w3 validator página para validar que tengamos todas las etiquetas semánticas.

<!DOCTYPE html>(version html)

<html lang=”es”> (lenguaje)

<head> (configuración de la página, comunicación con el navegador)

<meta charset=”UTF-8”>

<title>Mi nuevo doc</title> (título que aparece en la pestaña del navegador)

</head>

<body>

<header>

<nav>

<ul>

<li></li>

</ul>

</nav> (barra de navegación: se hace con listas)

<h1></h1>

<p></p>

</header>

<main>(contenido principal del documento)

<section>

<article></article>

<div></div>

</section>(página dividida en secciones)

</main>

<footer></footer>

</body>

</html>

<span></span> Para agrupar objetos y darle estilo al mismo.

LISTAS

Desordenada:<ul></ul>. Atributo type=(circle, disc, none, square)

Ordenada: <ol></ol>. Atributo type=(1, A, I). Atributo start=”20”. Atributo reversed para que vaya de atrás para adelante.

<dl>

<dt></dt> (término)

<dd></dd> (descripción, definición)

</dl>

ENLACE

<a href=”” target=”\_blank”></a>

ANCLAS

Atributo id en la sección que queremos identificar. Luego, con una etiqueta a, en el href se pone #nombreDelID.

Para que el scroll no sea tan brusco, se coloca en la etiqueta html, el atributo style=scroll-behavior:smooth”.

IMAGEN

<img src=”ruta” alt=”descripción de la imagen”>

ENLAZAR CSS

<link href=”css/style.css” rel=”stylesheet”> (se pone en cada html que queramos llamar al css).

SELECTOR DE CLASE

HTML: class=”nombre”

CSS: .nombre{

}

SELECTOR DESCENDENTE

div p{

} (le aplica estilo a todos los parrafos que esten en un div)

div>p{

} (le aplica estilo a todos los hijos directos que estén en un div).

SELECTOR DE ETIQUETA

h1,h2{

} (para agrupar selectores que comparten estilos)

CSS IMÁGENES DE FONDO

* background-image: url(); (para imagen de fondo)
* background-size: (tamaño)
  + contain(proporciona la imagen en alto y ancho para que aparezca completa
  + cover (completa el espacio del contenedor, aunque la imagen no se vea completa).
  + También se pueden usar pixeles.
* background-repeat:no-repeat (para que la imagen no se repita).
* background-position: 20px 50px (con el primer valor, me muevo en el eje x. Si quiero ir para la izquierda, tengo que usar números negativos. Con el segundo valor nos movemos en el eje y) (También se puede usar center, top, bottom, left, right)
* background-attachment: fixed (para que quede fijo), scroll.

COLORES:

* color:red;
* color:#fff;
* color: rgb(255,255,0);
* color: rgba(255,255,0,0.2); (el último es de un valor de 0 a 1 que le da transparencia al elemento).

FUENTES

* Tipografía: font-family: Verdana,Montserrat;.
* Tamaño: font-size: 15px;
* font-style:italic;
* font-weight:bold;
* text-align;
* text-decoration:none (elimina el subrayado de un enlace); underline; overline y line-through.
* text-transform:none; uppercase; lowercase.
* line-height: 23px; (interlineado, suele ser 8 más que el size).
* Web: Google Fonts (se aconseja no utilizar más de dos tipos de fuentes en un proyecto).
  + Entrar a la página. Seleccionar la familia deseada. Buscar el estilo deseado de la misma y seleccionarlo (puede ser más de uno). Nos va a dar una etiqueta link que la debemos poner en el head del html. Con esta forma, tenemos que colocar la etiqueta link en cada html que tengamos.
  + Entrar a la página. Seleccionar la familia deseada. Buscar el estilo deseado de la misma y seleccionarlo (puede ser más de uno). Seleccionar @import y copiar el código que nos da sin la etiqueta style y se coloca arriba de todo en el CSS. Una vez que hayamos enlazado, debemos colocar la propiedad font-family en los selectores que queramos. OPCIÓN RECOMENDADA.
* Local: descargamos la familia. Creamos una carpeta fonts en la raíz de nuestro proyecto y ahí descomprimimos el zip que se descargó. Luego en el css, ponemos la propiedad @font-face{url(“../fonts/Roboto/Roboto-light.ttf”); font-family: “Roboto”}.

ICONOS:

Al ingresar a FontAwesome, buscamos el icono que queremos y lo seleccionamos. Copiamos la etiqueta i que nos aparece. Para que aparezca el ícono, debemos enlazar la librería mediante el cdn. El color y el tamaño del icono se cambia con las propiedades de las letras.

BOOTSTRAP es una librería con código hecho. Podemos elegir lo que queremos y copiarlo en nuestro html, enlazando el cdn de la librería de bootstrap al mismo.

PROPIEDAD DISPLAY:

display: inline; inline-block (se coloca uno al lado del otro como los elementos en línea pero recibe las propiedades de caja); block; none(oculto);flex.

PROPIEDADES DE CAJA

* PADDING: espacio entre el contenido y el borde. Con esta propiedad le damos el alto al elemento, en vez de utilizar height (con el height, la página queda muy dura).
* BORDER: 2px solid red
* BORDER-RADIUS: para bordear los ángulos.
* MARGIN: espacio que hay entre los elementos. Se utiliza para separar hermanos.
* BOX-SIZING: border-box (para que respete el ancho indicado y meta dentro de ese número el padding y el border). ESTO VA EN EL SELECTOR UNIVERSAL JUNTO CON PADDING:0 Y BORDER:0.

POSICIÓN

Por defecto, todos los elementos vienen con position:static.

* position:relative. Para mover el elemento sin que los otros tomen su posición. El movimiento va a ser relativo al espacio que ocupa.
* position:absolute. Sale del flujo de html y los otros elementos ocupan su lugar. Se va a mover en relación a la página. Para cercarlo, ponemos al contenedor en posición relativa.
* position:fixed. Es para que quede fijo cuando se scrollea. Se le debe dar la posición (top, bottom, right, left).
* Posición:sticky. El elemento es posición relativa hasta que al scrollear, llega a la posición que le decimos. Siempre va con la propiedad top para indicar cuando pasa a ser fijo. Solo se puede usar en los padres, no en los hijos.

Z-INDEX

Se trabaja sobre la dimensión z. Con números negativos se manda para atrás y con positivos, se trae adelante.

CSS AVANZADO

* Se almacena un valor en un variable para poder llamarlo después, y si hay que modificar en todas las etiquetas, solo modificamos el valor de la variable.

:root{

--colorPrimario: orange;

}

Para declararla, se pone “root” porque queremos que la variable sea global, que aplique a todo.

El nombre de la variable se declara con dos guiones y el nombre de la variable.

Para llamarla se utiliza la función var() y como argumento se pasa el nombre de la variable.

h1{

color:var(--colorPrimario);

}

* Para simplificar el archivo css y que no sea tan largo, podemos hacer varios css agrupando las propiedades y luego importarlo en el css principal, que va a ser el que va a estar linkeado al html.

Para importarlo en el css principal se hace así:

@import “nombre.css”;

FORMULARIOS

<form action=”indicarLaRutaDondeSeEnviaElFormulario” method=”get o post”>

(viene con el get por defecto. El get muestra todo lo que se ingresó en la barra de navegación, por lo que no es muy seguro. Se recomienda usar “post”. Si en action ponemos”#”, al enviar nos va a mantener en la misma página)

<label for=”nombre”></label>

(el “for”es para enlazar el label con el input y debe ser igual que el “id” del input. Debe ser una palabra descriptiva de lo que se va a ingresar)

<input type=”text, password, tel o mail” id=”nombre” name=”nombre” placeholder=”Pablo” required></input> (campo de una sola línea)

(id es para enlazarlo con el label. Name es el valor que va a recibir la base de datos junto con el dato que se ingresa, para identificarlo en la misma. Placeholder es para indicar un valor de referencia. Required es para indicar que el valor es requerido; si no está completo, no se envía)

Con el atributo oninvalid=”setCustomValidity(“Por favor, completar correctamente!”)”, se cambia el texto que aparece por defecto cuando el usuario aprieta enviar sin completar un campo requerido.

<label for=”mensaje”>Mensaje</label>

<textarea id=”mensaje”></textarea> (campo de muchas líneas)

<label for=”pais”>Seleccione pais</label>

<select name=”pais” id=”pais”>

<option disabled selected> Seleccione país</option>(para que aparezca como texto predeterminado dentro del campo)

<option value=”Argentina”>Argentina</option>

<option value=”Brasil”>Brasil</option>

</select>

(los atributos name y value son los que van a ir a la base de datos. El value tiene que coincidir con lo que se muestra al usuario)

BOTÓN

<input type=”submit” value=”Enviar”><input>

(type=”submit” indica que es un botón para enviar los datos ingresados. Value indica el valor que queremos mostrar en el botón)(cuando se aprete enviar y un campo requerido no está completo, va a avisar)

<input type=”reset” value=”Resetear><input>

(se usa para crear un botón que borre los campos agregados antes de enviarlos).

SI NECESITAMOS QUE EL BOTÓN SEA UNA IMAGEN O UN ICONO, USAMOS LA ETIQUETA <BUTTON> PARA PODER ANIDARLOS. SI NO SE NECESITA ANIDAR, SE USA INPUT. SI QUEREMOS QUE EL BOTÓN SEA UN ENLACE, USAMOS ETIQUETA <a> Y CON CSS, LO DISFRAZAMOS DE BOTÓN.

</form>

FORMULARIOS II

<fieldset></fieldset>

(se usa dentro del form para dividir al mismo en categorías)

<legend></legend>

(se usa para darle un nombre al fieldset y que el mismo aparezca atravesando la línea del recuadro del fieldset).

**Radiobutton**

<input type=”radio” id= “categoria” name=”nombreCategoria” value=”opcion1”>

<label for= “categoria”>Opcion 1<label>

<input type=”radio” id= “categoria” name=”nombreCategoria” value=”opcion2”>

<label for= “categoria”>Opcion 2<label>

(es para que el usuario seleccione una única opción. Las opciones que se muestran se ponen en el label. Primero va el input y después el label, si queremos que aparezca primero el puntito para seleccionar y después el texto de la opción. El atributo name debe tener el mismo valor en todas las opciones. Si queremos darle estilo al input, debemos anidarlo al label. En value va la información que se envía a la base de datos).

**Checkbox**

<input type=”checkbox” id=”poderes” name=”poderesSuperheroe” value=”volar”></input>

<label for=”poderes”>Volar<label>

<input type=”checkbox”id=”poderes”name=”poderesSuperheroe” value=”superinteligencia”> </input>

<label for=”poderes”>Superinteligencia<label>

(es para que el usuario seleccione varias opciones. Las opciones se ponen en el label. Primero va el input y después el label, si queremos que aparezca primero el puntito para seleccionar y después el texto de la opción.).

Atributo checked: tanto en radio como en checkbox, aparece seleccionado por defecto.

**File**

<input type=”file” accept=”.jpg, .pdf”>

(es para pedirle al usuario que suba un archivo. En el label se pone un texto con el documento que se requiere)

**Date**

<input type=”date” min=”2018-01-31” max=”2018-12-31”>

(despliega un menú de mes para seleccionar una fecha)

<input type=”color” id=”micolor” name=”micolor” value=”red”>

(aparece una paleta de colores para que el usuario seleccione. Con el atributo value, aparece el color que pusimos como predeterminado)

Atributos minlength y maxlength en el input, te limita la cantidad de caracteres a ingresar.

Con el atributo pattern=”[0-9]{3}-[0-9]{5}” dentro del input en el type tel, indica que deben ingresar 3 números antes del icono y 5 después.

SI NECESITAMOS DARLE ESTILO AL INPUT, LO TENEMOS QUE ANIDAR EN EL LABEL, YA QUE EL LABEL PERMITE ESTILOS Y EL INPUT NO.

En la etiqueta <select> podemos distribuir las opciones en categorias agrupadas. Esto se hace con la etiqueta:

<select name=”modalidad” id=”modalidad”>

<optgroup label=”nombreCategoria”>

<option value=”oficina”>Oficina</option>

<option value=”virtual”>Virtual</option>

</optgroup>

</select>

Con el atributo tabindex=”num” en el input, se da un orden a los campos para que el usuario pueda cambiar con el tab.

PSEUDOCLASES

* :visited: aplicar estilos a los enlaces que hayan sido visitados.
* :hover: líneas de css cuando el usuario pase el cursor del mouse sobre un elemento específico.
* :focus: cuando un elemento html tiene el foco del cursor.
* :disabled: se aplica cuando un elemento está deshabilitado, es decir, cuando posee la propiedad disabled.
* :invalid: aplicar estilo a los input que tengan la propiedad required.
* :nth-child(): coincide con uno o más elementos en función de su posición entre un grupo de hermanos. Entre los parentesis va un numero que representa las generaciones.
* :required: especifica que se debe completar un campo de entrada antes de enviar el formulario.
* :valid: cualquier elemento <input> u otro elemento <form> cuyo contenido se valide satisfactoriamente.

PSEUDOELEMENTOS

* ::before: para agregar contenido antes del contenido presente en un elemento determinado. Propiedad content, la cual recibirá la cadena de texto de aquel contenido que queramos insertar.
* ::after: para agregar contenido después. Atributo content.

Las etiquetas <input> no admiten ::after ni ::before. Si queremos usarlos, debemos anidar el input dentro del label.

ANIMACIONES

Propiedad transform:

* Rotate(deg):para girar el elemento. Nº positivo: gira en el sentido de las agujas del reloj. Si lo usamos con :hover, se gira cuando el cursor se para encima solamente.
* Translate(): mover el elemento de un lugar al otro. Si utilizamos un único valor, se moverá hacia la derecha el elemento, si el número es negativo se moverá a la izquierda. Si se coloca un segundo valor, podremos mover el elemento de manera vertical.
* Scale(): podemos modificar el tamaño de un elemento.
* Skew(deg): convierte al elemento en un romboide.

Propiedad transition: recibe dos valores (la propiedad que queremos transicionar y el tiempo). Va en la regla que no tiene el hover.

KEYFRAME

Para que la animacion esté siempre.

@keyframe nombre{

0%{

Estado inicial de la animación.

}

100%{

Estado final de la animación.

}

}

Sobre la regla de estilo del elemento al que queremos aplicarle la animacion se utiliza la propiedad animation para llamar la animacion.

animation: nombreAnimacion segundosDuracion infinite (que se repita) alternate(para que retroceda y no se vea cortada) linear segundosDelay(para que empiece un poco despues de que cargue la pagina)

animation: nombre 2s infinite alternate linear 2s

VIEWPORT

<meta name=”viewport” content=”width=device-width, initial-scale=1”> (dentro de la etiqueta head para configurar el viewport).

MEDIDAS RELATIVAS:

* Porcentaje: el contexto va a ser su contenedor directo.
* vh o vw: se comportan igual que el porcentaje, salvo que el contexto que va a tomar es el viewport.
* Em: relativo al font-size del padre. Si lo utilizamos en una propiedad que no sea font-size, tomará para el cálculo el font-size que tenga el elemento que estamos modificando.
* Rem: el contexto es el font-size del html que por defecto es 16px.

MEDIA QUERIES

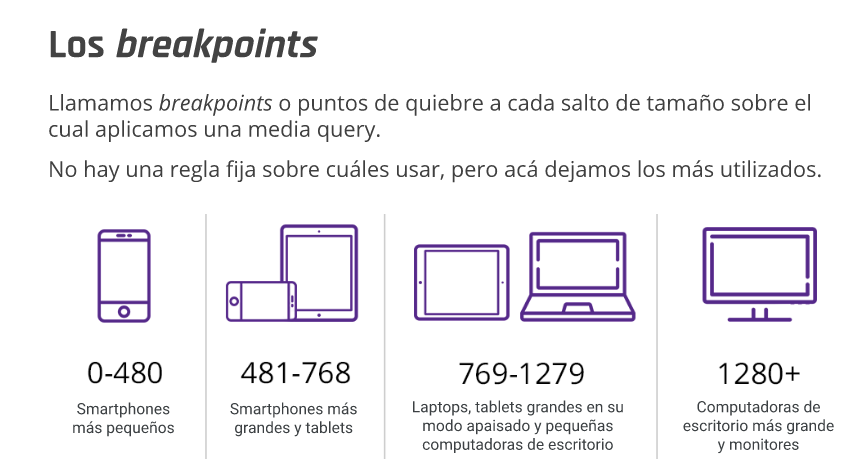
Regla de estilo que contendrá un conjunto de otras reglas que se aplicarán si y sólo si se cumple una condición determinada.

@media “tipo” and (min-width/max-width: Npx){

}

Tipos:

* Print: solamente si el usuario quiere imprimir el sitio web.
* All: todo tipo de media.
* Screen: para pantallas de computadoras, tablets y telefonos.
* Speech: para lectores de pantalla.
* Si no se pone el tipo, es “all” por defecto.



Usar poco position absolute (reemplazarlo por flex), poco margin (reemplazarlo por el padding), medidas relativas.

FLEXBOX

La mayoría de las propiedades se declaran sobre el contenedor padre de los elementos que queremos organizar.

Ejes

* Eje principal o main axis: eje por defecto. Main axis start (inicio del eje. Por defecto a la izquierda) y main axis end (fin de eje. Por defecto a la derecha).
* Eje transversal o cross axis. Cross axis start (por defecto el superior) y cross axis end (por defecto el inferior).

Propiedades

* **display:flex:** en principio esto hará que los elementos queden uno al lado del otro.
* **flex-wrap:wrap:** lograr que las cajas hijas conserven su width, ubicándose en una nueva línea cuando no haya más lugar.
* **flex-direction:** orientación del eje principal. Por defecto, viene con el valor row, lo que define que el eje principal será el horizontal. Flex-direction:column para que el eje principal sea el vertical. Flex-direction:row-reverse: eje principal horizontal con punto de inicio en la derecha. Flex-direction: column-reverse: eje principal vertical con punto de inicio inferior.

Alinear los elementos

* Justify-content (main axis):
  + Flex-start (defecto), se ubican desde el inicio del eje al final.
  + Flex-end, se ubican desde el fin del eje al inicio.
  + Center.
  + Space-between, se distribuyen de manera uniforme, desde el inicio al fin, repartiendo el espacio libre para separar los mismos. Tanto el primer como el último ítem, quedan alineados con el inicio y el fin del eje.
  + Space-around, se distribuyen de manera uniforme, repartiéndose el espacio libre entre todos los elementos. Del espacio que le toque a cada elemento, la mitad irá a la derecha y la mitad a la izquierda del mismo (arriba y abajo si están encolumnados). Se genera también un espacio entre el primer y el último ítem en relación con el inicio y el fin del eje.
* Align-items (cross axis):
  + Stretch (defecto): los elementos ocupan todo el espacio disponible. Todos tendrán la altura del más alto.
  + Flex-start: se alinean al inicio del cross axis.
  + Flex-end: se alinean al final.
  + Center.

Align-items se utiliza para el caso de que el contenedor sea de una sola línea. Pero en caso de tener un contenedor multilínea, debemos utilizar align-content.

Propiedades exclusivas de los hijos

* Order: cambiar el orden de ubicación de un ítem determinado. Por defecto propiedad order:0, ubicándose en el orden dispuesto en el html. Order:-1 si lo queremos mover a la izquierda de todo. Order: 1 para moverlo a la derecha de todo.
* Align-self: toma los mismos valores que align-item, afectando solo al ítem al que fue asignado.
* Flex-grow: con esta definimos cuánto puede llegar a crecer un ítem en caso de disponer espacio libre en el contenedor. El número que le asignamos determina qué cantidad de espacio disponible dentro del contenedor debe ocupar ese ítem. Es un número entre 0 y 1 donde 1 es el 100%.

Estas propiedades se aplicarán para el ítem siempre que el padre tenga configurado display:flex.