

Ejercitación 1:

- 1) **HTML** (HyperText Markup Language) es un **lenguaje de marcado** utilizado para la elaboración de páginas web. Sin embargo, HTML fue diseñado principalmente como un lenguaje para describir semánticamente documentos científicos.

El mismo fue creado en 1990 y fue evolucionando de la siguiente manera:

HTML 2.0	En 1995 se publica el estándar HTML 2.0. A pesar de su nombre, HTML 2.0 es el primer estándar oficial de HTML, es decir, el HTML 1.0 no existió como estándar. HTML 2.0 no soportaba tablas.
HTML 3.2	La versión HTML 3.2 se publicó en 1997 y es la primera recomendación de HTML publicada por el W3C (World Wide Web Consortium). Esta revisión incorporó los últimos avances de las páginas web desarrolladas hasta 1996, como applets de Java y texto que fluye alrededor de las imágenes.
HTML 4.01	La última especificación oficial de HTML se publicó en diciembre de 1999 y se denomina HTML 4.01. Desde la publicación de HTML 4.01, el W3C se centró en el desarrollo del estándar XHTML. Por este motivo, en el año 2004, las empresas Apple, Mozilla y Opera mostraron su preocupación por la falta de interés del W3C en HTML y decidieron organizarse en una nueva asociación llamada WHATWG (Web Hypertext Application Technology Working Group) que comenzó el desarrollo del HTML 5, cuyo primer borrador oficial se publicó en enero de 2008. Debido a la fuerza de las empresas que forman el grupo WHATWG y a la publicación de los borradores de HTML 5.0, en marzo de 2007 el W3C decidió retomar la actividad estandarizadora de HTML, dentro del cual decidió integrar el XHTML.
HTML 5, HTML 5.1, HTML 5.2	El consorcio internacional W3C, después de una evolución de varios años, liberó el HTML5 como estándar oficial a finales de octubre de 2014. HTML5 incorpora nuevos elementos no contemplados en HTML 4.01. Hay diversos cambios respecto a HTML 4.01. Hay nuevas etiquetas, se introduce la posibilidad de introducir audio y video de forma directa en la web sin necesidad de plugins o complementos en los navegadores, y otras novedades.

La última versión de HTML a la fecha es la 5.2. Desde 2017 se está desarrollando la versión 5.3 sin fecha de liberación pactada.

2)

- a) **Separar estructura y presentación:** El HTML tiene sus raíces en SGML, que siempre ha sido un lenguaje para la especificación de código estructural. A medida que el HTML madura, un número cada vez mayor de sus elementos y atributos presentacionales ha sido reemplazado por otros mecanismos, en particular las hojas de estilo. La experiencia ha demostrado que separando la estructura de la presentación se reduce el coste de servir a un amplio espectro de plataformas, medios, etc. y se facilitan las revisiones del documento.
- b) **Considerar la accesibilidad universal a la Web:** Para hacer la Web más accesible a todos, en especial a aquéllos con discapacidades, los autores deberían considerar cómo pueden representarse sus documentos en diferentes plataformas: navegadores basados en voz, lectores braille, etc. No estamos recomendando a los autores que limiten su creatividad, sólo que consideren representaciones alternativas de sus diseños. El HTML ofrece un número de mecanismos con este fin (p.ej., el atributo alt, el atributo accesskey, etc.) Además de esto, los autores deberían recordar que sus documentos pueden llegar a una audiencia muy lejana con diferentes computadoras y configuraciones. Para que los documentos sean correctamente interpretados, los autores deberían incluir en sus documentos información sobre el idioma natural y la dirección del texto, cómo está codificado el documento, y otras cuestiones relacionadas con la internacionalización.
- c) **Ayudar con la representación incremental:** Mediante un diseño cuidadoso de las tablas y haciendo uso de las nuevas características de representación incremental, los autores pueden ayudar a los agentes de usuario a representar los documentos más rápidamente.

- 3) **Desaprobado:** Un elemento o atributo desaprobado es aquel que ha quedado anticuado por la presencia de estructuras nuevas. Los elementos desaprobados pueden declararse obsoletos en el futuro. Los agentes de usuario deberían seguir dando soporte a los elementos desaprobados por razones de compatibilidad.

Obsoleto: Un elemento o atributo obsoleto es aquél para el cual no hay garantía de soporte por parte de un agente de usuario. Los elementos obsoletos han dejado de estar definidos en la especificación, pero se enumeran por motivos históricos en la sección de cambios del manual.

- 4) Una **DTD** (Declaración del tipo de documento) es un documento que define la estructura de un documento.

En HTML 4.01 los posibles DTDs contemplados son:

- a) **EI DTD HTML 4.01 Estricto** (Strict DTD): Incluye todos los elementos y atributos que no han sido desaprobados o que no aparecen en documentos con marcos.
- b) **EI DTD HTML 4.01 Transicional** (Transitional DTD): Incluye todo lo que incluye el DTD estricto más los elementos y atributos desaprobados (la mayoría de los cuales están relacionados con la presentación visual).
- c) **EI DTD HTML 4.01 para Documentos con Marcos** (Frameset DTD): Incluye todo lo que incluye el DTD Transicional más los marcos.

- 5) **Metadatos:** Información sobre un documento más que contenido del propio documento.

Por ejemplo:

- a) **Para especificar el autor:** El elemento META puede utilizarse para identificar propiedades de un documento (p.ej., el autor, la fecha de caducidad, una lista de palabras clave, etc.) y para asignar valores a esas propiedades.
- b) Puede utilizarse el atributo **lang** de META para especificar el idioma del valor del atributo content. Esto permite a los sintetizadores de voz aplicar reglas de pronunciación dependientes del idioma. Ejemplo: declara que el nombre del autor está en francés.
- c) Para especificar la **codificación de caracteres**.
- d) Un uso común de META es especificar **palabras clave** que pueden usar los motores de búsqueda para mejorar la calidad de los resultados de una búsqueda. Cuando se proporcionan varios elementos META con información para varios idiomas, los motores de búsqueda pueden utilizar el atributo lang como filtro para mostrar los resultados de la búsqueda usando las preferencias de idioma del usuario:

Ejercitación 2:

A. *<!-- Código controlado el día 12/08/2009 -->*

- a. Sección: Cualquier sección.
- b. Efecto: Comentario
- c. Etiquetas:
 - i. *<!--* Apertura
 - ii. *-->* Cierre
- d. Atributos: - No posee. -

B. *<div id="bloque1">Contenido del bloque1</div>*

- a. Sección: Solo BODY.
- b. Efecto: Crear divisiones o secciones
- c. Etiquetas:
 - i. *<div>* Apertura
 - ii. *</div>* Cierre
- d. Atributos:
 - i. id: texto. No obligatorio.

C. **

- a. Sección: Solo BODY.
- b. Efecto: Crear un espacio para la imagen referenciada en *src*.
- c. Etiquetas:
 - i. **
- d. Atributos:
 - i. src: URL. Obligatorio.
 - ii. alt: texto. Obligatorio
 - iii. id: texto. No obligatorio.
 - iv. name: texto. No obligatorio
 - v. width: pixeles. No obligatorio
 - vi. height: pixeles. No obligatorio
 - vii. longdesc: URL. No obligatorio

D. **`<meta name="keywords" lang="es" content="casa, compra, venta, alquiler " />`**
`<meta http-equiv="expires" content="16-Sep-2019 7:49 PM" />`

- a. Sección: Solo HEAD.
- b. Efecto: Definir información acerca de los datos.
- c. Etiquetas:
 - i. **`<meta>`**
- d. Atributos:
 - i. name: texto. No obligatorio.
 - ii. lang: Código ISO de lenguaje. No obligatorio
 - iii. content: texto. Obligatorio si está name o http-equiv.
 - iv. http-equiv: texto. No obligatorio.

E. **`<a href="http://www.e-style.com.ar/resumen.html" type="text/html"`**
`hreflang="es" charset="utf-8" rel="help">Resumen HTML `

- a. Sección: Solo BODY.
- b. Efecto: Definir un hipervínculo.
- c. Etiquetas:
 - i. **`<a>`** Apertura
 - ii. **``** Cierre
- d. Atributos:
 - i. href: *URL*. Obligatorio.
 - ii. type: Tipo de medio. No obligatorio
 - iii. hreflang: Código ISO de lenguaje. No obligatorio.
 - iv. charset: Codificación de caracteres. No obligatorio. **Obsoleto en HTML5.**
 - v. rel: Tipo de link definido. No obligatorio.
(Ejemplos de links definidos
https://developer.mozilla.org/en-US/docs/Web/HTML/Link_types)

F. `<table width="200" summary="Datos correspondientes al ejercicio vencido">`
`<caption align="top"> Título </caption>`
`<tr>`
`<th scope="col"> </th>`
`<th scope="col">A</th>`
`<th scope="col">B</th>`
`<th scope="col">C</th>`
`</tr>`
`<tr>`
`<th scope="row">1°</th>`
`<td> </td>`
`<td> </td>`
`<td> </td>`
`</tr>`
`<tr>`
`<th scope="row">2°</th>`
`<td> </td>`
`<td> </td>`
`<td> </td>`
`</tr>`
`</table>`

- a. Sección: Solo BODY.
- b. Efecto: Definir una tabla.
- c. Etiquetas:
 - i. `<table>` Apertura
 - ii. `<caption>` Apertura titulo descriptivo.
 - iii. `</caption>` Cierre titulo descriptivo.
 1. `<tr>` Apertura row
 - a. `<th>` Apertura header
 - b. `</th>` Cierre header
 - c. `<td>` Apertura data
 - d. `</td>` Cierre data
 2. `</tr>` Cierre row
 - iv. `</table>` Cierre
- d. Atributos:
 - i. table:
 1. width: pixeles. No obligatorio. **Obsoleto en HTML5.**
 2. summary: texto. No obligatorio. **Obsoleto en HTML5.**
 - ii. caption:
 1. align: top, bottom, left o right: No obligatorio
 - iii. th:
 1. scope: col, colgroup, row, o rowgroup. No obligatorio.

Ejercitación 3:

A.

- a. `Click aquí para ir a Google`
Abre google en la misma pestaña.
- b. `Click aquí para ir a Google`
Abre google en una nueva pestaña
- c. ``
No muestra dónde hacer el click y tiene un error de tipeo en href.
- d. `Click aquí para ir a Google`
Lleva a la misma página.
- e. `Click aquí para volver arriba`
``
Lleva a una sección de la misma página con name="arriba".

B.

- a. `<p>Click aquí</p>`
Si se clickea en *Click Aquí* lleva a google.
- b. `<p> Click aquí</p>`
Si se clickea en *La imagen* lleva a google.
- c. `<p>Click aquí</p>`
Si se clickea tanto en *Click Aquí* como en *La imagen* lleva a google.
- d. `<p> Click aquí</p>`
Si se clickea tanto en *Click Aquí* como en *La imagen* lleva a google. Pero es ineficiente la forma en la que está escrito.

C.

a. ``
`xxx`
`yyy`
`zzz`
``

Genera una lista no ordenada con los ítems xxx, yyy, zzz de esta forma:

- xxx
- yyy
- zzz

b. ``
`xxx`
`yyy`
`zzz`
``

Genera una lista ordenada con los ítems xxx, yyy, zzz de esta forma:

1. xxx
2. yyy
3. zzz

c. ``
`xxx`
``
``
`<li value="2">yyy`
``
``
`<li value="3">zzz`
``

Genera tres listas ordenadas con un ítem cada una, xxx, yyy y zzz respectivamente. Modificando el valor del índice de la lista mediante el atributo value.

d. `<blockquote>`
`<p>1. xxx
 2. yyy
 3. zzz</p>`
`</blockquote>`

Genera una imitación de una lista pero mediante la utilización de bloques, párrafos y saltos de línea.

D.

a.

```
<table border="1" width="300">
  <tr>
    <th>Columna 1</th>
    <th>Columna 2</th>
  </tr>
  <tr>
    <td>Celda 1</td>
    <td>Celda 2</td>
  </tr>
  <tr>
    <td>Celda 3</td>
    <td>Celda 4</td>
  </tr>
</table>
```

Genera una tabla que tiene por encabezado Columna 1 y Columna 2, y tiene dos filas con los contenidos Celda 1, Celda 2 y Celda 3, Celda 4 respectivamente.

b.

```
<table border="1" width="300">
  <tr>
    <td>
      <div align="center">
        <strong>Columna1</strong>
      </div>
    </td>
    <td>
      <div align="center">
        <strong>Columna 2</strong>
      </div>
    </td>
  </tr>
  <tr>
    <td>Celda 1</td>
    <td>Celda 2</td>
  </tr>
  <tr>
    <td>Celda 3</td>
    <td>Celda 4</td>
  </tr>
</table>
```

Genera una tabla que tiene por encabezado Columna 1 y Columna 2, con la particularidad de que simula este encabezado mediante la utilización de td, div y letra en negrita. Y tiene dos filas con los contenidos Celda 1, Celda 2 y Celda 3, Celda 4 respectivamente.

A pesar de los cambios en la escritura, ambos ejemplos poseen la misma apariencia:

Columna 1	Columna 2
Celda 1	Celda 2
Celda 3	Celda 4

E.

```
a. <table width="200">
    <caption>Título</caption>
    <tr>
        <td bgcolor="#dddddd">&nbsp;</td>
        <td bgcolor="#dddddd">&nbsp;</td>
        <td bgcolor="#dddddd">&nbsp;</td>
    </tr>
    <tr>
        <td bgcolor="#dddddd">&nbsp;</td>
        <td bgcolor="#dddddd">&nbsp;</td>
        <td bgcolor="#dddddd">&nbsp;</td>
    </tr>
</table>
```

Genera una tabla de tres filas y tres columnas. El tag *caption* se adapta de manera automática a la cantidad de columnas de la tabla creando de esta manera una fila de una sola columna.

b. `<table width="200">`
 `<tr>`
 `<td colspan="3">`
 `<div align="center">Título</div>`
 `</td>`
 `</tr>`
 `<tr>`
 `<td bgcolor="#dddddd"> </td>`
 `<td bgcolor="#dddddd"> </td>`
 `<td bgcolor="#dddddd"> </td>`
 `</tr>`
 `<tr>`
 `<td bgcolor="#dddddd"> </td>`
 `<td bgcolor="#dddddd"> </td>`
 `<td bgcolor="#dddddd"> </td>`
 `</tr>`
`</table>`

Genera una tabla de tres filas y tres columnas. Dado que no utiliza caption, al querer insertar un título debe crearse una row y un data con un atributo que señale la cantidad de columnas que posee la tabla, con el fin de extender la longitud de esa celda hasta la que generan la cantidad de columnas declaradas.

Ambos poseen la misma apariencia:

Título		

F.

a. `<table width="200">`
 `<tr>`
 `<td colspan="3">`
 `<div align="center">Título</div>`
 `</td>`
 `</tr>`
 `<tr>`
 `<td rowspan="2" bgcolor="#dddddd"> </td>`
 `<td bgcolor="#dddddd"> </td>`
 `<td bgcolor="#dddddd"> </td>`
 `</tr>`
 `<tr>`
 `<td bgcolor="#dddddd"> </td>`
 `<td bgcolor="#dddddd"> </td>`
 `</tr>`
`</table>`

Genera una tabla con un encabezado, dos filas y tres columnas, con la particularidad que la primer celda posee un rowspan (dicha celda ocupa por altura la generada por la cantidad de rows que está como valor de dicho atributo) plasmando esta apariencia:

Título


```

b. <table width="200">
    <tr>
        <td colspan="3">
            <div align="center">Título</div>
        </td>
    </tr>
    <tr>
        <td colspan="2" bgcolor="#dddddd">&nbsp;</td>
        <td bgcolor="#dddddd">&nbsp;</td>
    </tr>
    <tr>
        <td bgcolor="#dddddd">&nbsp;</td>
        <td bgcolor="#dddddd">&nbsp;</td>
        <td bgcolor="#dddddd">&nbsp;</td>
    </tr>
</table>

```

Genera una tabla con un encabezado, dos filas y tres columnas, con la particularidad que la primer celda posee un colspan(dicha celda ocupa por longitud la generada por la cantidad de columnas que está como valor de dicho atributo) plasmando esta apariencia:

Título		

G.

a.

```
<table width="200" border="1">
  <tr>
    <td colspan="3">
      <div align="center">Título</div>
    </td>
  </tr>
  <tr>
    <td colspan="2" rowspan="2">&nbsp;</td>
    <td>&nbsp;</td>
  </tr>
  <tr>
    <td width="50%">&nbsp;</td>
  </tr>
</table>
```

Genera una tabla con un encabezado, dos filas y dos columnas, con la particularidad que la primer celda posee un colspan y un rowspan de valor 2, generando un cuadrado y plasmando esta apariencia:

Título	

b.

```
<table width="200" border="1" cellpadding="0" cellspacing="0">
  <tr>
    <td colspan="2">
      <div align="center">Título</div>
    </td>
  </tr>
  <tr>
    <td rowspan="2">&nbsp;</td>
    <td>&nbsp;</td>
  </tr>
  <tr>
    <td width="50%">&nbsp;</td>
  </tr>
</table>
```

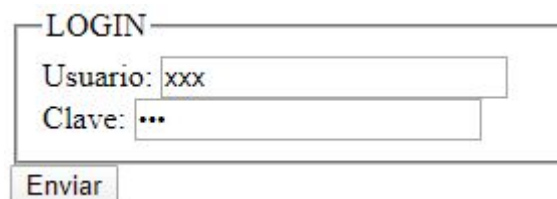
Genera una tabla con un encabezado, dos filas y dos columnas, eliminando los espacios entre celdas, con la particularidad que la primer celda posee un colspan y un rowspan de valor 2, generando un cuadrado y plasmando esta apariencia:

Título	

H.

- a. `<form id="form1" name="form1" action="procesar.php" method="post" target="_blank">`
 `<fieldset>`
 `<legend>LOGIN</legend>`
 `Usuario: <input type="text" id="usu1"`
 `name="usu1" value="xxx" />
`
 `Clave: <input type="password" id="clave1"`
 `name="clave1" value="xxx" />`
 `</fieldset>`
 `<input type="submit" id="boton1" name="boton1"`
 `value="Enviar" />`
`</form>`

Genera un formulario con un grupo de campos, declarando título, el campo usuario y el campo clave, los cuales tienen valores iniciales. Cuando se presiona el boton de envio hace una request POST y lleva a *procesar.php*. Muestra esta apariencia:



- b. `<form id="form2" name="form2" action="" method="get" target="_blank"> LOGIN
`
 `<label>Usuario: <input type="text" id="usu2"`
 `name="usu2" /></label>
`
 `<label>Clave: <input type="text" id="clave2"`
 `name="clave2" /></label>
`
 `<input type="submit" id="boton2" name="boton2"`
 `value="Enviar" />`
`</form>`

Genera un formulario con dos campos, usuario y clave, los cuales no poseen valores iniciales. Cuando se presiona el boton de envio hace una request GET pero no envia la informacion a ninguna parte pues action se encuentra vacío. Muestra esta apariencia:

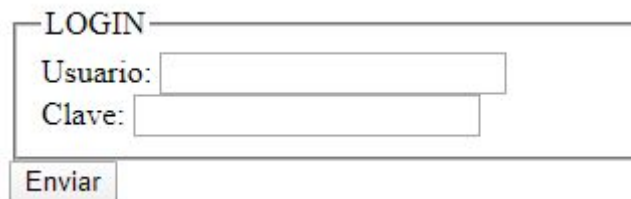


- c. `<form id="form3" name="form3" action="mailto:xx@xx.com" enctype=text/plain method="post" target="_blank">`
`<fieldset>`
`<legend>LOGIN</legend>`
`Usuario: <input type="text" id="usu3" name="usu3" />
`
`Clave: <input type="password" id="clave3" name="clave3" />`
`</fieldset>`
`<input type="reset" id="boton3" name="boton3" value="Enviar" />`
`</form>`

Genera un formulario con un grupo de campos, declarando título, el campo usuario y el campo clave, los cuales no tienen valores iniciales.

El type del botón es reset por lo que no envía nada, sino que vuelve los valores de los campos del formulario a los iniciales.

Si tuviese valor submit, el mismo abre el gestor de correos llenando como destinatario el mail especificado en *mailto* e ingresando como contenido el resultante del formulario. Muestra esta apariencia:



I.

- a. `<label>Botón 1`
`<button type="button" name="boton1" id="boton1">`
`
`
`CLICK AQUÍ`
`</button>`
`</label>`
- b. `<label>Botón 2`
`<input type="button" name="boton2" id="boton2" value="CLICK AQUÍ" />`
`</label>`

Genera un botón con una imagen y el texto en negrita.



Genera un botón mediante un atributo de input. La diferencia es que input no admite contenido (Única etiqueta), por lo que no se le podría agregar una imagen o estilo al texto. Genera la siguiente apariencia:



J.

a. `<p>`

```
<label>
```

```
<input type="radio" name="opcion" id="X" value="X" />X
```

```
</label>
```

```
<br />
```

```
<label>
```

```
<input type="radio" name="opcion" id="Y" value="Y" />Y
```

```
</label>
```

```
</p>
```

Genera una lista de opciones de la que solo puede seleccionarse una opción, puesto que el atributo name posee el mismo valor para todas las opciones.

b. `<p>`

```
<label>
```

```
<input type="radio" name="opcion1" id="X" value="X" />X
```

```
</label>
```

```
<br />
```

```
<label>
```

```
<input type="radio" name="opcion2" id="Y" value="Y" />Y
```

```
</label>
```

```
</p>
```

Genera una lista de opciones de la que pueden seleccionarse múltiples opciones, puesto que el atributo name posee distintos valores para cada opción.

Ambos poseen la misma apariencia:

☐ X

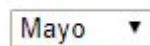
☐ Y

K.

a. `<select name="lista">`
 `<optgroup label="Caso 1">`
 `<option>Mayo</option>`
 `<option>Junio</option>`
 `</optgroup>`
 `<optgroup label="Caso 2">`
 `<option>Mayo</option>`
 `<option>Junio</option>`
 `</optgroup>`
`</select>`

Genera un desplegable con diversas opciones, de las cuales solo se puede elegir una.

Muestra la siguiente apariencia:



b. `<select name="lista[]" multiple="multiple">`
 `<optgroup label=" Caso 1">`
 `<option>Mayo</option>`
 `<option>Junio</option>`
 `</optgroup>`
 `<optgroup label=" Caso 2">`
 `<option>Mayo</option>`
 `<option>Junio</option>`
 `</optgroup>`
`</select>`

Genera un desplegable con diversas opciones, de las cuales se pueden elegir múltiples opciones.

Muestra la siguiente apariencia:

