

## MOOC based seminar report On

**HTML5 AND CSS FUNDAMENTALS**

#### BY :AMAN RAUTELA ROLL NO: 1961011 COURSE: B.TECH BRANCH : CSE

UNDER THE GUIDANCE OF DR. NAVEEN TEWARI ASSISTANT PROFESSOR DURING THE YEAR 2020-2021



**ACKNOWLEDGEMENT**

I would like to give a special thanks to several people who made this all possible.

It is because of their tremendous support that I was able to finish this project with immense success .

I would like to thank each and everyone who helped me in the completion of this project, but firstly and fore mostly I would like to pour my heart out in gratitude and humility towards our MOOC coordinator Dr. Naveen Tewari who ostensibly supported me in this project whole heartedly.

I consider myself very fortunate and gratified that I was given an opportunity and once again would like to show my appreciation whole heartedly to everyone to make this possible.

[rautelaaman7409@gmail.com](mailto:rautelaaman7409@gmail.com) **AMAN RAUTELA**



## BHIMTAL CAMPUS

THIS IS TO CERTIFY THAT MR **AMAN RAUTELA** HAS SATISFACTORILY PRESENTED

MOOC BASED SEMINAR. THE COURSE OF THE MOOC REGISTRATION **HTML5 and**

**CSS Fundamentals** IN PARTIAL FULFILMENT OF THE SEMINAR PRESENTATION

REQUIREMENT IN **3rd**\_ SEMESTER OF B.TECH ( ✓) / M.TECH ( ) / BCA / MCA / BBA /

MBA DEGREE COURSE PRESCRIBED BY GRAPHIC ERA HILL UNIVERSITY BHIMTAL CAMPUS DURING THE YEAR **2020-21**

#### MOOC – Coordinator

Name : Dr. **NAVEEN TEWARI**

#### Signature :

HOD

Name : **DR. M. C. LOHANI**

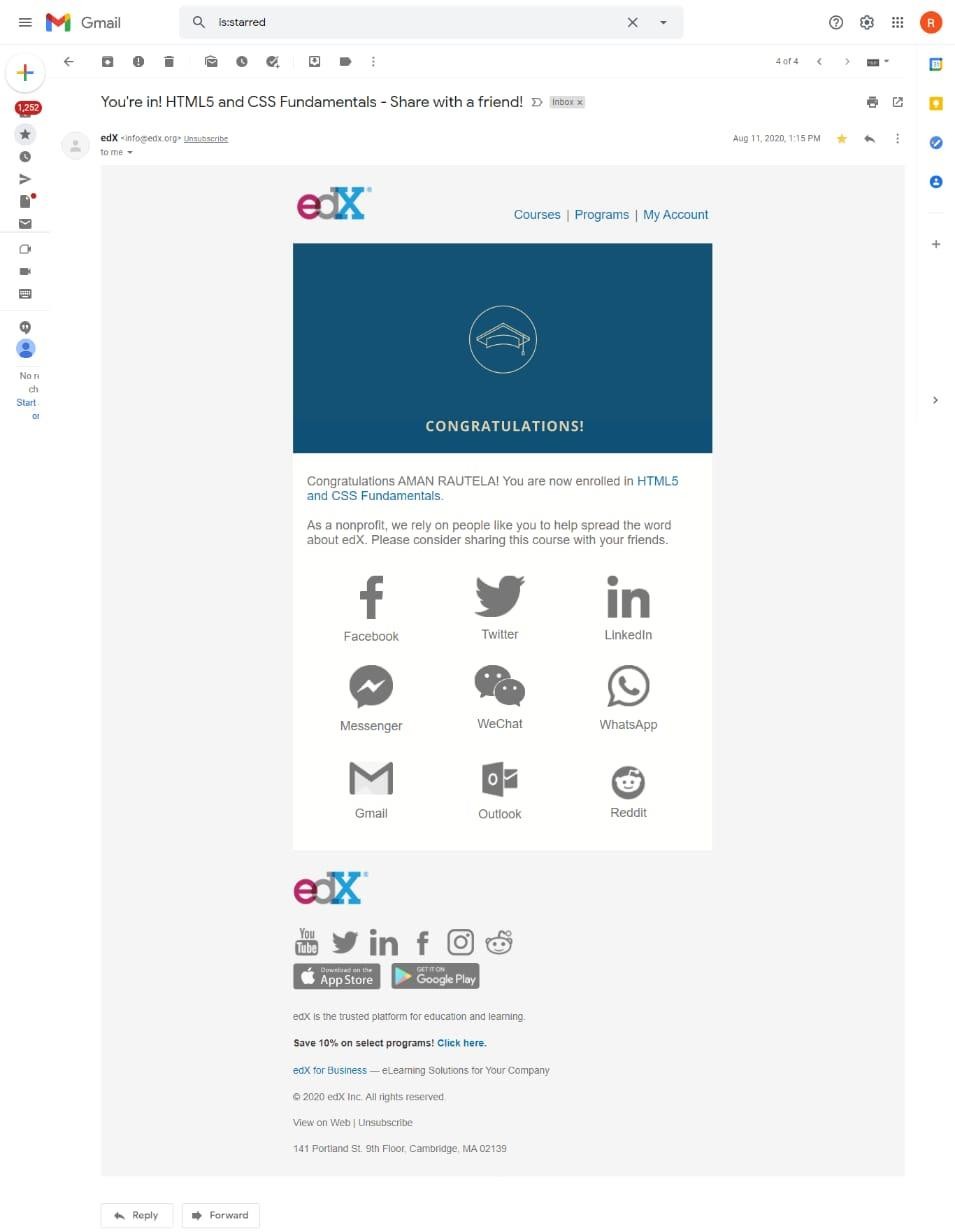
#### Signature :

**ABOUT THIS COURSE :**

This course is all about the basic understanding about how to build a Web Site using HTML 5 and CSS. I have enrolled in this course from a well renowned online learning plateform Edx.

Enrollment date : 11-Aug-2020

Time Duration : 8 Week (approx. 4 to 5 hrs per week)



**SYLLABUS :**

# This course is subdivided into 6 module as provided below:

**Module 1**: Web page

# The big three: HTML5, CSS and JavaScript Elements,

tags and attributes Character encoding Best practices

**Module 2** : Debugging Attributes, images and links Attributes Semantic meaning Images Hyperlinks

**Module 3 :** Adding style with CSS

# CSS basic syntax CSS properties Lists and selectors

**Module 4:** Fixing and debugging

# Debugging tools Debugging and the CSS box model Debugging CSS precedence

**Module 5:** More HTML5 and CSS

# Tables Multimedia Embedding content CSS tricks

**Module 6:** Basics of page layout

# Concepts Flexbox Recipe project Where to from here

## HTML5 and CSS Fundamentals

**WEEK 1**

#### HTML5

When people say 'HTML5', they usually mean a bit more than just the 5th version of the "Hyper Text Markup Language". Modern Web pages and Web applications are generally composed of at least three components, so what people often mean when they say 'HTML5' is the trio of languages:

HTML5, CSS3 and JavaScript.

The 'HTML' part contains all the content, organized into a logical structure. This is the part that an author might be most concerned with: the words, chapter headings, figures, diagrams, etc.

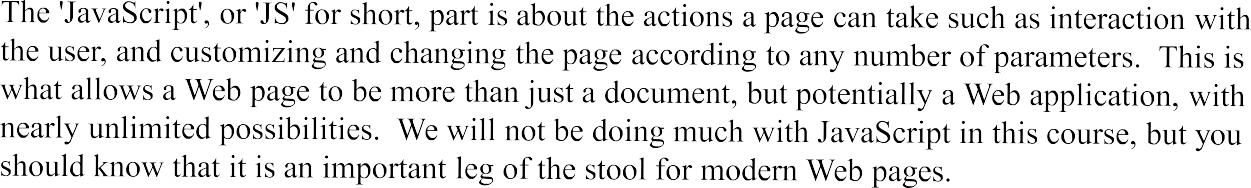
While there have been numerous versions of HTML since its inception, our focus in this course is the most recent version, HTML5. HTML5 was developed to provide more powerful and flexible ways for developers to create dynamic Web pages.

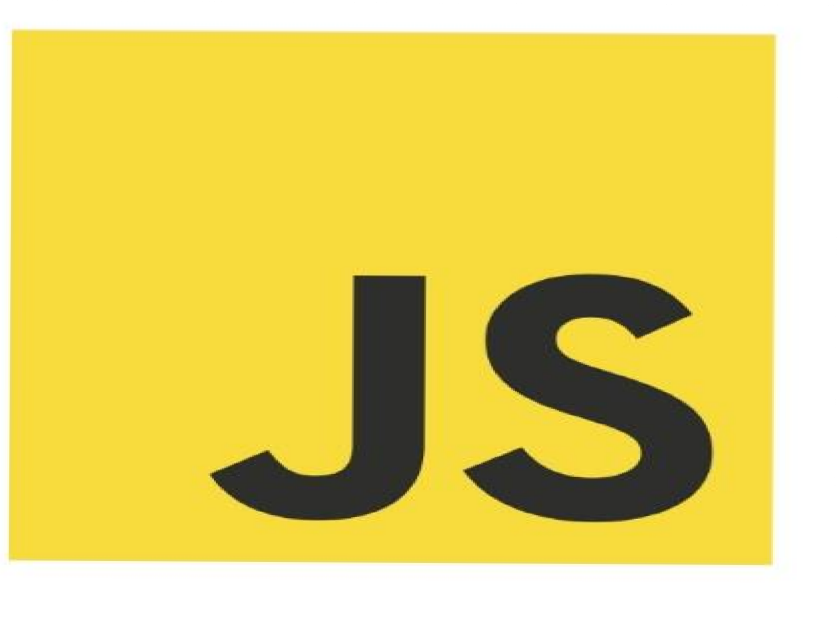
CSS :

The 'CSS' part (version 3 being current) is all about the presentation or style of the page; what it looks like without too much regard for the specific content. We'll be going into more detail on that later in this

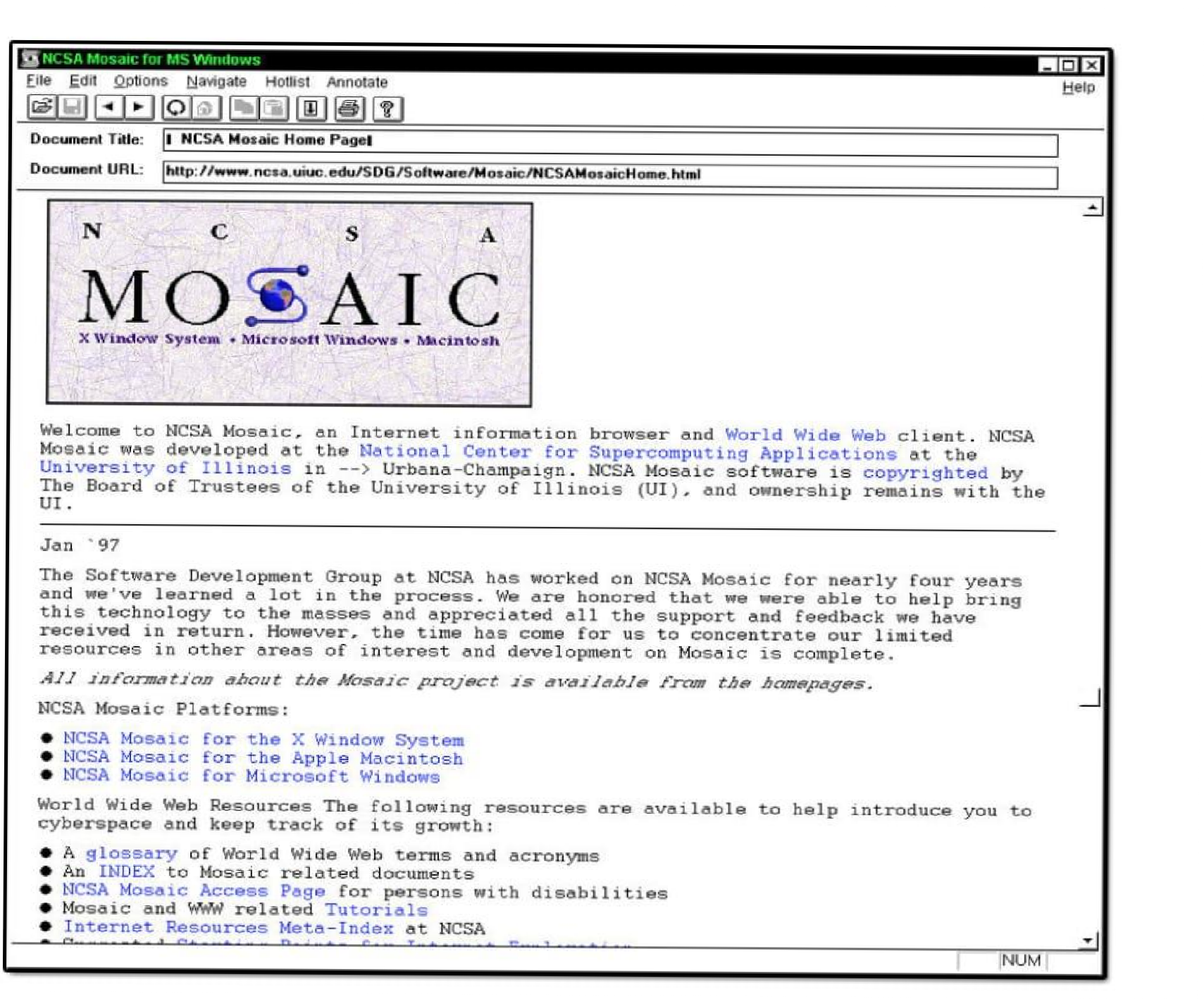
course, but for now, think of it as the way you might specify a "theme" in a word processing document, setting fonts, sizes, indentations and whatever else may apply to what it looks like.





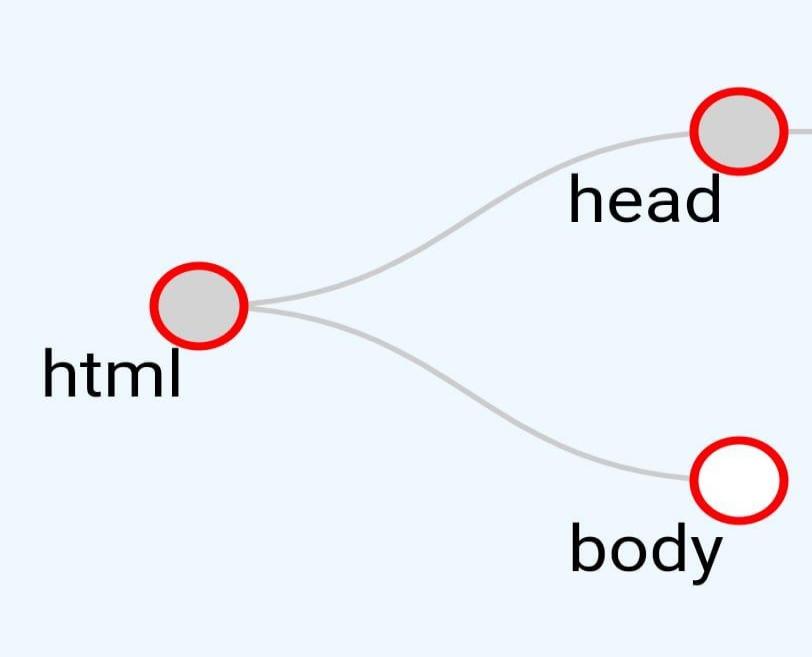
BROWSER :

The Internet existed long before the Web came to fruition, and lots of organizations were connected to it, including schools, companies and government organizations. As things progressed through the 80's, the Internet was used for file transfers, newsgroups (a kind of open forum), email and other conveniences.



ELEMEMTS, TAGS and ATTRIBUTES

If you are sitting at a coffee shop next to a table of Web developers, you will probably hear three words quite a bit: 'Tags', 'Attributes' and 'Elements' (or sometimes 'DOM elements', same thing just more precise and wordy).

'Elements' are the pieces themselves, i.e. a paragraph is an element, or a header is an element, even the body is an element. Most elements can contain other elements, as the body element would contain header elements, paragraph elements, in fact pretty much all of the visible elements of the DOM.

Consider the figure above.

It contains a single 'html' element. It turns out this includes within it the entire content of your html file. If you click on the "html" node, you'll find that it contains two components, a head and a body.

Clicking on each of those will reveal their respective contents. This structure is what we computer scientists call a "tree". Any given element (except for the outermost

'html' element) is wholly contained inside another element, referred to as the "parent" element. Not surprisingly, the elements that a given element contains are its "child" elements. And, yes, children of a common parent are often referred to as "siblings".

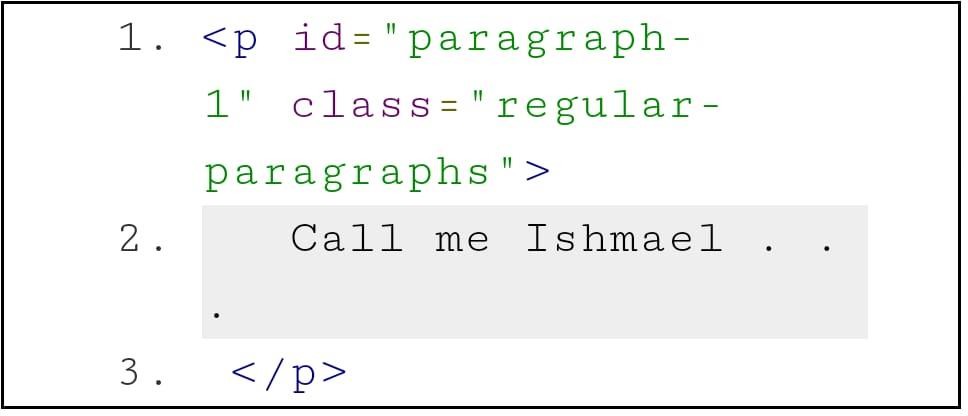
Thus in the example above, the top element is the html element, which contains just two elements, the head and body. The head element contains a title element and the body contains an 'h1' element and a 'p' element. In a more typical example, the body would contain many more children, but for our purpose this is enough.

That may be a great picture, but how do we represent such a structure in a text file? Well, that's where "tags" come in.

ATTRIBUTES :

Most of what we'll cover about attributes will come later, but I wanted to introduce the idea briefly. Basically, a given element on your Web page can be distinguished by any number of unique or common attributes. You can identify it uniquely with an 'id' attribute, or group it with a class of other elements by setting the 'class' attribute.

Attributes in HTML are written inside the opening tag like this:

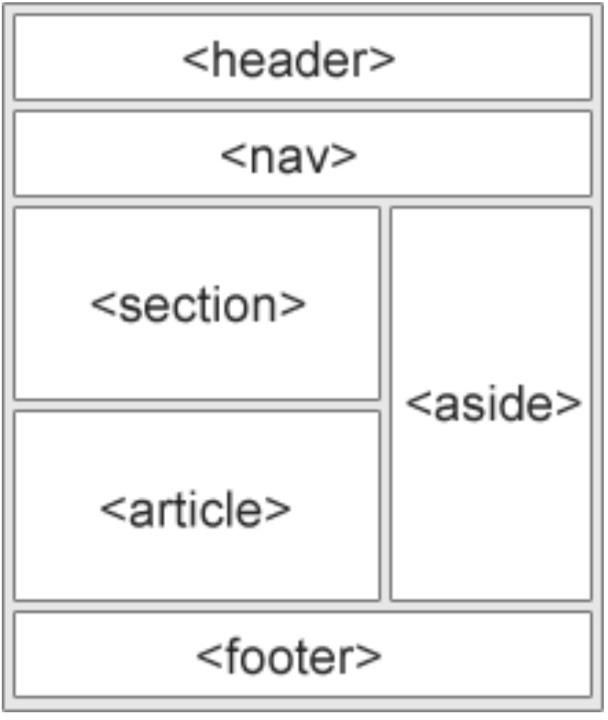
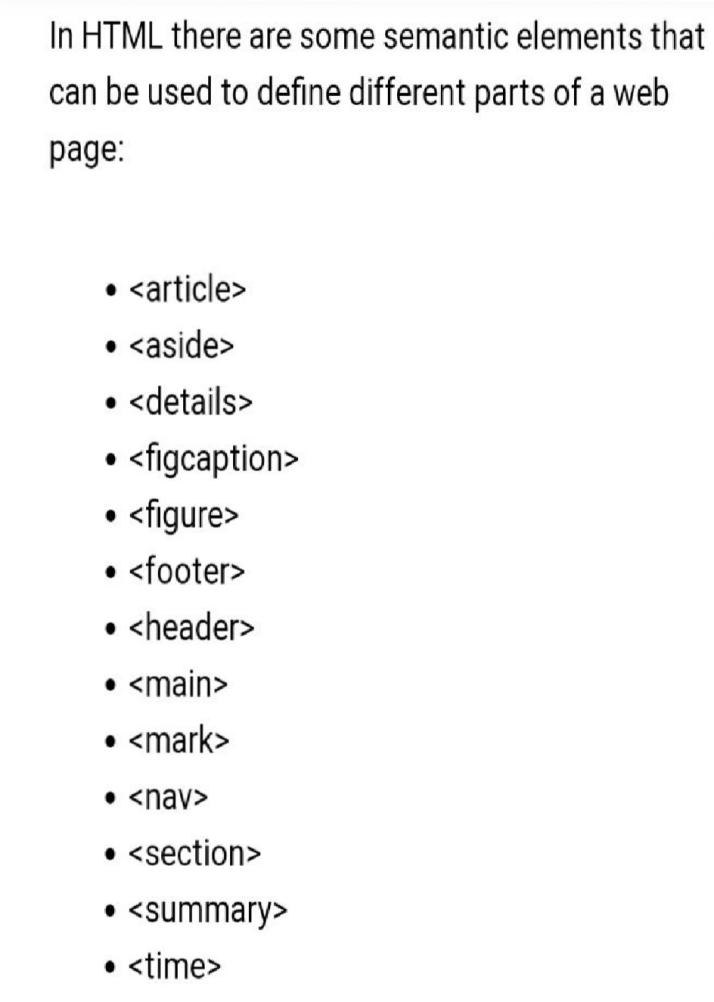


###### SEMANTIC ELEMENT :

**WEEK 2**

Elements such as <header>, <footer> and <article> are all considered semantic because they accurately describe the purpose of the element and the type of content that is inside them.

A semantic element clearly describes its meaning to both the browser and the developer. Examples of semantic elements: <form>, <table>, and <article> - Clearly defines its content.



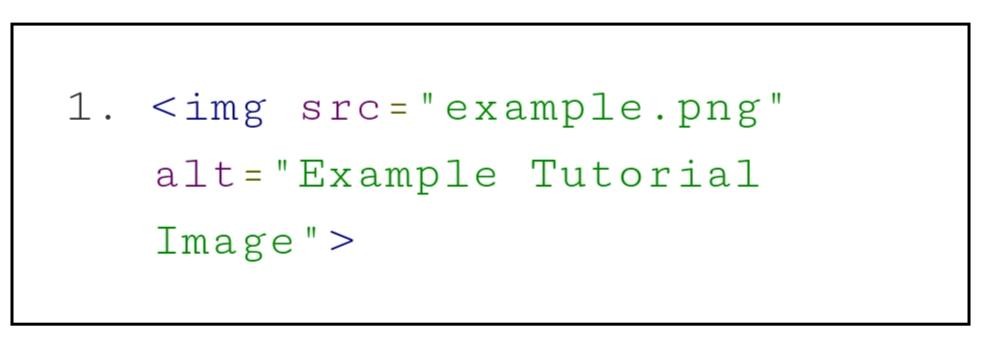
#### WHY IT IS IMPORTANT????

HTML5's semantic elements help structure the code we create, making it more readable and easier to maintain. They help us think about the structure of our dynamic data, and to choose titles' hierarchy properly. They help us differentiate the semantic elements of our markup from the ones we use solely for layout.

#### INTRODUCTION TO IMAGES :

The <img> tag :

In this age of visual culture, what is a Web page without images? Boring! Pictures and images make everything more interesting and engaging.

Here is the most basic <img> tag:

The image tag has several attributes out of which only src and alt are required. The rest are useful but optional attributes.

The source attribute from the <img> tag tells us where to fetch the image from. There are two different types of URLs you can give for source.



## WEEK 3

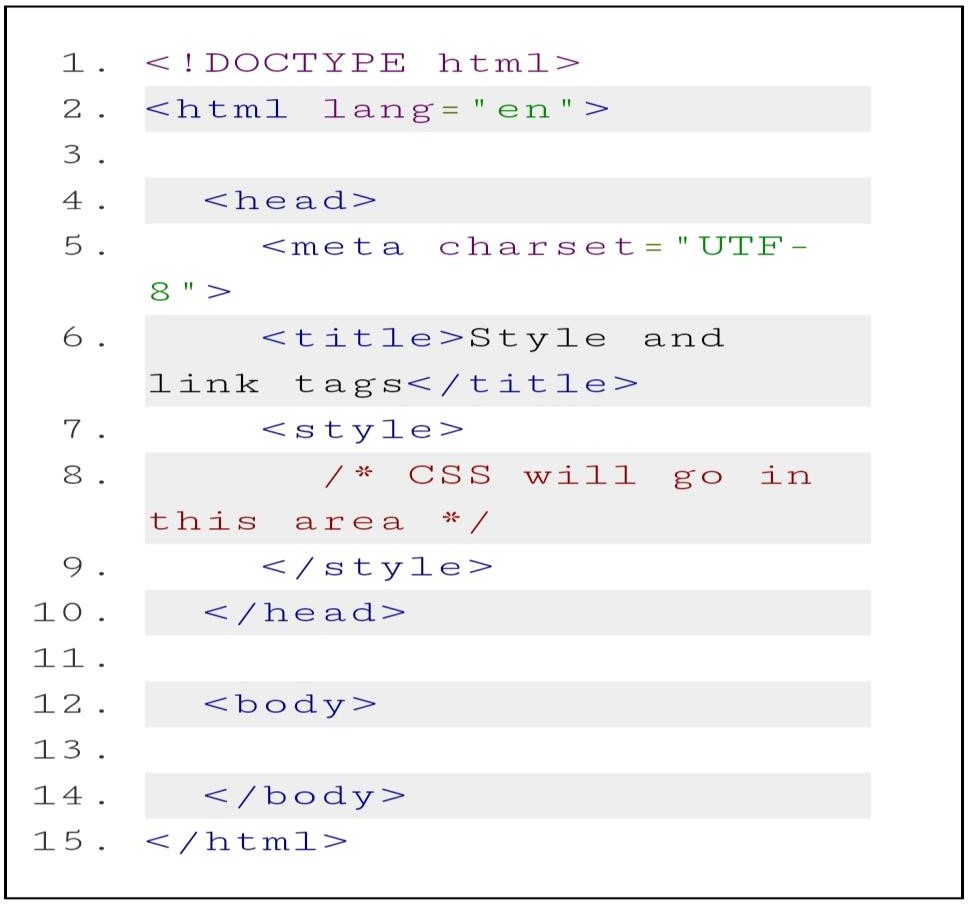
### CSS BASIC SYNTAX :

The <style> tag :

The best practice when working with CSS is to keep it in an external file using the <link> tag, however, when starting, it is simpler to merely place it directly into the document under edit.

To place CSS directly into an HTML document, we use the <style> tag. This tag can appear anywhere in an HTML document, however, the most common practice is to place it in the

<head> section. Such as:



The <link> tag :

While <style> is convenient, the better practice is to put the CSS into a separate file. One of the key advantages of using a separate file is that the CSS styles can easily be re-used between your different .html pages. Many authors further divide their CSS up into different files (for example: one for text styles and another one for layout).

Simply put your CSS into a separate file. This file does not need any HTML markup (i.e., no

<sty1e> tag required). Use the .css file extension and use a <1ink> tag to bind it in. The blink> tag must appear in the <head> section. By convention, css files are kept in a directory named css.

Use this <1ink> as a template:



**CSS PROPERTIES** :

There are hundreds of CSS properties for you to use. The complete list is available on the W3C Web site (or also, see the CSS reference page on the MDN Web site).

Below we've gathered a more manageable list of the most useful and common CSS properties: font-size, line-height, text-align, text-decoration, font-weight, font-style and font-family9.

Font Size :

font-size can be used to size the text of a tag. The value for the font-size has two parts: a number and a unit. Some of the most common units are: px, em, %, vh. For example:

p { font-size: 18px; } q { font-size: .8em; }

blockquote { font-size: 10vh; } These units are discussed below.

Additionally, font-size supports a more readable set of values that many authors prefer: xx-small, x-small, small, medium, large, x-large, xx-large

and relative sizing (relative to the text of the parent): larger, smaller. For example: p { font-size: medium; }

q { font-size: small; } blockquote { font-size: larger; }

LINE HEIGHTS :

Whereas font-size may drive the size of the text itself, the line-height property drives the height of the space it is drawn into. A large line-height will give the text more spacing. A small line- height will smash the text lines together.

For example, all of the Middlemarch text below has font-size:16px; But on the left, we see line-height:0.5; and on the right, line-height:3;



The used value is this unitless <number> multiplied by the element's font size. The computed value is the same as the specified <number>. In most cases this is the preferred way to set line- height with no unexpected results in case of inheritance. Read more on the MDN Web site.

TEXT ALIGNMENT :

Anyone familiar with a text editor will be familiar with this property. It can be used to align the text left, center or right. There are additional possible values like justify and justify-all . It usually defaults to left. However, remember that you shouldn't use text-align unnecessarily.

Note that text-align may not work as expected if applied to elements that are the same width as their text, or whose width is determined by the text within them (i.e., inline elements). The tags

<span>, <a>, <i>, <b>, <q> and others are considered "inline" because they do not receive their own new line when used. And text-align is often not useful on these tags.

But it is useful on block level text tags, such as <p>, <li>, <ul>, <ol>, <div>, and <blockquote> p { text-align: left; }

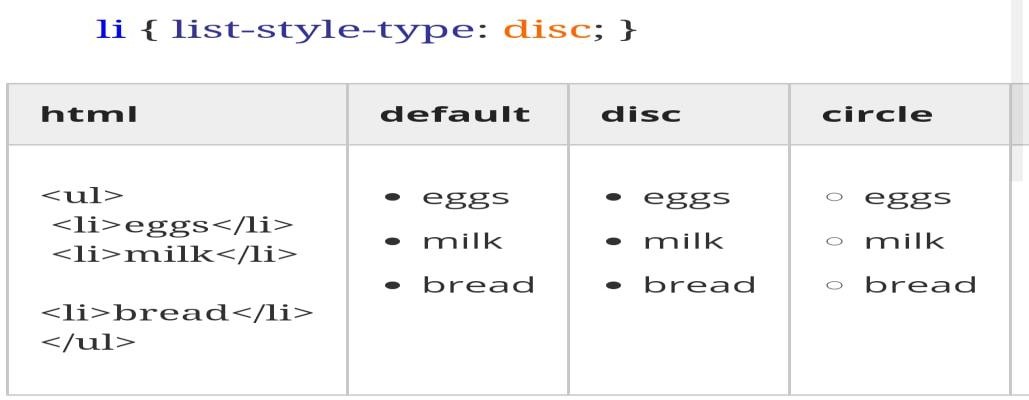
**LIST and SELECTORS :**

**Styling List :**

###### The list markup tags (<ul>, <ol> and <li>) are some of the most frequently used specific purpose tags in HTML. There are a few CSS style properties that are available for lists.

**List-style-type :**

list-style-type governs the little list marker that is usually positioned to the left of any list item. For un-ordered lists (<ul>), there are several popular values: disc, circle, square, and none.



### WEEK 4

IDENTIFYING HTML ELEMENTS :

Remember that elements are the intangible parts of your Web page, which are described by the text in tags and are rendered on the screen of whatever device you‟re looking at your Web page with. The two things (the text code and the pixels on the screen) correspond to each other, but it‟s not always obvious which bit of the screen corresponds to which bit of text.

There are two opposite directions in which you might need to figure out in these two different things that both correspond to an element. You might have some HTML5 code that you‟ve written and want to find out where on the Web page that code shows up. The other direction can be needed as well, i.e. Given a particular part of the page, what part of your code produced it?

When you hover over an element in the DOM Explorer window in your browser developer

tools, the corresponding element on the displayed page is highlighted:

It is also possible to go the other direction, i.e. Click on a point on the displayed page and it will highlight the code in the source that corresponds to that element. This is helpful when you want to figure out where something came from and what might be affecting it‟s styling (size, color, font, any number of other characteristics). To do that in your browser developer tools, use the DOM explorer pane and the “select element” option, you can also right click on the section on your page you want to inspect and select “Inspect element” which will bring up the developers tools and highlights the HTML element or code for that section on the page.

MODIFYING HTML ELEMENTS :

Another handy feature of the developer tools is the ability to make temporary modifications to your code to try out different things and see what works the way you want it to. When you have a visible element selected in the DOM explorer tab, you can make style changes in the “Styles” panel, or use the “Computed” panel to see the values for each property and how they were determined.

It‟s possible to change things a few different ways. If you double click on an element in your HTML5 source code, you can change the source code. For example you could click on an attribute to modify it or it‟s value, or you can change the type of the tag or even the contents of an element.

You can use this same approach to add a “style” attribute to a particular element, which should override any other settings, there is also an easier way to do that. In the panel just to the right of the elements panel is the another panel with tabs including “Styles” and “Computed” and a few others. Most of the time we‟ll want the “Styles” tab activated. Once you do that, you can modify CSS properties of the current element by adding them to the “element.style” box at the top of the “Styles” panel.

Just click in between the two curly braces on the “Inline style” rule at the top of the Styles panel. After clicking you should see a little text entry box with which you can type property value pairs that will then effect the currently active element.

It‟s important to know that any changes you make in the developer tools will have no effect on the original Web page. They only affect that particular instance of that page during that debugging session. If you navigate to another page and come back, you‟ll need to make the same changes again if you want to get back to where you were. It‟s not that easy to break the Web!

### WEEK 5

CSS BOX MODEL :

Before we get too far into debugging, it‟s helpful to understand a couple of things about CSS more deeply. The placement of elements on a Web page can be fairly complicated. One of the most basic features that influence where things go on a Web page is the CSS Box Model. The Box Model governs 3 important spacing features of CSS. We learned about margins previously as the space between elements. There are two other similar notions, padding and borders.

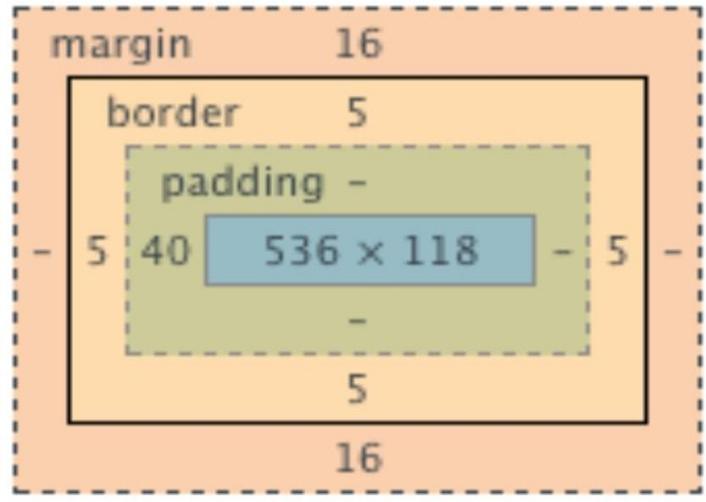
Perhaps the best way to understand is with a picture. All elements in an html document end up being treated as rectangles somewhere in the window. The content of each rectangle corresponds to the innermost rectangle in the image below. Just outside the content is the padding. This is kind of like an internal margin, meaning that it separates the contents from the border. The border essentially traces the sides of the padding rectangle. It‟s important to note that the border goes around the content and the padding. There are sometimes visible things associated with an element that are not technically part of the content of the element. One such example is the list item:



The box does not include the bullets because it is outside of the content. Sometimes when you see that it might be a bit confusing, especially because it also affects padding (which is inside the border).

DEBUGGING WITH THE BOX MODEL :

In any browser‟s debugger, you will see a box model diagram. It looks like this:



This is an example of a diagram of the box model information for a selected element.

The innermost box gives the dimensions of the element, outside of that is the padding, then the border around which is the margin. On each side of each corresponding rectangle is the width in pixels of that side, with “-” when it is 0 (essentially non-existent). Also, when you hover over one of the rectangles, that portion of element is highlighted on the rendered page, so you can see exactly where the margin, the border, the padding and the element are.

### WEEK 6

##### TEXT BASELINE and DISPLAY PROPERTY :

When newbie developers are groping around CSS blindly, they often stumble upon a variety of CSS properties that could be used to alter the positioning or size of an element such

as left, top, and margin.

However, when using the properties, these developers get confused because the properties fail to behave consistently. Sometimes the properties work, sometimes they don‟t, sometimes they do the opposite of what they are doing in a different rule. We have not covered properties

like left and top yet, but we have introduced margin and the intrepid readers may have already discovered in their exercises that margin can have some unexpected behavior. Why is that?

The answer has to do with the two CSS properties: display and position. The display property, in particular, has different default values for different tags. Some tags start with display:block, and others are display:inline. They behave very differently. These two properties

(display and position) often change how an element responds to certain other layout

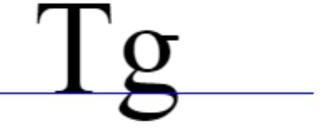
properties. And when this is not understood, then it may seem random to a developer struggling to get stuff to work.

So, let‟s start with understanding a very important difference between block and inline display.

And that begins with the baseline.

Baseline :

The text “baseline” is a key concept to understanding how the browser makes its layout decisions.



In the image above, we see two text characters placed next to each other, the blue line indicating the baseline. The baseline determines how and where the characters are positioned. Note that the tail of the “g” hangs below the baseline.

The baseline is never drawn by the browser, it is not exposed directly to you as a developer, and CSS only may have some properties related to it. However, the baseline governs the placement of all inline elements.

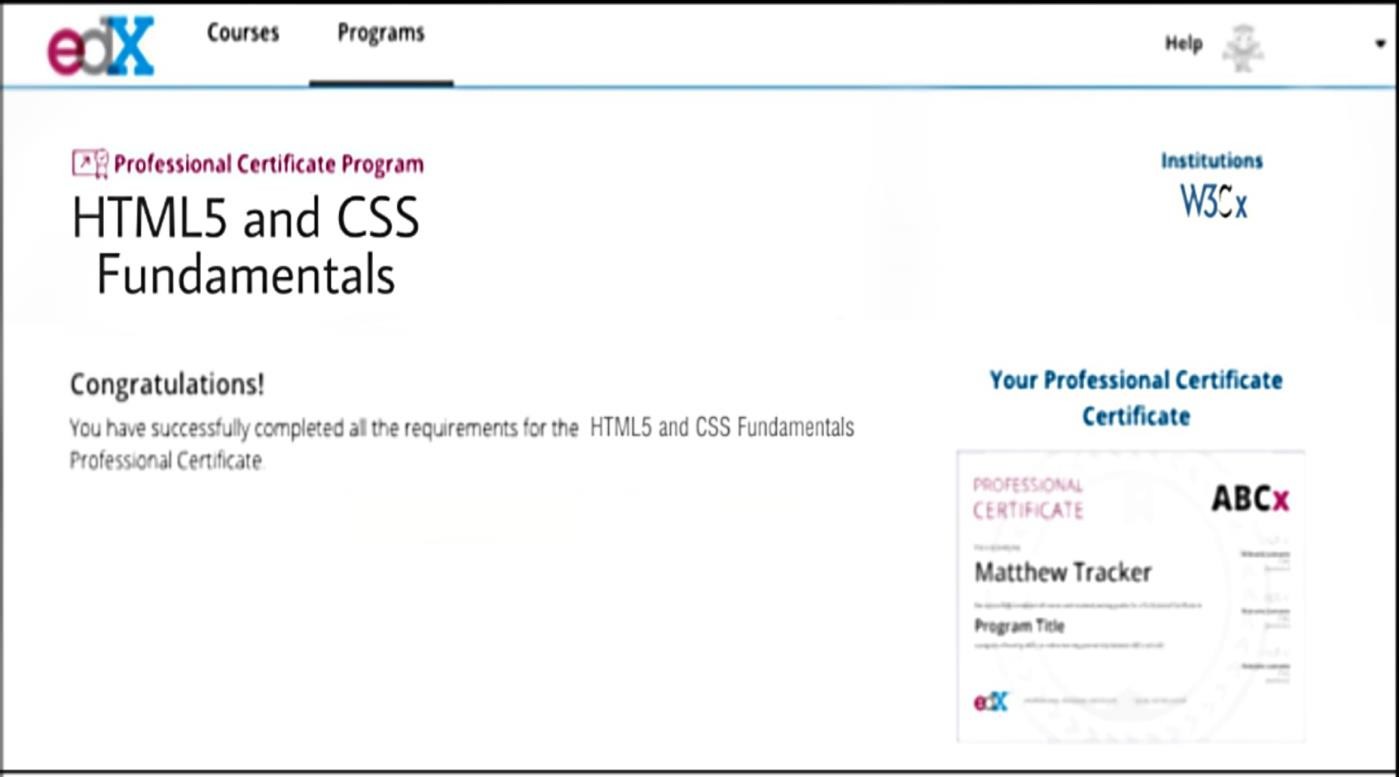
Display: box versus inline :

As the browser is rendering your page, every time it encounters the next tag it has a simple question: “Do I give this element its own line?” For example, every <p> tag gets a new line, but <a> tags do not. This is the key distinction between the ”block” level elements (like

the <p> tag) and the ”inline” elements (like the <a> tag). Here is a quick table of the default

values for some of the tags we‟ve already learned.

## Proof of Completion of Course :



### Web Site used for reference to complete this course :

**W3Schools - <https://www.w3schools.com/> EDX - <https://www.edx.org/>**

**Google – <https://www.google.com/>**

**THANK YOU.....**