Notas de HTML y CSS

Lucho Cervantes Jorge Luis

20 de septiembre de 2024

1. Comentario inicial

No hay mejores notas que la documentación original, de este modo, el presente trabajo no busca sustituir a dicha documentación. Por el mismo motivo, la estructura y los conceptos no son tan rigurosos. Este trabajo es más un catálogo, con una muy breve y probablemente incompleta descripción, de las herramientas que utilizo frecuentemente. El presente trabajo tiene como base inicial el curso ***Curso de HTML y CSS desde CERO (Completo)*** [1] y posteriormente se va complementando según valla aprendiendo más cosas.

Índice

2. Preeliminares

2.1. Hints

Para acceder al código de una página web, basta dar clic derecho y luego seleccionar la opción inspeccionar.

Para hacer comentarios, en HTML se usa:

```
<! --  Comentario  -->

```

Para colocar de manera automática 6 renglones con las etiquetas <**hi**></**hi**> (i=1, ... , 6) se usa h$*6.

2.2. HTML

(Hipertext markup language). Sirve para establecer la estructura de la página web mediante la creación de elementos (tag o etiquetas). Los elementos (por ejemplo, un botón) tienen propiedades como tamaño, color, color de fondo, etc. Las propiedades son, en principio, determinadas según los valores por default de cada navegador, de modo que una página luce distinta en cada navegador. Para resolver esto se usa CSS.

2.3. CSS

(Cascading style sheets). Se encarga de establecer las propiedades estéticas de los elementos o etiquetas de la página web. A demás de esto, se encarga de que estas sean las mismas en cada navegador.

3. Inicio en HTML

Se trabaja con archivos de extensión .html. En particular, la mayoría de los servidores compartidos están configurados de forma que el archivo de la página que tienen que leer, sea el de nombre index.html.

Para abrir una etiqueta se usa:

```
<nombre_etiqueta>

```

Para hacer el código más corto se usa la palabra tag. Para cerrar un tag se usa:

```
</nombre_etiqueta>

```

Para indicar un texto en letras negritas se usa la etiqueta **b**:

```
<b> Texto en negritas </b>

```

aunque en general, CSS se encarga de la parte estética. Otra importancia de las etiquetas es que ayudan a posicionar mejor nuestra página en el buscador. Por ejemplo, las etiquetas **h1**, y **h2** son para definir un título y un subtítulo. Si bien con CSS podemos hacer que visualmente ambos sean idénticos, al momento de que un navegador realice una búsqueda, se dará prioridad a lo que dice en h1 para ver si posiciona o no nuestra página en lo primeros lugares de la búsqueda.

4. Estructura de una página web

Para cargar la estructura de una página web se usa html:5. Esto carga una ”plantilla” de la estructura de una página web:

```
<!-- Etiqueta que define la version de html usada (no se cierra) -->
<!DOCTYPE html>

<!-- Se indica todo lo que abarcara la pagina web y su lenguaje. se puede usar 'es' (espanol) -->
<html lang="en">

<!-- Se indica la informacion sobre la pagina -->
<head>
```

```
    <!-- Se indica el conjunto de caraceres a utilizar -->
    <meta charset="UTF-8">

    <!-- Configuracion de la pantalla. Primero se indica que la pagina se adapte a la pantalla
    del dispositivo y luego que inicialmente muestre un zoom de 1 -->
    <meta name="viewport", content="width=device-width, initial-scale=1.0">

    <!-- Titulo de la pestana de la pagina-->
    <title>Document</title>
</head>

<!-- Cuerpo de la pagina con todo el contenido-->
<body>

</body>
</html>

```

4.1. Párrafos y encabezados

Los párrafos se indican con la etiqueta **p** y los encabezados con **hi** (con i=1,...,6):

```
<hi> Encabezado o titulo </hi>
<p> Texto que compone a un parrafo </p>

```

Naturalmente, estas etiquetas se colocan dentro del cuerpo de la página. En el caso de los **hi**, el tamaño de los títulos disminuye conforme aumenta i. También disminuye la jerarquía. En particular, una página web solo debe contar con un **h1** de otro modo se perjudica la visibilidad de la página en los buscadores.

4.2. Listas

Son útiles para crear menús de navegación. Existen dos tipos de listas, las ordenadas (**ol**) y las no ordenadas (**ul**). Los elementos de las listas se indican con la etiqueta **li** (list item):

```
<1-- Lista ordenada -->
<ol>
    <li> Elemento 1 </li>
    <li> Elemento 2 </li>
</ol>

<1-- Lista no ordenada -->
<ul>
    <li> Elemento 1 </li>
    <li> Elemento 2 </li>
<ul>

```

4.3. Enlaces

Un sitio web es un conjunto de páginas web, cada una de ellas con un propósito específico. Este conjunto de páginas web está conectado mediante enlaces o hipervínculos.

Para generar un enlace se usa la etiqueta **a**, en conjunto con el atributo **href**:

```
<1-- Enlace en el area de trabajo -->
<a href = 'url'> Enlace local </a>

<1-- Enlace externo -->
<a href = 'https://url'> Enlace externo </a>

```

Si se quiere que el enlace se abra en una nueva pestaña, se tiene que agregar el atributo **target**. Este por defecto toma el valor _self (el enlace se abre en la misma pestaña). Cuando se especifica que tome el valor _blanck, el enlace se abre en una nueva pestaña.

También se cuenta con el atributo **title**, en el cual se coloca información que será desplegada cuando el mouse pase por encima del botón del enlace.

4.4. Imágenes

Para insertar una imagen se utiliza la etiqueta **img**. Esta etiqueta no requiere cerrarse (es una etiqueta autocerrada) y se usa como:

```
<img src = '' alt ='' title =''>

```

donde el atributo **src** corresponde a la ruta de la imagen, **alt** es un mensaje alternativo mostrado en el caso de que la imagen no se encuentre y **title** es la información desplegada al pasar el mouse sobre la imagen. Nuevamente, solo cargamos la imagen en la página, el formato y atributos estéticos quedarán a cargo de CSS.

4.5. Rutas

Se consideran dos tipos de rutas, las absolutas y las relativas. Las primeras son aquellas en las que no importa desde donde se genere la petición de entrada, la página y su contenido siempre se mostrarán (Se usan para redirigir a páginas externas). Las segundas son aquellas donde depende desde donde se genere la petición de entrada (Se usan para redirigir a páginas del propio sitio web).

En una ruta relativa, para indicar que se entre a una carpeta se usa /nombre_carpeta. Para que se regrese una carpeta se usa ../

4.6. Formularios

Sirven para pedirle datos al usuario. Estos se indican con la etiqueta **form** y adentro de ellos se coloca el tipo de entrada de datos deseamos. Esto último se hace con la etiqueta **input**:

```
<form>
    <input type = ''>
</form>
```

Hay varios tipos de inputs. Algunos de ellos son text, color, submit, checkbox, radius, file, email etc.

Cuentan con el atributo **required**, que al colocarlo los vuelve obligatorios.

El atributo **name** sirve para poder referenciar y manipular los datos. Es como si fuera una variable "name" que almacena los datos computados por el usuario.

El atributo **placeholder** Sirve para poner un texto de ejemplo que desaparece cuando el usuario computa la información requerida. No todos los inputs tienen **placeholder**.

minlength='n' es un atributo que pide un mínimo de caracteres para que el input proceda.

5. Inicio en CSS

Se distinguen 3 formas de enlazar código CSS con objetos de HTML.

- **Estilos en línea.** Cuando se da el estilo en una sola línea de código (el estilo se da como atributo). Por ejemplo, en un encabezado y un párrafo:

```
<h1 style='color:blue ; font-size: 40px;'>
    Titulo
</h1>
<p style='color:red ; font-size: 20px;'>
    Parrafo
</p>
```

  La desventaja de esto es que se mezclan dos tipos de código y los atributos solo se aplican en los objetos en los que se colocaron (I.e. si se escribe otro párrafo, no se aplicará el estilo a menos que se le indique nuevamente).

- **Dando el estilo como etiqueta.** Se hace uso de la etiqueta **style**:

```
<h1> Titulo </h1>
<p> Parrafo </p>
<style>
    h1 {
        color:blue;
        font-size: 40 px
    }
    p {
        color:red;
        font-size: 20 px
    }
</style>
```

  En este caso, se tienen bloques separados de código HTML y código CSS. .

- **Archivos con HTML Y CSS separados.** Para cada página web, se tiene un documento con todo el código HTML y un documento con todo el código CSS. Para enlazar ambos archivos, dentro de la etiqueta **head** del archivo HTML se usa la etiqueta **link**:

```
<link rel = 'stylesheet' href='arch.css'>
```

5.1. Selectores

Como se vió anteriormente, al usar

```
p{
    atributos;
}
```

Los atributos especificados se aplican a todos los objetos definidos con una etiqueta **p**. Sin embargo, puede ocurrir que queramos que dos párrafos tengan un formato distinto. Para hacer esto, en el archivo HTML, a los párrafos de diferente formato se les define una clase como atributo:

```
<p class = 'nombre_clase'>
    Parrafo
</p>
```

y se asignan los atributos (separados por ;) en el archivo CSS:

```
p {
    atributo_1;
    atrubuto_2;
}
.nombre_clase {
    atributos distintos
}
```

En particular, puede haber más de un párrafo en la clase definida. Si queremos que el párrafo pertenezca a más de una clase, estas se nombran separadas por un espacio en **class**.

Si queremos que solo un párrafo tenga un formato en específico y distinto, conviene identificarlo con un id:

```
<p id = 'Identificacion'>
    Parrafo
</p>
```

y se asignan los atributos en el archivo CSS:

```
p {
    atributos
}
#Identificacion {
    atributos distintos
}
```

Si se desea asignar otras propiedades solo a una parte del párrafo, en el archivo HTML dicha parte se encierra con la etiqueta **span** y luego, en el archivo CSS se indican sus propiedades:

```
span {
    atributos
}
```

Naturalmente, todo lo anterior aplica para las otras objetos como los **hi** etc.

5.2. Atributos de texto

Los atributos de texto más usados son:

- **color:** color del texto.
- **font-family:** fuente.
- **font-size:** tamaño de la fuente.
- **font-weight:** grosor de fuente. Es mejor usar font-weight: bold[1] que la etiqueta **b**, para poner el texto en negritas. Si se desea que solo una parte del párrafo se ponga en negritas, dicha parte se debe encerrar en la etiqueta**strong**.
- **font-style:** grado de inclinación de la fuente.
- **text-aling:** alineación del texto.
- **text-decoration:** subrayar, tachar el texto, colocar un overline etc.
- **line-height:** espacio entre renglones.
- **letter-spacing:** espacio entre letras.
- **text-transform:** poner todo en mayúsculas o minúsculas, poner el inicio de cada palabra en mayúsculas, etc.

5.2.1. Fuentes de texto externas

Primero se debe buscar una fuente de nuestro gusto. Esto puede hacerse en Google Fonts. Elegida esta, se copia el código anexo a ella y se pega en el archivo HTML dentro de la etiqueta **head**. El código aparece en una etiqueta **link**. Hecho esto, el nombre de la nueva fuente ya debería aparecer como opción en font-family.

6. Modelo de caja

Cada objeto en HTML se puede pensar como una caja. Según cambiemos las propiedades de cada caja es como se le da forma a una página web. Para generar una caja vacía (o una división) se usa la etiqueta **div** en la etiqueta **body** del HTML. En el CSS se le pueden dar propiedades como sigue:

```
div {
    <!-- Obligatorias para que aparezca la caja-->
    width: 200px ;
    heigth: 90% ;

    <!-- demas propiedades -->
}
```

Es un input para volúmenes de texto más grand Cuando se pone un porcentaje en **width** el ancho de la caja es dicho porcentaje del ancho de la caja que la contiene. Ocurre lo mismo con **height**. Esto es útil cuando la pantalla cambia de tamaño. Pueden agregarse **min-width** y **max-width** para que al modificarse estos no sobrepasen un límite fijo.

Las propiedades más importantes de las cajas son: es.

- **Contenido.** Texto, imagen o archivo que contiene la caja.
- **Padding.** Espacio entre el contenido y los límites de la caja. Se tienen las siguientes formas:

```
<!-- Misma separacion en los 4 bordes -->
padding: 10px

<!-- Horizontal y vertical -->
padding: 10px 20px

<!-- arriba derecha abajo izquierda -->
padding: 10px 20px 11px 19px

<!-- O equivalentemente -->
padding-top: 10 px
padding-right: 20px
padding-bottom: 11px
padding-left: 19px
```

  Esto también aplica para el margen.

- **Borde:** Líneas que resaltan los límites de la caja. Se tienen las siguientes propiedades:
  - **border-width:** tamaño o grosor del borde.
  - **border-style:** permite elegir si el borde es una línea continua, de puntos, línea doble, etc.
  - **border-color:** color del borde.
  - **border-top:** borde de arriba. Análogamente, se usa **-bottom, -right y -left**.
  - **border-radius:** permite redondear las esquinas de las cajas.
  - **border-top-left-radius:** para redondear la esquina superior izquierda. Tiene sus versiones análogas para las otras esquinas. Puede suplirse con:

```
<!-- arr_iz, arr_der, ab_der, aj_iz-->
border-radius: 0px 0px 20px 20px;
```

  Para que un borde aparezca, se necesita especificar al menos su ancho y su estilo. Al igual que con **padding**, se pueden especificar las primeras 3 propiedades en una sola línea de código, por ejemplo:

```
border: 5px solid black;
```

  En el caso de **-radius** el redondeo al colocar el valor en píxeles es muy distinto al obtenido colocando un porcentaje.

[1] Es equivalente al valor de 700. Puede usarse **bolder** en su lugar. Este toma el siguiente valor del que toma **bold**.

- **Margin:** Distancia colocada entre la caja y otros objetos. Al colocar margin = auto el objeto se centra horizontalmetne.
- **background:** es el fondo
- **background-color:** para indicar el color del fondo
- **background-image:** Si se quiere colocar una imagen de fondo se usa:

```
background-image: url(ruta_de_la_imagen);

<!-- Para indicar el % del ancho de la caja
que abarcara la imagen-->
background-size: %;

```

Si se coloca el tamaño en píxeles, este siempre será el mismo y si la caja es más grande, la imagen se repetirá hasta llenar el espacio siempre manteniendo el tamaño especificado.

Si se coloca la propiedad **contain**, se procura que la imagen entre al menos una vez en el espacio de la caja. Si la caja es más grande, la imagen se repite. Esto puede modificarse con:

```
background.repeat: no-repeat;

<!-- Tambien estan las opciones
repeat-x y repeat-y -->

```

El espacio sobrante se queda con el color definido inicialmente.

Si se coloca **cover** el tamaño de la imagen se adapta de manera que cobra exactamente o más a la caja. En el caso de que dicho tamaño sobrepase al de la caja, la imagen se centra con:

```
background-position: center;

```

La posición puede cambiar. Todas las propiedades anteriores se pueden colocar en una sola línea de código al usar únicamente background y colocando las propiedades separadas con espacio. Por ejemplo:

```
background: url() top / cover red fixed;

```

Nota: siempre, entre la propiedad position y la propiedad size, se coloca /.

6.1. Box sizing

Al aplicar propiedades como el margen y el padding el tamaño de la caja puede verse aparentemente modificado[2], lo cual puede ser un problema si es que se quieren mantener un tamaño específico, así como agregar dichas propiedades. Para resolver esto se usa box sizing:

```
.clase_caja {
    box-sizing: border-box;
}

```

En este caso, el tamaño que permanece fijo es el tamaño que abarca hasta el margen de la caja.

6.2. Unidades de medida

Existen dos tipos

- **Absolutas**: La unidad más común es el pixel, px. Se usa comúnmente para el ancho y alto de imágenes, márgenes, el paddings y el tamaño de la tipografía. Naturalmente, estas unidades se reflejan en la pantalla.

  También existen los puntos, pt[3], y las picas, pc[4].

  Si queremos que las unidades se reflejen en una impresión se pueden usar los cm, mm o in.

- **Relativas:** Dependen de alguna otra cantidad. Por ejemplo, el tamaño de la pantalla (el cual cambia con cada dispositivo o con el tamaño de la pestaña).

  Una de estas unidades es el porcentaje del tamaño definido en la caja que contiene el objeto. Se especifican con %

  Otra unidad es el em, el cual tiene el mismo tamaño que el elemento padre.

  Se tienen los rem. Es análogo al em solo que toma como referencia a un tamaño de fuente "raiz" que es definido en todo el archivo html como sigue:

```
html {
font-size: 500px
}

```

  Si este no se define se tiene por defecto un tamaño de 16 px.

  Se tienen las medidas definidas a partir del view port es decir, se toma como referencia el tamaño de la ventana. Se especifican con **vh, vw**

  Se tienen también las unidades **vmin** que toma el mínimo entre vh y vw. y **vmax** que toma el máximo.

[2] Esto es porque el tamaño fijo se define para el contenido de la caja.
[3] 1/72 de una pulgada inglesa.
[4] equivalentes a 12 pt cada una.

6.3. Gradientes de color

La primera forma de establecer un gradiente entre n colores es colocando **linear-gradient(direccion, color1, ..., colorn)** en la propiedad **background**. En lugar de alguno de los dos colores se puede colocar **transparent**[5]. Para que el color se vaya desvaneciendo. En la dirección se puede colocar **to right, to bottom, to top, to left**.

Funciones completamente análogas son **radial-gradient** y **conic-gradient.**

6.4. Sombras

Para darle sombra a una caja se usa:

```
box-shadow: a, b, c, d, color:

```

donde a es el movimiento (de preferencia en pixeles) horizontal, b es el movimiento vertical, c es el desenfoque, y d es la extensión de la sombra.

De igual forma, se le puede dar una sombra al texto con:

```
text-shadow: a, b, c, color:

```

A notar que en este caso no se cuenta con d.

7. HTML intermedio

7.1. Favicons

Es el ícono que aparece en la pestaña a la izquierda del nombre de la página. Este debe indicarse dentro de la etiqueta **head** con la etiqueta **link** como sigue:

```
<link rel='icon' href='ruta' type='image/tipo'>

```

donde el tipo puede ser png, ico etc.

7.2. Metatags

Son etiquetas **meta** para hacer configuraciones que no son visibles en la página. Ya se vio <meta charset> y <meta name='viewport' content= ...>. Además de viewport se tienen los siguientes nombres:

- **description:** en content se coloca la descripción de la página.
- **keyword:** se colocan las palabras clave de la página. Estas favorecen la presencia en los buscadores.
- **autor:** se coloca al creador de la página.
- **robots:** Si content toma el valor 'nofollow' la página no aparece en los buscadores. Si toma el valor 'nosippet' no permite que el buscador muestre una sección de la página en la sección de resultados. Si toma el valor 'noarchive' el buscador no puede almacenar una copia de la página en la memoria caché. Si toma el valor 'noimageindex' la imagen de la página no aparecen el buscador de imágenes.
- **title:** se coloca el título de la página.
- **OG:algo** Sirve para configurar como se vería nuestra página como un enlace de otras páginas en otras redes sociales. Se recomienda ir a la bibliografía para ver sus atributos.

7.3. Textarea

Es un input para volúmenes de texto más grandes. Se crea con la etiqueta **textarea**.

Además de las mencionadas anteriormente para los otros objetos, los textareas tiene las siguientes propiedades:

- **resize:** Sirve para que el usuario pueda modificar su tamaño en tiempo real sobre la página. Si se coloca **both** se puede modificar el ancho y el alto, si se coloca **vertical** no se puede modificar el ancho. Se tiene algo análogo con **horizontal.** Es importante recalcar que el usuario tiene que realizar la modificación manualmente. Se coloca **none** si se quiere mantener fijo su tamaño.

  Nota: este se coloca en CSS.
- **readonly** Se coloca en HTML y vuelve al contenido solo de lectura. En este caso, el contenido se coloca dentro de la etiqueta y no es introducido por el usuario. A pesar de esto, al colocar el mouse encima el textarea se enfocará y al enviar el formulario su contenido será registrado.
- **disabled** Es análogo a **readonly** solo que el textarea no se enfoca y su contenido no es registrado al enviar el formulario. I.e. Solo aparece el texto.
- **max-length** Establece el máximo de caracteres.
- **rows='n'** Sirve para indicar que el alto del textarea debe ser tal, que quepan n líneas. Así puede ahorrarse el hacer cálculos con el height en CSS.

7.4. Label

Se indican con la etiqueta **label** dentro de la caja que contiene al objeto que se va a etiquetar. Para enlazar el label con el objeto que etiqueta, se tiene que darle un **id** al objeto y luego usar la propiedad **for** en label. Por ejemplo, con un textarea:

```
<div>
    <label for='referencia'>Etiqueta</label>
    <textarea id='referencia' ></textarea>
</div>

```

Una forma alternativa de hacer sin usar id, es como sigue:

[5]Se puede colocar **Transparent** 10 % para indicar que el 10 % de la imagen es transparente.

```
<div>
    <label>Etiqueta
    <textarea></textarea>
    </label>
</div>
```

7.5. Select

Permite generar un selector de opciones. Se hace con la etiqueta **select**. Las opciones de indican adentro de ella con etiquetas **option** como sigue:

```
<div>
    <select>
    <option value='val_1'>Opcion 1</option>
    <option value='val_2'>Opcion 2</option>
    <option value='val_3'>Opcion 3</option>
    </select>
</div>
```

lo encerrado por la etiqueta es el mensaje que selecciona el usuario y **value** es el valor enviado en el formulario asociado con dicho mensaje.

7.6. Datalist

Es como un **select**, pero las opciones pasan a ser sugerencias de autocompletado. El usuario puede computar algo que no necesariamente esté como opción. Esto se hace con el siguiente código:

```
<div>
    <input list='palabra' name='palabra'>
    <datalist id='palabra' name'otra_palabra'>
        <option value='opcion'>Desc</option>
        <option value='opcion'>Desc</option>
        <option value='opcion'>Desc</option>
    </datalist>
    </input>
</div>
```

donde **value** es el valor que se selecciona y el que se envía en el formulario y Desc es una descripción de lo que dicho valor significa.

7.7. Fieldset

Sirve para agrupar los campos de un formulario y encerrarlos en una caja con título propio. Se generan con el siguiente código:

```
<fieldset>
    <legend>Titulo</legend>
    <div></div>
    <div></div>
    <div></div>
</fieldset>
```

7.8. Summary and details

Sirve para colocar un objeto y desplegar un resumen al darle click con el mouse. Se generan con la etiqueta **details** sobre el cuerpo de la página (dentro de la etiqueta **body** si colocarse dentro de una caja **div**). Su estructura es como la siguiente:

```

<details>
    <summary> Lo que se muestra </summary>
    Lo que se despliega
</details>

```

7.9. Más sobre enlaces

Si se desea colocar un enlace hacia algún objeto en la misma página, dicho objeto debe ser identificado por un **id**, posteriormente se coloca el enlace con la etiqueta **a** con el atributo **href** cin el valor del **id** del objeto. Por ejemplo, si se quiere referenciar un subtítulo:

```
<a href='#identificador'> Enlace </a>
                .
                .
                .
<h2 id='identificador'>Subtitulo</h2>

```

Nota: Al ser un enlace a un elemento de la misma página, es necesario colocar el símbolo # antes del **id**.

Si se desea que el enlace sea para descargar un archivo entonces se coloca el atributo **download** y en **href** se coloca la ruta del archivo a descargar:

```
<a href='ruta_archivo' download> Descarga </a>

```

Cuando se crea un enlace a una página externa con el atributo **target='_blank'** (en una ventana nueva.) al ingresar nos volvemos vulnerables [6]. Para solucionar esto se coloca el atributo **ref**

```
<a href='' rel='noopener noreferer nofollow'>
    Enlace
</a>

```

el valor **'noopener'** niega el acceso a nuestro window.opener. En particular, esto también mejora el rendimiento de la página. Con el valor **'noreferer'** la página que abre el enlace se vuelve anónima y con el valor **'nofollow'** la página que abre el enlace y, por tanto, los enlaces colocados por terceros en ella, no se indexa en los buscadores. Por ejemplo, si los usuarios "publican" enlaces maliciosos en nuestra página, estos no aparecerán el Google cuando se busque nuestra página.

También se pueden crear enlaces para mandar un correo y para llamar a un número:

```
<a href='mailto:correo_destino' >
    Mandar correo
```

[6] La página anfitriona tiene acceso a nuestro window.opener (que proporciona una referencia de la página que abrió el enlace, en este caso), permitiendo la modificación o sustitución de nuestra página por una idéntica, para robar información, o la redirección a una página maliciosa

```
</a>

<a href='tel:+telefono_destino' >
    Llamar
</a>

```

7.10. Tablas

Se generan con la etiqueta **table**. Dentro de ellas se generan las filas con la etiqueta **tr** y dentro de estas se colocan los valores de campo, cada uno con la etiqueta **td**. Para indicar los encabezados de la tabla se genera una fila más, pero los nombres se indican con la etiqueta **th**. Para indicar el encabezado, el cuero y el pie de la tabla, se usan las etiquetas **thead, tbody** y **tfoot** respectivamente.Por ejemplo:

```
<div>
    <table cellspacing>
        <thead>
            <tr>
                <th>Columna 1</th>
                <th>Columna 2</th>
                <th>Columna 3</th>
            </tr>
        </head>
        <tbody>
            <tr>
                <td>11</td>
                <td>21</td>
                <td>31</td>
            </tr>
            <tr>
                <td>12</td>
                <td>22</td>
                <td>32</td>
            </tr>
        </tbody>
        <tfoot>
            <tr>
                <td>13</td>
                <td>23</td>
                <td>33</td>
            </tr>
        </tfoot>
    </table>
</div>

```

Genera:

Columna 1	Columna 2	Columna 3
11	21	31
12	22	32
13	23	33

Si en la etiqueta **td** se coloca el atributo **colspan='n'** el campo abarcará n columnas. El atributo **cellspacing** de la etiqueta **table** sirve para que en el archivo CSS se pueda usar el atributo **border-spacing: npx** para separar las columna n pixeles.

7.11. Videos

Para insertar un video se utiliza la etiqueta **video**:

```
<video src='ruta' controls loop autoplay mute>
</video>

```

donde el atributo **controls** es para colocar la barra de control del video. **autoplay** sirve para que cada que se abra la página se reproduzca el video. En particular, si se refresca la página, el video no se reproduce automáticamente. Para que esto ocurra se agrega **mute**, sin embargo, el video se reproduce con el volumen al mínimo. **loop** hace que el video se reproduzca en bucle.

Para cargar un archivo de audio solo se sustituye la etiqueta **video** por la etiqueta **audio.**

En el caso del video, dentro de su etiqueta puede colocarse la etiqueta **track**, encargada de colocar subtítulos:

```
<video src='ruta'>
    <track src='ruta'>
</video>

```

donde el **src** de la etiqueta **track** es un archivo de subtítulos (.vtt). Si en la etiqueta se coloca el atributo **default** los subtítulos se colocan sin activarlos en la barra de control. Se coloca **kind** para indicar si son subtítulos, captions etc. Se coloca **srclang** para indicar el idioma, hacerlo también se indica el idioma en la barra de control. Si se desea indicar con una palabra distinta a la que aparece por defecto, esta última se indica con el atributo **label**.

7.12. Carga diferida

Cuando la página es muy extensa, en términos de rendimiento es más conveniente que solo se carge la parte mostrada al usuario conforme este va navegando. Para esto, a los objetos que querramos que se cargen solo hasta que el usuario llege a la parte de la página donde se encuentran, se les coloca el atributo **loading='lazy'** .

7.13. Semántica de HTML

Si bien una página se puede armar con puros **div** es conveniente indicar las partes de la página con otras etiquetas que si bien no cambian nada visualmente, permiten posicionar mejor la página debido a que asigna una jerarquía a los elementos de la misma, la cual es aprovechada por los buscadores para recomendar la página.

Se usan las siguientes etiquetas:

- **header.** Indica el encabezado de la página. Puede haber más de uno. No confundir con **head** (que es el encabezado del archivo HTML que contiene información de la página, no mostrada en pantalla).
- **nav.** Indica una barra de navegación, usualmente se coloca dentro de la etiqueta **header**. Por ejemplo:

```
<header>
    <nav>
        <ul>
            <li>boton 1</li>
                .
                .
```

```
                .
            <li>boton n</li>
        </ul>
    </nav>
</header>
```

- **section.** Sirve para separar el contenido en secciones.
- **main.** Indica el contenido principal de la página.
- **aside.** Indica una sección secundaria relacionada con el contenido principal.
- **footer.** Indica el pie de página (donde se colocan la información de actualización de la página, cuestiones de derechos de autor, términos y condiciones, etc.) de la página.

7.14. Accesibilidad

Hace referencia a la facilidad que deben de tener las personas con discapacidad para entrar y navegar en la página. Lo más crucial que debe tener la página para lograr esto, es que se pueda sustituir el mouse y su click por las teclas tab y enter. Aunado a esto, cada elemento clickeable debe contener un atributo **aria-label** con una descripción detallada pero breve de la función del objeto.

8. CSS intermedio

8.1. Selectores avanzados

Para seleccionar un objeto que tiene una propiedad específica se usan los corchetes. Por ejemplo, para seleccionar una imagen por su nombre de archivo:

```
img [src=''] {
   Propiedades css
}

<!-- Para seleccionar todas las imagenes que
teminan con "algo": -->
img [src$=algo] {
   propiedades css
}

<!-- Para seleccionar todas las que empiecen
con "algo": -->
img [src^=algo] {
   propiedades css
}
```

También se pueden seleccionar objetos de manera descendente. Por ejemplo, para seleccionar solo las **etiqueta2** que están contenidas en una **etiqueta1** se usa:

```
etiqueta1 etiqueta2 {
    propiedades css
}

<!-- Si no se quieren etiquetas intermedias
entre etiqueta1 y etiqueta 2:-->
etiqueta1 > etiqueta2 {
    propiedades css
}

<!-- Si se quieren las primeras etiqueta2 que
estan despues de una etiqueta 1 (sin estar
contenidas):-->
etiqueta1 + etiqueta2 {
    propiedades css
}
<!-- Si se quieren todas las etiqueta2 que
estan despues de una etiqueta 1 :-->
etiqueta1 ~ etiqueta2 {
    propiedades css
}
<!-- Si se quieren todas las etiqueta2 y
etiqueta 1 a la vez :-->
etiqueta1, etiqueta2 {
    propiedades css
}
```

8.2. Herencia

Se refiere a que algunas propiedades declaradas para un objeto (padre) se aplican automáticamente a los objetos que contienen o hijos, al menos, hasta que específicamente las indiquemos en el archivo CSS. Por practicidad, no todas las propiedades se heredan.

Si se quiere que una propiedad sea heredada, se coloca el valor **inherit**. Si no se quiere que sea heredada se coloca **initial**. Por ejemplo, para heredar el color en un párrafo:

```
P {
    color: inherit;
}
```

El color se hereda del primer elemento superior que tenga definido un color.

8.3. Cascada y especificidad

Se refiere a la resolución de conflictos cuando a un objeto se le asigna dos veces la misma propiedad. Para esto se usa un sistema de ponderación donde si el objeto solo es seleccionado por su etiqueta, la especificidad es 1, si tiene una clase es 10 y si tiene un id es 100. CSS asignará las propiedades de la selección con el mayor valor de especificidad. Si en ambas asignaciones la especificidad es la misma, las propiedades asignadas serán las de la última instrucción ejecutada en el archivo CSS.

Hay más niveles de prioridad, pero estos se obtienen fuera del archivo CSS. Así Tiene más prioridad la asignación de las propiedades mediante la etiqueta **style** dentro del código HTML. A su vez tiene más prioridad la asignación dentro de las mismas etiquetas de los objetos con el atributo **style=”**. Finalmente, el nivel más alto de prioridad es cuando a la propiedad se le agrega **!important**. Por ejemplo:

```
P {
    color: red; !important
}
```

Naturalmente, se pueden combinar los niveles de especificidad mencionados en los dos párrafos anteriores. La importancia de la especificidad radica en que se tienen alternativas en el caso de que la asignación más específica no funcione o no sea soportado por el navegador. **!important**.

8.4. Pseudo clases

Una clase sirve para seleccionar un objeto. Dada una clase, una pseudo clase del objeto en cuestión sirve para asignar sus propiedades en un momento, condición o estado específico (como al pasar el mouse por encima, al hacer la ventana pequeña, al haberle hecho click una vez, etc.) Estas se indican como:

```
objeto:pseudo_calse {
    Propiedades
}

```

Se tienen las siguientes pseudo clases:

- **hover.** Indica cuando el mouse pasa por encima del objeto.
- **active.** Cuando se da click en el elemento.
- **first-child.** Indica el primer objeto de dicho tipo. Su análogo es **last-child** que selecciona el último objeto. Para elegir al n-ésimo objeto de ese tipo se usa **nth-child(n)**. Se puede colocar un patrón, por ejemplo, si en el argumento se coloca 2n se seleccionan los pares. En este caso el patrón se ve afectado si hay más hijos de diferentes tipos, para solucionar esto se usa **nth-of-type**.
- **not(.clase)** Indica que se tienen que excluir los objetos con la clase especificada.
- **empty** Cuando el objeto está vacío, i.e., que sean de la forma:

```
<objeto></objeto>

```

- **checked.** Cuando un objeto de tipo checkbox o similar es checkeado.
- **link.** Indica un enlace no visitado
- **visited.** Indica un enlace visitado.
- **focus.** Para que el mouse desaparezca al pasar por el objeto que lo tenga activado.
- **invalid.** Cuando un input es inválido. Por ejemplo cuando se pide una contraseña de n caracteres y se proporciona una con menos.
- **valid.** Cuando un input es válido.

8.4.1. where is y has

Suponiendo que se tiene la siguiente situación:

```
<div class='part'>
    <h1></h1>
    <h2></h2>
    <h3></h3>
</div>

```

y se quieren establecer propiedades solamente a los hi que están contenidos en la caja de clase part. Esto se tendría que hacer con:

```
.part h1, .part h2, .part h3 {
    Propiedades
}

```

Esto puede abreviarse con la pseudo clase **where**:

```
.part :where(h1,h2,h3) {
    Propiedades
}

```

Ambas maneras tienen la misma especificidad, de modo que si ambas se colocan, se aplicará la instrucción que se ejecute al último.

Si en lugar de **where** se usa **is**, la especificidad de la instrucción será mayor que la de la forma original.

Finalmente, si se quiere dar propiedades a una etiqueta1 solo cuando esta contenga una etiqueta2, se usa la pseudo clase **has**:

```
etiqueta1 :has(etiqueta2) {
    Propiedades
}

```

Como vimos, para seleccionar una etiqueta2 una que es antecedida por una etiqueta 1 se usa etiqueta1 + etiqqueta2, sin embargo, para seleccionar una etiqueta1 que es seguida por una etiqueta2 se usa:

```
etiqueta1 :has(+etiqueta2) {
    Propiedades
}

```

8.5. Pseudo elementos

Sirven para indicar partes específicas de un elemento, a diferencia de las subclases, estos se indican con :: en CSS. Se tienen algunos de los siguientes pseudo elementos:

- **first-letter.** Selecciona la primera letra
- **first-line.** la primera línea.
- **selection.** el recuadro que aparece al seleccionar una parte del texto o de la página. Si se coloca *::selection se modifica el selector de toda la página.
- **placeholder.** El texto por default adentro de un input.

- **marker.** Los marcadores de las listas
- **before.** Para colocar algo antes del objeto.

8.6. Metodología BEM

Si bien en HTML la estructura del código es más o menos clara debido a la identación y a la contención de objetos, En el caso de CSS puede llegar a ser confuso, pues las propiedades se asignan linealmente y en algunos casos se pueden definir en varias partes del código, por lo que se debe tomar en cuenta la especificidad, el orden, etc. Esto puede llegar a ser confuso. Por lo cual se requiere seguir una metodología. Una opción es la metodología BEM (block element modified). En esta, se les da formato a todos los bloques similares, asignándoles una clase "general" especificando su funcionalidad (caja, formulario, barra de tareas, etc.). Si hay un bloque que además de las propiedades generales asignadas a los elementos "general" tiene propiedades particulares, se le asigna una segunda clase modificadora de la forma "general–particular" y a esta se le asignan dichas propiedades. Para darle legibilidad al código, si una etiqueta2 está contenida en una etiqueta1, su clase general será de la forma clase1__clase2.

8.7. Display

Es una propiedad que permite cambiar los elementos en línea [7] (**display:inline**) a elementos de bloque[8] (**display:block**) y viceversa; aunque no solo existen estos valores, también pueden tenerse elementos con valores de display **none, table, cell-table** y **inline-block**. En particular, este último es la combinación de **inline** y **block** en el sentido de que colocan bloques en un mismo renglón. Hay que recordar que a los bloques sí se les puede asignar un largo y un ancho a diferencia de los elementos inline. Esta es la principal herramienta para crear barras de navegación.

8.8. Posicionamiento.

Cuando se genera un objeto se le asigna por default una posición fija (**static**) según el orden en el que se coloquen en el código y según sean elementos **block**,**inline**, etc. Si se cambia el valor de la propiedad **position** a **relative**, el espació ocupado por el objeto en su posición **static** seguirá apareciendo como ocupado por este, sin embargo, el objeto puede moverse respecto a dicha posición según se especifiquen los valores **top, bottom, right, left**. En particular, si se colocan **bottom** y **top** a la vez, se le dará prioridad a **top**. Lo mismo ocurre con **right** y **left**, Donde se le da prioridad a **left.**

Cuando los objetos se van definiendo en el código HTML, intrínsecamente se les va asignando una profundidad, Así, si el objeto1 se define primero en el código, este se encontrará detrás del objeto2 definido posteriormente. Al superponer los objetos cambiando su posición relativa, el objeto dos estará por encima del objeto1. Para cambiar la profundidad de dichos objetos se cambia el valor de su propiedad **z-index.**

Si se coloca el valor **absolute** se tiene un resultado similar al **relative** solo que ya no se aparta la región que ocuparía el objeto en su posición **static** y la posición del objeto se indica respecto el inicio de la página (esquina superior izquierda, donde se abre la etiqueta **HTML**). Sin embargo, si el objeto con posición **absolute** se encuentra contenido en un objeto que tiene una posición **relative**, la posición del objeto se indica respecto a la esquina superior izquierda del objeto que lo contiene. En particular, si a todos los valores de posición se les asigna un valor cero, pero se coloca un **margin: auto**, el objeto se coloca en el centro del objeto que lo contiene.

Si se coloca el valor **fixed** se tiene el mismo efecto que con absolute, la diferencia es que aunque el objeto al que se le aplique se encuetre contenido en otro objeto con posición **relative** la posición siempre se indicará respecto al inicio de la página. Adicionalmente, y a diferencia de los casos anteriores, la posición permanecerá fija incluso al navegar en la página.

Si se coloca el valor **sticky** se tiene el mismo efecto que **relative** hasta que se cumpla una condición especificada por nosotros (como que alcanse algún número de pixeles por debajo de la parte superior de la página) en tal caso se comporta como **fixed.**

8.9. Transiciones

Dado un objeto, una transición se lleva a cabo entre una clase y alguna pseudo clase de dicho objeto. Las transiciones deben indicarse en la clase, con las siguientes propiedades:

- **transition-property:** indica la propiedad del objeto que va a transicionar. En particular se puede colocar el valor **all** que especifica a todas las propiedades, pero esto no es una buena práctica, pues consume más recursos de los necesarios.
- **transition-duration:** indica la duración de la transición.
- **transition-delay:** indica el tiempo de retraso de la transición.
- **transition-timing-function:** indica la función de velocidad con la que se realiza la animación. Tiene tres valores por defecto, aunque se puede computar una curva bezier cúbica al computar **cubic-bezier(x1,x2,x3,x4)** donde las x son parámetros que definen a la curva. Tmabién se puede colocar **steps(n)** para que la transición se de en n saltos marcados.

Las cuatro propiedades se pueden resumir en una sola línea de la siguiente manera:

[7] A estos elementos no se les puede asignar ni un ancho ni un alto, solo toman en cuenta, márgenes y paddings horizontales. Se colocan en la misma línea

[8] Estos elementos abarcan todo un bloque horizontal o toda una línea.

```
transition: propiedad duracion timing delay

/* Si se tiene mas de una propiedad:*/
transition:propiedad1 duracion1 timing1 delay1,
           propiedad2 duracion2 timing2 delay2,
                              .
                              .
                              .

```

Los navegadores tienen una opcion para desactivar las transiciones y/o animaciones de una página. Para que nuestra página sea compatible con esta opción y en efecto estas se desactiven se usa un ,edia querie:

```
@media (prefers-reduced-motion: reduce){
    codigo CSS con transiciones mas lentas

    /*O si no se quiere ninguna transicion:*/
    transition: none

    /*y si no se quiere ninguna animacion:*/
    animation: none
}

```

8.10. Overflow

Es una propiedad que sirve para manejar el contenido que sobresale del contenedor de un objeto. Si toma el valor **visible** el contenido es visible aunque esté afuera de la caja donde se definió. Si toma el valor **hidden**[9] el contenido no se muestra y se pierde. Para solucionar esto, se coloca el valor **scroll** el cual permite moverse en el contenido mediante una barra de navegación.

Se puede separar en las componentes **overflow-x** y **overflow-y** que solo actúan en las respectivas direcciones.

Si se coloca el valor **auto**, la barra de navegación solo aparece si el contenido es lo suficientemente largo.

8.10.1. Flujo de texto

La propiedad principal para controlar como se distribuye el texto en la caja que lo contiene es **white-space**. Este puede tomar los siguientes valores:

- **nowrap.** No coloca saltos de línea para generar renglones que quepan en la caja.
- **pre.** Se respetan solo los saltos de línea y los espacios en blanco que se especifican desde el HTML.
- **pre-wrap** Funciona como **pre** solo que se agregan los saltos de línea necesarios para que el texto no salga de la caja.

También se tiene la propiedad **textoverflow**, la cual al tomar el valor **elipsis** en conjunto con **white-space: nowrap** y **overflow: hidden** coloca 3 puntos suspensivos cuando el texto sobrepasa las dimensiones de la caja.

La propiedad **word-break** indica como se recorta el texto para encajar en la caja. Si toma el valor **break-all** las palabras se recortan sin seguir las reglas gramaticales. Si toma el valor **keep-all** no se recortan las palabras y el renglón se recorta en un espacio en blanco.

Para que las palabras se recorten según las reglas gramaticales se usa la propiedad **word-wrap** con el valor **break-word**. Si se coloca el valor **anywhere** se tiene el mismo comportamiento que con **breack-word** solo con las palabras muy extensas.

Se tiene también la propiedad **text-wrap**, la cual, al tomar el valor **balance**, además de cuidar que el texto quede dentro de la caja, este se distribuye de manera armoniosa. Si se coloca el valor **pretty** lo que se busca es que tanto el primer como el último renglón tengan más de una palabra.

8.11. object fit y object position

Funcionan de manera completamente análoga a las propiedades **background** (sección 6) solo que en ves de ajustar la imagen de fondo en la caja que lo contiene, se ajusta la imágen o video a la propia etiqueta **img**.

8.12. Outline

Como se ha visto, un objeto está conformado por elementos como el contenido, el padding, el borde y el margen, a los cuales se les pueden modificar sus propiedades. En particular, si se cambia su tamaño, el tamaño del objeto se ve también afectado. A demás de estos elementos, un objeto también cuenta con un elemento outline, el cual es prácticamente otro borde después del borde, pero las propiedades de este solo son visuales, i.e. no afectan el tamaño del objeto. En particular, la separación ntre el outline y el borde del objeto se controla con la propiedad **outline-offset.**

Si se desea que no aparezca el outline que aparece por defecto cuando se enfoca un formulario (.clase:focus), se debe colocar el valor **none**, sin embargo, esto afecta a la accesibilidad de la página cuando se coloca la configuración para personas discapacitadas. Para solucionar esto, a la clase donde se especifique **outline: none** se le debe de agregar **-visible**.

8.13. Flexbox

Además de las cajas **inline**, **block**, **inline-block**, etc. se tiene el modelo de caja **flexbox.** La caja con esta propiedad no es la que tiene "flexibilidad" de movimiento, más bien son sus elementos hijo (flex items) las que se adaptan a las dimensiones de la caja padre, siempre y cuando sean hijos directos.

[9]Existe el valor **clip**, el cual es completamente análogo, la única diferencia es que este último no reserva espacio para las barras de navegación.

8.13.1. Cajas padre

A la caja padre se le pueden dar los siguientes atributos:

- **display:** indica como se van a mostrar flex items. Si se coloca el valor **flex**, cuando la ventana disminuya sus dimensiones, no solo se recorrerán, sino que también modificarán sus dimensiones de manera proporcional a la ventana.

- **flex-direction:** indica como se ordenan los flex items. Si se coloca el valor **row** se ordenan horizontalmente de izquierda a derecha. Si se coloca **row-reverse** se ordenan horizontalmente de derecha a izquierda. Si se coloca **colum** se ordenan verticalmente de arriba hacia abajo. Si se coloca **colum-reverse** se ordenan verticalmente de abajo hacia arriba. Esto si el texto sigue la dirección de izquierda a derecha (como el español). Para invertir la dirección del texto, a la propiedad **direction** se le coloca el valor **rtl**. Esto también invierte las direcciónes de **flex-direction**.

- **flex-wrap:** tiene el valor **nowrap** por defecto, que indica que los box items no pueden cambiar de línea al disminuir la ventana, por lo que cambian sus dimensiones. Si se coloca **wrap** los elementos si cambian de línea y pierden su "flexibilidad". Si se coloca **wrap-reverse** el salto de línea se hace hacia arriba.

- **flex-flow:** Se usa para indicar el **flex-direction** y **flex-wrap** en una sola línea de código.

- **justify-content:** sirve para alinear los box items sobre el eje principal (establecido por el **felx-direction**). Si se coloca **start** se alinea del lado del que parte la flecha del eje. Si se coloca **end** se alinea del lado en que termina. Si se coloca center, los box items se centran. Si se coloca **space-between** se coloca un elemento al inicio, otro al final, y el resto se distribuye en el espacio del centro. Si se coloca **space-around** se coloca el mismo margen a los box items en la dirección del eje. Si se coloca **space-evenly** el espacio en la dirección del eje se distribuye equitativamente.

- **aling-items:** sirve para alinear en el eje perpendicular al eje principal. Tiene el valor **stretch** por defecto, en el cual, de no contar con un tamaño establecido, los box items se estiran para llenar por completo el contenedor en dicha dirección. También cuentan con los valores **start, end, center**, etc. Aunque también cuenta con el valor **baseline** en el cual, si los box items cuentan con dimensiones distintas, la alineación se hará respecto al texto que contengan.

- **aling-content:** es el análogo de **justify-content** solo que para alinear en la dirección ortogonal a la principal.

- **gap:** indica la separación entre los elementos (tanto a los lados como arriba y abajo).

Orden

Para modificar el orden de visualización de los box items, se asigna un valor entero a la propiedad **order**. Los valores menores se colocan primero. De haber más de un box item con el mismo valor, el orden se determina por el orden que ocupan en el código HTML.

8.13.2. cajas hijo

A las cajas hijo se les pueden dar las siguientes propiedades.

- **flex-grow:** indica la proporción del espacio disponible que tienen que abarcar los box items. Si se coloca el valor 1, todo el espacio se ocupa. En este caso ya no importa lo que se coloque en **justify-content** a menos que se coloque un límite superior con **max-whidth.**

- **flex-shrink:** indica la proporcion en la que se van a encoger los box items cuando la ventana disminuya de tamaño.

- **flex-basis:** indica el tamaño mínimo en la dirección del eje principal, al que pueden encogerse los box items al disminuir el tamaño de la ventana. Es útil por que se puede combinar con **flex-wrap:wrap** para que los box items se encojan hasta un valor mínimo y después hagan un salto de línea.

  Para resumir las 3 propiedades anteriores en una sola línea de código se usa:

```
flex: grow shrink basis;
```

- **aling-self:** Sirve para alinear un box item de manera particular, dadas todas las alineaciones anteriores, Siempre y cuando se use **aling-items** en lugar de **aling-content**. Puede tomar los valores **start, center, end, baseline** o **stretch** (ocupar todo el espacio).

8.14. Layouts con flexbox

El layout de una página es básicamente su esqueleto, indica la distribución del espacio que ocuparan sus elementos.

9. Responsive desings

Básicamente se hace referencia a los diseños que se adaptan al tamaño de la pantalla del dispositivo que se esté usando y al tamaño de la ventana.

9.1. srcset y size en imágenes

Una página puede abrirse desde dispositivos con resoluciones diferentes. De este modo, no tiene sentido cargar imágenes de muy alta resolución si el dispositivo es de baja resolución (sería un gasto innecesario de recursos). Para optimizar esto, se pueden usar varias imágenes iguales, pero con distintas

resoluciones, e indicar que se cargue aquella que esté más acorde a la resolución del dispositivo que abra la página. Esto se hace con **srcset** en el HTML:

```
<img src='img1' srcset='img1 r1w, ..., imgn rnw'>

```

Donde las img van de la resolución más baja (img1) a la más alta (imgn). La resolución se da en píxeles solo que en lugar de colocar px, se coloca w. De acuerdo a esto, si la resolución es menor a r1w se carga img1, si la resolución esta entre r1w y r2w se carga img2 y así sucesivamente. En este caso, lo que se indica es qué imagen cargar. Para indicar con qué tamaño cargas dichas imágenes se agrega la propiedad **sizes** de la siguiente manera:

```
<img src='img1'
    srcset='img1 r1w, ..., imgn rnw'
    sizes='(max-width= a1px) b1px, ...
            ..., (max-width= anpx) bnpx'>

```

En donde el valor de la propiedad **sizes** puede leerse como sigue: Si la pantalla tiene un tamaño de menos de a1px la imagen se carga con un tamaño de b1px, si la pantalla tiene un tamaño entre a1px y a2px la imagen se carga con un tamaño de b2px y así sucesivamente. Un efecto apreciable para el desarrollador, es que con **sizes**, al variar continuamente la resolución de la página, la imagen da saltos en lugar de adaptarse de manera continua.

9.2. Picture, source y media

Además de **src**, se pueden usar **Picture, sourse** y **media**:

```
<picture>
  <source media=(max-width: a1px) srcset='img1'>
                    .
                    .
                    .
  <source media=(max-width: anpx) srcset='imgn'>
  <img src='imagen_de_preferencia>'
</picture>

```

Para obtener el mismo resultado. Este código funciona como un *if*, en el sentido de que checa la primera condición y si no se cumple se pasa a la segunda y así sucesivamente. Debido a esto, las opciones deben colocarse en el orden adecuado. La ventaja de esta forma, es que se puede cargar una imagen según el formato, indicándolo con el atributo **type='image/formato'** en **source**. De este modo se agrega la condición de que si el navegador soporta imágenes de un cierto formato, estas sean las que se carguen.

9.3. Media queries

Sirven para adaptar la página a diferentes resoluciones. Hacen uso de condiciones i.e. El contenido de las media queries solo se aplica si se cumple la condición para la que se especifican. Se indican con **@media** y su estructura es de la forma:

```
@media condicion objeto and (esp) {
    Propiedades CSS
}

```

En la condicion no se coloca nada si queremos que las propiedades se apliquen al objeto especificado. Se coloca **not** si queremeos que se apliquen a todo menos al objeto especificado. se coloca **only** para que los navegadores que no soportan las media queries no intenten aplicarlas.

Si se coloca **print** como objeto, las propiedades brindadas se aplicaran solo al formato de la página cuando esta es impresa en papel. Si se coloca **screen** se hace referencia al formato de la página en la pantalla del dispositivo. En este caso se agrega **and** con las especificaciones de la pantalla a las que se quieren aplicar las propiedades indicadas.

9.4. Layout Holy grial

Con todo lo visto hasta ahora, ya se puede realizar un layout de manera seria. El formato de layout más utilizado es el layout holy grial, el cual está compuesto por una barra de navegación, seguido del bloque principal (subdividido en 3 columnas, usualmente con el contenido principal en el centro[10]), y un footer.

9.5. Movile first

Hace referencia a la práctica en la que el diseño de una página web se enfoca inicialmente en la resolución del dispositivo más pequeño (dispositivos móviles, prácticamente siempre) y posteriormente, mediante media quereies, adaptarse a dispositivos cada vez más grandes.

9.6. Feature queries

Se usa para especificar propiedades recientes, las cuales no necesariemente son aceptadas en todos los navegadores. Son de la forma:

```
@supports (propiedades css recientes){
    codigo css usando tales propiedades
}

```

Aunque también se usa en casos en que las propiedades no son soportadas debido a la resolución. Por ejemplo cuando se sustituye **grid** por **flex**.

9.7. Container quieries

Sirven para definir contenedores y asignarles propiedades según estos cumplan las condiciones de tamaño que nosotros les establezcamos.

Para definir una etiqueta (de clase *class*) como contenedor se asigna la propiedad **container-type**, para nombrar a dicho contenedor se usa la propiedad **container-name**:

```
.class {
    container-type: valor;
```

[10]Las columnas que no son de contenido principal se indican con la etiqueta **aside** en lugar de **main.**

```
    container-name: nombre;
}
```

Donde el valor por defecto (y donde no se define el contenedor) es **none**. Si se coloca el valor **inline-size**, las propiedades se aplican condicionando el ancho. Si se coloca **size**, se condiciona el ancho y el largo. Las propiedades y las condiciones se especifican como:

```
@container nombre (tamano){
    codigo css
}
```

10. Grid

Funciona de manera similar a **flex**, sin embargo se trabaja tanto en filas como en columnas. Para indicar que una etiqueta es un **grid** se usa nuevamente la propiedad **display:**

```
etiqueta {
    display: grid;
}
```

La primera diferencia con **flex**, es que no hay colapso de márgenes[11].

Otra diferencia es que por defecto, un **grid** colocará a sus elementos (grid items) en un columna incluso aunque a dichos elemento tengan la propiedad **inline** o **inline-block**. Para cambier esto, debemos especificar las medidas de los grid item. Paara las columnas esto se hace con la propiedad **grid-template-columns**:

```
grid-template-columns: m1 ... mn;
```

Esto indica que se tendrán n columnas, de anchos m1, ..., mn respectivamente. En particular, el orden de los grid item también sigue la dirección que está a favor del texto de escritura.

Para las filas se usa la propiedad **grid-template-rows**

10.1. Unidades auto y fr

Además de contener valores de medida, los valores m1,...,mn de los grid item también puede contener valores como **auto**, con el cual la fila o columna (no necesariamente igual a los grid item que contienen) tiene un ancho que abarca todo el espacio disponible. En el caso de haber más de 2 columnas con **auto**, el espacio se reparte en partes iguales. El espacio disponible es el espacio que no tiene contenido.

También se tienen los valores **fr**, los cuales indican la fracción del espacio total, si considerar sin considerar el espacio que abarca el contenido de los grid item.

10.2. funciones repeat y minmax

El código:

```
grid-template-colum: 1fr 1fr 1fr 1fr;
```

Es equivalente al código:

```
grid-template-colum: repeat(4,1fr);
```

El primer argumento de **repeat()** se coloca el número de veces que se desea repetir el patrón de código indicado como segundo argumento. En el contexto de grid corresponde al número de columnas. El primer argumento también puede tener el valor **auto-fit**, con el cual, se agregan el número de grid item que quepan según el tamaño que se indique. Si se acaban los grid item, pero el tamaño de la pantalla sigue aumentando, los grid item comienzan a crecer para ajustarse o hasta que los límites especificados según y luego solo se recorren. También puede colocarse **auto-fill**, el cual es análogo a **auto-fill**, con la salvedad de que siempre se van agregando celdas, aunque estas no contengan un grid item definido.

La función **minmax(v1,v2)**, al colocarse en la propiedad **grid-template-coolum**, indica que la columna en cuestión puede aumentar su ancho hasta v2 y disminuirlo hasta v1.

10.3. Grid implícito y explícito

Cuando especificamos valores para **grid-template grid-template-rows** indicamos un cierto número de grid item. Al conjunto de estos grid item indicados se les conoce como el grid explícito. Sin embargo, puede que se tengan más grid item indicados en el HTML, a los cuales no se les aplican las propiedades por no estar contemplados en las propiedades de grid. Al conjunto de estos grid item se les conoce como gris implícito y tienen ciertas propiedades por defecto. En particular, el flex implícito conserva el número de columnas indicado en el grid explícito, por lo que el grid implícito se agrega como nuevas filas. Para cambiar esto se usa la propiedad:

```
grid-auto-flow: column;
```

Para establecer las dimensiones del gris implícito se usa **grid-auto-columns** y **grid-auto-rows**.

10.4. Grid gap

Al igual que con **flex**, se tiene **gap** lo cual facilita la separación de los grid items. Debido a las dos dimensiones se tienen que usar **colum-gap row-gap**. Si ambos tienen el mismo valor solo se usa **gap.**

10.5. grid row y grid colum

Sirven para indicar las líneas de la cuadrícula entre las que se encuentra a un grid item. Se indican con **grid-colum-start**, **grid-colum-end**, **grid-row-start**, **grid-row-end**. Son útiles en el caso de que queramos que un grid item abarque más de

[11] cuando se toma el margen más grande para separar objetos en lugar de sumar los dos espacios de margen.

una fila o más de una columna. Como argumento, en estas propiedades se coloca el número de la fila deseada. En el caso de **grid-colum-end grid-row-end** puede colocarse **span n** que en lugar de decir que la última línea de fila o columna es la n-ésima, quiere decir que el grid item abarcará n filas o columnas. Para resumir todo esto se eliminan las palabras **start** y **end** y el valor se expresa como **n/span m**.

10.6. Dense

Cuando se utilizan valores de la forma **n/span m** en conjunto con los argumentos **auto-fill** y **auto-fit**, pueden quedar espacios en blanco debido a que los los grid item de tamaño **n/span m** ocupan más de un espacio en el que se subdivide el grid. Esto es debido a que el orden en que se muestran los grid item se mantiene. Para solucionar esto, se usa el valor **dense** en la propiedad **grid-auto-flow**: de este modo, el orden se rompe en pos de no dejar espacios en blanco.

10.7. Grid area

En principio, puede suplir en algunos casos a **grid-colum** y **grid-row**. Por ejemplo, el código:

```
etiqueta {
    grid-column: a/span n;
    grid-row: b/span m;
}

```

Es equivalente al código

```
etiqueta {
    grid-area: a/span n / b/span m;

    /* Inclusive es equivalente a: */
    grid-area: a/n/b/m;
}

```

También sirve para distribuir y nombrar las áreas del objeto usado como grid. Para esto, se usa:

```
etiqueta {
    grid-template-columns: c1 c2 c3;
    grid-template-rows: r1 r2 r3;
    grid-template-areas:
    "espacio1 espacio1 espacio1"
    "espacio2 espacio3 espacio4"
    "espacio5 espacio5 espacio5"
    ;
}

```

De este modo se indica que la etiqueta es una cuadrícula de 3x3, con dimensiones especificadas por las ri y ci, tal que la primera fila corresponde a un solo elemento de nombre *espacio1*, la segunda fila se divide en tres columnas (elementos de nombre*espacio2*, *espacio3* y *espacio4*) y la tercera fila corresponde enteramente al objeto de nombre espacio5. Posteriormente, se indica en las propiedades de cada objeto o etiqueta el espacio que abarcarán. Por ejemplo, se puede hacer el layout holygrail con las siguientes indicaciones:

```
header{
    grid-area:espacio1;
}
nav{
    grid-area:espacio2;
}
main{
    grid-area:espacio3;
}
aside{
    grid-area:espacio4;
}
footer{
    grid-area:espacio5;
}


```

10.8. Alineación de los grid items

De manera análoga a **flex**, los objetos establecidos como **grid** tienen la propiedad **justifý-items**, que indica la alineación de los grid item respecto a las celdas que los contienen, de manera horizontal. Puede tomar los siguientes valores:

- **stretch.** Valor por defecto. Estira los grid item para que abarquen toda la celda.
- **start.** Alinea los grid item al inicio y a favor de la dirección del texto sin tomar en cuenta nada más.
- **self-start.** Alinea los grid item al inicio y a favor de la dirección del texto, pero si los grid item tienen especificada la otra dirección, esta se respeta y se alinean acorde a dicha dirección.
- **left.** Alinea los grid item a la izquierda de su contenedor o celda
- **center.** Centra los grid item
- **right.** Alinea los grid item a la derecha de su contenedor o celda.
- **end.** Alinea los grid item al final y a favor de la dirección del texto sin tomar en cuenta nada más.
- **self-end.** Alinea los grid item al final y a favor de la dirección del texto, pero si los grid item tienen especificada la otra dirección, esta se respeta y se alinean acorde a dicha dirección.
- **baseline.** Alinea los grid item de modo que el texto que contienen quede alineado.
- **first baseline.** Alinea los grid item sobre la primera línea de texto.
- **last baseline.** Alinea los grid item sobre la última línea de texto.

Además de esto, puede indicarse la aineación de un grid item particular, colocando alguno de estos valores en la propiedad **justify-self**.

Para alinear las celdas respecto de la etiqueta definida como grid, se utiliza la propiedad **justify content**. De manera similar a su correspondiente en **flex**, puede tomar los valores **center, space around, space betwen, space evenly,** etc.

Para realizar todo lo anterior de manera vertical, se cambia **justifý-items** por **aling-items**, justify-self por aling-self y **justify content** por **aling content**.

10.9. Subgrid

Puede generarse un grid dentro de otro grid de manera sencilla. Para esto, se genera un **div** dentro del grid item de nuestra elección. Es dicho **div** y no en el propio grid item, el que se indicará como sub grid usando la propiedad **display:grid** y posteriormente se le agregarán los grid items correspondientes como cualquier otro grid. El único inconveniente de esto es que al especificar el tamaño y número de columnas del subgrid, estas pueden aplicarse sin modificar la distribución de espacio del grid original, provocando que algunos grid item se traslapen. Esto se resuelve especificando únicamente en el grid item que contiene el sub grid, las propiedades **grid-template-rows: subgrid** y **grid-template-columns: subgrid**.

11. Animaciones

Se definen de la siguiente manera:

```
@keyframes nombre_animacion{
    from{
        propiedades css antes
    }
    to{
        propiedades css despues
    }
}

```

Si se quieren indicar pasos intermedios:

```
@keyframes nombre_animacion{
    0%{
        propiedades css al 0%
    }
    x1%{
        propiedades css al x1%
    }
                .
                .
                .
    xn%{
        propiedades css al xn%
    }
    100%{
        propiedades css al 100%
    }
}

```

Para indicar que un objeto tiene dicha animación, se le coloca la propiedad **animation-name: nombre_animacion**. Para que la animación se ejecute se necesita indicar también al menos la propiedad **animation-duration**. Algunas propiedades adicionales son:

- **animation-delay:** indica el retraso con el que comienza la animación. Se pueden colocar delays negativos.
- **animation-fil-mode:** indica el estado del objeto cuando se termina la animación. Si se coloca el valor **backwards**, se queda el estado que tenía el objeto antes de empezar la animación. Si se coloca **forwards** se queda el estado en el que termina la animación. Si se coloca **both** el objeto se coloca automaticamente con las propiedades de inicio (to o 0 %) y luego arranca la animación (incluyendo el delay), además, se queda el estado con el que termina la animación.
- **animation-timing-function:** indica la función de velocidad con la que ejecuta la animación. Es igual que **transition-timing-function.**
- **animation-iteration-count:** indica las veces que se va a repetir la animación. Si se le coloca el valor **infinite** la animación se repite en bucle siempre.
- **animation-direction:** si se coloca el valor **reverse** la animación se ejecuta al recvés. Si se coloca **alternate**, la animación se ejecuta una vez en la dirección definida y luego una vez al revés. Si se coloca **alternate-reverse** la animación se ejecuta una vez al revés y una vez en la dirección definida.
- **animation-play-state:** indica el estado de la animación. Habitualmente se usa en momentos especificos como cuando se coloca el mouse (:hover) o se da click (:active). Si se coloca el valor **paused** la animación se detiene. Si se coloca **running** la animación se ejecuta.

11.1. Animaciones con scroll

Hasta ahora, la duración y la velocidad de una animación están determinadas por el **animation-duration** y el **animation-timing-function**. Sin embargo, Se puede hacer que una animación se ejecute con el avance de la barra de scroll en lugar de con el tiempo. Para esto, en el objeto que se anima se tienen que quitar todas las propiedades referentes a la duración temporal de la animación (como **animation-duration**, **animation-timing-function**, etc.) y se debe colocar la propiedad **animation-timeline:scroll()** de este modo el avance de la animación dependerá de cuanto se recorra la barra de scroll principal (la de la propia página). Se puede definir un scroll própio para la animación pero eso es muy poco utilizado. Si se llegara a requerir, se recomienda consultar la documentación o el video.

11.2. Animación por view

Un inconveniente de la animación por scroll es que si la página es muy larga, al recorrerla, puede perderse de vista la animación. Para solucionar esto, puede hacerse que la animación, además de depender del scroll dependa de si el objeto animado está en pantalla. Esto se hace dándole el valor **view()** a la propiedad **animation-timeline.** Si se desea que la animación comience cuando aparece un objeto distinto en pantalla, se debe asignar la propiedad **view-timeline: –nombre block** de este modo, al colocar **animation-timeline: –nombre** en el objeto animado la ejecución de la animación dependerá del

otro objeto. Si se quiere que la animación empiece, por ejemplo, un poco después de aparecer el objeto o que termine un poco antes de que salga de pantalla, se coloca el valor **cover** seguido del porcentaje deseado en las propiedades **animation-range-start** y **animation-range-end**.

cover indica que la animación comienza cuando el objeto comienza a aparecer en el límite establecido. Este valor se puede cambiar por **contain**, que indica que la animación indica hasta que el objeto cruza por completo el límite establecido. En ambos casos se pueden colocar porcentajes negativos.

12. Funciones

Se indican como **nombre_funcion()**.

12.1. funciones en Filter y en backdrop filter

La propiedad **filter** sirve para colocar filtros de imagen a un objeto y a todos sus hijos. Algunos de los valores que puede tomar son los siguientes:

- **drop-shadow().** Agrega e intensifica la sombras del contorno del objeto al que se le aplique.
- **blur().** Desenfoca el objeto al que se le aplique, así como a todos sus hijos sin importar su posición.
- **brightness().** Ajusta el brillo del objeto.
- **contrast().** Ajusta el contraste del objeto.
- **grayscale().** Ajusta la escala de grises.
- **saturate().** Ajusta la saturación.
- **hue-rotate(ndeg).** Cambia los colores al rotar la escala circular de color n grados.
- **opacity().** Controla la opacidad.
- **invert().** Invierte los colores (saca el negativo).
- **sepia ().** Le da un tono sepia al objeto.

Por otro lado, la propiedad **backdrop-filter** aplica los filtros a todo lo que se encuentre detrás del objeto al que se le aplique. Puede tomar los mimos valores que **filter.**

12.2. Funciones de transformación

Modifican la forma u orientación del objeto al que se le aplican. Se colocan como valores de la propiedad **transform**. Se tienen las siguientes funciones:

- **scale().** Cambia la escala del objeto al que se le aplique. Para modificar solo el ancho se usa **scalex()**. Para modificar el alto se usa **scaley()**. Todo lo anterior es equivalente a colocar **scale:** como propiedad:

```
etiqueta {
    scale: valor;

    /*Para ancho y alto diferentes*/
    scale: ancho alto;
}

```

- **skew(ndeg).** Inclina la vertical n grados. También se tienen **skewx()** y **skewy()** que actúan en las respectivas direcciones.
- **rotate().** Rota el objeto.
- **translate(x, y).** Desplaza el objeto. También se tienen **translatex()** y **translatey()**. También se puede colocar directamente como la propiedad **translate:**

12.3. Funciones min, maz y Clamp

max(a1, ..., b1) y **min(a1, ..., b1)** entregan el valor máximo y mínimo de a1, ..., b1, respectivamente. Esto es útil cuando se quieren comparar medidas con diferentes unidades. Por ejemplo, 900**px** con 100**vh**, etc.

La función **clamp()** es tal que si **propiedad: clamp(a,x,b);**, **clamp()** vale **x** si **a** $\leq$ **propiedad** $\leq$ **b**, vale **a** si **propiedad** $\leq$ **a** y vale **b** si **b**$\leq$**propiedad**.

12.4. Variables

Cuando un valor se repite muchas veces a lo largo de todo el código, es útil definir una variable con dicho valor para simplificar el código. Esto se hace de la siguiente forma:

```
etiqueta{
    --nombre_variable: valor;
}

```

De manera que puede colocarse **var(−nombre_variable)**, en lugar del valor. El alcance de la variable son todos los hijos del elemento, etiqueta o clase en la que se defina. Las variables también se pueden redefinir, con la misma estructura de código, dentro de los hijos. Para que la variable tenga el alcance de toda la página, se define en :root

Otra forma de definir una variable de manera global es de la siguiente manera:

```
@property --nombre_variable {
    syntax: "<type>" ;
    inherits:  ;
    initial-value: valor;
}

```

donde type, especifica el tipo de variable (color, number, etc.). En inherits se coloca si el valor de la variable se puede heredar (true) o si no (false).

12.5. Función calc()

Permite realizar operaciones aritméticas básicas, la cuales se colocan como argumento. La utilidad de esto es que las operaciones se pueden realizar entre elementos de diferentes unidades de medida.

13. scroll

Una vez establecido un scroll (por ejemplo con **overflow:scroll**) en el objeto deseado, se pueden poner establecer las siguientes propiedades de scroll:

- **scroll-behavior:** control de la velocidad con la que se desliza el scroll. Solo toma los valores **Auto** y **smooth** que suaviza el recorrido.
- **scrollbar-color:** se indican los colores del fondo de la barra de navegación y de la barra misma.
- **scrollbar-width:** indica el ancho de la barra. Solo puede tomar los valores **none, auto y thin**

Para modificar las propiedades del scroll principal, Las propiedades no se indican en la etiqueta **body**, se indican en la subclase **root.**

14. Initial letter

Para acceder a la primera letra de un párrafo **p**, se usa:

```
p::first-letter {
    Propiedades CSS primera letra
}
```

Se tienen las siguientes propiedades:

- **initial-leter: a b;** indica que la primera letra va a abarcar **a** líneas desde la base y dicha base se encontrará en la línea **b**.
- **padding:** el padding de toda la vida.
- **font-family:** para indicarle una fuente en particular.

15. Unidades dinámicas del viewport

vh(**vw**) son medidas visibles de la página. En el caso de dispositivos móviles estas no son constantes, pues dicho dispositivo puede agregar y quitar menús externos en la pantalla, por lo cual un elemento que está centrado en un ordenador puede no estarlo en un dispositivo móvil debido a la aparición o remoción de uno de estos menús. Una forma de adaptar las medidas es usando **svh(svw)** cuando no hay menús y **lvh(lvw)** cuando si lo hay; sin embargo, esto requiere de aplicar muchas condiciones, lo cual no es práctico. Esto se soluciona con las medidas **dvh(dvw)** las cuales se adaptan automáticamente.

16. Función color-mix

Sirve para combinar colores. Tiene la siguiente estructura:

```
color-mix(in type, color1 a%, color2 b% )
```

donde **type** hace referencia al tipo de color (rgb, hexadecimal, etc.) y el método de combinación., y a,b a los proporciones en la mezcal de los colores 1 y 2 respectivamente.

17. Clip path

Propiedad de las imágenes que permite recortarlas. Se indica como propiedad **clip-path:**. Los valores a colocar se pueden obtener de una página que genera el código segun la forma en la que deseemos cortar la imágen.

Referencias

[1] S. Dalto, "Curso de html y css desde cero (completo)." https://www.youtube.com/watch?v=J5bXOOmkopc&list=PLeo1K3hjS3uvCeTYTeyfe0-rN5r8zn9rw&index=9, marzo 2024. Accedido en julio de 2024.