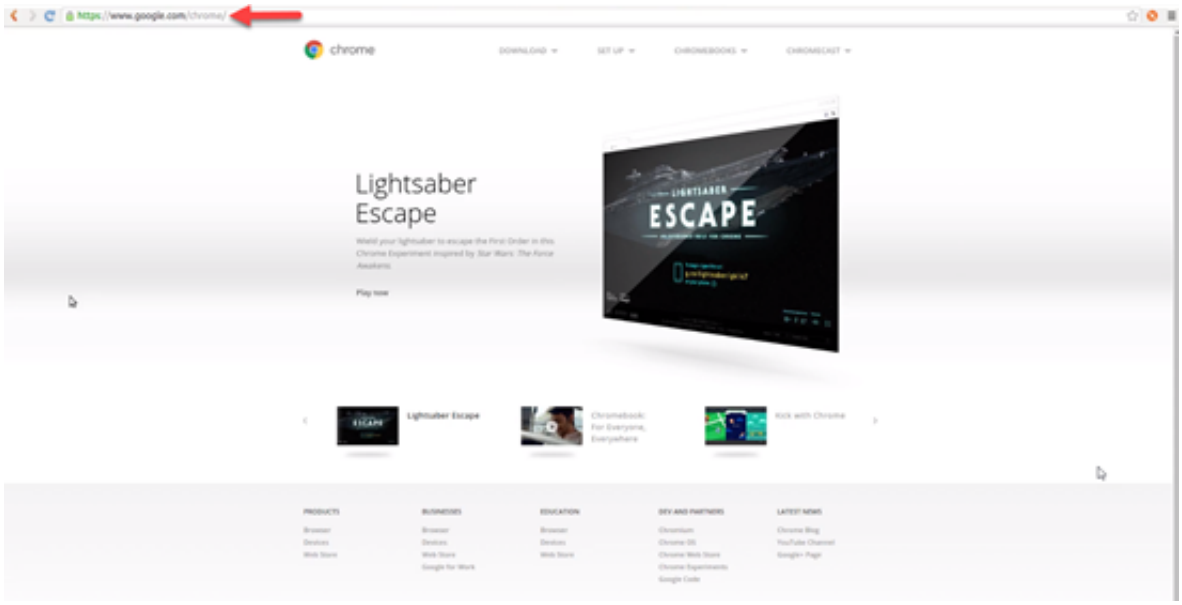
In this lesson we will be getting the development environment ready. We will be going step by step and will have a Hello World HTML project.

Prerequisite One

You will need a current **web browser**, I will be using **Google Chrome** throughout this course. Feel free to use **Firefox** or even **Edge** if you desire.



Prerequisite Two

You will need a proper **code editor** to complete this course.

You will not be able to use Microsoft Word, or the Notepad version that comes with Windows, you will **NEED** a proper **code editor**.

I will be using the code editor called **“Atom”** throughout this course, and it works on all platforms: Windows, Mac, and Linux. Atom is a free code editor and can be downloaded from the web.

The link to download Atom is here: https://atom.io/

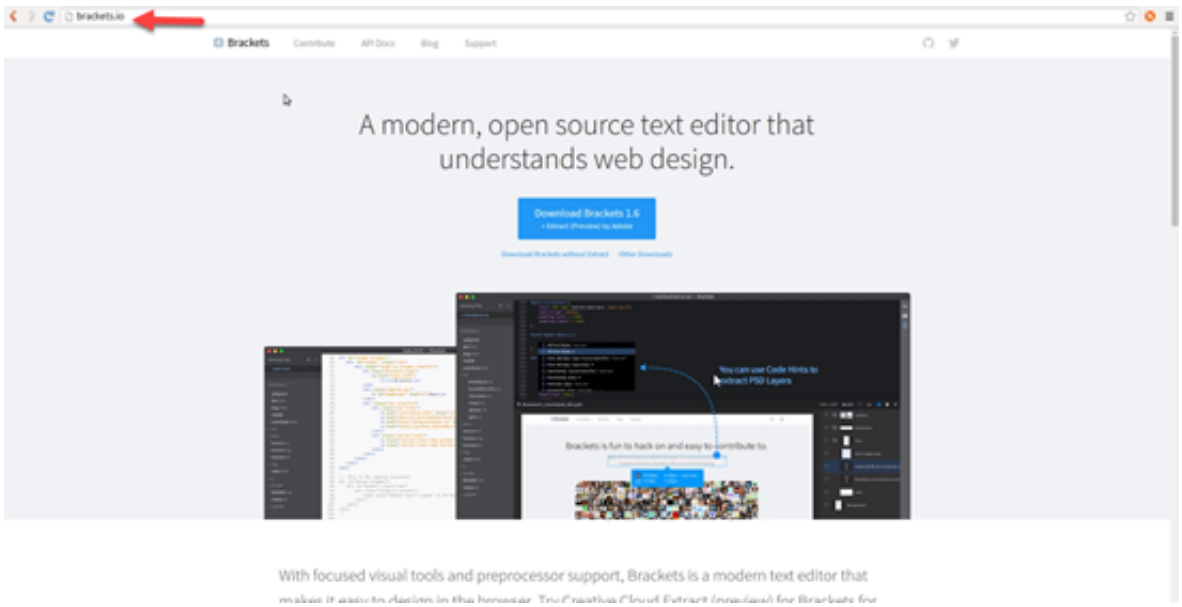
Atom IDE

There are other free versions out there, so feel free to use anyone of them that suits you the best, the other alternative code editors are listed below:

Another great option is **Adobe Brackets**.

The link to download Brackets is here: http://brackets.io/

[Brackets IDE]



Notepad++ is a particularly simple code editor.

The link to download Notepad ++ is here: https://notepad-plus-plus.org/download/v7.6.2.html

[Notepad++ IDE]

Komodo Edit.

The link to download Komodo Edit is here: https://www.activestate.com/products/komodo-ide/downloads/edit/

Komodo Edit IDE

Sublime Text is a highly popular code editor and has a free evaluation version as well.

The link to download Sublime Text is here: https://www.sublimetext.com/3

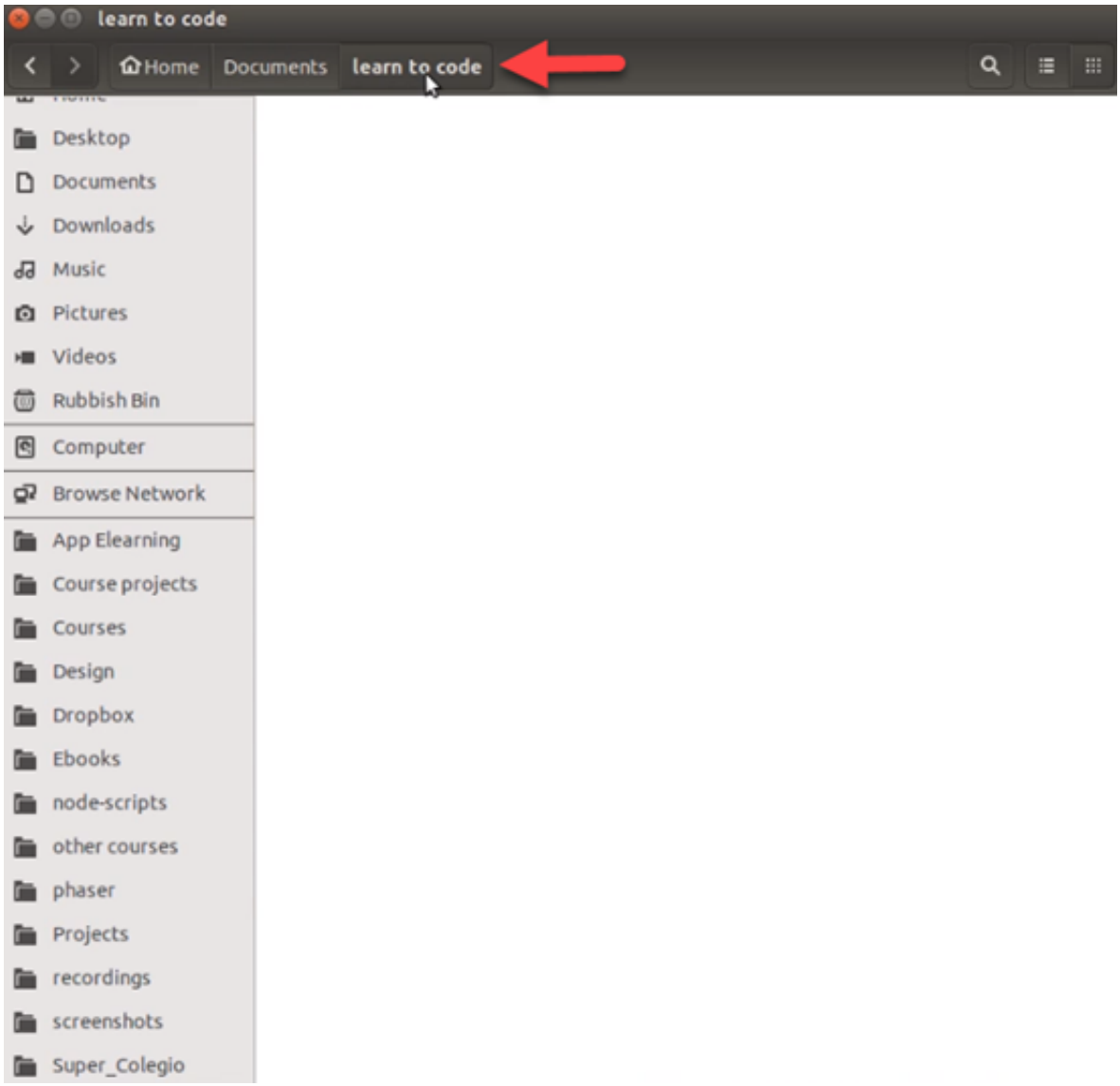
Sublime Text IDE

Creating the Project Folder

Once you have **downloaded** and **installed** your chosen code editor that suits you best, you will need to **create** a **project folder**.

I will be saving mine in My Documents, and I have already created a folder called **“Learn To Code.”**

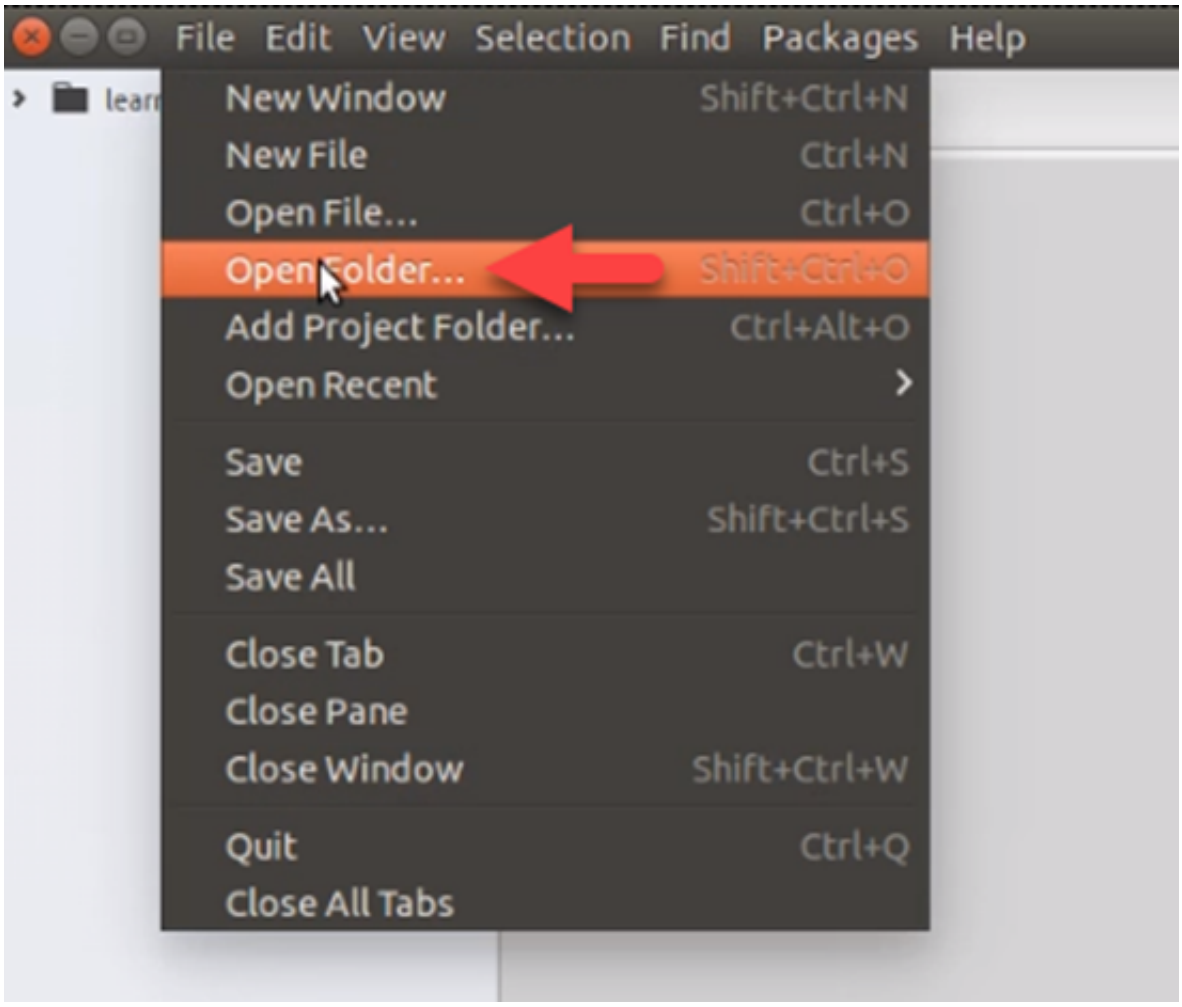
You don’t have to save your folder in My Documents, but be sure to put it somewhere where you won’t forget where its saved at.



Opening the File in the Code Editor

Next, **open** up the **code editor** you have installed.

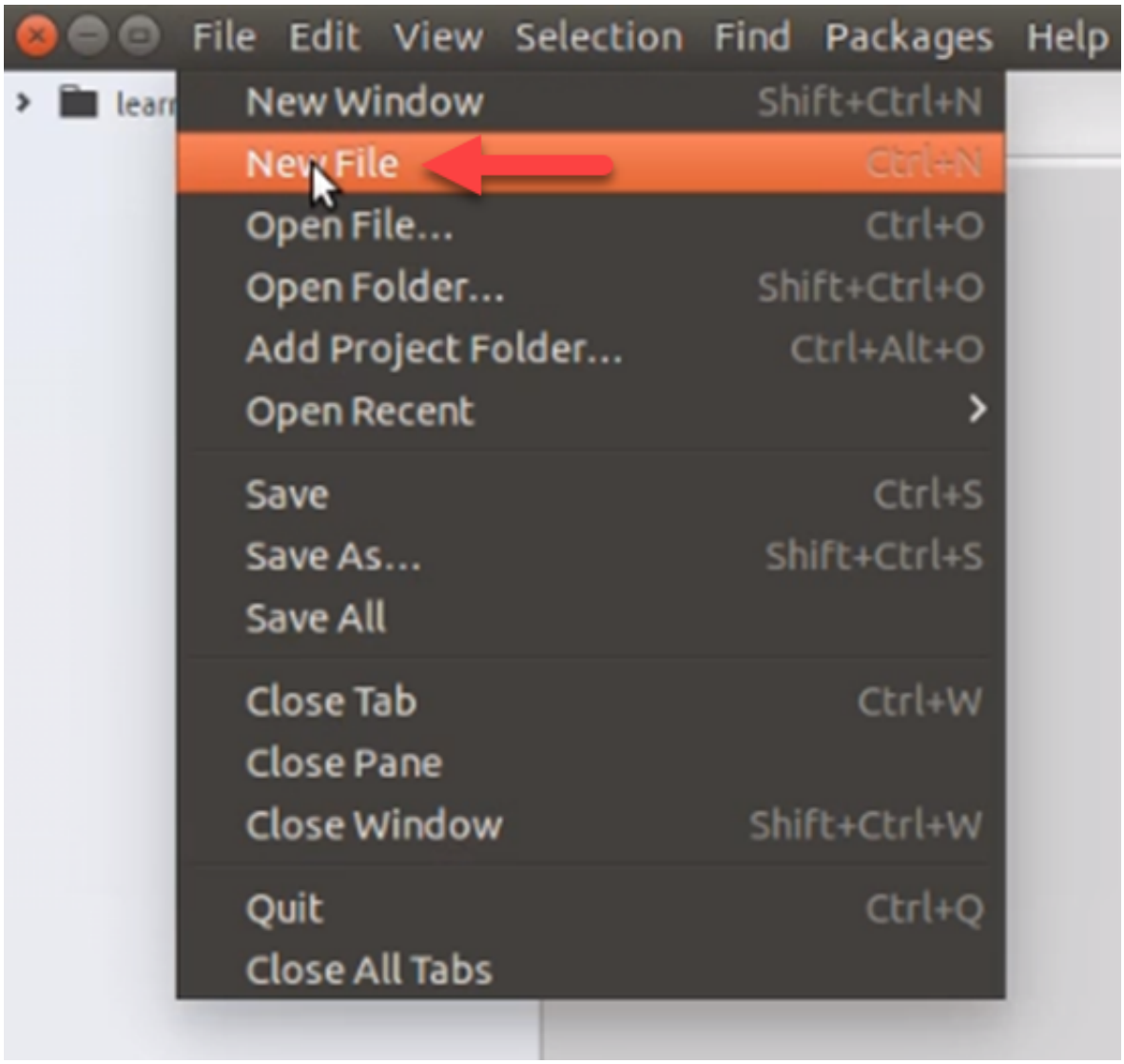
Navigate to **File>Open Folder**, and find the newly created folder you just created.



The Learn To Code folder is currently empty, but we will create out first HTML file.

Creating the First HTML File

Navigate to **File>New File**.

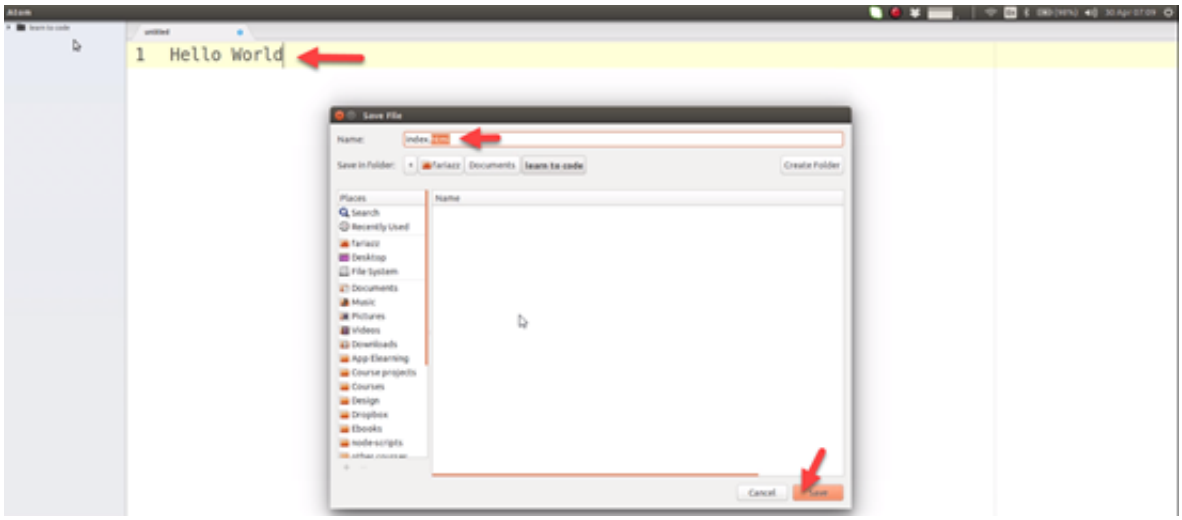


On line number one I will type **“Hello World.”**

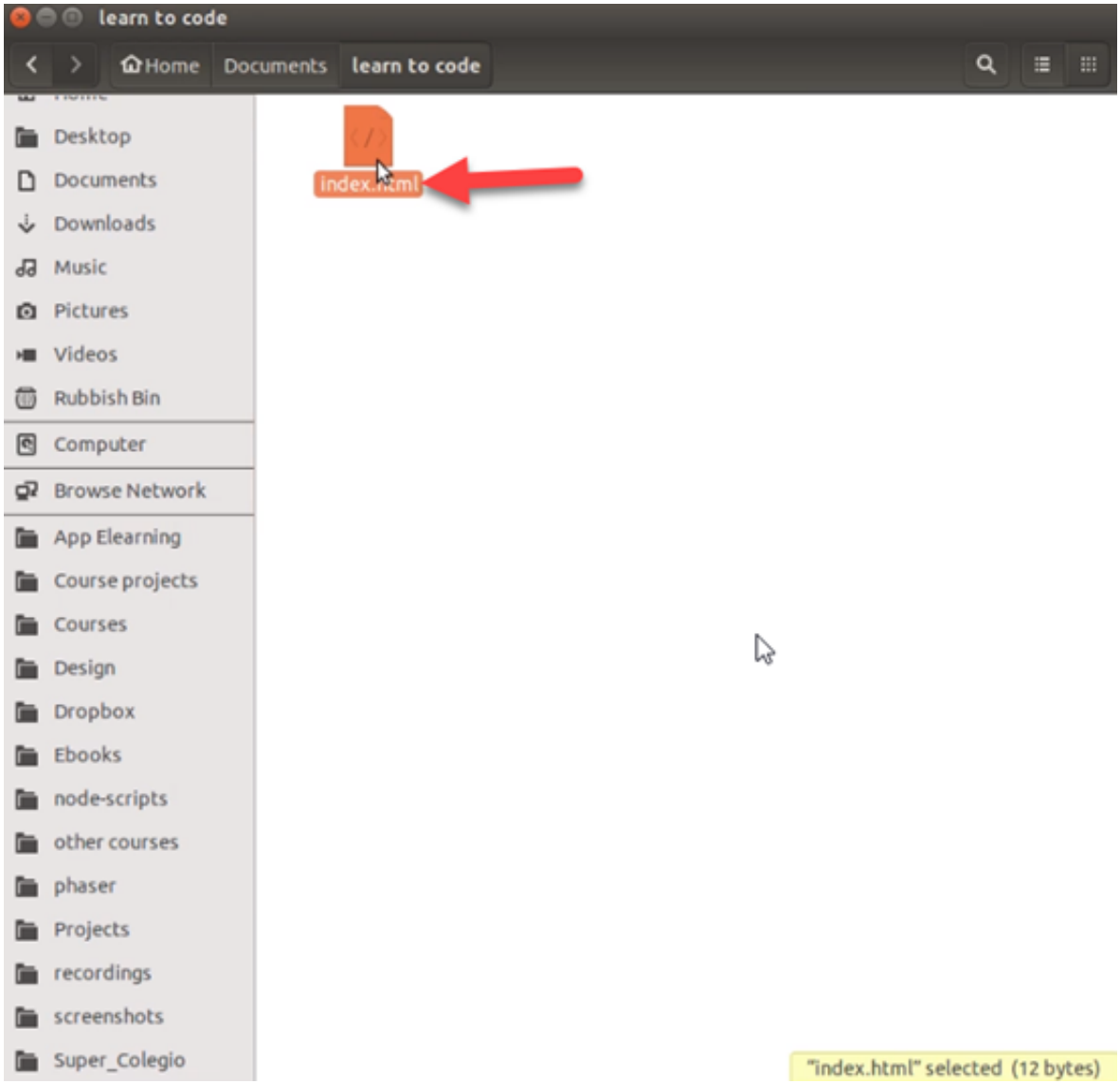
```
Hello World
```

This is what we want to display in the browser.

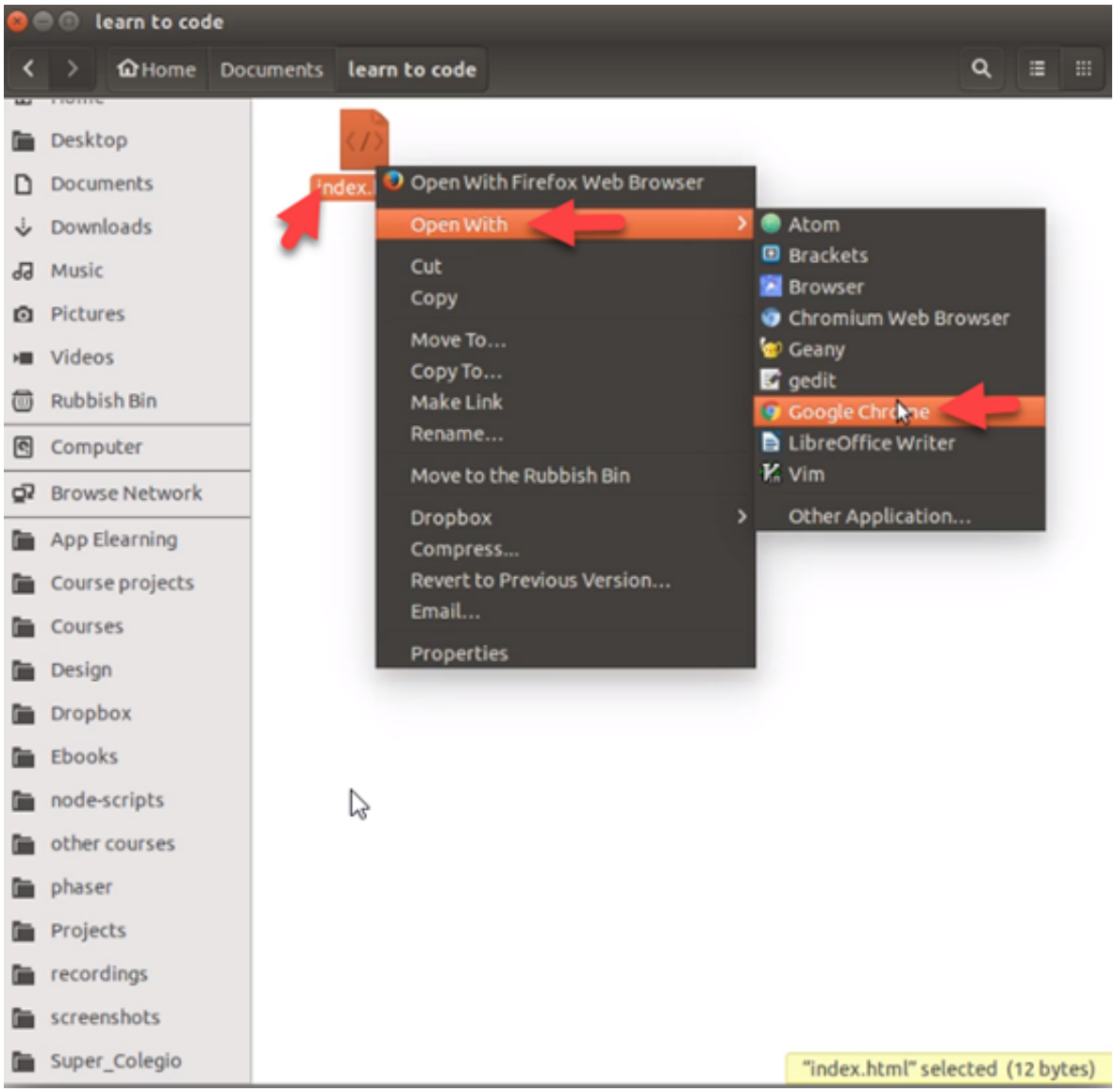
Save the file, and name it **“index.html”** HTML files always have the HTML extension.



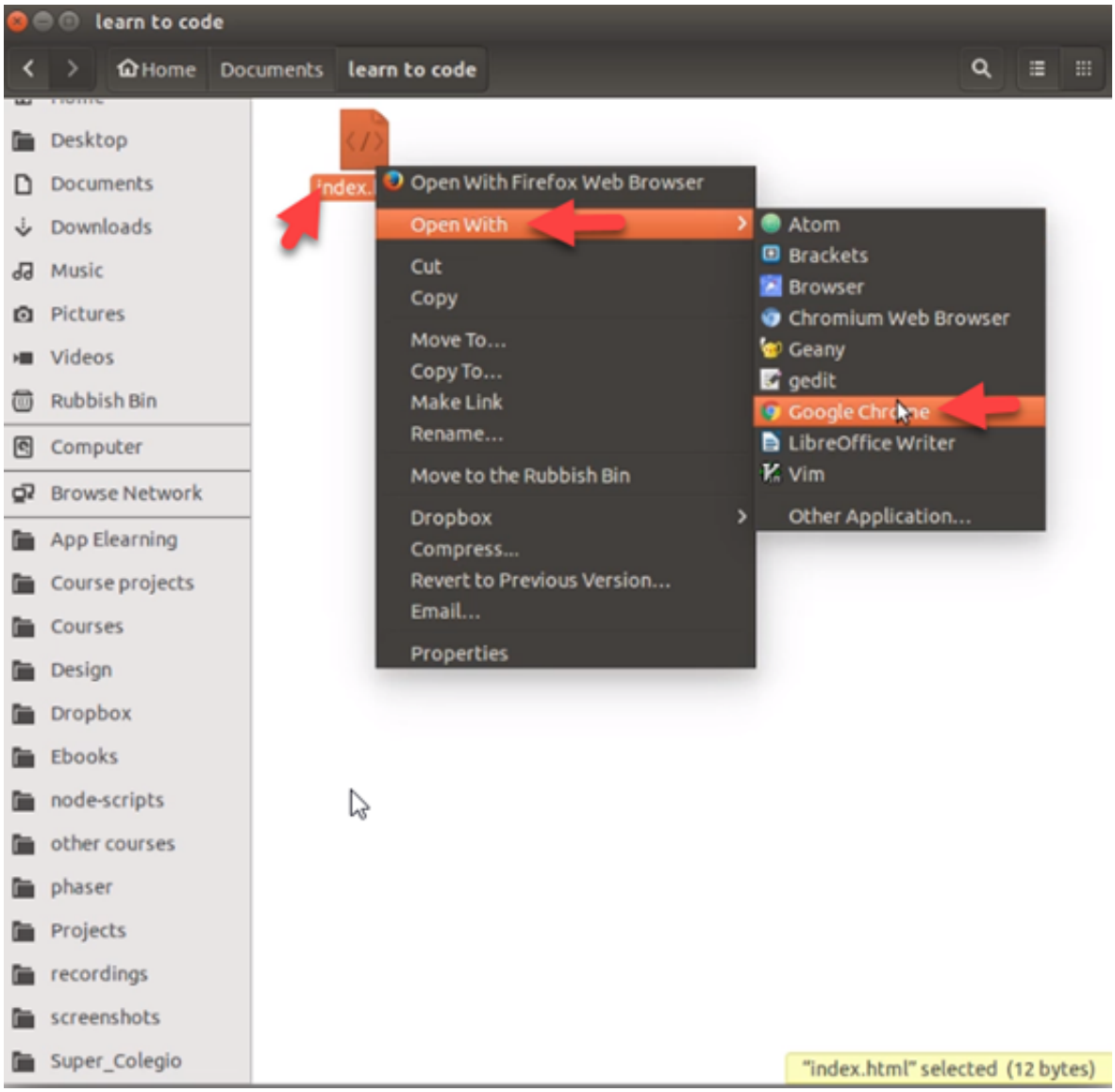
Now if you navigate back to your Learn To Code folder you will see the newly created file.



If you double click the file, it should show up in the web browser, but I am going be more specific and go to **Open With>Google Chrome**.



Navigate to the **Chrome** web browser and as you can see we have our Hello World up and running.



If you have followed along and reached this point, you have successfully setup your developer environment.

You are now ready to begin your coding journey.

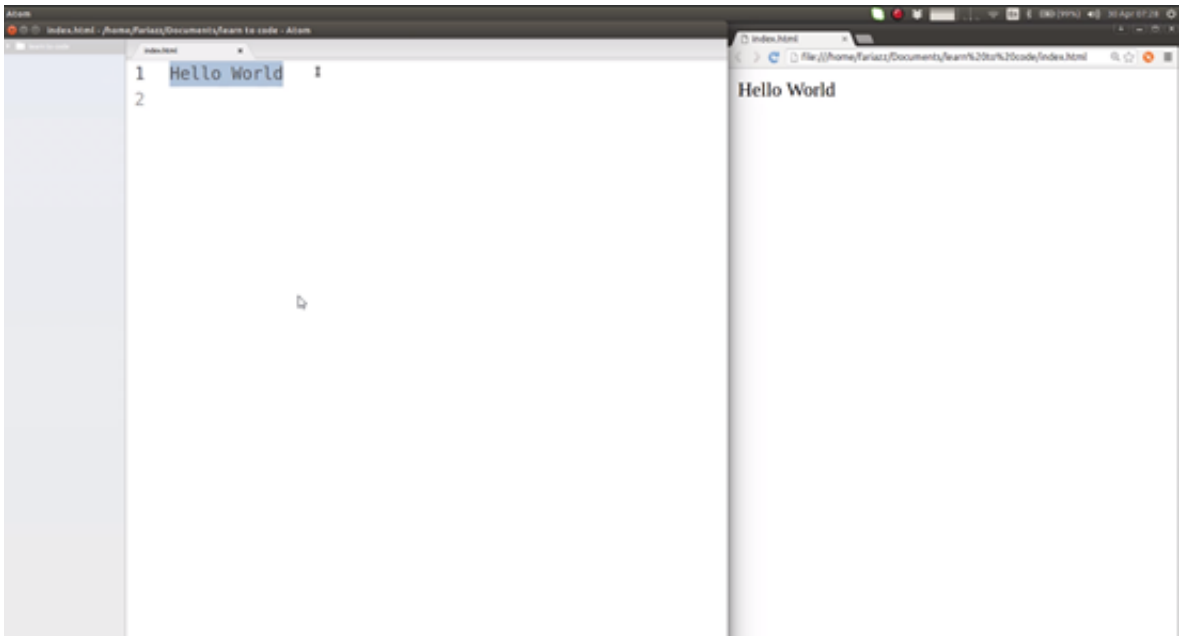
See you in the next lesson!

In this lesson you will create your very first **HTML tags**.

The role of **HTML** in a website is to **specify** the content you will **see** in the **web browser**.

Websites contain text, forms, tables, images, and many other things. So how do we tell the browser what it is we want to show?

This is where **tags** come into play. For example, I want to tell my browser that this “Hello World” is actually a main heading in my document.

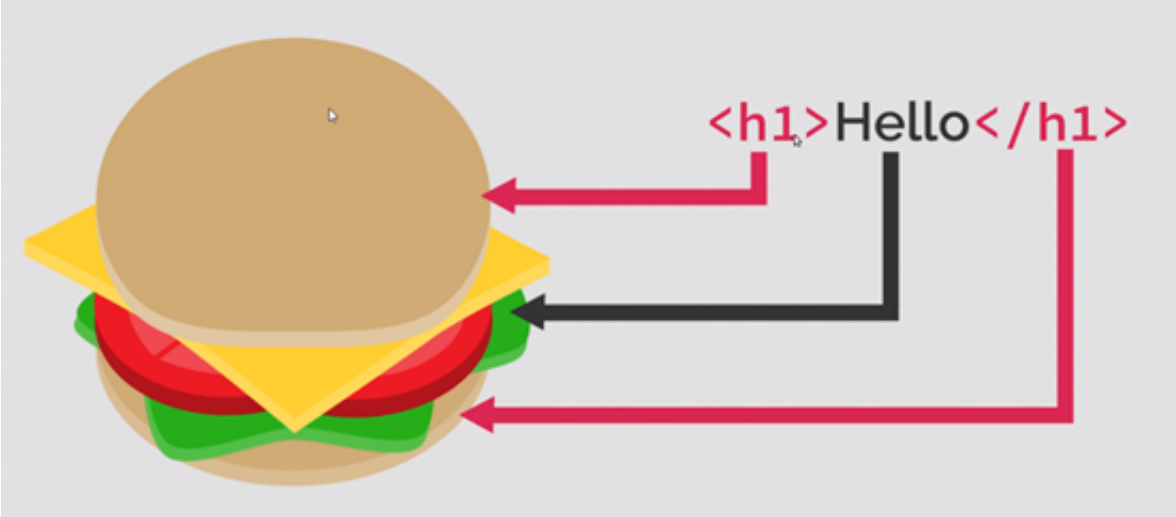


In an ideal world, we would be able to just talk to the browser and have what I want displayed as the main heading. This doesn’t work for now, so we need to use tags.

The Main Heading Tag

The first tag we will be learning is the **main heading tag**.

Tags are a bit like a sandwich: a tag has an **opening**, tags have the **filling**, and tags have a **closing**. This example below using the heading sandwich shows the **h1 tag**, which is a **main heading**.

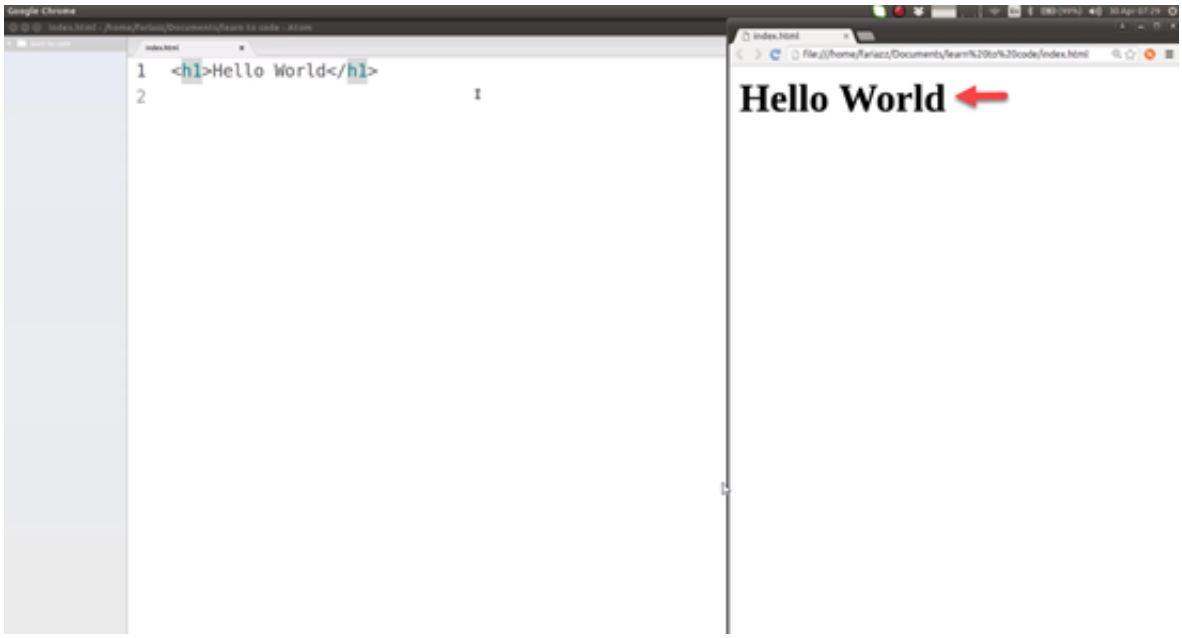


We will begin by **adding** the **h1 tag** to the Hello World code we already have:

```
<h1>Hello World</h1>
```

Make sure and **save** this.

If we look at **Google Chrome** and **refresh** the page, we now have the **main heading** displayed:



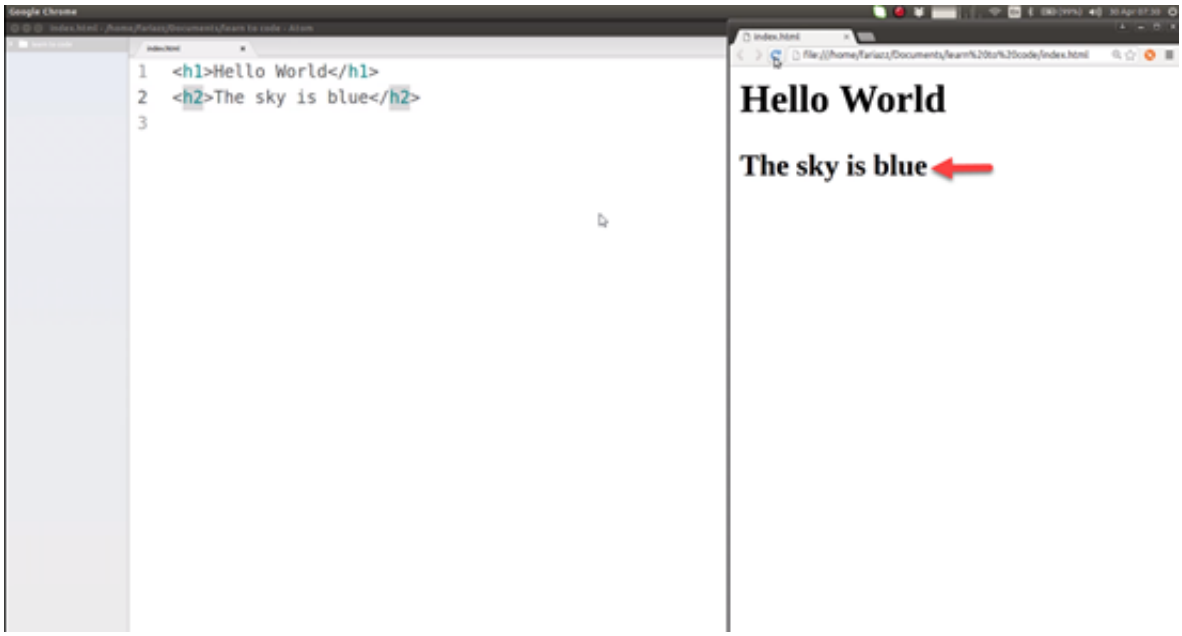
What if I want a secondary heading?

Secondary Heading

If I want a **secondary heading** I can use the **h2 tag**.

```
<h2>The sky is blue</h2>
```

Always make sure to **close** the tag, just like the sandwich, **save**, then **refresh** the page:



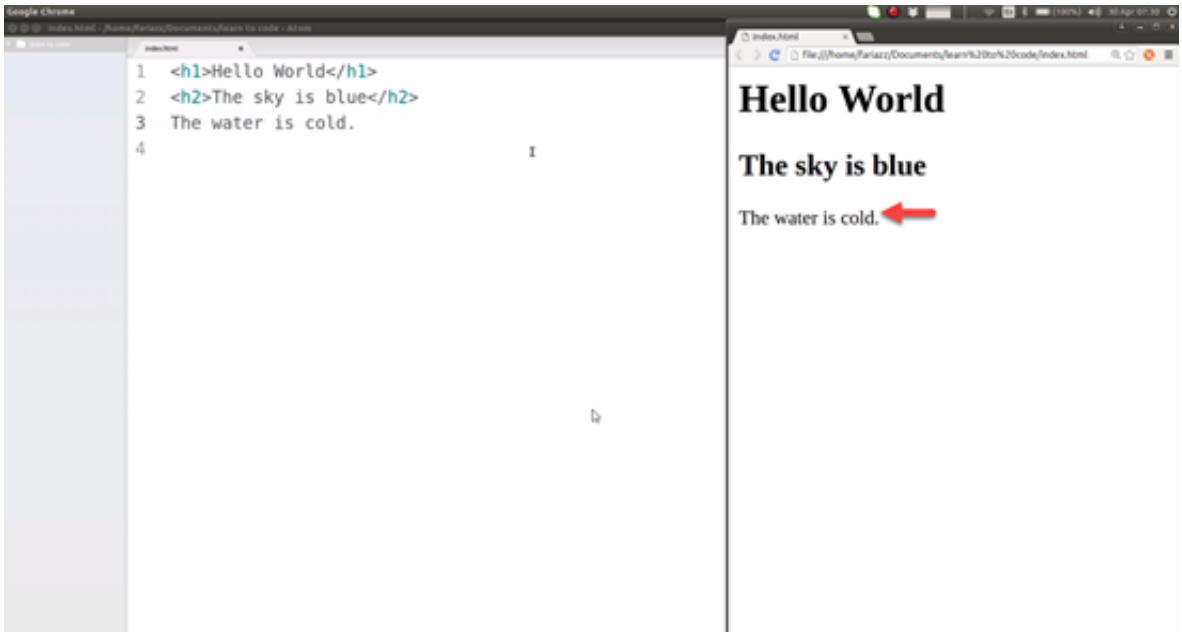
Now we have a secondary heading displayed.

How would i just add some other text?

Adding Text

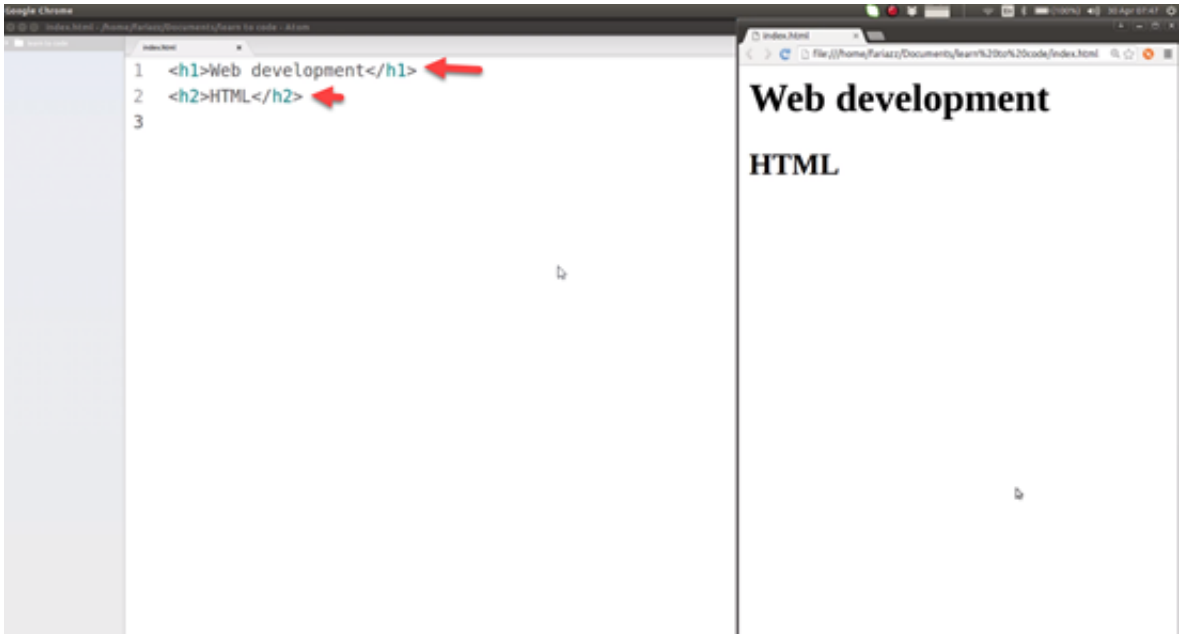
You just type the **text** below the main and secondary headings:

```
The water is cold.
```



HTML tags are the way for us to tell the page how we want to display are content.

In the lesson we will be talking about new lines and text paragraphs in HTML. We are seeing two different headings here:



These **headings** show on **different lines**. This makes us think that because they are shown on different lines its because they are on different lines in our code.

The truth is the browser does not care about the amount of times you press the enter key in your code.

The amount of times you press the enter key in your code will not effect the line spacing in your browser.

Even if you put everything on the same line and then refresh the page, it will still show separated on different lines:

The reason why we want to separate the line is because we want to make it more readable for humans.

Showing Paragraphs

What if I wanted to show two different paragraphs of text? See the code example below:

```
<p>I want to learn coding</p>

<p>Find free resources on zenva.com</p>
```

The ** tag** is used to tell the browser we want a paragraph. These work just like the main and secondary headings. You must have an opening, the filling, and then the closing.

The ** tag** for the **opening** tells the browser we are **starting** a paragraph here, and then the ** tag** for **closing** tells the browser we have **finished** the paragraph, please skip a line.

This is the basic of text paragraphs in HTML, and new lines in the browser.

Remember, it is the **HTML tags** that tell the browser how to display the content we want displayed in the browser, but when we learn **CSS** is when we will really have the capacity to modify these default behaviors. With CSS we will be able to make elements show the way want with regards to spacing, how much separation there will be between them, this is all handled by CSS.

For now, we are relying on a default style that the browser provides, and we are learning how to tell the browser what it is we want to show.

So far we have looked at individual pages, but the web is all about interconnection of different pages on different devices.

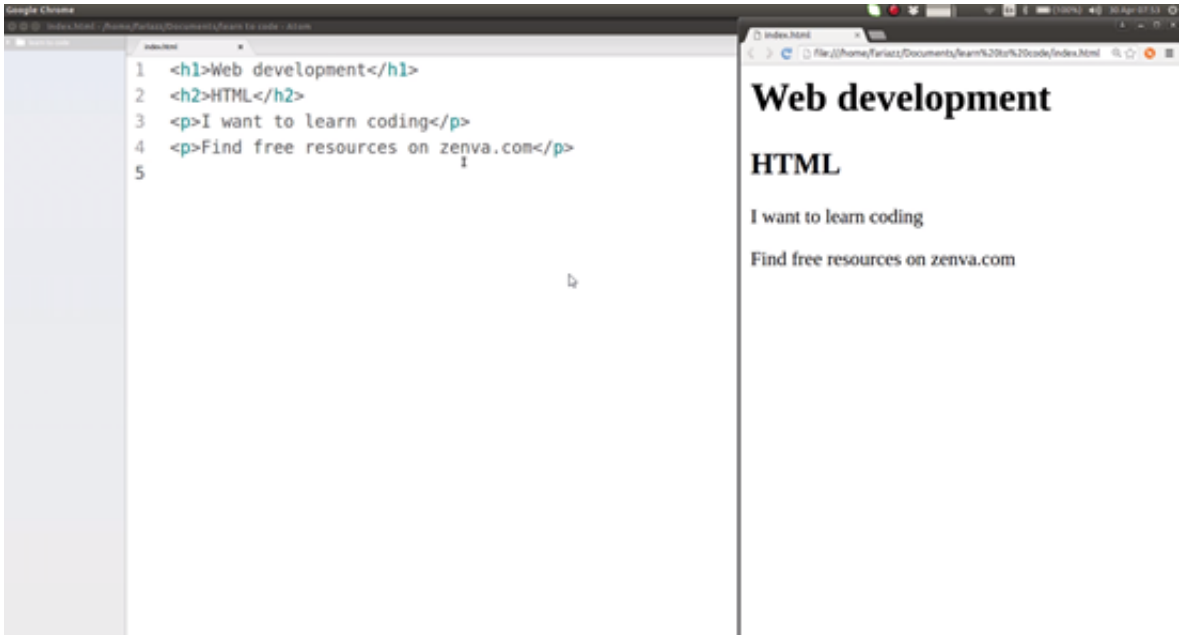
This is what the internet has always been about.

So, how do websites link to one another?

Well, we need to actually **create** the **links** for that. So that a website can link to other websites, and so forth.

Creating Our First Link

Open up the example we have been working with from the previous lessons:



Once you have it open we will begin creating our first link.

The **tag** that we use for a **link** is called the **anchor tag**, and it is just the letter **“a.”** See the code example below:

```
<p>Find free resources on <a>Zenva.com</a></p>
```

So we have **opened** the **link** with the **<a> tag**, then the Zenva part is the filling or what goes inside the link, and then we have **closed** the link with the ** tag**.

But, we are missing something here. Here we will introduce a new element of **HTML tags**, which are called **attributes**.

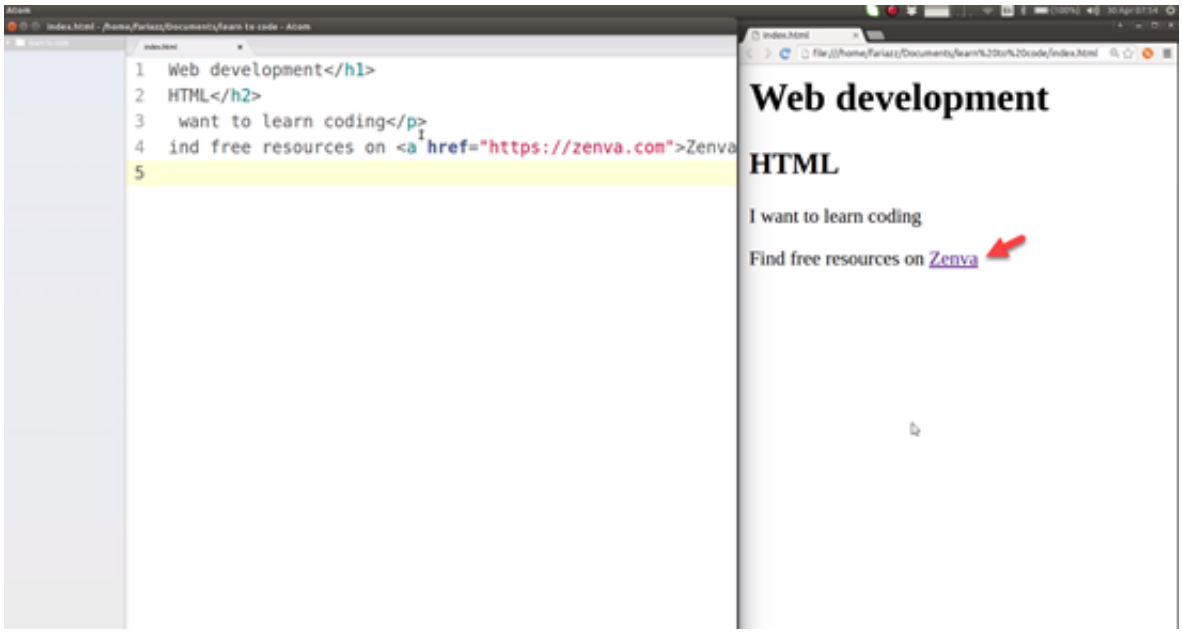
Attributes

Attributes allow you to specify more parameters of your tag.

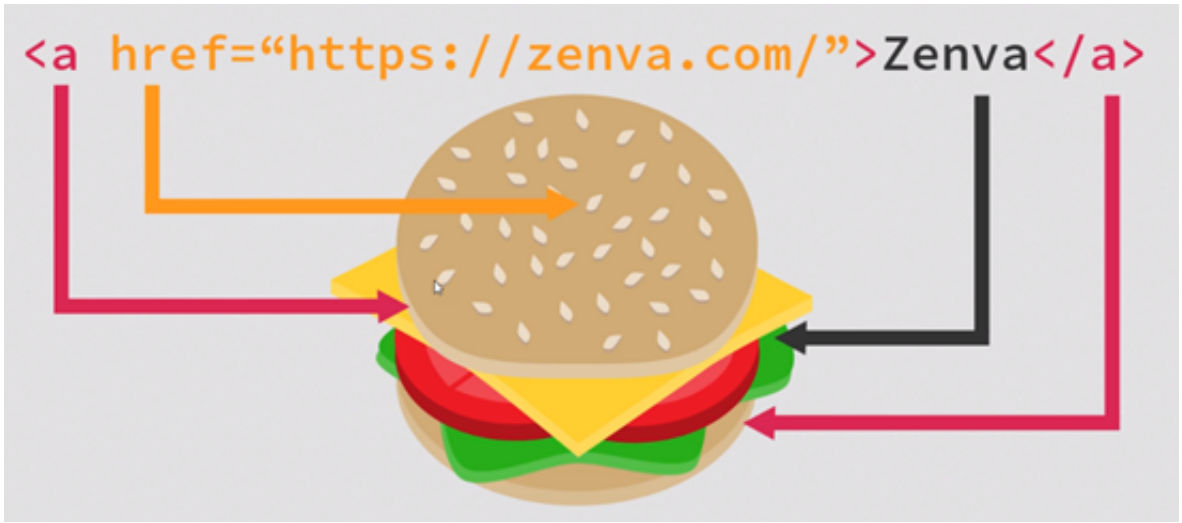
In this case we are going to use the **href attribute**, see the code below:

```
<p>Find free resources on <a href="https://zenva.com">Zenva</a>
```

So we now have setup the link:



The best way to think about attributes is to go back to the example from earlier with the sandwich:



So we have the **opening tag <a** and inside of your opening tag, we can **add attributes**.

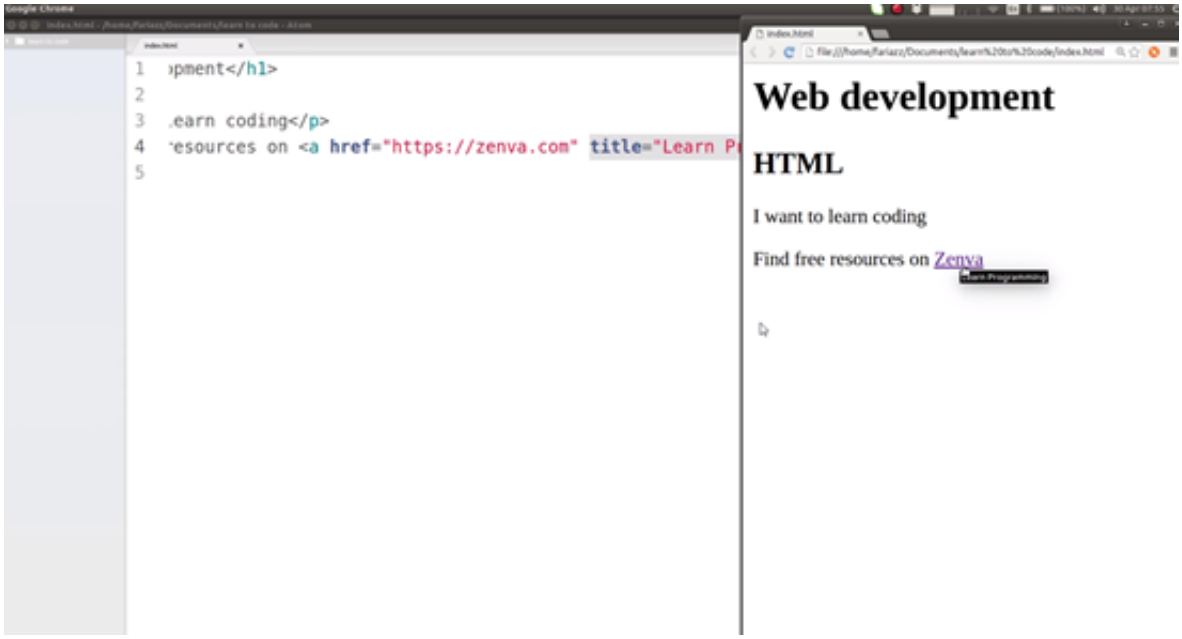
There are other attributes we can use as well, and the other one that will be shown in this lesson is the **title attribute**. See the code example below:

```
<p>Find free resources on <a href="https://zenva.com" title = "Learn Programming">Zen
va</a>
```

The **title attribute** is used so that when the mouse is **hovered** over the **link** this is what is seen.

So we have the **href attribute**, which is **necessary** for a **link** because its telling us where the link is going.

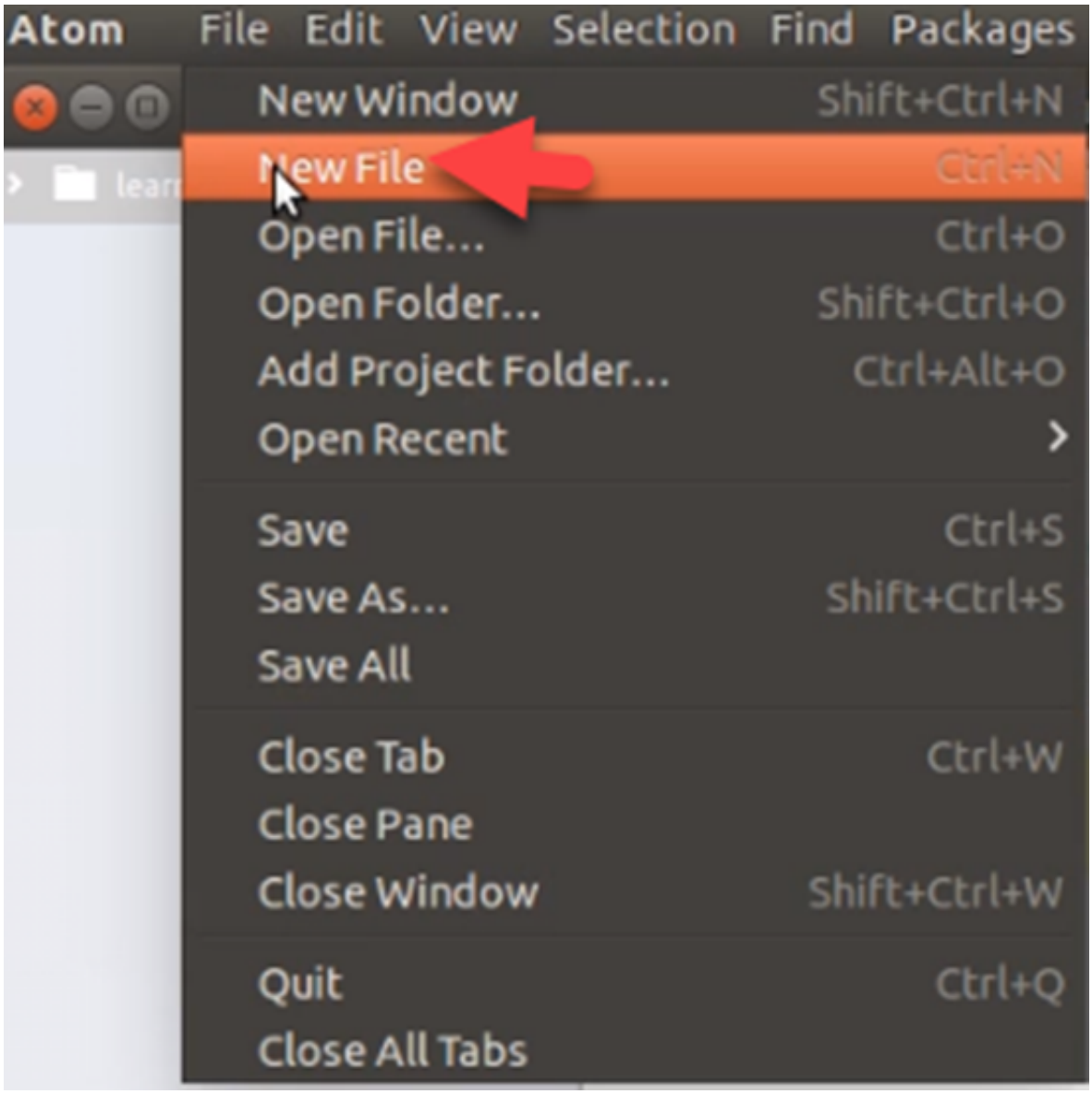
The **title attribute** displays the text when the **mouse** is **hovered** over the **link**.

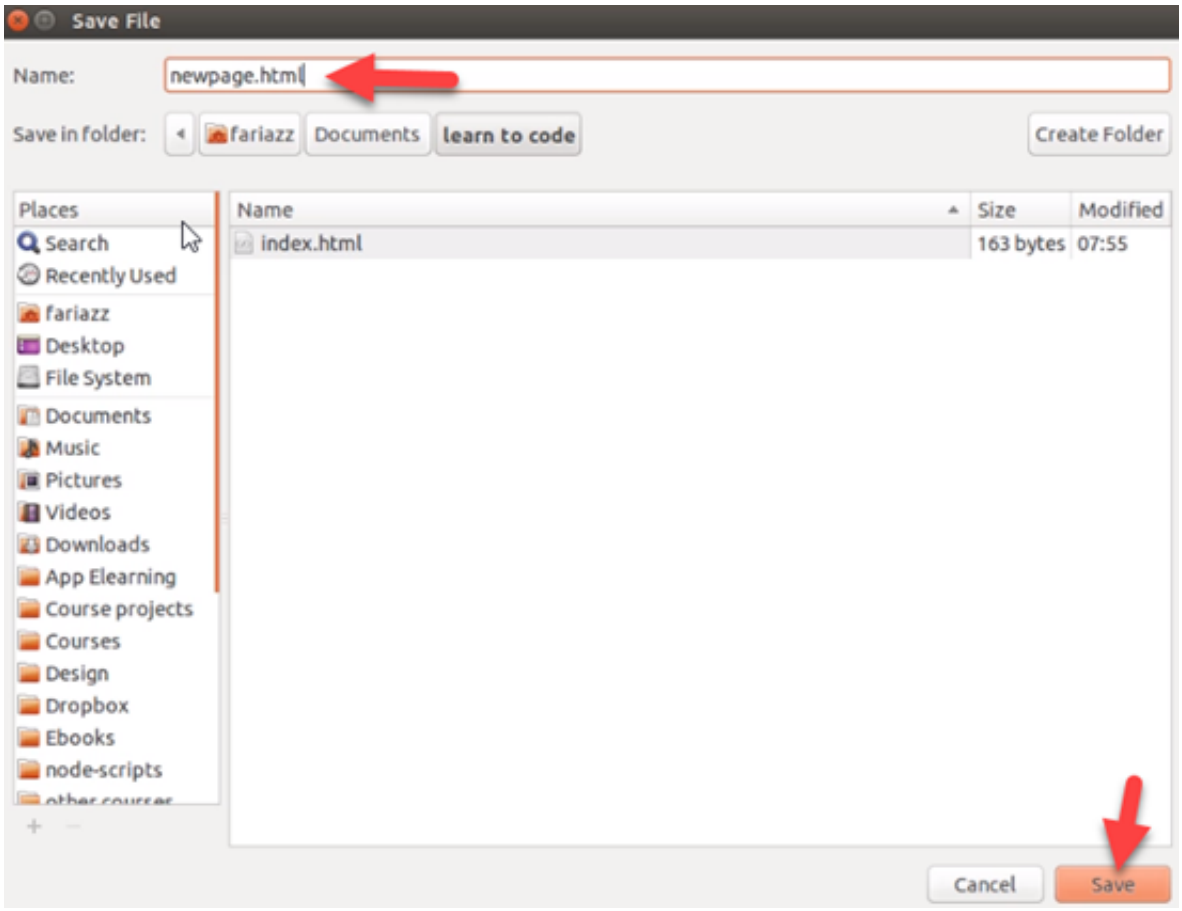


Links will allow you to connect different documents and they can be located in external URLs.

What if I wanted to link to a page that was also in the same directory of learn to code?

We can **add** a **new file** by selecting **File>New File**, and call this new file **“Newpage.html.”**



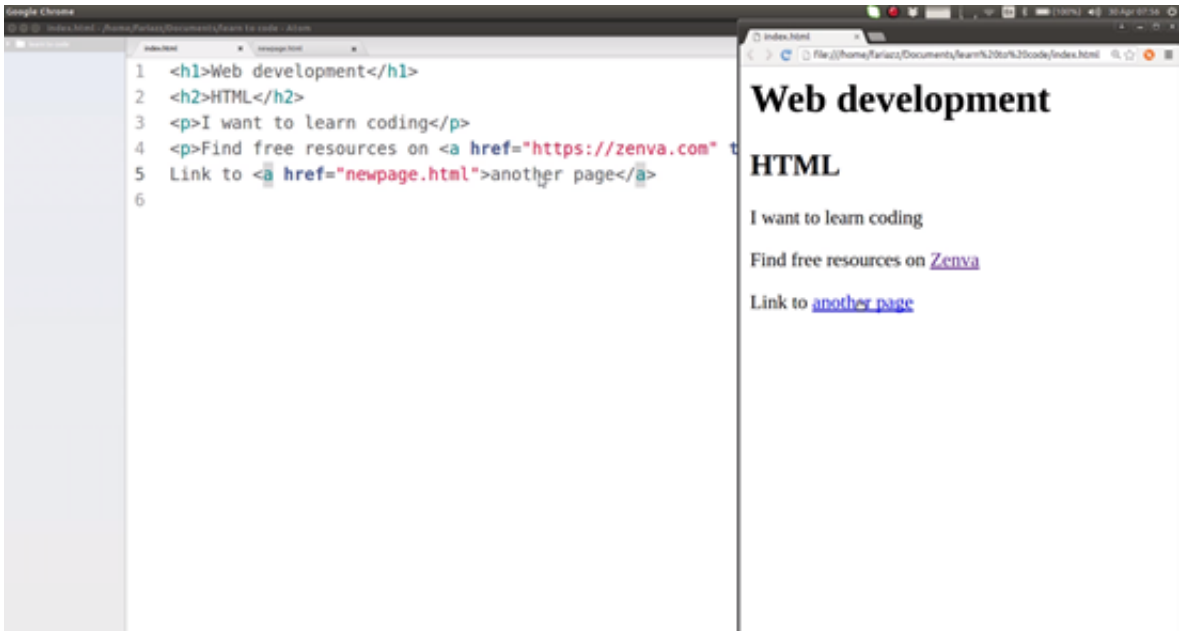


We can type **“New page”** on the first line of this file.

Then we are going to add another link to the **index.html** file, see the code below:

```
Link to <a href = "newpage.html">another page</a>
```

Save this, and **refresh** the page in the browser.



If you **click** on this **new link** we created it will take you to the new page that was created.

Summary

In summary, the web is an interconnection of different pages. Pages have never existed on their own and they have always had links to other pages.

In order for us to **create** those links we use the **anchor tag**.

We need to specify a **href attribute**, with the target of where this is taking us, which can be an external link, or it can be a page within our folder.

There are more attributes and options than these, but this is the core of it.

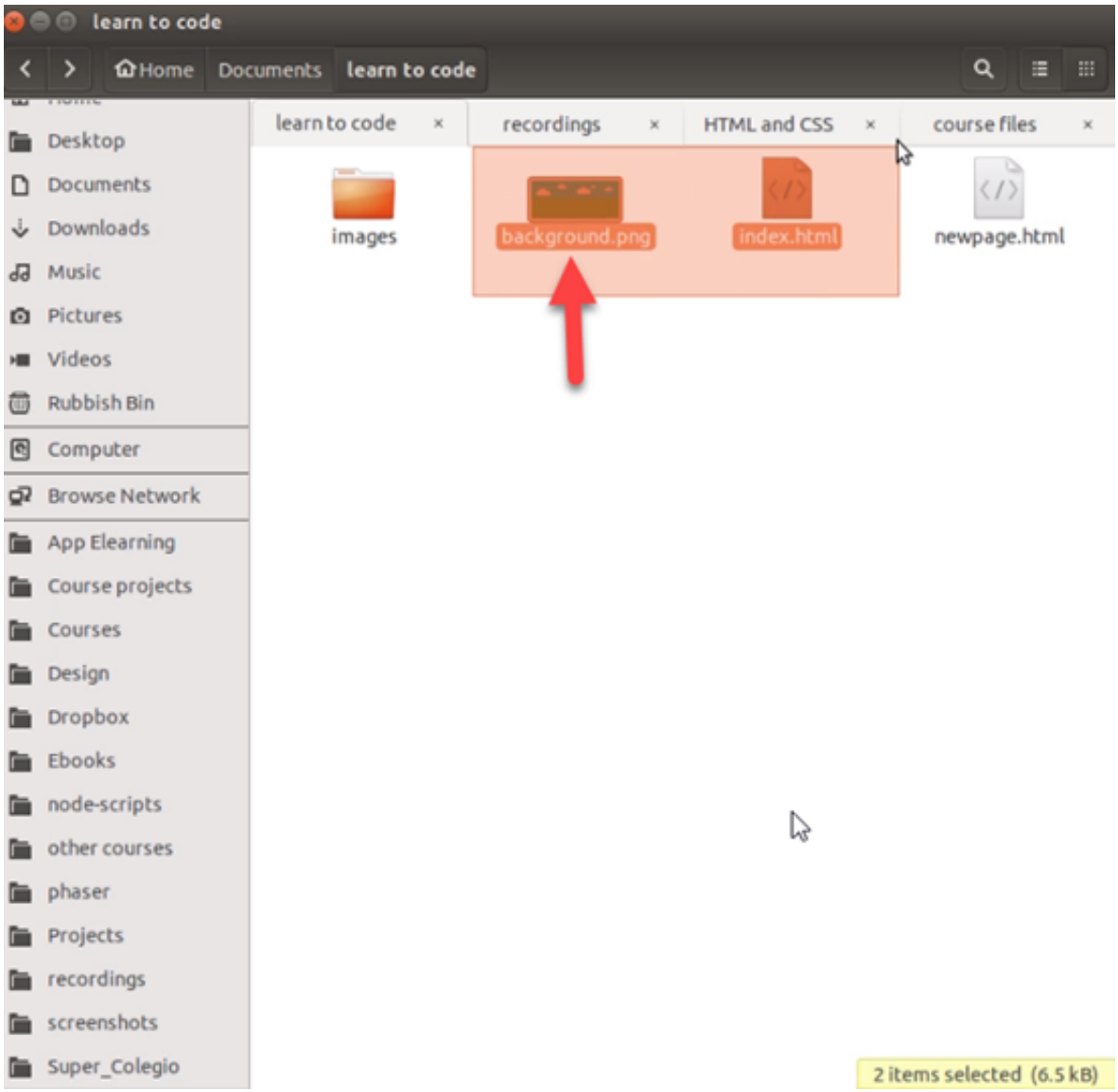
Everything goes **inside the <a> tag**.

It doesn't matter in which order you add the attributes either, you could add the title first, and then href next.

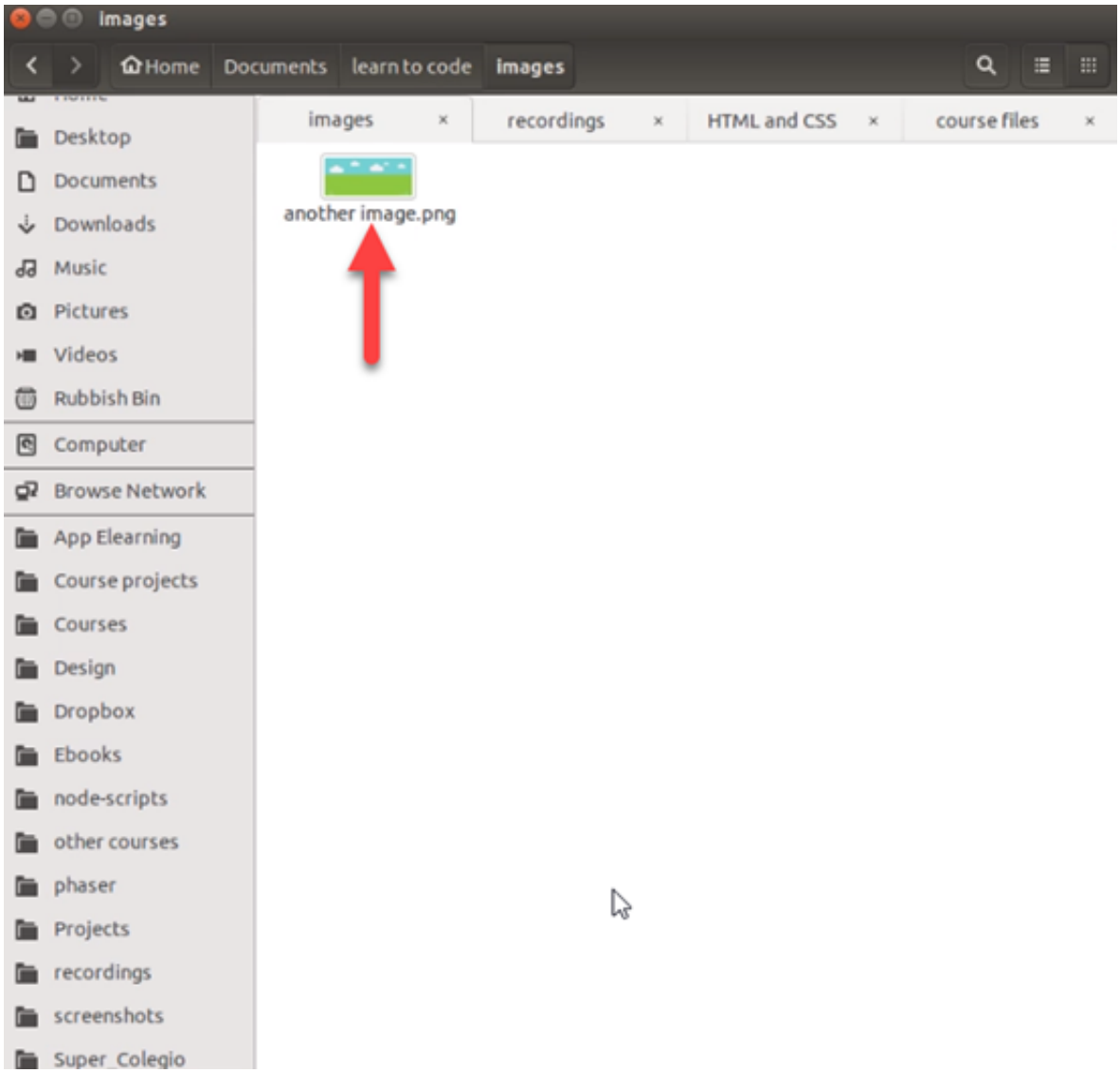
In this lesson we will be going over how to include **images** on your page.

We will be looking at two different cases where the image is in the same folder as your HTML files, and the second case we will look at is where the image is in a different folder or sub folder.

Make sure you have **downloaded** the **source code** files for this course, and if you have done so go to the folder that's called **images**, that has the assets for the images lesson,and you will see there is an image called **"background.png"** and we will be using this image file for the case in which the image is in the same folder as the index file.



There is also a folder called **images** and inside that folder there is another image called **"another image.png"** this will be used for the case where the image is located in a sub folder.



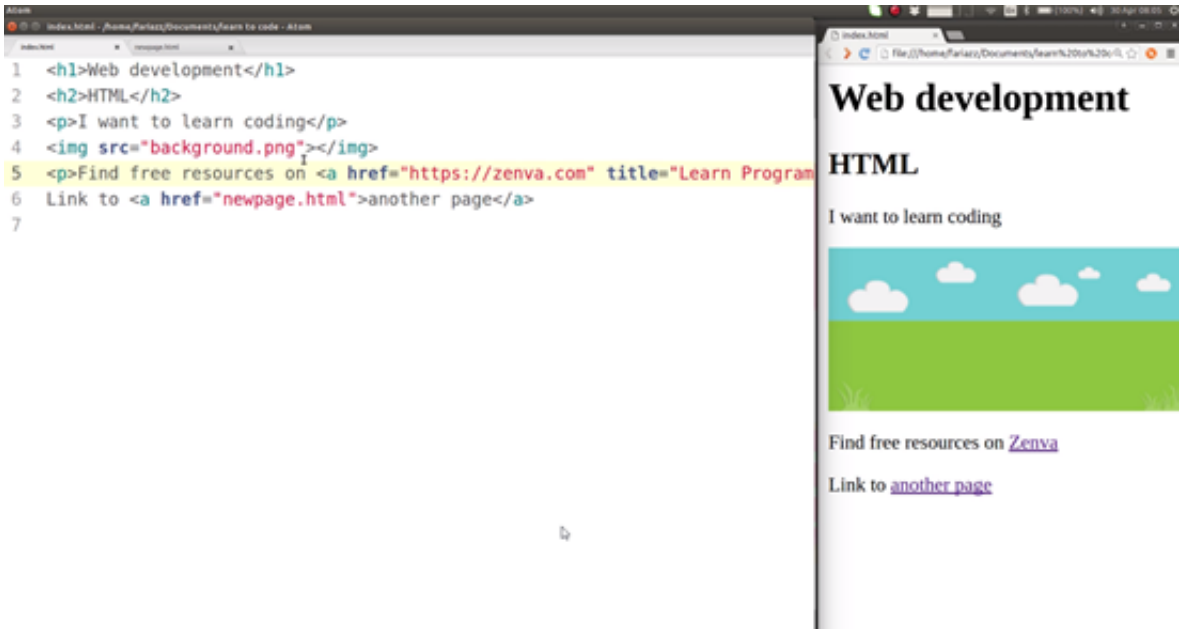
The Image Tag

The **tag** that we use for **images** is called the **“img” tag**.

Add the following code below to the file we have been working with in previous lessons:

```
<img src="background.png"></img>
```

Save this and **refresh** the web page.



Now, this is a sandwich without filling, and there is something that feels wrong about this.

There is something in **HTML** called **self-closing tags**, and you use that whenever you don't have this situation. This is used when **tags** don't have an **inside filling**, you just **close** them using the **self-closing tag**.

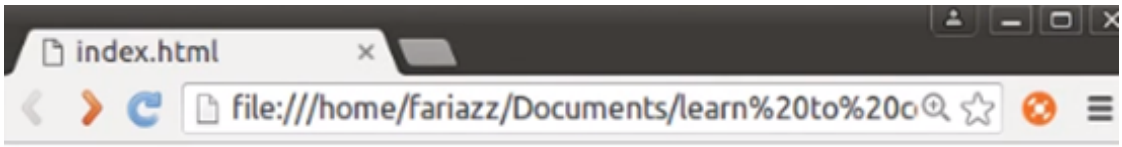
The best way to think of them is by thinking of a donut, which is in some way a self closing sandwich with no real filling.

They can have **attributes** as well, but it doesn't have an actual filling. The tag that is shown in the image above is called the **"break" tag**. This tag is a tag that can be used to **add new lines** inside of a paragraph.

So for example we can add a break tag like so:

```
<p>I want <br/> to learn coding</p>
```


Save and then **refresh** the page ans it will look like this in the browser:



Web development

HTML

I want
to learn coding

Find free resources on Zenva

Link to another page

So we have jumped a new line, and you can add multiple lines.

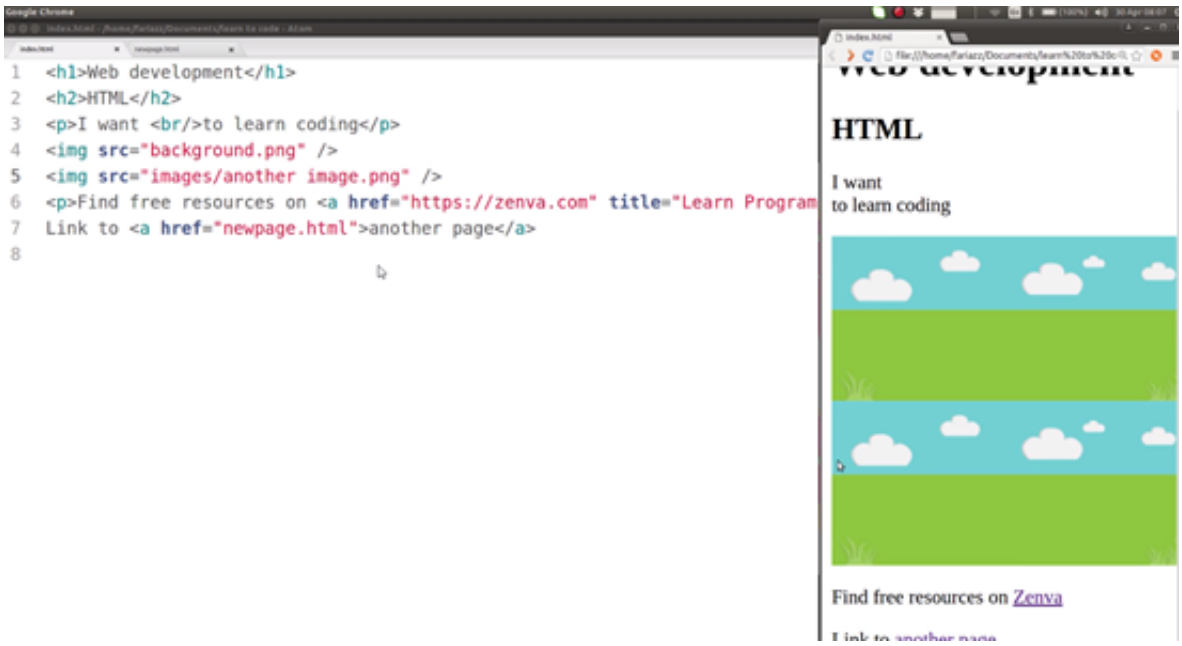
Don't use this for actually positioning your elements because that's done with **CSS**.

Images in Sub Folders

If you want to include an **image** that is located in a **sub folder**, you have to type in the name of the other folder and then forward slash and the name of that file which is **"another image.png."** See the code below:

```
<img src ="images/another image.png" />
```

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



So this has shown the other image that we wanted to include.

Summary

To summarize, the **"img" tag** is what we use to include **images**.

The source or the **SRC attribute**, allows you to **specify** the location of this image. If the image is an external page and you know the full URL, you can include it.

If the image is in the same folder as your page, you can just include the name.

If the image is in a sub folder or many sub folders, you can use the forward slash until you find the file, and then specify the full name of the file.

Some tags and the **img tag** is one of them, that **don't have a filling**, can be presented as **self-closing tags**, for which the donut example was used.

So now have a play with the code you have written, and continue to the next lesson.

In this lesson we will be creating **lists** in **HTML**.

There are two types of **lists** in HTML, and the **first type** deals with elements where the **order of the elements** does not matter. You can think of a to-do list, you have many things to do on the list, but it doesn't really matter in which order you execute them, but you have to complete them all on the list. This is called an **unordered list**.

On the other hand when you have any sort of recipe where the **steps** have to be followed in a **specific order**, this is called an **ordered list**.

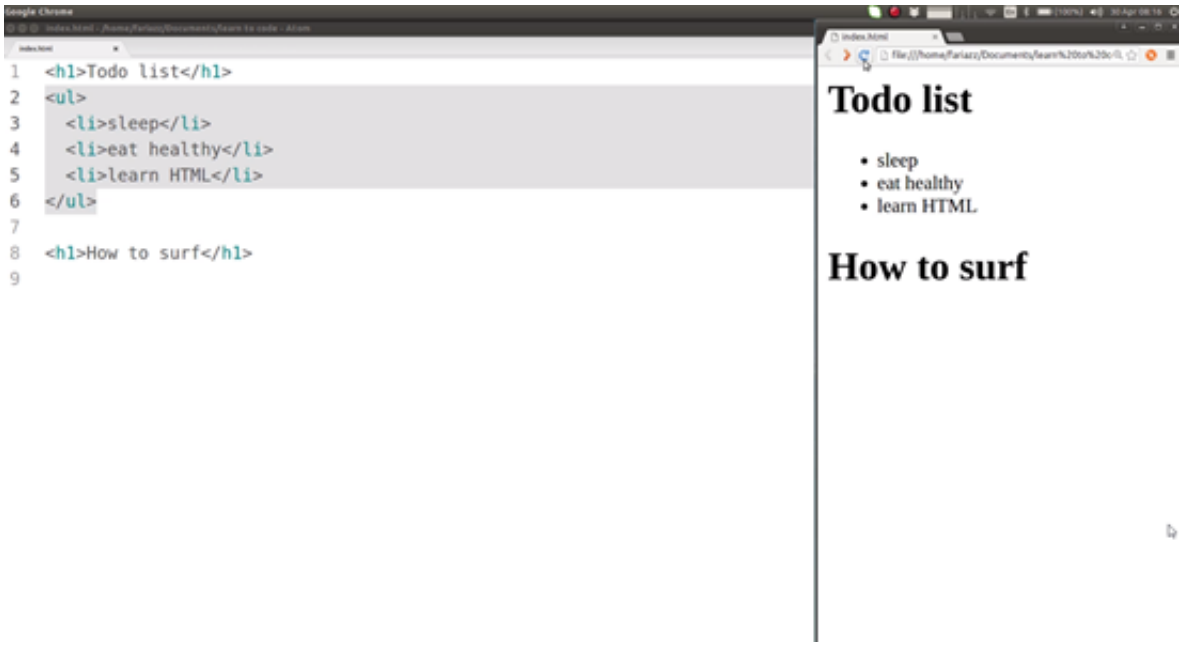
Unordered List

The **"ul" tag** is used for unordered lists in HTML.

We will start with an unordered list example, see the code below:

```
<h1> Todo list</h1>
<ul>
  <li>sleep</li>
  <li>eat healthy</li>
  <li>learn HTML</li>
</ul>

<h1>How to surf</h1>
```

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



We have successfully created an unordered list above.

Ordered List

What about ordered lists? Where the order of the elements does matter?

We will use the **“ol” tag** for ordered lists. We will create an ordered list next, where the order in which the elements are executed matters.

See the code below:

```
<h1> Todo list</h1>
<ul>
  <li>sleep</li>
  <li>eat healthy</li>
  <li>learn HTML</li>
</ul>

<h1>How to surf</h1>
<ol>
  <li>go to the beach</li>
  <li>spot a wave</li>
  <li>paddle</li>
  </ol>
```

Save this and **refresh** the page.

Lists are used to tell the browser there is going to be a **series** of list items.

We have to **tell** the **browser** what we want **ordered** and what we want **unordered**.

Forms are the main way that we use to **interact** with a **website**. If you want to **submit** some **information**, you will usually do it **through** a **form**.

HTML is the language that you use to **create** the **forms**.

The processing of forms themselves, such as storing the information in a database, or doing something with the information provided is **not executed** with **HTML**. That is done by a server side or client-side JavaScript.

In this lesson you will be introduced to forms in HTML.

We will be adding a couple of **input fields** and we will add a **button** to submit a form.

Creating a HTML Form

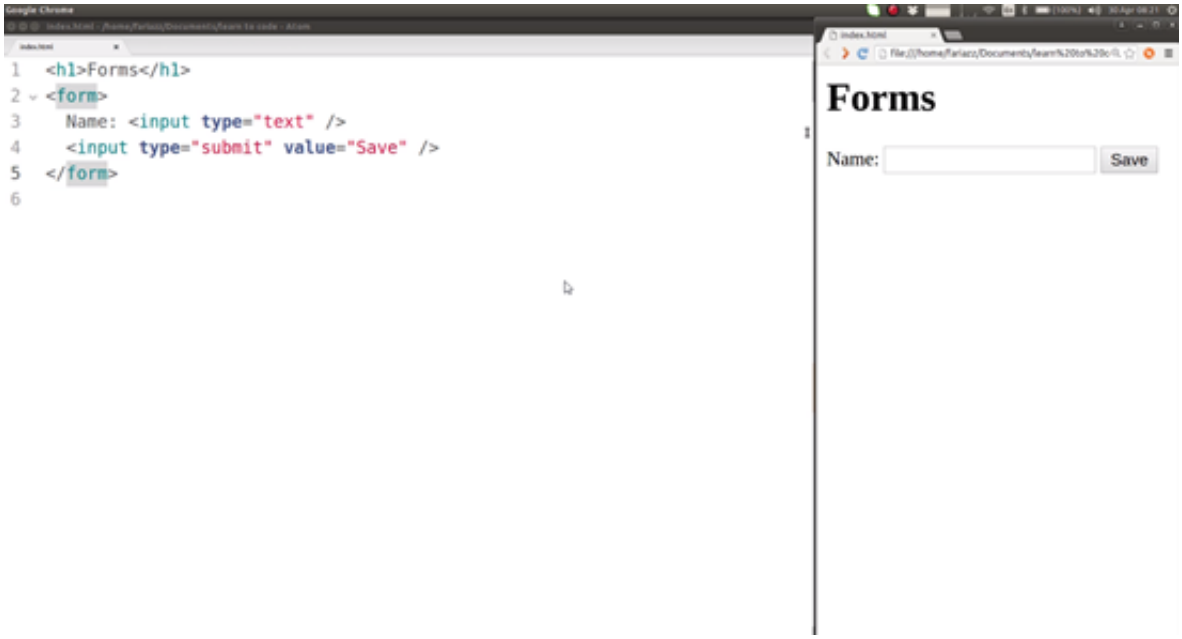
The way to initiate a form is by using the **form tag**, which tells the browser we are about to start a form.

The form tag is **<form>**.

See the code below:

```
<h1>Forms</h1>
<form>
  Name: <input type="text" />
  <input type= "submit" value="Save" />
</form>
```

Save this and **refresh** the page.

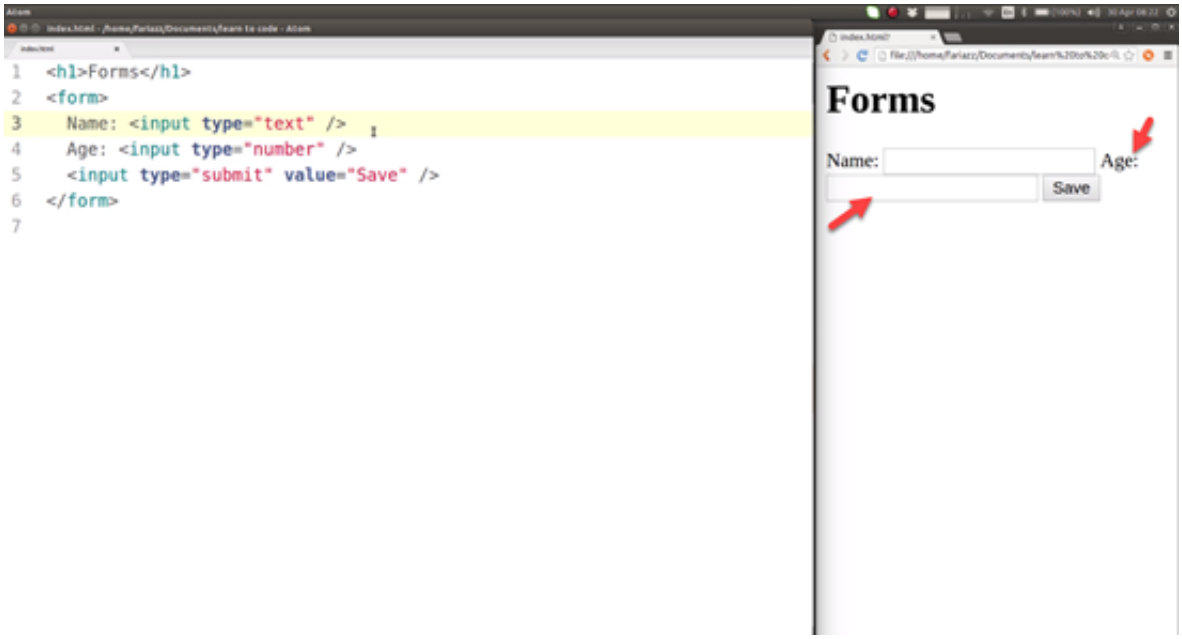


You have successfully created a form. It may be an extremely basic form, and nothing happens if the Save button is clicked, except reload the same page. This is not connected to a server site, so the saving of the information when the save button is clicked will not work.

Of course you can add more fields to the form, see the code below for the added fields:

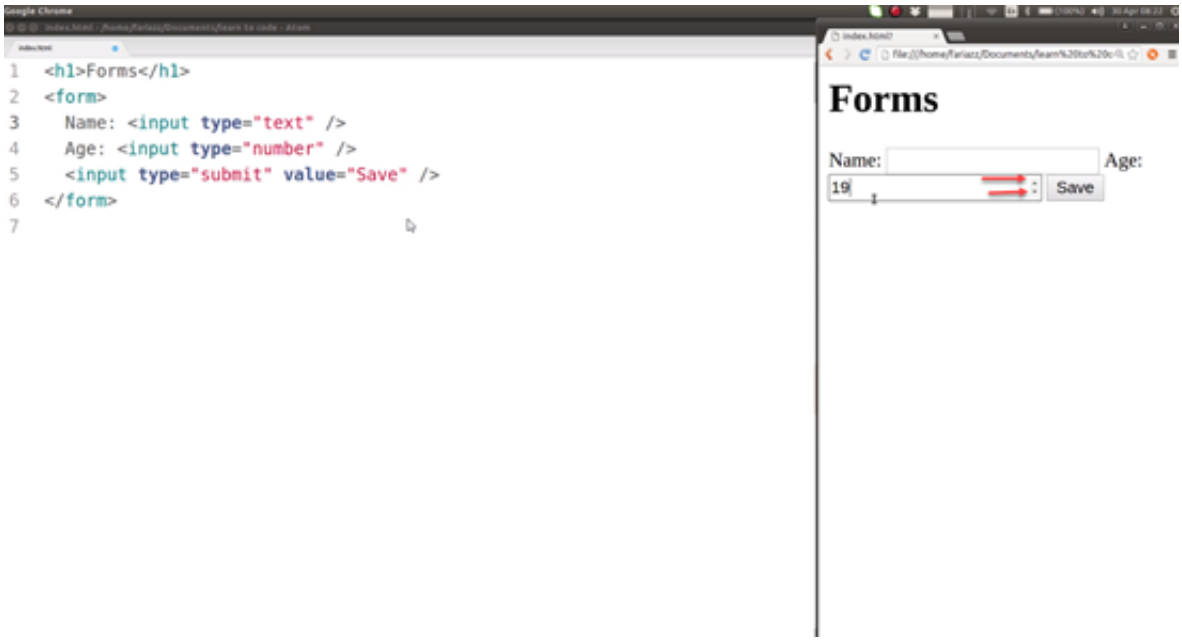
```
<h1>Forms</h1>
<form>
  Name: <input type="text" />
  Age: <input type="number" />
  <input type= "submit" value="Save" />
</form>
```

Save this and **refresh** the page.



You will notice that the form is not displaying correctly in different lines because as, I mentioned before the lines that you enter here do not matter, so you could have multiple lines, but it will not change how the information is displayed in the browser.

But, before the fix for the form is shown, take a look at the number field and you can see that you are only able to enter numbers. If letters are typed nothing will happen. You can also modify the number using the up and down arrow buttons.



We will now fix the multi-line issue we have with the form.

We could use something like the brake tag to add a new line, or we could even put this in a paragraph, but it doesn't feel correct.

The paragraph is for a text paragraph.

The break is to indicate a brake inside of a text paragraph, or a break in text.

What we want to do here is to use another **tag** called **"div."**

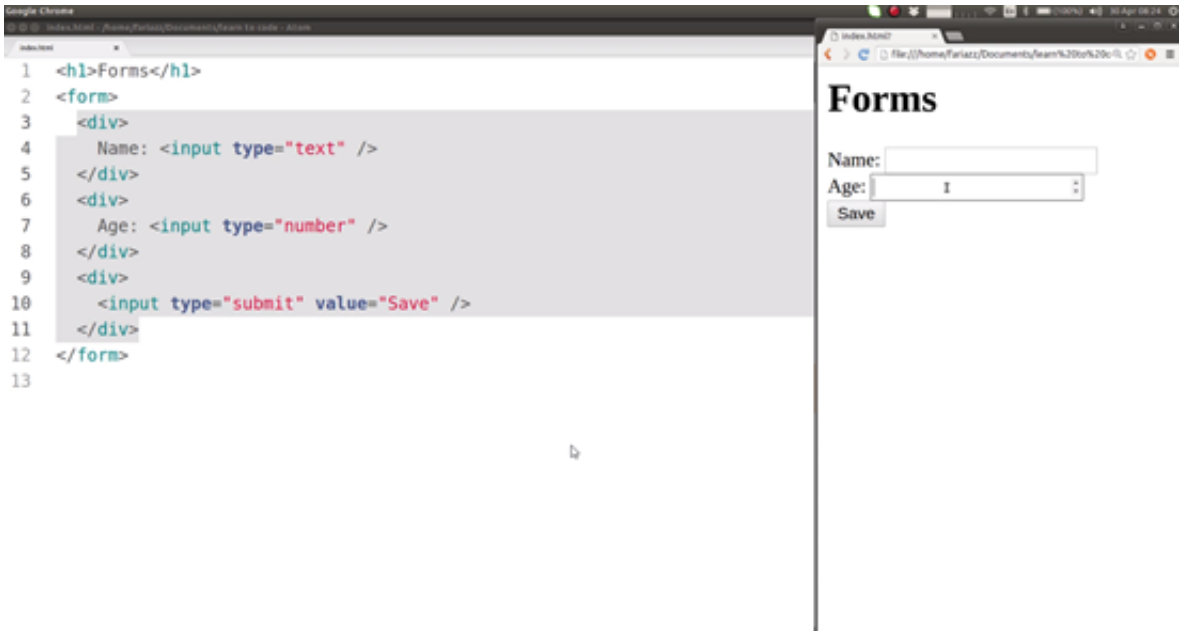
The div tag

The **div tag** is used for generic containers.

See the code below:

```
<h1>Forms</h1>
<form>
  <div>
     Name: <input type="text" />
  </div>
  <div>
     Age: <input type="number" />
  </div>
  <div>
    <input type="submit" value="Save" />
  </div>
</form>
```

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



So we have the basic form here that we created and the **input field areas** are arranged by using the **div tag**.

Remember, the behavior of the button is not handled by HTML.

Experiment with the code you have done here in this lesson, and in the next lesson we add some more fields.

In this lesson we will be adding some more fields to the form we created earlier in the course. You will also learn how to make the **forms required** on the **client side**.

Checkbox field

You will learn how to setup the **input** of type **check box**. See the code below:

```
<h1>Forms</h1>
<form>
  <div>
    Name: <input type="text" />
  </div>
  <div>
    Age: <input type="number" />
  </div>
  <div>
    <input type="checkbox" />
  </div>
  <div>
    <input type="submit" value="Save" />
  </div>
</form>
```

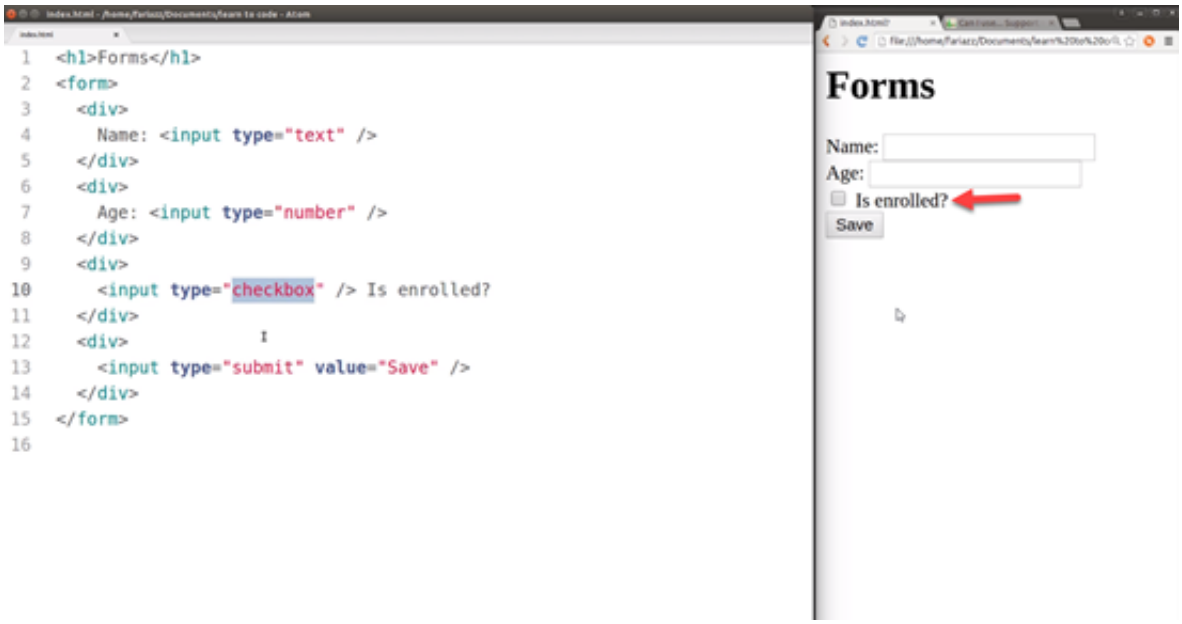
Save this, and **refresh** the page.

We now have a **checkbox** added to the page, and this box can be **clicked** on to activate and deactivate a **check mark** in the box.

You can even **add** some **text** to the **checkbox**.

```
<h1>Forms</h1>
<form>
  <div>
    Name: <input type="text" />
  </div>
  <div>
    Age: <input type="number" />
  </div>
  <div>
    <input type="checkbox" />  Is enrolled?
  </div>
  <div>
    <input type="submit" value="Save" />
  </div>
</form>
```

Save this and **refresh** the page.



Multi-Lined Text Inputs

We can use the **“textarea”** input for **multi-lined text** inputs in HTML.

See the code below:

```
<h1>Forms</h1>
<form>
  <div>
    Name: <input type="text" />
  </div>
  <div>
    Age: <input type="number" />
  </div>
  <div>
    <input type="checkbox" />  Is enrolled?
  </div>
  <div>
     <textarea><?textarea>
    <input type="submit" value="Save" />
  </div>
  <div>
     <input type+"submit" value= "Save" />
  <div>
</form>
```

Save this and **refresh** the page.

So we have the multi-lined text area.

Compulsory Field on the Client Side

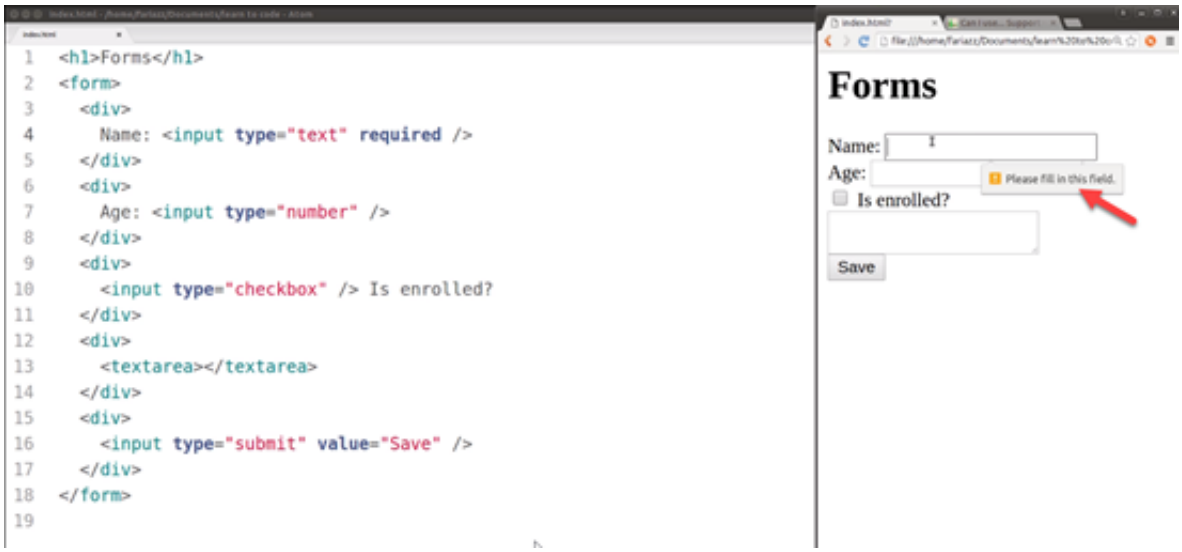
We need to add an **attribute** called **“required.”**

See the code below:

```
<h1>Forms</h1>
<form>
  <div>
    Name: <input type="text" required />
  </div>
  <div>
    Age: <input type="number" />
  </div>
  <div>
    <input type="checkbox" />  Is enrolled?
  </div>
  <div>
     <textarea></textarea>
    <input type="submit" value="Save" />
  </div>
  <div>
     <input type+"submit" value= "Save" />
  <div>
</form>
```

Save this and **refresh** the page.

Now if you try to hit the Save button without inputting anything into the Name field then you will get the popup message to “Please fill in this field.”



The browser will automatically detect that the Name field is a required field due to the “required” attribute we used. The user will be prompted to complete the field if they try to submit the form without filling in the required field.

You can also use the **“checked” attribute**, and the checkbox will show up checked already.

Summary

We’ve added a couple of new fields, added a checkbox, added a textarea, but the main thing with the textarea is that you cannot have it as a self-closing tag.

The textarea is a tag that can contain inside content, and that needs to be specified even if its going to be empty.

The required attribute allows us to make a field compulsory on the client side, and the browser will not let us submit the form if the field is empty.

You can also make a checkbox checked by default by using the “checked” attribute.

In this lesson we will add a couple more fields to the form we created earlier. We will add a field to **select** a **color** from a **list**, so there will be different colors and the user will be able to choose a color.

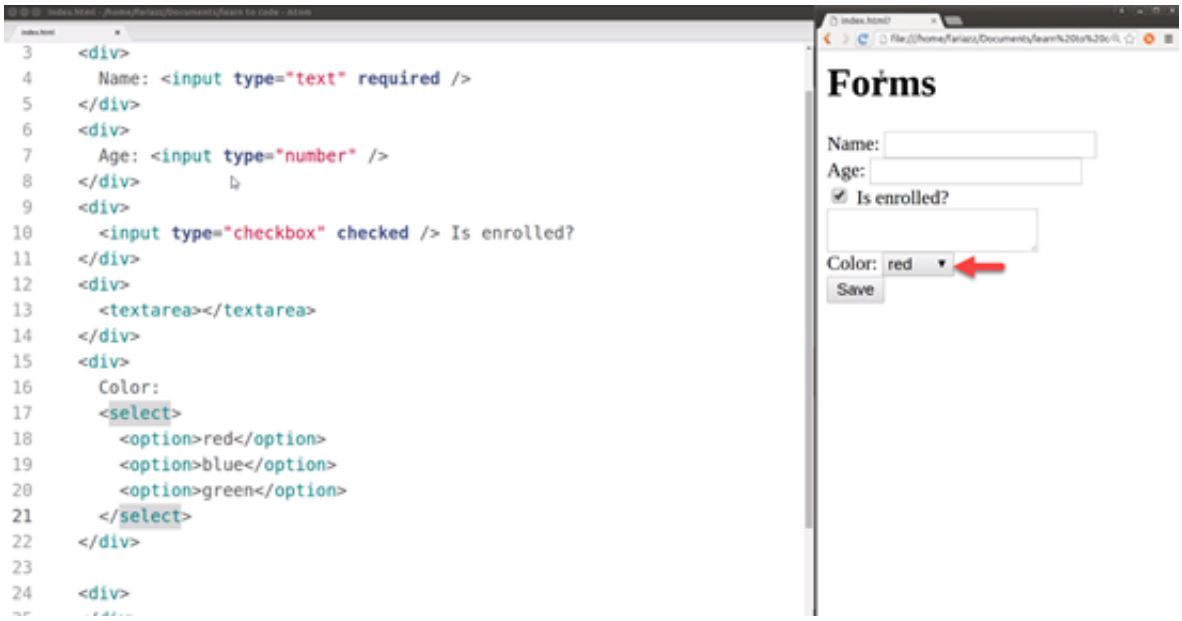
We will also add a **list** where the user can **pick multiple elements** from the list at the same time.

For both of these additions we will be using the **“select” tag**.

See the code below:

```
<h1>Forms</h1>
<form>
  <div>
    Name: <input type="text" required />
  </div>
  <div>
    Age: <input type="number" />
  </div>
  <div>
    <input type="checkbox" />  Is enrolled?
  </div>
  <div>
     <textarea></textarea>

  </div>
  <div>
    Color:
    <select>
      <option>red</option>
      <option>blue</option>
      <option>green</option>
    </select>
  </div>

  <div>
    <input type="submit" value="Save" />
  </div>
  <div>
     <input type+"submit" value= "Save" />
  <div>
</form>
```

Save this and **refresh** the page.



So as you can see this shows the user a **simple drop down menu** with the colors red, blue, and green. The user then can **select** a color from the drop down menu.

But, what if we wanted to have a **selection** start by **default**, so what if we wanted to start with the green color first, and not the red one?

Selected Attribute

This is where the **“selected” attribute** comes into use. We can use this **attribute** to make something in the color drop down menu show selected first.

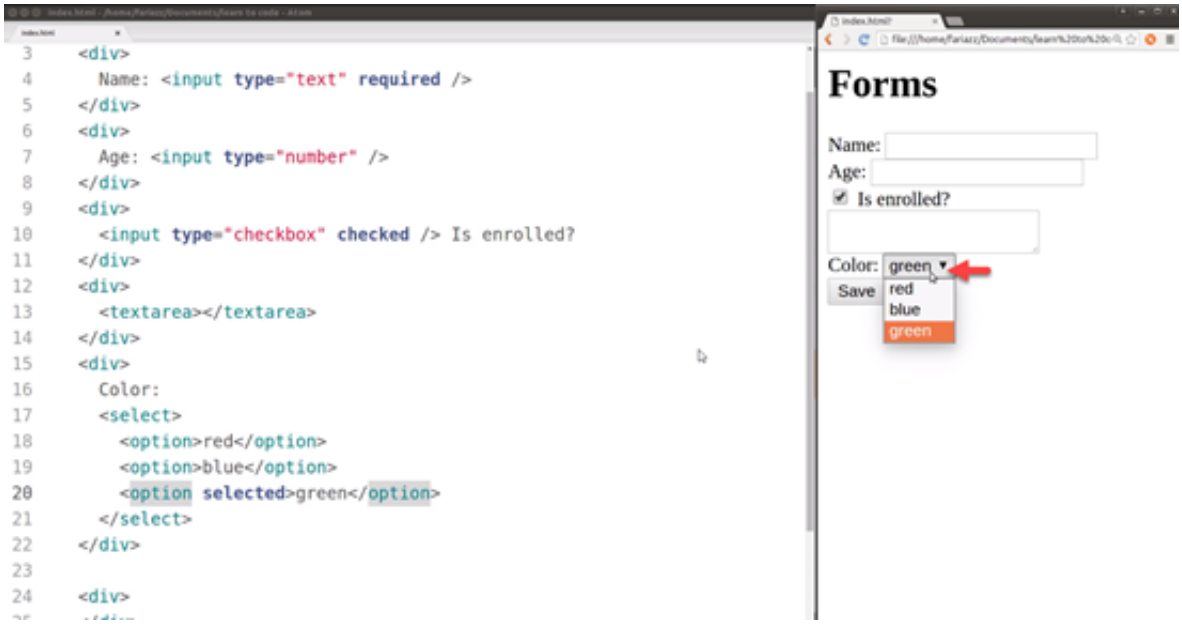
```
<h1>Forms</h1>
<form>
  <div>
    Name: <input type="text" required />
  </div>
  <div>
    Age: <input type="number" />
  </div>
  <div>
    <input type="checkbox" />  Is enrolled?
  </div>
  <div>
     <textarea></textarea>

  </div>
  <div>
    Color:
    <select>
      <option>red</option>
      <option>blue</option>
      <option selected>green</option>
    </select>
  </div>

  <div>
```



```
    <input type="submit" value="Save" />
  </div>
  <div>
     <input type+"submit" value= "Save" />
  <div>
</form>
```

Save and **refresh** the page.

Now the green color is the default option instead of the red color.



Now, what if we wanted to make it so the **user** could **select multiple items** from a list?

We can use the **attribute “select”** for this, but there’s a change that we need to make.

See the code below:

```
<h1>Forms</h1>
<form>
  <div>
    Name: <input type="text" required />
  </div>
  <div>
    Age: <input type="number" />
  </div>
  <div>
    <input type="checkbox" />  Is enrolled?
  </div>
  <div>
     <textarea></textarea>

  </div>
  <div>
    Color:
```



```
    <select>
      <option>red</option>
      <option>blue</option>
      <option selected>green</option>
    </select>
  </div>
    Locations:

    <select>
      <option>Brisbane</option>
      <option>Santiago</option>
      <option>Beijing</option>
      <option selected>Madrid</option>
    </select>

  <div>
    <input type="submit" value="Save" />
  </div>
  <div>
     <input type+"submit" value= "Save" />
  <div>
</form>
```

Save this and **refresh** the page.

So far, this is exactly the same we had before. So the change we actually need to make is to add an **attribute** called **“multiple.”**

See the code below:

```
<h1>Forms</h1>
<form>
  <div>
    Name: <input type="text" required />
```



```
  </div>
  <div>
    Age: <input type="number" />
  </div>
  <div>
    <input type="checkbox" />  Is enrolled?
  </div>
  <div>
     <textarea></textarea>

  </div>
  <div>
    Color:
    <select>
      <option>red</option>
      <option>blue</option>
      <option selected>green</option>
    </select>
  </div>
    Locations:
    <select multiple>

    <select>
      <option>Brisbane</option>
      <option>Santiago</option>
      <option>Beijing</option>
      <option selected>Madrid</option>
    </select>

  <div>
    <input type="submit" value="Save" />
  </div>
  <div>
     <input type+"submit" value= "Save" />
  <div>
</form>
```

So now if the user presses the **CTRL key** on the keyboard they are able to pick more than one option from the drop down menu.

Summary

The select tag allows us to create two different types of selections: Selections where you only pick one element, and selections where you can pick more than one element. Inside of select we can use option to specify individual elements, and we can use the selected attribute to specify which one is selected by default.

In this lesson you will learn how to **create** and add **tables** to your web pages.

Tables can be used to **show some information**, some data on your page.

Tables are not used for layout purposing or to position elements in a page.

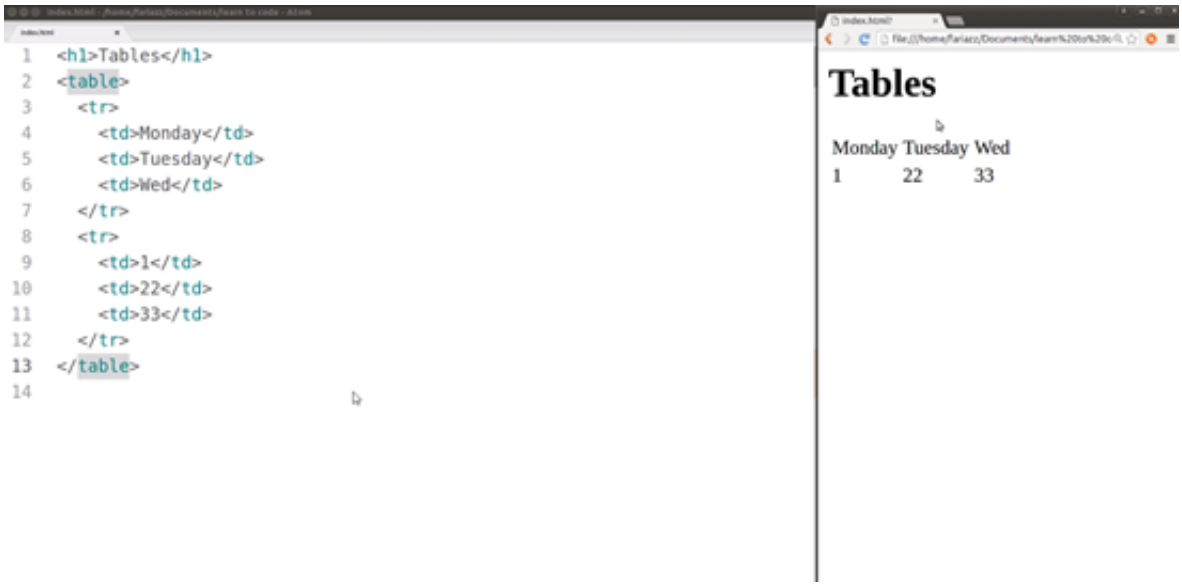
Create a Table

The **“table” tag** is used for creating a table.

See the code below:

```
<h1>Tables</h1>
<table>
  <tr>
    <td>Monday</td>
    <td>Tuesday</td>
    <td>Wed</td>
  </tr>
  <tr>
    <td>1</td>
    <td>22</td>
    <td>33</td>
  </tr>
</table>
```

Save this and **refresh** the page.



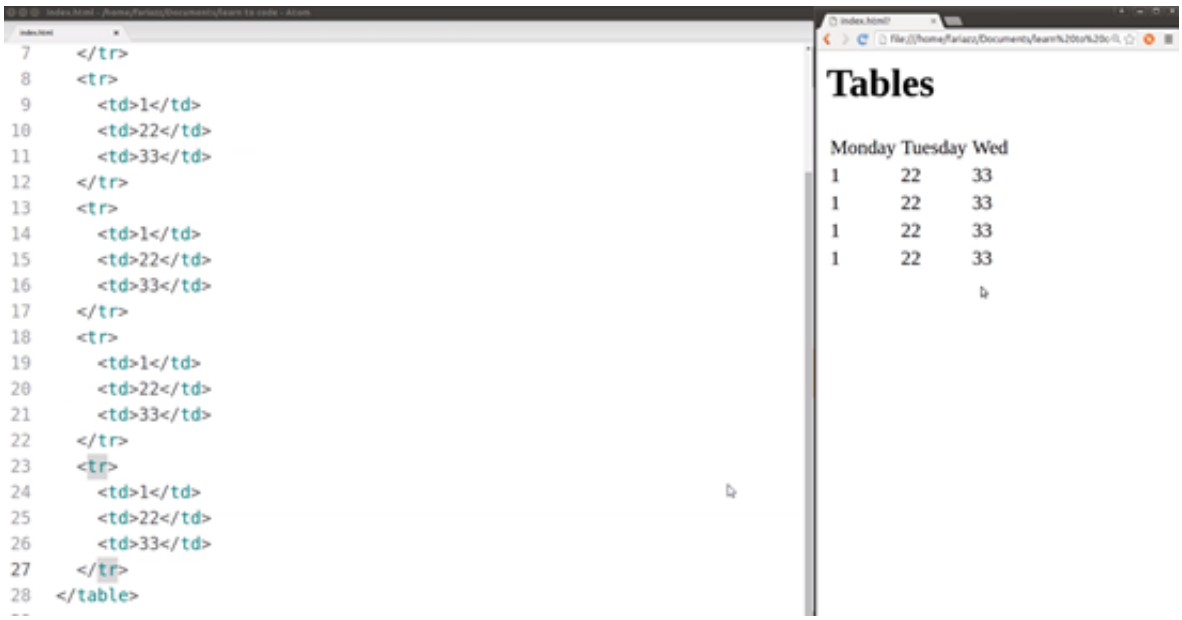
So, if we add more rows to the table:

```
<h1>Tables</h1>
<table>
  <tr>
    <td>Monday</td>
    <td>Tuesday</td>
```



```
    <td>Wed</td>
  </tr>
  <tr>
   <td>1</td>
   <td>22</td>
   <td>33</td>
  </tr>
  <tr>
   <td>1</td>
   <td>22</td>
   <td>33</td>
  </tr>
  <tr>
   <td>1</td>
   <td>22</td>
   <td>33</td>
  </tr>
   <tr>
   <td>1</td>
   <td>22</td>
   <td>33</td>
  </tr>
</table>
```

Save this and **refresh** the page.



Usually the first row will contain **headings**, so you may want to tell the browser that the first elements are actually table headers.

So for **table headers**, instead of using the “td” the tag called **“th”** is used instead.

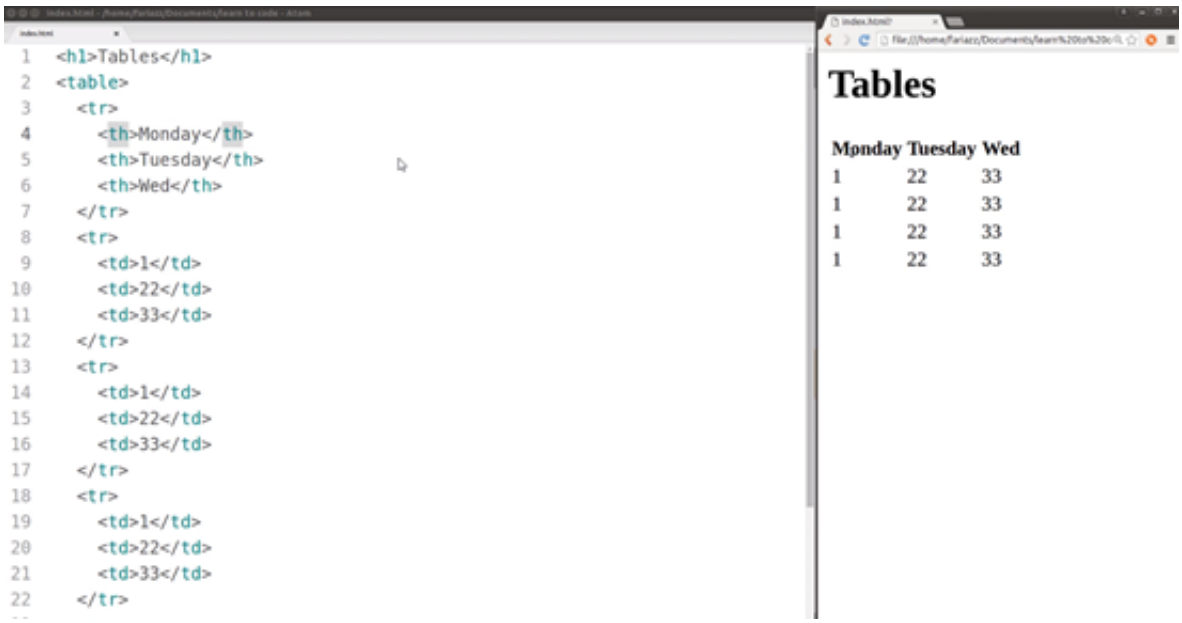
Table Headers

The **tag “th”** is used for **table headers**.

See the code below:

```
<h1>Tables</h1>
<table>
  <tr>
    <th>Monday</th>
    <th>Tuesday</th>
    <th>Wed</th>
  </tr>
  <tr>
   <td>1</td>
   <td>22</td>
   <td>33</td>
  </tr>
  <tr>
   <td>1</td>
   <td>22</td>
   <td>33</td>
  </tr>
  <tr>
   <td>1</td>
   <td>22</td>
   <td>33</td>
  </tr>
   <tr>
   <td>1</td>
   <td>22</td>
   <td>33</td>
  </tr>
</table>
```

Save this and **refresh** the page.



The **table headers** are actually shown in **bold** by default.

As you can see creating tables is quite easy once you get used to the row by row approach.

HTML doesn’t work in the way where you can create one column at a time. HTML tables need to be created one row at a time.

Whenever you look at a web site on the internet what's shown on the actual browser content is called the **body** of the page, but there's more to a page than just the body.

The actual title for a web page, the text you see on the tab for the web site you visit is added in an area called the **page head**.

The **page head** contains information about the page itself.

There is a **general structure** of a web site which is defined by a set of standards, the **HTML5 standards**.

Required Standards

The standards require that you do the following:

```
<!DOCTYPE html>
<html>
  <head>
     <title>This is a page </title>
  </head>
  <body>
    <h1>HTML page structure.</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.

The **title** on the **tab** now says **"This is a page."**

What are some of the other things we can add?

If we enter a **non-English character** in the code:

The Spanish characters are not displayed correctly.

This can be fixed by **defining the encoding of the page** so that it can display all Unicode characters, not just English default letters and numbers.

You add the **“meta” tag**, which is a tag that **specifies information about the page**, and this is all done in the **head**.

See the code below:

```
<!DOCTYPE html>
<html>
  <head>
     <title>This is a page </title>
     <meta charset="UTF-8">
  </head>
  <body>
    <h1>HTML page structure.</h1>
    <p>This is a complete page</p>
    <img src ="background.png" />
    <p>HTML is easy!</p>
  </body>
</html>
```

So with the meta tag added we can now display non-English characters.

The **meta tags** do **not** need to be **self-closed**, this is part of the HTML standard.

Another interesting meta tag to add has to do with the description, so we type in name, description, and then you can add a description to your page in content.

See the code below:

```
<!DOCTYPE html>
<html>
  <head>
     <title>This is a page </title>
     <meta charset="UTF-8">
     <meta name ="description" content="This is what Google will show!">
     <link rel="icon" href="favicon.ico" >
  </head>
  <body>
    <h1>HTML page structure.</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.

Now we have the fav icon displaying on the tab.

In the next lesson we will be going over how to use the **Chrome developer tools**. This is where you can look at the page source code.

Summary

Your page is not just the body, its not just what you see. There is more to it than what you actually see. There is information about the page. That has to do with that your browser shows with specifications of the language. There was just one last thing that needed to be added and this will tell the browser and Google as well what language this page is on. So this page is in English, see the code below:

```
<!DOCTYPE html>
<html lang= "en">
  <head>
     <title>This is a page </title>
     <meta charset="UTF-8">
     <meta name ="description" content="This is what Google will show!">
     <link rel="icon" href="favicon.ico" >
  </head>
  <body>
    <h1>HTML page structure.</h1>
    <p>This is a complete page</p>
    <img src ="background.png" />
    <p>HTML is easy!</p>
  </body>
</html>
```

So, you begin by typing in DOCTYPE html and this tells the browser this is our website and this is an HTML document.

Then you start with the HTML tag where you specify the language of the page, and this is also used by Google to know how to index your page, should it be in English search results, should it be in Spanish search results?

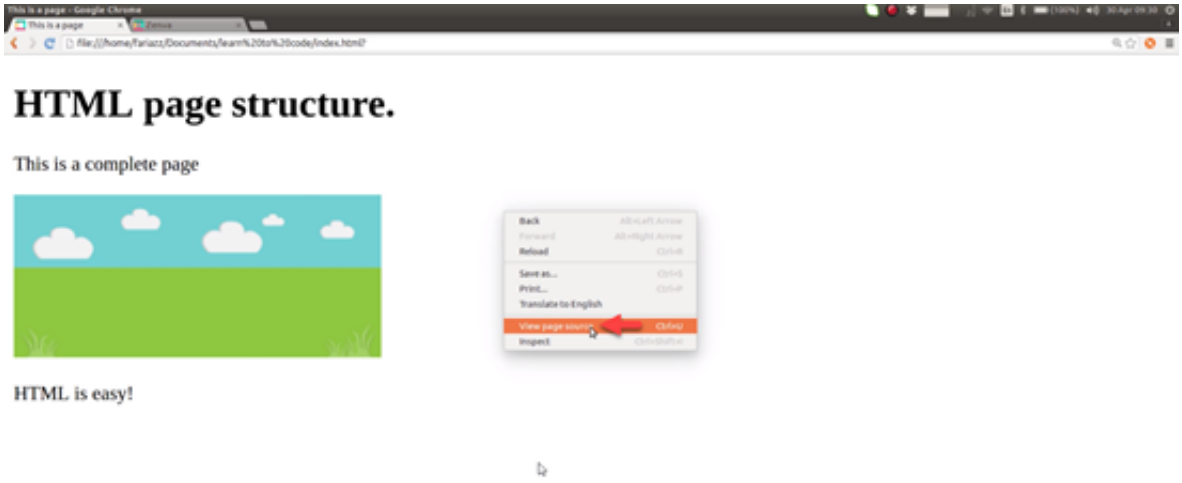
The head section has information about the page. Such as the title, the character set, if you want to have international characters. There are other options as well like the description and you can specify a favicon.

Then the body is what you see in the actual content area of your browser.

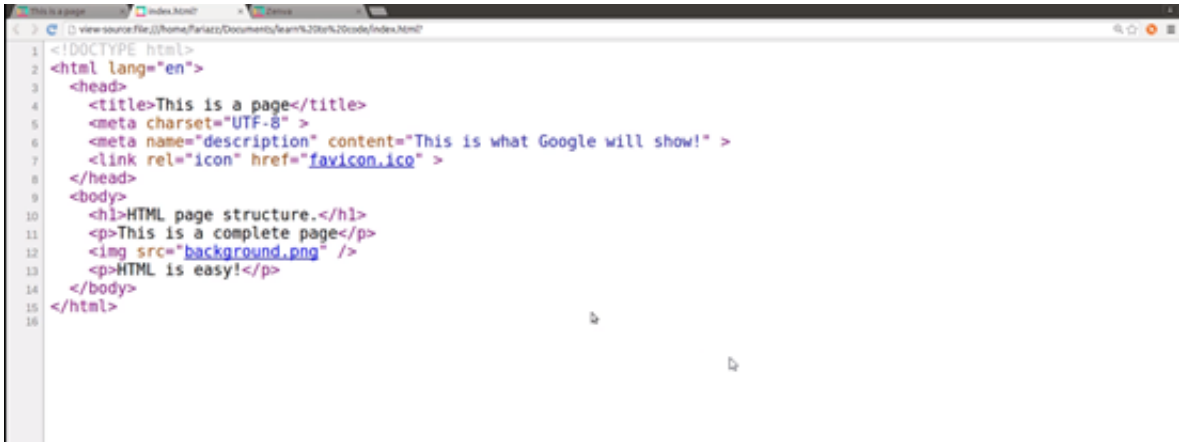
In this lesson you will learn how to look at the **source code** of any web page on the internet.

You can view the **source code** of a web page by using Google Chrome or any modern web browser. Most web browsers all have similar tools for this purpose.

So we have the previous example, see the screen shot below:



You can **right click>View page source**. Its that simple and then you will see the following window popup:



By doing this you will be able to see the **HTML code** of your page.

So you can go to any page on the internet, and you can do the same thing, so say if you went to zenva.com and viewed the source code:

This is also a fantastic way to learn. If you are **visiting a page** and you want to see **how that page is made**, you can look at it by **viewing the source code**.

Because, you haven't learned **CSS** yet there may be some weird stuff on the page, and there will be JavaScript code as well in the source code for the web page you view.

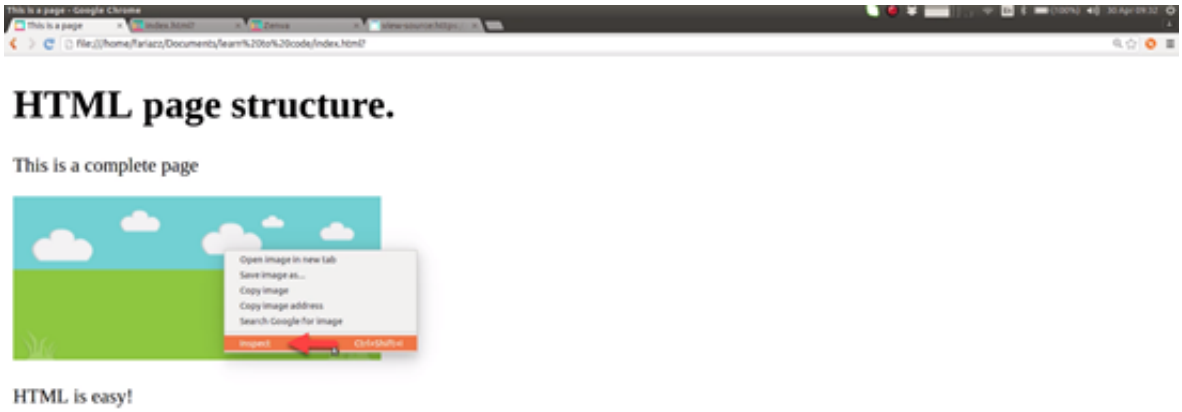
You will see a lot of familiar things, and this will give you a lot of confidence, but at the same time it can be overwhelming, but you will be learning and its all a process.

It's strongly recommended that you view the source code of a few pages and get used to their being things you may not understand.

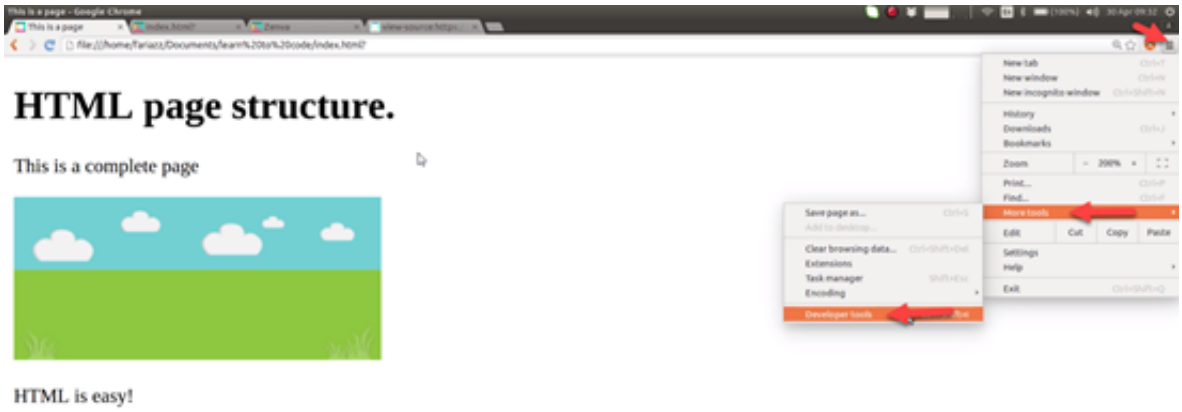
Try to find the ones that you do understand and then you can consolidate that knowledge.

But, this is not where this lesson ends. There is actually an even better tool! It is a part of Google Chrome, and its the ability to **inspect elements**.

There are different ways to reach this section. One of the ways is to again **right click>Inspect.**

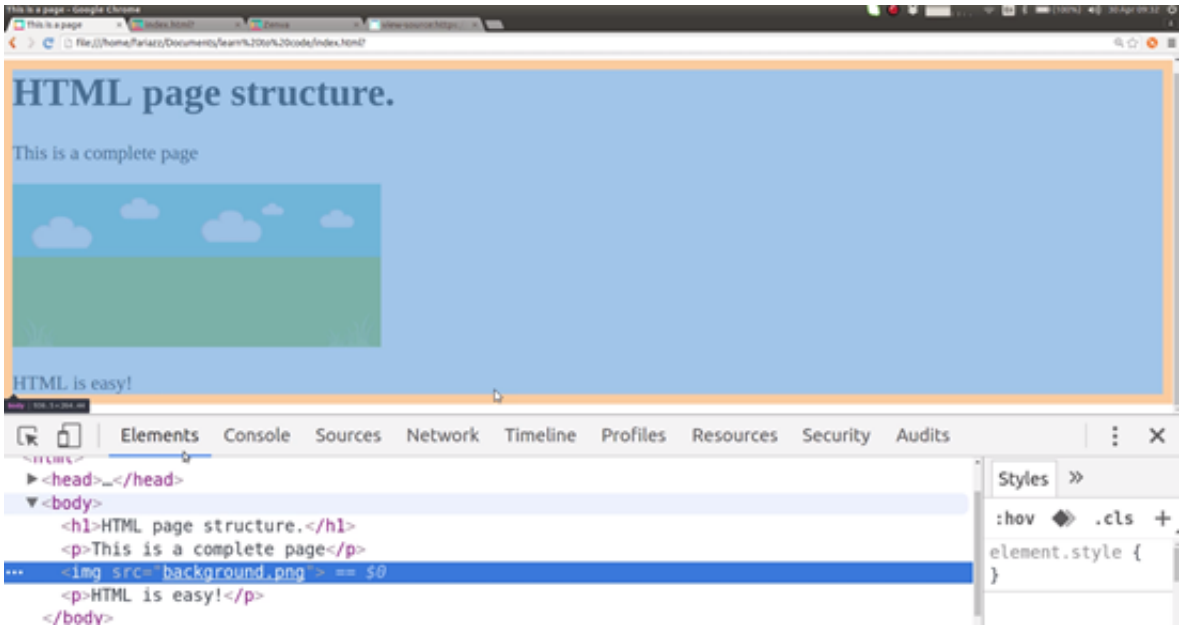


The other way is to click on the **Options tab>More Tools>Developer tools**.

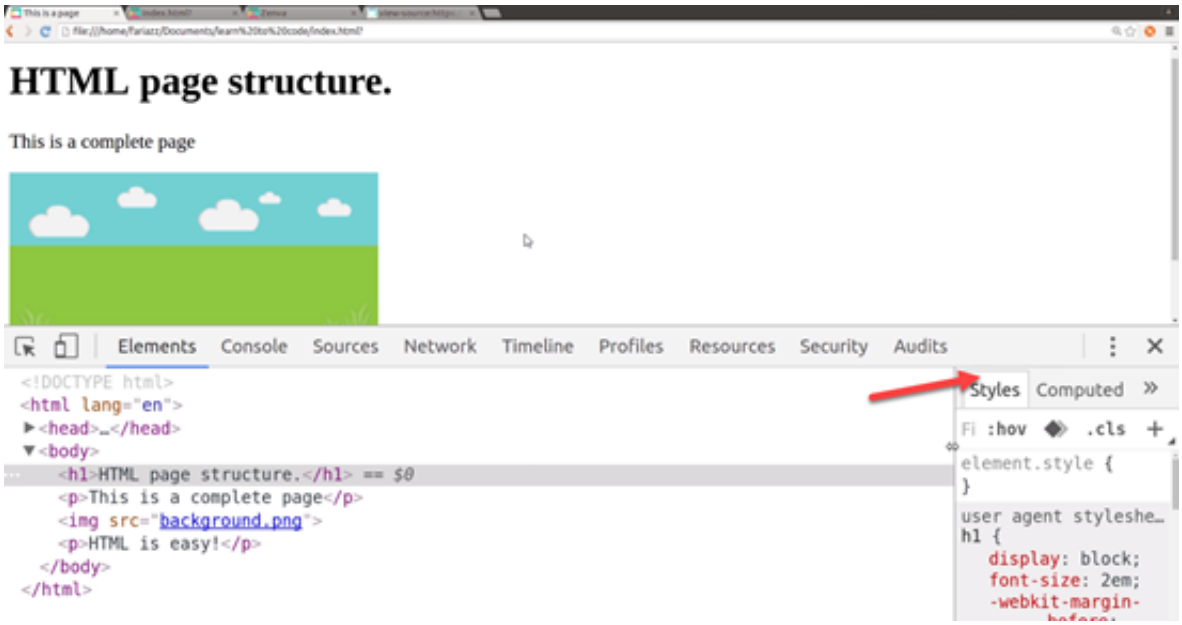


Any of these methods will take you to the same place, which opens up this **“Magical Area”** beneath.

You do not have to worry about all the sections. The one that you should be looking at now is the **Elements section**. If you select a line of code you will be shown the different sections from the web page. This is really useful because you can actually see how they did a particular part.

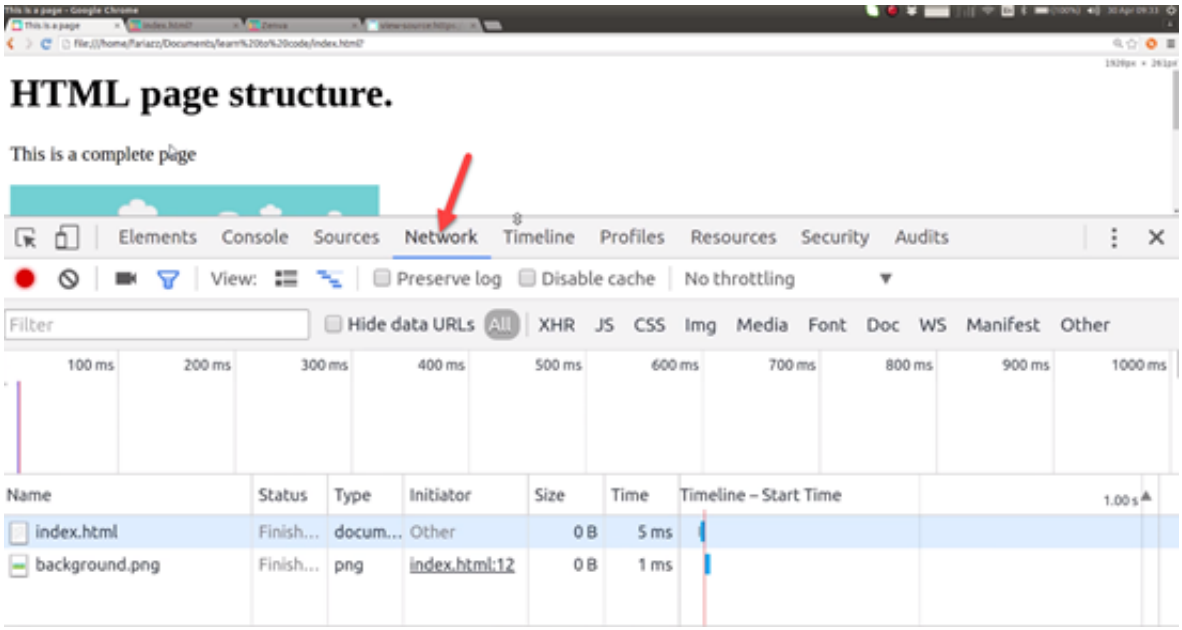


The **styles section** in the lower right hand corner **shows CSS**.



We will cover this area once we get into CSS in this course.

The other section that is interesting is the **Network section**.



This will **show you all the files that were loaded**. You can see the type of file and the size of the file here in this section. The time it took to load is also displayed.

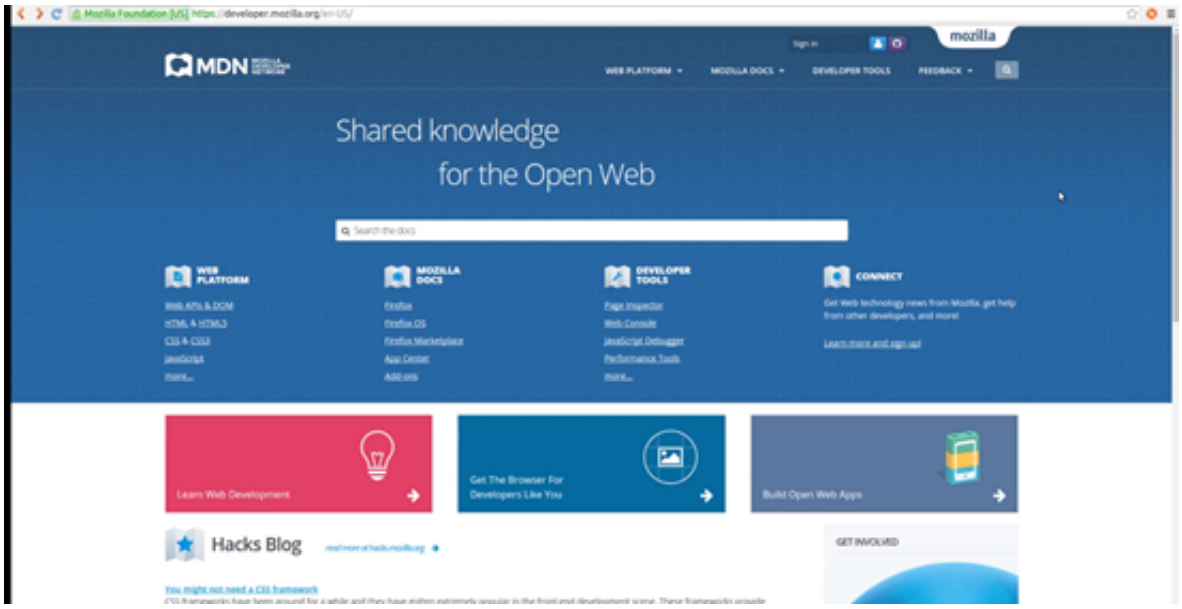
Be sure to open up and start using the Chrome developer tools or use the other tools that come with other web browsers that you may be using.

Recommended Websites to Support your Learning

There are some **recommended** websites that can be used to support your learning process and **learn** a lot more about **HTML**.

Mozilla Developer Network

The first recommended website is the **Mozilla Developer Network**.



You can visit this site here: https://developer.mozilla.org/en-us/

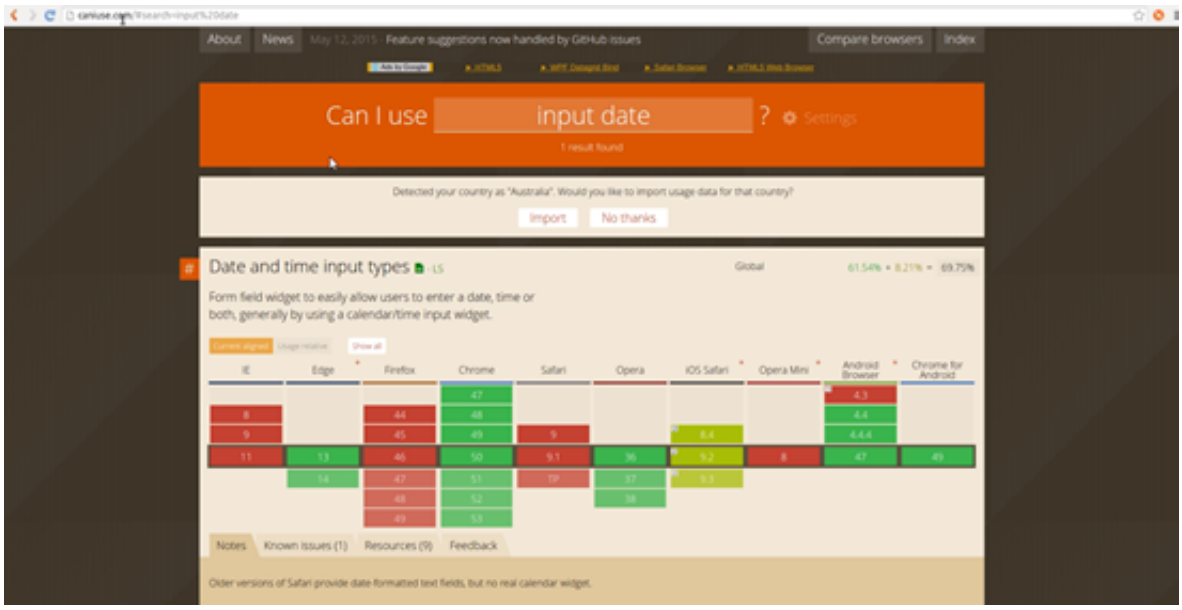
The direct link: [Mozilla Developer Network]

This website is **maintained** by the **Mozilla Foundation** and you can find **information** about **HTML tags**, and what there attributes are. The attributes that developers should no longer use because they are being **deprecated**.

This website is also great for finding **standard compliant** information.

Can I Use

The second recommended website to use is called **Can I use...Support tables for HTML5, CSS3, etc**



You can visit this site here: https://caniuse.com/

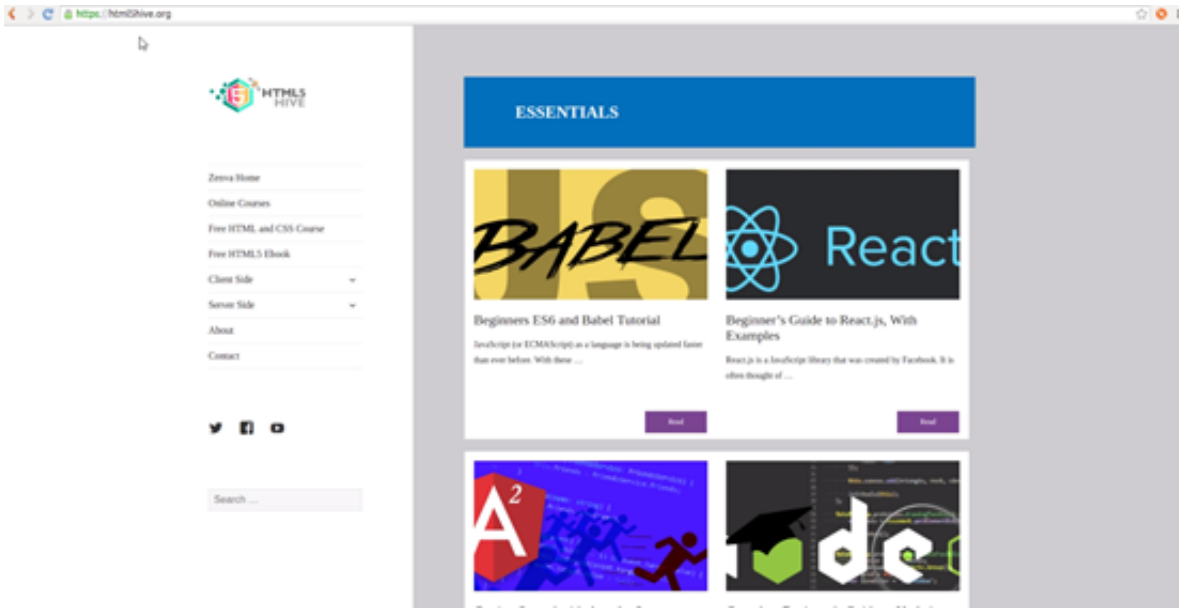
The direct link: [Can I Use]

This website will let you **search** for any **HTML tag** and it will show you which browser supports this technology and which ones are in the process of supporting it or which ones that do not support it. So, before using any cutting-edge feature of the HTML language make sure you consult caniuse.com to see if its supported or not.

HTML5 Hive

The third recommended website to use is **HTML 5 Hive**, which is also on the Zenva Blog. This website contains a lot of **free tutorials** on web development, learning materials on HTML5 and JavaScript. On this website you can also find a free e-book, which covers similar content of this course.

There is also a free online course on HTML and CSS on this site.



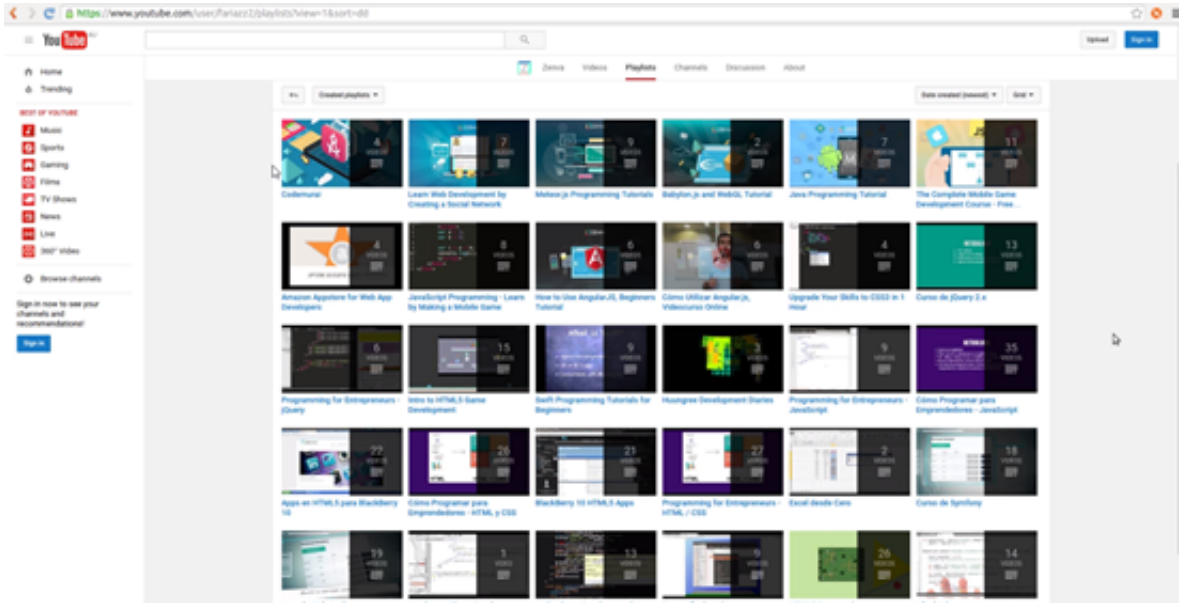
You can visit this site here: https://html5hive.org/

The direct link: HTML5 Hive

Zenva Youtube Channel

The fourth recommended web site to use is the **Zenva youtube channel**.

This channel contains a lot of videos that you can watch for free. If you go to the channel and the Playlists section you can see all the videos listed sorted by topic. There are videos on HTML, JavaScript, videos on all sorts of web technologies, and even mobile app development.



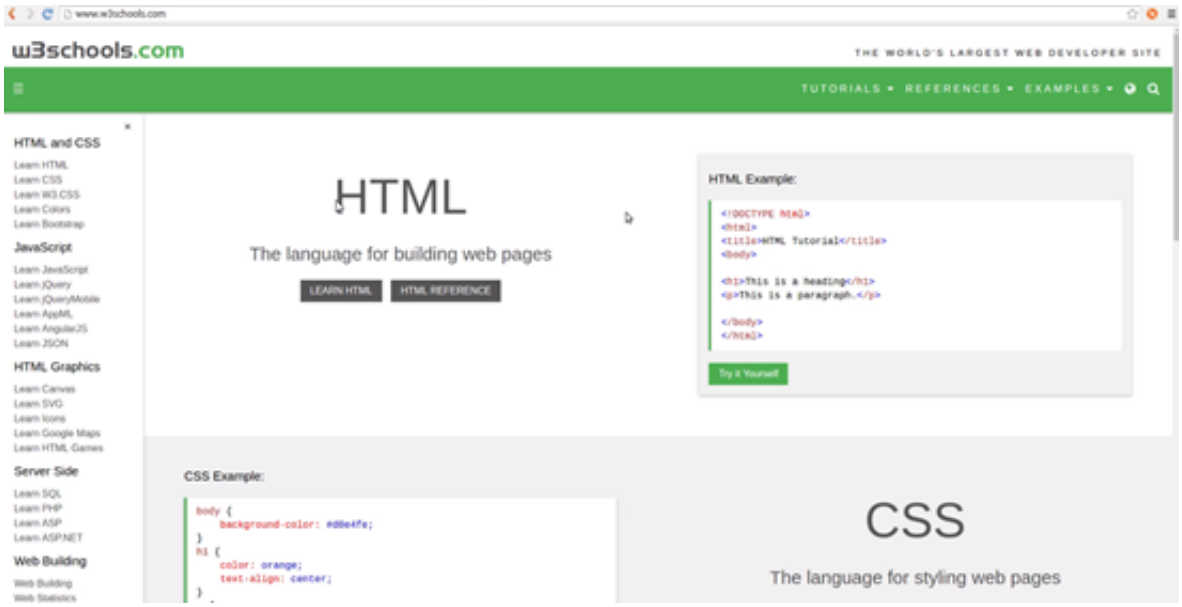
You can visit the Zenva YouTube channel here: https://www.youtube.com/user/fariazz2/playlists

The direct link: Zenva YouTube Channel

W3Schools

The fifth recommended website to use is the **w3schools.com**. This website is to be taken with a little bit of care because it doesn't have very good explanations for beginnings and they are not always standard compliant. This is why the Mozilla Developer Network was recommended first, before this site. Mozilla Developer Network enforces and teaches developers the correct way to do things.

This website contains a lot of good **quick explanations** for things, but just don't forget that there might be things that are **not fully updated** or that are not fully standard compliant on w3schools.com.



You can visit this site here: https://www.w3schools.com/

The direct link: [W3schools]

