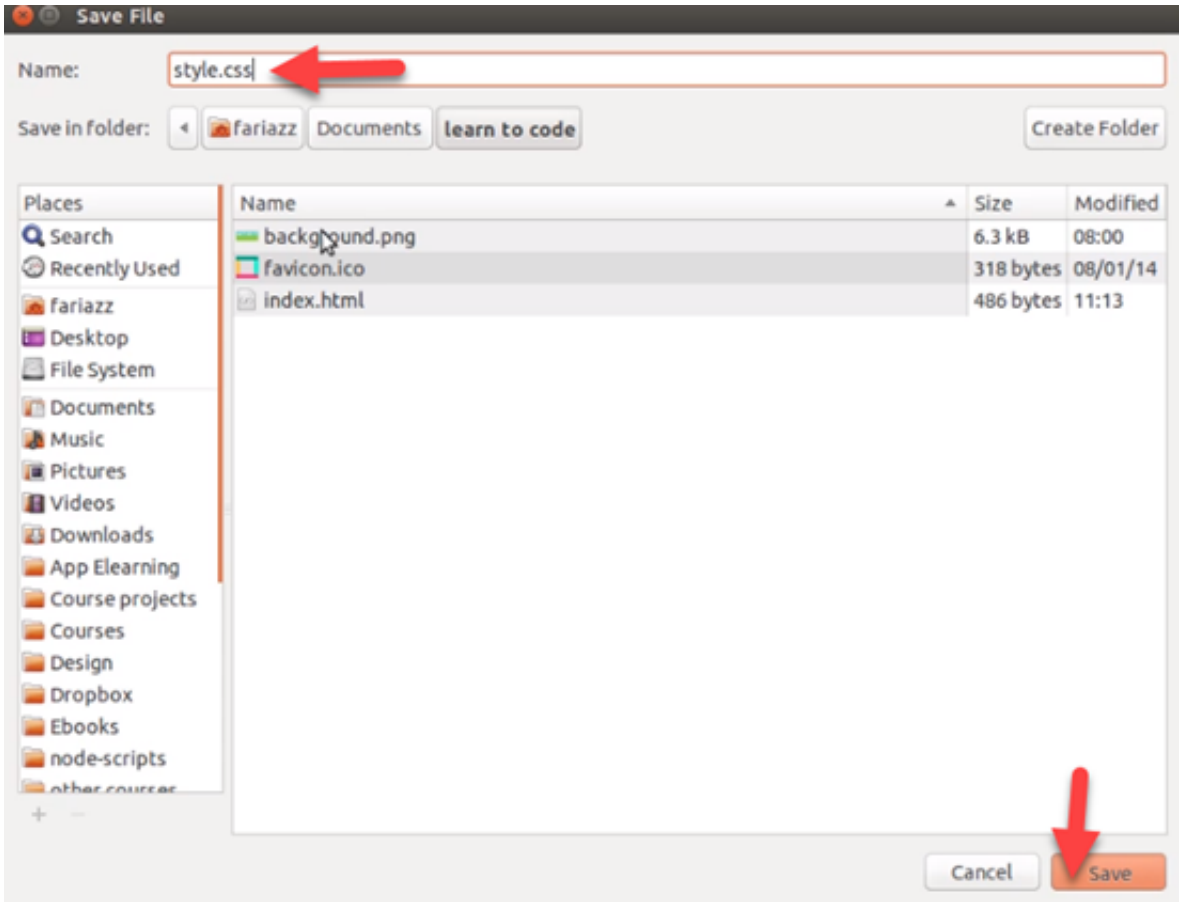
In this lesson you will learn to include your CSS code in a different file. This is so you will not have everything mixed up in one file.

CSS in External Files

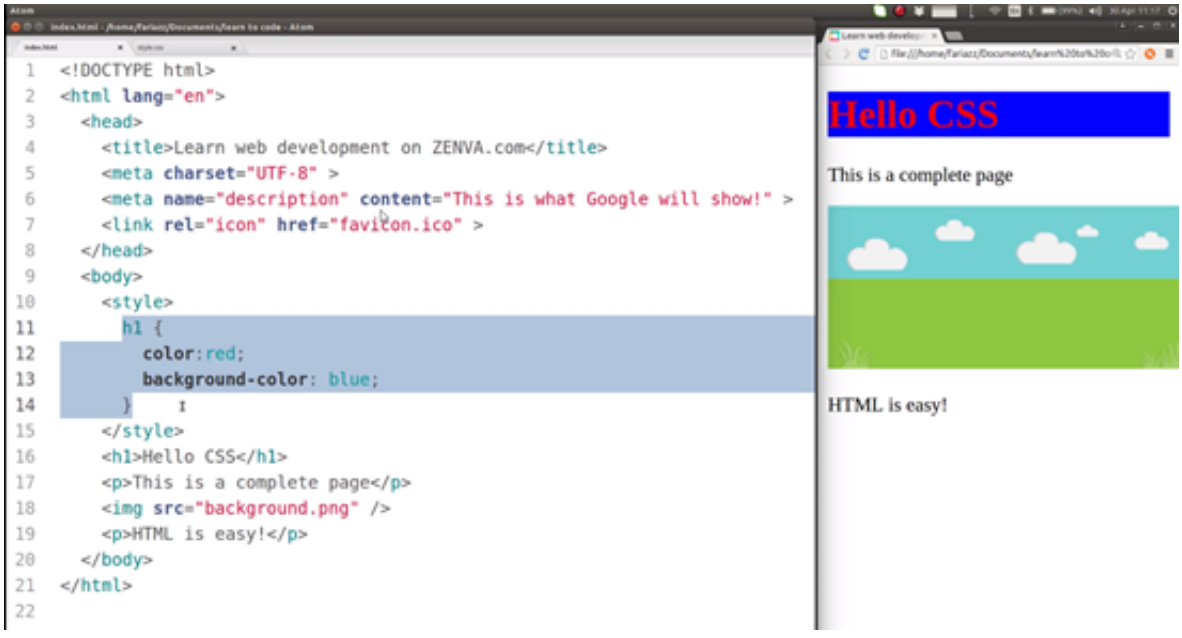
The way to **include CSS in external files** is to create a new file called “CSS.”

So go to **File>New File**, save this file, and name it **style.css**.





Next, **copy the CSS code** we did earlier in he previous lesson, **cut it** and remove the style tags.

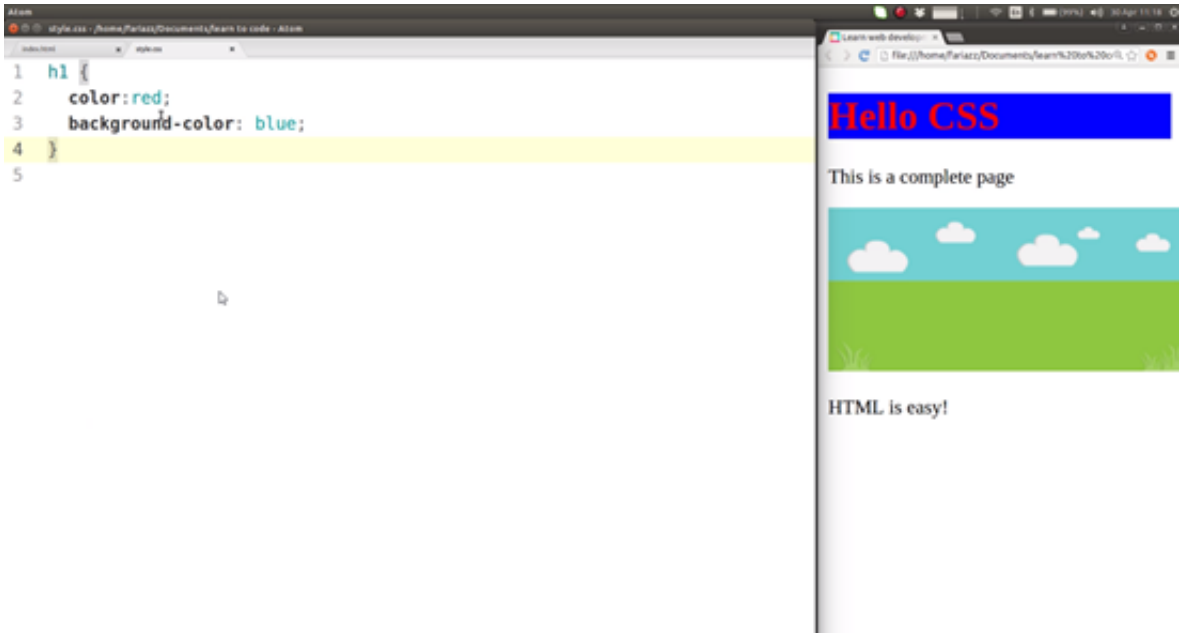


Then **paste** the code into the **new CSS file** we created.

```
h1 {
  color:red;
  background-color: blue;
```



```
}
```



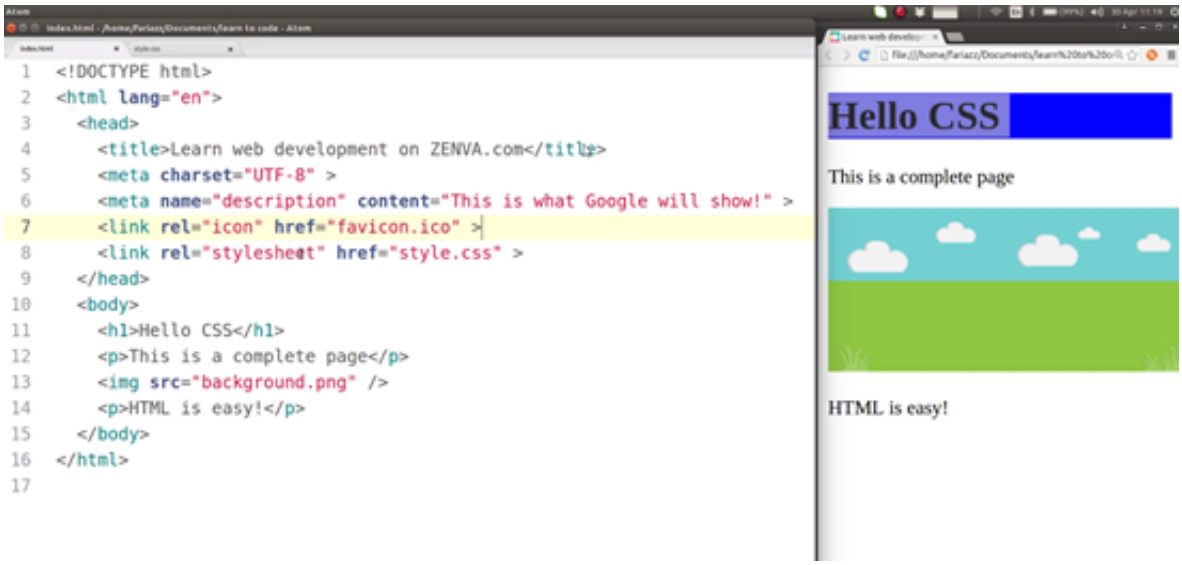
The last part is to **include this file** somehow.

In order to include the new file we will use this same **link tag** which is used to include external files, like CSS or the favicon, but the **rel** in this case is going to be **stylesheet**. This is what you type for the **attribute**. The **href** will indicate the **location of the file**. In this case the location of the file is in the same folder as the rest of the page. If it was located in a sub folder we would have to type the name of the sub folder and forward slash.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Learn web development on ZENVA.com</title>
    <meta charset="UTF-8" >
    <meta name="description" content="This is what Google will show!" >
    <link rel="icon" href="favicon.ico" >
    <link rel="stylesheet" href="style.css" >
  </head>
  <body>
    <h1>Hello CSS</h1>
    <p>This is a complete page</p>
    <img src="background.png" />
    <p>HTML is easy!</p>
  </body>
</html>
```

Save this and **refresh** the page.

You will see that the CSS is showing up because it was included as an external file.



Normally it is recommended to put **CSS** into an **external file** so that you can work separately in different files and the project will be made more **manageable** by doing this.

There is another way to include CSS, but this way **isn't recommended**, but you do have to be aware it exists because you might see it around. It is done by **including CSS as an attribute**. So you can add a style attribute to the tag and then add CSS rules to it.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Learn web development on ZENVA.com</title>
    <meta charset="UTF-8" >
    <meta name="description" content="This is what Google will show!" >
    <link rel="icon" href="favicon.ico" >
    <link rel="stylesheet" href="style.css" >
  </head>
  <body>
    <h1>Hello CSS</h1>
    <p style="color:green;">This is a complete page</p>
    <img src="background.png" />
    <p>HTML is easy!</p>
  </body>
</html>
```

So doing it this way in the code above, we do not make a selection because we are already selecting by placing our style attribute.

Save this and **refresh** the page.

You can see how the text changed to green.

This way is **not recommended** because you do want to have separation between the content and the presentation of the content, and the best way to achieve this is by putting it an external file.

Summary

Separation of content and presentation is important.

Put all your CSS files in an external file.

Include the CSS file using the link tag.

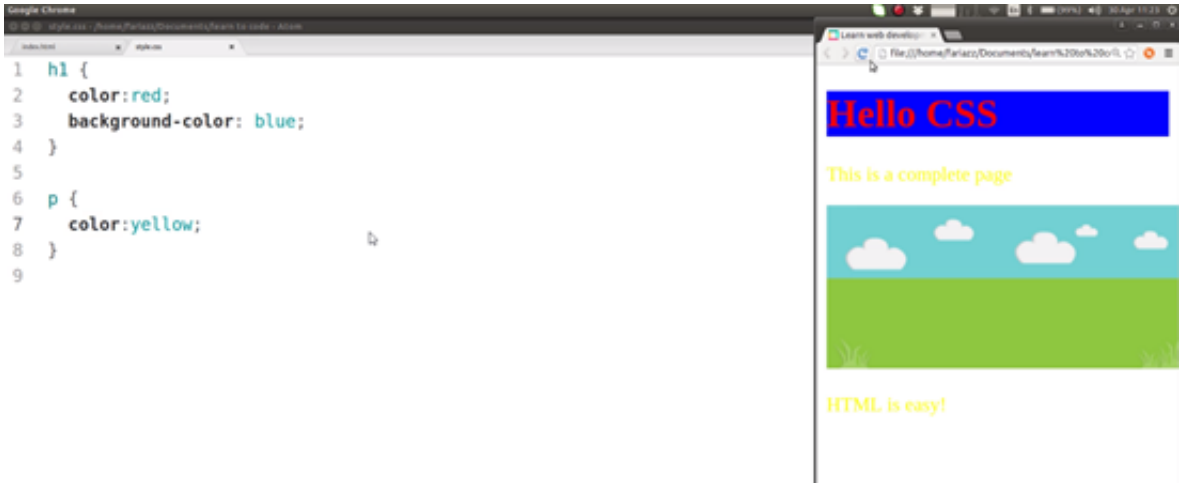
Inside your CSS file you can then add any CSS code, and you don’t need to add any style tags.

In this lesson we will talk about **ID's**, and how we can use ID's to **select single elements** in our HTML page.

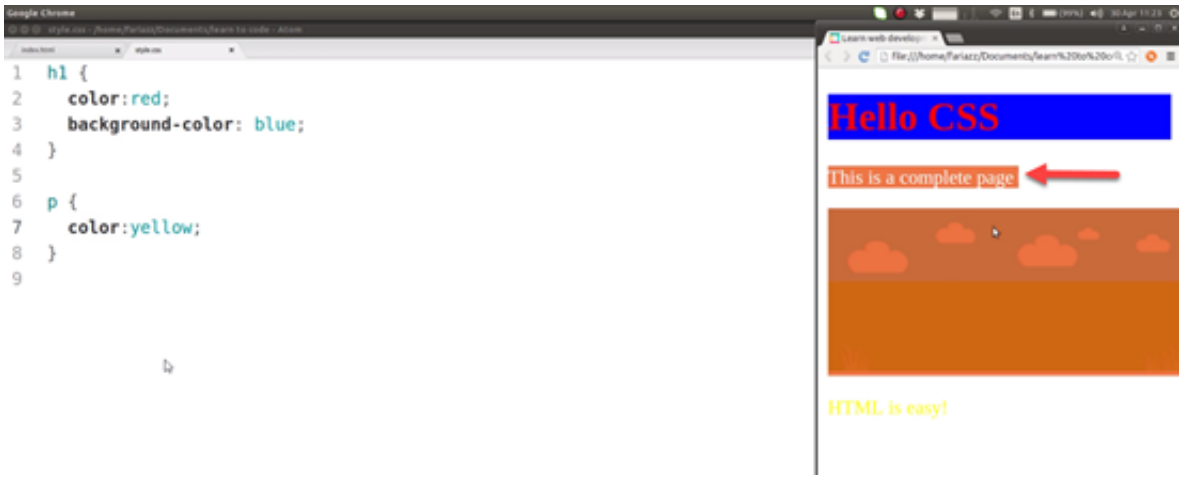
For example if we selected this paragraph here:

Then, if I go and type in **"p"** that will **select all the paragraphs** in the page.

```
h1 {
  color:red;
  background-color: blue;
}

p {
  color:yellow;
}
```



But, if you wanted to just select this one below:



There are multiple ways to do this in **CSS**, but you will be shown how to do it through the concept of **ID** for this particular example.

We can give this paragraph a **unique ID**. For that we need to add the **attribute ID**, equal sign, double quotes, and in here we enter the unique ID of that particular element. An **ID is a unique identifier**. It cannot contain spaces.

You can say something like mainPar, and if you are going to use upper case you need to be consistent with that in your CSS code.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Learn web development on ZENVA.com</title>
    <meta charset="UTF-8" >
    <meta name="description" content="This is what Google will show!" >
    <link rel="icon" href="favicon.ico" >
    <link rel="stylesheet" href="style.css" >
  </head>
  <body>
    <h1>Hello CSS</h1>
    <p id="mainPar">This is a complete page</p>
    <img src="background.png" />
    <p>HTML is easy!</p>
  </body>
</html>
```

So, the element now has an ID. So, instead of selecting all the paragraphs like this:

We can select them by ID instead, but we need to type in the **hash sign**, and then the **name of the ID**, and that will allow us to **select a single element by ID**.

```
h1 {
  color:red;
  background-color: blue;
}

#mainPar {
  color:yellow;
}
```

You can have multiple elements in your page that have different ID's, but ID's need to be unique.

You cannot have two elements that have the same value for ID.

In this lesson you will be shown how to **select** elements by **class**.

We will start by doing this with and empty style sheet, and a different example on HTML. Which is just an unordered list with a few food items.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Learn web development on ZENVA.com</title>
    <meta charset="UTF-8" >
    <meta name="description" content="This is what Google will show!" >
    <link rel="icon" href="favicon.ico" >
    <link rel="stylesheet" href="style.css" >
  </head>
  <body>
    <ul>
      <li>Spinach</li>
      <li>Chicken</li>
      <li>Pasta</li>
      <li>Carrot</li>
      <li>Cucumber</li>
      <li>Fish</li>
    </ul>
  </body>
</html>
```

What we want to do is make all the elements that are vegetables the color green.

So this would be spinach, carrot, and cucumber.

Then we want to make all the meat elements the color brown.

We can do this by **selecting by tag**, by the **li tag** but that would **select all the elements**.

If we used the **selection by ID**, we would only be able to **select one element** or we would have to **repeat** the style for **each elemen**t. Because, you can only select one element by ID. You cannot have an ID shared by more than one element.

This is when **classes** come into play. Class is another attribute, that can be entered with **class**. See the code below:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Learn web development on ZENVA.com</title>
    <meta charset="UTF-8" >
    <meta name="description" content="This is what Google will show!" >
    <link rel="icon" href="favicon.ico" >
    <link rel="stylesheet" href="style.css" >
  </head>
  <body>
    <ul>
      <li class="veggie">Spinach</li>
```



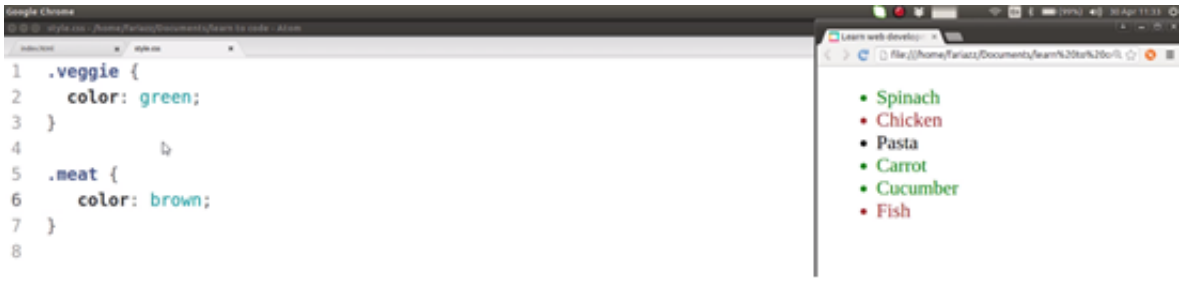
```
      <li class="meat">Chicken</li>
      <li>Pasta</li>
      <li class="veggie">Carrot</li>
      <li class="veggie">Cucumber</li>
      <li class="meat">Fish</li>
    </ul>
  </body>
</html>
```

Save this and then in **CSS**, we can refer to that **class** and give it a **style**, and this can be any style you want.

See the code below:

```
.veggie {
  color: green;
}

.meat {
   color: brown;
}
```

Save this and r**efresh** the page.

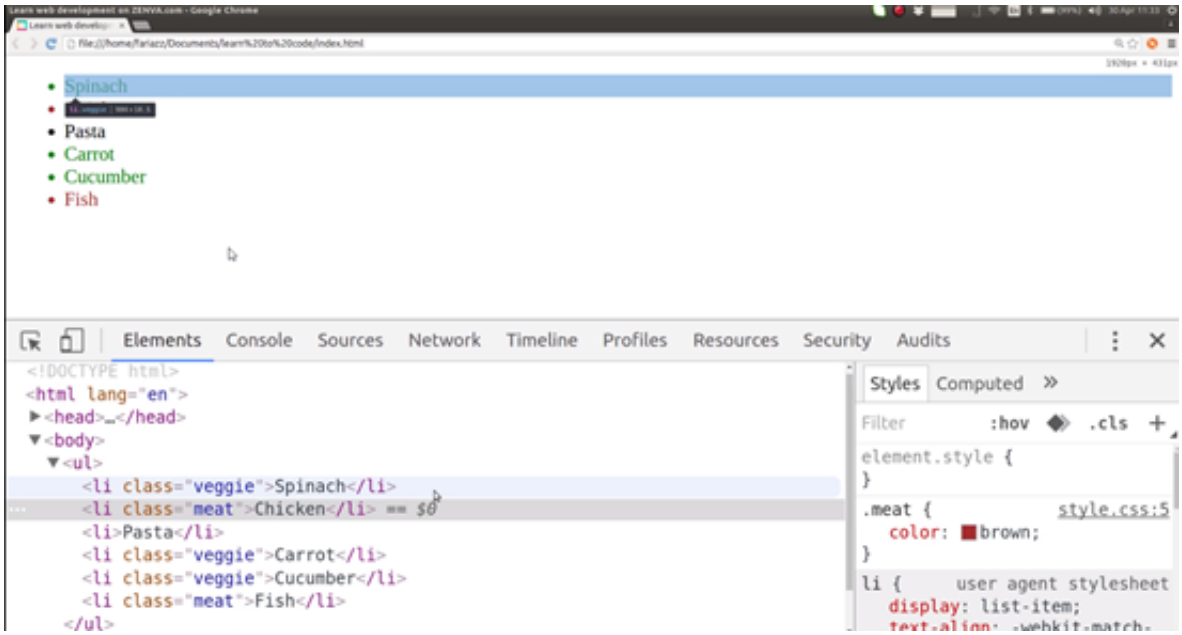


Classes allow you to select multiple elements, but you can also just select one this way. This is a powerful tool to use when your using CSS.

All of this has to deal with the selection part, so now you know how to select by tag, you know how to select by ID, and now you know how to select by class.

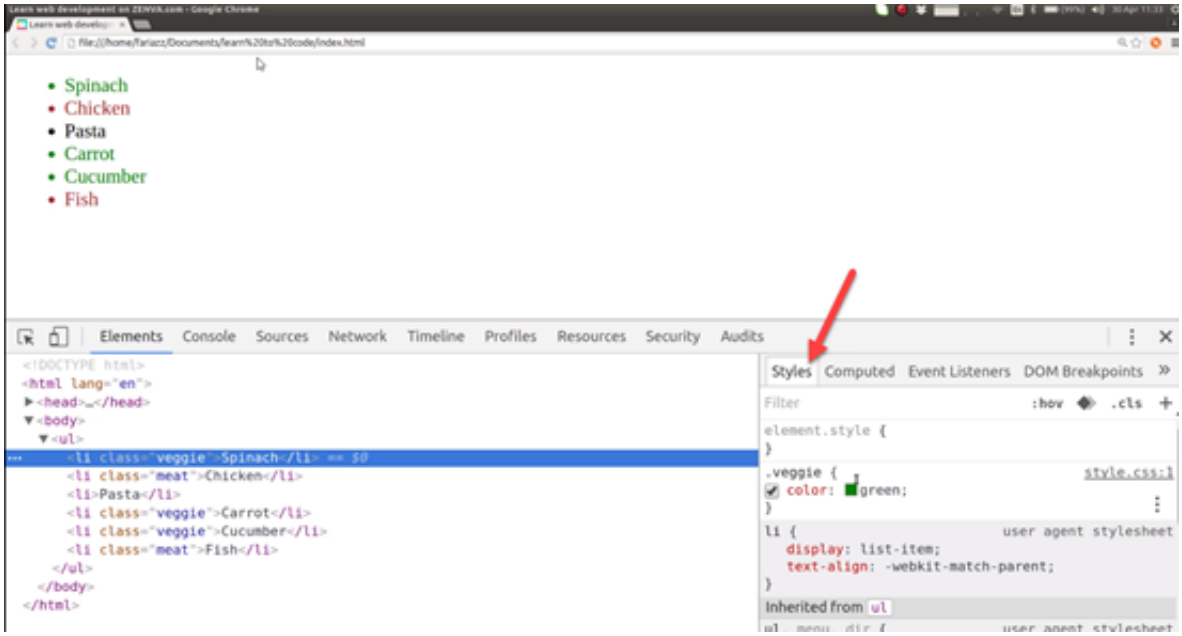
Chrome Developer Tools and CSS

The Chrome Developer Tools have some interesting features for CSS. If you **right click** on the page and then select **Inspect**, it will bring up the **Element Inspector**.



This will show you what the **classes are of each element**, so once you start looking at real web sites you will see that they all have classes and ID's.

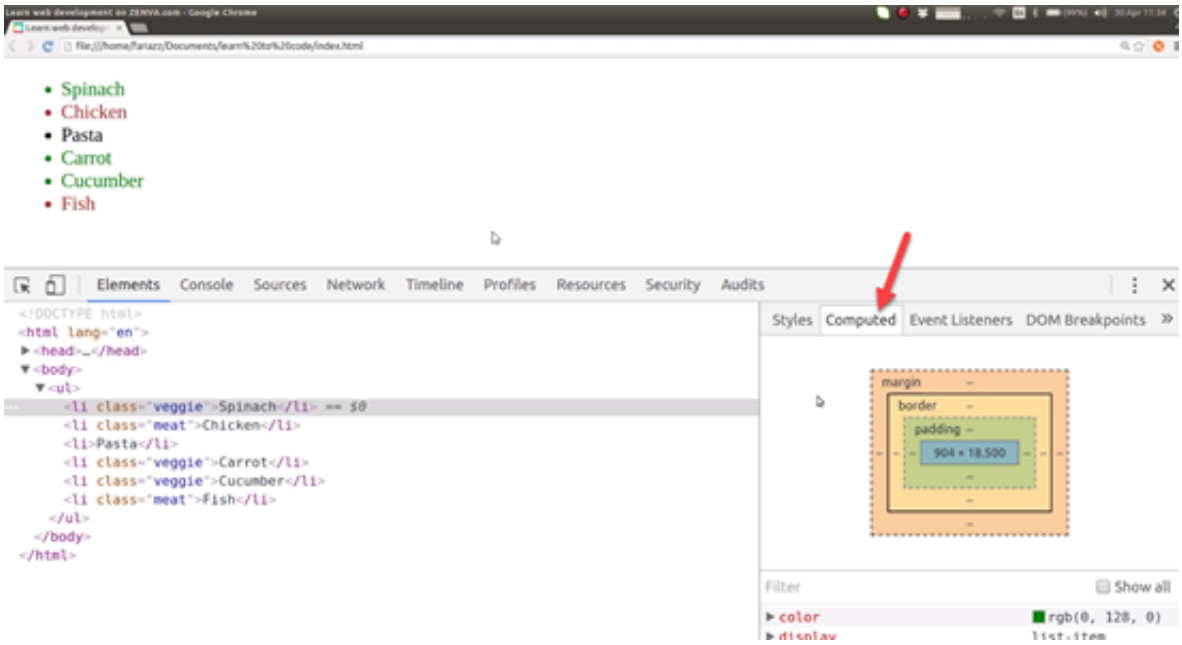
Then in the **Styles section** you can see as well what the style elements the selected element has.



So the example above is showing the veggie has the color property of green. You can also **deactivate** it just to see what it looks like, or you can add some more features. None of these changes made like this would get **saved** once you **refresh** the page. This is just for you to try things out.

You will see other styles there too, and all the **user agent stylesheet** are **default** styles. Your browser has default styles so that things already look a little bit styled, like headings would already have larger fonts.

If you go to **Computed** it will show a summary of all the styles that apply to that particular element.

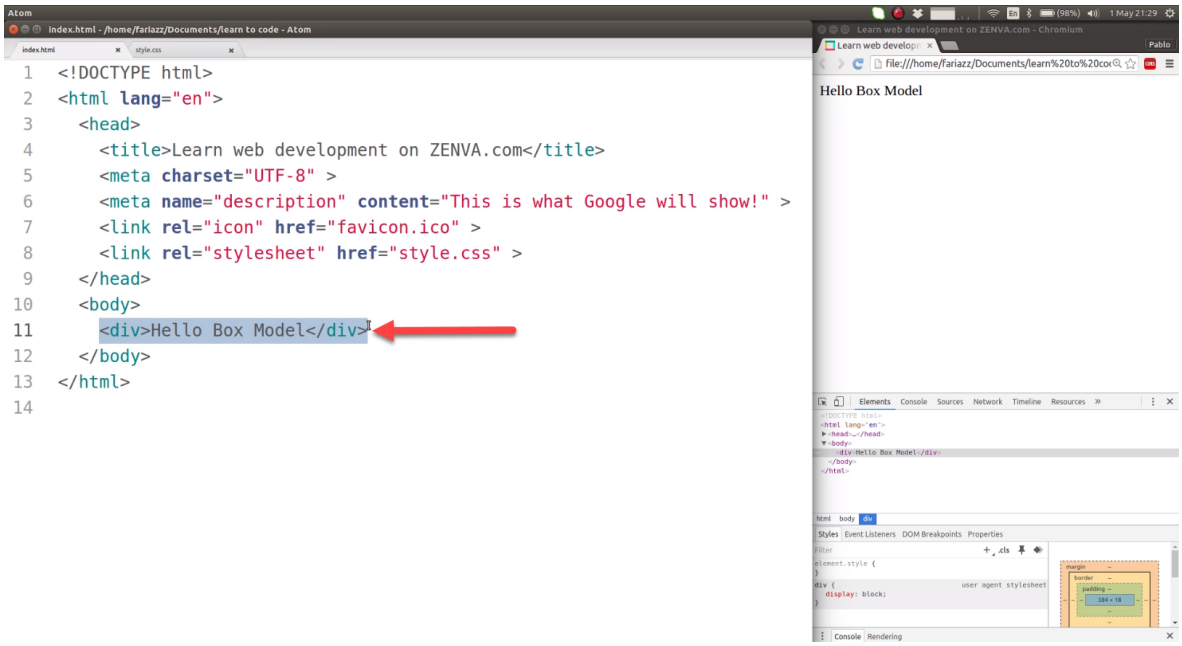


This shows you a few things you can do with the Chrome Developer Tools, and how you can explore CSS on the right side using the Styles section.

In **CSS** containers are represented by **rectangles**. You can check this out by going to any **div container** that you may have previously created and **Inspecting** that **element** in the Chrome Developer Tools. This will quickly highlight a **whole rectangle**. Those rectangles have properties and those **properties** are what are known as the **box model**. By accessing and **modifying** those **properties** you can **change** how the rectangle will **look**. You can give the rectangle a certain width, height, add padding to the content, add borders, and add margins to the other elements.

The Box Model

The first thing we want to do is give this container a **unique ID** so that we can select in in CSS:



See the code below for the HTML:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Learn web development on ZENVA.com</title>
    <meta charset="UTF-8" >
    <meta name="description" content="This is what Google will show!" >
    <link rel="icon" href="favicon.ico" >
    <link rel="stylesheet" href="style.css" >
  </head>
  <body>
    <div id="box">Hello Box Model</div>
  </body>
</html>
```

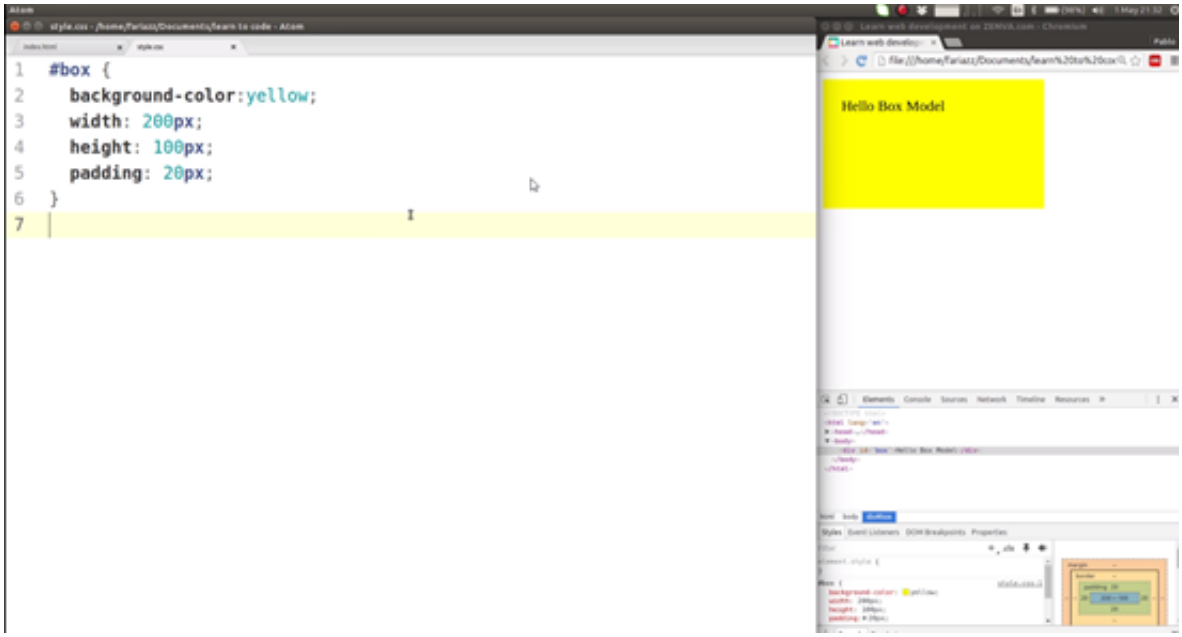
See the code below for the CSS:

```
#box {
  background-color:yellow;
  width: 200px;
  height: 100px;
```



```
  padding: 20px;
}
```

Save this and **refresh** the page.



Summary

Elements in CSS are represented as rectangles and by modifying the properties of this rectangle, which is the same as talking about the box model, that's just how CSS represents it's rectangles, you can change the width, the height, the padding, and we'll see a few more options in the next lesson.

In this lesson we will continue **exploring** the **Box Model**. We will look at some other properties, but some are just options in general.

Besides padding being added to the rectangle, you can also **add** a **border**.

Adding A Border Margin

Adding a border is easy, all you need to do is add the **border property**.

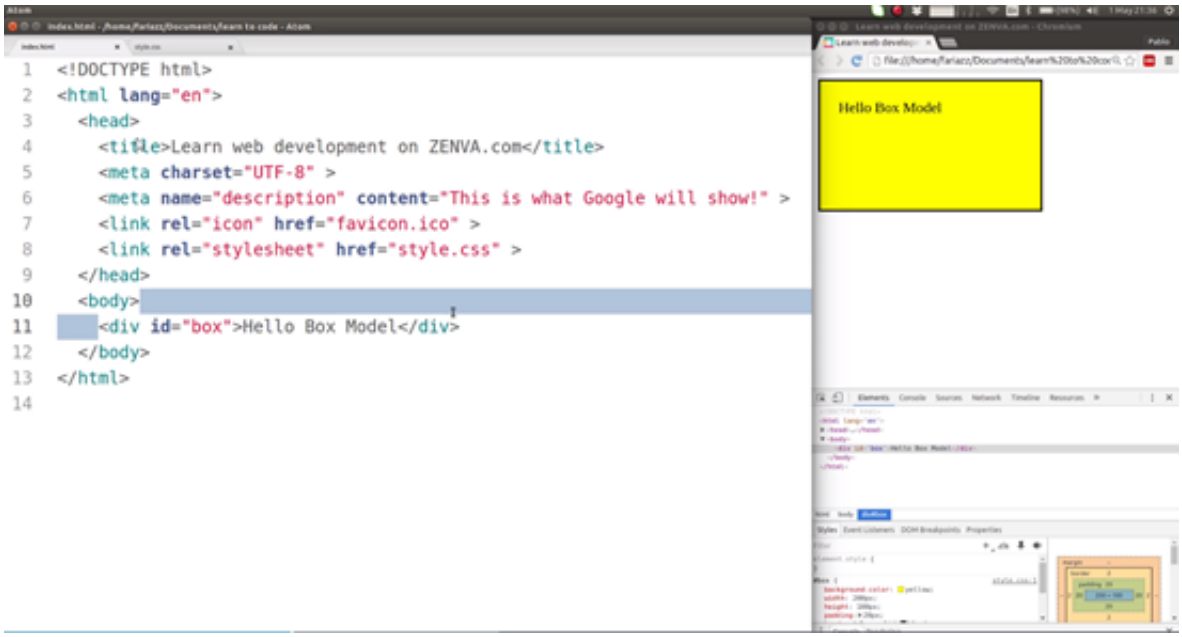
See the HTML code below:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Learn web development on ZENVA.com</title>
    <meta charset="UTF-8" >
    <meta name="description" content="This is what Google will show!" >
    <link rel="icon" href="favicon.ico" >
    <link rel="stylesheet" href="style.css" >
  </head>
  <body>
    <div id="box">Hello Box Model</div>
  </body>
</html>
```

See the CSS code below:

```
#box {
  background-color:yellow;
  width: 200px;
  height: 100px;
  padding: 20px;
  border: 2px solid black;
}
```

Save this and **refresh** the page.



And what if we had a second box that goes below the first one.

See the HTML code below:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Learn web development on ZENVA.com</title>
    <meta charset="UTF-8" >
    <meta name="description" content="This is what Google will show!" >
    <link rel="icon" href="favicon.ico" >
    <link rel="stylesheet" href="style.css" >
  </head>
  <body>
    <div id="box">Hello Box Model</div>
    <div id="box2">Hello Box Model</div>
  </body>
</html>
```

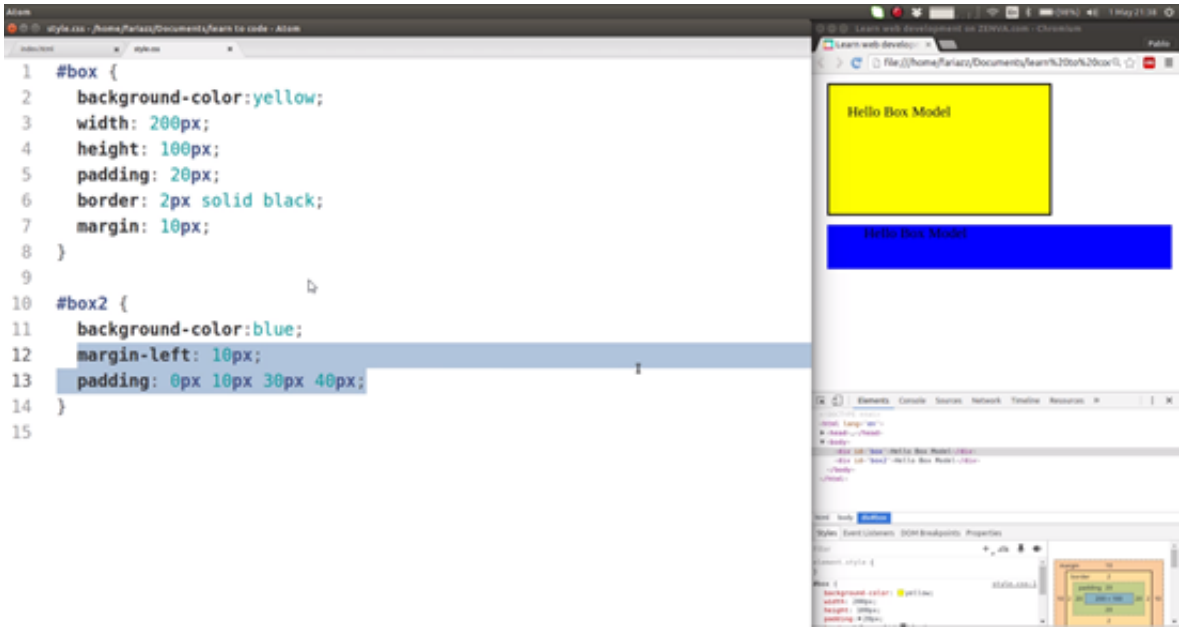
See the CSS code below:

```
#box {
  background-color:yellow;
  width: 200px;
  height: 100px;
  padding: 20px;
  border: 2px solid black;
  margin: 10px;
}

#box2 {
  background-color:blue;
  margin-left: 10px;
```



```
  padding: 0px 10px 30px 40px;
}
```

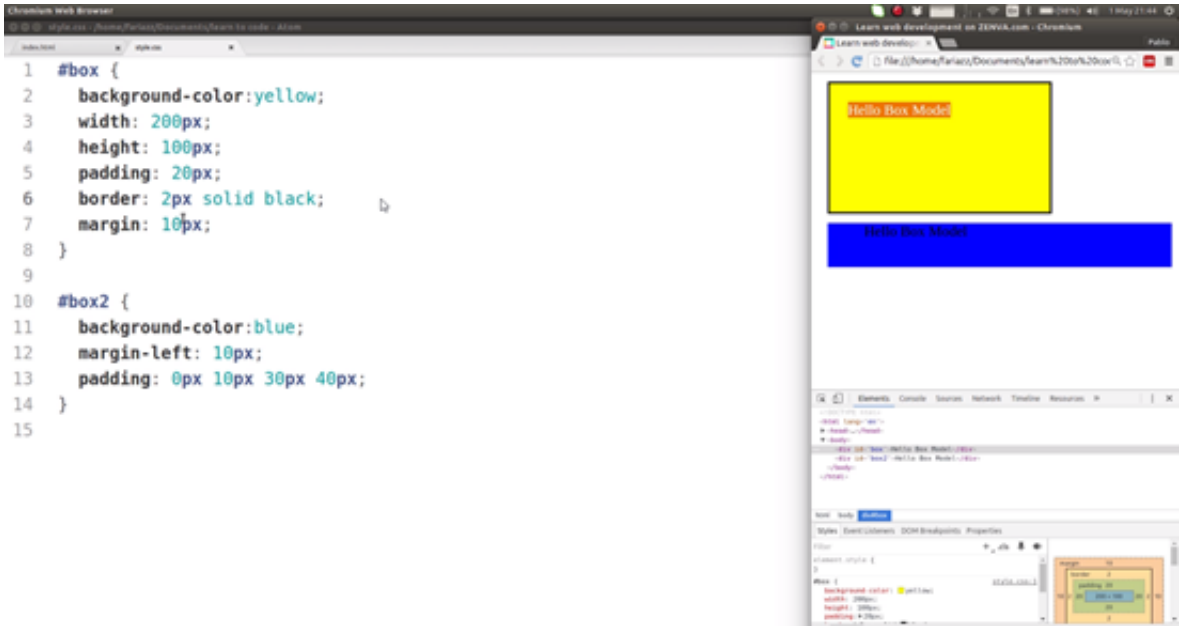
Save this and **refresh** the page.



Summary

The box model is quite popular and a powerful way for us to define how our content is going to look. So what dimensions it's going to have, what separation it's going to have from other elements, and also add options for the border and make the text or whatever content you have inside have a bit of padding too. This is a very important concept in CSS and you will be using this all the time when you create websites using CSS and HTML.

In this lesson we will look at some options to **align our content** the we want. We are going to begin by centering the Hello Box Model text that is the yellow rectangle from the previous lesson.



We can use the **text align property** for this. The **text align** property has different options such as: **left, right, center, and justify**.

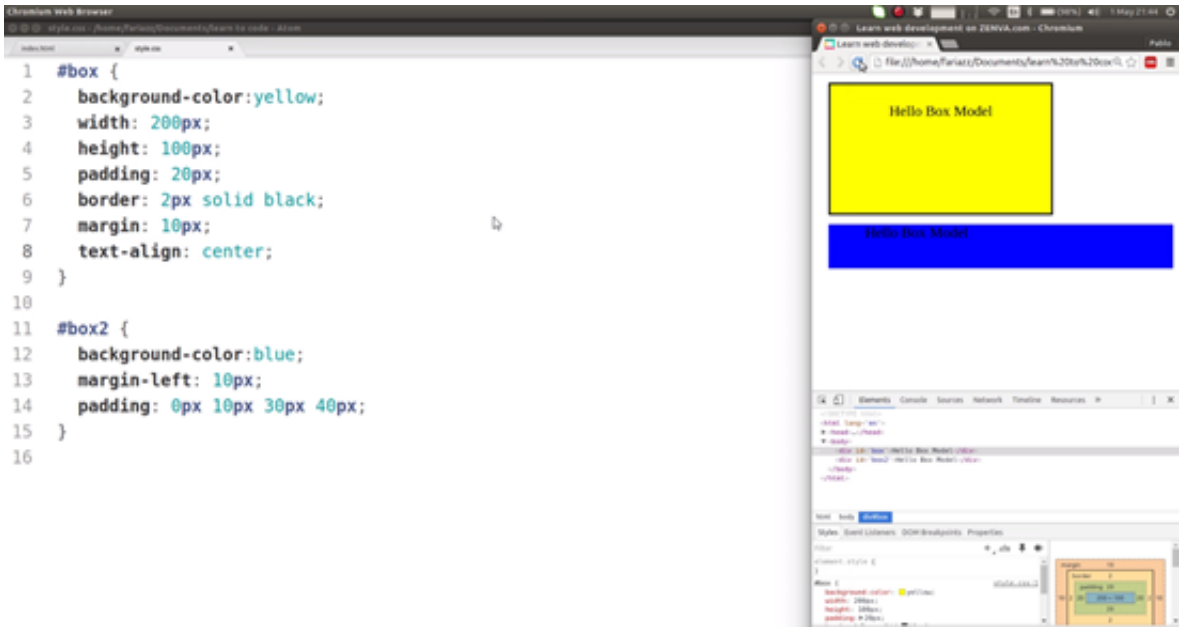
Center Alignment

Open the CSS code from the previous lesson and we will **add** the **text-align** property to the first box:

```
#box {
  background-color:yellow;
  width: 200px;
  height: 100px;
  padding: 20px;
  border: 2px solid black;
  margin: 10px;
  text-align: center;
 }

#box2 {
  background-color:blue;
  margin-left: 10px;
  padding: 0px 10px 30px 40px;
}
```

So we have set it to center.

Save this and **refresh** the page.



The area that it actually is going to center is the actual content area. So, remember that we have padding as well. If you had a lot of padding on one side the content would not look like its in the middle of the rectangle. It would just be in the middle of the available area that it has.

Vertical Alignment

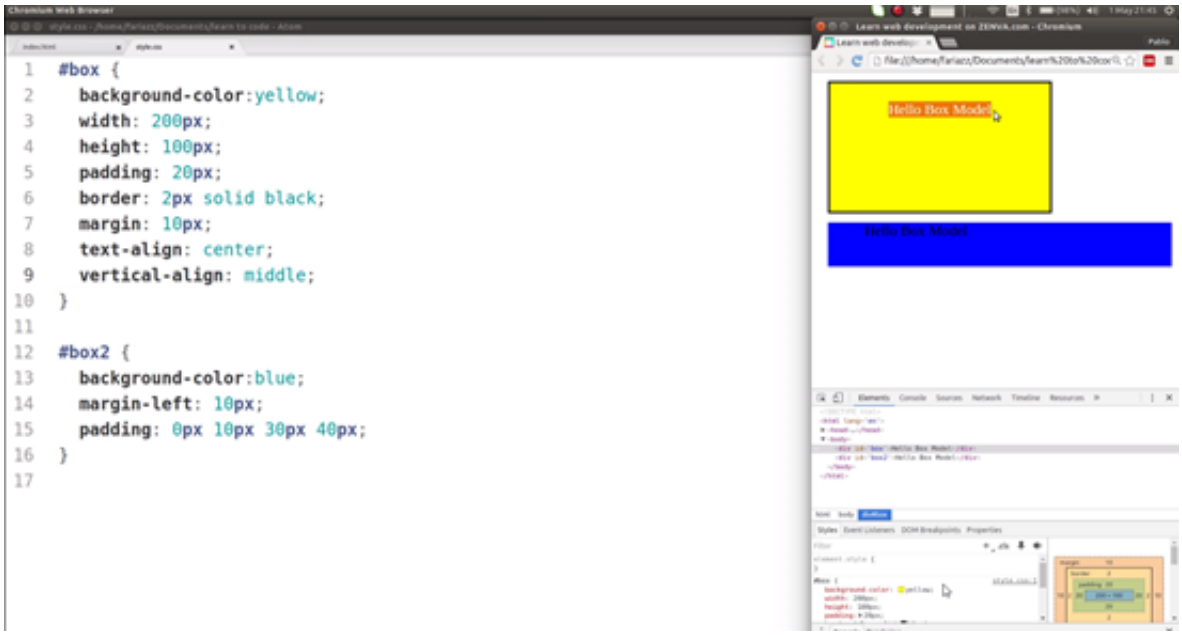
We can also make the content **align vertically**.

In order to do this we need to do a couple of things.

There is a property called **“vertical-align”** and this property has different options. The one that we are interested in now is the **“middle” option**, but as you will see it doesn’t align vertically like we want it.

See the CSS code below and then the screenshot:

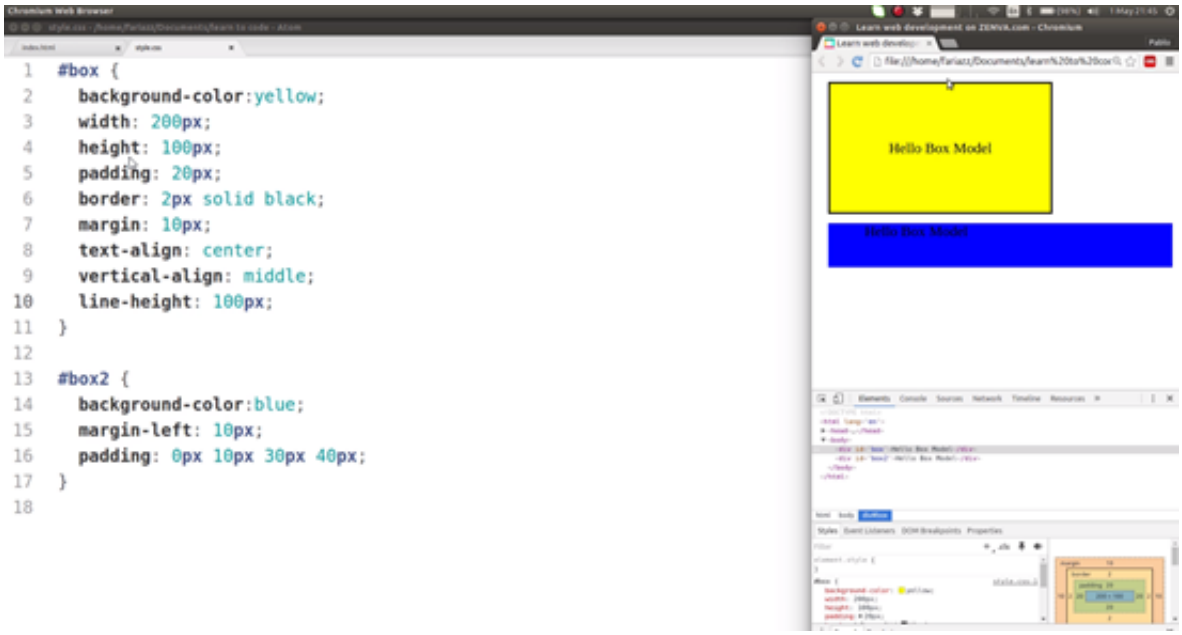
```
#box {
  background-color:yellow;
  width: 200px;
  height: 100px;
  padding: 20px;
  border: 2px solid black;
  margin: 10px;
  text-align: center;
  vertical-align: middle;
}
#box2 {
  background-color:blue;
  margin-left: 10px;
  padding: 0px 10px 30px 40px;
}
```

The reason why this is happening is because there is another property that we need to specify which is called the **“line-height”** and this is the **height that the text actually has to occupy**.

See the CSS code below and the screenshot:

```
#box {
  background-color:yellow;
  width: 200px;
  height: 100px;
  padding: 20px;
  border: 2px solid black;
  margin: 10px;
  text-align: center;
  vertical-align: middle;
  line-height: 100px;
}

#box2 {
  background-color:blue;
  margin-left: 10px;
  padding: 0px 10px 30px 40px;
}
```

If you **set** the **line-height** to the **same height of your box** it will give the text all the room so that it can actually **center correctly**.

Another thing we need to do for this lesson is actually center our whole div container. We want to have the whole div container in the middle of the page, and if we change the size of the page, we still want the div container to be int he middle.

We can do this by **defining the width**, which is the case we already have with the CSS code we wrote earlier. The way to do this is to specify an **automatic margin** on both sides.

Remember the margin can have all four sides on one call. So lets say we want the 10 we already have on the top for the margin, and we want to keep that, but then on the right side we need to specify auto.

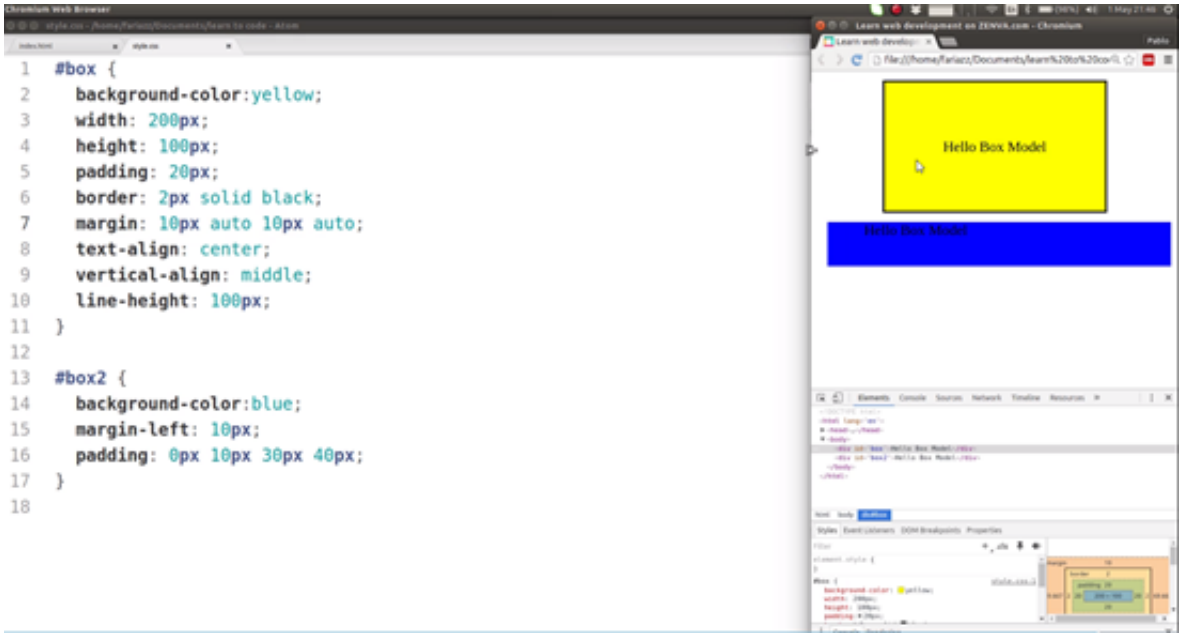
On the bottom let's say that we wanna keep that 10, but then on the left side we wanna keep that order.

See the CSS code and screenshot below:

```
#box {
  background-color:yellow;
  width: 200px;
  height: 100px;
  padding: 20px;
  border: 2px solid black;
  margin: 10px auto 10px auto;
  text-align: center;
  vertical-align: middle;
  line-height: 100px;
}

#box2 {
  background-color:blue;
  margin-left: 10px;
  padding: 0px 10px 30px 40px;
```



```
}
```



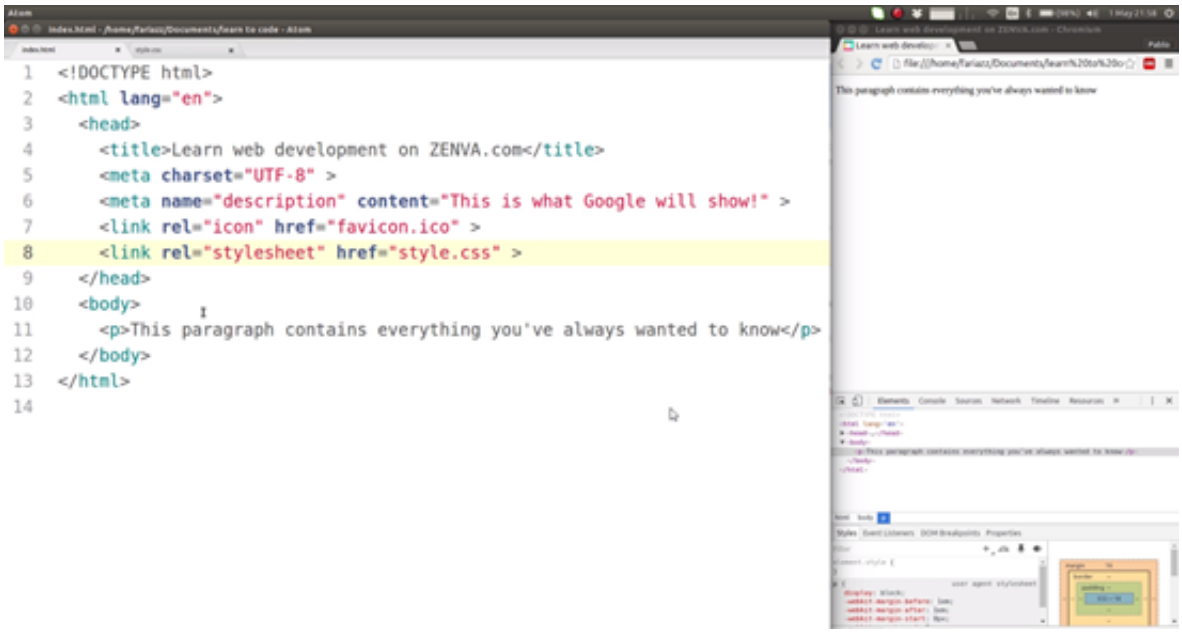
So if the **window** was **resized** the content is still **centered horizontally**, so we have looked at the different options and properties that have to do with alignment of the content, either text or whole blocks.

In this lesson we will talk about typography. **Typography is used to change fonts**.

See the HTML code below for a simple example of a text paragraph and I am going to start by setting the size of the font:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Learn web development on ZENVA.com</title>
    <meta charset="UTF-8" >
    <meta name="description" content="This is what Google will show!" >
    <link rel="icon" href="favicon.ico" >
    <link rel="stylesheet" href="style.css" >
  </head>
  <body>
    <p>This paragraph contains <span class="keypoint">everything</span> you've always
 wanted to know</p>
  </body>
</html>
```

This is how the page looks:



So normally on a page you have a **font size** that is a **default** for the whole page. By default it is just 16 pixels, it's just a unit that is used to give fonts size.

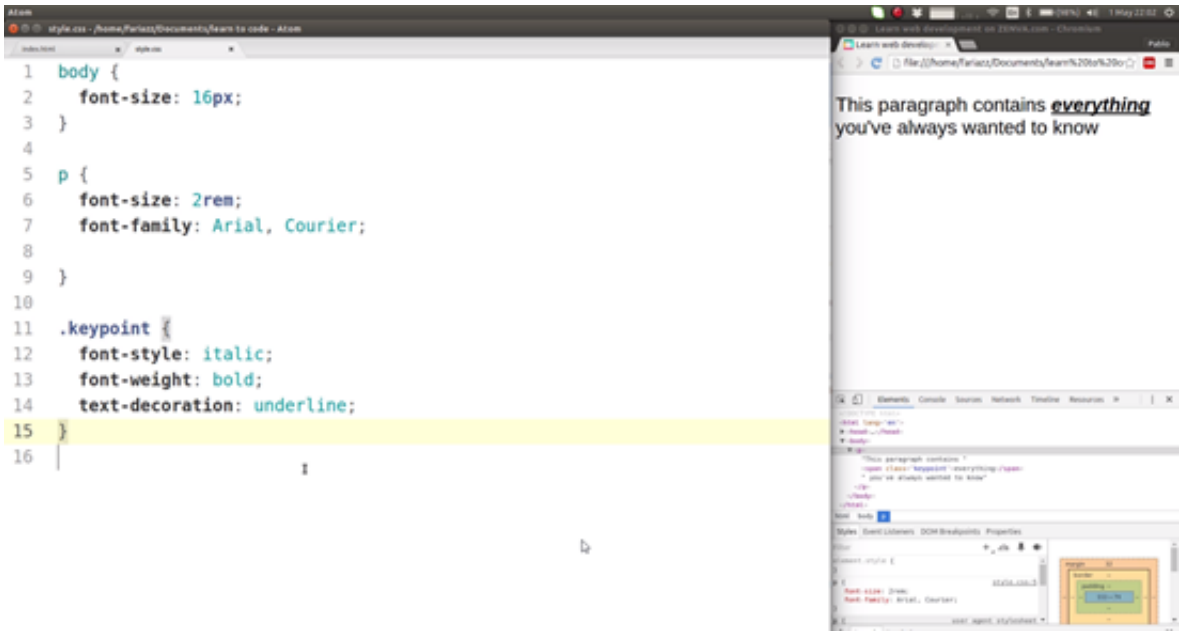
You can make that explicit. See the CSS code below:

```
body {
  font-size: 16px;
}
```


Save this and **refresh** the page and you will see the font size has not changed. This is because **16 is default**. We will change this see the CSS code below:

```
body {
  font-size: 16px;
}

p {
  font-size: 2rem;
  font-family: Arial, Courier;

}

.keypoint {
  font-style: italic;
  font-weight: bold;
  text-decoration: underline;
}
```

Save this and r**efresh** the page.



So as you can see working with text with just these basic options will allow you to do pretty much anything with text.

There are of course **more options** that you can explore in the **Mozilla Developer Network** that was mentioned in a previous lesson.

What you have learned here in this lesson is just the basics for text options using **Typography**.

Summary

There is the font sizing aspect, and there are different options you could in theory just use pixel on

every single one, but it’s recommended to use the relative units. So it could be REM which always will depend on the root value. For font family you can specify one or multiple fonts.

You can add decoration to the text, but if you wanna select just one word or one phrase you could use span, give it a class, and then refer to that class here, select by class, and use options such as italic or bold, and then underlining text as well.

Most computers have the most popular **fonts** like Times New Roman, Courier, Arial, etc.

There are going to be some times when you are making web pages that you will want to use some special fonts.

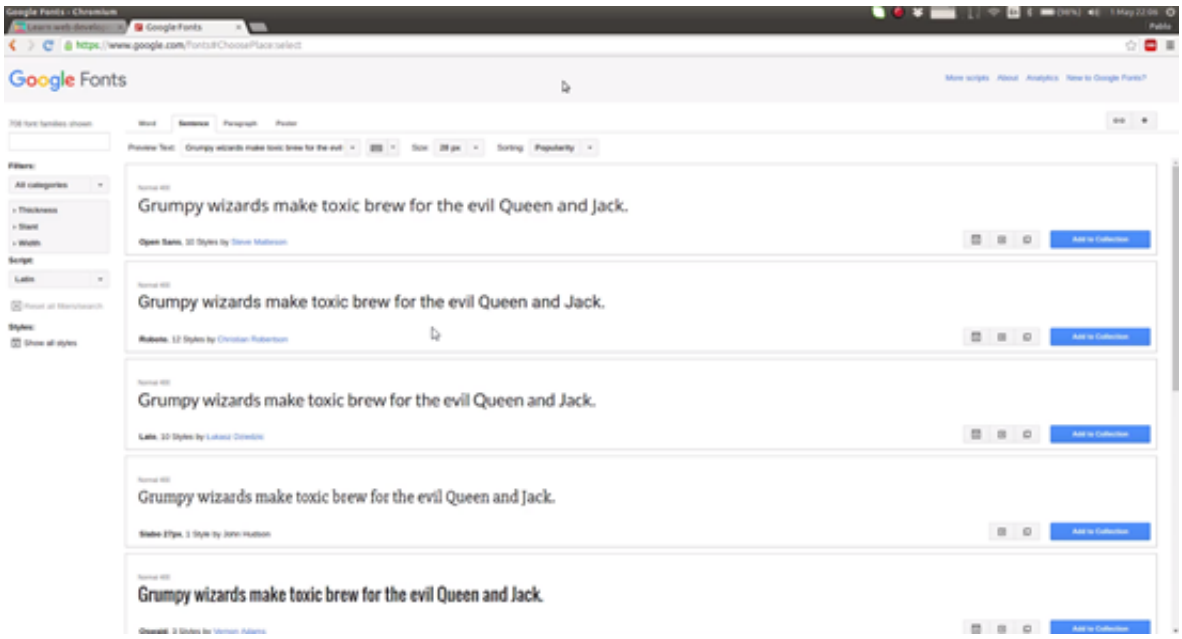
A great tool to use to find some fonts you can **use for free** and get downloaded on the visitors browser is by using **Google Fonts**.

You can visit the site here: https://fonts.google.com/

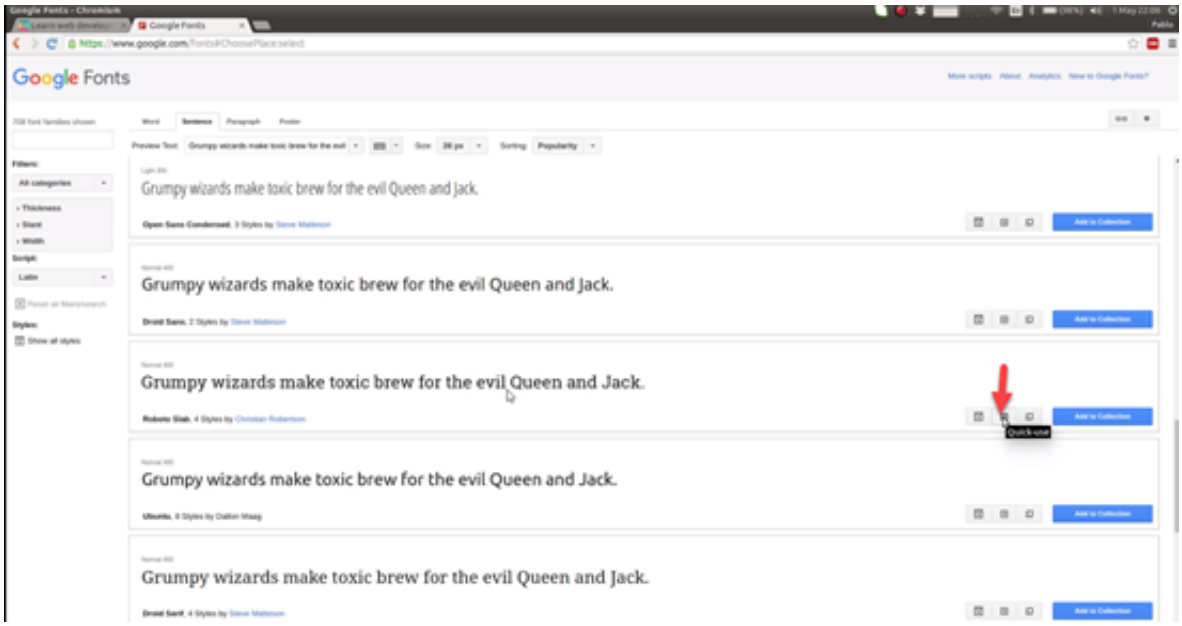
The direct link is here: Google Fonts

Google Fonts

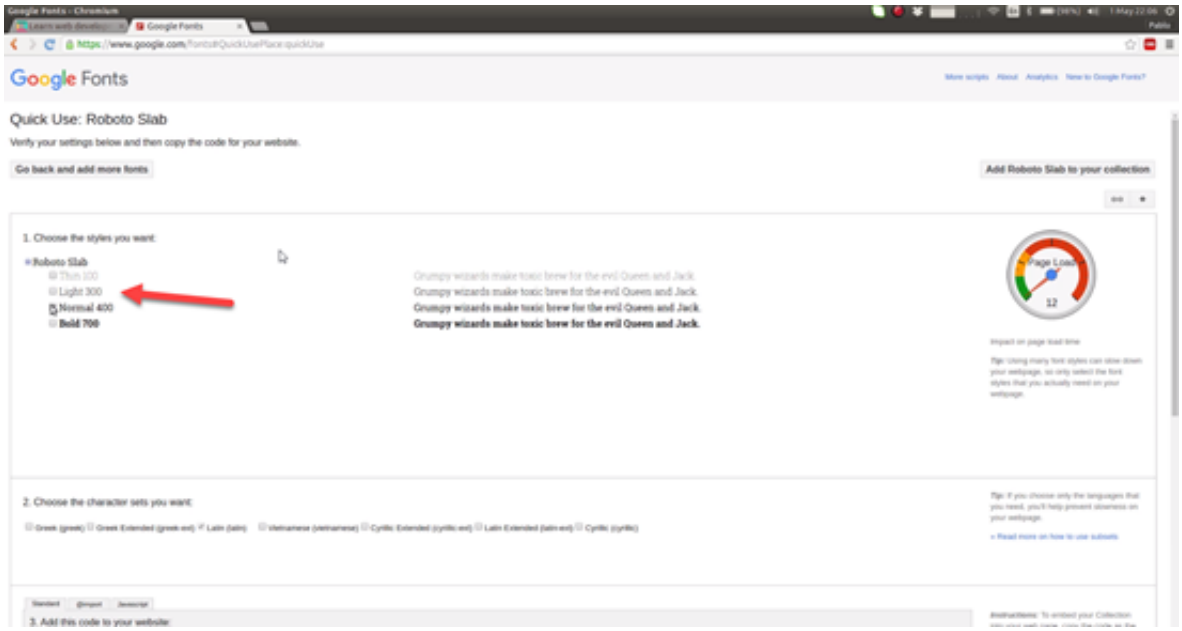
So if you go ahead an visit the site, you will find a very large collection of different fonts.



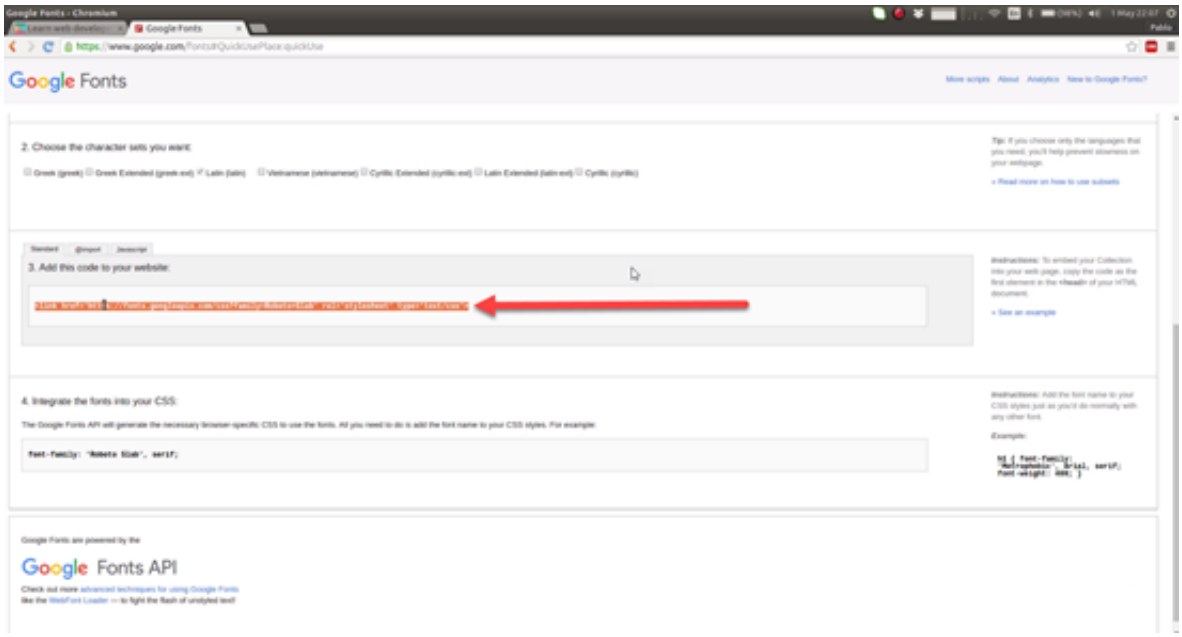
If you want to use one of these fonts for your page, then you would just select the **Quick Use** option.



It will give you some options for the font on the left side.



You can grab the code here in this section:



Select it and then **copy** it.

Then you will open your HTML and CSS code up, you will have to include what you copied from Google Fonts before your CSS file:

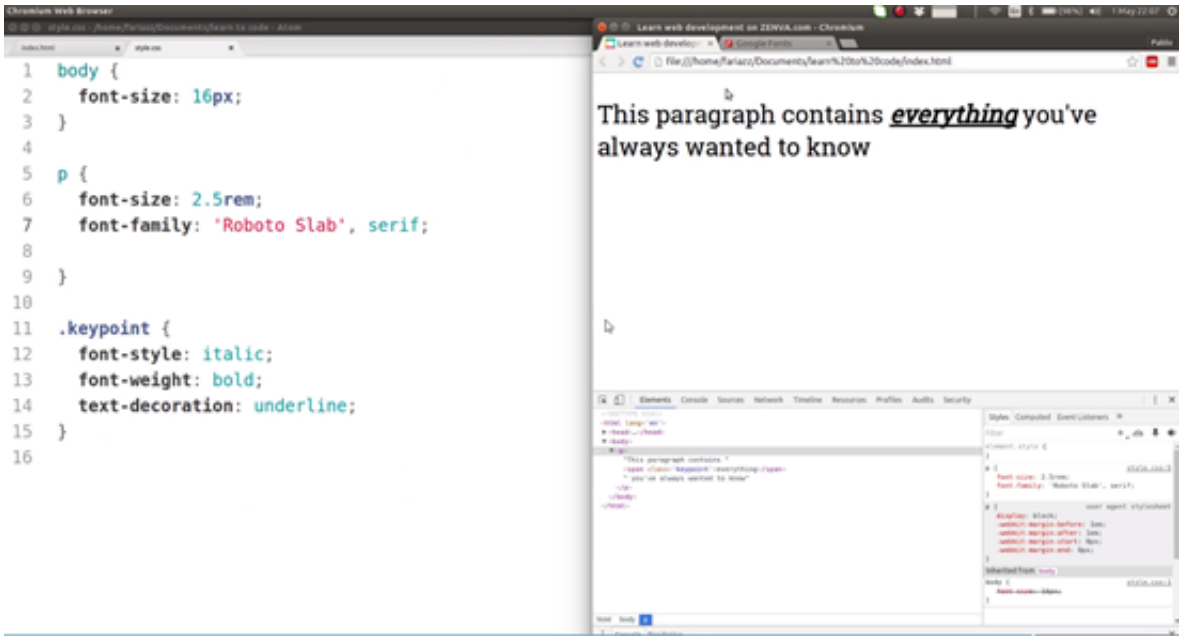
```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Learn web development on ZENVA.com</title>
    <meta charset="UTF-8" >
    <meta name="description" content="This is what Google will show!" >
    <link rel="icon" href="favicon.ico" >
    <link href='https://fonts.googleapis.com/css?family=Roboto+Slab' rel='stylesheet'
 type='text/css'>
    <link rel="stylesheet" href="style.css" >
  </head>
  <body>
    <p>This paragraph contains <span class="keypoint">everything</span> you've always
 wanted to know</p>
  </body>
</html>
```

```
body {
  font-size: 16px;
}

p {
  font-size: 2.5rem;
  font-family: 'Roboto Slab', serif;

}

.keypoint {
```



```
  font-style: italic;
  font-weight: bold;
  text-decoration: underline;
}
```

Save this and **refresh** the page.

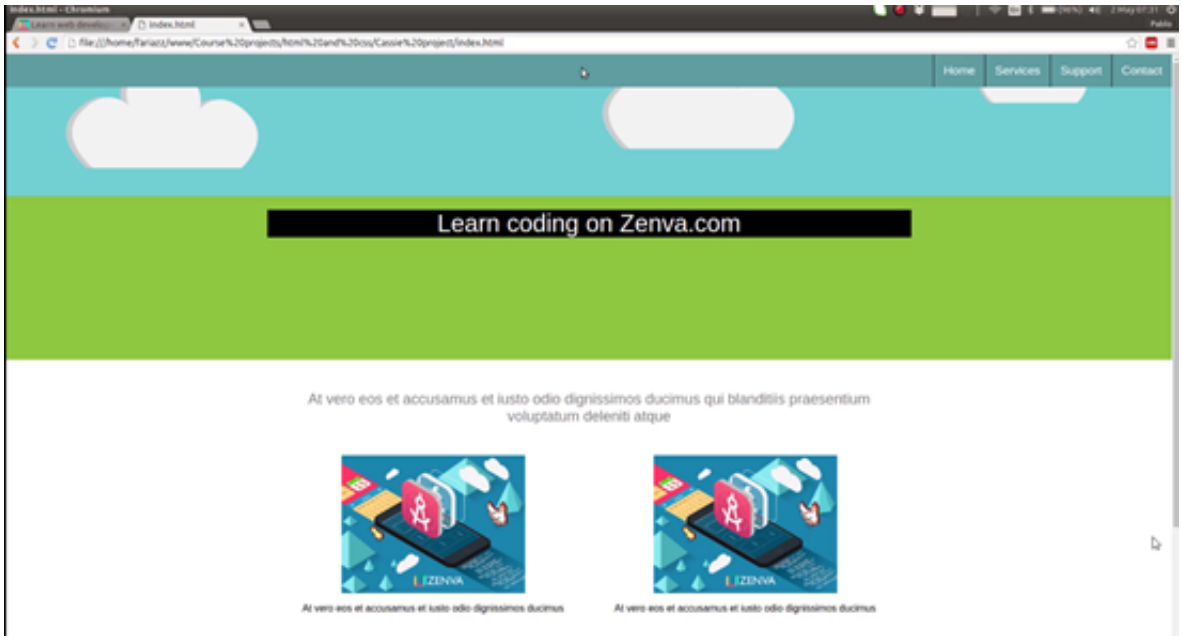


So we are now displaying the text on the page with the new font from the Google Fonts web page we copied.

Google Fonts is a great resource to find fonts to use for your web pages you will be creating. They will give your pages an original look.

In this lesson we will begin working on our **landing page project**.

Here is an example of a landing page:



There is something special about this landing page, and its special because it **responsive**.

Responsive means that the content can **adapt** to different screen sizes.

The best way to understand responsive design is to think of content as if it were a liquid, so that content needs to be adjusting to different screen sizes, whether its a smart TV, a desktop computer, a tablet, or even a phone.

What we are going to do in this lesson in regards to coding is to work on the main elements on the **HTML** side of things.

The approach that we will be taking is a mobile-first approach, and this means that you actually create the mobile page first and then later on using something called **media queries** we will be able to create the desktop version just by adding some extra **CSS** with a few more rules.

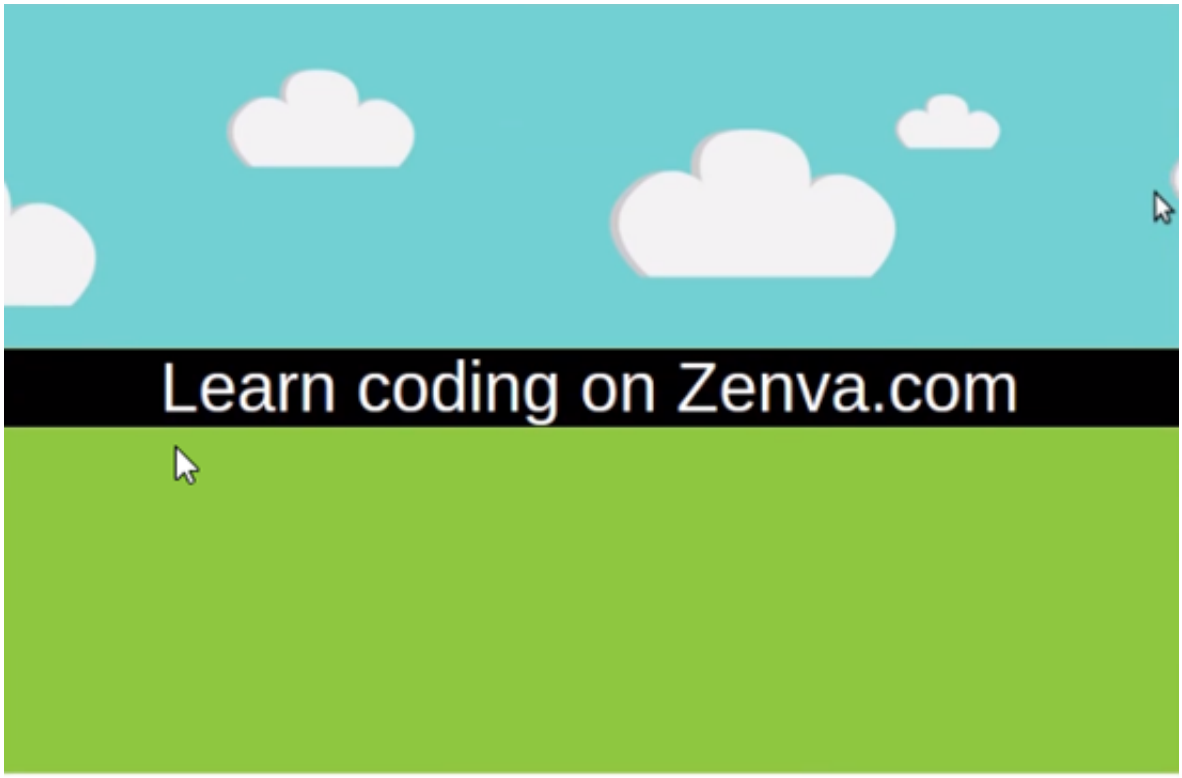
Both the mobile and desktop versions will share the **same CSS code**. There are some conditions that will only apply to certain screen sizes.

The Landing Page

This very first area in the screen shot below will be its own **div container**.

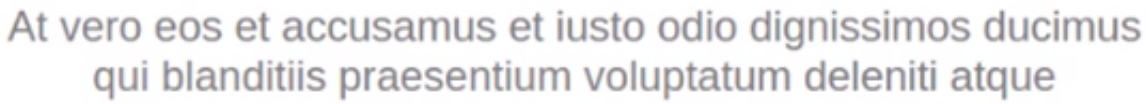
Then we are going to have a **div containe**r which we will call **wrapper** for everything else.

The **image area** will also be a **div container**:



This div container has a background image, and inside that container we have **another container** that we will use for the **title** “Learn coding on Zenva.com.”

We will have a **container** for the **intro text** below:



Then there will be **containers** for each one of the **service elements** below:

At vero eos et accusamus et iusto odio dignissimos ducimus

At vero eos et accusamus et iusto odio dignissimos ducimus

We will now get started by adding some basic structure that we can later add some style to in further lessons.

Starting the HTML code

Start with a **blank HTML page**, see the HTML code below:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Learn web development on ZENVA.com</title>
    <meta charset="UTF-8" >
    <meta name="description" content="This is what Google will show!" >
    <link rel="icon" href="favicon.ico" >
    <link rel="stylesheet" href="style.css" >
  </head>
  <body>
  </body>
</html>
```

The **CSS page** is included, but its completely **empty**.

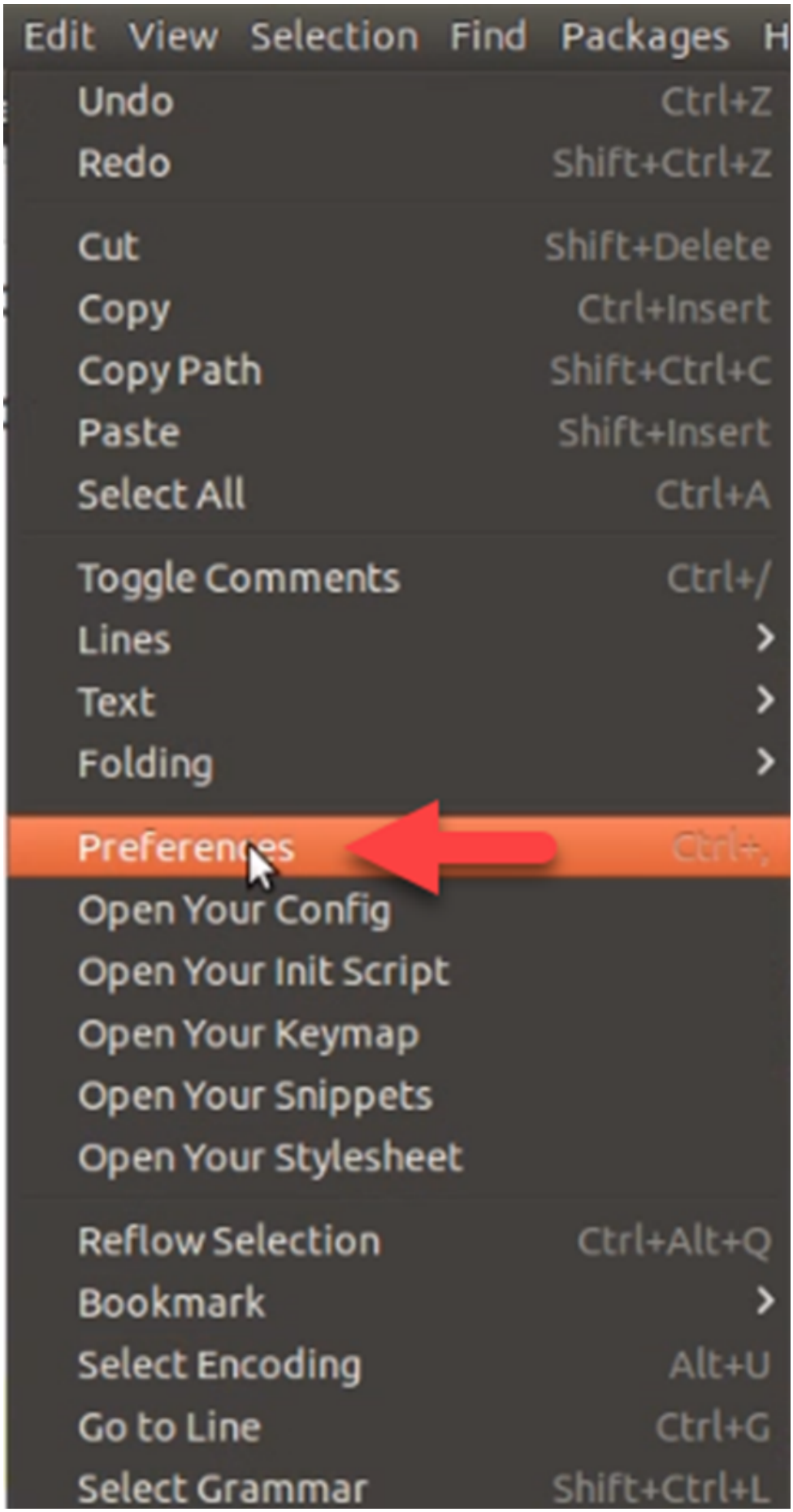
Remember, we will only be doing the HTML aspects in this lesson.

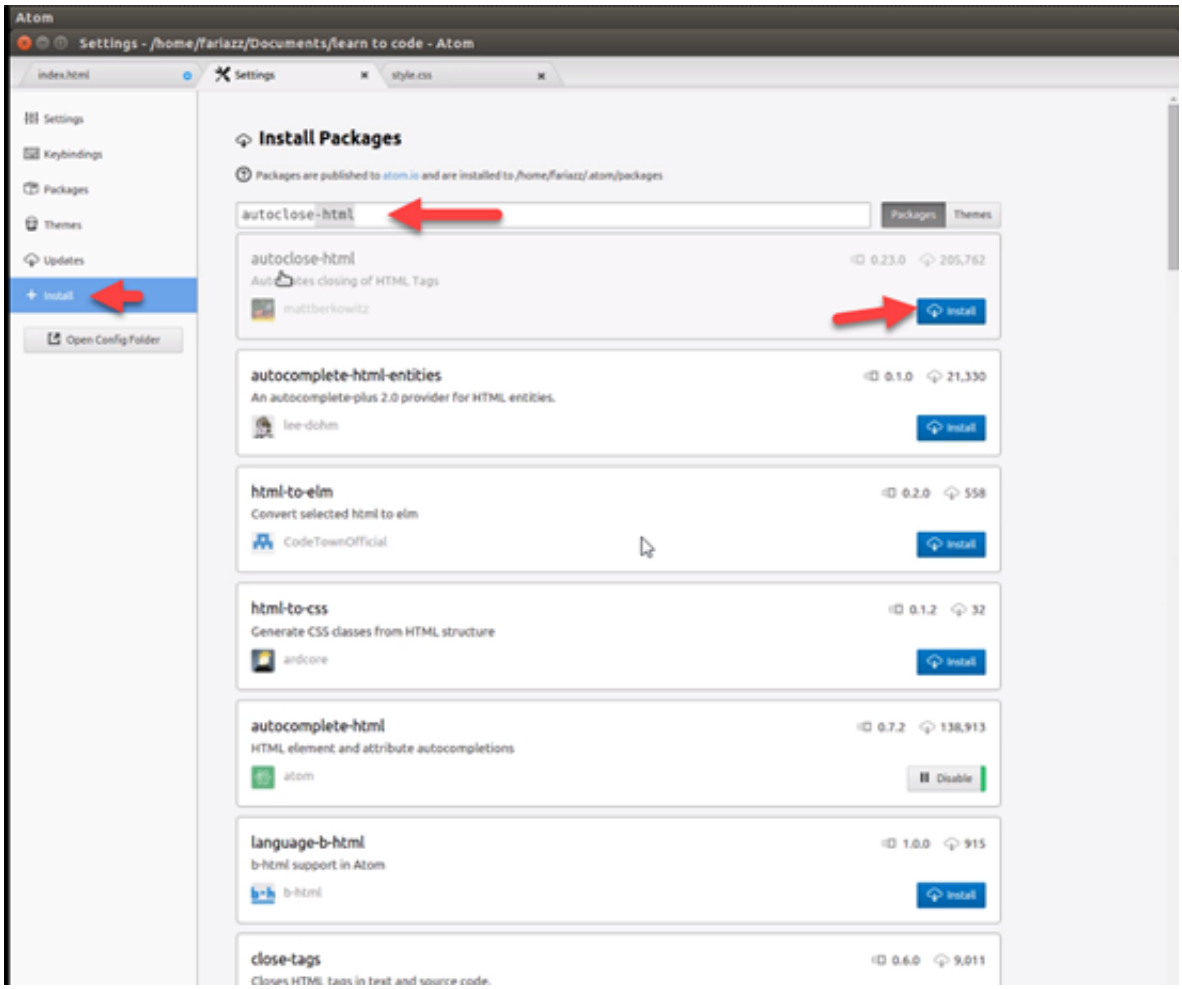
So the very first part is a **div container**, and we will give **div** the **class “nav”**, so that we can style it later.

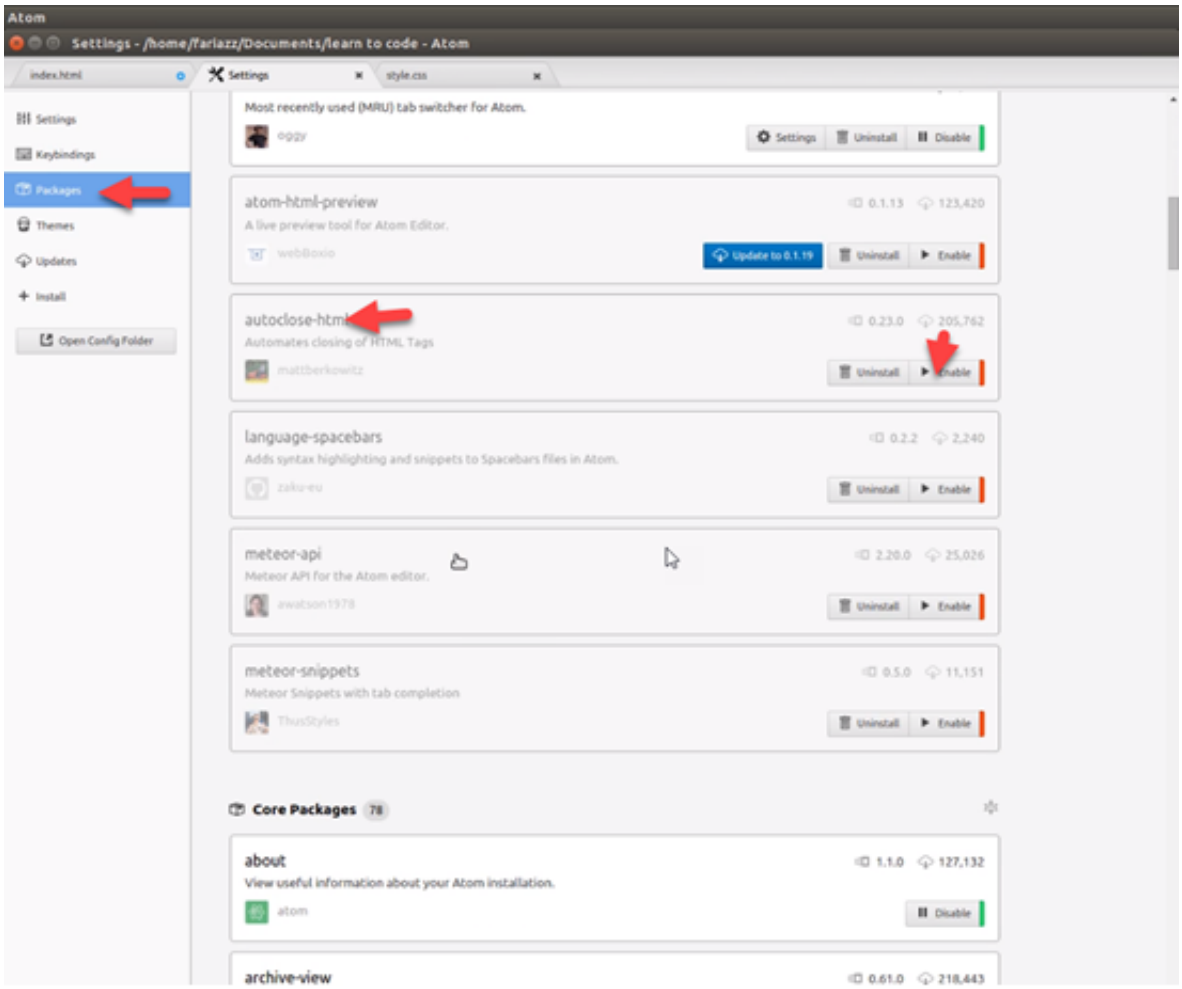
Because this part of the HTML code will require a lot of typing, I am activating an **extension** in the **atom editor** that will allow me to **auto close** the **tags**.

So navigate to **Edit>Preferences>Install**, and then search for **autoclose-html**.

Then click **Install,** and **activate** it.







Enabling this extension will make your typing for this part of the lesson much easier.

See the HTML code below:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Learn web development on ZENVA.com</title>
    <meta charset="UTF-8" >
    <meta name="description" content="This is what Google will show!" >
    <link rel="icon" href="favicon.ico" >
    <link rel="stylesheet" href="style.css" >
  </head>
  <body>
    <div class="nav">
      <ul>
        <li>Home</li>
        <li>Service</li>
        <li>Support</li>
        <li>Contact</li>
      </ul>
    </div>
    <div class="wrapper">
      <div class="image-area">
        <div class="image-text">Learn coding on Zenva.com
```



```
</div>
      </div>
      <div class="intro-text">At vero eos et accusamus et iusto odio dignissimos duci
mus qui blanditiis praesentium voluptatum deleniti atque
</div>
      <div class="services">
        <div class="single-service">
          <img src="images/zenva-banner.png" />
          <p>
            At vero eos et accusamus et iusto odio dignissimos ducimus
          </p>
        </div>
        <div class="single-service">
          <img src="images/zenva-banner.png" />
          <p>
            At vero eos et accusamus et iusto odio dignissimos ducimus
          </p>
        </div>
        <div class="single-service">
          <img src="images/zenva-banner.png" />
          <p>
            At vero eos et accusamus et iusto odio dignissimos ducimus
          </p>
        </div>
        <div class="single-service">
          <img src="images/zenva-banner.png" />
          <p>
            At vero eos et accusamus et iusto odio dignissimos ducimus
          </p>
        </div>
      </div>
    </div>

  </body>
</html>
```

In the next lesson we will start working on the actual styling of what we just created.

Save this and proceed to the next lesson.

In this lesson we will begin **styling** our **landing page** and we will concentrate on the **top part** of the **menu**.

We need to start by **setting** up some **defaults** on the **body** of our **page**.

Web browsers have some defaults that they assume that you want, but in the hypothetical case that the web browser doesn't have those same defaults, you will want to be more specific.

For example, web browsers give you a white background by default and they give you black font by default. We never said we wanted our font to be black, but it's what we are getting by default.

So, we are going to begin styling the page by selecting the body of our page and setting up some defaults so that we actually have made this explicit and if it wasn't the default of the browser it would show the way we want.

Styling the Landing Page

Open the **CSS** page and begin adding the CSS to it for the body:

```
body {
  background-color: white;
  color: black;
  font-size: 16px;
  font-family: Arial, Helvetica, sans-serif;
  padding: 0px;
  margin: 0px;
}
```

Now we will concentrate on the **nav area**. Which is the **div container** that contains the Home, Service, Support and Contact.

Open the **CSS** code back up and add the CSS:

```
body {
  background-color: white;
  color: black;
  font-size: 16px;
  font-family: Arial, Helvetica, sans-serif;
  padding: 0px;
  margin: 0px;
}

.nav {
  background-color: cadetblue;
  color: white;
}
```

Save this and **refresh** the page.

You will see that we are getting closer:

Next, lets concentrate on the UL element.

We will be **selecting** the **UL element** by entering the **parent element**, in this case **nav**, and then a **sub element**.

See the CSS code below:

```
body {
  background-color: white;
  color: black;
  font-size: 16px;
  font-family: Arial, Helvetica, sans-serif;
  padding: 0px;
  margin: 0px;
}

.nav {
  background-color: cadetblue;
  color: white;
}

.nav ul {
  margin: 0px;
  padding: 0px;
  list-style-type: none;
}
```

Save this and **refresh** the page.

```
  background-color: white;
  color: black;
  font-size: 16px;
  font-family: Arial, Helvetica, sans-serif;
  padding: 0px;
  margin: 0px;
}

.nav {
  background-color: cadetblue;
  color: white;
}

.nav ul {
  margin: 0px;
  padding: 0px;
  list-style-type: none;
}
```

We are on the right track so far.

Next, we need to style the actual individual items.

We can follow the same approach we used above to select those elements.

See the CSS code below:

```
body {
  background-color: white;
  color: black;
  font-size: 16px;
  font-family: Arial, Helvetica, sans-serif;
  padding: 0px;
  margin: 0px;
}

.nav {
  background-color: cadetblue;
  color: white;
}

.nav ul {
  margin: 0px;
  padding: 0px;
  list-style-type: none;
}

.nav ul li {
  padding: 16px;
  text-align: center;
  font-size: 1.2rem;
  border-bottom: 1px solid black;
  transition: background-color 0.2s;
}

.nav ul li:hover {
  background-color: black;
```

```
}
```

Save this and **refresh** page.

We now have the top menu working properly.

Summary

We specified some default properties for the body of our page, so this applies to the entire page. Some of these things are obvious like the background color white, but what if the browser you are using doesn't show white by default? You may want to set that and make it explicit. This goes for the font and the font-family.

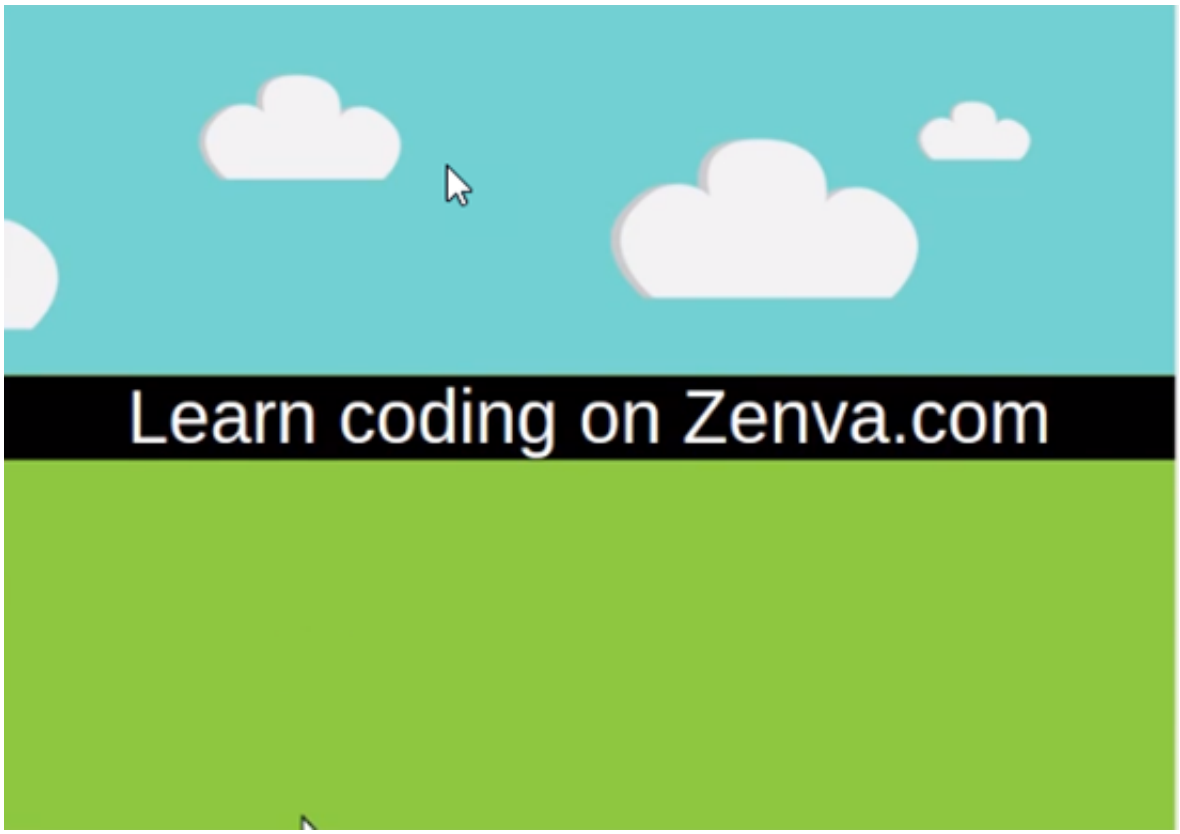
The navigation area has a background color and a color for the font. Then we selected the UL element and gotten rid of all the default styles.

We have the individual list items and gave them properties for padding, alignment, and size.

Lastly, we looked at the hover effect by selecting the hover state and giving it a different background color. We then made the transition changes smooth, but not too abrupt.

In this lesson, we will work on the **cover image**.

So where it says learn coding on Zenva.com will look like this when you finish this lesson:



We will get started by taking care of the **image-area** and then look into taking care of the **image-text** next.

Open up the **HTML** and **CSS** code we have from the previous lesson.

Image-Area Styling

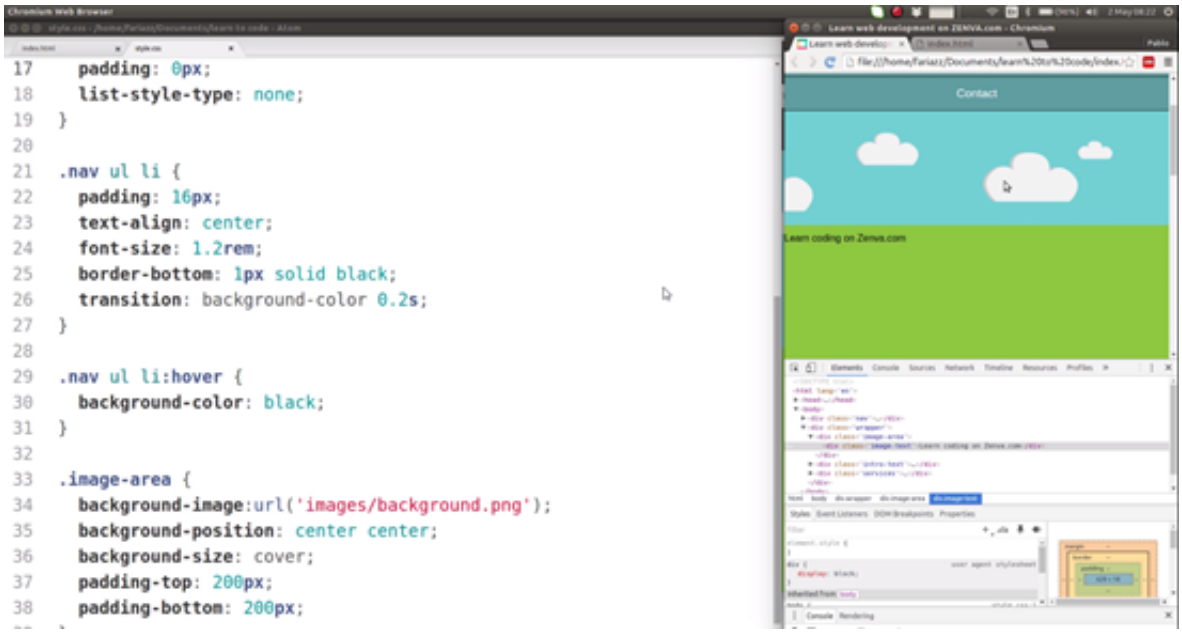
We will begin by **selecting** the **image-area** and there are going to be a few things we need to add here.

See the code below:

```
body {
  background-color: white;
  color: black;
  font-size: 16px;
  font-family: Arial, Helvetica, sans-serif;
  padding: 0px;
  margin: 0px;
}

.nav {
  background-color: cadetblue;
  color: white;
}
```

```
.nav ul {
  margin: 0px;
  padding: 0px;
  list-style-type: none;
}

.nav ul li {
  padding: 16px;
  text-align: center;
  font-size: 1.2rem;
  border-bottom: 1px solid black;
  transition: background-color 0.2s;
}

.nav ul li:hover {
  background-color: black;
}

.image-area {
  background-image:url('images/background.png');
  background-color:black;
  background-position: center center;
  background-size: cover;
  padding-top: 200px;
  padding-bottom: 200px;
}
```

Save this and **refresh** the page.



This is everything for the image itself.

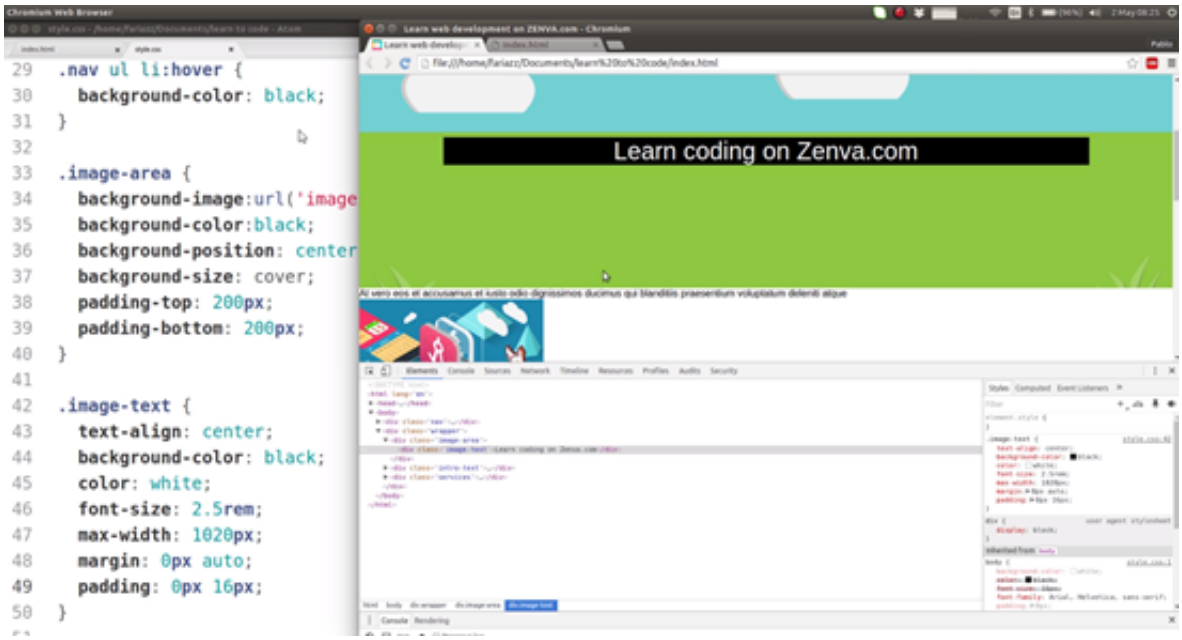
Image-Text Styling

Now we can start **adding** the **styling** specifications for the **image-text**.

See the CSS code below:

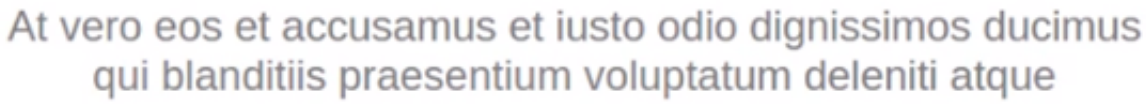
```
body {
  background-color: white;
  color: black;
  font-size: 16px;
  font-family: Arial, Helvetica, sans-serif;
  padding: 0px;
  margin: 0px;
}

.nav {
  background-color: cadetblue;
  color: white;
}

.nav ul {
  margin: 0px;
  padding: 0px;
  list-style-type: none;
}

.nav ul li {
  padding: 16px;
  text-align: center;
  font-size: 1.2rem;
  border-bottom: 1px solid black;
  transition: background-color 0.2s;
}

.nav ul li:hover {
  background-color: black;
}

.image-area {
  background-image:url('images/background.png');
  background-color:black;
  background-position: center center;
  background-size: cover;
  padding-top: 200px;
  padding-bottom: 200px;
}

.image-text {
  text-align: center;
  background-color: black;
  color: white;
  font-size: 2.5rem;
  max-width: 1020px;
  margin: 0px auto;
  padding: 0px 16px;
}
```

Save this and **refresh** the page.



In the next lesson we will start working on the service elements that are below the background image.

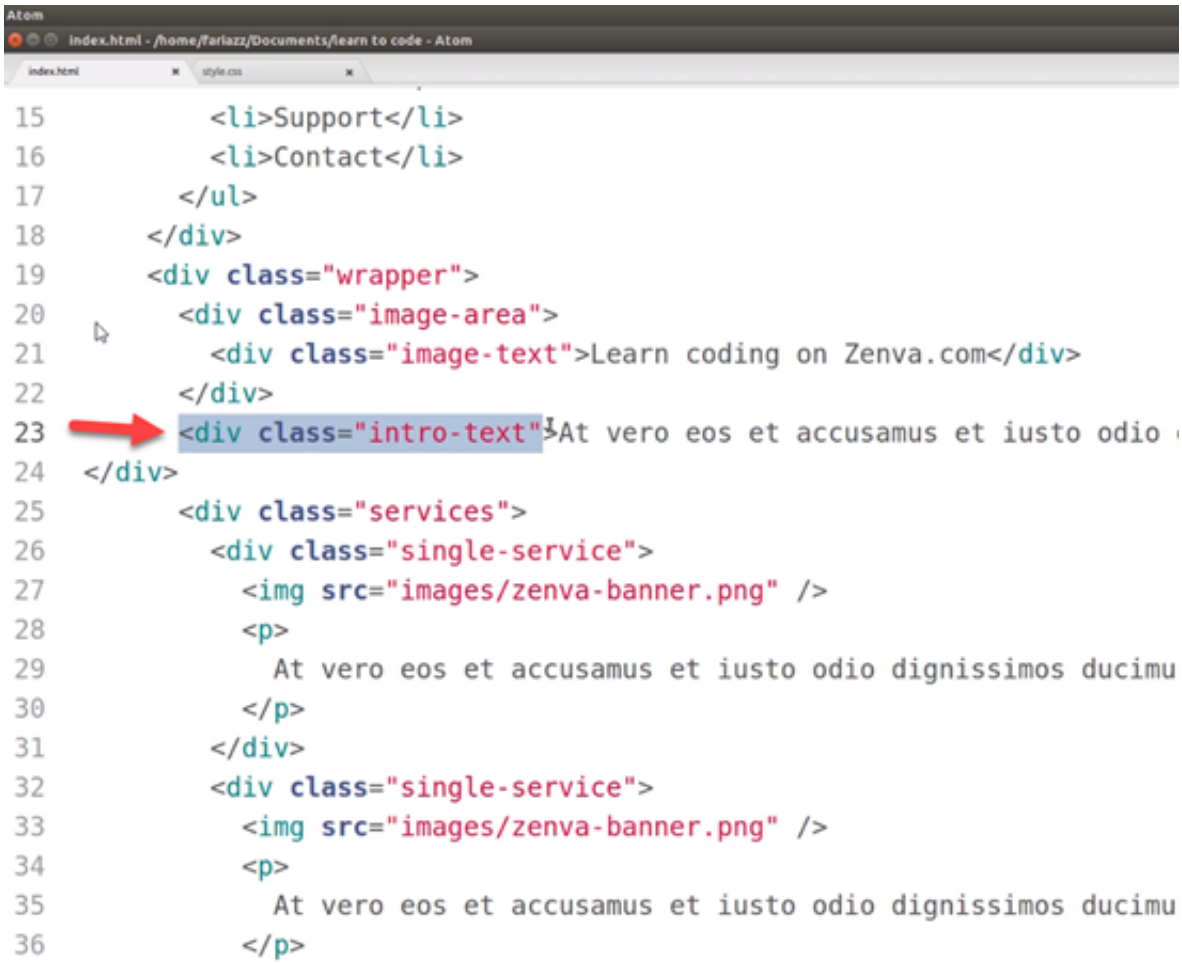
In this lesson, we will work on styling what is remaining, which is the **introduction text** and the **Services area**.



At vero eos et accusamus et iusto odio dignissimos ducimus

At vero eos et accusamus et iusto odio dignissimos ducimus

Let’s get started by looking at the **intro-text**. You can see that the **intro-text has a class intro-text**.



We will be **selecting** that and **start** with our **styling**.

Open up the **HTML** and **CSS** code from the previous lesson.

Intro-Text Styling and Services Area

See the CSS code below:

```
body {
  background-color: white;
  color: black;
  font-size: 16px;
  font-family: Arial, Helvetica, sans-serif;
  padding: 0px;
  margin: 0px;
}

.nav {
  background-color: cadetblue;
  color: white;
}

.nav ul {
  margin: 0px;
  padding: 0px;
```

```
  list-style-type: none;
}

.nav ul li {
  padding: 16px;
  text-align: center;
  font-size: 1.2rem;
  border-bottom: 1px solid black;
  transition: background-color 0.2s;
}

.nav ul li:hover {
  background-color: black;
}

.image-area {
  background-image:url('images/background.png');
  background-color:black;
  background-position: center center;
  background-size: cover;
  padding-top: 200px;
  padding-bottom: 200px;
}

.centered-container {
  max-width: 1020px;
  margin: 0px auto;
  padding: 0px 16px;
}

.image-text {
  text-align: center;
  background-color: black;
  color: white;
  font-size: 2.5rem;
}

.intro-text {
  color: gray;
  font-size: 1.5rem;
  text-align: center;
  padding-top: 50px;
  padding-bottom: 50px;
}

.single-service {
  text-align: center;
  margin-bottom: 32px;
}

.single-service p {
  padding: 0px 16px;
}
```


See the HTML code below:

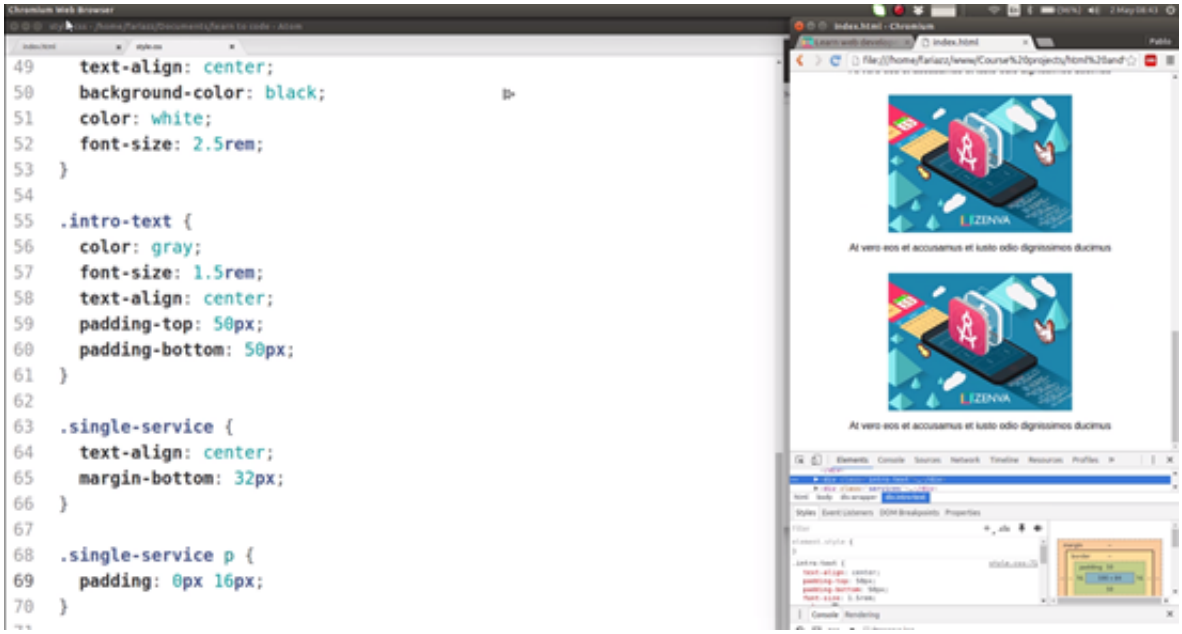
```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Learn web development on ZENVA.com</title>
    <meta charset="UTF-8" >
    <meta name="description" content="This is what Google will show!" >
    <link rel="icon" href="favicon.ico" >
    <link rel="stylesheet" href="style.css" >
  </head>
  <body>
    <div class="nav">
      <ul>
        <li>Home</li>
        <li>Services</li>
        <li>Support</li>
        <li>Contact</li>
      </ul>
    </div>
    <div class="wrapper">
      <div class="image-area">
        <div class="image-text centered-container">Learn coding on Zenva.com</div>
      </div>
      <div class="intro-text centered-container">At vero eos et accusamus et iusto od
io dignissimos ducimus qui blanditiis praesentium voluptatum deleniti atque
</div>
      <div class="services centered-container">
        <div class="single-service">
          <img src="images/zenva-banner.png" />
          <p>
            At vero eos et accusamus et iusto odio dignissimos ducimus
          </p>
        </div>
        <div class="single-service">
          <img src="images/zenva-banner.png" />
          <p>
            At vero eos et accusamus et iusto odio dignissimos ducimus
          </p>
        </div>
        <div class="single-service">
          <img src="images/zenva-banner.png" />
          <p>
            At vero eos et accusamus et iusto odio dignissimos ducimus
          </p>
        </div>
        <div class="single-service">
          <img src="images/zenva-banner.png" />
          <p>
            At vero eos et accusamus et iusto odio dignissimos ducimus
          </p>
        </div>
      </div>
    </div>
```



```
  </body>
</html>
```

Save this and **refresh** the page.

Alright so we have the bottom area of the landing-page done.



In the next lesson we will work on making the bottom area **responsive**.

This is all for the mobile version, we have the mobile version now ready for us and this could be enough for just a mobile page.

We want it look a bit nicer on the desktop computer as well, so we will finish up the responsive design in the next lesson.

As a web developer you will have no idea what device people will be using to view your created web pages.

There is no telling what people will use to access your web content.

As a **web developer/designer** you need to **create** your **web pages** thinking of **screen sizes**, not devices.

So the approach you should take is the approach as if the content were liquid that you could literally pour on any sort of recipient and it should do the job as in people can read your content. they can access different options of the page.

This is the approach we will be taking in this lesson with our landing page project.

So far our project looks great on a small screen. We have taken a **mobile-first approach**, which is the smallest screen.

Once you have that established it's easier to adapt to a bigger screen.

The way that we will be **implementing** this is by setting an **arbitrary limit** saying, that screens that are more than **700 pixels** will have the bigger version of the site.

Keep in mind that we are going to be using the same **CSS** code in both cases. It is not a different page for mobile or a different page for desktop.

So what will happen is once the screen **reaches over** the **700 pixels** we will call a **group of CSS rules** that will change some things on the page.

We can do this by using something called **"media queries."** Let's now create the first **media query.**

Open up the **HTML** and **CSS** code from the previous lesson.

See the CSS code below:

```
body {
  background-color: white;
  color: black;
  font-size: 16px;
```



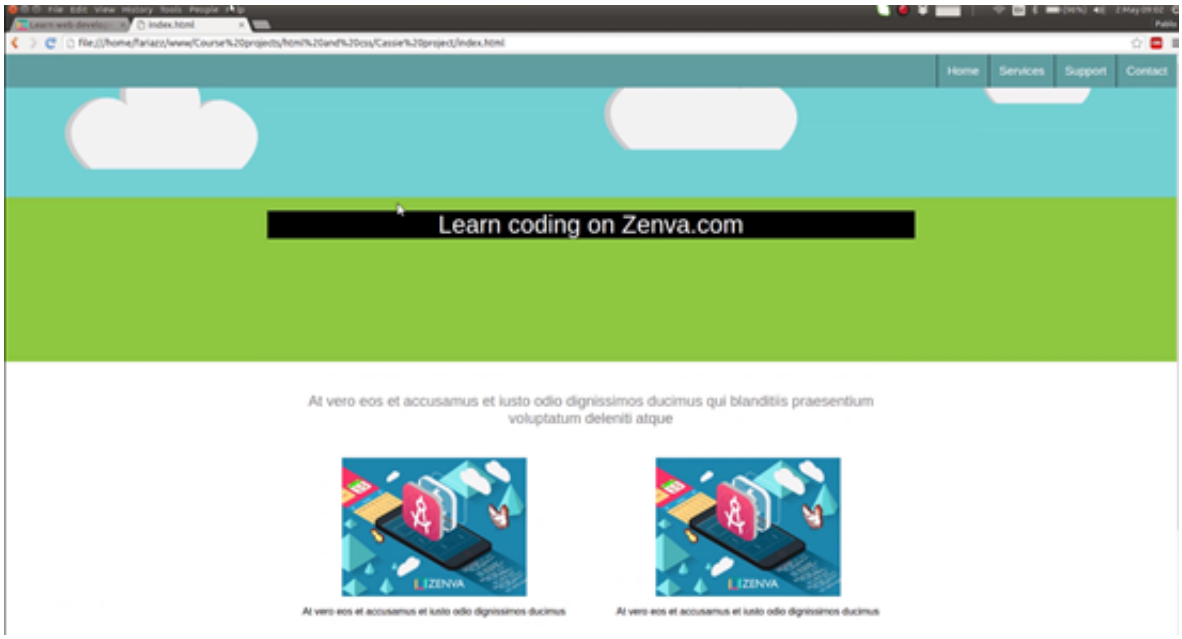
```
  font-family: Arial, Helvetica, sans-serif;
  padding: 0px;
  margin: 0px;
}

.nav {
  background-color: cadetblue;
  color: white;
}

.nav ul {
  margin: 0px;
  padding: 0px;
  list-style-type: none;
  font-size: 0px;
}

.nav ul li {
  padding: 16px;
  text-align: center;
  font-size: 1.2rem;
  border-bottom: 1px solid black;
  transition: background-color 0.2s;
}

.nav ul li:hover {
  background-color: black;
}

.image-area {
  background-image:url('images/background.png');
  background-color:black;
  background-position: center center;
  background-size: cover;
  padding-top: 200px;
  padding-bottom: 200px;
}

.centered-container {
  max-width: 1020px;
  margin: 0px auto;
  padding: 0px 16px;
}

.image-text {
  text-align: center;
  background-color: black;
  color: white;
  font-size: 2.5rem;
}

.intro-text {
  color: gray;
  font-size: 1.5rem;
  text-align: center;
```



```
  padding-top: 50px;
  padding-bottom: 50px;
}

.single-service {
  text-align: center;
  margin-bottom: 32px;
}

.single-service p {
  padding: 0px 16px;
}

@media screen and (min-width: 700px) {
  .nav ul {
    text-align: right;
  }

  .nav ul li {
    display: inline-block;
    border-bottom: 0px;
    border-left: 1px solid black;
    text-align: left;
  }

  .single-service {
    width: 50%;
    float: left;
  }
}
```

Save this and **refresh** the page.

As you can see we now have a fully responsive page:



The page will look great on different devices, and we have built this from absolute scratch. We started with **HTML** by building a skeleton that looked quite horrible at the beginning and step by step we've added our **CSS** rules to make this page a fully responsive landing page.

These are the basics and once you have reached this point, you have reached the basic level of CSS.

You should now be able to take some time and practice and build some new things, and play with all the rules that we have covered in this course.

You are now on the right track for your coding journey!

Thank you for taking this course.