

INTRODUCCIÓN AL DISEÑO WEB USANDO EL LENGUAJE

HTML



Pág.

Estructura de contenidos

Introducción	4
Mapa de contenido	6
Desarrollo de contenidos	7
1. El lenguaje HTML	7
1.1 Estructura de un browser o navegador	7
1.2 Estructura de un archivo html	8
1.2.1 Etiquetas básicas	11
1.2.2 Etiquetas para definir bloques o contenedores	12
1.2.3 Etiquetas para elaborar tablas	13
1.2.4 Etiquetas para el manejo de formularios	14
1.2.5 Etiqueta para el manejo de imágenes	15
1.3. Contenido estático vs contenido dinámico	16
1.4. Fases en el diseño de un sitio web	17
1.5. Herramientas para el diseño y desarrollo web	18
1.6 Maqueteado (layouts) en html	19
1.7 Novedades en la versión 5 de html	20
1.7.1. El elemento <canvas>	20
1.7.2. Video	20
1.7.3. Almacenamiento local	20
1.7.4. Web workers	20
1.7.5. Aplicaciones web fuera de línea	21
1.7.6. Geolocalización	21
1.7.7. Nuevos tipos de cajones para formularios	21
1.7.8. Nuevas etiquetas introducidas en la versión 5 del lenguaje html	22
2. Estilos en cascada (“cascade-style sheet” o css)	24
2.1. Reglas o comandos css	24
2.1.1 Selectores	25
2.1.2. Métodos para incorporar estilos a las páginas html	27
2.2. Conceptos básicos de css	30
2.2.1. Herencia	30
2.2.2. Modelo de la caja (box model)	31
2.2.3 Formas de aplicación de los estilos	32
2.3 Librerías y frameworks css	32
2.4. Ejemplo de maquetación con html y css	33
2.5 Formularios con html y css	41
3. El lenguaje javascript	44
3.1 Definición de un programa javascript “inline”	44
3.2 Definición de un programa javascript en una sección de la página web	46
3.3 Definición de un programa javascript en un archivo separado	46

3.4 Introducción a la programación en javascript	47
3.4.1. Implementación de un modelo de eventos	47
3.4.2. Manipulación del dom	48
3.4.3. Ajax	50
3.5 Ejemplo de utilización de html, css y javascript	50
4. Librerías y frameworks javascript	52
4.1. Librería jquery	52
4.2 Framework angularjs	53
5. Aspectos de seguridad en páginas web	54
Glosario	55
Bibliografía	56
Control del documento	57

INTRODUCCIÓN AL DISEÑO WEB HTML

Introducción



El desarrollo de aplicaciones para la web es una tendencia importante en la actualidad y es fundamental que los analistas y desarrolladores se apropien de las tecnologías que hacen posible este tipo de aplicaciones.

En primer lugar se deben apropiar los tres conceptos fundamentales o trilogía del diseño web: HTML, CSS y Javascript.

El lenguaje de marcación de hipertextos (HyperText Markup Language) es la primera pieza de esta trilogía y permite definir la estructura de los contenidos que aparecen en una página web. Su concepción se deriva de una tecnología llamada SGML (Standard Generalized Markup Language) el cual es un lenguaje de marcación que aplicaba a cualquier tipo de documento.

Para describir la apariencia que tendrán las páginas web se propusieron las hojas de estilos en cascada o (CSS). Este mecanismo, la segunda pieza de la trilogía, permite dar una personalización o estilo a los elementos de las páginas web como son textos, imágenes, tablas, encabezados, pies de página entre otros.

El último elemento es el lenguaje Javascript que añade interactividad a las páginas web. Tanto HTML como CSS son lenguajes declarativos, es decir, solamente indican qué debe hacerse sin indicar los detalles. Por otra parte está Javascript que es un lenguaje imperativo que indica cómo debe hacerse una tarea paso a paso.

Es con la ayuda de Javascript que las páginas web se pueden realizar tareas más complejas como validar los datos de entrada, conectarse a bases de datos, emitir mensajes de error, consultar la versión del navegador, entre otras. Es importante anotar que Javascript no tiene ninguna relación con el lenguaje de programación Java, son dos cosas distintas que tienen un nombre similar por mera casualidad.

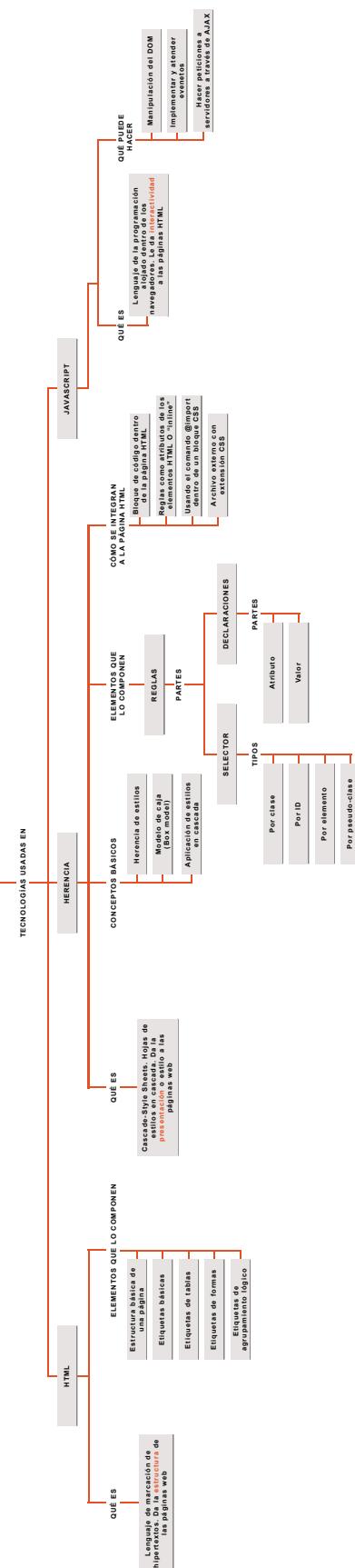
Por último han surgido componentes de software estructurados en los llamados “Frameworks” que facilitan el desarrollo de aplicaciones mediante la integración de las tecnologías descritas anteriormente. Ejemplos de frameworks Javascript son: AngularJS desarrollado por Google, ReactJS desarrollado por Facebook. Un ejemplo de un framework CSS es Bootstrap desarrollado por Twitter.

También están disponibles componentes estructurados en librerías. Uno de los más famosos y utilizados es Jquery que es una librería javascript que permite modificar los elementos HTML de manera dinámica.

En este recurso se hará una introducción a las tecnologías HTML, CSS y Javascript. También se hará una introducción a la librería Jquery y se mostrarán las novedades introducidas en la versión 5 de HTML. Es importante anotar que el conocimiento completo de las tres tecnologías requiere un espacio y dedicación más amplio. Por tanto una vez apropiado este recurso el aprendiz podrá profundizar estas temáticas mediante la consulta de la bibliografía anexa o mediante investigación en otros recursos o por Internet.

Las tecnologías que hacen posible las aplicaciones web están en constante evolución. Por lo anterior es importante que el desarrollador se esté constantemente actualizando en las distintas tendencias y conceptos.

Mapa de contenido



Desarrollo de contenidos

1. El lenguaje HTML

El HTML es el lenguaje de marcación principal para la creación de documentos en la WWW o Red Mundial (W3C, 2016).

El lenguaje de marcación para hipertextos o HTML utiliza un conjunto de etiquetas o “tags” para describir la estructura que tendrán las páginas web. Además del HTML existen otros lenguajes de marcación entre ellos el XML o eXtensible Markup Language.

Los archivos .html son planos, es decir, pueden ser leídos por un editor de texto.

Una vez creado un archivo html válido este puede ser interpretado por un analizador sintáctico o “parser” que está incluido en los browsers o navegadores.

1.1 Estructura de un browser o navegador.

Los navegadores modernos interpretan el código escrito en HTML y CSS y lo transforman en órdenes que son transmitidas a los distintos recursos que maneja un computador como son su interfaz gráfica, su componente de redes o su capa de almacenamiento permanente (NIEDERST, 2012).

En la figura 1.2 se aprecia una arquitectura de un navegador moderno .

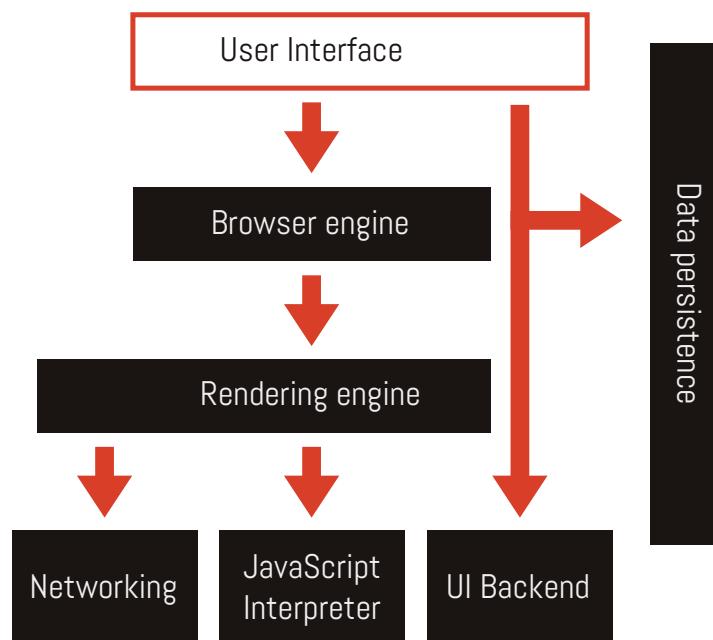


Figura 1.2. Arquitectura de un navegador.

- a) **Interfaz de usuario (user interface)**: incluye la barra de direcciones, los botones de “adelante” y “atrás”, el menú de bookmarks. En resumen todo lo que se muestra en el navegador excepto donde se dibujan las páginas.
- b) **El motor del browser (browser engine)**: coordina todos los llamados de la interfaz de usuario y el motor de dibujo (rendering engine).
- c) **El motor de dibujo**: responsable por dibujar la página solicitada. Si la página solicitada está en HTML , el motor de dibujo analiza el código HTML y CSS. Luego dibuja el contenido analizado en la pantalla.
- d) **Redes (Networking)**: para hacer solicitudes de redes tales como solicitudes HTTP.
- e) **Soporte de Interfaz de Usuario (UI Backend)**: usada para dibujar figuras geométricas básicas como combo boxes y ventanas. Este soporte es independiente de la plataforma (Windows , Linux , Mac). Por debajo se usan los métodos propios de la plataforma usada.
- f) **Intérprete de Javascript**: usado para procesar y ejecutar código en lenguaje Javascript.
- g) **Almacenamiento de datos (Data Persistence)**: es la capa de persistencia. El browser puede necesitar almacenar muchos tipos de datos tales como las cookies. Los browsers también soportan otros mecanismos de almacenamiento tales como localStorage, IndexedDB, WebSQL y filesystem.

1.2 Estructura de un archivo HTML.

Un documento HTML está compuesto por un árbol de elementos y texto. Cada elemento se describe a través de una etiqueta o tag de inicio y otra de terminación. (W3C, 2016).

Las partes que componen una etiqueta son (NIEDERST, 2012):

- a. **Etiqueta de apertura**: se componen por el carácter “<” y un comando. Por ejemplo “<html>”, “<body>”.
- b. **Atributos**: nombre y valor. Modifican el comando. Ejemplo: “<html lang=es>”
- c. **Contenido**: pueden ser párrafos, palabras, oraciones. Ejemplo: <h1> hola mundo </h1>
- d. **Etiqueta de cierre**: se compone por el carácter “</” más el nombre de comando y los caracteres “>”. Ejemplo: </html>

La figura 1.3a muestra la anatomía de una etiqueta HTML estándar.

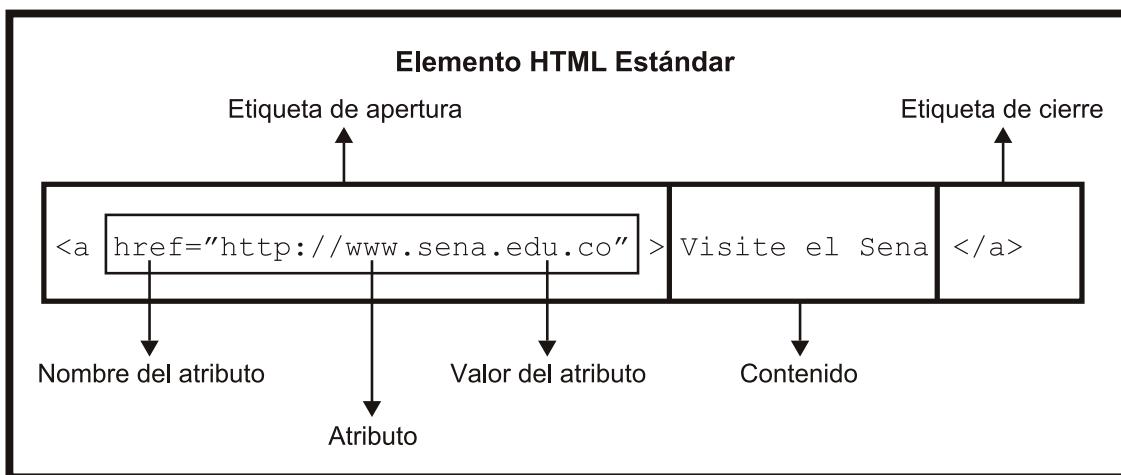


Figura 1.3a. Anatomía de una etiqueta en HTML estándar.

Algunos tags del lenguaje HTML no requieren tag final y se denominan etiqueta vacias.

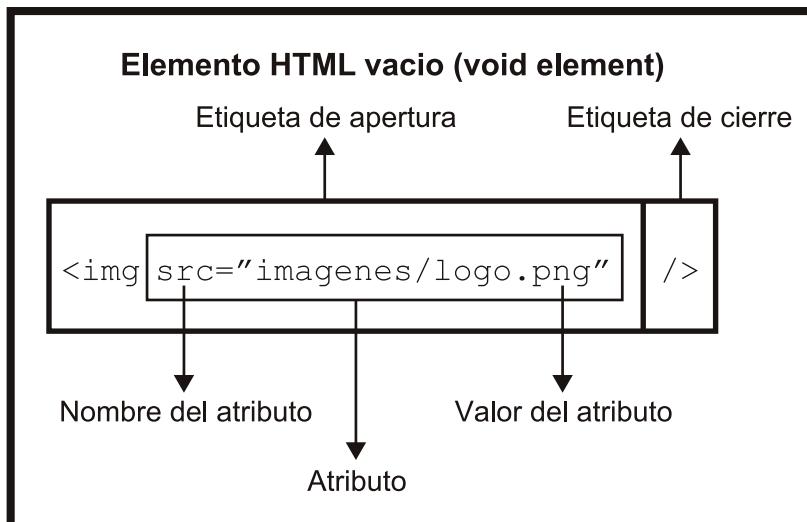


Figura 1.3b. Anatomía de una etiqueta en HTML vacía.

Los tag vacios son: area, base, br, col, embed, hr, img, input, keygen, link, menuitem, meta, param, source, track y wbr.

A continuación un ejemplo sencillo de una página HTML básica.

```

1 <html>
2   <body>
3     <b> ¡¡hola mundo!! 
4   </body>
5 </html>

```

Figura 1.4. Ejemplo de código HTML.

Al guardar este código como un archivo con extensión html se obtiene lo siguiente cuando se abre con un navegador:

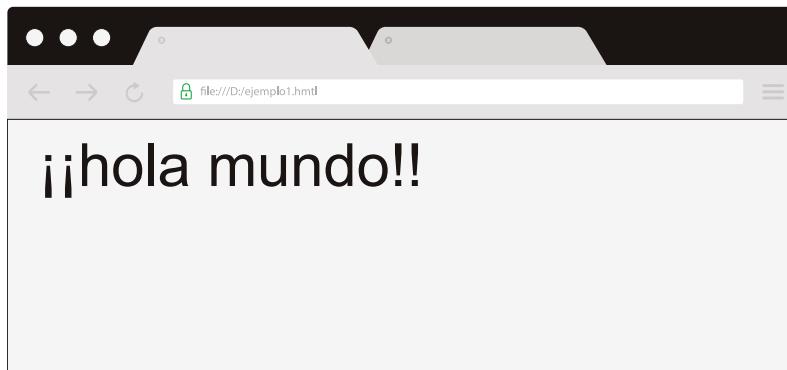


Figura 1.5. Página web de ejemplo

La estructura básica de un archivo HTML contiene lo siguiente:

Etiqueta	Descripción
<!DOCTYPE html>	Esta etiqueta es informativa e indica al navegador que el tipo de documento es html5.
<html>	Todo el documento está contenido en esta etiqueta también llamado nodo raíz.
<head>	La sección <head> provee información sobre el documento en sí mismo. También se usa para invocar archivos externos Javascript o de estilos. Esta sección no se dibuja en la pantalla.
<meta charset="utf-8">	La etiqueta <meta> se usa para introducir información acerca del documento. En este caso se está indicando que use la codificación de caracteres 'utf-8' que permite visualizar caracteres en español, entre otros.
<title>Título aquí </title>	La etiqueta <title> define un título para la página que aparecerá en el borde superior de la página.
</head>	Esta etiqueta indica el final del bloque <head>.
<body>	La sección <body> es el cuerpo de la página y contiene las etiquetas que se visualizarán o dibujarán en la pantalla del navegador.
Contenido de la página	
</body>	Esta etiqueta indica el final del bloque <body>
</html>	Por último esta etiqueta indica el final del documento html.

Figura 1.6. Estructura básica de una página HTML.

La especificación oficial y completa de los elementos que componen el lenguaje HTML están disponibles en <https://www.w3.org/TR/html5/>

A continuación se enunciarán algunas de las más importantes (W3C, 2016).

1.2.1 Etiquetas básicas

Etiqueta	Descripción	Etiqueta	Descripción
<h1></h1>	Encabezados Grande		Define una lista de elementos sin importar su orden
<h2></h2>	Encabezados Mediano		Define una lista de elementos donde importa el orden de los mismos.
<h3></h3>	Encabezados pequeño		Representa un ítem de una lista tipo o
<p>	Define un párrafo de texto		Para resaltar texto importante.
<pre></pre>	Define un párrafo de texto ya formateado		Para enfatizar un contenido.
 	Salto de línea	<!-- -->	Se usan para realizar comentarios.

Un ejemplo para las etiquetas anteriores.

```

1  <!DOCTYPE html>
2  <html>
3      <head>
4          <meta charset="UTF-8">
5          <title>Ejemplo con etiquetas básicas</title>
6      </head>
7      <body>
8          <h1>Etiquetas HTML</h1>
9          <p>
10         Este ejemplo muestra cómo combinar algunas de las etiquetas más básicas de HTML5. <br>
11         Recuerde que <strong>es importante entender la diferencias entre ellas</strong>
12
13
14         <h2>Etiqueta ul+li</h2>
15         <p>
16         Si hacemos una lista de mercado donde el orden no es importante, debemos usar <em>ul</em>.
17         </p>
18         <ul>
19             <li>Piña</li>
20             <li>Manzanas</li>
21             <li>Uchuva</li>
22             <li>Uvas</li>
23         </ul>
24         <h2>Etiqueta ol+li</h2>
25         <p>
26         En el caso de que estemos listando elementos donde el orden es importante, como por ejemplo la clasificación mundial de tenistas , debemos usar <em>ol</em>.
27         </p>
28         <ol>
29             <li>Andy Murray</li>
30             <li>Rafael Nadal</li>
31             <li>Stan Wawrinka</li>
32             <li>Novak Djokovic</li>
33         </ol>
34     </body>
35 </html>

```

Figura 1.7. Ejemplo de utilización de etiquetas básicas.

Al visualizar la página aparece lo siguiente:



The screenshot shows a web browser window with the title "Ejemplo con etiquetas b". The address bar displays "file:///D:/ejemplo11.html". The page content is as follows:

Etiquetas HTML

Este ejemplo muestra cómo combinar algunas de las etiquetas más básicas de HTML5.
Recuerde que **es importante entender la diferencias entre ellas**.

Etiqueta ul+li

Si hacemos una lista de mercado donde el orden no es importante, debemos usar *ul*.

- Piña
- Manzanas
- Uchuva
- Uvas

Etiqueta ol+li

En el caso de que estemos listando elementos donde el orden es importante, como por ejemplo la clasificación mundial de tenistas , debemos usar *ol*.

1. Andy Murray
2. Rafael Nadal
3. Stan Wawrinka
4. Novak Djokovic

Figura 1.8. Página web de ejemplo con etiquetas básicas.

1.2.2 Etiquetas para definir bloques o contenedores.

El elemento <div> se usa para definir agrupamientos lógicos dentro de un página HTML. Estos agrupamientos son útiles ya que facilitan el manejo de los elementos agrupados o contenidos en el bloque al momento de asignar estilos o agregar dinámicamente nuevos nodos hijos (W3C, 2016).

El elemento también sirve para agrupar elementos pero aplica dentro de una misma línea. Es útil para el manejo de estilos dentro de una misma línea de texto.

1.2.3. Etiquetas para elaborar tablas.

Para la elaboración de tablas se utilizan las siguientes etiquetas (W3C, 2016):

Tag	Descripción	Tag	Descripción
<table> </table>	Define una tabla	<tr></tr>	Define una fila
<thead></thead>	Define los elementos que conforman el encabezado de la tabla	<td></td>	Define un campo
<tbody> </tbody>	Define los elementos que componen el cuerpo de la tabla	<th></th>	Define un encabezado de una columna

A continuación un ejemplo de una tabla definida en lenguaje HTML:

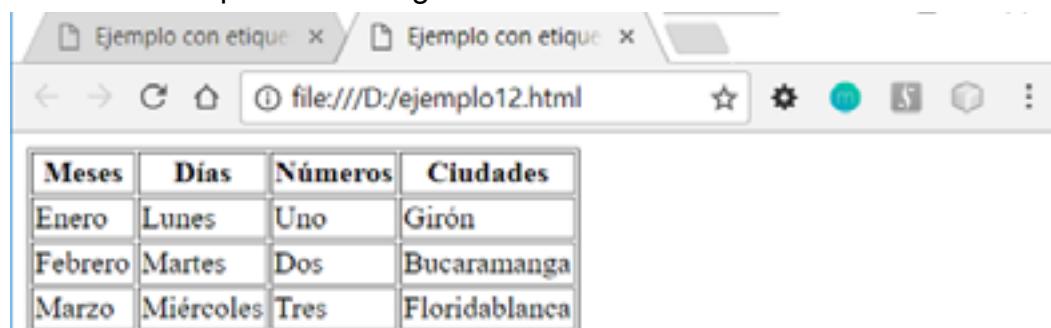
```

1  <!DOCTYPE html>
2  <html>
3      <head>
4          <meta charset="UTF-8">
5          <title>Ejemplo con etiquetas básicas</title>
6      </head>
7      <body>
8          <table border=1>
9              <thead>
10             <tr>
11                 <th>Meses</th>
12                 <th>Días</th>
13                 <th>Números</th>
14                 <th>Ciudades</th>
15             </tr>
16             </thead>
17             <tbody>
18                 <tr> <td>Enero</td>      <td>Lunes</td>      <td>Uno</td> <td>Girón</td>      </tr>
19                 <tr> <td>Febrero</td>     <td>Martes</td>     <td>Dos</td> <td>Bucaramanga</td>   </tr>
20                 <tr> <td>Marzo</td>       <td>Miércoles</td> <td>Tres</td> <td>Floridablanca</td></tr>
21             </tbody>
22         </table>
23     </body>
24 </html>

```

Figura 1.9. Ejemplo de una tabla en HTML.

Una vez visualizada aparecerá lo siguiente:



Meses	Días	Números	Ciudades
Enero	Lunes	Uno	Girón
Febrero	Martes	Dos	Bucaramanga
Marzo	Miércoles	Tres	Floridablanca

Figura 1.10. Página web con una tabla.

1.2.4 Etiquetas para el manejo de formularios.

Los formularios HTML son una herramienta de gran utilidad para el desarrollo de aplicaciones de bases de datos. Un formulario almacena temporalmente los campos que luego de ser validados serán enviados a la base de datos.

La estructura básica del un formulario en HTML es la siguiente (W3C, 2016):

```
<form action="procesarFormulario.php">
    Enunciado del campo:
    <input type="text" name="campo1">
    <input type="submit" value="Enviar">
</form>
```

La figura 1.10 describe la estructura de un formulario.

Etiqueta	Función
<form>	Inicia un bloque de formulario
<form action="programa.php"	El atributo "action" indica al navegador cual programa procesará el formulario una vez se haya oprimido el botón tipo "submit" o botón de enviar.
<input type="tipo de campo">	Define un campo del formulario. El atributo "type" indica el tipo de campo.
<input type="text">	Define un campo de tipo texto. Además de "text" existen otros 21 tipos de campos. Entre ellos están : number, date, password, hidden, color, checkbox.
<input type="text" name="nombre_de_campo">	Define un campo tipo texto y define "nombre_de_campo" como el nombre que se asociará al valor ingresado en dicho campo. Este valor es el que se envía cuando se oprime el botón de enviar.
<input type="submit">	Define un tipo de campo especial que indica al navegador que ya puede procesar el formulario
</form>	Finaliza un bloque de formulario

Figura 1.10. Estructura de un formulario HTML

A continuación un ejemplo de un formulario y su correspondiente representación en un navegador.

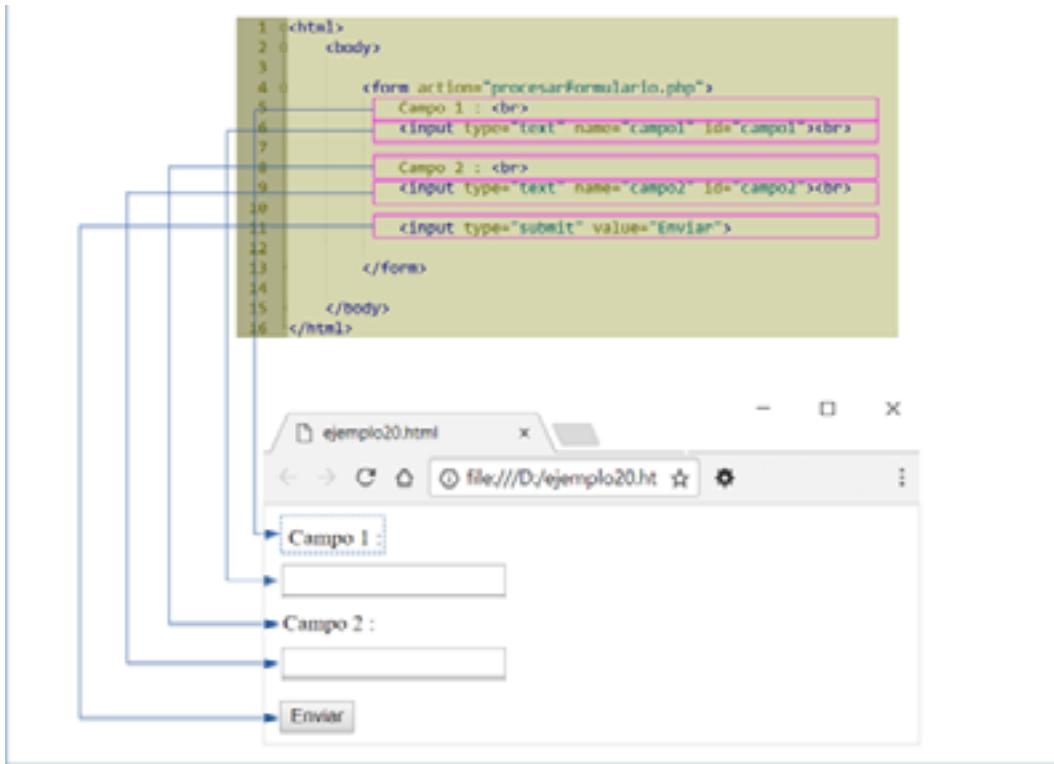


Figura 1.11. Formulario en HTML.

1.2.5 Etiqueta para el manejo de Imágenes.

Un documento HTML puede contener una o varias imágenes las cuales se pueden incluir usando la etiqueta vacía `` y su estructura es la siguiente (W3C, 2016):

```

```

Donde:

- src : archivo fuente de la imagen.
- alt : nombre alternativo que aparece cuando la imagen no se puede cargar
- width : ancho definido
- height : altura de la imágenes.

A continuación un ejemplo de una página web que incorpora una imagen.

```

1 <html>
2   <head>
3     <title> Ejemplo Imagen </title>
4   </head>
5   <body>
6     <h2>Glaciar Argentino </h2>
7     
8   </body>
9 </html>

```

Figura 1.12. Ejemplo diseño web con una imagen incorporada.

Al visualizar la página en un navegador se obtiene lo siguiente:



Figura 1.13. Página web con una imagen.

1.3. Contenido estático vs contenido dinámico.

En el diseño web existen dos tipos de contenido:

- a) **Contenido estático:** proviene de páginas desarrolladas y almacenadas en el sitio web.
- b) **Contenido dinámico:** proviene de páginas o elementos que son creados al instante por el navegador a petición de un programa.

1.4. Fases en el diseño de un sitio web.

Para el diseño web existen muchas metodologías. En este recurso se utilizarán los modelos de desarrollo de software vistos en la actividad de proyecto 1 en especial el modelo en cascada. Bajo este modelo de procesos aplicado al diseño web se toman las siguientes etapas: Planeación, Diseño, desarrollo e implantación.

- a) Planeación:** se determina el concepto de diseño que se quiere realizar. En esta etapa es importante que participe un diseñador gráfico que aporte su criterio en conceptos como: manejo de colores, elementos visuales, manejo de la marca.
- b) Diseño:** se realizan las maquetas con los distintos elementos que conforman el sitio como son: menús, formularios, encabezados, pies de página. Estas maquetas se pueden realizar en herramientas ya mencionadas como son Balsamic Mockups, Gomockingbird, entre otras. Algunos de los productos de esta etapa son: maqueta del sitio (wireframe), diagrama del sitio, entre otros.

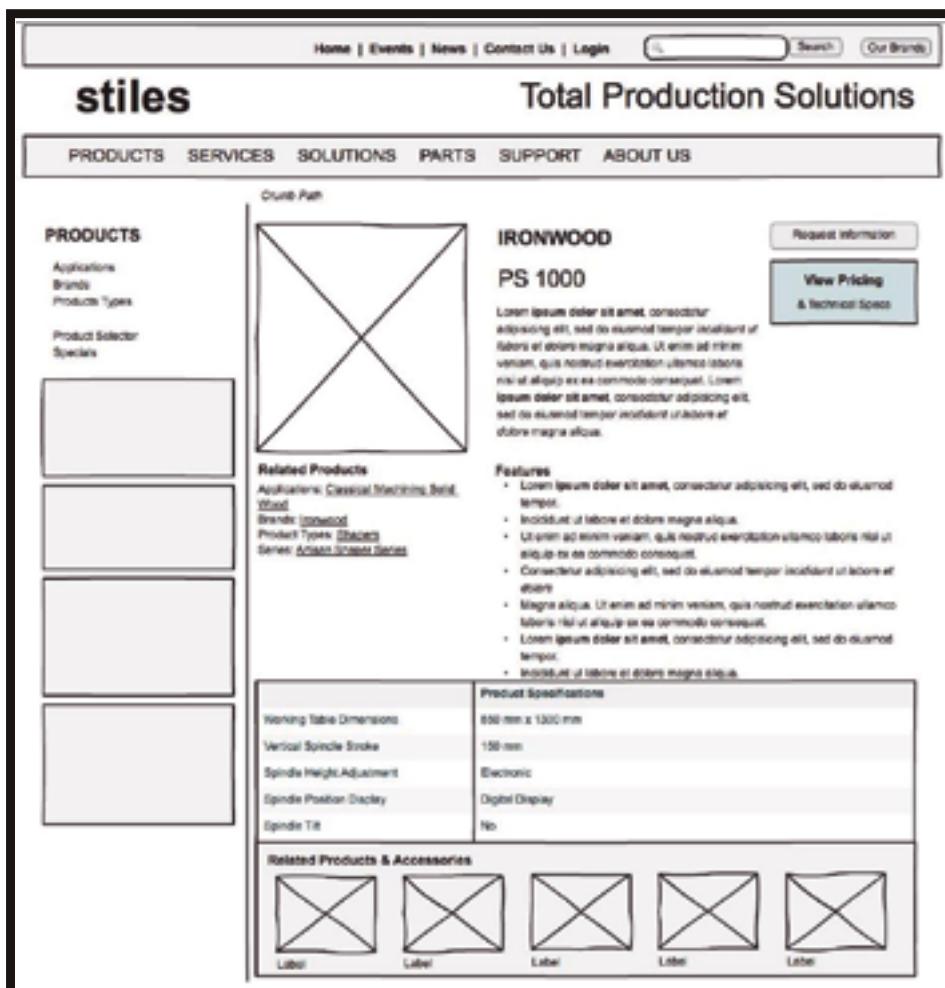


Figura 1.14: maqueta de un sitio web.

- c) **Desarrollo:** en esta fase se crean los distintos archivos html, css y javascript en un servidor de desarrollo o pruebas. También se realizan pruebas al código fuente javascript y se atiende la realimentación por parte del cliente.



Figura 1.15. Página web en producción.

- d) **Implantación:** en esta etapa el sitio web se instala en el servidor de producción para que entre a operar (ver figura 1.15).

1.5. Herramientas para el diseño y desarrollo web.

Además de HTML, CSS y Javascript es importante que el diseñador web conozca de las siguientes herramientas:

- a) **Editor de texto:** como notepad, notepad++,
- b) **FTP:** como winSCP.
- c) **Editor de imágenes:** Gimp, Adobe
- d) **Frameworks CSS:** Como Bootstrap
- e) **Git y github:** para administrar el código fuente.

1.6 Maqueteado (Layouts) en HTML.

La maqueta o “layout” de una página HTML define la ubicación o distribución de sus elementos dentro de la misma página.

Una distribución típica de una página web es la siguiente:

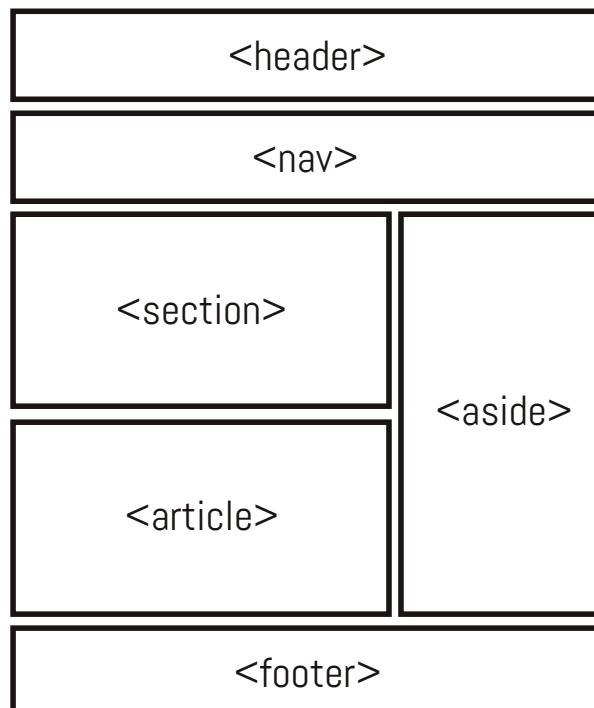


Figura 1.16. Distribución de elementos en una página web.

Donde:

Etiqueta	Función
<header>	Define el encabezado para un documento o sección.
<nav>	Define un contenedor para los links de navegación.
<section>	Define una sección en un documento.
<article>	Define un artículo independiente y autocontenido.
<aside>	Define un contenido “al lado” del contenedor principal.
<footer>	Define un pie de página para un documento o sección.
<details>	Define detalles adicionales.
<summary>	Es el encabezado del elemento <details>.

1.7 Novedades en la versión 5 de HTML.

La versión 5 del lenguaje introduce nuevos elementos que se relacionan más adelante, sin embargo, es importante anotar que HTML5 se refiere a un conjunto de nuevas características que se pueden utilizar individualmente si están soportadas por el navegador (NIEDERST, 2012) .

Para determinar si un navegador soporta una característica de HTML5 se utiliza una librería en Javascript llamada “Modernizr”. De esta forma un desarrollador puede hacer su código compatible tanto para nuevos navegadores como para versiones anteriores de los mismos.

1.7.1. El elemento <canvas>

Es un “lienzo” digital sobre el cual se pueden hacer dibujos en formato de mapa de bits usando el lenguaje Javascript.

1.7.2. Video.

HTML5 define un nuevo elemento llamado <video> para incorporar video en las páginas web de forma nativa.

Anteriormente los videos se introducían a las páginas web usando productos de terceros como Adobe Flash, Microsoft ActiveX o Apple Quicktime. Con el tiempo esos productos de terceros generaban problemas de incompatibilidades entre plataformas, versiones , etc.

1.7.3. Almacenamiento local.

El almacenamiento local provisto por HTML5 permite al sitio web almacenar datos en el computador para consultarlos posteriormente. Este sistema es parecido a las “cookies” pero fue diseñado para manejar mayor cantidad de información. Por otra parte las “cookies” son enviadas al servidor cada vez que se consulta una página lo cual incrementa el tráfico de la red.

1.7.4. Web Workers.

Los “web workers” permiten a los navegadores ejecutar código Javascript en background. Lo anterior quiere decir que un “web worker” hace uso de las capacidades multiproceso de los procesadores modernos para ejecutar tareas de manera simultánea con las tareas normales de un navegador.

1.7.5. Aplicaciones web fuera de línea.

Además de ejecutar páginas web en línea una nueva característica del HTML5 permite ejecutar sitios web aún estando fuera de línea. Para que esta característica funcione se requiere que el navegador visite el sitio y solicite todos los archivos requeridos para su ejecución fuera de línea. Posteriormente cuando el navegador se encuentre sin conexión puede seguir navegando en la copia almacenada.

Una vez se restablezca la conexión se actualizan los archivos almacenados y el navegador sigue funcionando en línea.

1.7.6. Geolocalización.

Es una característica que permite determinar la localización del computador que está ejecutando el browser. El objeto de geolocalización suministrará las coordenadas del sitio pero previamente solicitará autorización al usuario.

1.7.7. Nuevos tipos de cajones para formularios.

HTML5 define los siguientes nuevos “input types”:

Etiqueta	Descripción
<input type="search">	Utilizada para textos de búsqueda.
<input type="number">	Utilizada para capturar números.
<input type="range">	Utilizada para capturar número en un rango específico.
<input type="color">	Utilizada para capturar el código de un color en particular.
<input type="tel">	Utilizada para capturar números de teléfono.
<input type="url">	Utilizada para capturar direcciones de internet.
<input type="email">	Utilizada para capturar direcciones de correo.
<input type="date">	Utilizada para capturar fechas.
<input type="month">	Utilizada para capturar un mes en particular.
<input type="week">	Utilizada para capturar una semana del año en particular.
<input type="time">	Utilizada para capturar una hora del día.
<input type="datetime">	Utilizada para capturar una fecha y hora en formato de máquina.
<input type="datetime-local">	Utilizada para capturar una fecha y hora en formato local.

También se introducen los siguientes restricciones que se aplican a los contenidos en los campos tipo <input>:

Atributo	Valor esperado	Descripción
placeholder	Texto	Muestra una sugerencia en el cajón para el texto esperado.
required	No aplica	Valida que el campo no se envíe vacío.
pattern	Expresión regular	Permite validar el texto ingresado usando una expresión regular.
min, max	Número	Especifica el valor mínimo y máximo que pueden tomar los valores numéricos.
step	Número	Especifica los intervalos permitidos para el campo numérico.

1.7.8. Nuevas etiquetas introducidas en la versión 5 del lenguaje HTML.

Además de las etiquetas header, nav, section, aside, footer, con las cuales se define la estructura básica de una página web con HTML5, a continuación, se presentan nuevas etiquetas a utilizar dentro del contenido de la página web:

Etiqueta	Descripción
<article> </article>	Define unidades independientes de contenido dentro del elemento <section>.
<hgroup> <h1>titulo</h1> <h2>subtitulo</h2> </hgroup>	Etiqueta usada dentro de la sección header cuando esta contiene dos o más elementos de tipo h como h1, h2, h3. Su objetivo es ayudar al navegador a interpretar y procesar correctamente la página Web.
<mark>	Permite hacer énfasis en una palabra o frase.
<small>	Muestra una palabra o frase en letra pequeña con fuente de información legal.
<address>	Para definir información de contacto, normalmente usado dentro de la sección <footer>.
<time></time>	Permite formatear campos de hora para fácil entendimiento.

También se introducen los siguientes restricciones que se aplican a los contenidos en los campos tipo <input>:

Etiqueta	Descripción
<audio> Forma 1: <audio src="miAudio.mp3"></audio> Forma 2: <audio> <source src="miAudio.mp3"> <source src="miAudio.ogg"> </audio>	Usada para reproducir archivos de audio con extensión ogg o mp3. Dependiendo del navegador podrá reproducir un tipo de archivo u otro, por esta razón es recomendable incluir ambos archivos y especificar las dos posibles fuentes con la etiqueta interna source, para que el navegador procese el archivo apropiado. Atributos. src: url o nombre archivo a reproducir. controls: presenta los controles de reproducción. autoplay: reproducción automática del audio. loop: reproducciones infinitas del audio.
<video> Forma 1: <video src="miVideo.mp4"></video> Forma 2: <video> <source src="miVideo.mp4"> <source src="miVideo.ogg"> </video>	Define unidades independientes de video con extensión ogg o mp4. Dependiendo del navegador podrá reproducir un tipo de archivo u otro, por esta razón es recomendable incluir ambos archivos y especificar las dos posibles fuentes con la etiqueta interna source, para que el navegador procese el archivo apropiado. Atributos. src: url o nombre archivo de video a reproducir. controls: presenta los controles de reproducción. autoplay: reproducción automática del video. loop: reproducciones infinitas del video. poster: provee una imagen que será mostrada.

2. Estilos en cascada (“Cascade-Style Sheet” o CSS).

Las hojas de estilo en cascada o CSS es un lenguaje para describir la forma como los documentos estructurados como HTML o XML serán dibujados en un pantalla, escritos en papel o leídos en voz alta, etc. (W3C, 2016).

El lenguaje CSS es un componente fundamental en el diseño web cuyo principal objetivo es describir la presentación que tendrán los elementos estructurales que conforman una página HTML. Por medio de las reglas definidas en el lenguaje CSS se pueden insertar bordes, asignar colores al texto, asignar colores al fondo, entre muchos otros.

2.1. Reglas o comandos CSS.

CSS suministra un conjunto de reglas que son interpretadas por los navegadores al momento de dibujar las páginas en la pantalla.

Una regla básica CSS tiene la siguiente estructura:

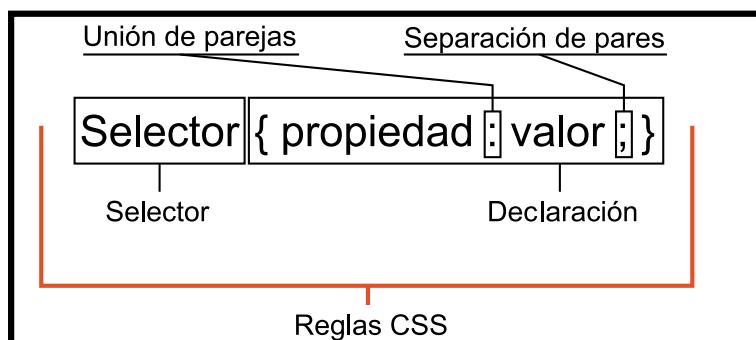


Figura 2.1a. Estructura de una regla CSS.

- a) **Regla CSS:** conjunto compuesto por un selector con una declaración.
- b) **Selector:** indica el elemento HTML sobre el cual operará la regla.
- c) **Declaraciones:** par o conjunto de pares con la estructura propiedad, valor.
- d) **Propiedad:** característica del elemento HTML que se quiere intervenir o afectar.
- e) **Valor:** valor a aplicar a la propiedad.

Ejemplo:

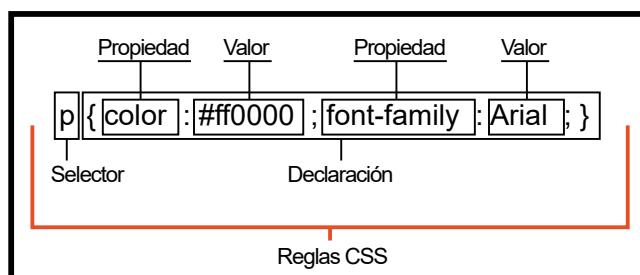


Figura 2.1b. Ejemplo de una regla CSS.

2.1.1 Selectores.

Los selectores se encargan de hallar o encontrar los elementos a los cuales aplicará la regla.

Los selectores según los elementos que intervienen son:

a) Selector de elemento.

Seleccionan un elemento HTML en particular por ejemplo el elemento `<p>` que se utiliza para definir párrafos en un página web.

```
p {  
    color : green;  
    text-align: center;  
}
```

Figura 2.2. Ejemplo selector de elemento.

En el ejemplo 2.2 se define una regla con dos declaraciones: una indica que el color del párrafo es verde, otra indica que la alineación del texto es centrado.

b) Selector por ID de elemento.

Este selector utiliza el atributo ID que puede ser asignado a cada elemento HTML. Inicia con el símbolo “#” seguido de un identificador único. Este estilo queda automáticamente aplicado al elemento con el identificador.

```
#fecha1 {  
    text-align: center;  
    color: red;  
}
```

Figura 2.3. Ejemplo selector por ID de elemento.

En el ejemplo se define una regla para que el elemento con ID igual a “fecha1” se aplique el color rojo y la alineación de texto centrada.

c) Selector por clase.

Este selector afecta a los atributos de una clase en particular. Su definición inicia con un punto seguido de un identificador. Para aplicar este estilo se usa el atributo “class” seguido por el nombre de la clase. Para este caso sería: <elemento class="miEstilo">

```
.miEstilo{
    background: yellow ;
    text-align: left;
}
```

Figura 2.4. Ejemplo selector de clase

d) Selector por pseudo-clase.

Algunos elementos del lenguaje HTML como por ejemplo los vínculos o links (<a>) pueden cambiar de color dependiendo si sobre ellos se ha dado click o no. Cuando se aplica un estilo sobre un elemento HTML cuyo estado ha cambiado se dice que se está aplicando un estilo a una pseudo-clase.

A continuación se relacionan algunos estados usados en CSS:

- a):link: que no ha dado click sobre el elemento.
- b):visited: se activa cuando se ha dado click sobre el elemento o ya fue visitado.
- c):focus: se activa cuando el elemento ya está listo para ingresar texto.
- d):hover: se activa cuando el puntero del ratón está sobre el elemento.
- e):active: se activa cuando se está interactuando con el elemento.

A continuación un ejemplo de selección de pseudo-clases:

```
1 a:hover {
2     color: maroon;
3 }
4 a:link {
5     color: blue;
6 }
7 a:visited {
8     color: red;
9 }
10 a:active {
11     color: green;
12 }
```

Figura 2.5. Ejemplo selector de pseudo-clase

En el ejemplo se dan diferentes colores a la etiqueta usada para vínculos externos o links de acuerdo a su estado.

Además de las pseudo-clases ya descritas existen otras que pueden llamarse estructurales ya que aparecen como resultado de cambios en la estructura del DOM.

Pseudo-clase	Aplica a	Ejemplo	Significado
:nth-child()	Un determinado hijo	p:nth-child(3) { background: #FFDDAA; }	El tercer elemento P de la página tendrá un determinado color de fondo
:first-child	El primer hijo de un determinado elemento	p:first-child { background: #FFDDAA; }	El primer elemento P de la página tendrá un determinado color de fondo
:last-child	El último hijo de un determinado elemento	p:last-child { background: #FFDDAA; }	El último elemento P de la página tendrá un determinado color de fondo
:only-child	El único hijo de un determinado elemento	p:only-child { background: #FFDDAA; }	El elemento P de la página tendrá un determinado color de fondo siempre y cuando sea el único P en la página
:even	Un determinado elemento siempre que ocupe una posición par dentro de la lista de elementos del mismo tipo	p:nth-child(even) { background: #FFDDAA; }	Los elementos P que estén en una posición par dentro de la lista de elementos P tendrán un determinado color de fondo
:odd	Un determinado elemento siempre que ocupe una posición impar dentro de la lista de elementos del mismo tipo	p:nth-child(odd) { background: #FFDDAA; }	Los elementos P que estén en una posición impar dentro de la lista de elementos P tendrán un determinado color de fondo
:not	Un elemento que se quiera dejar por fuera	not(p:nth-child(3) { background: #FFDDAA; }	Un determinado color de fondo no se aplica

Figura 2.6. Tipos de pseudo-clases.

e) Selector Universal.

El selector universal es el asterisco (*) e indica que la regla aplica a todos los elementos.

2.1.2. Métodos para incorporar estilos a las páginas HTML.

Existen tres maneras de incorporar los estilos CSS en un documento HTML.

- a) De manera externa a través de un archivo separado de estilos con extensión.css.
- b) De manera interna usando una sección de estilos definida en la sección HEAD de la página.
- c) De manera interna asignando un estilo a cada elemento o “Inline style”.
- d) Estilos importados desde un bloque CSS.

a) Hojas de estilos CSS externa.

Todas las reglas CSS se determinan en un archivo guardado con extensión (.CSS), para luego ser vinculada mediante la etiqueta HTML <link>. Un archivo CSS se puede vincular a cuantos HTML se quiera y un HTML puede contener cuantos archivos CSS sean necesarios.

Los archivos CSS se vinculan en el <HEAD> del HTML de la siguiente manera:

```
<head>
<link rel="stylesheet" type="text/css" href="miArchivo.css" />
</head>
```

Figura 2.7. Inclusión de estilos a través de archivo separado.

El archivo CSS se puede hacer en cualquier editor de texto guardándolo con la extensión .css.

Un estilo también puede ser traído directamente desde Internet como aparece a continuación:

b) Estilos CSS Incrustados en la sección <head>.

Se incrustan en el <head> usando la etiqueta <style> con el atributo type="text/css". Pero esos estilos solo funcionarán para ese documento HTML en específico.

```
<head>
<style type="text/css">
/*estilos incrustados*/
</style>
</head>
```

Figura 2.8: Inclusión de estilos en la sección <head> del documento.

c) Estilos CSS Inline.

El estilo inline se usa para dar un estilo a un elemento HTML en particular. Es de los menos utilizados ya que se mezcla elementos de estructura y visualización que deberían estar separados para mejorar su entendimiento. En este caso la palabra clave "style" va como un atributo de la etiqueta.

```
<p style="color:#0000ff; margin-left:20px">Este es el parrafo.</p>
```

Figura 2.9. Inclusión de estilos como un atributo de la etiqueta

d) Estilos importando desde un bloque CSS.

Un estilo puede ser importando usando la sentencia CSS @import como se ilustra en el siguiente ejemplo:

```

1 <style TYPE="text/css">
2 <!--
3   @import url(http://www.htmlhelp.com/style.css);
4 -->
5 </style>
```

Figura 2.10. Inclusión de una hoja de estilos dentro de un bloque CSS.

Ejemplo de aplicación de estilos CSS a una página HTML.

```

1 <html>
2   <head>
3     <title>Mi primer CSS</title>
4     <style type="text/css">
5       h1 {color:#F00;}
6       p {color:#006600;}
7       .textoAzul{color:#0033CC;font-family:Arial;}
8       #textoNaranja{color:#FF6600;}
9     </style>
10   </head>
11   <body>
12     <h1>Este encabezado es de color Rojo</h1>
13     <p>El color de este texto es verde</p>
14     <p class="textoAzul">yo soy de color azul y fuente Arial que
15       provienen de una clase</p>
16     <p id="textoNaranja">yo tengo el texto naranja por una regla de un selector ID</p>
17   </body>
18 </html>
```

Figura 2.11. Ejemplo de aplicación de estilos a una página web.

Al abrir el archivo en un navegador aparece lo siguiente:



Figura 2.12. Página web con ejemplos de estilos.

2.2. Conceptos básicos de CSS

CSS incorpora los siguientes conceptos de diseño que se deben tener en cuenta al momento de elaborar las páginas web (NIEDERST, 2012).

2.2.1. Herencia.

Como se describió anteriormente un documento HTML se convierte en un DOM que está compuesto por nodos. El nodo raíz es la etiqueta <html>. Por tanto esta etiqueta es el padre de todos los elementos que se crean de ahí en adelante.

En la medida que se van creando elementos estos van a tener un parent o en su forma generalizada van a tener unos ancestros. Los hijos de un mismo nodo serán nodos hermanos (siblings).

Muchas de las reglas CSS, en especial las relativas al texto como son fuente, tamaño, estilo, etc se heredan a los nodos hijos. Para saber exactamente si un elemento soporta herencia se debe consultar en el manual de referencia de CSS.

En este ejemplo se define un párrafo de texto de la siguiente forma:

```

1 <html>
2   <body>
3     <p style="color:blue;"> Esta es una prueba de <strong> herencia </strong> </p>
4   </body>
5 </html>
```

Figura 2.13. Ejemplo de herencia en un documento HTML.

Al visualizar la página en un navegador se obtiene lo siguiente:



Figura 2.14. Ejemplo de herencia visualizado en un navegador.

Se puede observar que el texto resaltado, “herencia”, heredó el color azul definido para el párrafo ya que la etiqueta “” es hija del nodo “<p>”.

2.2.2. Modelo de la caja (Box Model)

Desde el punto de vista de diseño cada etiqueta HTML puede ser representada como una caja o rectángulo. Esta caja introduce unas propiedades que tiene cada etiqueta como se muestra en la figura 2.15.

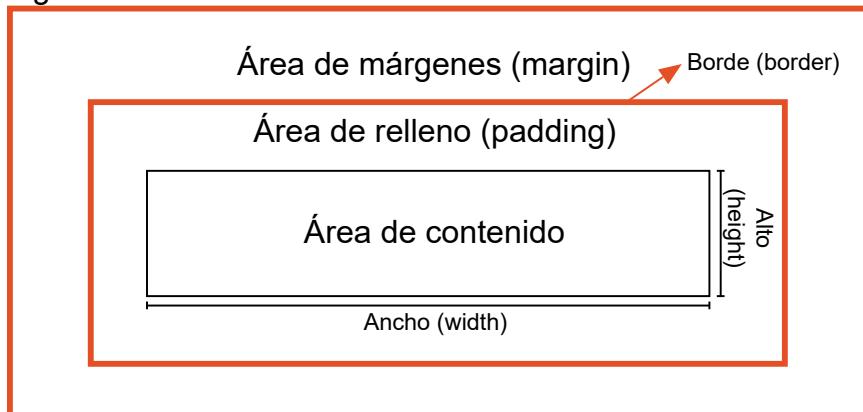


Figura 2.15. Modelo de Caja en CSS

A continuación un ejemplo de aplicación del modelo de caja a un elemento HTML.

```

1 <html>
2   <head>
3     <style type="text/css">
4       p {
5         background : yellow;
6         height      : 50px ;
7         width       : 100px ;
8         padding     : 20px ;
9         border     : 2px solid blue ;
10        margin    : 30px
11      }
12    </style>
13  </head>
14  <body>
15    <p> HOLA MUNDO </p>
16  </body>
17 </html>

```

Figura 2.16. Ejemplo para visualizar el modelo de caja.

Una vez visualizado en el navegador aparece lo siguiente:

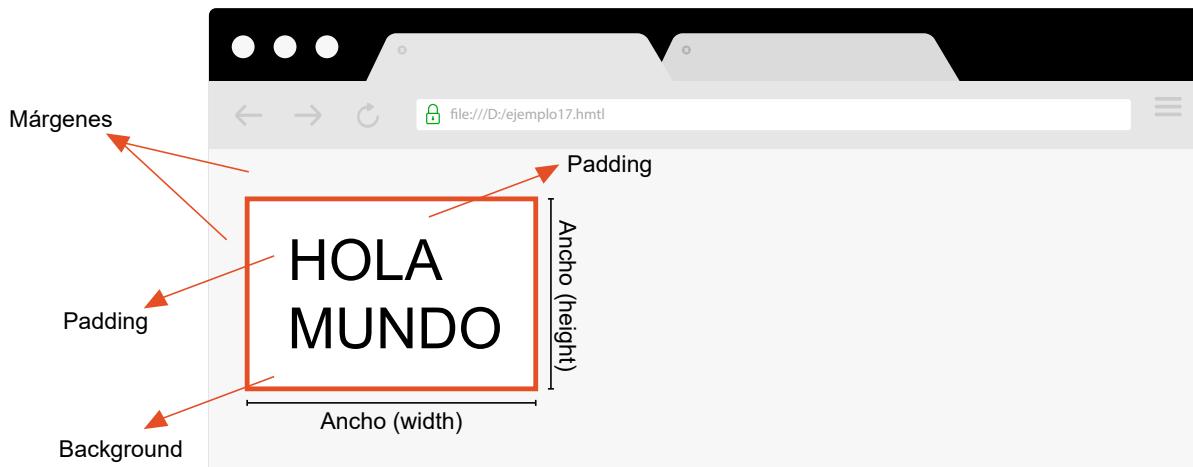


Figura 2.17. Ejemplo del modelo de caja aplicado a un elemento HTML.

2.2.3. Formas de aplicación de los estilos.

Como se mencionó anteriormente los estilos se pueden definir en un archivo separado, en una sección del archivo HTML o de forma incrustada o “inline”.

Puede suceder que para un mismo elemento, clase o atributo hayan sido definidas reglas que pueden ser contradictorias.

En general entre más cerca esté la regla definida de la etiqueta, más prioridad tiene al momento de aplicarse. El orden aplicado es el siguiente de menor a mayor prioridad:

- 1) Los estilos por defecto del navegador.
- 2) Estilos de usuario definidos en el navegador (reader style sheet)
- 3) Estilos en archivos separados y vinculados con la etiqueta <link>
- 4) Estilos importados con la función @import.
- 5) Estilos embebidos dentro de la página html en una sección aparte.
- 6) Estilos aplicados un elemento en particular con el comando “style”
- 7) Cualquier regla marcada como importante (!important) por el autor.
- 8) Cualquier regla marcada como importante (!important) por el usuario.

2.3 Librerías y frameworks CSS.

Existen bibliotecas de estilos que facilitan la labor del desarrollador al integrar un conjunto de estilos que se pueden integrar en sus aplicaciones web en su mayoría de manera gratuita.

Un ejemplo es la librería es animate.css que permite realizar animaciones a los textos. La librería alertify permite generar mensajes dentro de la página web de manera sencilla.

También existen frameworks que son un conjunto de librerías que abarcan no solo uno sino varios aspectos de la tecnología CSS.

Un framework ampliamente utilizado (según su popularidad en el sitio Github.com) es Bootstrap el cual provee gran variedad de estilos para cada elemento HTML y además habilita el diseño adaptativo (Responsive Design) del sitio web.

El diseño adaptativo es una estrategia para que el sitio web se adapte a los distintos tamaños de pantallas que existen como son: celulares, tabletas, portátiles y computadores de escritorio.

2.4. Ejemplo de maquetación con HTML y CSS.

A continuación una página HTML sin personalización del estilo.

```
1  <!DOCTYPE html>
2  <html lang="es">
3      <head>
4          <meta charset="iso-8859-1">
5          <meta name="description" content="Ejemplo de Maquetacion en HTML">
6          <title>Este texto es el título del documento</title>
7      </head>
8      <body>
9          <header>
10             <h1>Encabezado de la pagina web, elemento header</h1>
11             </header>
12             <nav>
13                 <h4>barra de menu, elemento nav</h4>
14             </nav>
15             <section>
16                 Este es el contenido principal<br /> de la
17                 pagina web, elemento section
18             </section>
19             <aside>
20                 Elemento barra lateral<br />
21                 uno<br />dos<br />tres<br />
22             </aside>
23             <footer>
24                 Información de la empresa, elemento footer
25             </footer>
26         </body>
27     </html>
```

Figura 2.18. Ejemplo definición de página web sin estilo personalizado.

Al visualizar la página en un navegador se obtiene lo siguiente:



Figura 2.19. Ejemplo visualización de página web sin estilo personalizado.

Se aprecia en la imagen anterior que no obstante se estructuró la página Web con las respectivas zonas header, nav, aside y footer, el resultado obtenido difiere de un modelo estructurado. Esto se debe al modelo de interpretación por defecto de los navegadores conocido como modelo de caja ya introducido en una sección anterior.

Para personalizar la apariencia de esta página Web, se requiere emplear la tecnología CSS como se describe a continuación.

Dentro de un archivo CSS se deben definir las reglas de estilo que se deben aplicar a las etiquetas HTML de una página web. Cada regla de estilo se puede aplicar a todos los elementos del mismo tipo, a un conjunto de etiquetas identificadas con la misma clase o a un elemento en particular.

Para el ejercicio se dará estilo a cada uno de los elementos HTML para eso primero se identifican los elementos a intervenir.

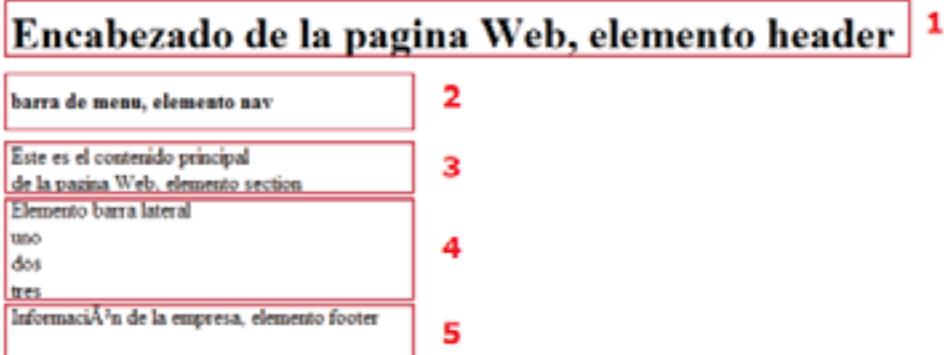


Figura 2.20a. Identificación de las áreas a intervenir con estilos.

Luego se agrupan usando la etiqueta <div> y se identifican con el atributo "id" como se ilustra en la figura 2.20b.

```

1  <!DOCTYPE html>
2  <html lang="es">
3  <head>
4      <meta charset="iso-8859-1">
5      <meta name="description" content="Ejemplo de Maquetación en HTML">
6      <title>Este texto es el título del documento</title>
7  </head>
8  <body>
9      <div id="cuerpo">
10         <header id= "encabezado">
11             <h1>Encabezado de la pagina web, elemento header</h1>
12         </header>
13         <nav id="menu">
14             <h4>barra de menu, elemento nav</h4>
15         </nav>
16         <section id="seccion">
17             Este es el contenido principal<br/> de la
18             pagina web, elemento section
19         </section>
20         <aside id="lateral">
21             Elemento barra lateral<br />
22             uno<br />dos<br />tres<br />
23         </aside>
24         <footer id="pie">
25             Información de la empresa, elemento footer
26         </footer>
27     </div>
28 </body>
29 </html>
```

Figura 2.20b. Identificación de las áreas a intervenir con estilos.

Luego se crea el archivo CSS donde se definirán las características de diseño que aplicarán a la página web. Para esto se digita el siguiente código en un editor de texto.

```

1  @charset "utf-8"
2  * {
3      margin:0px;
4      padding:0px;
5  }
6  h1 {
7      font:bold 20px Verdana, Geneva, sans-serif;
8  }
9  h2 {
10     font:bold 14px Verdana, Geneva, sans-serif;
11 }
12 body{
13     text-align:center;
14 }
15 #cuerpo{
16     width:960px;
17     margin:15px auto;
18     text-align:left;
19 }
```

Figura 2.21a. Reglas CSS para el ejemplo. Parte 1.

```

20 #encabezado{
21     background:#ffffbb9;
22     border: 1px solid #999999;
23     padding: 20px;
24 }
25 #menu{
26     background: #cccccc;
27     padding: 5px 15px;
28 }
29 #seccion{
30     float:left;
31     width:660px;
32     margin:20px;
33 }
34 #lateral{
35     float:left;
36     width:220px;
37     margin:20px 0px;
38     padding:20px;
39     background:#cccccc;
40 }
41 #pie{
42     clear:both;
43     text-align:center;
44     padding:20px;
45     border-top:1px solid #999999;
46 }

```

Figura 2.21b. Reglas CSS para el ejemplo. Parte 2.

A continuación la descripción de las reglas:

Línea	Descripción
2..5	Se establece mediante el selector * que todos los elementos van a tener un margen de 0 píxeles y un espacio entre el contenido de una etiqueta y su contenedor de 0px
6..11	Se define un estilo, tamaño y tipo de fuente específico para todas las etiquetas H1 y H2
12..14	Se define la alineación del texto central para el cuerpo de la página web
15..19	Se establece el ancho, las márgenes y la alineación del texto del elemento identificado con el ID "cuerpo", el cual corresponde a la etiqueta div que encierra al resto de etiquetas
20..24	Se establece un color de fondo, borde y espacio entre el texto y el elemento contenedor para la sección identificada como "encabezado", la cual corresponde a la sección header de la página web
25..28	Se define un color de fondo y un espacio entre el texto y su contenedor para la etiqueta identificada con el nombre "menu", correspondiente a la sección nav de la página web
29..33	Se establece una ubicación (izquierda del área disponible), ancho y margen para la etiqueta identificada con el nombre seccion, correspondiente a la sección section de la página web
34..40	Se establece una ubicación (izquierda del área disponible), ancho, margen y color de fondo para la etiqueta identificada con el nombre "lateral", correspondiente a la sección aside de la página web
41..46	Se restaura el flujo normal de la página web (propiedad clear), es decir que la ubicación y posición del elemento no está ligada a la posición del elemento anterior. Se define la alineación del texto, el espacio entre el texto y su contenedor y el estilo del borde para la etiqueta identificada con el nombre "pie", correspondiente a la sección footer de la página web

Este archivo se guarda con el nombre "miEstilo.css" y se llama desde la sección de la página web con la sentencia: <link rel="stylesheet" href="miEstilo.css">

Una vez visualizada la página con un navegador a aparece lo siguiente:

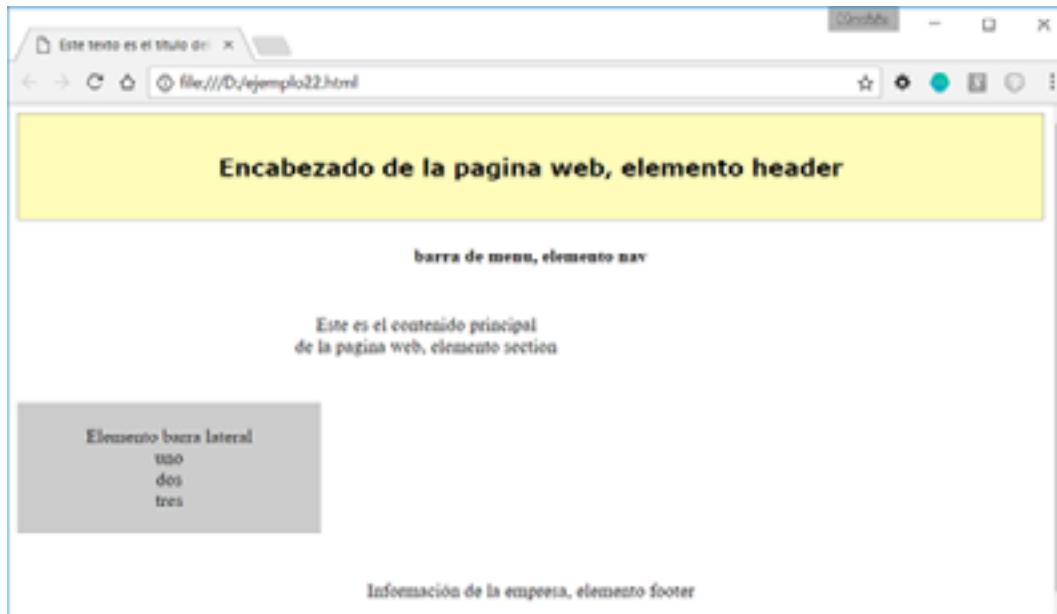


Figura 2.22. Pagina web con estilos aplicados.

A continuación se hará uso de otras etiquetas HTML para complementar el diseño:

```

16 <nav id="menu">
17   <ul>
18     <li>menu 1</li>
19     <li>menu 2</li>
20     <li>menu 3</li>
21   </ul>
22 </nav>
23
24 <section id="seccion">
25   <article>
26     <h2>Titulo del primer articulo</h2>
27     <p>Este es el contenido principal del <mark>articulo 1</mark></p>
28     <footer>
29       <small> fin articulo 1</small>
30     </footer>
31   </article>
32   <article>
33     <h2>Titulo del segundo articulo</h2>
34     <p>Este es el contenido principal del <mark>articulo 2</mark></p>
35     <footer>
36       <small> fin articulo 2</small>
37     </footer>
38   </article>
39 </section>

```

Figura 2.23. Pagina web con etiquetas complementarias.

Una vez visualizado en el navegador aparece lo siguiente:

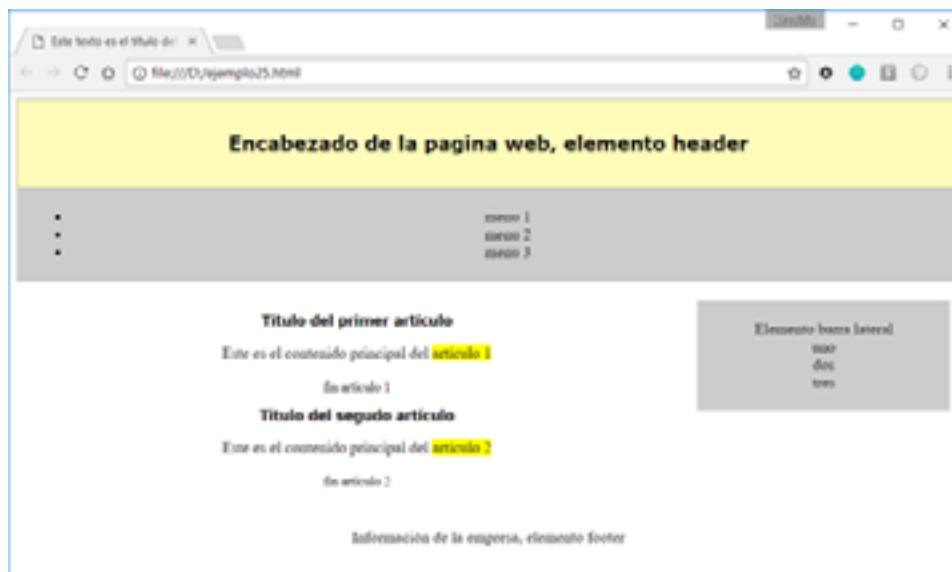


Figura 2.24. Ejemplo de pagina web con etiquetas complementarias.

Ahora se puede personalizar el estilo de este contenido incorporando las siguientes líneas de código al archivo miEstilo.css

```

48 article{
49     background:#FFFBC;
50     border: 1px solid #999999;
51     padding: 20px;
52     margin-bottom: 15px;
53 }
54 article footer{
55     text-align: right;
56 }
57 #menu li{
58     display: inline-block;
59     list-style: none;
60     padding: 5px;
61     font: bold 14px verdana, sans-serif;
62 }
```

Figura 2.25. Ejemplo de estilos adicionales.

La descripción del código CSS es la siguiente:

Línea	Descripción
48..53	Se establece un color de fondo, borde y espacio entre el texto y el elemento contenido para cada etiqueta tipo <article> presente en la página web.
54..56	Define la alineación del texto a la derecha para toda etiqueta <footer> que se encuentre dentro de una etiqueta <article>
57..62	Se establece la forma en la que se debe mostrar (horizontal, sin viñeta, espacio y fuente) cada elemento (lista) dentro de la sección identificada como "menu".

Al visualizar en el navegador aparece lo siguiente:



Figura 2.26. Ejemplo de estilos

2.5 Otros estilos CSS.

Durante las últimas versiones de CSS y gracias a su integración con HTML5, han sido grandes los avances que en relación a estilo e interactividad ha tenido esta tecnología.

Para el uso de estos nuevos estilos, es recomendable incluir los prefijos –moz- y –webkit- para su correcta interpretación en los navegadores basados en motores Gecko y WebKit como Firefox, Safari y Google Chrome.

Ejemplo, para definir la propiedad border-radius a un valor de 20px, se haría así:

-moz-border-radius: 20px y -webkit-border-radius: 20px.

Etiqueta	Efecto	Ejemplo
Border-radius	Esquinas redondeadas. Recibe el valor de la curvatura.	boder-raius:20px
Box-shadow	Sombras. Recibe el color, desplazamiento horizontal y vertical y la difuminación.	box-shadow: #000000 5px 5px 10px
Text-shadow	Sombra para texto. Recibe el color, desplazamiento.	text-shadow: #000000 5px 5px 10px
Linear-gradient	Función aplicada a la propiedad background para dar un efecto de difuminación a un fondo. Recibe el punto inicial y los dos colores horizontal y vertical y la difuminación.	background:linear-radient(top,color,color)
Transform	Modifica la forma de un elemento a través de las funciones scale (escalar), rotate (rotar), skew (inclinar) y translate (trasladar).	<ul style="list-style-type: none"> ▶ transform: scale (2) ▶ transform: rotate (45deg) ▶ transform: skew (20deg,20deg) ▶ transform: translate (50px)

Se puede mejorar el estilo de las esquinas así:

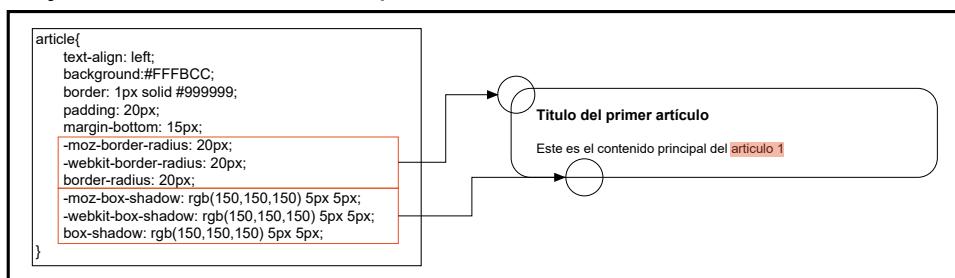


Figura 2.27. Ejemplo del estilo border-radius

Se puede mejorar el estilo de la barra del menú así:

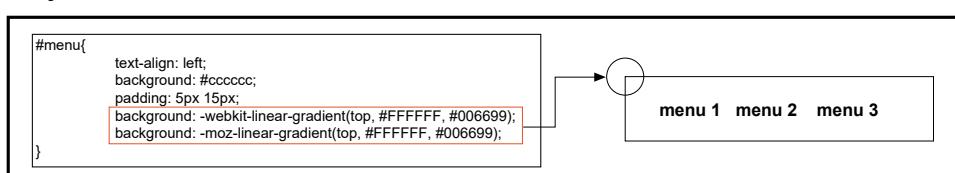


Figura 2.28. Ejemplo del estilo linear-gradient.

Se puede usar la rotación de un elemento así:

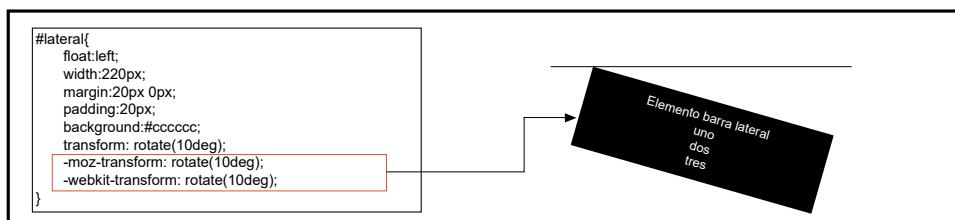


Figura 2.29. Ejemplo del estilo rotate.

La página web con los cambios introducidos queda así:

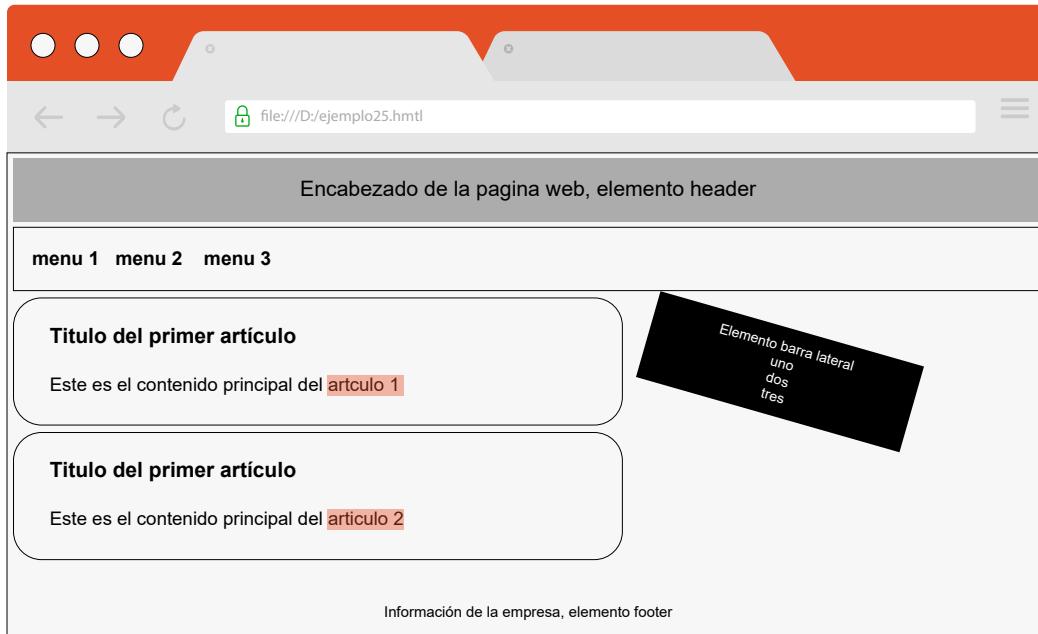


Figura 2.30. Página web con estilos aplicados.

2.5 Formularios con HTML y CSS.

Una vez revisadas las tecnologías HTML junto con las mejoras introducidas en la versión 5 y CSS se pueden construir formularios web con diseños y estilos más atractivos.

Para el ejemplo del recurso se puede agregar un formulario web en la sección <aside> como se muestra en el código siguiente:

```

42 <aside id="lateral">
43   <form class="miForma" name="formulario">
44     <h2> Formulario Web </h2>
45     <label for="txtNombre">Nombre:</label>
46     <input type="text" name="txtNombre" required><br />
47     <label for="txtEmail">Correo:</label>
48     <input type="email" name="txtEmail" placeholder="Ingrese email"/><br />
49     <label for="numEstrato">Estrato:</label>
50     <input type="number" name="numEstrato" min="1" max="6" step="1"/><br />
51     <label for="datFecha">Fecha:</label>
52     <input type="date" name="datFecha" required><br />
53     <input type="submit" name="btnEnviar" value="Enviar"/>
54   </form>
55 </aside>

```

Figura 2.31. Página web con un formulario.

La descripción del código fuente es la siguiente:

Línea	Descripción
43	Se define el inicio del bloque del formulario.
44	Se define un encabezado con el título del formulario.
45	Se usa la etiqueta <label> que es la recomendada para los títulos de los campos de los formularios. El atributo "for=" de la etiqueta <label> indica el campo al cual está asociado.
46	Se define el campo "txtNombre" con la restricción "required" lo cual evita que el campo se envíe vacío.
48	Se define el campo "txtEmail" de tipo "email" lo cual indica al navegador que valide una dirección de correo bien formada. Se usa el atributo "placeholder" que muestra el mensaje "Ingrese email" dentro del campo como una ayuda.
50	Se define el campo "numEstrato" con las restricciones "min" de 1 y "max" de 6 y un incremento o "step" de 1.. Lo que le indica al navegador que valide que el número entrado sea un entero entre 1 y 6.
52	Se define el campo "datFecha" de tipo "date" lo cual indica al navegador que disponga un calendario para su ingreso. Además el campo debe llenarse obligatoriamente (atributo "required").
53	Se define el campo especial tipo "submit" para enviar el formulario. Si se viola alguna regla el formulario no se envía y muestra el mensaje de error.
53	Se cierra el bloque del formulario.

Para la alineación de los campos de un formulario no se aconseja el uso de una tabla ya que este elemento no fue concebido para esta labor. Por otra parte dificulta la adopción de una estrategia de diseño adaptativo. En su lugar se deben aplicar los conceptos de CSS vistos en el recurso.

La versión preliminar del formulario sin estilos es la siguiente:

Formulario Web

Nombre:

Correo:

Estrato:

Fecha:

Figura 2.32. Formulario web mostrado en el navegador.

Para mejorar la apariencia se aplican las siguientes reglas CSS:

```

82 .miForma {
83     width: 180px;
84     clear: both;
85 }
86 .miForma input {
87     width: 100%;
88     clear: both;
89 }

```

Figura 2.33. Estilos para aplicar a los formularios.

Una vez aplicados los estilos el diseño final de la página es el siguiente:

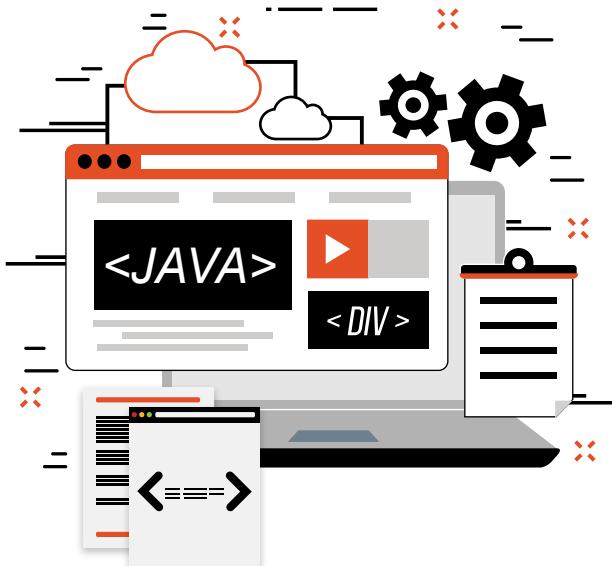


Figura 2.34. Página web con formulario.

3. El lenguaje Javascript.

Javascript provee a las páginas web de interactividad, es decir, permite que se pueda comunicar con el mundo exterior. Es un lenguaje interpretado lo cual quiere decir que no requiere que sea compilado o convertido en código de máquina para ejecutarse.

Una de las principales tareas de Javascript es la validación de los datos del usuario. Por ejemplo validar que una fecha sea válida, que un correo sea válido, etc. Además puede mostrar estos errores al usuario generando una ventana con un mensaje.



Javascript es el lenguaje del navegador y como tal está incorporado dentro del mismo y no requiere otra pieza de software para funcionar. Es importante anotar que Javascript es la principal herramienta de programación del lado del cliente (Client-Side Programming) en contraposición a los lenguajes de programación del lado del servidor (Server-Side Programming) como PHP.

De la misma forma como HTML hace uso de la tecnología CSS también los archivos Javascript pueden ser incorporados a una página HTML de distintas maneras.

Una primera forma es definiendo el programa Javascript dentro de la página HTML. La segunda es definiendo un archivo separado con extensión .js e invocándolo en la página.

3.1 Definición de un programa Javascript “inline”.

A continuación una página web que hace uso de un programa o sentencia Javascript “inline” o empotrado.

```

1 <html>
2 <body>
3     <h2>Mi primer programa</h2>
4
5     <button onclick="alert('hola mundo');">De click en este boton </button>
6
7 </body>
8 </html>

```

Figura 3.1. Ejemplo de Javascript “inline”.

La página generada es la siguiente:



Figura 3.2. Ejemplo de Javascript “inline”.

Una vez se da click sobre el botón aparece lo siguiente:

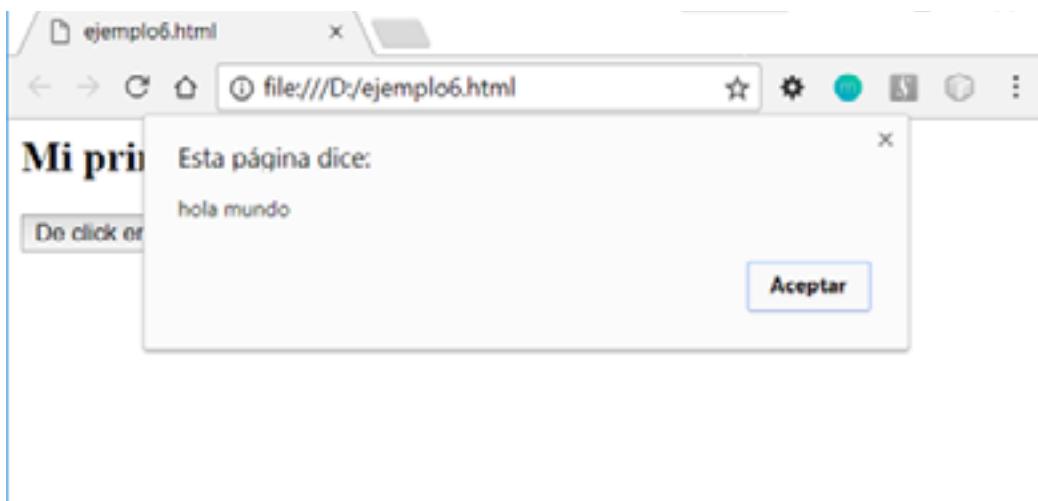


Figura 3.3. Página HTML con programa Javascript empotrado.

3.2 Definición de un programa Javascript en una sección de la página web.

El mismo ejemplo del numeral 3.1. se puede realizar con un programa Javascript definido dentro de la misma página HTML como se muestra a continuación:

```

1 <html>
2 <body>
3     <h2>Mi primer programa</h2>
4     <button onclick="miFuncion();"> De click en este boton </button>
5 </body>
6
7 <script>
8 function miFuncion(){
9     alert('hola mundo');
10 }
11 </script>
12
13 </html>

```

Figura 3.4. Página HTML con programa Javascript en una sección.

En el anterior ejemplo se usó el par de etiquetas `<script></script>` para definir una función Javascript.

3.3. Definición de un programa Javascript en un archivo separado.

En este ejemplo se creó un archivo separado llamado: “programa.js” el cual se invoca en la página HTML en la sección `<head>` usando la expresión `<script src='programa.js'></script>`.

```

1 <html>
2 <head>
3     <script src='programa.js'></script>
4 </head>
5 <body>
6     <h2>Mi primer programa</h2>
7     <button onclick="miFuncion();"> De click en este boton </button>
8 </body>
9
10 </html>

```

Figura 3.5. Página HTML con programa Javascript invocado en la sección <head>

```

1 function miFuncion(){
2     alert('hola mundo');
3 }

```

Figura 3.6. Programa Javascript

3.4 Introducción a la programación en Javascript.

En el numeral anterior se enunciaron las distintas formas como se puede integrar un código Javascript en una página HTML.

Aplicando los patrones de diseño vistos a lo largo del curso, en especial el patrón MVC, se puede decir que la mejor forma para que las tres tecnologías interactúen es a través de archivos de código separado. De esta forma un sitio web contiene:

- a) Archivos .html para las páginas web.
- b) Archivos .css para los estilos.
- c) Archivos .js para los programas.

La interactividad que Javascript provee a los sitios web se puede agrupar en los siguientes tipos:

- a) Implementación de un modelo de eventos.
- b) Manipulación del DOM
- c) AJAX.

3.4.1. Implementación de un modelo de eventos.

Un evento es algo que ocurre durante la ejecución de un programa Javascript. Por ejemplo cuando se presiona el teclado se activa el evento “onKeyDown”, cuando se da click en el ratón se activa el evento “onClick”, y así sucesivamente.

Si se quiere ejecutar un programa Javascript cuando se oprime click sobre un botón se puede usar el siguiente código:

```
1 <button onclick="miFuncion()">Oprima este botón</button>
2
3
```

Los eventos también se pueden definir dinámicamente utilizando la función “addEventListener()” mediante la siguiente forma:

```
elemento.addEventListener('evento', función, boolean)
```

Donde:

- | | |
|----------|---|
| Elemento | : Es la referencia a cualquier elemento HTML |
| Evento | : Nombre del evento que se desea controlar |
| Función | : Nombre de la función encargada de gestionar el evento |
| Boolean | : Recibe el valor True o False dependiendo si se desea emplear el evento anidado con otros eventos o no. Generalmente este parámetro es False |

Ejemplo:

```
document.querySelector("#miTitulo").addEventListener('click',procesarClick,false);
```

Define que la función procesarClick se debe ejecutar al hacer click sobre el elemento HTML identificado con el id “miTitulo”. La función “procesarClick()” debe ser estar definida en el archivo HTML o en un archivo .js separado.

3.4.2. Manipulación del DOM.

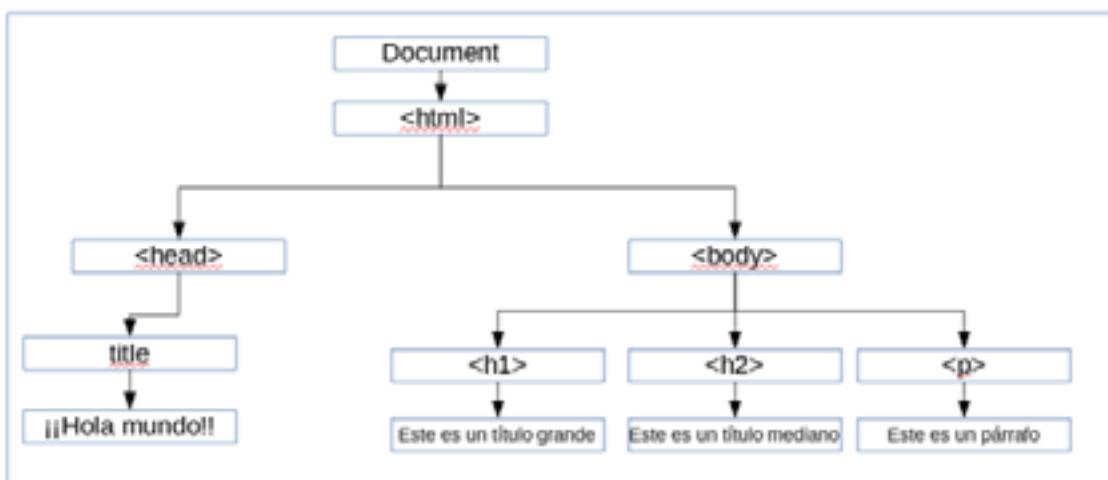
El DOM es un API de programación para documentos. Una vez el navegador procesa un archivo en lenguaje HTML crea una estructura en memoria llamada el objeto Documento. Ese objeto documento tiene estructura de árbol y su elemento o nodo raíz es la etiqueta <HTML>. A partir de ahí se van anexando los demás elementos del lenguaje HTML como <head>, <body>, <h1>, <p>,
, <table>, <form>, etc.

A continuación un ejemplo de una página web y su correspondiente DOM.

```

1 <html>
2   <head>
3     <title>; Hola mundo !!</title>
4   </head>
5   <body>
6     <h1> Titulo Grande </h1>
7     <h2> Titulo Mediano </h2>
8     <p> Este es un párrafo </p>
9   </body>
10 </html>
```

3.7. Página HTML de ejemplo.



3.8. Ejemplo de HTML DOM.

Javascript permite acceder, agregar, modificar o borrar elementos o nodos de este árbol a través del objeto document.

Selector	Elemento	Ejemplo
getElementsByName	Todos los elementos de una determinada etiqueta	<pre>getElementsByName('p')</pre> <p>Retorna un arreglo con todos los elementos existentes en la página Web.</p> <pre>getElementsByName('p')[0]</pre> <p>Accede al primer elemento tipo P de la página Web.</p>
getElementById	Un elemento en particular identificado con un determinado ID	<pre>getElementById('miId')</pre> <p>Accede al elemento de la página Web cuyo ID es miID</p>
getElementsByClassName	Uno o varios elementos con un className determinado	<pre>getElementsByClassName('miClase')</pre> <p>Accede a los elementos cuyo className sea miClase</p>
querySelector	Un elemento mediante la sintaxis de selección de CSS	<pre>querySelector('p:first-child')</pre> <p>Accede al primer elemento P de la página Web</p>
querySelectorAll	Una lista de elementos que coincida con el patrón de búsqueda CSS	<pre>querySelectorAll('#principal p')</pre> <p>Retorna un arreglo con todos los elementos P que sean hijos de principal</p>

Para acceder a los elementos del DOM Javascript suministra las siguientes funciones:

Para crear elementos o nodos se siguen los siguientes pasos:

- Crear el nodo con el comando: var elemento = document.createElement(nombre_del_elemento) .
- Crear los atributos del elemento creado e integrarlos.
- Integrar el elemento creado al nodo requerido.

A continuación un ejemplo:

```

1 <html>
2   <body>
3     <div id="div1">
4       <p id="p1">Este es un párrafo</p>
5       <p id="p2">Este es otro párrafo</p>
6     </div>
7
8     <script>
9       var parrafo = document.createElement("p");
10      var nodo = document.createTextNode("Este es un párrafo nuevo");
11      parrafo.appendChild(nodo);
12
13      var element = document.getElementById("div1");
14      element.appendChild(parrafo);
15
16     </script>
17   </body>
18 </html>
```

3.9. Ejemplo de creación de elemento HTML con Javascript.

3.4.3. AJAX.

Los formularios web y en general las páginas web fueron concebidas para recargarse cada vez que hay un cambio en su estructura. Esto genera interrupciones que desmejoran la experiencia al usuario. Para resolver se desarrolló una tecnología basada HTML, CSS y Javascript que permite realizar consultas a servidores sin tener que recargar completamente la página.

Esta tecnología recibe el nombre de “Asynchronous Javascript and XML” o AJAX y está basada en el objeto “XMLHttpRequest”. Este recurso no aborda esta tecnología por ser un tema avanzado.

3.5 Ejemplo de utilización de HTML, CSS y Javascript.

Se pueden añadir eventos a los elementos que conforman el menú del ejemplo desarrollado en este recurso. Esto se hace a través del lenguaje Javascript y en este caso se usará un archivo separado que será vinculado en la sección <head> del ejemplo.

A continuación el programa Javascript que asocia a cada opción del menú un bloque de código que se ejecuta al oprimir el botón derecho del ratón. Este programa se llamará “miPrograma.js”.

```
1 window.addEventListener("load",asignarEventos,false);
2 function asignarEventos()
3 {
4     document.getElementById("menu1").addEventListener("click",click1,false);
5     document.getElementById("menu2").addEventListener("click",click2,false);
6     document.getElementById("menu3").addEventListener("click",click3,false);
7 }
8 function click1()
9 {
10     window.alert("Hizo click en el menu 1");
11 }
12 function click2()
13 {
14     window.alert("Hizo click en el menu 2");
15 }
16 function click3()
17 {
18     window.alert("Hizo click en el menu 3");
19 }
```

3.10. Programa Javascript que asigna eventos a elementos HTML.

La siguiente es la descripción del código fuente:

Línea	Descripción
1	Al cargar la página (evento load) llama a la función asignarEventos
2..7	Accede a los elementos de la página web identificados como "menu1", "menu2" y "menu3" y le asigna al evento "click" la ejecución de una determinada función (click1, click2 o click3)
8..19	Funciones que se ejecutan al hacer clic sobre alguno de los menús y que presentan un respectivo mensaje

En el código de la página web se agrega el archivo en la sección <head> así:

```

3   <head>
4     <meta charset="iso-8859-1">
5     <meta name="description" content="Ejemplo de Maquetacion en HTML">
6     <link rel="stylesheet" href="miEstilo2.css">
7     <title>Este texto es el titulo del documento</title>
8     <script src="miPrograma.js"></script>
9   </head>

```

3.11. Ejemplo de inclusión de programa Javascript en la sección <head>

4. Librerías y frameworks Javascript.

El auge de Internet y la demanda por aplicaciones web cada vez más interactivas ha llevado a que los sitios web contengan cada vez más líneas de código Javascript.

Para facilitar la labor de los programadores se han creado dos tipos de herramientas: las librerías y los frameworks.

Las librerías como su nombre lo indica son bibliotecas de código fuente para una gran variedad de tareas comunes que van desde la validación de direcciones de correo, validación de rangos, etc. Hasta la manipulación del DOM y la consulta a otros sitios web a través de AJAX.

Los frameworks por otra parte también están conformados por librerías pero a diferencia de estas últimas que cobijan un tema particular cada framework puede abarcar un amplio espectro de temas que incluyen: validación de formularios, conexión a bases de datos, enruteamiento, entre otros.

En resumen un framework es más amplio y abarca más funciones que una librería

4.1. Librería Jquery

Es una librería de amplio uso que permite simplificar el código Javascript al realizar lo siguiente:

- a) Manipulación de HTML, DOM y CSS.
- b) Manipulación de eventos en HTML
- c) Animaciones
- d) Consultas HTTP asíncronas (AJAX)

La librería Jquery se invoca en la sección Head de la página web de la siguiente manera:

```
1 <!doctype html>
2 <html>
3   <head>
4     <meta charset="utf-8">
5     <title>Demo</title>
6     <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>
7 </head>
```

4.1. Ejemplo de inclusión de la librería Jquery en una página HTML.

En el ejemplo la librería se descarga desde el CDN (Content Delivery Network) de Google cada vez que se carga la página. También se puede descargar el archivo "jquery.min.js" e invocarlo desde la misma sección Head. En este caso la línea de invocación sería: "<script src="jquery.min.js"></script>".

Una vez cargada la librería se puede comenzar a utilizar. El esquema de Jquery es el siguiente:

```
$(selector).accion()
```

Donde:

El símbolo \$ se usa para indicar el inicio del llamado a la librería Jquery.

El selector indica el elemento HTML al cual aplicará la regla. Son los mismos selectores que en CSS.

La acción será el código que aplicará al elemento.

4.2 Ejemplo de documento HTML que usa la librería Jquery.

En el ejemplo de la figura 3.13 se tiene la siguiente estructura:

El selector es “document” es decir la ventana del navegador actual.

La acción en este caso es el evento “ready” que indica que la página ya fue cargada.

Una vez cargada la página se ejecuta la línea “alert('hola mundo')” que muestra un ventana emergente con el mensaje.

4.2 Framework AngularJS

AngularJS es un framework basado en el lenguaje Javascript desarrollado por Google con amplio uso que provee un conjunto de herramientas que realizan lo siguiente:

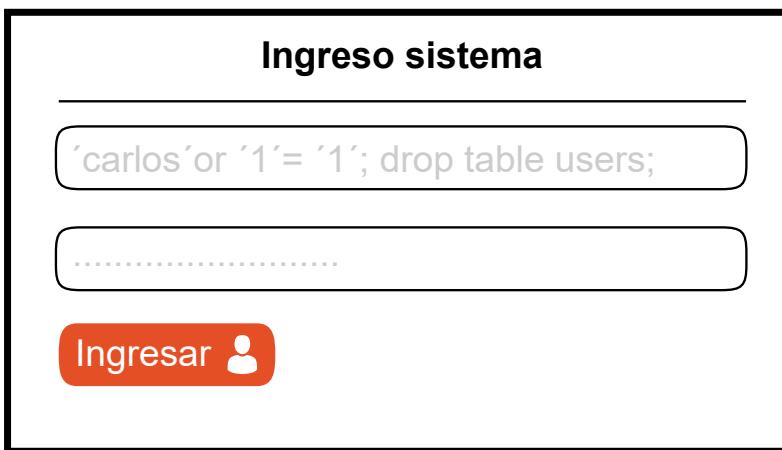
- a) **Data Binding (anclaje de datos):** esta característica permite asignar un campo de base de datos a un elemento HTML, generalmente un campo de un formulario.
- b) **Enrutamiento:** Angular integra elementos para el manejo de rutas de los sitios web.
- c) **Validación de formularios:** Angular permite realizar validación a los formularios con instrucciones ya definidas en el framework.
- d) **Integración a bases de datos:** Angular provee herramientas para conectarse a bases de datos de manera directa.

5. Aspectos de seguridad en páginas web.

Debido al auge de los servicios ofrecidos a través de Internet como son los servicios bancarios, sanitarios, educativos, etc, en los cuales se realizan transacciones monetarias han aparecido softwares maliciosos que buscan entre otras cosas robar contraseñas, robar datos personales, acceder a sitios sin autorización, entre otros.

Es por esta razón que la seguridad de un sitio web es importante y debe dar confianza al usuario del mismo.

El ataque más común se llama “SQL Injection” que consiste en introducir comandos sql a través de los cajones de texto o “inputs”. Un ejemplo típico es el siguiente:



The screenshot shows a web-based login interface. At the top, it says "Ingreso sistema". Below that is a text input field containing the SQL command: "'carlos' or '1' = '1'; drop table users;". Below the input field is a red button labeled "Ingresar" with a user icon. The entire interface is enclosed in a black rectangular border.

Figura 5.1. Ejemplo de inyección de código SQL.

En la figura 3.14 se está intentando vulnerar la seguridad para ingresar al sistema usando la sentencia “or ‘1’ = ‘1’ ”. Cuando este código llegue a la base de datos el sistema dejará entrar cualquier usuario ya que la condición ‘1’ = ‘1’ siempre será verdadera.

Para evitar la inyección de código malicioso como este se recomienda no enviar las sentencias SQL tal como las introduce el usuario sino que usar una opción llamada “comandos preparados” (PreparedStaments) presente en los lenguajes de programación o querys parametrizados.

Glosario

AJAX: acrónimo de Asynchronous Javascript and XML. Tecnología que permite a las páginas web actualizar los datos sin tener que recargar la página.

Atributo: característica o propiedad de un elemento HTML que puede ser definida por el programador.

Browser: navegador o visualizador de páginas web.

Cookies: pequeños archivos con información que son enviados por los servidores de internet y son almacenados en los computadores de los usuarios.

CSS: acrónimo de Cascade-Style Sheet. Hojas de estilos en cascada. Tecnología que permite personalizar la presentación de las páginas web.

Diseño adaptativo: estrategia para abordar el diseño web para distintos tamaños de pantalla al mismo tiempo.

DOM: acrónimo de Document Object Model. Modelo de objeto para documentos que permite ver un documento como un conjunto de nodos en forma de árbol.

Etiqueta: elementos que en su conjunto conforman una página web.

Framework: conjunto de programas y librerías que apoyan varias aspectos de la programación.

Herencia: en CSS es la cualidad que tienen los nodos hijos de tener los mismos atributos que sus padres.

HTML: acrónimo de Hypertext Markup Language. Lenguaje de marcación de hipertexto.

Javascript: lenguaje de programación empotrado dentro de los navegadores web.

Librería: conjunto de programas afines que apoyan un aspecto de la programación.

Layout: maqueta o prototipo de un diseño web.

Selector: parte de una regla CSS que identifica los elementos HTML a los cuales aplicará.

WWW: acrónimo de World Wide Web o Red Mundial que se comparte a través de la Internet.

XML: acrónimo de eXtensible Markup Language. Lenguaje de marcación extensible.

Bibliografía

Niederst, J. (2012). *Learning Web Design*. Cambridge. O'Reilly, Fourth Edition.

W3C The World Wide Web Consortium. (2016). HTML 5.1 *W3C Recommendation*. Recuperado de <https://www.w3.org/TR/html>

W3C The World Wide Web Consortium. (2017). *CSS Snapshot 2017*. Recuperado de <https://www.w3.org/TR/CSS>

Control del documento

**CONSTRUCCIÓN
OBJETO DE
APRENDIZAJE**



INTRODUCCIÓN AL DISEÑO WEB - HTML

Centro Industrial de Mantenimiento Integral - CIMI
Regional Santander

Líder línea de producción: Santiago Lozada Garcés

Asesores pedagógicos: Rosa Elvia Quintero Guasca - V2
Claudia Milena Hernández Naranjo - V2

Líder expertos temáticos: Rita Rubiela Rincón Badillo

Experto temático V1: Andrés Julián Valencia Osorio
Experto temático V2: Nelson Mauricio Silva Maldonado

Diseño multimedia: Jesús Antonio Vecino Valero

Programador: Francisco José Lizcano Reyes

Producción de audio: Victor Hugo Tabares Carreño



Este material puede ser distribuido, copiado y exhibido por terceros si se muestra en los créditos. No se puede obtener ningún beneficio comercial y las obras derivadas tienen que estar bajo los mismo términos de la licencia que el trabajo original.