**HTML CSS by freeCodeCamp**

**HTML 🡪** It is a markup language, not programming language. It is responsible for webpage structure:

Syntax: **<element>** content **</element>**

* Install Chrome, Install VS Code, some extensions too
* Create a folder, open it in VS Code and add a HTML file
* Una vez en VS Code asociamos el Repository Git: https://github.com/PadillaTom/FrontEndJourney.git

(Log in user.name // user.email // git init // git remote origin LINK //git pull origin master // git push –u origin master)

* Una vez Asociado Git con Remote y Folder comenzamos los tutoriales

**Structure and Elements:**

<!DOCTYPE html> 🡪 HTML version.

<html> 🡪Root Element.

<head> 🡪 Information about the page (Meta, Links, Title, etc) .

Content won’t be visible.

<title> xxx 🡪 Page title (shows in tab).

<body> 🡪 What will be displayed on the page.

* There are Parent Tags and Children Tags:  
  <head>

<title> **Titulo 1**

* If something is too long we can use **ALT + Z** (word wrap)
* What is Emmet? Something that speeds up my work. In VS Code it’s already built in. EJ: Every time I want to create a HTML element I don’t need to type the < >, Emmet will fill with the suggested abbreviation. <html></html>
* Vemos la estructura general y una vez armada podemos comenzar: Se recomienda utilizar Live Server (right click on VS Code + open with Live Server)

**Headings <h>:**

They are a kind of title. There are six types, being h1 the bigger size:

<h1> <h2> <h3> <h4> <h5> <h6>

**Paragraph <p>:**

Where we are going to put some kind of Text component. HTML it is “white space collapsing” (HTML is going to ignore extra spacing) : <p> **Hello**

**this is a paragraph**</p> 🡪 HTML will ignore all the spaces and print it all together.

* Si queremos crear SPACES tenemos que usar un tag especial
* Por momentos sabemos que tenemos content, pero no lo tenemos, podemos usar **LOREM X** (x = number of words):

<p> **Lorem 20** </p>

**Images <img src = “ “ alt=” “ >:**

We can upload images for every kind of content. They must be in the same folder as the website or can be uploaded by URL().

SRC = ” “ 🡪 Path (Folder **./** or URL() )

ALT = “ “ 🡪 What we can see when the image is broken, like a short description

./ 🡪 “in the same directory”

../ 🡪 “outside the directory”

URL() 🡪 Get to the image file (Jpg, Png, etc) and copy link.

Debemos usar website de busqueda para copyright free images (Pixabay, Gratisography, etc).

* Render size 🡪 via HTML (we will use CSS or even cropping

them before)

<img src=”” width=”260 alt=”””> 🡪 Width =” “ // Heigth will be

automatically adjusted.

* Ahora … si tenemos muchas imagines el usuario va a experimentar un website muy lento…. Podríamos ajustar las imágenes previamente Photoshop, Mac, Etc. o **utilizar CSS.**
* Creamos una IMG folder y dentro sub folders según utilidad.

**Comments <!—xxx -->:**

We should comment at the Start or End of every section.

**CTRL + }** 🡪 To add or convert into a comment.

**Line Breaks <br>:**

This will break the white space collapsing. Just like print(“ “) in python. It adds lines.

**BR\*6** 🡪 Shortcut that will add 6 times the <br>

**Links <a href=” “>:**

It is called an Anchor Tag:

<a href= ” # ” > **Some clickable Text** </a> 🡪 **#** it is a placeholder that will be replaced later in time for a real link

* Target =” **\_blank** “ 🡪 It will open link on a new Tab
* What about internal surfing? Creamos un **About.html** y agregamos un **<a>** que nos lleve a **Index.html**.
* “Back to the top” 🡪 Usamos **src =” # “. #** Como en CSS # representa un ID, podríamos poner cualquier **#ID** y nos llevaría hasta él.
* Podemos usar distintos elementos para clickear un Link:

<a>

<img>

**<Sup> and <Sub> Elements:**

These elements will be places in top or bottom of the RENGLON.

<h1> **Hello I am** <sup>**1st**</sup> **John to arrive in** <sub>**town**</sub></h1>

* 1st and Town will be shown NOT IN LINE.

**<Strong> and <Em>phasis Elements:**

Within some text we can have STRONG and EM.

<p>**lorem20**<strong>**Bold Text**</strong><em>**Italic Text**</em></p>

Strong 🡪 Bold text

Em 🡪 Italic Text

* The ideal is to have a CSS for style and HTML for structure. Not to mix them both.

**Special Characters” &char; “ :**

Por ejemplo el logo de Copyright:

<h1> Copyright **&copy;** </h1> 🡪 Replace copy for another character, Emmet will suggest the names.

**List Structures <ul> <li>:**

<ul> 🡪 Unordered List: Uses points, can be removed by CSS

<ol> 🡪 Ordered List: Uses numbers, CSS can remove

<li> 🡪 List Item

* Usually we have List items inside the Lists, we can have as many as we need
* We can have Elements inside List Items

<li> <a href =” “ > **About** </a> </li>

<li> <img src=” “> </li>

* We can also have “Nested Lists”

<ul>

<li> **John** </li>

<li> **Peter**

<ul>

<li> **Html 🡪** Appears like a sub list of Peter.

<li> **Css**

</ul>

</li>

</ul>

**Tables <table> <tr> <th> <td>:**

Like any other table we need:

<tr> 🡪 Table row

<th> **Name**  🡪 Table Headings

<th> **Age**

<tr>

<td> **John** 🡪 Table Data

<td> **Peter**

<tr>

<td> **20**

<td> **25**

**Forms:**

**Input and Submit <form> <input> <button>:**

Generalmente las usamos para colectar data. HTML trabaja únicamente con el Front End, no podría coleccionar data, simplemente armar la estructura **(FormSpree website will créate the form and send info to our email)**

<form action =”” method=””>

<input type=”text” name=”” id=” ID ”>

🡪 Type: tipo de input // name: lo que se pide

<button type=”submit”> **Submit**

<input type=”submit” value=””> 🡪 Funciona como botón de submit // Value: Ingresamos texto del boton

* A su vez podemos especificar la info solicitada antes de la casilla del input

<label for=” ID “> **First Name** </label>

<input type=”text” id=” ID “>

🡪 La ID debe coincidir con la ID del input. Ejemplo: NAME. Si se hace click en la Label se resalta el casillero de Input

* Solicitar una password sin usar labels:

<p> Password

<input type=”password” placeholder=”Type in your password”>

🡪 When we type password it is NOT Visible

**Textarea Radio Checkbox <textarea> <input> <select>:**

<**textarea** name =”” id=”” cols=”10” rows=”10”>

🡪 Es simplemente un área para escribir, podemos especificar cantidad de renglones. Cols y Rows : Cantidad de renglones.

<input type=”**radio**” value=”Javascript“> **Javascript**

<input type=”**radio**” value =”Python”> **Python**

**🡪** You can ONLY select ONE of the items. **Si bien uno selecciona lo que diga, el valor real que se envia es el VALUE.** Name debe concidir entre las 3, sino podrá elegir el usuario entre varias opciones.

<input type=”**checkbox**” **checked** value=” ”>

🡪 You can select MULTIPLE values within the options. **Si bien uno selecciona lo que diga, el valor real que se envia es el VALUE.** Checked = Have the box already checked by default

<**select** name=”” >

<**option** value=” ”> Javascript </option>

<**option** value=” ”> Python </option>

</select>

🡪 You can select ONE of the list. **Si bien uno selecciona lo que diga, el valor real que se envia es el VALUE.**

**Prettier and format:**

We can install an extensión to use the formatting function: Prettier Code 🡪 Whenever I Save it will automatically format it to make it prettier and understandable.

File -> Preferences -> search for : Format On (type/save)

**Keyboard Shortcuts:**

Look the JPG for windows shortcuts.

Most importants :

* Ctrl + Z : Undo
* Alt + Click : Multiple Cursors
* Ctrl + I : Select whole Line
* Alt + <- or -> arrows : Navigate Tabs
* Ctrl +up down arrows : Start or End document
* Ctrl + } : Comment Line or Start a comment
* Shift + Alt + up down : Copy line
* Alt + up down: Move the line up or down
* Ctrl + Enter : Creates a new line
* Para H Refs = “ ../ “ : Buscar en directorios

**First HTML Project DONE!!!**

**CSS 🡪**  Cascade Style Sheet. We need to create a new file with the extension **.css.** It is responsible for styling the web, the looks.

Selector **{**property: value **;** property: value , etc. **}**

* There are different ways of using CSS:
  + Inline CSS 🡪Inside our HTML <h1 style=” “>
  + Internal CSS 🡪 Inside HEAD: <style> h1{ ; }
  + External CSS 🡪 Inside HEAD: <link rel =” “

href= “Style.css”>

* Lo major esc rear un Style.css y asociarlo con LINK al head element de las pages.
* **Borramos todo lo anterior!!!! (Tests de los diferentes CSS)**

**Syntax:**

H1 {

Color: red;

}

**H1 🡪** Selector

**{ } 🡪** Declaration block

**Color:** 🡪 Property

**Red** 🡪 Value

**All together 🡪** Css Rule

**Selectors:**

* Element Selectors:

We should use the name of the element as the selector

h1 {Color;}

h2 {Color;}

p {Color;}

Selector Grouping:

Podriamos simplemente usar el **<body>** siendo el elemento que contiene todo. 🡪 Body {color}

Tambien podemos usar “ **,** “ 🡪 h1, h2 {color}

* ID Selectors:

Damos un ID al elemento en el html y luego lo usamos en el css. IDs should be unique, one per element.

<h1 id = “heading”> 🡪 #heading {}

* CLASS Selectors:

Las clases pueden ser compartidas de manera de hacer un estile a la class y que se aplique a todas a la vez.

<h3 class = “Green”> 🡪 .Green {}

Podemos combinar ID junto con CLASS y una segunda CLASS:

Todos los TITLE tendrán cierta fontsize, color verde y además podemos hacer lowercase al text.

<h3 id=”title” class=”Green lowercase”>

**#title** {font-size} **.green** {color} **.lowercase** {text-transform}

**Div and Span elements <div> <span>:**

Cuando trabajamos con mucho contenido nos conviene usar Div y Span. Se usarán para agrupar a la hora de dar style.

<div> 🡪 Tag de DIV, agrupamos h3 y p

<h3> dentro.

<p>

</div>

<h3>

<p> **blablablá** <span>**blablablá**</span>

Style:

div {color; bgcolor} 🡪 Everything inside the div will be changed.

Span {text-transform} 🡪 Everything within span will change.

Podemos mezclar DIV o SPAN con CLASS tambien:

<div class=”red”> 🡪 Entonces se estiliza como **.red**

<span class=”red”> 🡪 Se le aplica la **.red**

* DIV is for starting a new line. “Block level element”
* SPAN inline style. “Inline element” 🡪 To style something within a text already, so we don’t want to create a new line.

**Inheritance:**

Whatever style we apply to a Parent element in HTML, children element are going to inherit. Unless we specifically style the children element.

* <body><p> **blablá** </p><p> **blabla** </p></body>

\*\* body {color: red} 🡪Color will be applied to both P

* <body><div><p> **blablá** </p><p> **blabla** </p></div></body>

\*\* Body {color: red} div {color: blue} 🡪They will turn blue, since DIV is a child element and it was specifically styled to blue.

* <body><div><h2>**Hello**</h2><p> **blablá** </p><p> **blabla** </p></div></body>

\*\* Body {color: red} div {color: blue} h2 {color:green} 🡪 h2 will be green the rest will be blue. H2= child.

\*\* Body {font-family; line-height} div {color: blue} h2 {color:green} 🡪 They will change FONT and LINE, but NOT colors, because font and line hasn’t been overwritten yet.

* Todo será inherited hasta que se modifique el valor especifico. NO TODO: border por ejemplo no.

**Last Rule and Specificity – Universal Selector \* :**

Pequeñas reglas de la lógica detras de CSS styling

Last Rule 🡪 Si usamos **p {color: red}** y luego **p {color: blue}.**

¡Se aplica a la última! Ganará la **BLUE**

Specificity 🡪 <p class= “red”> // .red {color: red} p {color: blue}

¡Se aplica la más específica! Ganará la **CLASS**

Universal Selector 🡪 **\*** {color: blue} p {color: red}

¡La que menos fuerza tiene! Ganará el **P**

Se usa cuando queremos resetear el Default del browser.

**Colors:**

<h3 id=”first”> **I am first** // #first **{COLOR; BGCOLOR}**

<h3 id=”second”> **I am 2nd** // #second **{background}**

* Color 🡪 Da color a textos
* BGcolor 🡪 Da color a fondos
* BG 🡪 Da color como tambien imagines **url()**

RGB (red, green, blue)

Color: rgb (00, 00, 00) 🡪 255 the max

RGBA (red, green, blue, alpha)

Color: rgba (255, 0, 0, .25) 🡪 Opacity

HSL and HEX

Color: # RR GG BB 🡪Numbers and letters a=10 f=15.

Ej: Red Color = #ff0000 🡪 F= highest, A= lowest.

Hay muchos colores que alcanza con los primeros #000 (black)

4.56.01

**Datos Curiosos**

* DIV is for starting a new line. “Block level element”
* SPAN inline style. “Inline element” 🡪 To style something within a text already, so we don’t want to create a new line.
* Everything will be inherited unless we change the property’s value specifically. (font-site: 1.5rem)

🡪 BORDER por ejemplo NO inherit.

* Pick nice colors: coolors website.