# What I reviewed about HTML, also what I coded

HTML (HyperText Markup Language) is a markup language, not a programming language.

<!DOCTYPE html> mentions that a document is an HTML5 document. It is different from the <html></html> tag that is our root element in an HTML page.

HTML element: is defined between two tags, like headers, paragraphs, links, images, etc.

Attribute: attributes some characteristics to the HTML element, like href in <a>, src, width, height, alt in <img>, style in p, lang in html, title in p.

**Note**: alt is just for image elements, the title is just for text elements.

**Note**: <p> ignores spaces and line breaks, but <pre> keeps them all and shows them just in a fixed font like the Courier font.

Some style examples:

<tagname style="property:value;">

<body style="background-color:powderblue;">

<p style="color:red;">This is a paragraph.</p>

<p style="font-family:courier;">This is a paragraph.</p>

<p style="font-size:160%;">This is a paragraph.</p>

<p style="text-align:center;">Centered paragraph.</p>

<h1 style="border:2px solid Tomato;">Hello World</h1>

Formatting tags which are used for formatting the text are listed below:

* <b> - Bold text
* <strong> - Important text
* <i> - Italic text
* <em> - Emphasized text
* <mark> - Marked text
* <small> - Smaller text
* <del> - Deleted text
* <ins> - Inserted text
* <sub> - Subscript text
* <sup> - Superscript text

Commenting a HTML tag is in this way: <!—some code here -->

For colors with can compose them with different quantities of 3 colors.

Bookmark is used for moving in a page. I took a note from this, it can be found on the folder of the note. Its main idea is to work with “id” and address it by #id.

**Image maps**: putting link on images

<img src="workplace.jpg" alt="Workplace" usemap="#workmap">  
  
<map name="workmap">  
  <area shape="rect" coords="34,44,270,350" alt="Computer" href="computer.htm">  
  <area shape="rect" coords="290,172,333,250" alt="Phone" href="phone.htm">  
  <area shape="circle" coords="337,300,44" alt="Coffee" href="coffee.htm">  
</map>

For complex shapes, we can poly as follows:

<area shape="poly" coords="140,121,181,116,204,160,204,222,191,270,140,329,85,355,58,352,37,322,40,259,103,161,128,147" href="croissant.htm">

Good usage of image maps: cooperation with JS

<map name="workmap">  
  <area shape="circle" coords="337,300,44" onclick="myFunction()">  
</map>  
  
<script>  
function myFunction() {  
  alert("You clicked the coffee cup!");  
}  
</script>

*<picture> and <img> difference*: (1) **Bandwidth**: small devices don’t need large files (2) **Format Support**: to put different formats of images

<picture>  
  <source srcset="img\_avatar.png">  
  <source srcset="img\_girl.jpg">  
  <img src="img\_beatles.gif" alt="Beatles" style="width:auto;">  
</picture>

How to create a horizontal list: <https://www.w3schools.com/html/tryit.asp?filename=tryhtml_lists_menu>

Inline and Block Elements: Inline ones just come after an element before them and it only takes up as much width as necessary, but block ones start a new line and take up the full width available, <div> and <span> is the most famous block and inline element.

Iframe is for showing a web page within a web page.

<iframe src="demo\_iframe.htm" style="height:200px;width:300px;border:none;" title="Iframe Example"></iframe>

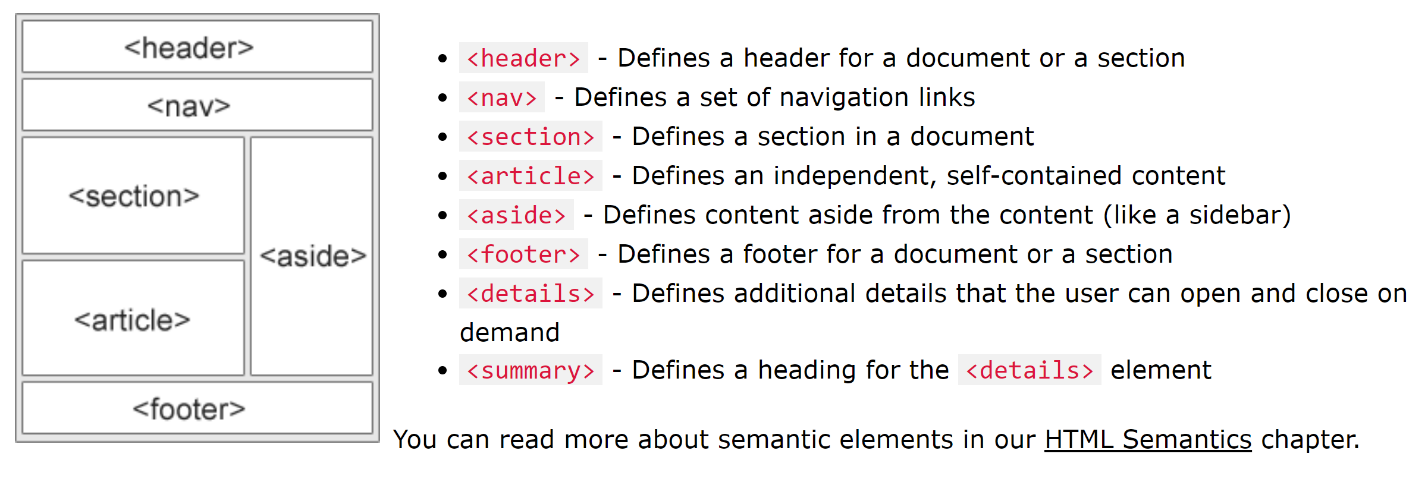
We can use JS codes in out HTML file by writing our script between two <script> and </script> tags. The example can be viewed in the following link:

<https://www.w3schools.com/html/tryit.asp?filename=tryhtml_script_html>

<head> is for metadata, like <title> which is for SEO, <style> for CSS, <link> for linking to CSS stylesheets, <meta> for different purposes from language, character set to keywords and viewpoint.

<meta name="viewport" content="width=device-width, initial-scale=1.0">

**Layout in HTML**:



These can be formatted easily by the help of CSS frameworks like Bootstrap and W3.CSS.

HTML Forms: For receiving data from the user.

**Notes on GET:**

* Appends the form data to the URL, in name/value pairs
* NEVER use GET to send sensitive data! (the submitted form data is visible in the URL!)
* The length of a URL is limited (2048 characters)
* Useful for form submissions where a user wants to bookmark the result
* GET is good for non-secure data, like query strings in Google

**Notes on POST:**

* Appends the form data inside the body of the HTTP request (the submitted form data is not shown in the URL)
* POST has no size limitations, and can be used to send large amounts of data.
* Form submissions with POST cannot be bookmarked

There many input types, which make the validation very easy.

I also learned how to embed a video from YouTube using Iframe.

# What I reviewed about CSS, also what I coded

CSS stands for Cascading Style Sheets

External CSS:

<head>  
  <link rel="stylesheet" href="styles.css">

<link rel="stylesheet" href="https://www.w3schools.com/html/styles.css">  
</head>

Inside of a CSS file:

body {  
  background-color: powderblue;  
}  
h1 {  
 color: blue;  
}  
p {  
  color: red;

border: 2px solid powderblue;

  padding: 30px;

  margin: 50px;

  font-family: courier;  
  font-size: 160%;  
}

Link attributes:

<style>  
a:link {  
  color: green;  
  background-color: transparent;  
  text-decoration: none;  
}  
  
a:visited {  
  color: pink;  
  background-color: transparent;  
  text-decoration: none;  
}  
  
a:hover { // it is a pseudo-class  
  color: red;  
  background-color: transparent;  
  text-decoration: underline;  
}  
  
a:active { // it is also a pseudo-class  
  color: yellow;  
  background-color: transparent;  
  text-decoration: underline;  
}  
</style>

The “**float**” style of an image makes it out of a block.

I watched several videos and coded simultaneously with them on how to use class, id.

I learned how to change the color, and how to use gradient. How to put background image, style it, and give HTML elements borders.

Also, I learned how to use the inspect feature of the browser to change the style of the elements in a fast way.

Combinators in CSS are something that specify the parent child order like <ul li> or <ul > li>.

Grouping in CSS is a help for reducing code redundancy. It is something like as follows:

h1, h2, h3, .class, #id {

font-color: red;

}

And about text formatting: I learned how to change the font of the text which comes in the paragraphs. We can change the size of a font with a font-size attribute using pixels or rems. Their difference is that px is not scalable but em and rem are scalable. Change in the root of the em and rem is going to change the interpreted font size.

/\* style \*/

html {

font-size: 18px;

}

section {

font-size: 14px;

padding: 3em;

}

/\* computed style \*/

html {

font-size: 18px;

}

section {

font-size: 14px;

padding-bottom: 42px;

padding-left: 42px;

padding-right: 42px;

padding-top: 42px;

}

On rem:

/\* style \*/

html {

font-size: 18px;

margin: 2rem;

}

/\* computed style \*/

html {

font-size: 18px;

margin-bottom: 36px;

margin-left: 36px;

margin-right: 36px;

margin-top: 36px;

}

Text-align: center, right, left 🡪 changing the alignment of the text inside an element.

Changing the style of links 🡪 text-style: none;

Text-transform: uppercase;

Word-spacing: 2em;

Line-height: 2em;

Then I learned how to use Google Fonts using the prepared API by Google.

<link rel="preconnect" href="https://fonts.gstatic.com">  
<link href="https://fonts.googleapis.com/css2?family=**Source+Sans+Pro**&display=swap" rel="stylesheet">

font-family: 'Source Sans Pro', sans-serif;

**Note**: using too many fonts on a website can be distractive, so we have to try to keep it simple. For this purpose, we can use the fonts in a standard or import way by URL.

**Responsive CSS Images**: we can use margin and padding in auto mode inside a div in which our image elements are placed there. Also, we have to use a percentage to specify the images’ width and height. For example, we can give max-width a number in percentage and height an auto.

**Rows and columns in CSS**:

I learned how to do it and the code can be found in the subfolder of the CSS folder on my GitHub page.

**Building the Navbar**:

I built it using CCS and JS.

**Note**: Using class is for styling, but id is useful when we want to interact with the JS.

I finished by building a simple responsive website.

# What I learned about Bootstrap 4, also what I coded

Bootstrap is one the popular frameworks for building mobile first website or applications. The most interesting thing about it is that it is pre-built with loads of classes and stylings, so it virtually provides us what we are going to need in frontend design.

On the first step, I learned how to download and use it from its website. We can use the document of this framework to put what we want in our website.

I learned how to use documentation of the Bootstrap to use different component or arrange components in **row, column** responsive structures.

I enjoyed using **cards** for sign-in design, and **carousels** for image slide shows. Also, **modals** are fascinating. Finally, I ended this technology by writing a sample website. Additionally, I learned about “**awesome font**”.

# What I learned about Sass, also what I coded

**SASS** stands for **S**yntactically **A**wesome **S**tyle **S**heet.

# What I learned about JS, also what I coded

I watched the tutorial videos and everything was ok about how to use this language. It is like the Python language. I have to learn these concepts:

* 1. JS bind
  2. DOM

Also, developed a percent calculator listening to a form, then finally disabling its default behavior not to refresh the form after submit by using event.preventDefaults();.

I have to work on this language and learn its concepts at in Advanced Level.

# What I learned about C#, also what I coded

# What I learned about SQL, also what I coded

# I also reviewed the Git basics and you can find all I did on my GitHub page.

Always we have to try to use a control version system like Git.

The problems that these systems try to solve:

1. Reverting to old files
2. Maintaining code within a team.

Then, I reviewed some of Unix shell commands like cd, dir, mkdir, ls, touch, cp, mv, rm, rmdir, etc.

After that I moved to review what I knew about Git. I started add an SSH key to my GitHub page. Then, I created two projects in which in them I pushed what I did in several days. I list below the commands:

git init

git add .

git commit -m “the message”

git status

git log

git checkup #number-of-a-commit

git branch

git checkout master

git remote -v

git remote add <origin🡪 we can name it everything we like> <link-of-the-repository-which-we-are-going-to-push-our-project>

git pull origin master

if there are no conflicts, the merge is going to be done on the project. However, if any conflict occurs we have to solve it first by forcing it or talking with the teammates and solving it.

GitHub is a server for pushing for our codes on it, but Git is a CVS which can act locally. We can use any other server to synchronize our project’s codes with our teammates. Bitbucket is a competitor of GitHub. It is cheaper than GitHub. However, GitHub is more well-known for open source project rather than other services.