**PROGRAMACIÓN WEB 2**

**Unidad 1: Introducción al lenguaje HTML**

**Lección 1: Estructura básica de una página Web.**

HTML, por su siglas de HyperText Markup Language (lenguaje de marcas de hipertexto), hace referencia al lenguaje de marcado para la elaboración de páginas web. Está compuesto por una serie de etiquetas rodeadas por corchetes angulares (<,>) que el navegador interpreta y da forma en la pantalla. HTML dispone de etiquetas para imágenes, hipervínculos que nos permiten dirigirnos a otras páginas, saltos de línea, listas, tablas, etc.

HTML también puede describir, hasta un cierto punto, la apariencia de un documento.

HTML es un lenguaje de programación, que nos permiten crear páginas web estáticas, es decir que su información no cambia, y la posibilidad de integrar otros lenguajes de programación como PHP, creando así páginas Web dinámicas, es decir, que su contenido puede cambiar dinámicamente.

La programación de HTML se puede hacer desde cualquier editor de texto y el resultado de su ejecución se puede ver desde cualquier navegador (Chrome, Mozilla, Opera, etc).



|  |  |
| --- | --- |
| Título del recurso educativo | Navegadores Web |
| Descripción: | 1 imagen de navegadores web |
| Unidad | 1 |
| Lección | 1 |
| Tipo de recurso: | Recurso de observación |
| Instrucciones para el estudiante |  |
| URLs de ejemplo o sugerencias | Ubicación: img\_2\_1 /navegadores.jpg |

Los documentos HTML se deben guardar con la extensión **htm**, **html** o **php** en el caso que incluya código de PHP. Ejemplo: **mi\_pagina.html**

**Estructura Básica** **de un documento HTML**

<!DOCTYPE html>

<html>

<head>

<title>Título de la Página</title>

</head>

<body>

Contenido u otras etiquetas

</body>

</html>

Como vemos aquí, todo el documento se encierra entre las etiquetas **<html>** y **</html>** e internamente contiene las etiquetas **<head> </head>** y **<body> </body>** las cuales traducen en español cabeza y cuerpo, respectivamente.

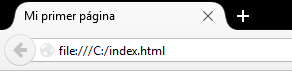
Entre las Etiquetas **<head> </head>**, pueden ir otras etiquetas, como por ejemplo **<title> </title>,** en esta se define el título que queremos que aparezca en la pestaña del navegador. Un ejemplo de esto, se muestra a continuación:

<head>

<title>Mi primer página</title>

</head>

Dando como resultado:



|  |  |
| --- | --- |
| Título del recurso educativo | Titulo pestaña |
| Descripción: | 1 imagen de título en pestaña |
| Unidad | 1 |
| Lección | 1 |
| Tipo de recurso: | Recurso de observación |
| Instrucciones para el estudiante |  |
| URLs de ejemplo o sugerencias | Ubicación: img\_2\_1 /titulo\_pestana.png |

Entre las etiquetas **<head> </head>** se puede especificar el autor de la página, una descripción, la codificación de caracteres etc, utilizando la etiqueta **<meta>** de la siguiente forma:

<head>

<meta name=*"author"* content=*"DAP"*>

<meta name=*"description"* content=*"Curso de HTML gratuito"*>

<meta charset=*"UTF8"* >

</head>

En nuestro caso debemos establecer la codificación de caracteres en UTF8, esta permitirá que en la página se reconozcan las tildes y otros caracteres propios del español.

Todo lo que va entre las etiquetas **<head>** y **</head>** no será visible en la página,a excepción del título que aparece en la pestaña con la etiqueta **<title> </title>**.

Todo lo que será visible irá entre las etiquetas **<body></body>**.

La etiqueta **<!DOCTYPE html>** indica el tipo de documento, y para este caso está indicando que es un documento html en su última versión, la cual corresponde a HTML5.

Los navegadores aun no soportan en su totalidad HTML5 dado que es una versión relativamente nueva, sin embargo poco a poco lo irán haciendo hasta soportarlos a un 100%.

**Lección 2: Títulos, Párrafos y listas en HTML.**

**Títulos:** HTML provee una serie de etiquetas para establecer títulos dentro de la página. Estas etiquetas son:<h1>, <h2>, <h3>, <h4>, <h5>, <h6>. Cada una con su respectivo cierre. Un texto establecido entre alguna de estas etiquetas se mostrará en negrita, con un espaciado arriba y abajo, y con un tamaño que depende de la etiqueta que se use, es decir, un texto entre las etiquetas **<h1>** y **</h1>** se verá más grande que untexto entre las etiquetas **<h6>** y **</h6>**.

Cabe anotar que estas etiquetas y todas las demás que veremos en esta unidad se escribirán dentro de las etiquetas **<body></body>**

Un ejemplo de del uso de estas etiquetas, se muestra a continuación:

<h1> Titulo de la página con h1 </h1>

<h2> Titulo de la página con h2 </h2>

<h3> Titulo de la página con h3 </h3>

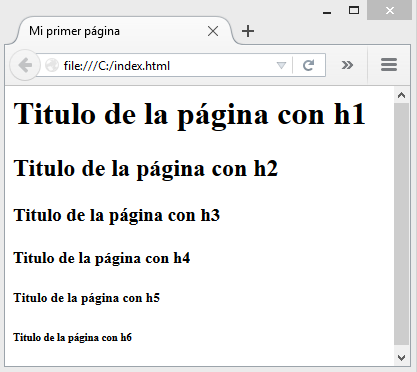
<h4> Titulo de la página con h4 </h4>

<h5> Titulo de la página con h5 </h5>

<h6> Titulo de la página con h6 </h6>

Dando como resultado:

|  |  |
| --- | --- |
| Título del recurso educativo | Títulos de página |
| Descripción: | 1 imagen de títulos de página |
| Unidad | 1 |
| Lección | 2 |
| Tipo de recurso: | Recurso de observación |
| Instrucciones para el estudiante |  |
| URLs de ejemplo o sugerencias | Ubicación: img\_2\_1 /titulos\_pagina.png |



**Párrafos:** Para los párrafos HTML proporciona las etiqueta **<p>** y **</p>.** Cada párrafo de la página se debe encerrar entre este par de etiquetas y entre cada párrafo se mostrará un espaciado.

Un ejemplo de del uso de estas etiquetas, se muestra a continuación:

<p>

HTML, por su siglas de HyperText Markup Language (lenguaje de marcas de hipertexto), hace referencia al lenguaje de marcado para la elaboración de páginas web.

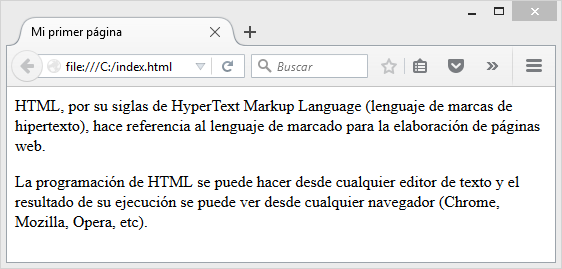
</p>

<p>

La programación de HTML se puede hacer desde cualquier editor de texto y el resultado de su ejecución se puede ver desde cualquier navegador (Chrome, Mozilla, Opera, etc).

</p>

Dando como resultado:



|  |  |
| --- | --- |
| Título del recurso educativo | Párrafos |
| Descripción: | 1 imagen de párrafos en HTML |
| Unidad | 1 |
| Lección | 2 |
| Tipo de recurso: | Recurso de observación |
| Instrucciones para el estudiante |  |
| URLs de ejemplo o sugerencias | Ubicación: img\_2\_1 /parrafos.png |

**Listas:** En HTML las listas son similares a las viñetas que colocamos a un texto. A continuación veremos 2 tipos de listas, las listas ordenadas y las no ordenadas.

**Listas no ordenadas:** Las listas no ordenadas van dentro de la etiqueta **<ul>** y **</ul>**. Cada ítem que queramos añadir a la lista, lo haremos dentro de la etiqueta **<li>** y su cierre **</li>** de la siguiente manera:

<ul>

<li type="circle">Esta es una viñeta tipo circle.</li>

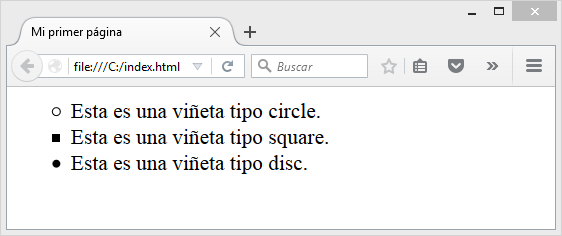
<li type="square">Esta es una viñeta tipo square.</li>

<li type="disc"> Esta es una viñeta tipo disc.</li>

</ul>

Las etiquetas HTML pueden tener propiedades, en este caso vemos que la etiqueta **<li>** tiene una propiedad **type** en la cual determinamos el tipo de viñeta que queremos utilizar, las cuales pueden ser circle, square o disc.

Lo anterior daría como resultado la siguiente lista:



|  |  |
| --- | --- |
| Título del recurso educativo | Listas no ordenadas |
| Descripción: | 1 imagen de listas no ordenadas en HTML |
| Unidad | 1 |
| Lección | 2 |
| Tipo de recurso: | Recurso de observación |
| Instrucciones para el estudiante |  |
| URLs de ejemplo o sugerencias | Ubicación: img\_2\_1 /listas\_no\_ordenadas.png |

**Listas ordenadas**: Las listas ordenadas van enmarcadas dentro de las etiquetas **<ol>** y **</ol>**. Cada ítem de la lista se escribe con la misma etiqueta que en las no ordenadas: **<li>**. Pero al ser listas ordenadas las viñetas serán números y éstos se irán generando automáticamente por orden, conforme agreguemos nuevos ítems.

<ol>

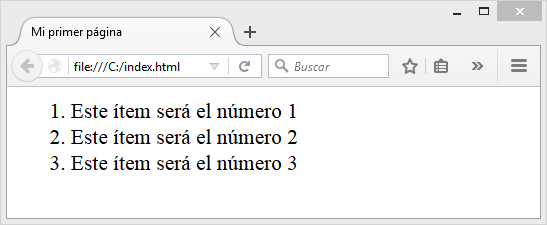
<li>Este ítem será el número 1 </li>

<li>Este ítem será el número 2</li>

<li>Este ítem será el número 3</li>

</ol>

Lo anterior daría como resultado la siguiente lista:



|  |  |
| --- | --- |
| Título del recurso educativo | Listas ordenadas |
| Descripción: | 1 imagen de listas ordenadas en HTML |
| Unidad | 1 |
| Lección | 2 |
| Tipo de recurso: | Recurso de observación |
| Instrucciones para el estudiante |  |
| URLs de ejemplo o sugerencias | Ubicación: img\_2\_1 /listas\_ordenadas.png |

**Lección 3: Imágenes e hipervínculos.**

**Imágenes:** HTML nos proporciona la etiqueta **<img>** para insertar imágenes dentro de nuestra página web. A diferencia de las etiquetas que vimos anteriormente, esta no requiere de la etiqueta de cierre, pero si de una serie de propiedades o atributos, los cuales veremos a continuación.

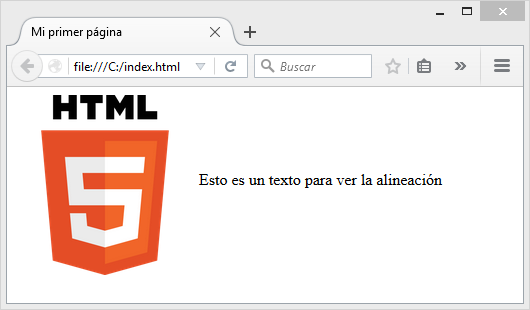
* **src:** Este atributo es el que indica la URL o ruta dónde se encuentra alojada la imagen que se quiere mostrar.
* **align:** Este atributo alinea la imagen, utilizando el valor "left" se alineará la izquierda, "right" para alinearla a la derecha, "top" para alinear la imagen con un texto en la parte superior, "bottom" para alinear la imagen con un texto en la parte inferior y "middle" para alinear la imagen con un texto al centro.
* **title**: Con este atributo se puede colocar una descripción a la imagen. Esta se mostrará al pasar el ratón por encima.
* **height,**  **width**: El atributo height marca la altura de la imagen, mientras que width marca el ancho de esta.
* **alt:** Con este atributo se puede especificar un texto alternativo para la imagen.

Un ejemplo del uso de esta etiqueta se muestra a continuación:

<img src="logo.jpg" width="256" height="256" alt="mi logo" align="middle">

Dando como resultado:

|  |  |
| --- | --- |
| Título del recurso educativo | Imágenes |
| Descripción: | 1 imagen de inserción de imagen en HTML |
| Unidad | 1 |
| Lección | 3 |
| Tipo de recurso: | Recurso de observación |
| Instrucciones para el estudiante |  |
| URLs de ejemplo o sugerencias | Ubicación: img\_2\_1 /imagenes.png |



**Hipervínculos:** Un hipervínculo es un enlace que permite de una manera muy cómoda redirigir al usuario a otra página web o a otra parte dentro de la misma página o archivo, dando click sobre este.

Las etiquetas para especificar un enlace es **<a>** y **</a>**. El texto que se mostrará para el enlace se debe colocar entre estas etiquetas de apertura y cierre.

Los principales atributos o propiedades de los hipervínculos son:

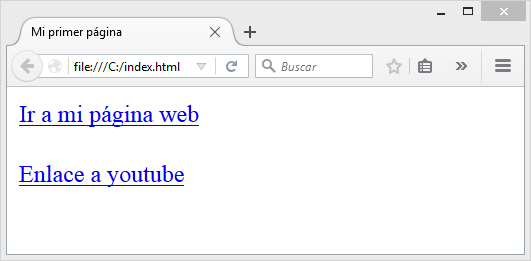
* **href:** Especifica la URL, ruta o dirección a la cual se quiere ir al hacer clic sobre el enlace.
* **target:** Específica donde se abre el documento enlazado. El target puede ser “\_blank” para indicar que el documento enlazado se abrirá en otra pestaña del navegador o ventana si el navegador no soporta pestañas, “\_self” para indicar que se abrirá en el mismo frame donde se encontraba el enlace, “\_parent” para indicar que se abrirá en un frame padre y “\_top” para indicar que se abrirá en un frame de tamaño completo de toda la página.

A continuación veremos algunos ejemplos del uso de los enlaces.

1. <a href="mipag.html" target="\_blank">Ir a mi página web</a>

2. <a href="http://www.youtube.com" target="\_blank">Enlace a youtube</a>

Dando como resultado:

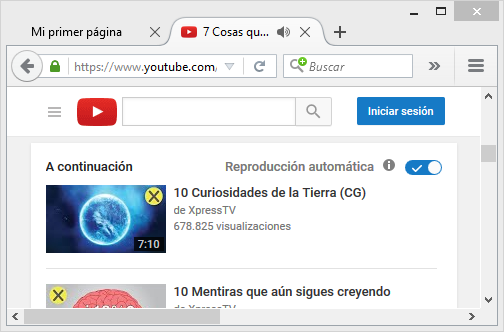


|  |  |
| --- | --- |
| Título del recurso educativo | hipervínculos |
| Descripción: | 1 imagen de inserción de hipervínculos en HTML |
| Unidad | 1 |
| Lección | 3 |
| Tipo de recurso: | Recurso de observación |
| Instrucciones para el estudiante |  |
| URLs de ejemplo o sugerencias | Ubicación: img\_2\_1 /hipervinculos.png |

Por defecto los enlaces se muestran en color azul y subrayado, esto nos ayuda a identificarlos fácilmente dentro de un documento.

En los 2 casos cuando demos click en el enlace, la nueva página se mostrará en una nueva pestaña y el enlace cambiará a un color morado indicando que el enlace ya ha sido visitado.

|  |  |
| --- | --- |
| Título del recurso educativo | Enlace a youtube |
| Descripción: | 1 imagen de un enlace a youtube |
| Unidad | 1 |
| Lección | 3 |
| Tipo de recurso: | Recurso de observación |
| Instrucciones para el estudiante |  |
| URLs de ejemplo o sugerencias | Ubicación: img\_2\_1 /enlace\_youtube.png |



También es posible establecer una imagen como enlace de la siguiente manera:

<a href="mipag.html" target="\_blank">

<img src="logo.jpg" width="256" height="256" alt="mi logo" align="middle">

</a>

Al hacer click sobre la imagen se redirigirá al enlace especificado.

**Lección 4: Creación de tablas en HTML.**

Una tabla en html viene marcada por las etiquetas **<table>** y **</table>**. Entre esas dos etiquetas se definen las filas y columnas de la tabla y las características de cada uno de estos elementos.

**Las filas <tr>:** Las filas se escriben gracias a las etiquetas **<tr>** con su correspondiente cierre **</tr>**. El contenido de las columnas que están dentro de la fila lo podemos alinear tanto horizontal como verticalmente.

**Las celdas <td>:** Las celdas que van dentro de cada fila se escriben con la etiqueta **<td>** y su correspondiente cierre **</td>**.

**Las celdas <th>:** Las celdas escritas con la etiqueta **<th>** y su correspondiente cierre **</th>**, admiten los mismos atributos que las etiquetas **<td>** y funcionan de la misma forma, salvo que el contenido que esté dentro de una etiqueta **<th>** está considerado como el encabezado de la tabla, por lo que se creará en negrita y centrado sin necesidad de indicarlo.

La estructura básica para crear una tabla en HTML es la siguiente:

<table>

<tr>

<th>Título 1</th><th>Título 2</th>

</tr>

    <tr>

<td>contenido 1</td><td>contenido 2</td>

</tr>

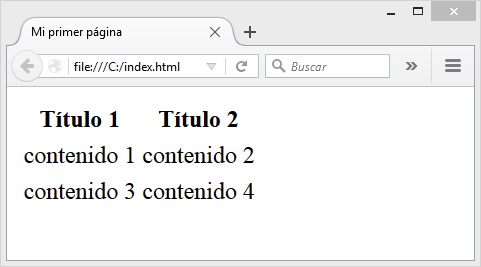
<tr>

<td>contenido 3</td><td>contenido 4</td>

</tr>

</table>

Lo cual nos dará como resultado lo siguiente:



|  |  |
| --- | --- |
| Título del recurso educativo | Estructura de una tabla |
| Descripción: | 1 imagen de una tabla simple en HTML |
| Unidad | 1 |
| Lección | 4 |
| Tipo de recurso: | Recurso de observación |
| Instrucciones para el estudiante |  |
| URLs de ejemplo o sugerencias | Ubicación: img\_2\_1 /tabla\_1.png |

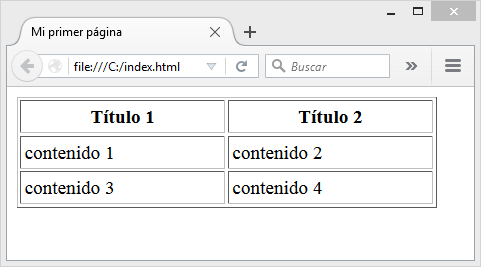
Aunque la información ya está contenida dentro de una tabla, aún nos falta darle algunas propiedades a la tabla para mejorar su presentación.

**Atributos principales de la etiqueta** **<table>**

* **border:** Especifica si la tabla llevará o no borde (1 ó “”).
* **cellpadding:** Especifica el espacio entre el borde de la celda y su contenido.
* **cellspacing:** Especifica el espacio entre celdas.
* **width:** Especifica el ancho de la tabla.

Ejemplo: <table border="1" cellpadding="3" cellspacing="2" width="400" >

Si agregamos estas propiedades a la tabla, esta se vería de la siguiente manera:



|  |  |
| --- | --- |
| Título del recurso educativo | Propiedades de una tabla |
| Descripción: | 1 imagen de propiedades de una tabla en HTML |
| Unidad | 1 |
| Lección | 4 |
| Tipo de recurso: | Recurso de observación |
| Instrucciones para el estudiante |  |
| URLs de ejemplo o sugerencias | Ubicación: img\_2\_1 /tabla\_2.png |

**Atributos principales de la etiqueta** **<tr>**

* **align:** Especifica la alineación del contenido de la fila, el cual puede ser “left” (izquierda), “right” (derecha) o “center” (centrado).
* **valign:** Especifica la alineación vertical del contenido de la fila, la cual puede ser “top” (arriba), “middle” (en el medio) o “bottom” (abajo).
* **bgcolor:** Especifica el color de fondo de la fila. Ej: “#ff0000”.

Ejemplo: <tr align="center" valign="middle" bgcolor="#FF6600">

Si agregamos estas propiedades a la segunda fila de la tabla, esta se vería de la siguiente manera:



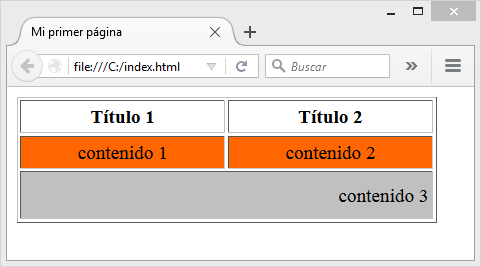
|  |  |
| --- | --- |
| Título del recurso educativo | Propiedades de fila |
| Descripción: | 1 imagen de propiedades de fila de una tabla HTML |
| Unidad | 1 |
| Lección | 4 |
| Tipo de recurso: | Recurso de observación |
| Instrucciones para el estudiante |  |
| URLs de ejemplo o sugerencias | Ubicación: img\_2\_1 /tabla\_3.png |

**Atributos principales de la etiqueta <td>**

* **align:** Especifica la alineación del contenido de la celda (left, right, center).
* **valign:** Especifica la alineación vertical del contenido de la celda (top, middle, bottom).
* **bgcolor:** Especifica el color de fondo de la celda. Ej: “#ff0000”.
* **width:** Especifica el ancho de la celda.
* **height:** Especifica la altura de la celda.
* **colspan, rowspan:** especifica el número de columnas y filas que una celda debe combinar o fusionar.

Ejemplo: <td align="right" height="40" valign="middle" colspan="2" bgcolor="#C0C0C0" >

Si agregamos estas propiedades a la primera celda de la tercera fila de la tabla, esta se vería de la siguiente manera:



|  |  |
| --- | --- |
| Título del recurso educativo | Propiedades de celda |
| Descripción: | 1 imagen de propiedades de celda de tabla HTML |
| Unidad | 1 |
| Lección | 4 |
| Tipo de recurso: | Recurso de observación |
| Instrucciones para el estudiante |  |
| URLs de ejemplo o sugerencias | Ubicación: img\_2\_1 /tabla\_4.png |

El código completo de esta última tabla es el siguiente:

<table border="1" cellpadding="3" cellspacing="2" width="400" >

<tr>

<th>Título 1</th><th>Título 2</th>

</tr>

<tr align="center" valign="middle" bgcolor="#FF6600">

<td>contenido 1</td><td>contenido 2</td>

</tr>

<tr>

<td align="right" height="40" valign="middle" colspan="2" bgcolor="#C0C0C0" >

contenido 3

</td>

</tr>

</table>

Aquí vemos que hemos quitado la celda donde se encontraba el contenido 4, esto debido a que la celda del contenido 3 la hemos fusionado en 2 celdas con la propiedad **colspan**, por tal razón se quitó esa celda del código.

**PROGRAMACIÓN WEB 2**

**Unidad 2: HTML Avanzado**

**Lección 1. Creación de formularios.**

Los formularios en HTML nos ayudan a aumentar la interactividad de nuestra Web y nos permiten recibir información de los usuarios de nuestra página. Estos están compuestos por diferentes componentes para capturar información y botones.

La etiqueta de los formularios es **<form>** y su cierre **</form>**. Todo lo que vaya dentro de estas etiquetas, serán parte del formulario.

**Principales Atributos:**

* **action:** El atributo “action” indica el tipo de acción que va a realizar el formulario. Se puede pasar  la url o ubicación del archivo donde se enviará el formulario o una dirección de correo si se quiere enviar a este.
* **method:** Mediante este atributo le indicamos al formulario la forma en la que el formulario será enviado. Existen dos valores posibles: get y post.
* **target:** Mediante este atributo indicaremos la forma en la que se desplegará la ventana que procesará el formulario. Sus valores posibles son: \_blank (nueva), \_self (mismo frame), \_parent (frame padre), \_top(cuerpo).

Ejemplos de definición de formularios:

**1.**<form action="http://www.mipag.com" method="post" target="\_blank">

Elementos del formulario

</form>

**2.**<form action="pagina\_local.php" method="post" target="\_self">

Elementos del formulario

</form>

**3.** <form action="mailto:micorreo@correo.com" method="post" target="\_parent">

Elementos del formulario

</form>

**Elementos o componentes del formulario:**

**Elemento <input>:** Esta etiqueta establece un campo que puede ser de diferente tipo, donde un usuario puede ingresar datos.

**Principales Atributos de la etiqueta <input>**

* **type:** especifica el tipo de elemento a utilizar. Entre los cuales se tienen:

***- text:*** Es el campo más simple, generalmente se usa para preguntas abiertas.

***- radio:*** Utilizado en preguntas de opción múltiple y única respuesta.

**- *checkbox:*** Utilizados para preguntas de opción múltiple y múltiple respuestas.

***- password*:** Utilizado para capturar contraseñas

***- submit*:** Botón para el envío del formulario.

***- hidden:*** Campo oculto. Este no será visible en el formulario.

***- file:*** Campo para subir archivos al servidor.

* **value:** Especifica el valor inicial del elemento.
* **name:** especifica el nombre que tendrá el elemento
* **id:** Especifica el identificador que tendrá el elemento.
* **disabled:** Especifica si el campo va a estar deshabilitado.
* **readonly:** Especifica si el campo es de sólo lectura.
* **size:** Especifica el tamaño en caracteres de un campo text  o password.
* **checked:** Indica que el elemento radio o checkbox inicia marcado o seleccionado.

A continuación veamos un ejemplo de estos componentes de formulario.

<form action="http://www.mipag.com" method="post" target="\_blank">

Componente text:

<input type="text" name="pelicula" value="titanic" size="20" disabled="disabled"><br>

Componente radio:

<input type="radio" name="genero" id="idgenero" checked="checked" ><br>

Componente checkbox:

<input type="checkbox" name="hobbie" readonly="readonly" ><br>

Componente password:

<input type="password" name="clave" id="idclave"><br>

Componente file:

<input type="file" name="archivo" id="idarchivo"><br>

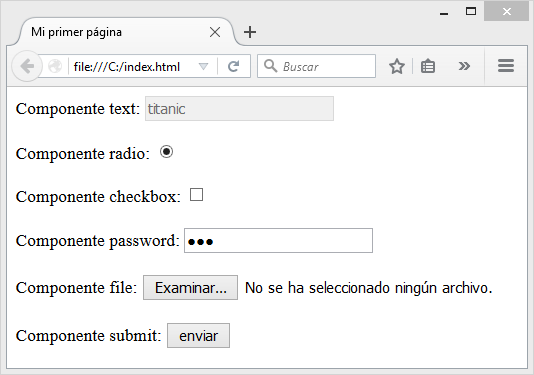
Componente submit:

<input type="submit" value="enviar" name="archivo" id="idarchivo">

</form>

**Nota:** La etiqueta **<br>** se utiliza en HTML paragenera un salto de línea.

El anterior formulario se vería de la siguiente manera:



formulario\_input.png

**Elemento <select>:** Estos elementos son los que se conocen comúnmente como listas desplegables o lista de opciones, donde el usuario podrá escoger una o múltiples opciones de acuerdo a como se haya establecido.

Cada una de las opciones de la lista vendrán entre las etiquetas **<option>** y **</option>** y estas a su vez entre las etiquetas **<select>** y **</select>** de la siguiente manera:

<select name="transporte">

<option value="1">Coche</option>

<option value="2">Avión</option>

<option value="3">tren</option>

</select>

**Atributos de la etiqueta <select>**

* **name:** Nombre del campo de selección.
* **id:** Identificador del campo de selección.
* **size:** Número de opciones visibles en la lista.
* **multiple:** Indica que se pueden seleccionar múltiples opciones en la lista.
* **disabled:** Indica que el elemento estará deshabilitado.

Ejemplo:

<select name="genero" id="idgenero" size="3" multiple="multiple" disabled="disabled">

**Atributos de la etiqueta <option>**

* **disabled:** Indica que la opción va a estar deshabilitada.
* **selected:** Indica que la opción va a estar seleccionada inicialmente
* **value:** El valor real que tendrá la opción.

**Ej:** <option value="F" selected="selected" disabled="disabled">Femenino</option>

A continuación veamos un ejemplo completo de un formulario con un componente **<select>**.

<form action="http://www.mipag.com" method="post" target="\_blank">

Componentes select:

<select name="transporte" size="2" multiple="multiple">

<option value="0">-Seleccione opción-</option>

<option value="1">Coche</option>

<option value="2" selected="selected">Avión</option>

<option value="3">tren</option>

</select>

<br><br>

Componentes select:

<select name="ingreso" id="idingreso" >

<option value="0">-Seleccione opción-</option>

<option value="2014">Año 2014</option>

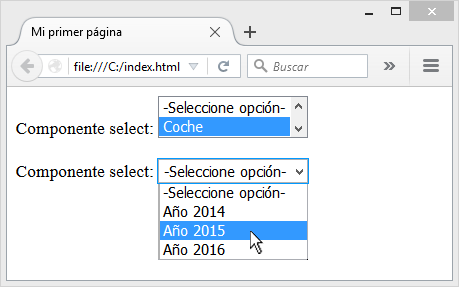
<option value="2015">Año 2015</option>

<option value="2016">Año 2016</option>

</select>

</form>

Este formulario se vería de la siguiente manera:



formulario\_select.png

**Elemento <textarea>:** Esta etiqueta permite crear un campo de texto de múltiples líneas, ideal para capturar textos extensos; por ejemplo: una observación, un resumen, una sugerencia.

El texto a ingresar estará entre las etiquetas de apertura y cierre **<textarea>** y **</textarea>**.

**Atributos de la etiqueta <textarea>**

* **cols:** Ancho del elemento basado en número de columnas a mostrar.
* **rows:** Altura del elemento basado en el número de filas a mostrar.
* **name:** Nombre del textarea.
* **id:** identificador del textarea.
* **disabled:** Indica que el elemento estará deshabilitado
* **readonly:** Indica que el elemento será de sólo lectura.

Ejemplo:

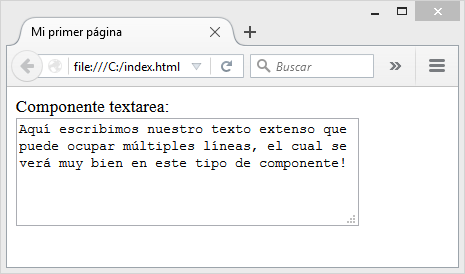
Componente textarea:

<textarea name="observacion" cols="100" rows="10">

Aquí escribimos nuestro texto extenso que puede ocupar múltiples líneas, el cual se verá muy bien en este tipo de componente!

</textarea>

El componente **<textarea>** se vería en el formulario de la siguiente manera:



form\_textarea.png

**Lección 2. Inserción de Audio y Video.**

HTML5 nos permite realizar la inserción de audio y video de una forma muy sencilla, lo cual no era posible en versiones anteriores de HTML.

**Inserción de Audio:** la inserción de audio, se realiza con las etiquetas **<audio>** y **</audio>**. Anteriormente para insertar audio en una página se utilizaba normalmente una tecnología llamada Flash y adicionalmente se tenía que instalar un componente llamado plugin en cada navegador, lo cual era una labor en ocasiones tediosa.

Los formatos de audio soportados son: MP3, WAV y Ogg.

**Atributos principales de la etiqueta <audio>:**

* **autoplay:** Indica que el audio se reproduce automáticamente cuando está listo.
* **controls:** Indica que se muestra controles, como los botones play, pause, etc.
* **loop:** Indica que el audio se reproduce de nuevo cada vez que termina.
* **muted:** Indica que el audio estará silenciado.
* **preload:** Indica si el audio carga previamente al tiempo que la página ha cargado. Los posibles valores de este atributo son “auto” o “none”.
* **src:** Indica la ubicación del archivo de audio a reproducir.

Ejemplo:

<audio src="miaudio.mp3" autoplay controls loop >

Audio no soportado

</audio>

En el caso que el navegador no soporte el componente de audio, en vez de este, se le mostrará en pantalla el texto “Audio no soportado” que hemos puesto adentro.

Adicionalmente si se quiere agregar más de una fuente de audio se utiliza la etiqueta **<source>** como se muestra a continuación:

<audio controls >

<source src="miaudio.ogg" type="audio/ogg">

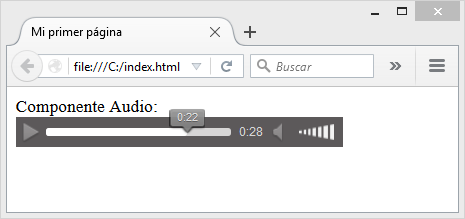
<source src="miaudio.mp3" type="audio/mpeg">

Tu navegador no soporta la etiqueta audio.

</audio>

Agregar múltiples fuentes es útil en el caso que la primer fuente no se encuentre por algún motivo, entonces tratará de buscar la siguiente fuente hasta encontrar alguna.

El componente audio se verá así en nuestra página:



audio.png

**Inserción de Video:** la inserción de video, se realiza con las etiquetas **<video>** y **</video>**. Al igual que el audio, esta funcionalidad no requiere de la instalación de plugins en el navegador para su funcionamiento.

Los formatos de video soportados son: MP4, WebM y Ogg.

**Atributos principales de la etiqueta <video>:**

* **autoplay:** Indica que el video se reproduce automáticamente cuando está listo.
* **controls:** Indica que se muestra controles, como los botones play, pause, etc.
* **loop:** Indica que el video se reproduce de nuevo cada vez que termina.
* **muted:** Indica que el video estará silenciado.
* **preload:** Indica si el video carga previamente al tiempo que la página ha cargado. Los posibles valores de este atributo son “auto” o “none”.
* **src:** Indica la ubicación del archivo de video a reproducir.
* **width:** Indica el ancho del video en pixeles.
* **height:** Indica la altura del video en pixeles.
* **poster:** indica la ubicación de una imagen que se mostrará mientras el video esta cargando.

Ejemplo:

<video src="movie.mp4" width="320" height="240" controls>

Video no soportado

</video>

Si se quiere agregar más de una fuente se utiliza la etiqueta <source> de la siguiente manera:

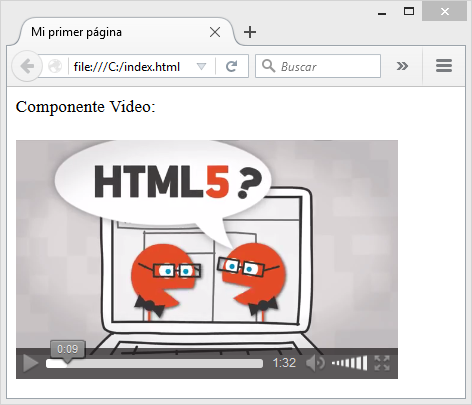
<video width="320" height="240" poster="miposter.jpg" controls>

 <source src="movie.mp4" type="video/mp4">

 <source src="movie.ogg" type="video/ogg" >

</video>

El componente de video se verá así en nuestra página:



video.png

**Lección 3. Canvas en HTML.**

El componente canvas es una especie de lienzo o contenedor que podemos incluir dentro de nuestra página, mediante las etiqueta **<canvas>** y **</canvas>**, en el cual podemos dibujar gráficos y realizar animaciones utilizando un lenguaje de programación web llamado JavaScript, el cual veremos con mayor detalle en otra unidad.

**Atributos principales de la etiqueta <canvas>:**

* **width:** Indica el ancho del canvas.
* **height:** Indica la altura del canvas.
* **Id:** identificador del canvas

El tamaño por defecto del canvas, si no se le proporciona un tamaño específico, es de 300px de ancho por 150px de alto.

Cuando se crea un elemento canvas, inicialmente este no tendrá contenido ni borde, ya que el contenido se le especificará con código Javascript y para el borde podemos utilizar estilos CSS los cuales veremos con mayor detalle en otras unidades.

Para el borde vamos a agregar la siguiente propiedad al canvas utilizando CSS:

style="border:1px solid #000000"

La propiedad **style** está estableciendo un borde de 1px de ancho, cuya línea es continua (solid) y cuyo color es negro (#000000).

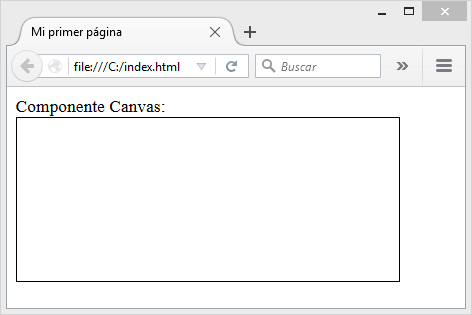
Ejemplo de inserción de un canvas:

<canvas id="miCanvas" width="350" style="border:1px solid #000000">

El navegador no soporta el componente canvas.  
</canvas>

En el caso que el navegador no soporte el componente canvas, en vez de este, se le mostrará en pantalla el texto “El navegador no soporta el componente canvas” que hemos puesto adentro.

El anterior ejemplo, se vería en nuestra página de la siguiente manera:



canvas.png

Ahora vamos a dibujar un simple gráfico dentro del canvas. En este caso dibujaremos un cuadrado utilizando código Javascript de la siguiente manera:

<script>

var c = document.getElementById("miCanvas");

var ctx = c.getContext("2d");

ctx.fillStyle = "#FF0000";

ctx.fillRect(50,20,150,75);

</script>

Para incluir código Javascript dentro de nuestro documento HTML , podemos utilizar las etiquetas **<script>** y **</script>**,lo cual le indicará al navegador que lo que se encuentre dentro de estas etiquetas los debe interpretar como código Javascript y no como código HTML.

La instrucción document.getElementById("miCanvas"); indica que vamos a obtener el canvas con identificador “miCanvas” para poder dibujar sobre este. Es por esto que es indispensable que cada canvas tenga asociado un identificador para poder referenciarlo.

La instrucción c.getContext("2d"); indica que el grafico que se va a dibujar será en 2 dimensiones.

La instrucción ctx.fillStyle = "#FF0000"; indica que el color de relleno del rectángulo será rojo (#FF0000).

La instrucción ctx.fillRect(50,20,150,75); indica que se va a dibujar un rectángulo empezando desde la coordenada 50, 20 y terminando en la coordenada 150,75 dentro del canvas.

Uniendo todo lo visto anteriormente, el resultado obtenido será el siguiente:



canvas2.png

Ahora vamos para dibujar un círculo. Para esto agregamos el siguiente código Javascript:

<script>

var c = document.getElementById("miCanvas");

var ctx = c.getContext("2d");

ctx.fillStyle = "#0000FF";

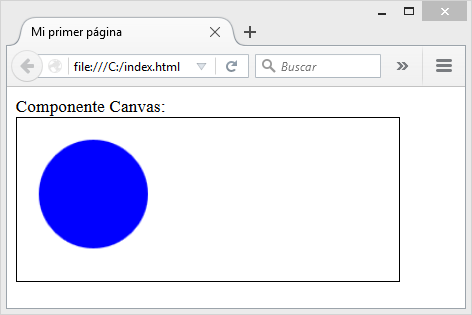
ctx.beginPath();

ctx.arc(70,70,50,0,2\*Math.PI);

ctx.fill();

</script>

Con el cual obtenemos el siguiente resultado:



canvas3.png

Dentro del canvas también podemos dibujar o insertar una imagen ya existente. Para lograr esto podemos utilizar el código que se muestra a continuación.

<html>

<head><title>Mi Página Web</title></head>

<body>

Canvas con imagen: <br>

<canvas id="miCanvas" style="border:1px solid #000000">

El navegador no soporta el componente canvas.  
</canvas>

<script>

var img = new Image();

img.src = "logo.png";

var c = document.getElementById("miCanvas");

var ctx = c.getContext("2d");

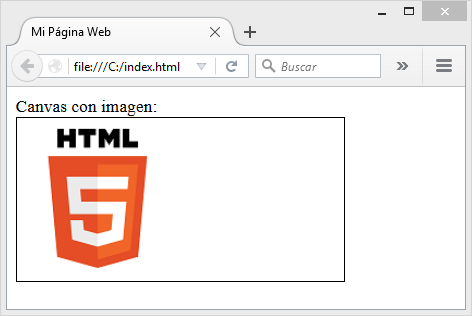
ctx.drawImage(img,10,10);

</script>

</body>

</html>

Dando como resultado:



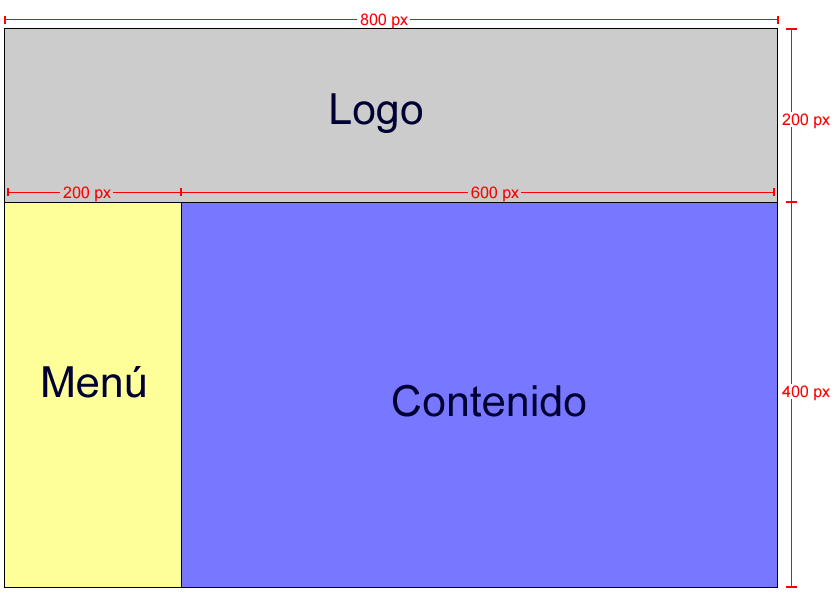
canvas4.png

**Lección 4. Maquetación de una página web.**

La maquetación es un aspecto muy importante en el diseño web, ya que por medio de esta le damos una organización adecuada a los elementos que contiene la página, de esta manera le damos un mejor aspecto a esta.

Para maquetar adecuadamente una página se debe tener en cuenta algunos aspectos y propiedades de los elementos HTML, así como también necesitaremos incluir un poco de estilos CSS.

Antes de empezar a maquetar o a distribuir nuestra página web debemos imaginarnos como queremos que esta se vea, o que secciones queremos que contenga, es decir, podemos imaginarnos una sección donde ira una cabecera con la imagen de la organización, una sección para el menú, una sección para el contenido general etc. Luego que hemos establecido esto debemos plasmarlo en un diagrama con las medidas de cada sección. A continuación veamos un ejemplo de una maquetación simple:



maqueta1.png

Cada uno de los recuadros en los que hemos dividido nuestra página, los vamos a llamar contenedores. En HTML tenemos diversas etiquetas que actúan como contenedores. Entre ellas tenemos: <div></div>, <nav></nav>, <article></article>, <section></section>, <header></header>, <footer></footer>, etc.

En la distribución que hemos realizado tenemos 3 contenedores. Para el que se encuentra en la parte superior vamos a utilizar <header> y </header> que significa cabecera; para el menú normalmente se utiliza <nav> y </nav> y para el contenido podemos utilizar <section></section>. Adicionalmente tendremos que agregar algunas propiedades CSS para establecer los tamaños de los contenedores.

<html>

<head><title>Mi Página Web</title></head>

<body>

<header style="width:800px; height:200px; background:#CCCCCC">

Aquí ira el logo

</header>

<nav style="width:200px; height:400px; float:left; background:#FFFF99 ">

Aquí ira el menú

</nav>

<section style="width:600px; height:400px; float:left; background:#7777FF">

Aquí ira el contenido

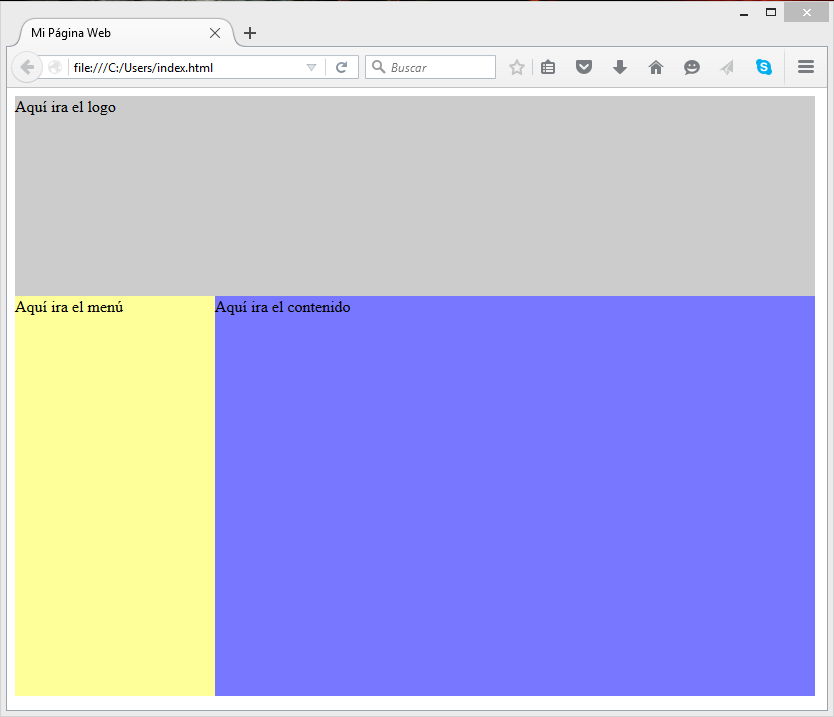
</section>

</body>

</html>

La propiedad **float** que hemos incluido, permite a un contenedor flotar o colocarse al lado de otro contenedor, ya que por defecto los contenedores se colocan uno debajo del otro y en este caso con esta propiedad de CSS los estamos obligando a flotar a la izquierda (left) cada uno.

El anterior código generará el siguiente resultado:



maqueta2.png

Hasta el momento hemos visto que siempre utilizamos pixeles (px) para establecer los tamaños, sin embargo en ocasiones conviene utilizar ona medida que no sea fija o absoluta, sino una medida que sea relativa. Esto hara que los contenedores se puedan adaptar a los diferentes tamaños de pantalla. Para esto utilizamos la medida en unidad porcentual (%).

<html>

<head><title>Mi Página Web</title></head>

<body>

<header style="width:100%; height:20%; background:#CCCCCC">

Aquí ira el logo

</header>

<nav style="width:30%; height:80%; float:left; background:#FFFF99 ">

Aquí ira el menú

</nav>

<section style="width:70%; height:80%; float:left; background:#7777FF">

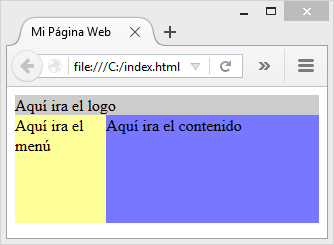
Aquí ira el contenido

</section>

</body>

</html>

En el caso anterior si ampliamos el tamaño del navegador, los contenedores se quedan del mismo tamaño, mientras que en este último caso cuando se amplía la ventana del navegador los contenedores se adaptan a estén.

maqueta3.png maqueta4.png

**PROGRAMACION WEB 2**

**Unidad 3: Estilos (CSS)**

**Lección 1. Introducción a CSS y sintaxis básica.**

CSS (Cascading Style Sheets o CSS): Es un lenguaje utilizado para especificar el aspecto visual o apariencia de una página web. Este lenguaje se integra con HTML de diferentes formas.

Una definición de un estilo en CSS tiene dos partes principales: un selector y una o más declaraciones.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Selector | Declaración 1 | | Declaración 2 | |  |
| **h1 {** | **color:** | **blue;** | **font-size:** | **12px;** | **}** |
|  | propiedad | valor | propiedad | valor |  |

Como vemos en este ejemplo, cada declaración está compuesta a su vez por una propiedad y su valor.

Normalmente los selectores son elementos HTML a los cuales se les quiere aplicar un estilo.

Entre los tipos de selectores, tenemos los siguientes:

**Selector universal ( \* ):** Se utiliza para seleccionar todos los elementos de la página.

\* { margin: 0; }

En este caso se va a aplicar un margen de tamaño 0 a todos los elementos html de nuestra página.

**Selector de etiqueta HTML:** Selecciona todos los elementos de la página cuya etiqueta HTML coincide con el valor del selector.

h1 { color: blue; } ó p { color: black; } ó h1 , p { color: blue; }

En el primer caso se establece que el color de letra de todos los títulos <h1> de la página estará en color azul. En el segundo caso se establece que el color de texto de todos los párrafos <p> estará en color negro. En el tercer caso se establece que el color del texto de todos los títulos <h1> y de los párrafos <p> de la página, será azul.

**Selector descendente:** Un elemento es descendiente de otro cuando se encuentra entre las etiquetas de apertura y de cierre del otro elemento. El selector del siguiente ejemplo selecciona todos los elementos <span> de la página que se encuentren dentro de un elemento <p>. Ejemplo:

p span { color: red; }

En este caso se aplicará un color de rojo a los textos de todos los elementos <span> que se encuentren dentro de una etiqueta de párrafo <p>, como en el siguiente ejemplo:

<p><div><span>Este es un texto dentro de un span</span></div></p>

**Selector de hijos:** Se utiliza para seleccionar un elemento que es hijo directo de otro elemento y se indica mediante el "signo de mayor que" (>). Ej:

p > span { color: blue; }

En este caso se aplicará un color azul al texto de todos los elementos <span> que se encuentran contenidos directamente dentro de un elemento de párrafo <p>, es decir que no existen contenedores intermedios, como en el siguiente ejemplo:

<p><span>Este es un texto dentro de un span</span></p>

**Selector de atributos:** permiten seleccionar elementos HTML en función de sus atributos y/o valores de esos atributos.

input[type="text"] { width:150px; }

En este caso se aplicará un ancho de 150px a todos los componentes <input> que sean de tipo “text”, como en el siguiente ejemplo:

<input type="text" name="estrato" value="1">

**Selector id:** El selector id se utiliza para definir un elemento, el cual tiene un valor único en todo el documento. Hacemos referencia a él anteponiendo el símbolo numeral (#).

#idelemento{

text-align: center;

color: red;

}

En este caso se centrará y se establecerá en color rojo, los textos que se encuentre dentro de elementos HTML con el atributo id=“idelemento”, como en el siguiente ejemplo:

<p id="idelemento"> Esto es un párrafo </p>

**Selector class:** Con el selector class se pueden definir diferentes estilos para un mismo elemento HTML. El valor del selector class debe ir antepuesto por un punto (.)

.elemento{

text-align: center;

color: red;

}

En este caso se centrará y se establecerá en color rojo, los textos que se encuentre dentro de elementos HTML con el atributo class=“elemento”, como en el siguiente ejemplo:

<p class="elemento"> Esto es un párrafo </p>

**Como incluir CSS en nuestra página web**

Existen 3 formas de incluir estilos CSS en un documento HTML

* **Internamente:** Se especifican los estilos CSS directamente en el archivo HTML.

<head>

<style>

p { margin-left:20px; }

body { background-image:url("images/back40.gif"); }

</style>

</head>

* **Externamente:** Se invoca un archivo externo que contendrá todos los estilos CSS de la siguiente manera:

<head>

<link rel="stylesheet" type="text/css" href="mystyle.css">

</head>

* **Entre líneas:** Se especifica el CSS directamente en el elemento HTML de la siguiente manera:

<p style="color:sienna; margin:10px;">Este es un párrafo.</p>

***Utilizar el siguiente ejercicio como recurso interactivo***

Para el siguiente código en HTML incluir para el selector **p** un estilo con la propiedad **color** con valor igual a **green** y la propiedad **font-weight** con valor igual a **bold.** Probar incluyendo el estilo CSS internamente y entre líneas.

<html>

<head>

<title>Mi página</title>

</head>

<body>

<p>Esto es un párrafo con mucho estilo</p>

</body>

</html>

**Lección 2. Textos, Fondos y colores.**

**Manejo de colores en CSS:**

En CSS existen diferentes formas de definir los colores. Entre ellas tenemos:

* **Colores Básicos:** CSS define 17 palabras clave para referirse a los colores básicos. Las palabras se corresponden con el nombre en inglés de cada color: aqua, black, blue, fuchsia, gray, green, lime, maroon, navy, olive, orange, purple, red, silver, teal, white, yellow



colores-basicos-html.png

* **RGB:** El modelo RGB consiste en definir un color indicando la cantidad de rojo, verde y azul que se debe mezclar para obtener ese color. Ejemplo: rgb(71,98,176).

Los valores deben estar entre 0 y 255.

El color del ejemplo anterior se puede representar también de forma porcentual. Ejemplo: rgb(27%, 38%, 69%).

* **RGB hexadecimal:** Este modelo se basa en la representación de colores por medio de números hexadecimales; por lo general 6 dígitos, aunque se puede comprimir a 3. Los 2 primeros dígitos para la cantidad de rojo, los 2 siguientes para la cantidad de verde y los últimos 2 para el azul: Ejemplo: #ff33aa

***Utilizar el siguiente ejercicio como recurso interactivo***

En el siguiente código probar las diferentes maneras de establecer colores sobre el selector **h1**.

<html>

<head>

<title>Mi página</title>

<style>

h1{color:red}

</style>

</head>

<body>

<h1>Esto es un título con mucho color</h1>

</body>

</html>

**Estilos de Textos:**

En CSS existen diferentes estilos que podemos aplicar a los textos de nuestra página, entre ellos tenemos los siguientes:

* **font-size:** Esta propiedad establece el tamaño del texto. Este tamaño puede ser absoluto (**mm**: milimetros, **cm**: centimetros, **in**: inches, **pt**: puntos, donde 1pt equivale a 1/72 pulgadas, **pc**: pica, donde 1pc equivale a 12pt, **px**: pixel). O relativos (**em**: 1em hace referencia al tamaño en puntos de la letra que se está utilizando, **%:** porcentaje, **ex**: 1ex es igual a la altura de la letra ‘x’ de la fuente que se está utilizando.)

también se puede definir el tamaño de letra de forma absoluta mediante alguna de las siguientes palabras clave: xx-small, x-small, small, medium, large, x-large, xx-large.

Ejemplos: p{font-size:12px} h1{font-size:2em} .texto{font-size:1cm}

* **font-weight:** Establece el ancho de la letra utilizada para el texto. Los valores que puede tomar esta propiedad son: normal, bold, bolder, lighter, 100, 200, 300, 400, 500, 600, 700, 800, 900, inherit.

Ejemplos: p{font-weight:bold} h1{font- weight:normal} .texto{font- weight:300}

* **font-family:** Con esta propiedad se puede establecer el tipo de letra para nuestro texto. Normalmente se establece por grupos llamados familias. Las familias más utilizadas son:
* Arial, Helvetica, sans-serif;
* "Times New Roman", Times, serif;
* "Courier New", Courier, monospace;
* Georgia, "Times New Roman", Times, serif;
* Verdana, Arial, Helvetica, sans-serif;

Ejemplos: p{font-family: Arial, Helvetica, sans-serif }

h1{font- family: Times New Roman", Times, serif }

* **color:** Con esta propiedad se puede establecer el color de la letra de los textos. Los valores que puede tomar este atributo puede ser cualquiera de los vistos en la sección “Manejo de colores con CSS”.

Ejemplos: p{ color: blue } h1{ color: #FF00BB }

***Utilizar el siguiente ejercicio como recurso interactivo***

En el siguiente código probar los diferentes valores para las propiedades **font-size, color, font-family,** **font-weight,**  del texto dentro del parrafo **p**.

<html>

<head>

<title>Mi página</title>

<style>

p{

color:black;

font-family: Arial, Helvetica, sans-serif;

font-size: 12px;

font-weight: normal;

}

</style>

</head>

<body>

<p>Esto es un texto con mucho estilo</p>

</body>

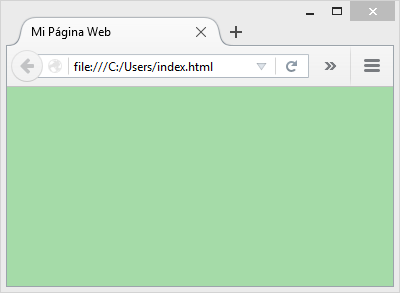
</html>

**Colocando Fondos con CSS:**

En CSS el fondo puede ser un color simple o una imagen. CSS define cinco propiedades para establecer el fondo de cada elemento:

* **background-color:** permite mostrar un color de fondo sólido en un elemento contenedor. Esta propiedad no permite crear degradados ni ningún otro efecto avanzado. Ejemplo:

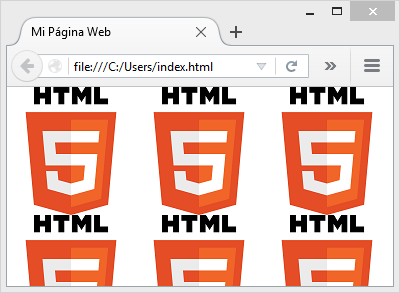
body { background-color: #A5DBA8; }



Fondo1.png

* **background-image:** Permite mostrar una imagen como fondo cualquier elemento contenedor. Ejemplo:

body { background-image: url("img/fondo.png") }

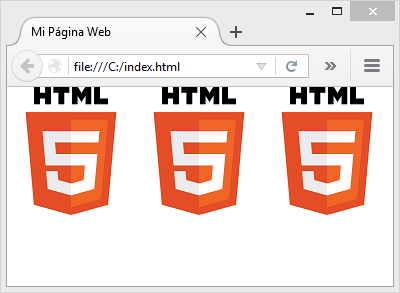


fondo2.png

* **background-repeat:** En ocasiones, no es conveniente que la imagen de fondo se repita horizontal y verticalmente. Para ello, CSS introduce la propiedad background-repeat que permite controlar la forma de repetición de las imágenes de fondo.

Esta propiedad puede tomar los siguientes valores:

* **repeat:** indica que la imagen se debe repetir en todas direcciones y por tanto, es el comportamiento por defecto.
* **repeat-x:** repite la imagen sólo horizontalmente.
* **repeat-y:** repite la imagen solamente de forma vertical.
* **no-repeat:** muestra una sola vez la imagen y no se repite en ninguna dirección.

Ejemplo:

body{

background-image: url("img/fondo.png");

background-repeat: repeat-x

}

fondo3.png

* **background-position:** La propiedad background-position permite indicar la distancia que se desplaza la imagen de fondo respecto de su posición original situada en la esquina superior izquierda. Si se indican dos porcentajes o dos medidas, el primero indica el desplazamiento horizontal y el segundo el desplazamiento vertical respecto del origen. Ejemplo:

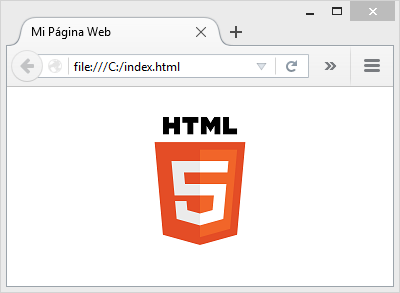
#contenedor {

background-image: url("images/help.png");

  background-repeat: no-repeat;

background-position: 50% -25%;

}



fondo4.png

* **background- attachment**: Opcionalmente, se puede indicar que el fondo permanezca fijo cuando la ventana del navegador se desplaza mediante las barras de scroll; para esto se debe añadir la propiedad:

#contenedor {

background-attachment: fixed;

}

La forma resumida de la propiedad **background** se muestra en el siguiente ejemplo:

background: #ff33aa url("imagen.jpg") no-repeat 10px 27px;

**Ejercicio interactivo:** Probar las diferentes propiedades de fondo, sobre el siguiente codigo

<html>

<head>

<style> body { background: #ff33aa } </style>

</head>

<body></body>

</html>

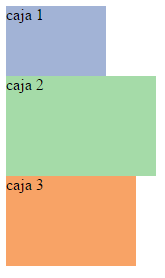
**Lección 3. Posicionamiento y visibilidad de elementos.**

**Visibilidad de elementos:**

CSS proporciona la propiedad **display** la cual permite ocultar completamente un elemento haciendo que desaparezca de la página. Como el elemento oculto no se muestra, el resto de elementos de la página se mueven para ocupar su lugar.

Esta propiedad también modifica la forma en la que se visualiza un elemento.

Los valores más utilizados son:

* **block:** Muestra un elemento como si fuera un elemento de bloque. Cada elemento se posiciona uno debajo del otro. Ejemplo:

#caja1{

width: 100px;

height: 70px;

display: block;

}

#caja2{

width: 150px;

height: 100px;

display: block;

}

#caja3{

width: 130px;

height: 90px;

display: block;

}

Caja\_block.png

* **inline:** Visualiza un elemento en forma de elemento en línea. Es decir, uno al lado del otro, pero no mantiene las propiedades del elemento, como alto, ancho, margen etc. Ejemplo:

#caja1{

width: 100px;

height: 70px;

display: block;

Caja\_inline.png

}

#caja2{

width: 150px;

height: 100px;

display: block;

}

#caja3{

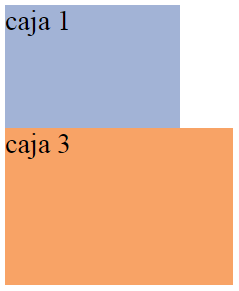
width: 130px;

height: 90px;

display: block;

}

* **none:** Oculta un elemento y hace que desaparezca de la página. Ejemplo:

#caja1{

width: 100px;

height: 70px;

}

#caja2{

width: 150px;

height: 100px;

display: none;

}

#caja3{

width: 130px;

height: 90px;

}

Caja\_none.png

* **inline-block:** Coloca los elementos en línea pero conservando las propiedades de bloque (alto, ancho, margen etc).

#caja1{

width: 100px;

height: 70px;

display: block;

}

#caja2{

width: 150px;

height: 100px;

Caja\_inline-block.png

display: block;

}

#caja3{

width: 130px;

height: 90px;

display: block;

}

* **table-cell:** Trata los elementos como si fueran celdas en una tabla. Las alturas de elementos adyacentes son iguales.

#caja1{

width: 100px;

height: 70px;

display: block;

}

#caja2{

width: 150px;

height: 100px;

display: block;

}

#caja3{

Caja\_table-cell.png

width: 130px;

height: 90px;

display: block;

}

***Utilizar el siguiente código como recurso interactivo para probar la visibilidad de elementos con la propiedad display.***

<html>

<head>

<title>Mi Página Web</title>

<meta charset="utf8">

<style>

#caja1{width: 100px; height: 70px; background:#A2B3D6; display: block;}

#caja2{width: 150px; height: 100px; background:#A5DBA8; display: block;}

#caja3{width: 130px; height: 90px; background:#F8A366; display: block;}

</style>

</head>

<body>

<div id="caja1">caja 1</div>

<div id="caja2">caja 2</div>

<div id="caja3">caja 3</div>

</body>

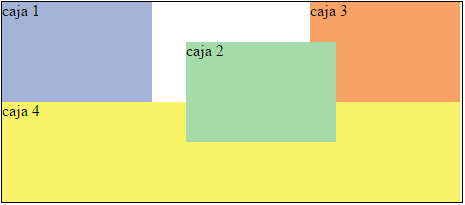
</html>

En este código se debe cambiar la propiedad display de las cajas para verificar su comportamiento.

**Posicionamiento de elementos:**

Utilizando CSS, un elemento se puede posicionar en la pantalla mediante la propiedad **position**. Este posicionamiento puede ser **static** (statico), **relative** (relativo), **absolute** (absoluto) o fixed (fijo).

* **Posicionamiento estatico:** Es la posición normal que tendría el elemento dentro de la pagina.
* **Posicionamiento relativo:** desplaza un elemento respecto de su posición original. El desplazamiento del elemento se controla con las propiedades top (arriba), right (derecha), bottom (abajo) y left (izquierda).

Ejemplo

#caja2 {

width:150px ;

height:100px;

display:inline-block;

**position: relative;**

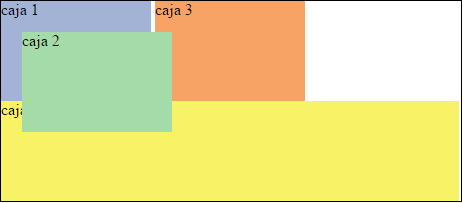
top: 40px;

left: 30px;

}

Caja\_relative.png

* **Posicionamiento absoluto:** la posición de un elemento se establece de forma absoluta respecto de su elemento contenedor y el resto de elementos de la página ignoran la nueva posición del elemento. El desplazamiento del elemento se controla con las propiedades top (arriba), right (derecha), bottom (abajo) y left (izquierda).

Ejemplo:

#caja2 {

width:150px ;

height:100px;

display:inline-block;

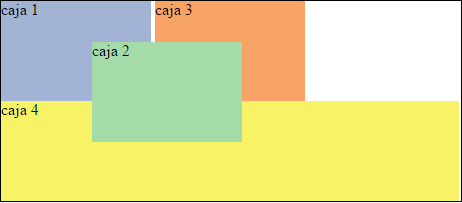
**position: absolute;**

top: 40px;

left: 30px;

}

Caja\_absolute.png

* **Posicionamiento** **fijo:** Hace que un elemento sea inamovible, de forma que su posición en la pantalla siempre es la misma independientemente del resto de elementos e independientemente de si el usuario sube o baja la página en la ventana del navegador.

Ejemplo:

#caja2{

width:150px;

height:100px;

**position:fixed;**

left:100px;

top:50px;

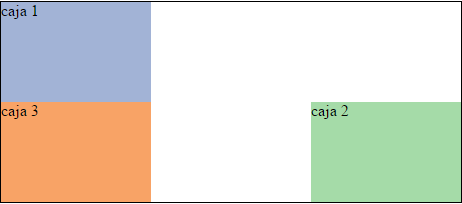
Caja\_fixed.png

}

Existe otro tipo de posicionamiento llamado flotante. Este se establece con la propiedad **float**. Cuando una caja se posiciona con el modelo de posicionamiento flotante, se desplaza hasta la zona más a la izquierda o más a la derecha de la posición en la que originalmente se encontraba.

La propiedad **float** puede tener alguno de los siguientes valores: left (Izquierda), right (Derecha) ó none (Ninguno).

Caja\_float.png

Ejemplo:

#caja1

{

width:150px;

height:100px;

**float: right;**

}

***Utilizar el siguiente código como recurso interactivo para probar los posicionamientos***

<html>

<head>

<title>Mi Página Web</title>

<meta charset="utf8">

<style>

#contenedor{width: 460px; height: 200px; border:1px solid black;}

#caja1{width: 150px; height: 100px; background:#A2B3D6; display: inline-block;}

#caja2{

width: 150px; height: 100px; background:#A5DBA8; display: inline-block;

position: relative; top:20px; left:40px;

}

#caja3{width: 150px; height: 100px; background:#F8A366; display: inline-block;}

#caja4{width: 458px; height: 100px; background:#F8F366}

</style>

</head>

<body>

<div id="contenedor">

<div id="caja1">caja 1</div>

<div id="caja2">caja 2</div>

<div id="caja3">caja 3</div>

<div id="caja4">caja 4</div>

</div>

</body>

</html>

En este código se debe cambiar la propiedad position, el top y left de la caja 2 para verificar su funcionamiento.

**Lección 4. Espaciado, márgenes y bordes.**

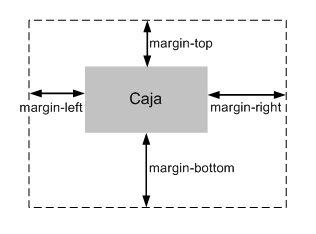
Los elementos en HTML pueden ser considerados como “cajas”. Aquí se establece un modelo de cajas el cual es esencialmente una representación de una caja que envuelve elementos HTML y está conformada por: Margin (Margen), Border (Borde), Padding (Espaciado) y el contenido que tendrá internamente el elemento.



modelo\_ cajas.jpg

* **Margin:** Deja un espacio libre alrededor del borde de la caja.

CSS define cuatro propiedades para controlar cada uno de los márgenes horizontales y verticales de un elemento los cuales son: margin-top, margin-right, margin-bottom, margin-left). Cada una de las propiedades establece la separación entre el borde lateral de la caja y el resto de cajas adyacentes.

Ejemplo:

#contenedor

{

margin-top: 5px;

margin-right: 8px;

margin-bottom: 10px;

margin-left: 7px;

}

Las propiedades anteriores se pueden

resumir de la siguiente manera:

margenes.gif

#contenedor {

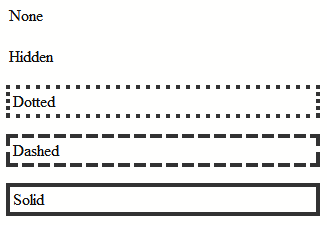
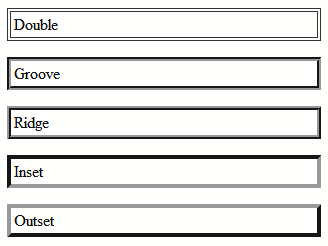
margin: 5px 8px 10px 7px;

}

* **Border:** Borde de la caja. Puede tener grosor, color y estilo de borde. Sus propiedades son:

**-border-width:** Especifica el ancho del borde, normalmente en pixeles. También se puede usar los valores predefinidos: thin, medium, ó thick.

**-border-color:** Especifica el color del borde. Los valores se pueden asignar indicando el nombre del color (red,black, white etc), el código rgb (rgb(255,0,0)) ó el código hexadecimal (#ff0000).

**-border-style:** Especifica el estilo de borde a mostrar el cual puede ser: none, hidden, dotted, dashed, solid, doublé, groove, ridge, inset, outset.

Ejemplos:

#elemento{

border-style:solid;

border-width:5px;

border-color:#ff0000

 }

Las propiedades anteriores se pueden resumir de la siguiente manera:

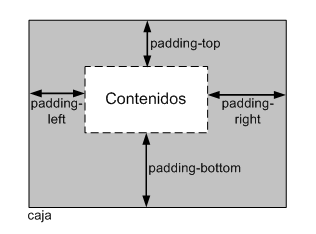
   #elemento{

border: solid 5px #ff0000;

  }

* **Padding:** Espacio entre el borde de la caja y el contenido de esta.

CSS define cuatro propiedades para controlar cada uno de los espaciados horizontales y verticales de un elemento (padding-top, padding-right, padding-bottom, padding-left). Cada una de estas propiedades establece la separación entre el contenido y los bordes laterales de la caja del elemento.



Ejemplo:

#caja {

padding-top: 5px;

padding-right: 8px;

padding-bottom: 10px;

padding-left: 7px;

}

Las propiedades anteriores se pueden

resumir de la siguiente manera:

padding.gif

#caja {

padding: 5px 8px 10px 7px;

}

* **Content:** El contenido de la caja (texto, imágenes, objetos, etc)

**PROGRAMACION WEB 2**

**Módulo 4: Lenguaje JavaScript y JQuery**

**Lección 1. Introducción a JavaScript y JQuery**

Ahora que ya hemos aprendido un poco de HTML y CSS, vamos a agregarle un poco más de interactividad a nuestra página con un sencillo pero poderoso lenguaje de programación llamado **JavaScript**.

JavaScript al igual que PHP y CSS, es un lenguaje que se puede introducir fácilmente dentro de un documento HTML. Esto se puede hacer llamando un archivo externo que contenga el código en JavaScript, o incrustándolo directamente dentro del código HTML.

Para llamar un archivo externo, se debe introducir una línea como la siguiente entre las etiquetas <head> y </head> de HTML de la siguiente manera.

<head>

**<script type="text/javascript" src="codigo.js"></script>**

</head>

Los archivos JavaScript tienen la extensión .js. En este caso estamos incluyendo el archivo codigo.js que contiene el código en este lenguaje.

Para incluir código Javascript directamente dentro del código HTML, se utiliza las etiquetas **<script>** y **</script>**, entre las cuales irá dicho código como se muestra a continuación.

<html>

<head>

<title>Mi página con JavaScript</title>

<meta charset="UTF-8">

</head>

<body>

<p>Esto es código HTML</p>

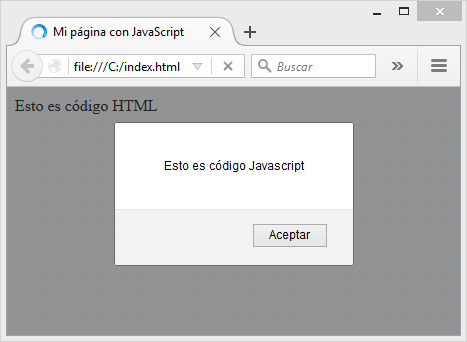
**<script>**

**alert("Esto es código Javascript");**

**</script>**

</body>

</html>



Javascript1.png

La instrucción **alert("texto a mostrar")**, genera una ventana emergente llamada también Pop Up, como la que se muestra en la imagen.

En JavaScript, cada instrucción termina con punto y coma (;).

**Variables en JavaScript:** Para crear una variable en JavaScript**,** utilizamos la palabra **var** seguida del nombre de la variable. En este lenguaje no hay necesidad de establecer el tipo de variable, es decir, no se necesita especificar si es una variable numérica, de tipo fecha, de texto, etc. El tipo de la variable va a estar definido por el valor que le haya sido asignado a esta.

var numero=10; var saludo='Hola'; var estatura=1.70;

**Cálculos matemáticos:** Para realizar un cálculo matemático, tenemos a disposición los operadores matemáticos **+** (suma), **-** (resta), **\*** (multiplicación), **/** (división), **^** (potencia). También se pueden utilizar los paréntesis en JQuery en los cálculos matemáticos.

A continuación veamos algunos ejemplos del uso de estos operadores:

var suma=10 + 2; var multiplicacion=10 \* 2; var division=10 / 2; var x= (3+7)/2;

También podemos hacer las operaciones utilizando las variables así:

<script>

var primerNumero=10;

var segundoNumero=5;

var suma= primerNumero + segundoNumero;

</script>

**Ejercicio interactivo**

Utilizando el siguiente código en JavaScript, asignar un valor a cada variable y realizar una operación para promediar 3 números.

<html>

<head></head>

<body>

<script>

var primerNumero; //Asignar aquí el primer valor

var segundoNumero; //Asignar aquí el segundo valor

var tercerNumero; //Asignar aquí el tercer valor

var promedio; //Realizar aquí la operación matemática

alert(promedio);

</script>

<body>

</html>

**Nota**: El promedio de 3 números se calcula así: (nro1+nro2+nro3) / 3

**JQuery:**

JQuery es una librería de JavaScript que nos permite simplificar la manera de interactuar con los elementos HTML y también simplifica la manera de escribir el código JavaScript.

JQuery es un archivo con extensión.js que contiene varias funcionalidades y que se incluye dentro de un documento HTML de la misma forma en que incluíamos un archivo JavaScript externo. Lo primero que debemos hacer es descargar el archivo, guardarlo e incluirlo en nuestro proyecto.

Podemos descargar el archivo de JQuery desde el siguiente enlace:

<http://code.jquery.com/jquery-2.1.4.min.js>

y lo incluiríamos en nuestro proyecto asi:

<head>

<script type="text/javascript" src="jquery-2.1.4.min.js"></script>

</head>

Siempre que vayamos a trabajar con la librería JQuery debemos incluir lo siguiente dentro de las etiquetas <script> y </script>:

<script>

**$(function(){**

   //Aquí van todas las instrucciones de JQuery

**});**

</script>

**Lección 2. Encontrar y acceder a los elementos HTML**

Con Jquery podemos encontrar y acceder a las propiedades, valores y al contenidode los elementos o componentes de HTML, en este caso aprenderemos, como obtener los valores almacenados en los campos de entrada de información de un formulario en HTML.

Para acceder a cualquier componente HTML, es importante haberle asignado un valor a la propiedad **id** o a la propiedad **class** que posee cada uno, de esta manera podemos hacer referencia al elemento y obtener otras propiedades de este o el valor que contiene si fue asignado desde un formulario.

Para acceder al valor que contiene un campo de un formulario utilizando JQuery, se debe escribir la siguiente instrucción:

var valorCampo = $("#IdDelCampo").val();

Si queremos acceder al campo a través del **id**, ó:

var valorCampo = $(".ClassDelCampo").val();

Si queremos acceder al campo a través del **class**.

También podemos acceder a un campo, pero esta vez no para obtener su valor, sino para asignarle uno. Esto se realiza de la siguiente manera:

$("#IdDelCampo").val("Valor a asignar");

$(".ClassDelCampo").val("Valor a asignar");

$(".ClassDelCampo").val( $("#OtroCampo").val() );

O por intermedio de una variable, así:

var **numero**=10;

$("#IdDelCampo").val( **numero** );

Para entender mejor lo que hemos visto hasta el momento, vamos a realizar un ejemplo completo en el cual crearemos un formulario con un campo con id="CampoSaludo", al cual le vamos a asignar en la propiedad value, el texto "Hola, Como estas!", luego este valor lo asignaremos utilizando Jquery, a otro campo que crearemos en el formulario con id="resultado".

El código completo se muestra a continuación:

<html>

<head>

<title>Mi página con JQuery</title>

<meta charset="UTF-8">

<script type="text/javascript" src="jquery-2.1.4.min.js"></script>

</head>

<body>

<form>

Campo 1: <input type="text" id="CampoSaludo" value="Hola, como estas">

<br><br>

Campo 2: <input type="text" id="resultado">

</form>

<script>

$(function(){

var saludo= $("#CampoSaludo").val();

$("#resultado").val(saludo);

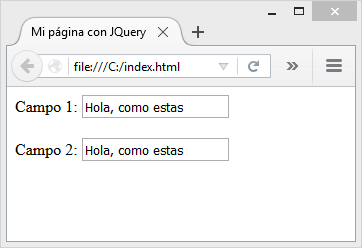
});

</script>

</body>

</html>

Obteniendo el siguiente resultado:



acceso1.png

Siempre que obtenemos un valor de un campo con JQuery, este se comportará como si de un texto se tratase, por lo tanto si quisiéramos hacer una operación matemática, no se podría hacer directamente. Para esto necesitaríamos convertir cada valor que se encuentra como texto a un valor numérico, lo cual se puede hacer con una funcionalidad llamada **parseInt**, y usa se la siguiente manera:

parseInt("10"); ó parseInt(variable);

O para nuestro ejemplo: **parseInt(** $("#CampoNumerico").val() **);**

Ahora veremos un ejemplo en el que sumaremos dos números de 2 campos de un formulario y mostraremos el resultado en un tercer campo del mismo:

<html>

<head>

<title>Mi página con JQuery</title>

<meta charset="UTF-8">

<script type="text/javascript" src="jquery-2.1.4.min.js"></script>

</head>

<body>

<form>

Primer número: <input type="text" id="numero1" value="10"> <br><br>

Segundo número: <input type="text" id="numero2" value="5"><br><br>

Resultado: <input type="text" id="resultado">

</form>

<script>

$(function(){

var nro1 = parseInt( $("#numero1").val() );

var nro2 = parseInt( $("#numero2").val() );

var suma = nro1 + nro2;

$("#resultado").val(suma);

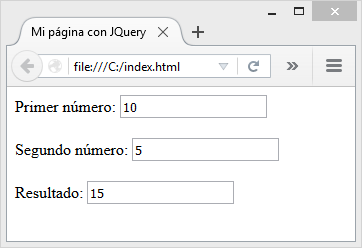
});

</script>

</body>

</html>

Obteniendo el siguiente resultado:



acceso2.png

**Ejercicio interactivo**

Utilizando el siguiente código, asignar un valor numérico a cada campo del formulario, luego promediar estos 2 valores con JQuery y asignar el promedio calculado al campo identificado con el id “resultado”.

<html>

<head>

<title>Mi página con JQuery</title>

<meta charset="UTF-8">

<script type="text/javascript" src="jquery-2.1.4.min.js"></script>

</head>

<body>

<form>

Primer número: <input type="text" id="numero1" value=""> <br><br>

Segundo número: <input type="text" id="numero2" value=""><br><br>

Resultado: <input type="text" id="resultado">

</form>

<script>

$(function(){

var nro1; //Obtener aquí el primer valor

var nro2; //Obtener aquí el segundo valor

var promedio; //Calcular aquí el promedio

$("#resultado").val(); //Asignar aquí el promedio

});

</script>

</body>

</html>

**Lección 3. Validación de formularios.**

La validación de formularios es un aspecto muy importante cuando se crea una página web que capturará información, y más aún si esta información será almacenada en una base de datos.

La validación consiste en verificar que la información ingresada, sea correcta y coherente de acuerdo a lo que se requiere en cada campo. Es decir, se verifican cosas como si un campo que es obligatorio esta vacío, si un campo que debe contener valores numéricos tienen un número asignado, si los valores ingresados en un campo están en cierto rango, etc.

Ya que hemos aprendido a acceder con JQuery a los valores que contiene un campo, podemos verificar y validar utilizando condiciones de programación, y así determinar si dejamos que los datos del formulario se envíen o no.

Para que los valores que contienen los campos de un formulario, se envíen a otra página, debemos colocar un botón que se encargará de ejecutar esta acción. En el momento que se de click en este botón se debe realizar la validación del formulario. Las instrucciones que realiza la validación, deben estar encerradas en una porción de código llamada función, las cuales se explicaran con más detalle en la siguiente lección.

La manera de determinar en JQuery si se ha hecho click sobre un botón es la siguiente:

$( "#idBoton" ).click(function() {

//Aquí colocaremos las instrucciones para validar el formulario

});

Donde **idBoton** es el id que se le asignó al botón dentro del formulario. Ejemplo:

<input type="submit" value="Enviar" id="idBoton">

Ahora vamos a realizar un pequeño ejemplo en el que validaremos si el valor asignado a un campo de un formulario se encuentra entre 1 y 99.

<html>

<head>

<title>Mi página con JQuery</title>

<meta charset="UTF-8">

<script type="text/javascript" src="jquery-2.1.4.min.js"></script>

</head>

<body>

<form action="pagina2.html" method="post" >

número: <input type="text" id="numero1"><br><br>

<input type="submit" value="Enviar" id="idBoton">

</form>

<script>

$( "#idBoton" ).click(function() {

var num = parseInt( $("#numero1").val() );

if (num > 0 && num<100){

alert("El número está en el rango permitido");

}

else{

alert("El número no está en el rango permitido");

}

});

</script>

</body>

</html>

Con la instrucción if (num > 0 && num<100), estamos preguntado si la variable **num** es mayor que cero (0) y si **num** es menor que cien (100), de ser así se muestra una ventana con el texto “El número está en el rango permitido”, y de no ser así muestra una ventana con el texto “El número no está en el rango permitido”.

En este ejemplo aunque se valida lo que requeríamos inicialmente, no estamos impidiendo que se envíen los datos a la siguiente página. Para esto vamos a modificar un poco nuestro código.

Lo primero que haremos es agregar la siguiente propiedad a la etiqueta **form**:

onSubmit="return validar()"

Con **onSubmit** estamos diciendo que cuando se envíe los datos del formulario, llame una función llamada **validar()**, y si esta función contesta verdadero (true) se envía el formulario y se abre la siguiente página especificada en la propiedad **action**, de lo contrario si contesta falso (false) no se enviará el formulario y no se abrirá la página.

<html>

<head>

<title>Mi página con JQuery</title>

<meta charset="UTF-8">

<script type="text/javascript" src="jquery-2.1.4.min.js"></script>

</head>

<body>

<form action="pagina2.html" method="post" onSubmit="return validar()">

número: <input type="text" id="numero1"><br><br>

<input type="submit" value="Enviar" id="idBoton">

</form>

<script>

function validar(){

var num = parseInt( $("#numero1").val() );

if (num > 0 && num<100){

alert("El número está en el rango permitido");

return true;

}

else{

alert("El número no está en el rango permitido");

return false;

}

}

</script>

</body>

</html>

**Otros ejemplos de validaciones**

Si quisiéramos preguntar si un campo está vacío podríamos escribir algo como esto:

if( $("#Campo").val()=="" )

Si un campo contiene la letra F o M

if( $("#Campo").val()=="F" || $("#Campo").val()=="M" )

Si un campo contiene un número mayor al de otro campo

if( parseInt( $("#Campo1").val() ) > parseInt( $("#Campo2").val() ) )

Si un campo contiene un valor diferente a la letra X

if( parseInt( $("#Campo1").val() != "X" )

**Ejercicio interactivo:** Utilizando el siguiente código, validar que los dos campos del formulario no estén vacíos antes de ser enviados.

<html>

<head>

<title>Mi página con JQuery</title>

<meta charset="UTF-8">

<script type="text/javascript" src="jquery-2.1.4.min.js"></script>

</head>

<body>

<form action="pagina2.html" method="post" onSubmit="return validar()">

Campo 1: <input type="text" id="campo1"><br><br>

Campo 2: <input type="text" id="campo2"><br><br>

<input type="submit" value="Enviar" id="idBoton">

</form>

<script>

function validar(){

if ( ) { // Realizar aquí la validación

alert(" "); // Poner aquí un mensaje

return true;

}

else{

alert(" "); // Poner aquí un mensaje

return false;

}

}

</script>

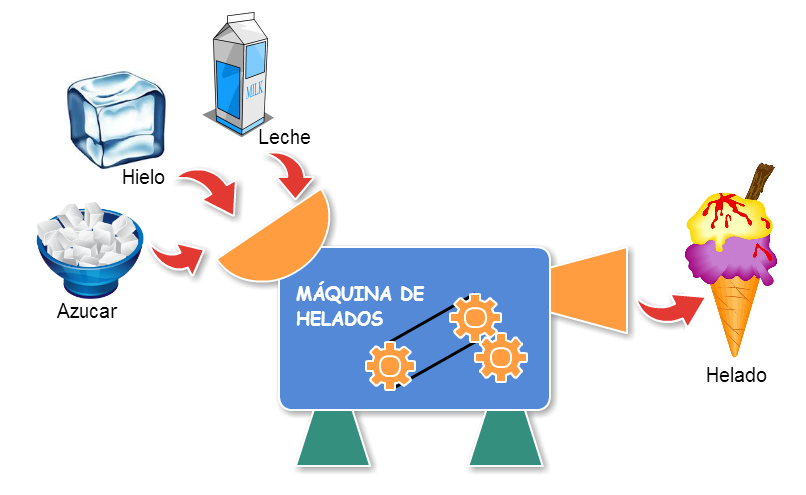
</body>

</html>

**Lección 4. Funciones en Javascript.**

Una función es una porción de código o de programa que realiza operaciones y cálculos por sí solo. Cada función tiene un nombre que la identifica, unos valores o parámetros de entrada, un valor de salida llamado valor de retorno, y además se encuentra claramente definido su inicio y fin.

Para entender mejor una función, imaginémonos que tenemos una máquina que hace helados.



funcion1.png

Vamos a imaginarnos que la maquina es similar a una función. Esta requiere que se le ingresen unos ingredientes, los cuales representarían en una función los valores o parámetros de entrada. Luego la maquina se encarga de hacer un proceso utilizando estos ingredientes y finalmente sale de ella un helado, el cual es similar al valor de salida o retorno de la función.

En JavaScript una función tiene la siguiente forma.

function nombre\_funcion(parametro1, parametro2){

//Aquí van todas las instrucciones

return valor;

}

Si pudiéramos representar la máquina de helados como una función en código JavaScript se vería más o menos así:

function maquina\_helados(azucar, hielo, leche){

//operaciones para hacer el helado

return helado;

}

Habiendo entendido estos conceptos, vamos a ver un ejemplo un poco más real del uso de una función.

Supongamos que necesitamos una función que se encargue de sumar 3 números. Para esto vamos a identificar cada uno de los componentes de una función (parámetros de entrada, valor de salida y el nombre de la función).

**Nombre de la función**: En este caso podemos escoger el nombre que nos parezca más adecuado, por ejemplo “sumar\_numeros”.

**Parámetros de entrada:** Es claro que para que la función pueda realizar la operación requerida necesita que se le envíe los 3 números que se necesitan sumar. Estos 3 números se pueden escribir directamente o se pueden enviar en 3 variables, las cuales llamaremos numero1, numero2 y numero3.

**Valor de salida o retorno:** El valor que requerimos que nos entregue la función, es el resultado de sumar los 3 números. Este resultado lo vamos a retornar en una variable que llamaremos suma.

Teniendo identificado esto, procederemos a construir la función. Esta quedaría de la siguiente forma:

function sumar\_numeros(numero1, numero2, numero3){

var suma;

suma= numero1 + numero2 + numero3;

return suma;

}

En este caso la operación que realiza la función consiste en sumar cada uno de los números que se ingresaron como parámetros de entrada y guardar este resultado en la variable suma para luego ser retornada.

Una función por sí sola no realiza ninguna acción, si esta no es llamada o invocada desde alguna parte del programa. Así mismo la máquina de helados no trabajaría si alguien no la enciende o activa. Para invocar una función, solo basta con llamarla por su nombre y enviarle los parámetros que esta requiere. El valor que la función retorna se puede capturar asignándolo a una variable de la siguiente manera:

var resultado = sumar\_numeros(numero1, numero2, numero3);

El código completo del ejemplo anterior quedaría así:

<html>

<head>

<title>Mi primera función</title>

<meta charset="UTF-8">

</head>

<body>

<script>

var numero1=10;

var numero2=20;

var numero3=30;

var resultado= sumar\_numeros(numero1, numero2, numero3);

alert(resultado);

function sumar\_numeros(numero1, numero2, numero3){

var suma;

suma= numero1 + numero2 + numero3;

return suma;

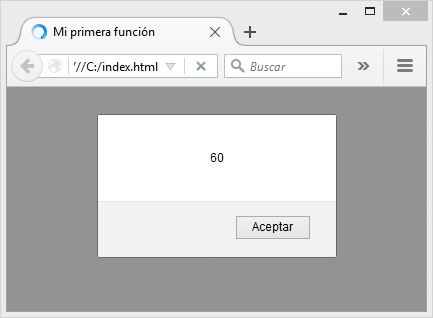
}

</script>

</body>

</html>

Obteniendo el siguiente resultado:



funcion2.png

Como vemos en este ejemplo, la función es una porción de código separado, que se encuentra claramente delimitado y que podemos invocar desde otra parte del programa. Dentro de nuestro programa se pueden agregar cuantas funciones sean necesarias, manteniendo siempre la misma estructura.

**Ejercicio interactivo:** Con ayuda del siguiente código, realizar una función que calcule el área de un rectángulo, teniendo en cuenta que **Area =Alto x Largo.**

<html>

<head>

<title>Mi primera función</title>

<meta charset="UTF-8">

</head>

<body>

<script>

var alto=0; //Asignarle un valor a esta variable

var largo=0; //Asignarle un valor a esta variable

var resultado= area\_rectangulo(alto, largo);

alert(resultado);

function area\_rectangulo (numero1, numero2, numero3){

var area;

// Escribir aquí las instrucciones necesarias

return area;

}

</script>

</body>

</html>

**WEBGRAFIA**

<http://activ.com.mx/javascript-para-principiantes/>

http://programacion.net/articulo/manual\_jquery\_para\_principiantes\_343