**TUTORIAL HTML**:

1. **Introducción**

**Los *tags* crean capas de ejecución o de priorización. Permiten personalizar el aspecto y el contenido de la página web**. El **cuerpo básico** de una página web contiene:

<!DOCTYPE html> 🡪 Define el tipo de documento del documento en el que se encuentra. Es un tag individual y no requiere cierre.

# A continuación, se muestran los *container* o contenedores:

<html> 🡪 tag de iniciación del programa. Contiene todo lo demás. Es padre de <head> y <body>.

<head> 🡪 Define *data* sobre el documento. Por ejemplo: título del doc; descripción; definir atributos como imágenes, scripts de JavaScript, etc. Se encuentra **indentado** dentro <html>; esto se debe a que está contenido en, ya que no necesita espacios. Es hijo de <html> y hermano de <body>.

</head> 🡪 cierre de head

<body> 🡪 Define la estructura de la página. Generalmente es el tag de mayor longitud y posiciona en el espacio cada elemento de la página. Es hijo de <html> y hermano de <head>. Todo lo que se introduzca en este tag tratará de printearse por pantalla.

</body> 🡪 cierre de body

</html> 🡪 tag de cierre del programa. Forma un par con <html>.

Los *browser* son capaces de leer html y descifrarán el contenido de las etiquetas por orden de prioridades.

1. **Otras etiquetas**

<title> XXXX </title> 🡪 Contenedor de XXXX; determina el nombre del archivo. Habitualmente en <head>.

**Un meta-tag es un tag que define información de nuestro documento (o metadata). Habitualmente en <head>.**

<meta XXXX> es un meta-tag que define metadata que no puede ser representada por otras etiquetas como: <base>, <script>, <link>, <style> o <title>. No requiere cierre de etiquetas y XXXX representa a los **atributos** de meta; algunos de los atributos son:

XXXX = charset=”UTF-8” 🡪 Define el cifrado de los caracteres a UTF-8

XXXX = name=”description” content=”This is an example”🡪 Crea una descripción de la web; necesaria por Google, por ejemplo.

**Los atributos, o propiedades, se llaman dentro del tag, simplemente con un espacio y llamando al atributo.**

<h1>XXXX</h1> 🡪 Representa un encabezado; y pueden ser llamados hasta 6 encabezados (h1, h2, h3, h4, h5) en los que cambia el tamaño e importancia de este. El contenido se escribe en XXXX y se encuentran en <body>. Puede sustituirse por un encabezado, por ejemplo.

<p>XXXX</p> 🡪 Crea un párrafo con el contenido XXXX. Tiene ciertas reglas de interlineado. Habitualmente en <body>.

<b>XXXX</b> 🡪 Negrita el contenido XXXX. Se encuentra en <p> habitualmente.

<i>XXXX</i> 🡪 Cursiva el contenido XXXX. Se encuentra en <p> habitualmente.

<br/> 🡪 Crea una línea de separación invisible. No requiere cierre. Se encuentra en <p> habitualmente.

<hr/> 🡪 Crea una línea de separación visible. No requiere cierre. Se encuentra en <p> habitualmente.

<big>XXXX</big> 🡪 Aumenta el tamaño de la fuente del contenido XXXX.

<small>XXXX</small> 🡪 Disminuye el tamaño de la fuente del contenido XXXX.

<sub>XXXX</sub> 🡪 Subíndice el contenido XXXX.

<sup>XXXX</sup> 🡪 Superíndice el contenido XXXX.

**CSS se encarga del estilo y colores de la web.**

1. **Comentarios**

Se pueden crear un comentarios mediante:

<!--> XXXX 🡪 Todo el contenido XXXX a la derecha del tag se comenta.

<!-- XXXX --> El contenido XXXX se pone como comentario.

1. **Introducción a CSS: Hojas de Estilo en Cascada**

Se puede llamar como **atributo** a CSS. Por ejemplo:

<p style=”color: red; background-color: blue; ”>

En el ejemplo, *color: red;* se trata de código CSS. Se debe tener en cuenta que es tipo cascada, y el estilo que se introduzca en un tag se aplicará también a todos los tag hijos.

1. **Formateo de una web**

Algunas etiquetas permiten dividir y clasificar todos los elementos del código de forma eficiente y óptima. Dividen principalmente el espacio en tres secciones: el encabezado; el cuerpo de la web y el pie de la web.

<header>XXXX</header>

Contenido propio:

<nav></nav> 🡪 Permite crear una lista de navegación entre diferentes enlaces.

<main>XXXX</main>

Contenido propio:

<article>XXXX</article> 🡪 XXXX es el contenido del artículo. Es como un <p>, pero el browser lo detecta como un artículo.

<section>XXXX</section> 🡪 Permite separar bloques de texto por etiquetas. Actúa como un <p>.

<aside>XXXX</aside> 🡪 Permite señalar que el contenido XXXX no pertenece al *core* de la página web.

<footer>XXXX</footer>

Todas tiene el mismo padre, <body>. Cada sección tiene unos contenidos propios. El objetivo de estas divisiones es facilitar la organización de los motores de búsqueda de los navegadores.

21/09/2020. Duración: 2,5 horas

1. **Enlaces**

<a href=”” target=”\_blank” >XXXX</a>

La etiqueta <a> crea un link. Debe estar cerrada y puede introducirse en cualquier sección. El contenido XXXX es el elemento html que incluye el link.

**El atributo *href* es el link al que se va a hacer referencia.**

**Si es externo, el enlace debe incluir el protocolo de comunicación; habitualmente https://**

**Si es interno, debe contener la ruta que contiene al documento html. Si está en la misma carpeta, basta con el nombre.**

**Si no quiere enlazarse con nada; es buena práctica incluir #**.

El atributo y valor target=”\_blank” permite abrir el link en una nueva pestaña.

1. **Imágenes**

<img src=”” alt=”” />

La etiqueta <img> crea la imagen referida por el atributo *src*. Se trata de un tag abierto y puede introducirse en cualquier sección. Suele cerrarse con /> es un tipo de cierre que cumple las condiciones del lenguaje XML; se aplica en tags que no requiere un cierre como <br>

El atributo src=”” hace referencia a la localización de la imagen. Puede ser online, en cuyo caso se pondría el enace, o interna, donde se pondría la ruta. **Es una buena práctica**.

El atributo alt=”” muestra la ruta alternativa en caso de que *src* falle.

El atributo width=”XXXX” acota la anchura de la imagen a XXXX pixels. El atributo height hace lo mismo con la altura. Si solo se ajusta uno, html calculará el otro en función de las dimensiones originales.

21/09/2020. Duración: 3/4 horas

1. **Vídeo**

<video src=”XXXX” poster=”YYYY” controls>YYYY</video>

El atributo src localiza en XXXX la ruta del vídeo. Para vídeos internos; se debe mostrar la ruta.

El atributo controls permite realizar las funciones básicas sobre el vídeo: play/parar.

El contenido YYYY se muestra cuando el navegador donde se ejecuta el código no es capaz de mostrar vídeos.

Se les puede añadir los atributos weidth y height para ajustar anchura-altura.

El atributo poster=”YYYY” muestra como imagen inicial del video la imagen que le pasas como argumento YYYY.

<iframe> es un tag específico para la localización de vídeos externos.

1. **Listas**

Para crear una lista sin un orden particular, se emplea:

<ul> 🡪 Se trata de un tag contenedor. Crea una lista indexada con un marcador (tipo Word) circular o flector.

<li>XXXX</li> 🡪 Define un elemento de la lista. Se necesita uno para cada elemento de la lista. El contenido puede ser muy variable. Por ejemplo: strings, links, imágenes.

<li>YYYY</li>

</ul>

Para crear una lista con un orden, se emplea <ol type=”XXXX”></ol> sustituyendo a ul anterior. Enumera los elementos de la lista.

El atributo type=”” define el tipo de enumeración deseada. Por ejemplo: “A” enumerará a partir de la A, por elementos del abecedario. “i” lo hará por números romanos.

Se pueden indexar listas dentro de listas.

Para crear una lista descriptiva, donde adjuntar una descripción a cada elemeto, se emplea el tag <dl>:

<dl> 🡪 *Descriptive list*

<dt> XXXX </dt>🡪 *Description term*. Funciona como un elemento normal de una lista.

<dd> YYYY </dd> 🡪 Descripción YYYY del término anterior. Tiene un nivel de indexación extra al <dt>.

</dl>

1. **Tablas**

Se crean con el tag <table>:

<table> 🡪 Crea la tabla. Simple contenedor

<thead> 🡪 Contenedor de la cabecera de la tabla. Solo aporta información.

<caption>ZZZZ</caption> 🡪 Este atributo permite nombrar a la tabla con el contenido ZZZZ.

<tr> 🡪 *Table Row.* Simple contenedor

<th>YYYY</th> 🡪 Table *header*. Crea los tipos de nombres en las columnas

</tr> </thead>

<tbody> 🡪 Contenedor del cuerpo de la tabla. Solo aporta información.

<tr>

<td colspan=”integer”>XXXX</td> 🡪 Table *data*. XXXX es cualquier valor. Por ejemplo: listas, imágenes, videos, links, etc. Cada <td> empleado entre <tr> crea una columna.

El atributo colspan=”” le dice al valor XXXX cuantos espacios (es decir, cuantas columnas) ocupar. Se le pasa un número entero.

</tr></tbody>

</table>

No se necesita que coincidan el número de columnas en cada fila; si falta, queda un hueco.

1. **Contenedores**

Son un conjunto de tags que envuelven a otros elementos del código. Es buena práctica envolver todo dentro de etiquetas más amplias por la interacción de html con CSS; pueden aplicarse *styles* en cascada. Existen dos tipos de *containers* que se diferencian por los conceptos de **block** (todo el contenido del contenedor de este tipo ocupa todo el ancho de la página) e **inline** (el tamaño se limita a las dimensiones del contenido).

Por ejemplo: <p> ocupa todo el ancho a pesar de la brevedad de su contenido; <a href> ocupa únicamente el espacio del elemento que llama al enlace.

<div> 🡪 Contenedor tipo **block**. No hacen nada más; simplemente limitan el espacio para el contenido que se envuelven. **No ponen el espacio que aplica <p>**.

</div>

El otro container se trata de:

<span> 🡪 Contenedor tipo **inline**. No hacen nada más; simplemente limitan el espacio para el contenido que se envuelven y permite aplicar CSS.

</span>

1. **Input y Formas**

Los tags **input** van a permitir añadir información a la página. Esto no significa que pueda utilizarse la información; eso requiere de **JavaScript**. Algunos de los elementos que permiten recibir información son:

<input type=”text” /> 🡪 Tag abierto. El contenido de type determina que tipo de entrada se va a crear. Existen muchas opciones, “text” permite la introducción de texto.

<input type=”password” /> 🡪 Puedes introducir texto, pero cubre la información introducida.

<input type=”text” value=””/> 🡪 El atributo **value** permite introducir un valor predeterminado al cuadro de texto.

<input type=”date” /> 🡪 Fecha

<input type=”email” />

<input type=”file” />

<input type=”checkbox” />

<textarea rows=”” cols=””> 🡪 Crea un cuadro de texto ampliable; no es únicamente una línea. Se le puede especificar el tamaño con atributos; por ejemplo: rows=”” y columns=””.

XXXX 🡪 El contenido que se encuentra entre los tag es el texto predeterminado.

</textarea>

<form> 🡪 Es un contenedor de tags de input. Es simplemente informativo y facilitador de CSS y JavaScript.

</form>

1. **Frames**

**El <iframe> es un elemento html que permite mostrar elementos de otras páginas web en tu propia web.** No todas las web se pueden acoplar; por motivos de seguridad, privacidad puede ser aconsejable no permitir aplicarla.

Ejemplo:

<iframe src=””> 🡪 Muestra en tu web el contenido que se le pasa a src. Se le pueden aplicar atributos como width, height o frameborder.

XXXX 🡪 El contenido que se muestra si el navegador no permite mostrar el contenido de src

</iframe>

1. **MetaTags**

Se suelen colocar en el <head> del documento. Se utilizan metatags para implementar metadata, que es *data* sobre *data*. Permiten definir información de la página web en concreto.

Ejemplos:

<meta name=”description” content=””> 🡪 Permite dar una descripción a la web para facilitar su búsqueda. Suele ser recomendable ~160 caracteres.

<meta name=”author" content=””> 🡪 Descripción del autor.

<meta name=”keywords” content=”HTML, Tutorial, JavaScript”> 🡪 Ayudan en la búsqueda de la web.

<meta name=”viewport” content=”width=device-width, initial-scale=1.0”> 🡪 Controla como se va a mostar la web en distintos dispositivos. Es decir, en un móvil ajusta el tamaño de la web a la escala y anchura en la que ha sido diseñada.

22/09/2020. Duración: 2 horas.

**FUENTE: HTML Full Course – Build a Website Tutorial. freeCodeCamp.org**

**CSS. Cascading Style Sheet**

Permiten la estilización de elementos html de forma generalizada. Se asignan atributos determinados dentro de <style> a elementos en concreto.

Para estilizar un **tag** ya existente, como <p>:

<style>

p {

color: blue;

}

</style>

Para estilizar una **nueva clase**:

<style>

.blueColor {

color: blue;

}

</style>

Para ser llamada dentro de un elemento, debe incluirse dentro del tag.

Ejemplo: <p class=”blueColor”>Hola<p>

Para estilizar un **ID**:

<style>

#blueColor {

Color: blue;

}

</style>

Para ser llamada dentro de un elemento, debe incluirse dentro del tag.

Ejemplo: <p id=”blueColor”>Hola<p>

Existe solapamiento de estilos. Si se encontrasen estilizados simultáneamente por tag, por clase y por ID, el orden de preferencia sería: ID > CLASS > TAG. A excepción de llamar en la declaración del atributo css a **!important**.

Por ejemplo: color: blue !important;

También se debe destacar que los valores de cualquier atributo CSS pueden referirse a otro atributo en concreto. Para ello, se debe realizar una notación especial.

<style>

.custom {

--backColor: grey; 🡪 Variable

}

p { 🡪 Debes asegurarte que el elemento <p> también sea clase custom.

background-color: var(--backColor);

}

h2 { 🡪 Debes asegurarte que el elemento <h2> también sea clase custom.

background-color: var(--backColor);

}

</style>

**Variables**

Son atributos que se guardan como variables locales y pueden ser empleados por cualquier elemento html dentro de su campo de aplicación. Requieren introducirse dentro de una clase que contenga al elemento en concreto para que surja efecto. Ejemplo:

<style>

.custom {

--backColor: grey; 🡪 Variable

}

</style>

Para su llamada, debe citarse de la siguiente manera:

var(--NOMBREVAR);

Ejemplo:

Background: var(--backColor);

También pueden declararse como variables globales declarándolas en:

<style>

**:root** {

--backColor: Orange;

}

</style>

Permitirá el empleo de la variable en cualquier elemento html del programa. Se puede sobrescribir la declaración mediante un atributo en un elemento más interior a root. Es decir, si se especifica la variable en cualquier elemento.

A través de las **media query** se puede modificar el valor de una variable del root cuando se cumple una determinada condición. Se estructura dentro del style mediante @media.

<style>

@media (max-width: 300px) {

:root {

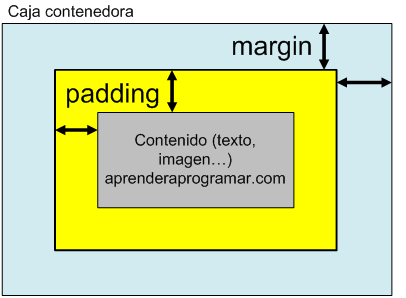
--backColor: blue;

}

}

En este ejemplo, cuando la ventana donde se abre la web es inferior a 300px, se cambia el color de la variable a Blue.

**Estructura**

La estructura de cualquier elemento html es un **rectángulo** con los siguientes elementos:

El **padding** es el rectángulo destinado al contenido del elemento html. Se puede manipular desde cuatro dimensiones: top, right, bottom, left.

padding: XXXXpx;

Determina el tamaño del rectángulo. Si se emplease padding-dimension indicaría la de ese dimensión. Se puede emplear una notación clockwise en donde se especifican todos simultáneamente. Por ejemplo:

padding: (top)px (right)px (bottom)px (left)px

En caso de XXXX < 1; se ocupa todo el espacio disponible.

El **margin** es la separación del elemento html respecto al resto de elementos. Tiene las mismas dimensiones que el padding: top, right, bottom y left y el mismo tipo de notaciones.

El **border** es un elemento diferenciador del padding. Se trata de un recubrimiento del rectángulo principal hacia dentro. Posee numerosos atributos.

**Atributos**

Relacionados con el color:

color: XXXX; 🡪 Color de la letra

background-color: XXXX; color del padding

Relacionados con la letra:

font-size: XXXX; 🡪 Tamaño de la letra

font-family: XXXX, YYYY, ZZZZ;

Se pueden importar fuentes a través de un <link href>

YYYY se emplearía en caso de no existir XXXX; ZZZZ semejante.

Relacionados con el borde:

border-color:

border-width:

border-style: *solid*/*dotted* (ejemplo);

border-radius: XXXX%; 🡪 Define el radio del borde; se pueden poner magnitudes absolutas o incluso relativas

Se deben diferenciar las **variables absolutas**, como píxels, milímetros, etc.; y las **relativas**, cuyo valor depende de otro atributo, como podría ser el tamaño del margin en función del tamaño de la fuente, como e/m.

**Plantillas (Grids)**

Las plantillas son la combinación de contenedores estilizados que contienen otros elementos html. Se debe recordar la diferencia entre contenedores block e inline, que tendrán una ocupación del espacio diferente.

Para crear una plantilla debe especificarse dentro de la clase:

<style>

.container {

display: grid; 🡪 Crea la plantilla. Si se pusiera flex, se permitiría que los elementos ocuparan el espacio de mayor optimización; esto es, el mayor espacio posible. De forma predeterminada ocuparán las filas, pero se puede indicar que ocupen las columnas con *flex-direction*.

grid-template-columns: 100 px 100px; 🡪 Divide la plantilla en 2 columnas de 100 pxs. Si se pusiera 50% lo dividiría en dos secciones iguales. *fr* dividiría ocuparía la fracción asignada del total de *fr*. *auto* ocupa el espacio sin padding.

Grid-template-rows: 50px 50px; 🡪 Divide el espacio de la clase en 2 filas de 50px. Para evitar la repetición se puede usar el atributo repeat:

grid-template-rows: **repeat**(2, 50px) 🡪 Repite 2 veces el espacio de 50 px.

También puede hacerse de tamaño variable, por ejemplo:

grid-template-rows: repeat(**auto-fill**, **minmax**(50px, 1fr)); 🡪 Repite 2 veces una columna con un mínimo de 50px y un máximo que ocupa todo su espacio (1fr). Además, **auto-fill** crea tantas columnas como soliciten el tamaño de los elementos de forma dinámica. Se asemejan mucho a **auto-fit**, solo que este siempre ocupa toda la línea como <div>.

grid-column-gap: 20px 🡪 Deja una columna de espacio del tamaño indicado entre columnas.

grid-row-gap: 20px 🡪 Deja una fila de espacio del tamaño indicado entre filas.

grid-gap: 10px 20px 🡪 Notación **clockwise** que indica rows-columns.

grid-template-areas: 🡪 Permite clasificar cada elemento del grid en el tipo de área que le especifiques. Se debe respetar el número de columnas y filas.

“header header”

“body advert”;

}

</style>

Puedes determinar el número de columnas que ocupa un determinado elemento mediante:

grid-column: 2/3; 🡪 Indica que ocuparía desde el gap 2 hasta el gap 3

Cuadrículas - Aprende sobre desarrollo web | MDN

Los *gutters* equivalen a los *gaps.*

Se puede realizar la misma función con grid-row: ;

Los elementos del grid ocupan de forma predeterminada todo el espacio de la plantilla. Se puede reducir el espacio ocupado a través de:

justify-self: center; 🡪 Elimina el padding y sitúa el elemento en el centro (horizontal) de la casilla.

align-self: center; 🡪 Alinea el elemento y su padding verticalmente al centro.

Este método necesita ser llamado como una clase independiente o por ID; pero se puede aplicar este espaciado directamente sobre toda la clase .container- del grid, mediante:

justify-items: center;

align-items: center;

Para asignar un elemento html a un área definida del grid, se emplea el atributo:

grid-area: NOMBRE DE LA CASILLA (ej: header);

Este atributo también permite asignar un elemento en la plantilla según las *gaps* que se le proporcionen. Ejemplo:

grid-area: 1/1/2/1; 🡪 El elemento comienza en el gap 1-1 y finaliza en el 2/1, correspondiéndose este 2 a las filas.

**Condicionales**

Se pueden poner restricciones dentro del <style> a través del elemento @media.

<style>

@media (min-width: 300px) { 🡪 Se ejecuta cuando se cumple la condición

}

</style>

25/09/2020: 3 horas