**Learning HTML, CSS and Bootstrap-4 in an hour**

**By- Girish Mehta**

**(February 5th, 2018)**

Introduction:

HTML, CSS and Grid 960 are our kind of requirements for developing good and beautiful websites.

Today I’ll be walking you through all three of them so that after an hour you are good to go and start building the website that you think of.

So below is a snippet of code that I’ll make sure that by the end of document you will be able to understand and write the same very easily.

<!DOCTYPE HTML**>**

**<html>**

**<head>**

**<title>**Welcome**</title>**

**<meta** charset=”utf-8”**/>**

**<link** rel=”stylesheet” type=”text/css” src=”css/vendor/960.css”**/>**

**<link** rel=”stylesheet” type=”text/css” src=”css/style.css”**/>**

**</head>**

**<body>**

**<div** class=”container\_12”**>**

**<div** class=”grid\_6”**>**

**<h1>**Welcome to first page**</h1>**

**</div>**

**<div** class=”grid\_6”**>**

**<h1>**Today we’ll learn HTML, CSS & Grid 960**</h1>**

**</div>**

**</div>**

**</body>**

**</html>**

Now, don’t worry if you do not understand what this code means, scroll down and bits by bits everything will make sense.

Note:-

If you already understand the above code then this document is not for you.

We’ll be talking about HTML5 & CSS3, you may look up online for the different syntaxes in older versions.

# Prerequisite:

Browser

An editor like – Notepad, Atom, Sublime, Visual Studio Code

### Structure:

Before we dive into anything, you need to know that we’ll be working with 3 different things- HTML, CSS & Grid 960 and so you should also know where what is to be kept and why? Like when you buy a lot of books, you start keeping them in bookshelf and probable in an arranged order by either category or your liking or name of height or any other property. Similarly, when you are going to be coding for your website, the person you receives your code-the browser, also would like to see files in some sorted format that is easy for it to understand like the bookshelf is for you. So, we put our files in the folders like :-

css – for all .css files

js – for all javascript files

img – for all images

vendor – for all files that are created or owned by someone else but utilized by us

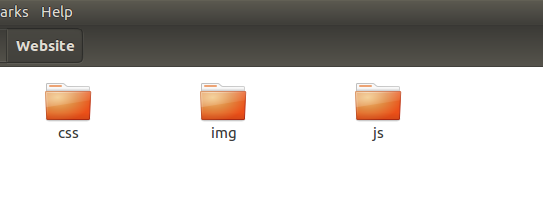
\*I’ll explain later what these files are, for now remember these places where they are kept.

There no hard and fast rule that you need to follow the above way of structuring your files in folders but, it is a good practice to keep your files sorted like this so that it helps browsers, search engines and other technologies online to read and utilize your code efficiently.

There are many more folders that we create for different purposes but for now we’ll be going through the above ones. The 3 main things being, structure, presentation and behavior of a website.

Presentation we define in CSS and behavior in Javascript.

Now that you know we have 3 folders, your directory would look like something below:-



# HTML (Hyper Text Markup Language):

HTML as the full form says is a Hyper Text markup Language.

**Q.** Now what does this mean?

**A.** Like we humans need a language to interact with each other, to share thoughts and transfer information, browser also uses a language called HTML, that is written using English language in the form of text to talk to the browser so as to what you want.

**Q.** Is this only specific to browser?

**A.** No! But, since our websites open in browser so for now we only talk in reference to the browser.

So we type in Hyper Text Markup Language to interact with the browser and convey our needs but, how does the browser know that my document is speaking in HTML?

The first line that I wrote- <!DOCTYPE HTML>

Now this means, that I am telling my browser that I will be talking to you in HTML and so do not use any other language with me as I will not understand.

**Q.** Why we need this?

**A.** Imagine you are out in a country where there are people who speak multiple languages and not everyone knows every language. You may know English and French but the person you meet may know English and Russian and so before you start a conversation you come up with something like You - “Do you speak English?”

Friend - “Yes”

-----And the conversation begins-----

Similarly, our code interacts with browser and tells that my language is HTML and so the further processing becomes easy and fast.

**Q.** Do I have to always specify?

**A.** Now imagine if you started talking to an unknown Chinese person in English and he understood and responded. Here, you didn’t ask if he/she knows English right? But, you started talking because the other person understood that you know English and he/she has the capability to talk to you in that language and so responded. So, even if you do not specify this, it will work but, it’s always a good practice.

**Q.** Why only **<!DOCTYPE** HTML**>** and not **<HI** HTML**>**?

**A.** This is same as you asking why do I say “Breakfast” and not something like “Asapuri”. Like we have fixed words in English language that we use to communicate our thoughts similarly, we have fixed words in HTML that we use to convey our message. And so, DOCTYPE- meaning Document Type. In the sense, my type of document is HTML.

**Q.** Why “**<**” and “**>**”?

**A.** Well one document may have languages like I might use css and even javascript in my html document so how would the browser know which in which language is it to interpret this word. So, we use “<” in the beginning of our element and “>” to end our element. Meaning, that anything between “<” and “>” is a part of HTML and not some other language.

**Q.** Now what is element?

**A.** All the particular words that the HTML has to convey the messages are known as elements or tags. Like every word in English language, every element in HTML has a specific meaning, e.g. DOCTYPE to specify the type of document.

Now that you know how to tell the browser what language your code will be speaking, let’s dive into the language.

There are many elements in HTML for different purposes and by purpose I refer to only telling the structure of our webpage.

Note-

A webpage is just one page of the website where as a website has one or more than one pages.

We begin with **<html>**

- this tells that okay, from here on, everything below is to be understood in html unless I specify that I am using some other language. We specified the starting and so we also keep the courtesy to specify where I have ended talking in html and so we specify the end with </html>

**Q.** What is “/” in “**</html>**”

**A.** While anything between “**<**” and “**>**” means the html element, similarly, anything between “**</**” and “**>**” means that here my element ends

In **<html>**, out major components include:

**<head>**

**<body>**

To understand this, we’ll refer to our human body. Our human body starts from top of head to the bottom toe or the other way round however you like saying. Similarly, **<html>** ...**</html>** tell that this is my whole human body out of which **<head>**..**</head>** will be my head area and **<body>**..**</body>** will be everything below the head till toe. Now like our head and body come between our body start and end similarly, **<head>**...**</head>** & **<body>**...**</body>** will also come between **<html>**...**</html>** and like the head comes first when we look at human body from the top similarly, in the code also it will be like:

**<html>**

**<head>**

**</head>**

**<body>**

**</body>**

**</html>**

Now you understand why we have **<!Doctype>**, **<html>**, **<head>** and **<body>**. Apart from this, all other elements can be looked up upon at the link -<https://www.w3schools.com/tags/default.asp>

Moving ahead,

**Q.** Why do we require head?

**A.** This is like why do humans need a head or a brain, basically to provide the base for the body to work. Every information that your body needs, your brain, from the head, supplies to it. Similarly, the head has information that maybe required for the body to work or the browser or search engines may look for information that also the head can provide.

Q. Why do we require body?

A. So basically everything that we want the user to see we put in body.

Coming to the head part of the code that I showed in the beginning:

**1 <title>**Welcome**</title>**

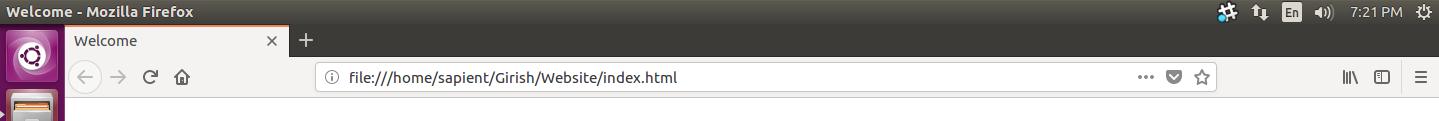
**2 <meta** charset=”utf-8”**/>**

**3 <link** rel=”stylesheet” type=”text/css” src=”css/vendor/960.css”**/>**

**4 <link** rel=”stylesheet” type=”text/css” src=”css/style.css”**/>**

Now lets understand this tag:

**<title>** - Like the name says, it represents the title of the something, which is our webpage(not website) . Let me show you where our title comes:



So, everything that we put inside **<title>** and **</title>** comes in the tab bar of our webpage. Also, it tells the search engines that this is the title of the webpage and thus the information is showed in the search results.

\*Note- To learn about how search engines work you may refer to google for it.

Next we have is **<meta>**

Meta basically represents meta data, which is data about the data. To explain this, consider you play a game where a friend tells you a name of movie and you have to act it out for your team to guess without telling the name of the movie. So to play this, we use the information like length of movie name, which actor played in it, actions, and all. So this information is actually data about the data that you have- the movie name. So data about the data is known as meta data and for short we have the tag **<meta>**.

Now what is “charset=”utf-8”?

We understand what tags are- that start with “**<**” right? So everything else between the tag starting and “**>**” are called attributes or properties. Attributes is basically a little more information about the information. Like we say that there is a table in our office. So the table is our element and its locations is office, it might have a height and also more properties like what type of table it is, what raw material is used to make it and many more such things. All these things are known as attributes.

Similarly, “charset” in **<meta>** tells the information about our data that tells the character set the browser may refer to for understanding our content. We here specify “UTF-8”. Like this, there are many more character sets like – ANSI, ASCII and more(refer to google for more). To understand what the use of charset set is let’s go back to our previous example of English language. We know that the English language has 26 letters and so when you write in English you use one of the 26 letters. But, what if I write a Russian letter in between and English sentence, you won’t understand write unless you know Russian. Similarly, ASCII has particular character, ANSI has some particular characters and UTF has some character that it can recognize. We use this charset when our content has multiple languages that we want to use to build our website that is understandable in different countries in their native language.

Now you must be wondering why I’ve written “**/>**” at the end of meta tag.

- To understand this you must know that there are different types of tags. Some that we tell where the starting is and where the ending is like we did **<head>**..**</head>** and some that do not need ending and hence are closed then and there. For such kind of tags we put “/” before “**>**”.

Examples of such tags are - **<meta>**, **<br>**

Next we have is **<link>**

This like the name specifies, is used to link to the outside of our current file. In this we give the relation of the file with our current file which is “stylesheet” and then then the type of file which is “text/css” and finally the source of our file kept outside.

Coming to the **<body>** tag. In this we put information that we want to display to the user. For this we have tags like <div> to create a division. To study about more tags you can refer to the link that I specified above but for now I’ll talk about a few tags in HTML 5.

**<video width=”320” height=”240” controls>**

**<source** src=”movie.mp4” type=”video/mp4”**>**

**<source** src=”movie.ogg” type=”video/ogg”**>**

The **controls** attribute adds video controls, like play, pause, and volume.

It is a good idea to always include **width** and **height** attributes. If height and width are not set, the page might flicker while the video loads.

The **<source>** element allows you to specify alternative video files which the browser may choose from. The browser will use the first recognized format.

The text between the **<video>** and **</video>** tags will only be displayed in browsers that do not support the **<video>** element.

To have a more detailed understanding of any tag, you can refer to<https://www.w3schools.com/html/html5_intro.asp>

Now that you understand what HTML is and how it is used, let us dive into CSS. But, before that you need to understand what is an “id” and a “class”.

### 

### Class:

To understand this, relate this to a normal classroom of your school. What happens when you say a student is of so and so class. That usually means that the student has some particular subjects to study, some teachers who teach, assignments and homework that they get and all this is same for all other students of the same class. Similarly, when we say that an element belongs to this “class”, we say that it will have the following properties that we define using CSS.

Id:

Like a class, an “class”, and “id” is also the mechanism to defining the properties using CSS but by an “id” we can also control the behavior/working of an element using Javascript.

**Q.** How do we write class and id?

**A.** For class we write as:

<body class=”styling”>

For id we write as:

<body id=”styling”>

# 

# CSS (Cascading Style Sheets):

We use this to give presentation to our website. By presentation I mean, how it looks, it’s color, it’s size, it’s background color, it’s position, it’s height, it’s space from other elements and all other properties.

There are many different properties in CSS and we’ll discuss some. But, first you need to understand what is a class and and id(refer from HTML). In CSS, we can create two things, a class or an id.



### Class:

A Class is created using a “.”[period] and followed by the class name. A class can have any name but two classes cannot have same names or else if will not work properly. We can define the properties and their values inside the class. It is defined using “property:value;”. Below is an example to understand.

**.heading {**

**color:** red**;**

**}**

Here we have created a class named heading and as the name suggests we’ll give this class to the elements in HTML where we have given headings to our paragraph or some text that will then have the color red. So it will be a red color heading.

It is a good practice to give the names of class that are understandable in such a way that it is understood for what purpose are they reason. Like we defined a heading class which is understood that I’ll give this class to some text which I want to make a heading. Like:

**<h1 class=”heading”>**Welcome to first page**</h1>**

### Id:

Like a class, we have an “id”. The only difference is that we use a “#” instead of “.” to define it. Example:

**#heading {**

**background-color:** black**;**

**color:** white**;**

**}**

Note:

We use “;” at the end of property:value to define another property. For single line CSS like “.heading” we do not require “;” at the end but, for multi-lines like “#heading” we require a “;” at the end.

Not only a class of an id is used in CSS but, we can also define the property of an element by its name. Like:

**.body {**

**background-color:** black**;**

**color:** white**;**

**}**

Now this by default will be applied to the body element of HTML and make the background color of our page as black and all the text color by default as white. Like this, we can just write the element name and start the block to define it’s CSS and this we do not need to give it a class or id but, keep in mind that this will give the following properties to all the elements in our webpage and thus to overwrite it we will need to give it a class or id. Here’s comes the reason why we call it a Cascading Style Sheet.

Let me explain you a little better. Consider the following code in HTML.

<!DOCTYPE HTML**>**

**<html>**

**<head>**

**<title>**CSS**</title>**

**<meta** charset=”utf-8”**/>**

**<link** rel=”stylesheet” type=”text/css” src=”css/vendor/960.css”**/>**

**<link** rel=”stylesheet” type=”text/css” src=”css/style.css”**/>**

**</head>**

**<body>**

**<h1>**First Heading**</h1>**

**<p** class=”color-black”**>**This is my first para**</p>**

**<h1>**Second Heading**</h1>**

**<p** class=”color-black”**>**This is my second para**</p>**

**</body>**

**</html>**

**body {**

**background-color:** black**;**

**color:** white**;**

**}**

**.color-body {**

**background-color:** white**;**

**color:** black**;**

**}**

**.color-blue {**

**background-color:** blue**;**

**color:** white**;**

**}**

Now what I am doing here is, first I’ve given a style sheet to body and hence defining my first default style that will be followed in my page. Second I am giving a style sheet to my paragraphs which will overwrite the default style sheet.

Hence my first para will not have a white color text as defined in body but a black color text with a white background as defined in class “color-black”. And same will happen in case of my second paragraph.

This is how we cascade style sheets. The order of writing the class or id names is very important because that is how the effect will be followed.

We can also define property to a particular element like suppose there is a list and I wish to apply a property to the list elements I can do it like:

**.ul li {**

**color:** red**;**

**}**

Now this will make my text color as red to the list elements. We can also use comma separated tags to apply a common property to different elements. Like:

**h1,**

**h2,**

**h3,**

**h4,**

**h5,**

**h6 {**

**color:** red**;**

**}**

Now this will give a red color effect to all the text which have element as either h1 or h2 or h3 or h4 or h5 or h6.

To get familiarized with all the properties you can refer to -<https://www.w3schools.com/cssref/default.asp>

This same CSS can also be applied inside HTML as inline CSS or internal CSS as:

### Inline CSS:

**<p** style=”color:black”**>**This is my first para**</p>**

We define inline CSS within the element area using the “style” attribute and then the css syntax.

### Internal CSS:

**<html>**

**<head>**

**<title>**Welcome**</title>**

**<meta** charset=”utf-8”**/>**

**<style** type=”text/css”**>**

**body {**

**backgound-color:** black**;**

**color:** white**;**

**}**

**</style>**

**</head>**

**<body>**

**<h1>**Welcome to first page**</h1>**

**</body>**

**</html>**

In inline CSS we write the css code inside **<head>** using **<style>** tag

### External CSS:

**<html>**

**<head>**

**<title>**Welcome**</title>**

**<meta** charset=”utf-8”**/>**

**<link** rel=”stylesheet” type=”text/css” src=”css/style.css”**/>**

**</head>**

**<body>**

**<h1>**Welcome to first page**</h1>**

**</body>**

**</html>**

For external CSS we use **<link>** tag to insert the source location of our css file.

# Bootstrap:

Bootstrap is a framework that we use to make our sites more beautiful. There are many frameworks out there and one of them is Bootstrap. In bootstrap we have classes that we can use to make elements like we want.

There are things like if you want spacing in an element you can use spacing utilities that apply to all breakpoints, from xs to xl, have no breakpoint abbreviation in them. This is because those classes are applied from min-width: 0 and up, and thus are not bound by a media query. The remaining breakpoints, however, do include a breakpoint abbreviation.

The classes are named using the format {property}{sides}-{size} for xs and {property}{sides}-{breakpoint}-{size} for sm, md, lg, and xl.

Where property is one of:

* m - for classes that set margin
* p - for classes that set padding

Where sides is one of:

* t - for classes that set margin-top or padding-top
* b - for classes that set margin-bottom or padding-bottom
* l - for classes that set margin-left or padding-left
* r - for classes that set margin-right or padding-right
* x - for classes that set both \*-left and \*-right
* y - for classes that set both \*-top and \*-bottom
* blank - for classes that set a margin or padding on all 4 sides of the element

Where size is one of:

* 0 - for classes that eliminate the margin or padding by setting it to 0
* 1 - (by default) for classes that set the margin or padding to $spacer \* .25
* 2 - (by default) for classes that set the margin or padding to $spacer \* .5
* 3 - (by default) for classes that set the margin or padding to $spacer
* 4 - (by default) for classes that set the margin or padding to $spacer \* 1.5
* 5 - (by default) for classes that set the margin or padding to $spacer \* 3
* auto - for classes that set the margin to auto

(You can add more sizes by adding entries to the $spacers Sass map variable.)

Also there is something known as the [media object](http://www.stubbornella.org/content/2010/06/25/the-media-object-saves-hundreds-of-lines-of-code/) helps build complex and repetitive components where some media is positioned alongside content that doesn’t wrap around said media. Plus, it does this with only two required classes thanks to flexbox.

It becomes easy to make a website by working with Bootstrap as many things that can be done shortly by using a css instead of coding everything. For example if I want a slideshow of image I can use the carousel. It is is a slideshow for cycling through a series of content, built with CSS 3D transforms and a bit of JavaScript. It works with a series of images, text, or custom markup. It also includes support for previous/next controls and indicators.

You can find out about all the elements here -<https://getbootstrap.com/docs/4.0/components/>