CSS establece el “estilo” de la pagina.

Iniciamos con configuraciones ya vistas, pero lo novedoso es agregar un tag <style> al <head>.

Dentro del tag <style>

Podemos plantear diferentes configuraciones, como por ejemplo centrar un titulo:

h1{

text-align: center;

}

Si deseo lo mismo por ejemplo para h2 y p. Puede configurarlo como h1,h2,p { … }

Como va a haber mas de un estilo seguramente, es mejor crear los diferentes estilos en diferentes archivos .css.

Ahora defino dentro de mi head, un nuevo tag <link> del modo:  
<link rel="stylesheet" href="styles.css">

Para que la pagina luzca similar en móvil, laptop o escritorio. Se agrega dentro de <head> un nuevo <meta/>  
meta < name=”viewport” content="width=device-width, initial-scale=1.0" />

Para modificar el color de fondo del body, dentro del style.css agrego:

body {

  background-color: brown;

}

El <div> es una **etiqueta contenedora genérica** que se usa para **agrupar elementos**.

Se utiliza principalmente para **diseño y estructura visual**. Por ejemplo:

* Agrupar varias etiquetas para aplicarles estilos con CSS.
* Dividir la página en columnas, menús, contenedores, etc.
* Organizar el contenido cuando no encaja en una etiqueta más específica.

A un <div> le puedo especificar un id.

En el .css puedo utilizar el id selector para seleccionar un elemento con el id especifico y darle un estilo.

La sintaxis es:

#id{

…

}

El atributo width: value; Permite especificar el ancho de un elemento. Puede ser en px o %.

Los comentarios en css son del tipo /\* … \*/

Para centrar un elemento puedo hacer uso de

margin-left = margin-right : auto;

Un class-selector es definido por un punto y el nombre de clase.

.class-name{

}

Para editar de esta forma, el elemento debe tener el atributo class definido.

    <div class="menu">

Puedo hacer que el background de mi <body> sea una imagen con el atributo background-image.

body {

  background-image:url(https://cdn.freecodecamp.org/curriculum/css-cafe/beans.jpg);

}

Para añadir ítems de menú puedo hacer uso del tag <article></article>.

<article> contiene elementos con información relacionada.

Por ejemplo, elementos de article pueden ser <p>.

Puedo tambien modificar el text-align: right; o left.

Si tengo un <article> que contiene <p>, cada <p> funciona como un inline-block, por lo que aparecerá cada uno debajo del otro. Para que queden uno al lado del otro, necesitamos asignar el atributo class del article, y luego mediante css especificamos que elementos del article queremos que estén uno al lado del otro.

.item p {

  display: inline-block;

}

Para que una misma configuración aplique para dos clases puedo hacer:

.class1, .class2{

}

Puedo generar espacio entre el borde de mi elemento y el texto mediante padding-left: 20px, y padding-right:20px;

De igual forma con padding-top y padding-bottom

Si modifica los 4 lados con el mismo padding: 20 px, directamente puedo hacer padding:20px;

Puedo definir un ancho máximo de un elemento con max-width: valuemax px; Esto para pantallas más “anchas”.

Puedo cambiar la fuente de un elemento con font-family: algunafuente;

Además puedo agregar una fuente “extra por si las dudas” agregándole al atributo Font-family con una coma y el nombre.

font-family: algunafuente, fuente2;

Puedo establecer texto en cursiva mediante:

font-style: italic;

Puedo determinar el tamaño de la fuente con font-size: 40px; por ejemplo.

Puedo utilizar un tag <hr> para dividir con una línea diferentes secciones de la pagina.

En el .css puedo modiicar hr de las formas:

hr{

height: 3px;

background-color: brown;

border-color: brown;

}

Puedo disminuir el espacio ocupante por debajo y superior de cada elemento mediante margin-top: 5px; o margin-bottom: 5px;

Para que un link no clickeado aun no sea azul, puedo modificarlo de la siguiente forma:

a{

color: black;

}

Si el link ya fue visitado:  
a:visited{

  color: grey;

}

Si deseo que cambie el color del link cuando el mouse está sobre el link:  
a:hover{

  color: brown;

}

Si deseo que cambie el color del link cuando clickeo sobre el:  
a:active{

  color: white;

}

Para centrar una imagen:  
img {

  display: block;

  margin-left: auto;

  margin-right: auto;

  margin-top: -25px;

}

Los **tags** de html, le dan una estructura a la página. Los tags se utilizan para agregar imágenes, botones, texto, etc.

El tag <h1> texto </h1>, permite colocar texto dentro.

Tags h1 tienen mayor prioridad respecto a h2,h3,..

Los tags <p> Parrafo </p>, se utilizan para colocar párrafos.

Para realizar comentarios utilizo <!-- … -->

HTML tiene elementos que identifican diferentes áreas.

El elemento <main> se usa para representar el contenido principal del cuerpo (<body>) de un documento HTML. El contenido dentro de <main> debe ser **único en ese documento** y **no debe repetirse** en otras partes del mismo documento. Los tags dentro de main van anidados, hay un nesting. Se utilizan dos espacios mas a la derecha respecto al elemento que los contiene, este espacio es llamado **indentacion.**

Para añadir imágenes a mi HTMl uso el tag <img> sin un closing tag.

Le puedo agregar un atributo al tag <img> del tipo src (source) para indicar la url de la imagen.

Todas las imágenes deberían tener un atributo alt. Este atributo de texto es usado por si la imagen falla al cargar, en ese caso se despliega el texto.

Para linkear a otro sitio web, uso el tag <a href=”url”> </a>.

Dentro de este texto, se coloca un “texto de link”. El texto se coloca llano, sin “”. Ademas, puedo colocar un prefijo al tag <a></a> y un postfijo. Ejemplo:

Ver mas <a href=”url”> </a> en este link

Se puede añadir todo un tag <a></a> dentro de un párrafo (tag p).

Creo que los links clickeados en lo que hicimos abren un link en mi sitio web.

Ahora veremos como abrir una nueva pestaña con el atributo target dentro del tag a. Si el atributo target =”\_blank” abre una nueva ventaña del navegador.

Además de convertir un texto a link, podemos convertir otros elementos en link anidándolos dentro de un tag a.

Como por ejemplo <a> <img ….> </a>.

Para añadir más contenido al cuerpo, se hace uso del tag section para definir secciones. Las subsecciones deben respetar la sintaxis de los tags superiores.

Para crear listas desordenadas de ítems hay q hacer uso del tag ul.

Dentro de cada tag <ul>, se anidan tags <li>.

El elemento <figure> representa contenido independiente y te permite asociar una imagen con una leyenda. Ademas, puedo agregar un tag <figcaption> para proveer una descripción de la imagen.

Ejemplo:

<figure>

          <img …>

          <figcaption> .. </figcaption>

</figure>

Para hacer enfasis en alguna palabra del texto de figcaption, puedo hacer uso del tag <em> palabra </em>.

Puedo tambien hacer uso de una lista ordenada con el tag <ol>, los ítems son numerados cuando se muestran. La sintaxis de los ítems es la misma que en <ul>.

Para resaltar un texto o una palabra en negrita, se hace uso del tag <strong>. Colocando el texto a resaltar dentro.

Existe el tag <form> que se utiliza para recolectar información del usuario. El atributo action **indica a dónde se enviarán los datos** del formulario cuando el usuario lo envíe (por ejemplo, al apretar un botón "Enviar").

El tag input permite la recolección de la información, es un elemento void y no requiere tags de cierre. Va anidado dentro del form.

A cada input se le puede asignar un type.

Ejemplo: <input type="text">.

A su vez, a cada input se le debe asignar un name. Para luego acceder a dicha información cargada por el usuario.

A su vez, el texto placeholder (texto de marcador o texto de ayuda) se usa para dar a las personas una pista sobre **qué tipo de información** deben ingresar en un campo de entrada (<input>).

Para que un usuario no cargue un formulario con campos esenciales vacíos, se utiliza en el input el atributo required, así llanamente.

Dentro de un formulario se puede agregar un elemento con tag <button>NombreBoton</button>. El comportamiento default de clickear un botón sin atributos es subir el formulario a la localización especificada en el atributo action del tag <form>.

Para ser específicos, se puede añadir el atributo type al button, con el valor submit, para dejar en claro su uso.

<button type="submit">Submit</button>

Los radio buttons, se usan para pregungas donde queres una sola respuesta de varias.

<input type=”radio”> nombre

Los elementos <label> se usan para **asociar el texto descriptivo** con un elemento <input>, especialmente para **tecnologías de asistencia** (como lectores de pantalla para personas con discapacidad visual).

Le da contexto al campo de entrada, y además permite que el usuario pueda hacer clic en el texto del <label> para seleccionar el campo (muy útil con checkbox y radio).

A cada elemento le puedo asignar un id como atributo, el id es único en toda la pagina.

Para que en múltiples radio buttons solo pueda escogerse uno solo, se le agrega a cada radio button del conjunto un atributo name, idéntico para todos.

Cuando presionas un radio button y submiteas el formulario, la información del botón será su nombre y su atributo value. Pero si el button no tiene un atributo value definido, se incluiría el name = on, lo que no es útil. Por eso es útil definirle un atributo value a cada botón. Se suele asignar al value el id del botón.

El elemento <fieldset> es utilizado para agrupar inputs y labels dentro de un formulario. Los elementos del fieldset aparecen en una sola línea.

Dentro de un <fieldset>, puede agregar un elemento <legend>Text</legend>.

Sirve para dar contexto acerco de lo que el usuario debe ingresar en esa parte del formulario.

Es posible también crear inpunts con el type=”checkbox” para preguntas con mas de una respuesta.

Existe otra manera de asociar un texto a un input mediante un <label> con el atributo for=id\_input. Y dentro del tag input colocar el texto asociado.

<input id="loving" type="checkbox">

<label for="loving">Loving</label>

A cada checkbox le puedo asociar mediante su atributo name, al “formulario de checkboxs” que pertenece.

<input id="loving" type="checkbox" name="personality">

Al igual que un radio button, la información recolectada de un formulario de checbox es name/value.

Tanto los radio buttons como checkbox, podes elegir uno de ellos seleccionado por defecto mediante el atributo checked en cada input.

Un elemento <footer> es utilizado para dar información acerca del autor, copyright, etc. Se suele usar para ser footer de una sección o del documento.

Debajo del tag <html> se coloca un tag <head> donde se coloca dentro de el metadata, titulo del documento, scrips, etc.

Dentro de <head>, se puede agregar un tag <title></title>.

Todo el contenido de la página está anidado dentro de un elemento <html>.  
El elemento <html> es el **elemento raíz** de una página HTML y **envuelve todo el contenido** de la página.

También podés **especificar el idioma** de tu página agregando el atributo lang al elemento <html>. En mi caso utilice Lang=”en”

En todo archivo html, se debe colocar en la primer línea:

<!DOCTYPE html>.

Dentro del <head>, si declaro

<meta charset="utf-8">

Le digo al navegador como codificar caracteres.

Añado la siguiente configuración en el <head> de mi <html>:  
<head>

<link href=”styles.css” rel=”stylesheet”>

<meta charset=”utf-8”>

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

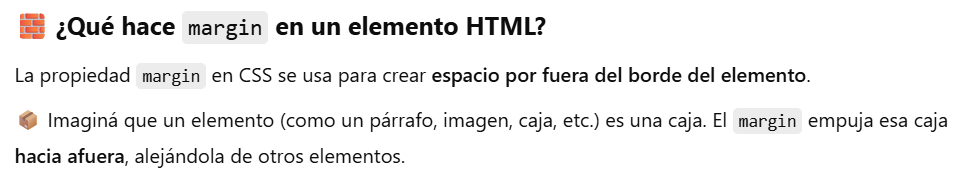
<title>Colored Markers</title>

</head>

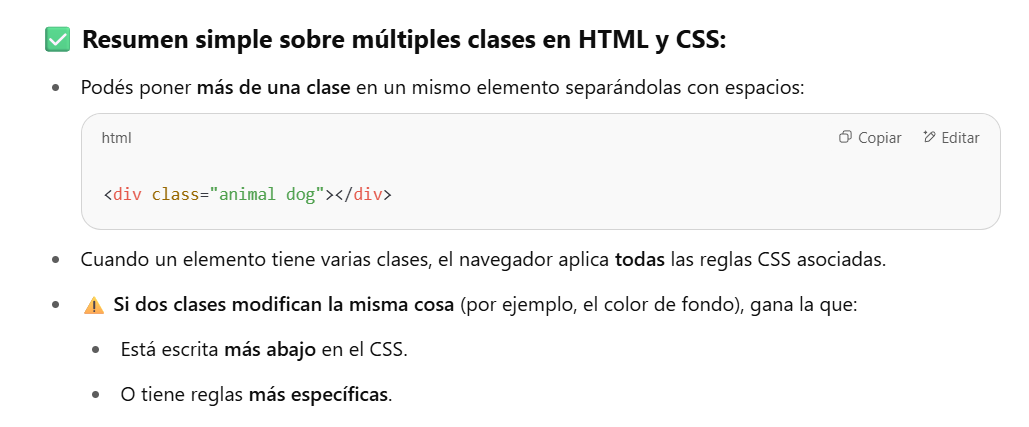
El tag <link> sirve para “estilear” la página HTML con un archivo .css externo.

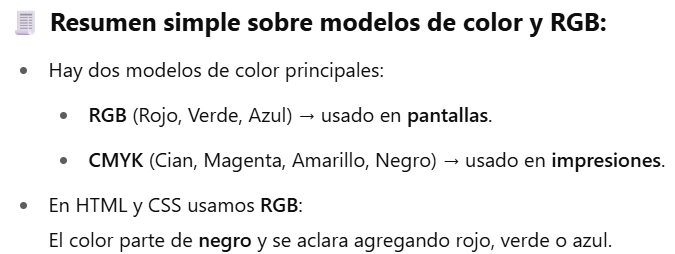
El tag <meta charset…> define como se codifican los caracteres.

El tag <meta name ..> permite que el ancho de la página se ajuste al dispositivo, y no se escale el contenido por defecto. Hace a la página **responsiva.**



Un elemento puede tener **dos clases**.





RGB es una **función** que recibe un índice de rojo, un índice de verde y un índice azul y produce un color.

Cada input está comprendida entre 0 y 255.

¿Cómo impactan los colores en el desarrollo web?

Usar dos colores muy brillantes y opuestos (como rojo y cyan) uno al lado del otro sin medida puede ser visualmente chocante y hacer difícil la lectura. Por eso, mejor elegí un color principal para la mayoría del diseño y usá el color opuesto solo para resaltar detalles importantes.

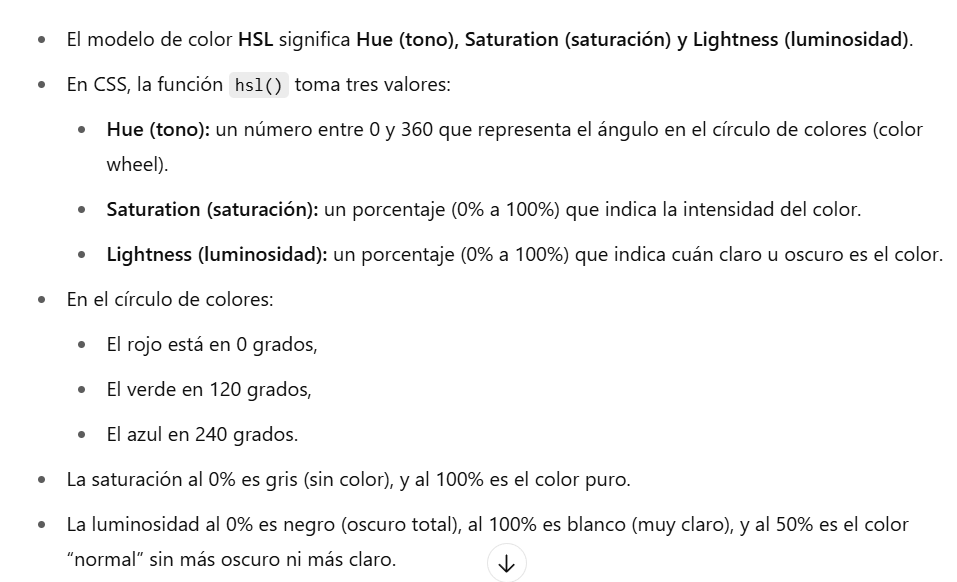
Otra forma de representar color es mediante el siguiente formato:  
#PAR1PAR2PAR3, donde PAR I = valor hexadecimal de 0 a FF.

Donde PAR1 representa dicho para valor rojo, PAR2 para verde y PAR3 para azul.

El rojo es entonces: #FF0000

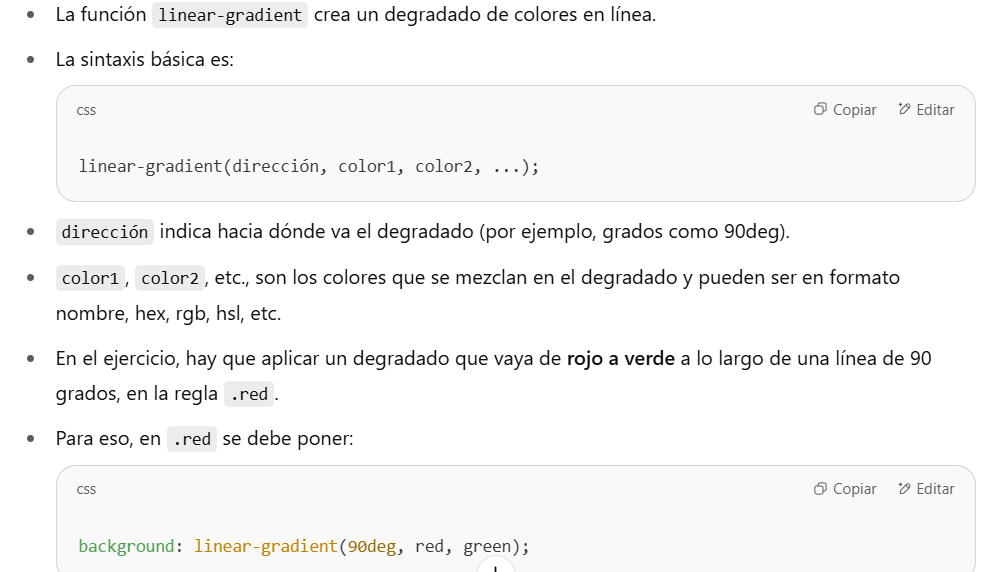
El azul: #0000FF

# Otro modelo:



En .css tenemos la función **hsv(value, value%, value%);** esto devuelve un color.

# Función **linear-gradiente()** como background:



A su vez, a cada color le puedo agregar un porcentaje de “ocupación”

.red {

background: linear-gradient(90deg, rgb(255, 0, 0) 75%, rgb(0, 255, 0), rgb(0, 0, 255));

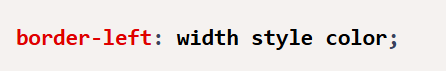
}

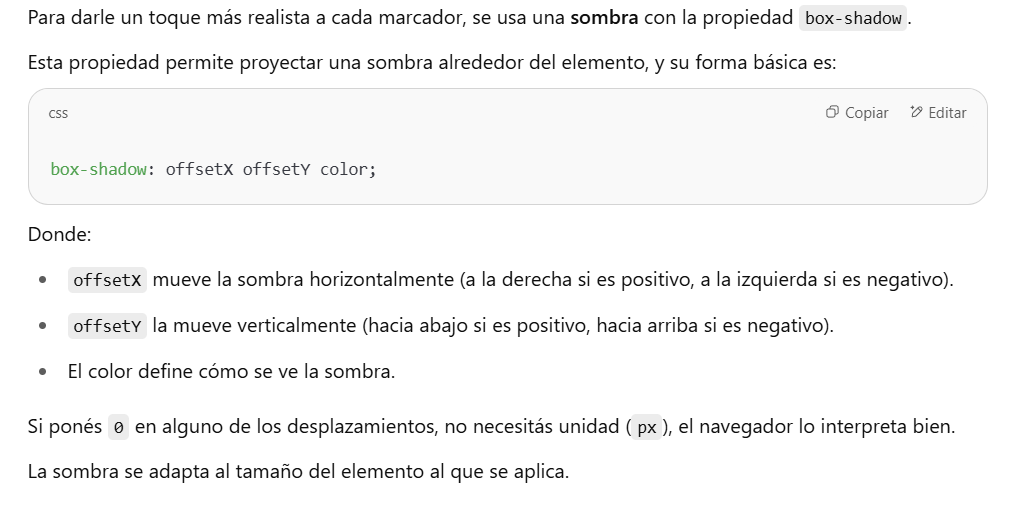
Por default, si no se define la dirección de linear-gradient esa es 180deg.

Conocemos la propiedad opacidad u **opacity**, valor numérico entre 0 y 1 que nos dice que tan opaco (que tan visible a través de él) es un componente.

Existe otra forma de determina la opacidad de un elemento con background-color y rgba.



Una forma de configurar rápido el borde de un elemento:  




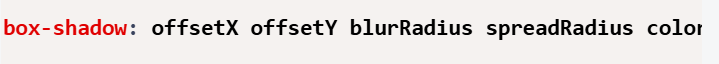
Esta configuración genera sombras “rectangulares” o “cuadradas”, para hacerlas más reales se utiliza la configuración:



Donde solo varía el tercer parámetro blurRadius, el cual es parámetro numérico con UNIDAD válida para css.

(En la configuración anterior, blurRadius estaba incluido implícitamente con un valor cero).

Otra configuración más:

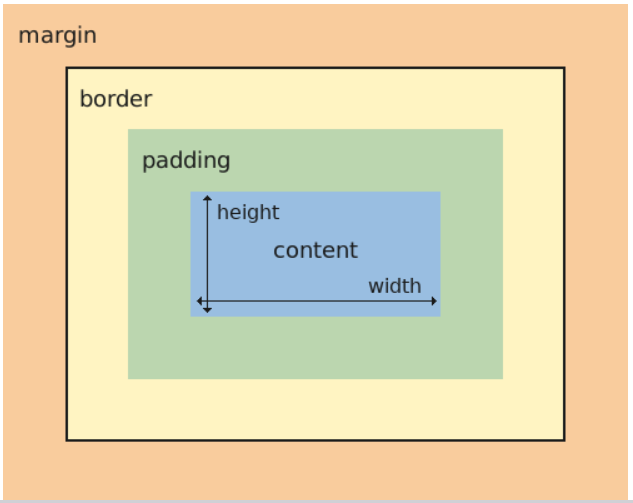


Agrega la opción spreadRadius, mismo parámetro que blurRadius, genera una sombra del estilo:



Si pongo 0, 0, y algún valor para blurRadius y 0 para spreadRadius y agrega un color con rgba o hsla queda cheto.

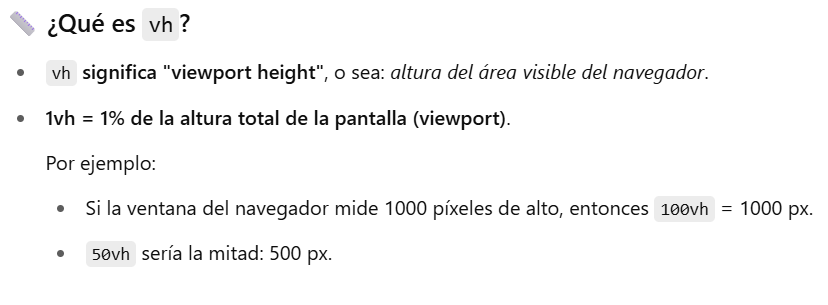
El model box de CSS plantea que la disposición de un elemento HTML es la siguiente:



Propiedad filter:blur(Xpx);

Se utiliza para “desenfocar”.

Con la propiedad transform: rotate(Xdeg). Podes rotar un elemento



Con esta configuración:

body {

  width: 100%;

  height: 100vh;

  margin: 0;

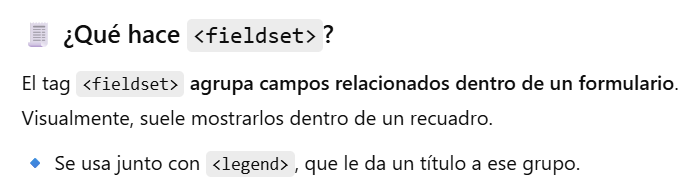
}

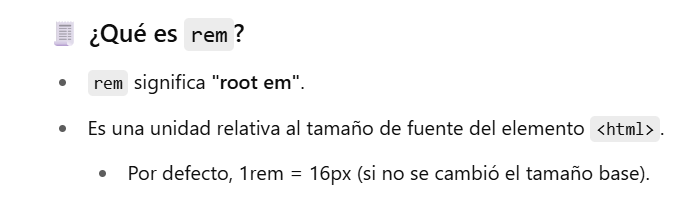
Nos deshacemos de la barra horizontal inferior.

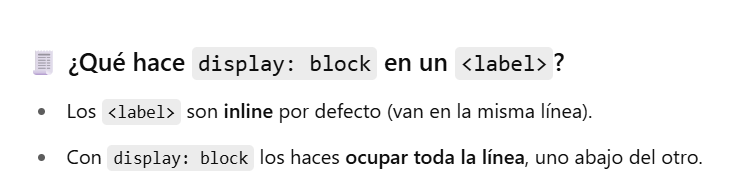
En html, la propiedad **color** de un elemento hace referencia al **color del texto del elemento.**

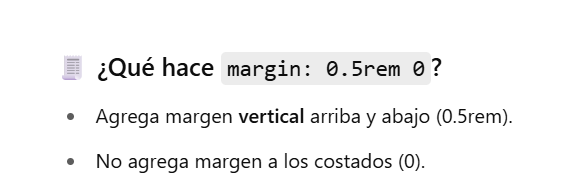
El atributo **action** del tag <form>, permite que cuando se envíe el formulario, **los datos del formulario se envíen como una solicitud HTTP** al sitio indicado en la URL del atributo action. Por defecto la petición es de POST, puedes modificar el tipo de acción mediante el atributo **method.**

Debemos recordar que el dentro de un <form>, podemos dividirlo en direrentes “secciones” mediante el tag <fieldset>.









Recordemos que cada **<input>** debe especificar su type, por defecto es text.

El type = ”email” solo permite emails con @ y . como dominio.

El type = “password” matchea la input a \* y advierte si el sitio no utiliza HTTPS.

Para enviar el formulario, luego de todos los <fieldset> dentro de debemos agregar un <input> con un atributo type = “submit”. Luego mediante el atributo value podemos darle un nombre diferente si queremos.

Recordar para enviar un formulario no vacía, agregar atributo **required** para los <input> necesarios.

Algunos <inputs> llevan validaciones de acuerdo a su type, como lo es “email”.

Si queremos agregar por ejemlo para un campo una longitud mínima, podemos usar el atributo minlength para ese <input>.

A su vez, podemos agregar una expresión regular mediante el atributo **pattern**. Por ejemplo:

<input id="new-password" type="password" pattern="[a-z0-5]{8,}" required />

Recordamos que para **seleccionar una opción entre múltiples** utilizamos <input> del type=”radio”.

A su vez, para relacionar todos los botones, debemos agregarle el mismo atributo **name**.

Como a continuación:  
<fieldset>

        <label><input type="radio" name="account-type"/> Personal</label>

        <label><input type="radio" name="account-type"/> Business</label>

    </fieldset>

Para darle un contexto a la selección de la opción, utilizamos el tag <Legend>texto</legend>.

Además, para seleccionar una opción por default, utilizamos el atributo **checked** en algún <input>.

Ejemplo:

 <fieldset>

        <legend>Account type (required)</legend>

        <label><input type="radio" name="account-type" checked/> Personal</label>

        <label><input type="radio" name="account-type" /> Business</label>

   </fieldset>

Para **marcar una única opción** hemos visto de usar **<input> del type=”checkbox”** con el atributo **required**. Aquí el ejemplo:

<label for="terms-and-conditions">

          <input id="terms-and-conditions" type="checkbox" required /> I accept the <a href="https://www.freecodecamp.org/news/terms-of-service/">terms and conditions</a>

</label>

También existen **<input> del type=”file”**, es útil por ejemplo si queremos cargar imágenes.

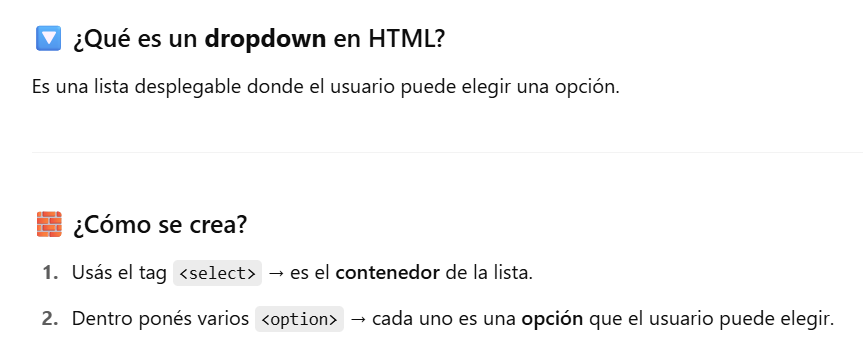
A su vez, también existen <input> del type=”number”, combinándolo con un atributo min y max queda una forma elegante.  
Ejemplo de estos dos últimos puntos:

<fieldset>

        <label>Upload a profile picture: <input type="file" /></label>

        <label>Input your age (years): <input type="number" min="13" max="120"/></label>

      </fieldset>



Si los **<option> no tienen un atribuo value**, la información enviada al servidor no será muy útil, se enviará el texto de <option>. Es recomendable agregar el atributo **value** significativo.

El elemento **<textarea>** actúa de forma similar al <input> del type=”text”, pero permite ingresar texto con múltiples líneas.

<textarea id="bio" rows="3" cols="30" placeholder="I like coding on the beach..."></textarea>

Recordatorio, para elegir una fuente y su tamaño:

font-family: Tahoma;

font-size: 16px;

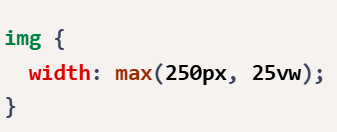
En .css Podemos seleccionar un elemento de la forma:

tipoElemento[type=”typeOfElementoX”]{

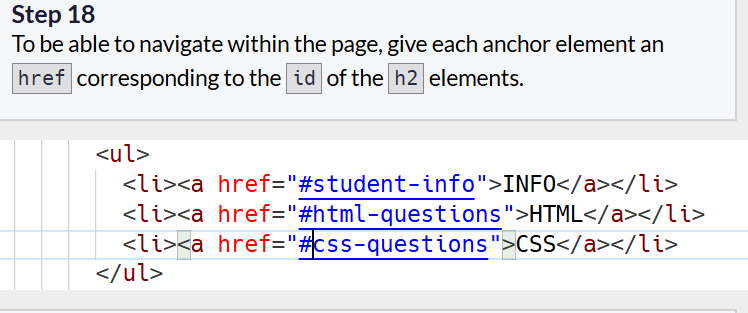
...

}

En css podemos elegir el width de una imagen con:



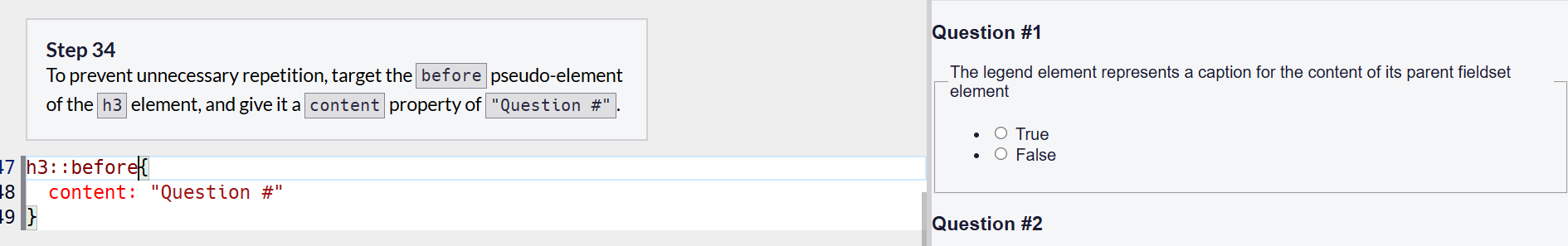
Para habilitar la **navegación** dentro de una página, hacemos uso del tag **<nav>.** Dentro de el colocamos una unordered list, cada elemento de la lista debe estar dentro de un tag <a>.



Vemos que cada **<li> tiene una referencia hacia el id de un elemento.**

Existe el selector elemento\_padre **>** elemento\_descendencia, para seleccionar hijos de un padre.

Para que los lectores de pantalla entiendan de qué trata una región (role="region"), se le debe poner un nombre. Una forma de hacerlo es colocando un título (como un <h2>) dentro de la región y usando **aria-labelledby** para conectar esa región con el título. Así, el lector de pantalla puede decir por ejemplo: “Región: Últimas Noticias”

Puedo utilizar **::before** en un elemento:  


flex-wrap: wrap; le dice a un **contenedor flex** que sus elementos **pueden "salirse de la fila" y pasar a la siguiente línea** si no hay suficiente espacio horizontal.