

# Apuntes de HTML

migueluisV

Realizadas: Diciembre 2022

# Índice

<b>1</b>	<b>HTML básico</b>	<b>6</b>
1.1	Tipos de etiquetas . . . . .	6
1.2	Etiquetas básicas de HTML . . . . .	6
1.3	Etiquetas de formato . . . . .	7
1.4	Atributos de etiquetas . . . . .	9
1.5	Comentarios . . . . .	10
1.6	Listas . . . . .	10
1.7	Tablas . . . . .	11
1.8	Imágenes . . . . .	13
1.9	Enlaces . . . . .	14
1.10	Formularios . . . . .	15
1.11	Frames . . . . .	17
1.12	Colores . . . . .	19
<b>2</b>	<b>HTML5</b>	<b>21</b>
2.1	Modelos de contenido . . . . .	21
2.2	Estructura de páginas HTML5 . . . . .	22
2.2.1	headers, nav y footer . . . . .	22
2.2.2	article, section y aside . . . . .	24
2.2.3	<i>div, class e id</i> vs la estructura HTML5 . . . . .	25
2.3	La etiqueta audio . . . . .	26
2.4	La etiqueta video . . . . .	27
2.5	La etiqueta progress . . . . .	28
2.6	APIs . . . . .	29
2.6.1	Web Storage . . . . .	29
2.6.2	Geolocation . . . . .	29
2.6.3	Drag & Drop . . . . .	31
2.7	SVG . . . . .	32
2.7.1	Dibujando figuras . . . . .	32
2.7.2	Animaciones y rutas . . . . .	36
2.8	Canvas . . . . .	39
2.8.1	Transformaciones Canvas . . . . .	40
2.8.2	SVG vs Canvas . . . . .	43
2.9	Formularios HTML5 . . . . .	44

# Índice de Figuras

1	Vista de las etiquetas básicas . . . . .	7
2	Vista de las etiquetas de formato . . . . .	8
3	Vista de algunos atributos de etiquetas . . . . .	10
4	Vista las Listas . . . . .	11
5	Vista las Tablas . . . . .	13
6	Vista de Enlaces previo al clic . . . . .	14
7	Vista de Enlaces después del clic . . . . .	15
8	Vista de los Formularios . . . . .	17
9	Vista de los Framesets . . . . .	18
10	Vista de los Framesets redimensionados . . . . .	19
11	Ejemplo de colores HTML . . . . .	20
12	Relación de los modelos de contenido HTML5 . . . . .	22
13	Estructura básica de un sitio web HTML5 . . . . .	22
14	Ejemplo de header y footer . . . . .	24
15	Aspecto de la etiqueta audio . . . . .	27
16	Aspecto de la etiqueta video . . . . .	28
17	Aspecto de la etiqueta progress . . . . .	28
18	Formas de presentar la geolocalización . . . . .	30
19	Ejemplo de geolocalización . . . . .	31
20	Dibujando un círculo con SVG . . . . .	33
21	Dibujando un rectángulo con SVG . . . . .	33
22	Dibujando una línea con SVG . . . . .	34
23	Dibujando una ruta con SVG . . . . .	35
24	Dibujando una elipse con SVG . . . . .	35
25	Dibujando un polígono con SVG . . . . .	36
26	Dibujando una ruta con SVG y path . . . . .	37
27	Dibujando un cuadrado con Canvas . . . . .	40
28	Trasladando una gráfico con Canvas . . . . .	41
29	Rotando una gráfico con Canvas . . . . .	42
30	Escalando un gráfico con Canvas . . . . .	43

## Índice de Tablas

1	Comparación entre Canvas y SVG . . . . .	43
2	Atributos y etiquetas nuevas de formularios en HTML5 . . . . .	44

Este documento se hizo con **Overleaf** y los ejemplos fueron desarrollados y probados con **Visual Studio Code**, con su extensión **Live Server** en el buscador **Microsoft Edge** o **Brave**, por lo que algunas propiedades puede que lleguen a requerir un *Prefijo de buscador* para funcionar.

La estructura HTML de los ejemplos en este documento serán omitidos, dejando únicamente lo vital para que los ejemplos funcionen y para evitar que este trabajo sea muy largo, en caso de que sea necesario la estructura completa, se incluirá.

# 1 HTML básico

Las etiquetas que siempre debe contener un documento *.html* o *.htm* son:

```
<html>
  <head>
  </head>
  <body>
    This is a line of text.
  </body>
</html>
```

## 1.1 Tipos de etiquetas

- **Block-levels:** establecen una estructura o característica primordial dentro de la estructura HTML, suelen ser la cabecera o contenedor de otras etiquetas, y dan un salto de línea cuando estas etiquetas se cierran.
- **Inline:** son aquellas etiquetas que no van dentro de otras para dar formato al contenido de las mismas, y no dan un salto de línea cuando se cierran.
- Existen elementos que son Block-level e Inline, algunos ejemplos son:
  - **applet:** subprograma Java embebido.
  - **iframe:** estructura frame.
  - **ins:** texto insertado.
  - **map:** mapa de imagen.
  - **object:** objetos embebidos.
  - **script:** un script dentro del documento HTML.
- Los elementos Inline no pueden contener elementos Block-level.

## 1.2 Etiquetas básicas de HTML

- **p:** agrega un párrafo de texto al sitio o página.
- **span:** contiene texto en el sitio o página.
- **h, h1 h2, ..., h6:** agrega un título 1, título 2, ..., hasta título 6 al sitio o página. No es recomendable utilizar los títulos solamente para hacer grande el texto, ya que los buscadores los utilizan para indexar el sitio o página con la web.
- **title:** es el título de la pestaña del sitio o página.
- **hr/:** crea una línea horizontal. **No requiere de una etiqueta de comienzo.**
- **br/:** agrega un espacio en blanco o salto de línea. **No requiere de una etiqueta de comienzo.**

- **div**: es un elemento Block-level que suele ser utilizado para contener otras etiquetas.

Un ejemplo del aspecto de las etiquetas anteriormente mencionadas es el siguiente código y la *Figura 1*:

```
<!-- Tipos de encabezados. -->
<h1>Título 1</h1>
<h2>Título 1.1</h2>
<h3>Título 1.1.1</h3>
<h4>Título 1.1.1.1</h4>
<h5>Título 1.1.1.1.1</h5>
<h6>Título 1.1.1.1.1.1</h6>

<!-- Etiquetas para objetos. -->
<p>Esto es un párrafo, contenido en un P,<br/>acabo de dar un salto de línea con BR.</p>
<span>Esto es otro párrafo, contenido en un SPAN.</span>

<!-- Salto de línea. -->
<hr/>

<!-- Contenedor que almacena un párrafo. -->
<div>
    <p>Esto es otro párrafo, separado de los demás con una línea horizontal HR; contenido
        en un P dentro de un contenedor DIV.</p>
</div>
```

Figure 1: Vista de las etiquetas básicas

## Título 1

**Título 1.1**

**Título 1.1.1**

**Título 1.1.1.1**

**Título 1.1.1.1.1**

**Título 1.1.1.1.1.1**

Esto es un párrafo, contenido en un P,  
acabo de dar un salto de línea con BR.

Esto es otro párrafo, contenido en un SPAN.

---

Esto es otro párrafo, separado de los demás con una línea horizontal HR; contenido en un P dentro de un contenedor DIV.

## 1.3 Etiquetas de formato

- **b**: vuelve el texto negrita.
- **big**: vuelve un texto un poco más grande.
- **i**: vuelve el texto cursiva.
- **small**: vuelve el texto un poco más pequeño.

- **strong**: resalta un texto importante con negritas.
- **em**: resalta un texto importante con cursiva.
- **sub**: posiciona un texto en un subíndice.
- **sup**: posiciona un texto en un superíndice.
- **ins**: subraya un texto.
- **del**: tacha un texto.

Un ejemplo del aspecto de las etiquetas anteriormente mencionadas es el siguiente código y la *Figura 2*:

```
<b>Este texto es negrita.</b><br/>

<strong>Este texto es negrita e importante para el buscador.</strong><br/>

<i>Este texto es cursiva.</i><br/>

<em>Este texto es cursiva e importante para el buscador.</em><br/>

<ins>Este texto está subrayado.</ins><br/>

<del>Este texto está tachado</del><br/>

<big>Este texto es más grande.</big><br/>

<small>Este texto es más pequeño.</small><br/>

<sup>Este texto tiene un <sup>super índice.</sup><br/>

<sub>Este texto tiene un <sub>sub índice.</sub>
```

Figure 2: Vista de las etiquetas de formato

**Este texto es negrita.**  
**Este texto es negrita e importante para el buscador.**  
*Este texto es cursiva.*  
*Este texto es cursiva e importante para el buscador.*  
Este texto está subrayado.  
~~Este texto está tachado~~  
Este texto es más grande.  
Este texto es más pequeño.  
Este texto tiene un <sup>super índice</sup>.  
Este texto tiene un <sub>sub índice</sub>.

## 1.4 Atributos de etiquetas

Son información adicional que modifican a la etiqueta. Estos atributos poseen valores.

- **align:** alinea un texto o etiqueta (img, table, p, texto, h1, etc). Sus posibles valores son:
  - **left:** alinea y justifica a la derecha.
  - **center:** alinea y justifica al centro.
  - **right:** alinea y justifica a la izquierda
  - **justify:** justifica el texto.
  - **char:** alinea el texto según un carácter.
- **width:** ajusta el ancho de una etiqueta. Las unidades utilizadas para este atributo son el *porcentaje del ancho del sitio o página web o imagen* y los *píxeles*.
- **height:** ajusta el alto de una etiqueta. Las unidades utilizadas para este atributo son el *porcentaje del ancho del sitio o página web o imagen* y los *píxeles*.
- **src:** enlaza links o archivos al sitio o página. Suele ser utilizado con la etiqueta *img* o *a*.
- **alt:** despliega o muestra una imagen o texto alternativo en caso de que el principal no pueda ser mostrado o cargado. Suele ser utilizado con la etiqueta *img* y es **obligatorio**.
- **border:** agrega un marco o borde a una etiqueta (imagen o tabla). El grosor del borde se establece mediante píxeles. Este atributo puede explotarse más con CSS.
- **bgcolor:** cambia el color de fondo de un elemento.

Un ejemplo del aspecto de algunos de los atributos anteriormente mencionados es el siguiente código y la *Figura 3*:

```
<!-- Imagen con distintos atributos. -->
![https://images.hola.com/imagenes/mascotas/20180925130054/consejos-para-cuidar-a-un-
    -gatito-recien-nacido-cs/0-601-526/cuidardgatito-t.jpg](https://images.unsplash.com/photo-1593288942460-e321b92a6cde?ixlib=rb-4.0.3&ixid=
    MnwxMjA3fDB8MHxleHBsb3JlLWZlZWR8Mnx8fGVufDB8fHx8&w=1000&q=80)
```

Figure 3: Vista de algunos atributos de etiquetas



Vemos que el ejemplo anterior aplica diversos atributos a una etiqueta **img**, se le asigna una imagen predeterminada y alternativa con los atributos *src* y *alt* respectivamente, un ancho y alto con *width* y *height* y un borde de 10 píxeles con *border*.

Si un atributo choca con otro, es decir, se centra el contenido de una etiqueta *p* y, dentro de este, existe otra etiqueta *p* que alinea el texto a la izquierda, el buscador alinearán todas las sub-etiquetas de la etiqueta *p* principal según el atributo de alineación de esta última, y esto aplica para otros tipos de atributos o situaciones.

## 1.5 Comentarios

Un comentario comienza con `<!--` y termina con `-->`:

```
<!-- Esto es un comentario. -->
```

## 1.6 Listas

Las etiquetas **ol** y **ul** generan una lista numerada y no numerada respectivamente. La etiqueta **li** representa los puntos o elementos de una lista numerada o no numerada.

Si las listas son contenidas dentro de un **div** o **p**, se le pueden poner los atributos *align*, *width*, *height* y *border*.

Un ejemplo del aspecto de listas es el siguiente, junto con la *Figura 4*:

```
<h2>Lista numerada</h2>
<ol>
    <!-- Comienza desde 1. -->
    <li>Opción numerada 1</li>
    <li>Opción numerada 2</li>
    <li>Opción numerada 3</li>
    <li>Opción numerada 4</li>
</ol>
<h2>Lista numerada comenzando a partir del 4</h2>
<ol>
    <!-- Comienza desde 4. -->
    <li value="4">Opción numerada 1</li>
    <li>Opción numerada 2</li>
    <li>Opción numerada 3</li>
    <li>Opción numerada 4</li>
</ol>
<h2>Lista no numerada</h2>
```

```
<ul>
  <li>Opción no numerada 1</li>
  <li>Opción no numerada 2</li>
  <li>Opción no numerada 3</li>
  <li>Opción no numerada 4</li>
</ul>
```

Figure 4: Vista las Listas

### **Lista numerada**

4. Opción numerada 1
5. Opción numerada 2
6. Opción numerada 3
7. Opción numerada 4

### **Lista numerada comenzando a partir del 4**

4. Opción numerada 1
5. Opción numerada 2
6. Opción numerada 3
7. Opción numerada 4

### **Lista no numerada**

- Opción no numerada 1
- Opción no numerada 2
- Opción no numerada 3
- Opción no numerada 4

*Nota:* este es el aspecto predeterminado de las listas en HTML, antes se le podía dar otro aspecto a ambos tipos de listas con un atributo *type*, sin embargo, este ya no es utilizado, para sustituirlo, se recomienda personalizar o estilizar las listas con CSS.

## 1.7 Tablas

La etiqueta **table** genera una tabla. Sus sub-etiquetas y atributos son:

- **tr**: etiqueta que representa una fila en la tabla. Dentro de estas, va el contenido, celdas o columnas de cada fila.
- **td**: etiqueta que representa una celda, contenido o columna de una fila de una tabla.
- **th**: etiqueta que representa la cabecera o título de cada columna de una tabla.
- **colspan**: atributo que combina *n* cantidad de columnas o celdas en una tabla.
- **rowspan**: atributo que combina *n* cantidad de filas en una tabla.
- Los atributos *align*, *width* y *height* aplican a las etiquetas **tr** y **td**.
- Los atributos *width*, *height*, *bgcolor* y *border* aplican a la esta etiqueta y a las sub-etiquetas.

Un ejemplo del aspecto de las tablas es el siguiente, junto con la *Figura 5*:

```
<h2>Tabla regular</h2>
<!-- Borde de 2 píxeles, ancho y alto de 200 píxeles. -->
<table border="2px" width="200px" height="200px">
    <!-- Elementos alineados al centro. -->
    <tr align="center">
        <td>1</td>
        <td>2</td>
        <td>3</td>
    </tr>
    <tr align="center">
        <td>4</td>
        <td>5</td>
        <td>6</td>
    </tr>
    <tr align="center">
        <td>7</td>
        <td>8</td>
        <td>9</td>
    </tr>
</table>
<h2>Tabla con columnas combinadas</h2>
<table border="2px" width="200px" height="200px">
    <tr align="center">
        <!-- Crea dos celdas que combinan dos columnas. -->
        <td colspan="2">1</td>
        <td>2</td>
    </tr>
    <tr align="center">
        <td>3</td>
        <td>4</td>
        <td>5</td>
        <td>6</td>
    </tr>
    <tr align="center">
        <td>7</td>
        <td>8</td>
        <td>9</td>
        <td>10</td>
    </tr>
</table>
<h2>Tabla con filas combinadas</h2>
<table border="2px" width="200px" height="200px">
    <tr align="center">
        <!-- Crea una fila que combina tres celdas. -->
        <td rowspan="3">1</td>
        <td>2</td>
        <td>3</td>
        <td>4</td>
    </tr>
    <tr align="center">
        <td>7</td>
        <td>5</td>
    </tr>
</table>
```

```

<td>6</td>
</tr>
<tr align="center">
<td>8</td>
<td>9</td>
<td>10</td>
</tr>
</table>

```

Figure 5: Vista las Tablas

**Tabla regular**

1	2	3
4	5	6
7	8	9

**Tabla con columnas combinadas**

1		2	
3	4	5	6
7	8	9	10

**Tabla con filas combinadas**

	2	3	4
1	7	5	6
	8	9	10

Vemos que el ejemplo anterior posee tres tablas, con un borde de 2 píxeles, con todo su contenido alineado al centro y con un tamaño de ancho y alto de 200 píxeles, la primer tabla es una tabla regular de 3x3, la segunda tabla combina dos y dos columnas y la tercer tabla combina tres filas.

## 1.8 Imágenes

Enlaza una imagen al sitio o página web con la etiqueta **img**. **Esta etiqueta no requiere una etiqueta de cierre**. Las imágenes son pesadas, es recomendado utilizar imágenes del tamaño que se deseen o requieran mostrar, ni tan grandes ni tan pequeñas, y en el formato adecuado (.jpg, .png u otros). Sus atributos son:

- **src**: la ruta o dirección web de la imagen.
- **alt**: el texto o imagen alternativo en caso de que la imagen principal no cargue.
- Los atributos *align*, *height*, *width* y *border* aplican a esta etiqueta.

El uso de imágenes ya se vio en la Figura 3.

## 1.9 Enlaces

Enlaza a un documento, imagen, archivo, sitio o página o dirección web con la etiqueta **a**. Sus atributos son:

- **href**: es el enlace a donde se redireccionará.
- **target**: establece como dónde y cómo se abrirá el enlace o documento. Sus posibles valores son:
  - **\_blank**: carga en una nueva pestaña.
  - **\_self**: carga en el actual *frame*.
  - **\_parent**: carga en el *frame* padre o del *frame* actual.
  - **\_top**: carga en la pestaña actual.
  - **framename**: carga en un *frame* determinado.
- El atributo *align* aplica a esta etiqueta.

Un ejemplo del aspecto de los enlaces es el siguiente, junto con la *Figura 6*, donde se muestra el aspecto del enlace antes de dar clic en él, la *Figura 7* muestra que se ha abierto otra pestaña después de haber dado clic en el enlace:

```
<a href="https://github.com/migueluisV" target="_blank">Github</a>
```

Figure 6: Vista de Enlaces previo al clic

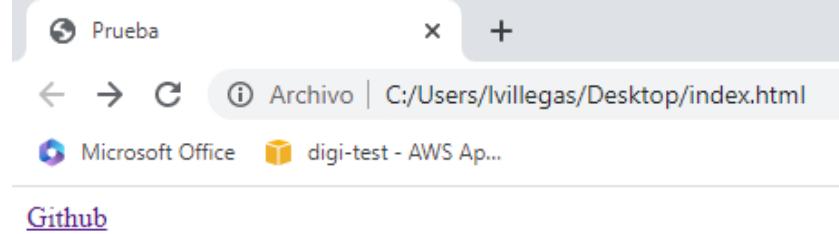
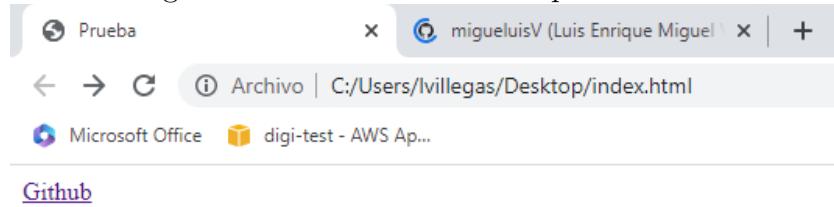


Figure 7: Vista de Enlaces después del clic



Al enlace se le configuró el atributo *target* con el valor **\_blank** para que se abriera el enlace en una nueva pestaña, puede probar con los otros valores de este atributo para ver el comportamiento.

## 1.10 Formularios

La etiqueta **form** crea un formulario para recolectar información del usuario, esta información es enviada a un servidor u otro sitio o página, siendo procesada por un lenguaje de programación (PHP o JavaScript por ejemplo). Sus sub-etiquetas y atributos son:

- **input**: etiqueta que crea un control para que el usuario interactúe con el. Sus atributos son:
  - **type**: es el tipo de control a generar. Sus posibles valores son:
    - \* **text**: caja de texto.
    - \* **password**: caja de texto con caracteres especiales para proteger texto.
    - \* **radio**: RadioButton.
    - \* **checkbox**: caja de marcado.
    - \* **submit**: botón que envía la información del formulario.
  - **placeholder**: texto por defecto que se muestra en una caja de texto.
  - **name**: nombre único del control.
  - **value**: el valor que almacena el control, se suele aplicar a *submit*, *radio* y *checkbox*.
- **textarea**: caja de texto multilínea. Tiene un inicio y final.
  - **rows**: indica las filas por defecto de la textarea.
  - **cols**: indica las columnas por defecto de la textarea.
  - Los atributos *name* y *placeholder* aplican a esta etiqueta.
- **action**: atributo de **form** que carga una dirección web una vez se llenó y envió el formulario. Si se le pone el valor **#** a este atributo, el formulario cargará la página donde está contenido (se refrescará la página sin cambio alguno).
- **method**: atributo de **form** especifica el método HTTP cuando el formulario es enviado. Sus posibles valores son:

- **GET**: la información recolectada puede verse en la dirección web del sitio a donde fue mandada.
  - **POST**: la información no es visible, y ofrece mejor seguridad si es que la información es sensible o se está actualizando.
- Dependiendo de cómo se presente el formulario (con **div**, **span**, **p** u otros), se le pueden aplicar los atributos más populares anteriormente vistos.

Un ejemplo del aspecto de los enlaces es el siguiente, junto con la *Figura 8*:

```
<form action="#" method="POST">
    p>Caja de texto: </p><input type="text" name="txt1" placeholder="Ingresa un texto..." /
        >

    <p>Caja de texto multi línea: </p><textarea name="txta1" rows="10" cols="50"
        placeholder="Ingresa un texto..."></textarea>

    <p>Contraseña: </p><input type="password" name="psw1" />

    <p>RadioButtons: </p>
    <input type="radio" name="rb1" value="1" />Valor 1
    <br/>
    <input type="radio" name="rb1" value="2" />Valor 2

    <p>Checkbox: </p>
    <input type="checkbox" name="chb1" value="3" />Opción 1
    <br/>

    <input type="checkbox" name="chb2" value="4" />Opción 2

    <p>Botones: </p><input type="submit" name="btn1" value="Enviar" />
</form>
```

Figure 8: Vista de los Formularios

Caja de texto:

hola mundo

Caja de texto multi línea:

```
asdfsadf
asdfsadf
asdfsadf
sdfasdf
asdfasdf
asdfsad
asdfsadf
```

Contraseña:

\*\*\*\*\*

RadioButtons:

Valor 1  
 Valor 2

Checkbox:

Opción 1  
 Opción 2

Botones:

Enviar

Se creó un formulario con una caja de texto, una caja de texto multilínea, una caja para contraseñas, dos opciones de RadioButtons, dos opciones de CheckBoxs y un botón que envía la información recaba al mismo sitio. El formulario tiene el atributo *action* y *method* con los valores **#** y **POST** respectivamente, envía la información protegida al mismo sitio donde está contenido el formulario. Algunas cajas de texto tienen el atributo *placeholder*, para que se muestre un texto predeterminado que se elimina una vez el usuario ingresa algún texto. El control textarea posee los atributos *rows* y *cols* para mostrar una cantidad determinada de filas y columnas respectivamente, esto para resaltar que es un control multilínea.

Fíjese que el nombre de los dos RadioButtons utilizados es el mismo, esto es así ya que este control solo permite seleccionar una opción, si tuviéramos dos o más RadioButtons con distinto nombre, se podrían seleccionar todos sin problema; esto no ocurre con los CheckBoxs.

Todos los controles utilizados en el ejemplo anterior poseen su nombre único con el atributo *name*; **existen más controles**.

## 1.11 Frames

Los **frames** son contenedores donde podemos almacenar distintas páginas o contenido dentro de una sola ventana o página. El **frameset** es la etiqueta Block-Level de los **frames**, esta define cuantos frames habrá en el frameset (columnas y filas), su espacio definido, borde, etc.

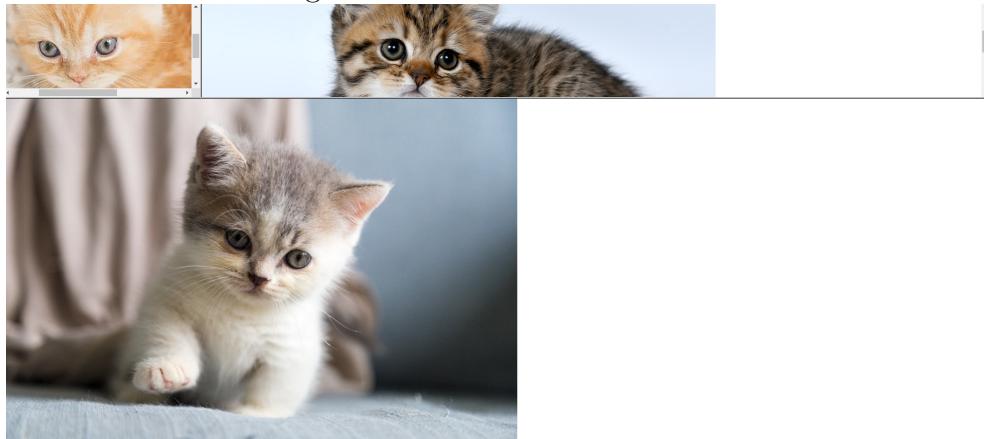
Esta etiqueta o estructura ya no está soportada en HTML5. **frameset** sustituye la etiqueta **body** en un documento HTML. Algunos de sus atributos son:

- **cols**: atributo que establece en cuántas columnas se seccionará la página web. El tamaño de estas columnas se determina por porcentajes o píxeles.
- **rows**: atributo que establece en cuántas filas se seccionará la página web. El tamaño de estas filas se determina por porcentajes o píxeles.
- **src**: atributo que determina el archivo o sitio web que se cargará en el frame.
- **noresize**: atributo que indica que un frame no puede cambiar su tamaño.
- **noframe**: etiqueta que indica al buscador que la página o sitio no soporta frames.

Veamos un ejemplo tomado de internet modificado por nosotros enseguida, en la *Figura 9*:

```
<html>
  <head>
    <title>Un documento simple con marcos</title>
  </head>
  <frameset rows="20%,80%">
    <frameset cols="20%, 80%">
      <frame src="https://images.hola.com/imagenes/mascotas/20180925130054/consejos-
        para-cuidar-a-un-gatito-recien-nacido-cs/0-601-526/cuidardgatito-t.jpg">
      <frame src="https://images.unsplash.com/photo-1593288942460-e321b92a6cde?ixlib=
        rb-4.0.3&ixid=MnwxMjA3fDB8MHxleHBsb3JlLWZlZWR8Mnx8fGVufDB8fHx8&w=1000&q=80">
    </frameset>
    <frame src="https://images.unsplash.com/photo-1591871937631-2f64059d234f?ixlib=rb
      -4.0.3&ixid=MnwxMjA3fDB8MHxleHBsb3JlLWZlZWR8MXx8fGVufDB8fHx8&w=1000&q=80">
  </frameset>
</html>
```

Figure 9: Vista de los Framesets



El ejemplo presenta un **frameset** principal que contiene dos columnas, una del tamaño del 20% de la pantalla y la otra del 80%: el primer **frame** contiene otro *frameset* con dos filas, igual con tamaños de 20 y 80 porcientos, así combinamos el uso de framesets. El principal posee un borde de 3 píxeles, que afecta al resto de frames o framesets internos, vemos que aquellos que requieren de desplazamiento se les inserta automáticamente barras deslizadoras para recorrer el frame, si ponemos el puntero en los bordes, estos pueden ser redimensionados, como se ve en la *Figura 10*:

Figure 10: Vista de los Framesets redimensionados



En caso de que no se requiera la posibilidad de redimensionamiento de frames, aplicar el atributo *noresize*. **frameset ya no es soportado por HTML5**.

## 1.12 Colores

Los colores son representados como valores hexadecimales:

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

Los colores en HTML son desplegados según el modelo RGB (Red, Green, Blue), para escribir un color es necesario escribir el símbolo de gato (#), seguido de su código de combinación de tres o seis, como vemos en la *Figura 11*:

Figure 11: Ejemplo de colores HTML



## 2 HTML5

Algunas de las nuevas características que incluye la versión más reciente de HTML son los formularios 2.0, que traen controles para seleccionar fechas, colores, números, cajas de texto para correos electrónicos, búsqueda, URL, los métodos PUT y DELETE, entre otros; el tipo de documento previo al inicio de la estructura HTML, nuevas etiquetas, la posibilidad de modificar la codificación de caracteres, entre otros temas que se verán en este documento.

### 2.1 Modelos de contenido

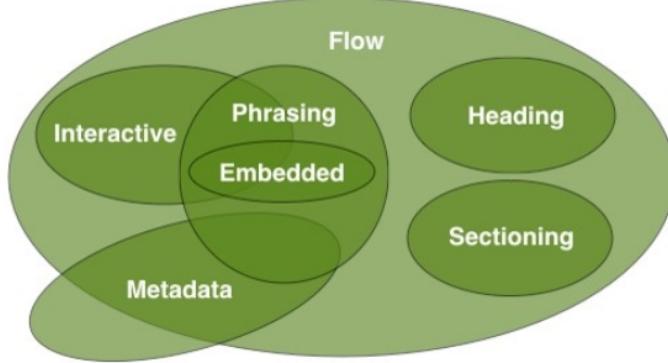
Los **modelos de contenido** son modelos de etiquetas que responden a una misma función o que tienen un fin determinado, los modelos pueden ir enfocados a la interacción, la organización, inserción, formateo o adición de información de una página o sitio web. Estos modelos suelen contener ciertas etiquetas que responden al fin del mismo.

Anteriormente, habíamos mencionado que HTML tenía etiquetas Block-Level e Inline, donde la segunda suele pertenecer a la primera (revisar la sección **Tipos de elementos**), en esta versión del lenguaje, se agregan siete modelos nuevos:

1. Metadata: establece información adicional o comportamiento al resto del contenido de la página o sitio. Podemos encontrar las siguientes etiquetas en este modelo: **base, link, meta, noscript, script, style, title**.
2. Embedded: contenido que es importado a la estructura o contenido de la página o sitio. Podemos encontrar las siguientes etiquetas en este modelo: **audio, video, canvas, iframe, img, math, object, svg**.
3. Interactive: contenido hecho para que el usuario interactue con él. Podemos encontrar las siguientes etiquetas en este modelo: **a, audio, video, button, details, embed, iframe, img, input, label, object, select, textarea**.
4. Heading: establece cabeceras o secciones en el documento. Podemos encontrar las siguientes etiquetas en este modelo: **h1 a h6 y hgroup**.
5. Phrasing: es un modelo para formatear contenido, posee varias etiquetas compartidas con HTML4: **img, span, strong, label, br/, small, sub** y más.
6. Flow: contiene la mayoría de las etiquetas que serán utilizadas en la estructura, flujo o funcionamiento del documento. Se puede decir que este modelo incluye la mayoría del resto.
7. Sectioning: establece el objetivo de las secciones, contenido, navegación o pies de páginas. Podemos encontrar las siguientes etiquetas en este modelo: **article, aside, nav, section**.

La *Figura 12* nos muestra un diagrama de como se relacionan estos modelos:

Figure 12: Relación de los modelos de contenido HTML5

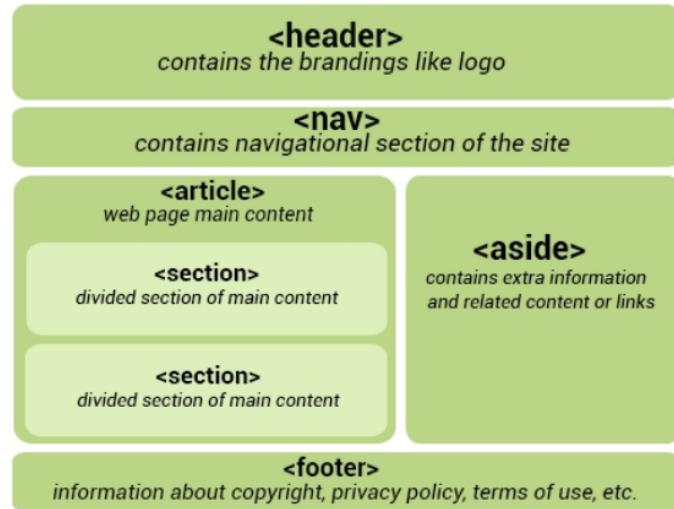


Ojo, no es como que tengamos que escoger un modelo para desarrollar nuestro proyecto, simplemente son contenedores de etiquetas que tienen un fin específico.

## 2.2 Estructura de páginas HTML5

La estructura básica o más simple de un sitio web moderno HTML5 se aprecia en la *Figura 13* (no es necesario que se agreguen todas las secciones de esta estructura):

Figure 13: Estructura básica de un sitio web HTML5



### 2.2.1 headers, nav y footer

La etiqueta **header** es la cabecera del sitio, donde está el logo del sitio o información más importante del sitio, si queríamos crear un *header* con HTML4, debíamos utilizar un contenedor **div**:

```
<div id="header">
```

Con HTML5, poseemos la etiqueta **header** que vuelve más fácil esta tarea:

```
<!-- Cabecera del sitio. -->
<header>
    <h1>Nombre y logo del sitio</h1>
</header>
<!-- Resto del sitio. -->
<h3>Esto es el cuerpo del sitio</h3>
<h3>Esto es el cuerpo del sitio</h3>
<h3>Esto es el cuerpo del sitio</h3>
```

La etiqueta **nav** es un menú horizontal que te permite navegar a lo largo de todo el sitio web, por lo que *nav* está constituido de enlaces:

```
<!-- Cabecera del sitio. -->
<header>
    <h1>Nombre y logo del sitio</h1>
</header>
<!-- Barra de navegación del sitio. -->
<nav>
    <ul>
        <li><a href="#">Home</a></li>
        <li><a href="#">Servicios</a></li>
        <li><a href="#">Acerca de nosotros</a></li>
        <li><a href="#">Organigrama</a></li>
    </ul>
</nav>
<!-- Resto del sitio. -->
<h3>Esto es el cuerpo del sitio</h3>
<h3>Esto es el cuerpo del sitio</h3>
<h3>Esto es el cuerpo del sitio</h3>
```

La etiqueta **footer** es el pie de página del sitio o página, nos permite poner información hasta abajo de la página: este tipo de información suele ser la información de contacto, política de privacidad, redes sociales, términos de uso, información de Copyright, documentos relacionados, mapa del sitio, etc:

```
<!-- Cabecera del sitio. -->
<header>
    <h1>Nombre y logo del sitio</h1>
</header>
<!-- Barra de navegación del sitio. -->
<nav>
    <ul>
        <li><a href="#">Home</a></li>
        <li><a href="#">Servicios</a></li>
        <li><a href="#">Acerca de nosotros</a></li>
        <li><a href="#">Organigrama</a></li>
    </ul>
</nav>
<!-- Cuerpo del sitio. -->
<h3>Esto es el cuerpo del sitio</h3>
<h3>Esto es el cuerpo del sitio</h3>
<h3>Esto es el cuerpo del sitio</h3>
<!-- Pie del sitio. -->
<footer>
```

```
<h3>Mapa del sitio/h3>
</footer>
```

La *Figura 14* muestra el resultado de los dos ejemplos anteriores:

Figure 14: Ejemplo de header y footer

## Nombre y logo del sitio

- [Home](#)
- [Servicios](#)
- [Acerca de nosotros](#)
- [Organigrama](#)

**Esto es el cuerpo del sitio**

**Esto es el cuerpo del sitio**

**Esto es el cuerpo del sitio**

**Mapa del sitio/h3>**

*Nota:* el buscador ya tiene definido la información de la cabecera, menú de navegación y pie de página, aunque visualmente no parezca nada relevante.

### 2.2.2 article, section y aside

La etiqueta **article** representa un contenedor para un artículo, foro, post, revista, artículo de periódico, entrada de blog, un comentario, un *widget* o *gadget* interactivo u otro elemento independiente del contenido del sitio, las etiquetas *article* pueden ser anidadas unas dentro de otras, y estos contenedores deben ser independiente al resto del contenido de la página o sitio web. Veamos el siguiente ejemplo:

```
<article>
  <article>
    <h2>Nombre del artículo</h2>
    <h1>Autor</h1>
  </article>
  <article>
    <h4>Fecha de publicación</h4>
    <h4>Sitio de publicación</h4>
  </article>
  <article>
    <p>Contenido del artículo</p>
  </article>
</article>
```

Anidamos etiquetas *article* dentro de una etiqueta mayor *article*, esto con el fin de que cada etiqueta anidada sea un dato o sección propiamente de un artículo que se esté insertando al documento HTML.

La etiqueta **section** es un contenedor para diversas etiquetas HTML, estas etiquetas contenidas deben tener algo en común, y es que deben pertenecer a una sección de la página o sitio web; algunas etiquetas que pueden ser utilizadas en las secciones son: **h1-h6**, **p**, otra etiqueta **section** o **article**. Veamos el siguiente ejemplo:

```
<article>
  <section>
    <h2>Nombre del artículo</h2>
    <h1>Autor</h1>
  </section>
  <section>
    <h4>Fecha de publicación</h4>
    <h4>Sitio de publicación</h4>
  </section>
  <section>
    <p>Contenido del artículo</p>
  </section>
</article>
```

Se sustituyen los *articles* anidados por *sections*. La diferencia entre *article* y *section* es que, *article* va enfocado a contener contenido independiente del sitio web, mientras que *section* puede contener otros artículos o secciones del propio sitio web.

La etiqueta **aside** representa contenido relacionado a otro contenido, pudiendo ser este de una etiqueta *article*, de un *section* o del propio sitio web. Veamos un ejemplo:

```
<article>
  <article>
    <h2>Nombre del artículo</h2>
    <h1>Autor</h1>
  </article>
  <article>
    <h4>Fecha de publicación</h4>
    <h4>Sitio de publicación</h4>
  </article>
  <article>
    <p>Contenido del artículo</p>
  </article>
  <aside>
    <a href="#">Vea estos otros artículos similares</a>
  </aside>
</article>
```

Si copia algunos de los ejemplos anteriores y lo ejecuta en el navegador de su preferencia, verá un sitio bastante sencillo ni diseño particular, pero para el buscador, la existencia de una etiqueta *article*, *section* o *aside* le permitirá una estructuración determinada del sitio.

### 2.2.3 *div*, *class* e *id* vs la estructura HTML5

Veamos sus diferencias:

- **div, class e id**: contenedores de uso muy general, utilizados para contener etiquetas HTML y aplicarles un estilo o evento.

- **article**: contenedor de uso muy específico, para contenido independiente al del sitio o página web.
- **section**: contenedor de uso general, utilizado para contener otras etiquetas *article* o *HTML*, pero que tengan una relación entre sí.
- **aside**: contenedor para contenido secundario relacionado con respecto a otro.

## 2.3 La etiqueta audio

Inserta un archivo de audio a la página o sitio web. Estos son los formatos de audio que aceptan los navegadores más populares:

- Internet Explorer: mp3.
- Edge: mp3 y wav.
- Google Chrome: mp3, wav y ogg.
- Firefox: mp3, wav y ogg.
- Safari: mp3 y wav.
- Opera: wav y ogg.

Por este motivo, es que debemos utilizar el siguiente ejemplo para insertar audios a nuestros proyectos:

```
<!-- Mejor método. -->
<audio controls>
    <source src="audio1.mp3" type="audio/mpeg">
    <source src="audio2.ogg" type="audio/ogg">
    <source src="audio3.wav" type="audio/wav">
    <!-- Texto a mostrar en caso de que no se pueda reproducir el audio. -->
    Audio no soportado por el buscador.
</audio>

<!-- Alternativa. -->
<audio src="audio1.mp3" controls>
    <!-- Texto a mostrar en caso de que no se pueda reproducir el audio. -->
    Audio no soportado por el buscador.
</audio>
```

El mejor método es denominado así porque el buscador reproduce el audio con el formato que primero reconozca, si no reconoce uno inmediatamente, pasa al siguiente, así sucesivamente; el otro método está limitado a únicamente un audio. Veamos sus atributos:

- **src**: es el archivo a reproducir.
- **autoplay**: reproduce el audio apenas cargue el sitio.
- **loop**: repite el audio indefinidamente.

- Los últimos dos atributos pueden funcionar o no según el buscador.

La *Figura 15* muestra el aspecto de un control **audio** en el buscador Google Chrome (cambia este aspecto entre buscadores):

Figure 15: Aspecto de la etiqueta audio



## 2.4 La etiqueta video

Inserta un archivo de vídeo a la página o sitio web. Estos son los formatos de audio que aceptan los navegadores más populares:

- Internet Explorer: mp4.
- Edge:
- Google Chrome: mp4, webm y ogg.
- Firefox: mp4, webm y ogg.
- Safari: mp4.
- Opera: webm y ogg.

Posee la misma declaración que las etiquetas *audio*, pero posee atributos muy interesantes:

```
<!-- Mejor método. -->
<video controls>
  <source src="video1.mp4" type="video/mp4">
  <source src="video2.ogg" type="video/ogg">
  <source src="video3.webm" type="video/webm">
  <!-- Texto a mostrar en caso de que no se pueda reproducir el video. -->
  Videono soportado por el buscador.
</video>

<!-- Alternativa. -->
<video src="video1.mp4" controls>
  <!-- Texto a mostrar en caso de que no se pueda reproducir el video. -->
  Video no soportado por el buscador.
</video>
```

- **src**: es el archivo a reproducir.
- **autoplay**: reproduce el vídeo apenas cargue el sitio.
- **loop**: repite el vídeo indefinidamente.

- **preload**: carga los datos del vídeo por adelantado.
- **controlList="nodownload"**: evita que el control para descargar el video aparezca.
- **width y height**: dimensionan el vídeo.
- **muted**: reproduce el vídeo sin audio.
- **poster**: asigna una miniatura, cartel o póster al vídeo, previo a su reproducción.

La *Figura 16* muestra el aspecto de un control **video** en el buscador Google Chrome (cambia este aspecto entre buscadores):

Figure 16: Aspecto de la etiqueta video



## 2.5 La etiqueta progress

La etiqueta **progress** permite insertar una **Progress Bar** (barra de progreso) en la página o sitio. Sus atributos más importantes son:

- **value**: especifica cuánto de la tarea ha sido completada.
- **min**: especifica el valor inicial de la tarea.
- **max**: especifica el total de trabajo que la tarea debe completar.

*Estado: <progress min="0" max="100" value="35"></progress>*

Resultado en Google Chrome (*Figura 17*):

Figure 17: Aspecto de la etiqueta progress

Estado: A horizontal progress bar with a blue segment indicating progress.

Esta etiqueta suele ser utilizada en conjunto con JavaScript, para completar tareas o procesos relacionados con un algo más que las etiquetas de HTML.

## 2.6 APIs

### 2.6.1 Web Storage

El almacenamiento web de HTML5 permite guardar información y datos en la computadora del usuario que acceda a nuestro sitio y genera datos, con esto, podemos acceder a ellos si es que el usuario entra en otro momento, evitamos enviar la información nuevamente (si es que ya fue almacenada), tomarla si ya fue almacenada, es más seguro, etc. Para utilizar esta API, es necesario conocer JavaScript.

#### Tipos del almacenamiento web.

- **sessionStorage()**: crea una sesión y almacena información en la misma; esta información y sesión serán destruidas una vez el usuario cierre el buscador.
- **localStorage()**: crea una sesión sin expiración y almacena la información en la máquina del usuario.

#### Operaciones con el almacenamiento web.

Todas estas operaciones son trabajadas con JavaScript, ya sea con un almacenamiento local o de sesión, los datos son almacenados por medio de un **par llave/valor**:

```
// Guardando datos.  
localStorage.setItem("llave1", "valor1");  
sessionStorage.setItem("llave1", "valor1");  
  
// Obteniendo e imprimiendo datos guardados.  
alert(localStorage.getItem("llave1"));  
alert(sessionStorage.getItem("llave1"));  
  
// Eliminando un dato.  
localStorage.removeItem("llave1");  
sessionStorage.removeItem("llave1");  
  
// Eliminando todos los datos.  
localStorage.clear();  
sessionStorage.clear();
```

### 2.6.2 Geolocation

Es la ubicación actual del usuario (siendo más precisa en dispositivos con GPS), este dato compromete la privacidad de los individuos, por lo que el sitio web debe preguntar al usuario si este desea compartir su ubicación:

```
navigator.geolocation.getCurrentPosition();
```

Donde los parámetros de *getCurrentPosition()* son:

- **showLocation**: obligatorio, establece un método que recibe la información de la ubicación.

- **ErrorHandler:** opcional, establece un método en caso de que un error ocurra en la llamada asíncrona.
- **Options:** opcional, define algunas opciones a la hora de recibir la información de la ubicación.

### Presentando la información

La información de la ubicación puede ser presentada en las siguientes formas:

- **Geometric:** es presentada según latitud y longitud.
- **Civic:** es presentada de una manera más simple para una persona, por medio de ciudades y otros datos civiles.

La *Figura 18* lo explica mejor:

Figure 18: Formas de presentar la geolocalización

Attribute	Geometric	Civic
<b>Position</b>	<b>59.3, 18.6</b>	<b>Stockholm</b>
<b>Elevation</b>	<b>10 meters</b>	<b>4 th floor</b>
<b>Heading</b>	<b>234 degrees</b>	<b>City center</b>
<b>Speed</b>	<b>5km / h</b>	<b>Walking</b>
<b>Orientation</b>	<b>45 degrees</b>	<b>North-East</b>

Nota: el método **getCurrentPosition()** siempre regresa un objeto, donde los atributos *latitude*, *longitude* y *accuracy* siempre son retornados.

El siguiente ejemplo nos ayudará a comprender la forma de trabajar la geolocalización (*Figura 19*):

```
<p>Click the button to get your coordinates.</p>
<!-- Botón que al presionarlo llama a la función "getLocation" del script insertado. -->
<button onclick="getLocation()">Try It</button>

<p id="demo"></p>

<script>
    var x = document.getElementById("demo");

    // Obtiene la localización del buscador.
    function getLocation() {
        if (navigator.geolocation) {
            navigator.geolocation.getCurrentPosition(showPosition);
        }
        else {
            x.innerHTML = "Geolocation is not supported by this browser.";
        }
    }
    // Muestra la localización en HTML.
    function showPosition(position) {
```

```

        x.innerHTML = "Latitude: " + position.coords.latitude +
        "<br>Longitude: " + position.coords.longitude;
    }
</script>

```

Figure 19: Ejemplo de geolocalización  
 Click the button to get your coordinates.

[Try It](#)

Latitude: 32.517548  
 Longitude: -116.973457

### 2.6.3 Drag & Drop

Todos las etiquetas y objetos HTML pueden ser *draggable*, solamente añada el atributo **draggable** con su valor **true**:

`<img draggable="true" />`

Esto es por parte de HTML, la API del Drag & Drop está basada en un evento JavaScript, como vemos en el siguiente ejemplo:

```

<html>
  <head>
    <!-- Script interno. -->
    <script>
      // Permite el arrastrado de etiquetas.
      function allowDrop(ev) {
        ev.preventDefault();
      }
      // Realiza la acción de arrastrar la etiqueta y su información.
      function drag(ev) {
        ev.dataTransfer.setData("text", ev.target.id);
      }
      // Realiza la acción de soltar la etiqueta y su información.
      function drop(ev) {
        ev.preventDefault();
        var data = ev.dataTransfer.getData("text");
        ev.target.appendChild(document.getElementById(data));
      }
    </script>
  </head>
  <body>
    <!-- Establece contenedor para soltar un elemento. -->
    <div id="box" ondrop="drop(event)" ondragover="allowDrop(event)"
      style="border:1px solid black; width:200px; height:200px">
    </div>
    <!-- Etiqueta capaz de ser arrastrada. -->
    
    </body>
</html>
```

La explicación es la siguiente: cuando una etiqueta u objeto es arrastrado (dragged), el atributo **ondragstart** llama a una función: **drag(event)**, el cual especifica la información a arrastrar. El método **dataTransfer.setData()** especifica el tipo y valores de la información arrastrada; este método te permite acceder también a la información del objeto o etiqueta arrastrada, regresará cualquier información asignada al tipo dado en el método **setData()**, este tipo puede ser el ID del elemento arrastrado (imágenes, videos, audios, paneles, etc).

El evento **ondragover** especifica dónde la información puede ser soltada. Por defecto, la información o etiquetas no pueden ser soltadas sobre otras etiquetas, por lo que debemos prevenir esto mediante el método **preventDefault()**; permite prevenir al buscador que se cambiará la forma por defecto para manejar la información (*por defecto está como enlace a soltar, no otros objetos*).

Al soltar el clic y soltar (drop) la etiqueta, objeto o información, ocurre el evento *drop*. El atributo **ondrop** va ligado al objeto que recibirá lo que sea que se le suelte, a quien recibirá la información u objeto.

## 2.7 SVG

SVG significa **Scalable Vector Graphics** (Gráficos vectoriales escalables), y es utilizado para dibujar figuras con un estilo HTML. En HTML funciona tanto como una etiqueta, como en un formato de imagen para una etiqueta *img*, y la podemos utilizar para dibujar cajas, líneas, círculos, textos y gráficas en imágenes (recordemos que los formatos *.svg* puede ser aumentada o disminuida de tamaño sin perder calidad).

```

```

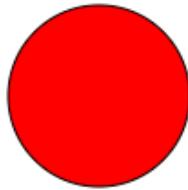
La etiqueta **svg** debe tener los atributos *width* y *height* de manera obligatoria, para dimensionar el panel o espacio donde será dibujada la figura; tome en cuenta que, si este espacio de dibujo es menor al tamaño de la figura, esta se dibujará incompleta en el espacio de dibujo.

### 2.7.1 Dibujando figuras

Crearemos un **círculo** en el siguiente ejemplo con su visualización (*Figura 20*):

```
<!-- Crea un panel de dibujo SVG de 3000x3000 píxeles. -->
<svg width="3000" height="3000">
    <!-- Dibuja un círculo de radio 50, de color rojo y borde negro.. -->
    <circle cx="100" cy="150" r="50" fill="red" stroke="black"/>
</svg>
```

Figure 20: Dibujando un círculo con SVG



Donde:

- **cx**: coordenada *x*, empuja el centro del círculo desde la izquierda de la página.
- **cy**: coordenada *y*, empuja el centro del círculo desde la parte superior de la página.
- **r**: radio del círculo.
- **fill**: rellena la figura con un color.
- **stroke**: añade un borde de color a la figura (atributo más personalizable en CSS).

Como podemos suponer, hay ciertos atributos únicos para cada figura posible para dibujar, otros son más generales, como *fill* o *stroke*.

Ahora crearemos un **rectángulo** en el siguiente ejemplo con su visualización (*Figura 21*):

```
<svg width="3000" height="3000">
    <!-- Dibuja un rectángulo de 300x100 píxeles, de color rojo y borde negro. -->
    <rect width="300" height="100" x="30" y="30" fill="red" stroke="black" />
</svg>
```

Figure 21: Dibujando un rectángulo con SVG



Donde:

- **width**: ancho del rectángulo.
- **height**: altura del rectángulo.

- **x**: coordenada *x*, igual que el atributo *cx*.
- **y**: coordenada *y*, igual que el atributo *cy*.

Una **línea** puede ser dibujada siguiendo el ejemplo a continuación, el resultado puede apreciarse en la *Figura 22*:

```
<svg width="3000" height="3000">
    <!-- Dibuja una línea negra con inicio en x(10,10) y y(200,100). -->
    <line x1="10" y1="10" x2="200" y2="100" stroke="black" />
</svg>
```

Figure 22: Dibujando una línea con SVG



Donde:

- **x1**: coordenada *x* para el punto inicial de la línea, empuja este punto desde la izquierda de la página.
- **y1**: coordenada *y* para el punto inicial de la línea, empuja este punto desde la parte superior de la página.
- **x2**: coordenada *x* para el punto terminal de la línea, empuja este punto desde la izquierda de la página.
- **y2**: coordenada *y* para el punto terminal de la línea, empuja este punto desde la parte superior de la página.

Una **polilínea** (una línea tras otra) puede ser dibujada siguiendo el ejemplo a continuación, el resultado puede apreciarse en la *Figura 23*:

```
<svg width="3000" height="3000">
    <!-- Dibuja un camino de líneas que forman dos triángulos uno al lado del otro con relleno negro. -->
    <polyline points="100 100, 150 150, 200 100, 250 150, 300 100" stroke="black" />
</svg>
```

Figure 23: Dibujando una ruta con SVG



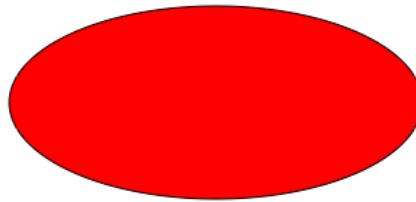
Donde:

- **points**: son las coordenadas  $x$ ,  $y$  de los puntos que seguirá la ruta, se separan únicamente por comas.
- **stroke**: en este caso, el atributo colorea por completo la ruta.

Una **elipse** puede ser dibujada siguiendo el ejemplo a continuación, el resultado puede apreciarse en la *Figura 24*:

```
<svg width="3000" height="3000">
    <!-- Dibuja una elipse roja con borde negro. -->
    <ellipse cx="200" cy="100" rx="150" ry="70" fill="red" stroke="black" />
</svg>
```

Figure 24: Dibujando una elipse con SVG



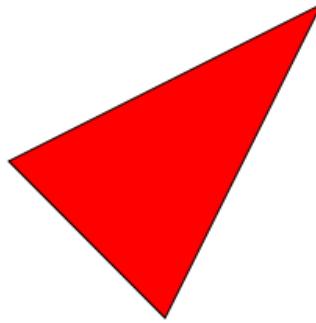
Donde:

- **cx**: coordenada  $x$ , aplicable igual al círculo.
- **cy**: coordenada  $y$ , aplicable igual al círculo.
- **rx**: tamaño del radio horizontal de la elipse.
- **ry**: tamaño del radio vertical de la elipse.

Un **polígono** puede ser dibujada siguiendo el ejemplo a continuación, el resultado puede apreciarse en la *Figura 25*:

```
<svg width="3000" height="3000">
    <!-- Dibuja un polígono rojo con borde negro. -->
    <polygon points="100 100, 200 200, 300 0" fill="red" stroke="black" />
</svg>
```

Figure 25: Dibujando un polígono con SVG



Donde:

- **points**: son las coordenadas  $x, y$  de los puntos que seguirá la ruta, se separan únicamente por comas.

*Nota:* recuerde que el atributo *stroke* puede ser modificado y personalizado a profundidad con CSS, al igual que el atributo *fill*.

### 2.7.2 Animaciones y rutas

Además de crear figuras, svg puede crear animaciones con la etiqueta **animate** y sus siguientes atributos:

- **attributeName**: especifica el atributo o variable que será afectada por la animación. Los valores que se le pueden dar a este atributo pueden ser: x, y, rx, ry, cx, cy, r, etc.
- **from**: valor inicial del atributo de la animación.
- **to**: valor final del atributo de la animación.
- **dur**: duración de la animación (3s, 4s, 10s).
- **fill**: indica qué ocurre con el valor del atributo o variable cuando la animación termina. Puede adoptar el valor "freeze" (mantiene el valor final que tuvo el atributo) o "remove" (resetea el valor del atributo).
- **repeatCount**: número de veces que se repetirá la animación. Puede utilizar el valor "indefinite" para que la animación se repita indefinidamente.

Pondremos un ejemplo donde un cuadrado baja infinitamente a continuación y puede ejecutarlo en su buscador:

```
<svg width="1000" height="1000">
  <rect width="150" height="150" fill="orange" stroke="black">
    <animate attributeName="y" from="0" to="300" dur="3s" fill="freeze" repeatCount="indefinite"/>
  </rect>
</svg>
```

Una **ruta** puede ser definida con la etiqueta **path**, su atributo **d** es la que establece cómo será la ruta; al igual que el dibujado de figuras en la sección anterior, se utilizan coordenadas *x*, *y*, sin embargo, el atributo **d** puede contener valores adicionales a las coordenadas:

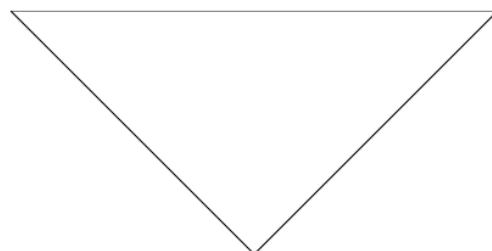
- **M**: move to (punto inicial de la ruta).
- **L**: line to (línea diagonal, horizontal o vertical).
- **H**: horizontal line to.
- **V**: vertical line to
- **C**: curve to.
- **S**: smooth curve to.
- **Q**: quadratic Bézier curve.
- **T**: smooth quadratic Bézier curve to.
- **A**: elliptical Arc.
- **Z**: close path (cierra la ruta).

Como puede ver, estos valores adicionales están en mayúsculas, esto indica que la ruta tendrá una **posición absoluta**, si utiliza letras minúsculas la **posición será relativa**.

Creamos una ruta triangular a continuación, puede ver el resultado en la *Figura 26*:

```
<svg width="3000" height="3000">
  <!-- Dibuja una ruta color negro. -->
  <path d="M 0 0 L200 200 L400 0 Z" stroke="black" fill="none" />
</svg>
```

Figure 26: Dibujando una ruta con SVG y path



Nuestra área de dibujo debe tener un punto de inicio para dibujar la ruta, *M 0 0* crea este punto de inicio, *L200 200* crea una línea entre las coordenadas (0, 0) y (200, 200), *L400 0* crea una línea entre las coordenadas (200, 200) y (400, 0), finalmente, *Z* dibuja una línea entre el último punto creado (400, 0) y el punto inicial (0, 0); de esta manera, podemos conseguir crear rutas en nuestros documentos HTML. Nótese que el atributo *fill* tiene el valor **none**, esto evita que la ruta se coloree completamente (cosa que ocurrió dibujando polilíneas).

## 2.8 Canvas

La etiqueta **canvas** es un contenedor para figuras o gráficos un poco más especializados o complejos, a diferencia de *svg*, es necesario un script (JavaScript suele ser utilizado) para dibujar dentro de este contenedor.

```
<canvas id="canvas1" width="200" height="200"></canvas>
```

El atributo *id* se hace presente, porque es necesario para pasárselo al script, que le regresará un gráfico al contenedor canvas:

```
<!-- Crea un panel de dibujo Canvas de 400x300 píxeles. -->
<canvas id="canvas1" width="400" height="300"></canvas>

<script>
    // Asigna a una variable el contenedor con id "canvas1".
    var can = document.getElementById("canvas1");
    // Establece el tipo de dibujo en 2D.
    var ctx = can.getContext("2d");
</script>
```

Al igual que en *svg*, las coordenadas que maneja canvas son *x*, *y*, cada una de ellas empujan el gráfico insertado igual que como ocurre en *svg* (de izquierda a derecha y de superior a inferior en la página). Al igual que con *svg*, existen métodos de JS para dibujar figuras y pasárselos al canvas.

El método **fillRect(x, y, w, h)** dibuja un cuadrado o rectángulo relleno (obligado) de color negro (por defecto), podemos suponer que los parámetros representan las coordenadas y dimensiones de la figura respectivamente. Pongamos un ejemplo y su resultado (*Figura 27*):

```
<canvas id="canvas1" width="900" height="900"></canvas>

<script>
    var c=document.getElementById("canvas1");
    var ctx=c.getContext("2d");
    // Dibuja un cuadrado con origen en (30,30) de 200x200 píxeles.
    ctx.fillRect(30,30,200,200);
</script>
```

Figure 27: Dibujando un cuadrado con Canvas



Existen algunos otros métodos para personalizar y crear gráficos:

- **fillStyle**: establece color, degradado o patrón de relleno para el gráfico.
- **moveTo(x,y)**: define el punto inicial para dibujar una línea.
- **lineTo(x,y)**: define el punto terminal para dibujar una línea.
- **beginPath()**: establece el modo para dibujar un círculo.
- **arc(x,y,r,comienzo,fin)**: establece los parámetros para dibujar un círculo.
- **stroke()**: dibuja un círculo.
- **createLinearGradient(x,y,x1,y2)**: crea un degradado lineal.
- **createRadialGradient(x,y,r,x1,y1,r1)**: crea un degradado circular.
- **font**: establece la letra, formato y tamaño del texto a dibujar.
- **fillText(texto,x,y)**: dibuja un texto rellenado.
- **strokeText(texto,x,y)**: dibuja un texto no rellenado.
- Entre otros.

### 2.8.1 Transformaciones Canvas

Un gráfico Canvas puede ser transformado una vez este fue creado, el método **translate(x,y)** mueve un gráfico a otra posición: **x** representa la distancia horizontal que se moverá el gráfico, **y** representa la distancia vertical que se moverá. La *Figura 28* muestra el resultado de un ejemplo de esta función:

```
<canvas id="canvas1" width="400" height="300"></canvas>

<script>
    var c=document.getElementById("canvas1");
    var ctx=c.getContext("2d");
    // Establece el estilo, tamaño y tipografía del texto.
    ctx.font="bold 22px Tahoma";
    // Establece la alineación del texto.
    ctx.textAlign="start";
    // Posición original.
    ctx.fillText("start", 10, 30);
    // Posición para transformación.
    ctx.translate(100, 150);
    ctx.fillText("after translate", 10, 30);
</script>
```

Figure 28: Trasladando una gráfico con Canvas

**start**

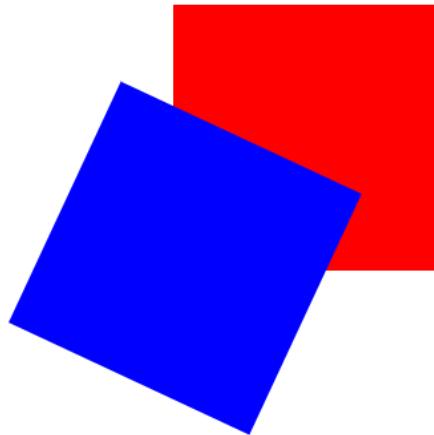
**after translate**

El método **rotate()** rota un gráfico. Los valores que acepte este método deben estar en **radianes**, no grados. La *Figura 29* muestra el resultado de un ejemplo de esta función:

```
<canvas id="canvas1" width="900" height="900"></canvas>

<script>
    var c=document.getElementById("canvas1");
    var ctx=c.getContext("2d");
    // Aspecto del cuadrado antes de la rotación.
    ctx.fillStyle = "#FF0000";
    ctx.fillRect(150,60,200,200);
    // Rotación de la figura.
    ctx.rotate((Math.PI / 180) * 25); // Rota 25 grados.
    // Aspecto del cuadrado después de la rotación.
    ctx.fillStyle = "#0000FF";
    ctx.fillRect(150,60,200,200);
</script>
```

Figure 29: Rotando una gráfico con Canvas



El método **scale(x, y)** escala un gráfico, el parámetro *x* indica la cantidad de veces que el gráfico será escalado en la dirección x, mientras que el parámetro *y* indica la cantidad de veces que el gráfico será escalado en la dirección y. La *Figura 30* muestra el resultado de un ejemplo de esta función:

```
<canvas id="canvas1" width="900" height="900"></canvas>

<script>
    var c=document.getElementById("canvas1");
    var ctx=c.getContext("2d");
    ctx.font="bold 22px Tahoma";
    ctx.textAlign="start";
    // Aspecto del texto en una ubicación.
    ctx.fillText("start", 10, 30);

    ctx.scale(1.5, 4); // Escala el texto 1.5 veces en el eje X y 4 veces en el eje Y.

    // Mantiene el texto, pero lo ubica en otro sitio.
    ctx.fillText("start", 10, 60);
</script>
```

Figure 30: Escalando un gráfico con Canvas



**start**

*Nota:* si escala una figura, las futuras figuras estarán escaladas también.

### 2.8.2 SVG vs Canvas

La *Tabla 1* compara las características de ambos métodos para dibujar figuras en documentos HTML:

Table 1: Comparación entre Canvas y SVG

Canvas	SVG
<ol style="list-style-type: none"> <li>1. Las figuras se dibujan por programación.</li> <li>2. El dibujo es por medio de píxeles.</li> <li>3. No maneja animaciones.</li> <li>4. Excelente rendimiento para figuras dibujadas con píxeles.</li> <li>5. Dependiente de la resolución.</li> <li>6. Sin manejadores de eventos.</li> <li>7. Se pueden guardar en formatos .png o .jpg.</li> <li>8. Muy adecuado para juegos con gráficos intensivos.</li> </ol>	<ol style="list-style-type: none"> <li>1. Forman parte del DOM del sitio web.</li> <li>2. El dibujo es por medio de vectores.</li> <li>3. Maneja animaciones.</li> <li>4. Basado en XML estándar.</li> <li>5. Independiente de la resolución.</li> <li>6. Con manejadores de eventos.</li> <li>7. No muy adecuado para juegos.</li> <li>8. Muy adecuado para aplicaciones con grandes áreas de renderizado (por ejemplo, Google Maps).</li> </ol>

Podemos utilizar svg y canvas en un mismo sitio o página, pero no podemos dibujar figuras canvas dentro de un svg ni viceversa.

## 2.9 Formularios HTML5

La *Tabla 2* contiene las etiquetas y atributos incorporados a esta versión de HTML, junto con ejemplos:

Table 2: Atributos y etiquetas nuevas de formularios en HTML5

Atributo Etiqueta	Definición	Ejemplo
autofocus	Atributo que da el foco a un control cuando el sitio o página carga	<input type="text" autofocus />
required	Atributo que vuelve a un control necesario para enviar el formulario	<input type="text" required />
autocomplete	Atributo que permite que un control o formulario pueda ser llenado con valores que el usuario ha ingresado previamente	<input type="text" autocomplete="off" />
list	Atributo que puede convertir una caja de texto regular en un menú desplegable (con la posibilidad de escribir todavía)	<input type="text" id="cajita" list="colores" />
search	Crea una caja de búsqueda	<input type="search" id="buscador" name="buscaitem" />
datalist	Control complemento para <b>search</b> que define una lista de pre-valores para la caja de búsqueda. El ID de la etiqueta <b>datalist</b> debe ser igual al valor del atributo <b>list</b> de un control para que las opciones del primero aparezcan en el segundo	<datalist id="buscador"></datalist>
options	Sub-control complemento para <b>select</b> o <b>datalist</b> que define una lista de opciones para el menú desplegable	<option value="1">
email	Crea una caja para un correo	<input type="email" placeholder="example@gmail.com" />
url	Crea una caja para una URL	<input type="url" placeholder="google.com" />
tel	Crea una caja para un número de teléfono	<input type="tel" placeholder="123 456 7890" />

Algunos otras **etiquetas** y **atributos** que no fueron detallados son:

- **color**.

- **date.**
- **datetime.**
- **datetime-local.**
- **month.**
- **number.**
- **range.**
- **time.**
- *formaction.*
- *formenctype.*
- *formmethod.*
- *formnovalidate.*
- *formtarget.*
- *height & width.*
- *list.*
- *min & max.*
- *multiple.*
- *patter (regexp).*
- *step.*