HTML5, CSS Y JAVASCRIPT

Crea tu web y apps con el estándar de desarrollo

José Dimas Luján Castillo

Alfaomega

libros R

HTML5, CSS Y JAVASCRIPT

Crea tu web y apps con el estándar de desarrollo

HTML5, CSS Y JAVASCRIPT

Crea tu web y apps con el estándar de desarrollo

José Dimas Luján Castillo

Alfaomega

libros

Diseño de colección, cubierta
y pre-impresión: Grupo RC

Datos catalográficos

Luján, José Dimas
HTML5, CSS Y JAVASCRIPT.
Crea tu web y apps con el estándar de desarrollo
Primera Edición
Alfaomega Grupo Editor, S.A. de C.V., México

ISBN: 978-607-622-642-1

Formato: 17 x 23 cm Páginas: 276

HTML5, CSS Y JAVASCRIPT. Crea tu web y apps con el estándar de desarrollo
José Dimas Luján Castillo
ISBN: 978-84-943450-9-8 edición original publicada por RC Libros, Madrid, España.

Primera edición: Alfaomega Grupo Editor, México, Febrero 2016

Pitágoras 1139, Col. Del Valle, 03100, México D.F.

Miembro de la Cámara Nacional de la Industria Editorial Mexicana
Registro No. 2317

Pág. Web: **http://www.alfaomega.com.mx**
E-mail: **atencionalcliente@alfaomega.com.mx**

ISBN: 978-607-622-642-1

Edición autorizada para venta en México y todo el continente americano.

Impreso en México. Printed in Mexico.

Empresas del grupo:

México: Alfaomega Grupo Editor, S.A. de C.V. – Pitágoras 1139, Col. Del Valle, México, D.F. – C.P. 03100.
Tel.: (52-55) 5575-5022 – Fax: (52-55) 5575-2420 / 2490. Sin costo: 01-800-020-4396
E-mail: atencionalcliente@alfaomega.com.mx

Colombia: Alfaomega Colombiana S.A. – Calle 62 No. 20-46, Barrio San Luis, Bogotá, Colombia,
Tels.: (57-1) 746 0102 / 210 0415 – E-mail: cliente@alfaomega.com.co

Chile: Alfaomega Grupo Editor, S.A. – Av. Providencia 1443. Oficina 24, Santiago, Chile
Tel.: (56-2) 2235-4248 – Fax: (56-2) 2235-5786 – E-mail: agechile@alfaomega.cl

Argentina: Alfaomega Grupo Editor Argentino, S.A. – Paraguay 1307 P.B. Of. 11, C.P. 1057, Buenos Aires,
Argentina, – Tel./Fax: (54-11) 4811-0887 y 4811 7183 – E-mail: ventas@alfaomegaeditor.com.ar

CONTENIDO

PREFACIO

Internet es la tecnología que involucra casi cualquier actividad que realiza el ser humano, hoy en día la mayoría de la información se traslada utilizándolo. Si quisiéramos cuantificar las tecnologías que utiliza internet podríamos llegar a varios miles de ejemplos, pero existe una tecnología que es la base de todos los proyectos, avances e innovación que suceden en internet: HTML.

HTML es una tecnología que no funciona por sí sola, se complementa con otras formando así el conjunto que consolida la base sólida de cualquier proyecto web: HTML5, CSS3 y JavaScript.

HTML5 es uno de los pasos más grandes en la definición del estándar HTML que se han establecido. Desde hace ya algunos años se comenzó a dar forma a este proyecto y actualmente podemos disfrutar de un gran estándar que nos permite realizar desarrollos que no podríamos creer si comparamos su estado con hace algunos años.

Existen muchos actores involucrados en la evolución de HTML5, compañías, consorcios y otros. Pero a lo largo de este libro nos centraremos en las características principales en las que se nota la evolución entre las versiones de los estándares HTML y CSS. Por otro lado conoceremos JavaScript y cómo es la programación relacionada con internet, ya que tiene sus características especiales al no ser un lenguaje de programación de “escritorio” y la peculiaridad de ejecutarse en el navegador.

Acerca del autor

El autor de este libro es un apasionado de la tecnología y la docencia. Comenzó en el mundo de la programación con el lenguaje BASIC a los 13 años de edad. Colaborador habitual de comunidades en español sobre temas como: desarrollo de videojuegos, programación orientada a objetos, desarrollo web y dispositivos móviles.

José Dimas Luján Castillo nació en 1986, tiene el grado de Maestría en Tecnologías de Información. En la docencia ha colaborado con más de 10 universidades a nivel presencial en Latinoamérica en los niveles de Licenciatura y Maestría. En la educación en línea es colaborador de las plataformas más importantes a nivel mundial con más de 60 cursos en línea en la actualidad; además de ser conferenciante habitual de eventos tecnológicos apoyando siempre la adopción de nuevas tecnologías.

Para que el lector pueda consultar y contactar con el autor, puede localizarlo en redes sociales con el alias josedlujan, twitter, Facebook, entre otras. En su web: www.josedlujan.com o por correo electrónico a josedlujan@gmail.com.

Agradecimientos

Este libro es el resultado de un camino largo en el mundo de la tecnología y que sin el apoyo de ciertas personas en el camino no se hubiera podido lograr. Agradezco a mi pareja Noemí, que además de su apoyo diario y soportar ausencias, se tomó el tiempo de revisar/sugerir temas y verificó textos para mejorarlos; también a mis padres, José y Fabiola, cuya prioridad fue siempre mi educación; a una muy buena persona, Mauricio Luckie, que me presentó (regaló) mi primer ordenador y que mi destino probablemente sería otro o hubiera tardado más en llegar sin ese gran obsequio a los 15 años; a Víctor Rubio Ragazzoni que, además de ser mi jefe, se hizo amigo mío y una persona que compartió su experiencia siempre con el fin de enseñar una “experiencia de vida positiva” y añadiendo siempre grandes consejos; a Rosendo Arciga que es mi mejor amigo, por enseñarme que una persona íntegra siempre sale adelante sin importar las pruebas, con su ejemplo vi que la integridad siempre será un sustento para ir por el camino correcto.

1 INTRODUCCIÓN

Actualmente es difícil hablar de algún tema y que este no tenga relación con internet. Debemos ser conscientes del papel protagonista de la red más grande en la actualidad (internet) en la vida del ser humano, es un punto de partida estratégico en la mayoría de las industrias como la militar, los videojuegos, la medicina y otras. Esa es la razón por la que hoy en día nos interesan preguntas como: ¿En dónde está internet? ¿Cómo hago un sitio web? ¿Qué tecnologías están involucradas? Sumadas a estas, podemos encontrar muchas otras preguntas en relación con la tecnología que hace posible la existencia de internet, y gracias a esta red podemos tener a nuestro alcance: información, productos, servicios, comunicación y casi cualquier otra cosa que podamos imaginar.

La persona que marcó la pauta para que esto comenzara fue Tim Berners-Lee, al que muchos debemos considerar como el padre de internet, ya que él propuso dos de las tecnologías bases que aún en nuestros días son pieza fundamental de todo lo que se mueve por internet. Las tecnologías que propuso son HTML y HTTP.

HTML es una tecnología que se puede definir como un “estándar”. La palabra estándar hace que lo equiparemos a los estándares que podemos encontrar en otros

temas, por ejemplo: seguridad, control, calidad, etc. Los estándares llevan un formato, tienen normas y otros conceptos que iremos abordando más adelante.

HTML es un conjunto de normas que debemos de seguir, cuyo objetivo es desplegar un sitio web de forma correcta e indicada para que los navegadores web puedan leer el lenguaje HTML e interpretarlo, para así finalmente mostrar al usuario un sitio web.

HTTP *(HyperTest Transfer Protocol)* es un protocolo como su nombre indica "de transferencia", con el cual se estableció la idea de transferir información entre las computadoras.

Todos podemos ver el famoso protocolo HTTP en funcionamiento en la barra de navegación cuando escribimos una dirección, como por ejemplo:

```
http://josedlujan.com
```

En la línea anterior estamos indicando que usaremos el protocolo HTTP (no importa el navegador que estemos usando).

Además de los aportes antes mencionados, Tim Berners-Lee también fundó la W3C (*World Wide Web Consortium*). Este grupo se creó en el año 1994 con el objetivo de desarrollar los protocolos que permitirían el funcionamiento de la web, se puede decir que este organismo representa la máxima autoridad en el momento de establecer las reglas de los protocolos involucrados en internet.

¿Qué es HTML?

HTML es un estándar desarrollado con el objetivo de mostrar archivos de texto a un usuario agregando colores, estilos, diseños, esto hace que el archivo sea mucho mas fácil en comparación con un archivo de texto plano (txt). En la actualidad ya son sorprendentes los alcances de HTML, los desarrolladores en la web hoy en día tienen mucho trabajo ya que los cambios que sufre HTML están sucediendo a gran velocidad.

La definición técnica de HTML (*HyperText Markup Lenguage*), si prestamos atención a la traducción de sus siglas al español, quiere decir: lenguaje de marcas de hipertexto. Entonces se puede definir de una forma sencilla como "lenguaje de etiquetado".

Después de la definición académica y la definición técnica, comparto mi definición de HTML: HTML es un lenguaje que sirve para modelar y estructurar documentos, estos documentos tienen el fin de llegar a un navegador para ser interpretados para que este muestre la información de acuerdo con la estructura del lenguaje HTML en el documento.

Uno de los conceptos básicos de HTML son las etiquetas, estas son las que escribimos para dar estructura al archivo y dependiendo de ellas el navegador muestra la información.

Por ejemplo: existe una etiqueta que es la *,* esta etiqueta significa *bold* o negrita. Lo que va a hacer es que el texto que sea etiquetado con ella se verá más oscuro o sobre marcado. Si yo quiero que "José Luján" se vea en negritas, tendré que hacer lo siguiente:

```
<b>José Luján </b>
```

Podemos ver en la línea anterior que el texto que deseamos se remarque en negrita se encuentra encerrado entre las etiquetas *.*

La mayoría de las etiquetas tienen una regla: cada vez que abrimos una etiqueta se debe cerrar. Decir que abrimos una etiqueta es colocarla así: "", al decir que cerramos la etiqueta se le coloca el símbolo de "/" como por ejemplo: "". No todas las etiquetas que abrimos las debemos de cerrar colocando la misma etiqueta con la diagonal, pero sí la gran mayoría. Saber cuáles se cierran y cómo se cierran lo vamos aprendiendo conforme vamos conociendo las etiquetas.

HTML no es un lenguaje de programación, ya que un lenguaje de programación tiene variables, ciclos, tipos de datos y otras propiedades exclusivas de los lenguajes. En cambio, HTML tiene como base las etiquetas, los atributos y los estilos. Por eso a

la persona que desarrolla sitios web se le conoce como “desarrollador web” y a las personas que utilizan un lenguaje de programador se les conoce como “programadores”. Más adelante hablaremos sobre esto.

Versiones HTML

HTML al ser un estándar también cuenta con una evolución, esto significa que se le van haciendo cambios que se van agregando al estándar. También se van eliminando funciones o propiedades que ya se consideran antiguas y que se van descartando con el paso del tiempo. En la actualidad nos encontramos en la versión de HTML5.

En la siguiente tabla podremos conocer la evolución de HTML.

1991	Se publica la primera descripción de HTML
1995	Se publica HTML 2
1997	Se publica HTML 3.2 sustentada por la W3C
1999	Se publica HTML 4.0
2000	Se da a conocer XHTML 1
2004-2008	Se trabaja en el borrador de HTML 5

El entender que HTML es un estándar es muy importante, ya que no es un producto como los que podemos encontrar en el supermercado. Al ser un estándar es constantemente sometido a revisiones, cambios y actualizaciones. Esto es fundamental ya que el estándar se utiliza en muchos casos cuando aún no está “finalizado”, por tal motivo se pueden presentar problemas sin previo aviso y el estándar puede ser modificado buscando arreglarlo o su mejorarlo.

Podemos decir que la versión se encuentra completa cuando se acaba, esto quiere decir que el día que pasemos a HTML6 significará que ya se acabó de definir HTML5. Actualmente se tiene planeado finalizar la definición de HTML5 en el año 2022.

Navegador web

El navegador web tiene una relación estrecha con HTML, es difícil separar los conceptos y hablar de alguno sin mencionar al otro. El navegador web se creó casi al mismo tiempo que HTML, ya que el navegador web es el que nos muestra el resultado de "etiquetar" nuestro texto con HTML.

Para poder ver el resultado de nuestro código, podemos usar cualquiera de los diferentes navegadores con los que contamos en la actualidad:

- Google Chrome
- Mozilla Firefox
- Internet Explorer
- Safari
- Opera
- Otros

En un principio cuando se comenzó a definir el estándar HTML los navegadores eran el principal problema para HTML, ya que no todos interpretaban las etiquetas de la misma forma, en muchos casos la misma etiqueta se interpretaba de diferente forma por un navegador, durante mucho tiempo fue uno de los retos para los desarrolladores, todos querían crear un sitio web que se viera de la misma forma en todos los navegadores, este era el reto principal en el momento de crear un sitio, ya que en ocasiones se tenía que implementar código específico para cada navegador.

En la actualidad podemos encontrar problemas de compatibilidad entre navegadores pero es algo que día a día disminuye y HTML5 ayuda a que esta brecha de compatibilidad sea más corta.

Editor de texto

Para crear un sitio web, en principio solamente necesitamos 2 cosas: conocimientos en HTML y un editor de texto. La elección de este último es una de las principales preguntas en el momento de iniciarnos en el mundo del desarrollo.

Es importante dejar claro que el editor de texto nunca te va a hacer mejor o peor desarrollador, tampoco va ayudarte a cobrar más por una web. El editor que seleccionemos solamente es una herramienta, pensemos en un medio de transporte: podemos cruzar Europa si utilizamos una bicicleta, también podemos usar un coche o un camión. Diremos que siempre va a ser mejor cruzarlo con "x" transporte, pero en caso de querer ir a un supermercado a la vuelta de nuestro hogar podremos encontrar más factible el uso de la bicicleta. Con este ejemplo quiero demostrar que no siempre el mismo editor será el mejor, esto dependerá de la situación y las circunstancias, por ejemplo: si nuestro desarrollo se combina con otros lenguajes de programación, si contamos con la suficiente memoria libre para ejecutarlo, si contamos con el sistema operativo que se necesita, etc.

El editor de texto que se va a usar a lo largo de este libro se deja a libre criterio del lector. Personalmente puedo comentar que yo utilizo las siguientes combinaciones: si trabajo en Windows utilizo un editor de Notepad++. En caso de trabajar en Mac OSx uso Komodo, en el caso de Linux, Komodo. En caso de necesitar combinar HTML con lenguajes como Java u otros lenguajes, utilizo el IDE Eclipse en Windows y Linux, en el caso de Mac OS uso el Xcode proporcionado por Apple.

Si el término de editor de texto aún no queda claro, basta con revisar el bloc de notas o *"Notepad"* que vienen en cualquier sistema operativo. Ese es un editor de texto, podríamos decir que con el bloc de notas es más que suficiente para crear un sitio web.

La pregunta es: ¿por qué no usamos todos el bloc de notas? También se responde con el mismo bloc de notas, sabemos que podemos escribir una carta con el bloc de notas, pero ¿por qué usamos Microsoft Word u otro procesador?

Respuesta: Utilizamos otros editores que cuentan con herramientas especiales o de más fácil manejo, sabemos que el bloc de notas contiene lo básico, pero en caso de querer realizar cosas avanzadas se nos queda pequeña la herramienta. Con esto estamos sustentando que los editores especiales para escribir código tienen herramientas que van a hacer más rápido, ágil y dinámico el desarrollo de nuestros sitios y debemos de conocer nuestras necesidades.

Mi recomendación hoy en día es no casarse con el uso de un editor, ya que el día de mañana puede que este editor simplemente desaparezca o que aparezca uno mejor, además de que si trabajamos en una empresa la mayoría de estas optan por una tecnología y sería bastante desafortunado perder una oportunidad laboral solamente por el hecho de no poder adaptarnos a "X" herramienta.

Para más información de los editores comentados, se puede visitar la web oficial de algunos de ellos:

Notepad++

La web es: https://notepad-plus-plus.org/

Este editor es completamente gratis y se pueden hacer donaciones si así se desea, está escrito en C++ y es compatible con Windows sin ningún problema, es de los editores más antiguos que podemos encontrar. Además, considero que es de los que más lenguajes soporta.

Komodo EdiT

La web es: http://komodoide.com/komodo-edit/

Este editor es gratis y se puede utilizar sin ningún problema, cuenta con versiones de pago que añade funciones al mismo, además de contar con su IDE. Este editor tiene versiones para distintos sistemas operativos, en caso de acostumbrarse puede ser útil esta característica, personalmente creo que es de los más potentes que conozco, además de que da estabilidad al desarrollo.

Sublime Text

La web es: http://www.sublimetext.com/

Es un editor de texto que la comunidad de desarrolladores adoptó de manera muy grata, contiene soporte para la mayoría de los lenguajes de programación actuales, aunque tiene un costo que te permite utilizarlo gratis sin límite de tiempo. Esta es una oportunidad para probarlo y conocerlo al 100%. Es uno de los editores de texto con una comunidad de tamaño considerable y esto implica que encontrarás en ella fácilmente ayuda cuando tengas problemas con el editor.

Editores de texto en línea

Con la tendencia de los últimos años sobre el trabajo con "la nube", el área de la programación no se queda atrás. Sabemos que contamos con una gran cantidad de editores y entornos de desarrollo para trabajar, pero actualmente contamos con herramientas que van un paso más adelante y nos proporcionan la posibilidad de trabajar desde cualquier lado con solamente tener una cuenta (no en todos es necesarios) y estar conectados a internet.

Esto es una ventaja ya que en un cambio imprevisto de oficina, de equipo o tener la intención de compartir el código sin tener que estar físicamente en el mismo lugar, no será un dolor de cabeza.

Los editores de código en línea están abriendo una nueva tendencia en el área de desarrollo, en los siguientes párrafos describimos las características de los que consideramos hoy en día están listos para trabajar con ellos.

Codepen

La web es: http://www.www.codepen.io

Este editor funciona en cualquier navegador habitual en el mercado, la mayoría de los desarrolladores que lo usan es para trabajar con HTML, CSS y JS. El diseño es simple y permite encontrar las funcionalidades de manera rápida. Tiene una

funcionalidad bastante atractiva que permite ver y editar al mismo tiempo un archivo HTML, CSS y JS, mostrando el resultado de la combinación de ellos en tiempo real.

Code AnyWhere

La web es: https://codeanywhere.com/

Esta herramienta es tan completa que se considera no solamente un editor de texto, sino también entra en la categoría de IDE (algo más avanzado). Una de sus principales características es que en el caso de trabajar con PHP o código del lado del servidor (la programacion relacionada con páginas web puede estar en el lado del cliente "el navegador" o del lado del servidor "servidores que contienen la página"), nos permite realizar estas funciones sin necesidad de instalar algo, como, por ejemplo, iniciar un servidor Apache. También contamos con una consola para ciertas necesidades.

JS Fiddle

La web es: http://jsfiddle.net/

Este editor está muy vinculado con JavaScript, ya que incluye soporte para librerías, frameworks y todo lo relacionado con este lenguaje de programación, permite la colaboración en línea con solamente un enlace y se podría trabajar entre un grupo de personas simultáneamente.

W3C

W3C es un consorcio internacional en el que se desarrollan los estándares de la web, la cabeza del grupo es Tim Berners-Lee, además de CEO. El objetivo principal es llevar a su máximo potencial todas las tecnologías. Uno de los últimos objetivos en que se está trabajando es que todo el mundo pueda acceder a la web desde cualquier dispositivo.

Además de este grupo, existe otro que está tomando a manera personal el desarrollo de HTML, el grupo es *WHATWG*. Se creó en el año 2004 como un esfuerzo en el que se buscaba que la W3C se interesara por el estándar HTML; el grupo está

conformado por gente de varias empresas como Apple, Mozilla, Opera y otras. Actualmente son muy activos en la definición del estándar HTML. Para más información se puede consultar la web oficial:

https://whatwg.org/

2 HTML 5

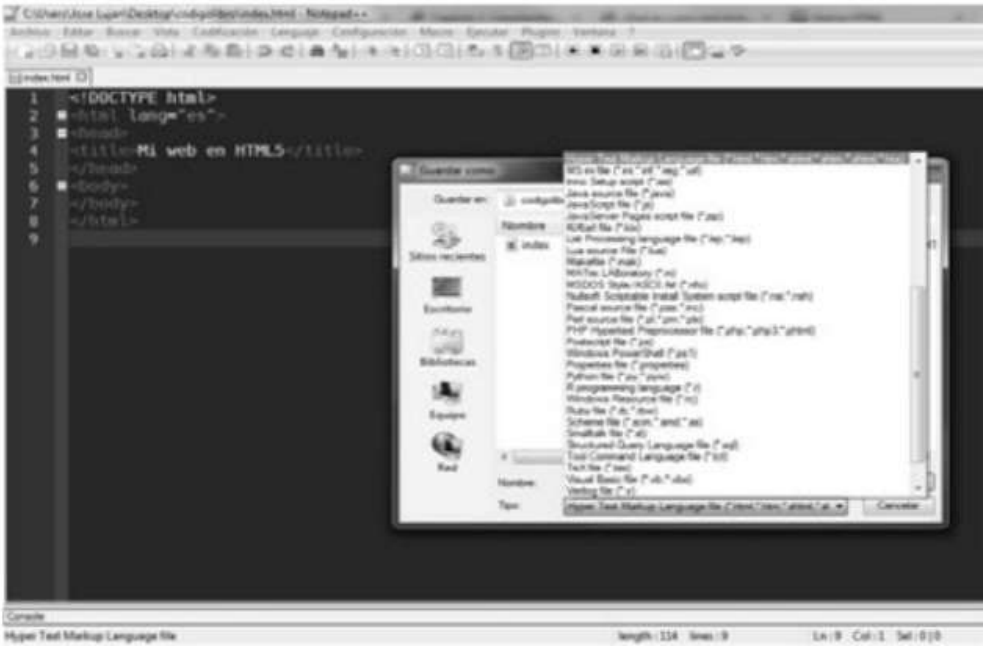
Es importante mencionar que en este capítulo no se trabajará nada de diseño, ya que en el capítulo de CSS se tratará a fondo el tema, así que si deseamos ver algunos elementos utilizados en este capítulo, la información se dará en el capítulo siguiente.

Durante mucho tiempo se comentó que HTML5 era el futuro de la web. Hace un par de años, esta frase sufrió un cambio y ahora se puede decir que HTML5 es la web. En este capítulo abordaremos HTML5 a fondo para conocer cómo se hacen las webs actuales y cómo lograr que nuestro sitio web cumpla con el estándar.

Crear un archivo HTML5

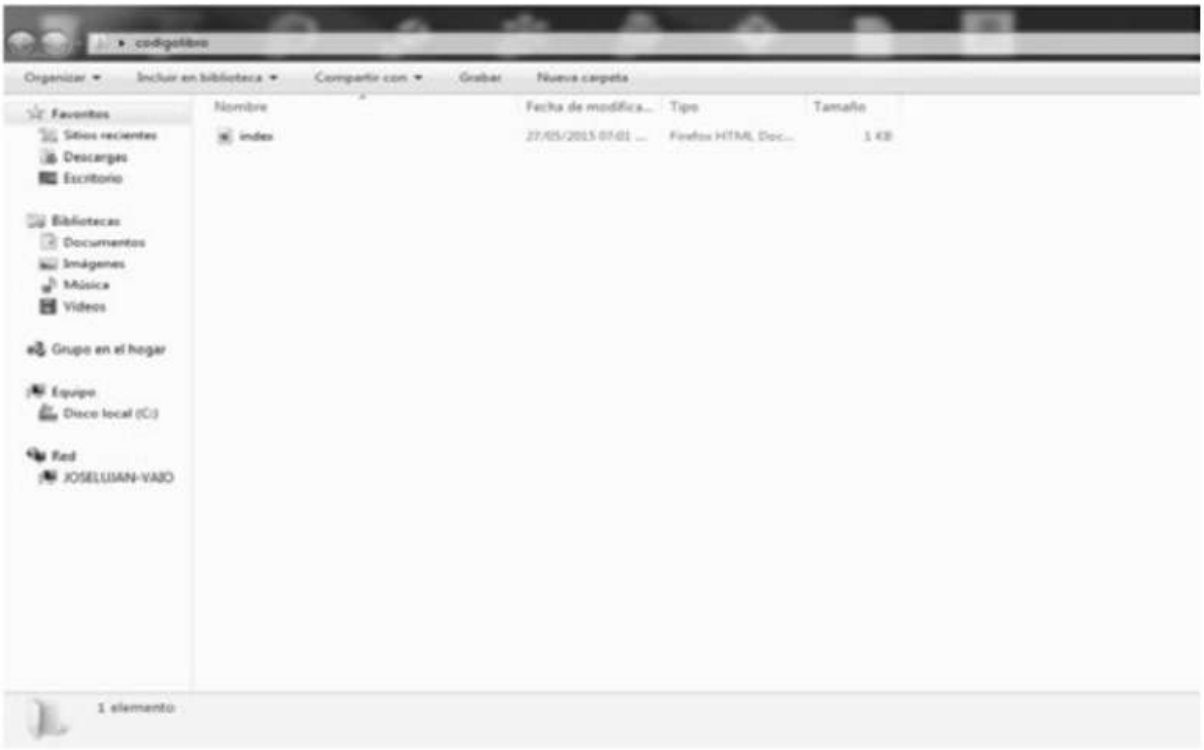
Para crear un archivo HTML5 o HTML, solo tenemos que abrir un editor de texto, seleccionar *Guardar como*, como la siguiente imagen:

Después, seleccionamos la extensión del archivo que sea “.html” como en la siguiente imagen.



Como recomendación podemos designar “index” al archivo, con ese nombre indicamos que es nuestra página principal, pero podríamos ponerle cualquier otro como: “archivo.html” o “pepito.html”, en el momento de guardar el archivo vamos a

ver que nuestro archivo toma el icono del navegador que tenemos por defecto seleccionado para nuestro equipo. Yo uso Firefox y se ve como en la siguiente imagen:



Si el archivo ya tomó el icono por defecto de nuestro navegador, es que ya tenemos un archivo HTML, aunque esté vacío.

Estructura básica de HTML5

La estructura básica de una web en HTML5 podría ser la siguiente:

```
<!DOCTYPE html>

<html lang=”es”>

<head>

    <title>Mi web en HTML5</title>

</head>
```

```
<body>

</body>

</html>
```

Al código anterior le faltan etiquetas que podemos considerar básicas en HTML5, pero como vemos no son elementales para crear una web. Examinemos el código.

Etiqueta <!DOCTYPE>

```
<!DOCTYPE html>
```

Doctype es lo primero que escribimos, desde versiones anteriores de HTML el *Doctype* ocupa el primer lugar de un documento HTML, en él declaramos el tipo del documento. Antiguamente el *Doctype* era muy largo y contenía mucha información, era algo como esto:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://w3.org/TR/html4/strict.dtd">
```

En esta línea podíamos leer la organización, el tipo, el nombre, el idioma, la URL, la disponibilidad, etc. Cualquier guion, coma, diagonal tiene un significado, así que se debe tener cuidado con cada carácter que se coloca.

En HTML5 todo se simplificó, así que lo podemos recordar de una manera muy sencilla, colocando un DTD.

El DTD *(document type definition*) es una definición del tipo de documento. No es lo mismo tener un archivo HTML, XHTML, XML y otros que existen, lo tenemos que indicar con DTD.

Es importante colocar en *doctype* el DTD que estamos utilizando, además de indicar el tipo de documento, la estructura y las etiquetas que podemos usar, asimismo tiene relación con eventos.

Otros tipos de DTD son:

```
HTML 4.01 Strict

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">



HTML 4.01 Transitional

!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"

  "http://www.w3.org/TR/html4/loose.dtd">



HTML 4.01 Frameset

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN"

  "http://www.w3.org/TR/html4/frameset.dtd">
```

Para conocer y tener más información, podemos verificar esta web de la W3C:

http://www.w3.org/QA/2002/04/valid-dtd-list.html

Etiqueta <html>

```
<html lang=”es”>
```

Esta etiqueta es *<html>* pero a diferencia de otras etiquetas podemos ver que tiene más texto a su lado. El texto *lang=”es”* se le considera un atributo y a este atributo se le está asignando un valor.

Para entender de mejor forma esto, debemos comprender que el atributo es *lang* y siempre que veamos que se *está* utilizando el símbolo “=” estaremos viendo una asignación de valor. Este es solo uno de los atributos que nos vamos a encontrar

dentro de las etiquetas, lo importante de los atributos es que no debemos de olvidar que siempre se colocan en la etiqueta de apertura.

El valor “es” que estamos colocando significa que esta web está en el idioma español, en el caso de estar en el idioma inglés cambiaría el valor por "en", cada idioma tiene un valor diferente, así que va a depender del lenguaje que usemos el valor que estemos por asignar, por ejemplo otras etiquetas son:

Italiano	it
Japonés	ja
Portugués	pt
Ruso	ru
Turco	tr
Chino	zh
Danés	da

Si te interesa conocer los códigos para cada lenguaje, puedes consultarlos en el siguiente enlace:

http://www.w3schools.com/tags/ref_language_codes.asp

Etiqueta <head>

La etiqueta *<head>* es una de las más antiguas de HTML, hace muchos años trabajamos con ella de la misma forma. Dentro de la etiqueta se colocan algunos elementos de vital importancia para un sitio, por ejemplo el título, la descripción del sitio, sus estilos y otros.

```
<head>
    <title>Mi web en HTML5</title>
</head>
```

En nuestro caso estamos colocando la etiqueta de título con el texto “Mi web en HTML5” para que nuestro sitio lo muestre.

La mayoría de las etiquetas que podemos agregar dentro de la etiqueta *<head>* son invisibles a primera vista al usuario, ya que ellas están configuradas para el navegador y para que este entienda algunas cosas o simplemente son para dar indicaciones.

Etiqueta <body>

A diferencia de la etiqueta *<head>,* la etiqueta *<body>* es la que marca la pauta de la parte “visible” de un sitio web. Todo lo que va ahí se va a ver reflejado en el navegador en primera instancia. En nuestro caso no contamos con nada de contenido por eso nuestra web estaría en este momento en blanco.

```
<body>

</body>
```

Al hacer doble clic al archivo y abrirlo en nuestro navegador por defecto, podemos ver una web como la siguiente:

Nuestra web está vacía, pero podemos ver el título en la parte superior en donde se encuentran las pestañas y notamos que tiene el título que le indicamos: Mi web en HTML5.

Etiquetas básicas

Existen muchas etiquetas, pero podemos considerar a un grupo de ellas como las elementales para que un documento HTML funcione correctamente.

Conozcamos algunas de ellas:

```
<head>
<meta charset=”utf-8”>
<meta name=”author” content=”José Luján” >
<meta name=”description” content=”Esta es un ejemplo básico con HTML5”>
<meta name=”keywords” content=”Ejemplo HTML5, Hola mundo HTML5”>
<link rel=”stylesheet”  href=”estilo.css”>
</head>
```

Etiquetas <meta>

```
<meta charset=”utf-8”>
<meta name=”author” content=”José Luján” >
<meta name=”description” content=”Esta es un ejemplo básico con HTML5”>
<meta name=”keywords” content=”Ejemplo HTML5, Hola mundo HTML5”>
```

En el documento nuevo de HTML5 encontramos varias etiquetas *<meta>,* estas etiquetas son consideradas como información específica para el navegador o los buscadores. La mayoría de ellas no se mostrarán en el cuerpo de la página, pero sí en

los resultados de búsquedas de un navegador o se utilizan para configurar el documento HTML.

Las etiquetas *<meta>* tienen al igual que otras la característica de no tener que cerrarse, se dice que son consideradas etiquetas abiertas o vacías.

Etiqueta <meta charset>

```
<meta charset=”utf-8”>
```

Charset lo podemos entender como codificación de caracteres en el documento, en esta etiqueta indicamos la codificación de los caracteres que se utilizan. Si no seleccionamos la correcta, encontraremos características exclusivas que no se podrán mostrar al usuario, como, por ejemplo, los acentos.

Para tener habilitados todos los caracteres de un teclado en español, como la letra “ñ”, los acentos en vocales, debemos de usar el charset “utf-8”.

Etiqueta <meta name=”author”>

```
<meta name=”author” content=”José Luján” >
```

En esta etiqueta podemos colocar información sobre el autor del documento que estamos viendo.

Etiqueta <name=” description”>

```
<meta name=”description” content=”Esta es un ejemplo básico con HTML5”>
```

Esta etiqueta nos permite colocar una descripción del contenido del documento, por ejemplo si en este documento estamos hablando sobre coches deportivos, se espera que coloquemos una descripción o resumen sobre los tópicos específicos que se ven en nuestro documento.

Cuando se realiza una búsqueda podemos ver en los resultados que se arrojan un pequeño texto debajo del título de la web o de la página, este texto es el que se coloca en la descripción, así que si colocamos una descripción errónea del contenido, probablemente confundamos al usuario o simplemente no entre.

Etiqueta <name="keywords">

```
<meta name="keywords" content="Ejemplo HTML5, Hola mundo HTML5">
```

Esta etiqueta nos permite colocar palabras claves que hacen referencia al contenido de nuestro documento, se relaciona en gran medida con las palabras que coloca cualquier usuario en un navegador, cuanto más similar sean las palabras claves que coloquemos en "keywords" a las que el usuario escribe para realizar una búsqueda, más aumentarán las probabilidades de aparecer en estos resultados. Es importante mencionar que esto no implica que si las palabras claves son idénticas a una búsqueda que hace un usuario, vamos a aparecer como resultado, ya que hay muchos otros factores que debemos considerar, pero sí es un buen comienzo.

Etiqueta <link rel>

```
<link rel="stylesheet"  href="estilo.css">
```

Esta etiqueta nos permite utilizar archivos externos, en la mayoría de los casos se enlaza con archivos que contienen estilos, también conocidos como archivos CSS o archivos de Cascada de estilos.

Etiquetas obsoletas para HTML5

En HTML como en cualquier estándar se van realizando cambios y esto implica que algunas cosas cambian para realizarse de una manera más sencilla, más rápida o simplemente mejor. Con la llegada de HTML5 veremos que algunas etiquetas que se usaban habitualmente ya no se están usando o se consideran como obsoletas, conozcamos estas etiquetas:

Etiqueta	Significado
Acronym	En esta etiqueta explicamos los acrónimos que colocamos en el documento.
Applet	En esta etiqueta se coloca un script adentro que contenía código, normalmente se usaba mucho con el lenguaje Java, desde la versión HTML5 se utiliza la etiqueta object.
Basefont	Nos permite determinar el tamaño de fuente predeterminado.
Big	Nos permite mostrar un texto en tamaño grande.
Center	Nos permite centrar un elemento.
Dir	Nos permite insertar una lista de directorios.
Font	Nos permite establecer un estilo a un texto.
Frame	Nos permite insertar un marco en la web.
Frameset	Nos permite colocar un grupo de marcos.
Noframe	Nos permite añadir contenido alternativo para el marco.
Strike	Nos permite tachar el texto.
Tt	Nos permite colocar el texto como teletype.
U	Nos permite subrayar texto.
Xml	Nos permite definir un texto preformateado.

Para estar al tanto de las etiquetas que se van acumulando en esta lista de "obsoletas", una fuente muy fiable es la que proporciona Mozilla:

https://developer.mozilla.org/en-US/docs/Web/HTML/Element

Atributos en HTML5

Algunas de las etiquetas de HTML pueden tener uno o más atributos. Los atributos nos sirven para especificar un valor a la etiqueta que lo contiene. La sintaxis de los atributos es la siguiente:

```
<etiqueta atributo="valor">
```

En el caso de colocar más atributos, hacemos algo como lo siguiente:

```
<etiqueta atributoA="valor" atributoB="valor" atributoC="valor">
```

Lo que tenemos que hacer es escribir la etiqueta y cada atributo separarlo por un espacio; además, los atributos siempre los tenemos que colocar en la etiqueta de apertura.

Vamos a conocer algunos atributos.

Atributo id

El atributo "id" nos permite asignar un identificador, esto significa que este "id" debe ser único y nos sirve para distinguirlo. Además, colocando un "id" podemos tener acceso utilizando el "id" desde el CSS o Javascript.

Ejemplo:

```
<p id="nombre"> José Luján</p>

<p id="twitter">@josedlujan</p>

<p id="correo">josedlujan@gmail.com</p>
```

Como recomendaciones para colocar un “id”, debemos de considerar lo siguiente:

- No se permiten caracteres especiales.
- Deben ser únicos.
- Utilizar las letras del abecedario y números enteros.

Atributo dir

El atributo *dir* habilita la dirección de los elementos en nuestra página web, así podremos colocar un elemento a la derecha o a la izquierda de otro elemento. El detalle que debemos recordar es que tenemos 2 valores que podemos utilizar:

- RTL – coloca el elemento de derecha a izquierda.
- LTR – coloca el elemento de izquierda a derecha.

Por defecto si no indicamos nada, el valor que se tomará es LTR.

Analicemos el siguiente ejemplo:

```
<!DOCTYPE html>

<html lang="es">

<head>

<title>Ejemplo de atributo dir </title>

</head>

<body >

    <p dir="rtl"> Primer texto </p>

    <p dir="ltr"> Segundo texto </p>

</body>

</html>
```

El primer párrafo que encontramos lo colocamos a la derecha:

```
<p dir="rtl"> Primer texto </p>
```

El segundo párrafo lo colocamos a la izquierda:

```
<p dir="ltr"> Segundo texto </p>
```

El resultado del código se vería de la siguiente forma:

Atributo class

Este atributo nos va a permitir asignar el nombre de una clase a un elemento seleccionado, cuando le colocamos el atributo *"class"* a un elemento decimos que este pertenece ya a una clase, no hay un límite de elementos a los que se les puede asignar a una clase, es decir, que a todos los elementos que lo permitan les podemos asignar una.

Esto es muy similar a lo que sucede con la programación orientada a objetos, una clase puede tener muchas clases hijo, en este caso una clase puede tener muchos elementos.

Por ejemplo:

```
<!DOCTYPE html>
<html lang=”es”>
<head>
    <style>
    .claserojo {color:red}
    .claseazul {color:blue}
    </style>
<title>Ejemplo de clases</title>
</head>
<body>
        <p class="claserojo">Esto es rojo</p>
    <p class="claseazul">Esto es azul.</p>
</body>
</html>
```

Ahora estamos definiendo las clases dentro del archivo HTML5, aunque más adelante lo haremos en el archivo CSS:

```
<style>
    .claserojo {color:red}
    .claseazul{color:blue}
</style>
```

En las líneas anteriores definimos dos clases: claserojo y claseazul, cada una de ellas tiene un color diferente dependiendo del nombre.

Tenemos dos párrafos y a cada uno le asignamos una clase diferente:

```
<p class="claserojo">Esto es rojo</p>

<p class="claseazul">Esto es azul.</p>
```

Esto implica que cada uno se va mostrar diferente ya que dependen de clases distintas.

Si agregáramos un tercer párrafo, podremos notar que no existe problema en el momento de querer reusar la clase:

```
<p class="claserojo">Esto es rojo</p>

<p class="claseazul">Esto es azul.</p>

<p class=”claserojo”>Otro texto en rojo</p>
```

El resultado de este código sería algo como lo siguiente:

Atributo style

Con este atributo podemos habilitar la opción de colocar estilo de forma directa sobre el elemento que lo contenga, actualmente no es recomendable realizarlo de esta forma, ya que como veremos más adelante lo ideal es tener el estilo separado en otro archivo.

El ejemplo de *style* sería el siguiente:

```
<!DOCTYPE html>

<html lang="es">

<head>

<title>Ejemplo de estilos </title>

</head>

<body style="background-color:gray">

      <p style="color:red"> Testo </p>

</body>

</html>
```

En el código anterior hemos usado en dos ocasiones el atributo *style*, la primera en la etiqueta <*body*> y le hemos indicado que el color de fondo sea gris:

```
<body style="background-color:gray">
```

En la segunda ocasión le hemos indicado de forma directa a la etiqueta nativa <p> que tome el color rojo:

```
<p style="color:red"> Texto </p>
```

Atributos obsoletos

Los atributos obsoletos no siempre son eliminados, en muchos casos los atributos solo se usan dependiendo de la etiqueta, con esto queremos decir que se puede seguir utilizando en la etiqueta “X”, pero en la etiqueta “Y” ya no se debe de usar. A continuación, hacemos una breve lista de los atributos y en qué elementos se hacen las recomendaciones.

Etiqueta	En qué elementos se elimina
Abbr	En los elementos td y th
Acceskey	En los elementos a, area, button, input, label, legend, textarea
Align	En todos
Alink, link, text, vlink	En el elemento body
Archive	En el elemento object
Axis	En los elementos td y th
Background	En el elemento body
Bgcolor	En los elementos table, tr, td, th y body
Border	En todos
Cellpadding	En el elemento table
Cellspacing	En el elemento table
Char, charoff	En los elementos col, colgroup, tbody, td, tfoot, th, thread, tr
Charset	En los elementos link y a
Classid	En el elemento object

Clear	En el elemento br
Codebase	En el element object
Codetype	En el element object
Compact	En los elementos dl, menu, ol y ul
Coords	En el elemento a
Declare	En el elemento object
Frame	En el elemento table
Frameborder	En el elemento iframe
Height	En los elementos td y th
Hspace	En los elementos img y object
Language	En el elemento script
Longdesc	En los elementos img y frame
Marginheight	En el elemento iframe
Marginwidth	En el elemento iframe
Name	En los elementos img y a
Nohref	En el elemento area
Noshade	En el elemento hr
Nowrap	En los elementos td y th
Profile	En el elemento head
Rev	En los elementos link y a

Rules	En el elemento table
Scheme	En el elemento meta
Scrolling	En el elemento iframe
Scope	En el elemento td
Shape	En el elemento a
Size	En el elemento hr
Standby	En el elemento object
Summary	En el elemento table
Target	En el elemento link
Type	En los elementos li, ol y ul
Valign	En los elementos col, colgroup, tbody, td, tfoot, th, thead, tr
Valuetype	En el element param
Version	En el element html
Vspace	En los elementos img y object
Width	En los elementos hr, table, td, th, col, colgroup, pre

HTML5 y el soporte de los navegadores

El crecimiento de la tecnología se acelera cada día y en la web lo vemos de forma más notoria con los avances que se muestran con HTML5. En estos casos la compatibilidad va de la mano no solamente de HTML, sino también podríamos agregar tecnología como CSS, API's y todas las que tienen relación con la web.

Para el desarrollador de sitios web un reto importante que se le presenta día a día es hacer que el sitio web desarrollado se vea y funcione de la misma forma en todos los navegadores posibles. Cuando hablemos de navegadores en el libro, estamos haciendo referencia a:

- Internet Explorer
- Google Chrome
- Mozilla Firefox
- Safari

Algo que no se debe olvidar es que no todas las características que nos ofrece HTML5 y las tecnologías mencionadas anteriormente son admitidas por los navegadores, por eso es importante consultar y conocer cuáles sí y cuáles no, ya que si damos por hecho que todas funcionan de igual forma y son parte vital de nuestro sitio, podríamos tener un peligroso resultado.

En relación con mi experiencia, uno de los sitios más completos para mantenernos informados sobre el soporte de elementos es HTML5 Test. Este sitio proporciona el servicio en una escala de puntuación, lo que se mide principalmente es el soporte a HTML5 en cada navegador.

Para comprobar su funcionamiento en el navegador tenemos que realizar la prueba desde el navegador que deseamos para evaluar el sitio, por ejemplo podemos evaluar el sitio web desde Explorer (Microsoft), Firefox (Mozilla), Safari (Apple), Chrome (Google).

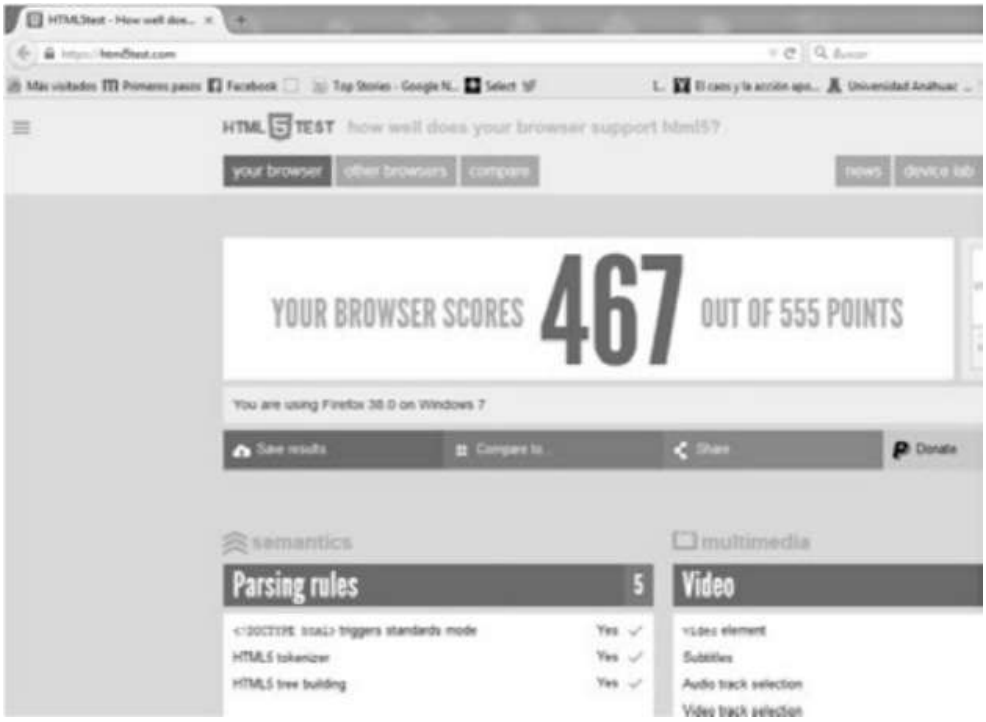
https://html5test.com/

Ya dentro del sitio esperaremos unos segundos (casi nada) y veremos un marcador en la parte superior del sitio, este marcador puede alcanzar como máximo 555 puntos. En caso de alcanzar el marcador 555 estaríamos evaluando a nuestro navegador con la calificación más alta.

Es importante mencionar que hasta el día de hoy, día en el que se escribe este libro, ningún navegador ha logrado alcanzar la máxima puntuación. Para esta

evaluación podemos revisar uno por uno los elementos que se toman en cuenta, por ejemplo: vídeo, audio, compatibilidad con redes p2p, gráficos 2d, gráficos 3d, seguridad, archivos, historial de navegación, localización, animaciones, microdata, entre otros.

La pantalla de resultados que deberíamos de ver es como la siguiente:



Estructura de un documento HTML5

HTML5 se considera como el cambio más importante que ha sufrido el estándar HTML desde sus inicios, durante versiones anteriores las etiquetas que desempeñaron un papel importante eran la etiqueta *<table>*, que después asumió el papel protagonista en los documentos HTML, y la etiqueta * .*

Las dos etiquetas tenían un objetivo: mostrar y organizar la información de la web como el texto, las imágenes y todo lo que podíamos colocar.

La etiqueta *<table>* nos permite crear una tabla en la que, dependiendo de la posición de filas y columnas, desplegamos de manera rápida y sencilla la información, en su momento fue algo que cambió el desarrollo de sitios ya que la velocidad de estructurar un documento aumentó de manera considerable. Después se encontraron algunos límites y se comenzó a abandonar como primera opción. Hoy en día se sigue utilizando pero con otros objetivos, por ejemplo: en el momento de crear un archivo HTML para enviarlo como *mailing* para una campaña publicitaria.

La etiqueta * * que vino a cubrir la necesidad de dinamismo que era bloqueada por la etiqueta *<table>* les enseñó a los desarrolladores que existía otra forma de trabajar. El comenzar a utilizar *HTML + CSS + Javascript* fue la ecuación que colaboró a que la etiqueta * * tomara el papel protagonista dentro de un documento HTML.

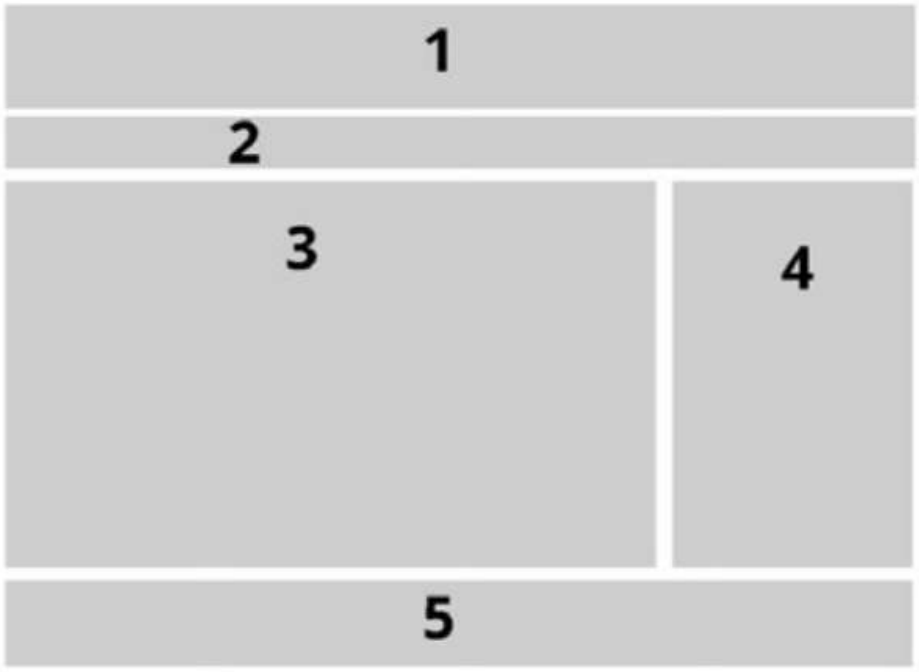
El principal problema de la etiqueta * * es que es genérica, cualquier elemento podría ser una etiqueta * * con solo colocarlo, podríamos tener imágenes, enlaces, textos, en resumen casi cualquier elemento. Esto nos limitaba a saber con exactitud qué tipo de elemento es el que teníamos y cómo se puede tratar para aprovechar la naturaleza del elemento.

Podemos decir que técnicamente la etiqueta solo se implementaba para dividir el espacio que se iba a usar, con la etiqueta *<table>* pasaba lo mismo, no contábamos con más información de ella, el contenido o de cuál era su objetivo.

Aquí es donde viene una de las frases que más vamos a escuchar: “HTML5 no son solo etiquetas nuevas, es una estructura adecuada para un documento en la web”. HTML5 trae para nosotros nuevas etiquetas, pero con el objetivo principal de saber qué es la información que se encuentra en ella, además de jerarquizar la importancia de la información, así podremos saber qué información es más importante mostrar, cuál puede esperar y que los documentos sigan un estándar con relación a una estructura.

Estructura general

La siguiente imagen representa la estructura general que se usa para un sitio web:



1. Sería tomado en cuenta como el encabezado o también llamada cabecera, aquí se coloca el logo de la página web, unido a la frase de la compañía o una breve descripción.

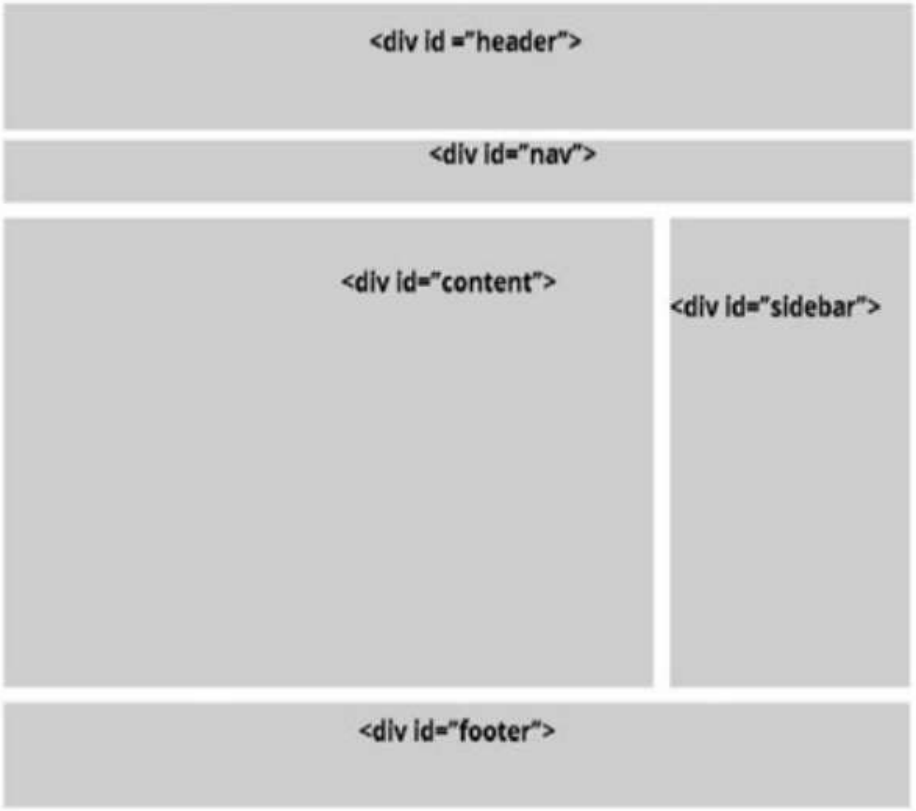
2. Se tomaría como la barra de navegación o menú, aquí es donde vemos las otras categorías u opciones que tenemos para movernos dentro del mismo sitio web.

3. Información o contenido principal, a veces esta sección toma el espacio de la barra lateral y crece por todo el ancho del sitio web, también se puede dividir en columnas o por filas, esta es la sección que más cambia porque depende de que tanto contenido muestre el sitio que estamos creando.

4. Barra lateral, normalmente mostramos enlaces que tienen relación con el contenido principal, también pueden ser enlaces a otro sitio web, puede ser un complemento para el contenido más importante.

5. Pie de página, en esta sección encontramos la información del sitio, por ejemplo: de quién es la web, quién la creó, los términos y las condiciones, las formas de contacto, la ubicación y otros datos de este tipo.

En HTML4 lo que se hacía normalmente era usar la etiqueta y se vería así:

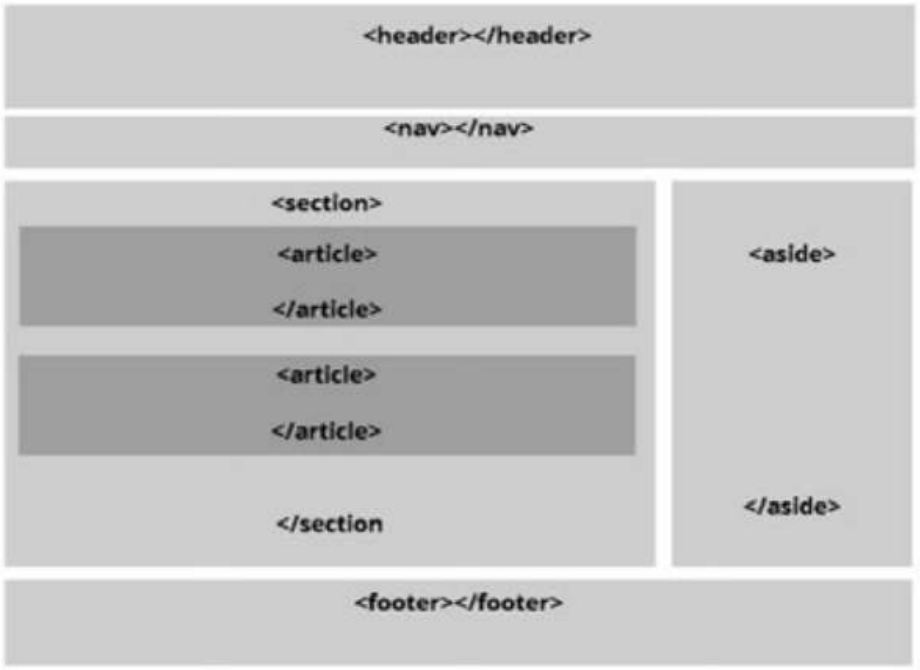


Todas las etiquetas anteriores estarían dentro de la etiqueta *<body>.*

Estructura general en HTML5

La estructura general presentada anteriormente se puede considerar como solamente "reservar espacio" para las secciones, pero sin ninguna jerarquía que nos marque una pauta entre ellas, todo lo que colocamos no nos indica nada, nos guiamos por el nombre o el identificador que colocamos para saber de qué trata esa sección. Esto cambia en HTML5 en donde se nos proporcionan etiquetas para cada sección con un significado y una jerarquía.

La estructura en HTML5 quedaría como la siguiente imagen:



Ahora vamos a analizar estas nuevas etiquetas que nos proporciona HTML5, no olvidemos que estas etiquetas se encuentran dentro de la etiqueta *<body>.*

Etiqueta <header>

Esta etiqueta tiene el objetivo de ser una cabecera, nos va a servir para colocar el logo y otros títulos, es importante no confundir con la etiqueta *<head>*, ya que esta etiqueta se mantiene y cumple una función completamente diferente a *<header>.* Podríamos colocar los títulos y quedar así:

```
<!DOCTYPE html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>Título de la web</title>
</head>
```

```
<body >
    <header>
        <h1>Título de la página</h1>
    </header>
</body>
</html>
```

Etiqueta <nav>

La etiqueta *<nav>* la utilizamos para colocar enlaces a otras páginas, es una sección de navegación en donde podemos encontrar la forma de ir a otras partes dentro del sitio web. La idea principal es que la sección la utilicemos como un menú.

La etiqueta la usaríamos de la siguiente forma:

```
<!DOCTYPE html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>Título de la web</title>
</head>
<body >
    <header>
        <h1>Título de la página</h1>
    </header>
```

```
        <nav>
            <ul>
                <li>Inicio</li>
                <li>¿Quiénes somos?</li>
                <li>Servicios</li>
                <li>Contacto</li>
            </ul>
        </nav>
    </body>
    </html>
```

Etiqueta <section>

Esta etiqueta nos permite colocar la información más relevante del sitio; su diseño es muy variado. Se puede tener cualquier cantidad de secciones dentro de la misma, columnas y otras características. El punto principal es crear bloques con contenido dependiendo de las necesidades.

Si agregamos la etiqueta, el código quedaría así:

```
<!DOCTYPE html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>Título de la web</title>
```

```
</head>
<body >
    <header>
        <h1>Título de la página</h1>
    </header>
    <nav>
        <ul>
            <li>Inicio</li>
            <li>¿Quiénes somos?</li>
            <li>Servicios</li>
            <li>Contacto</li>
        </ul>
    </nav>
    <section>

    </section>
</body>
</html>
```

Etiqueta <article>

Esta etiqueta es una de las que demuestra la importancia que le da HTML5 a la semántica, con *<article>* podemos distribuir esta semántica, cada etiqueta *<article>*

que coloquemos es independiente de la otra, así podremos agrupar los contenidos dependiendo de lo que necesitemos.

La etiqueta *<article>* se puede considerar como un camaleón, ya que puede tomar diferentes caras dependiendo de la estructura y del tipo de sitio web que estemos creando; por ejemplo, puede ser la entrada de un blog, una noticia, un comentario u otra cosa.

Por ejemplo: si tenemos una web que es de noticias, el contenedor que tiene las noticias sería la etiqueta *<section>* y cada noticia que vamos agregando a esta etiqueta podría ser una *<article>*.

El código sería el siguiente:

```
<!DOCTYPE html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>Título de la web</title>
</head>
<body >
    <header>
        <h1>Título de la página</h1>
    </header>
    <nav>
        <ul>
            <li>Inicio</li>
```

```
                    <li>¿Quiénes somos?</li>
                    <li>Servicios</li>
                    <li>Contacto</li>
                </ul>
        </nav>
        <section>
                <article>
                    Primera entrada
                </article>
                <article>
                    Segunda entrada
                </article>
        </section>
    </body>
    </html>
```

Etiqueta <aside>

Esta etiqueta la usamos para definir una barra lateral, en semántica la información que contiene esta etiqueta es de menos importancia que la de *<section>.* Esta información va a tener relación o complementa a la sección principal. Técnicamente podemos decir que es información secundaria.

El código sería el siguiente:

```
<!DOCTYPE html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>Título de la web</title>
</head>
<body >
    <header>
        <h1>Título de la página</h1>
    </header>
    <nav>
        <ul>
            <li>Inicio</li>
            <li>¿Quiénes somos?</li>
            <li>Servicios</li>
            <li>Contacto</li>
        </ul>
    </nav>
    <section>
        <article>
            Primera entrada
```

```
            </article>
            <article>
                Segunda entrada
            </article>
        </section>
        <aside>
            Complemento de información
        </aside>
    </body>
    </html>
```

Etiqueta <footer>

Esta etiqueta podría definirse como la última etiqueta de la jerarquía nueva que se propone en HTML5, esta etiqueta, al ser de pie página en la mayoría de los casos, contiene información de la web, de la compañía, de la empresa y del blog. Específicamente podríamos colocar cómo contactarnos, los derechos de la información, la dirección y el tipo de información que podemos dar a conocer de nuestro sitio.

El código sería el siguiente:

```
<!DOCTYPE html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>Título de la web</title>
</head>
```

```
<body >
    <header>
        <h1>Título de la página</h1>
    </header>
    <nav>
        <ul>
            <li>Inicio</li>
            <li>¿Quiénes somos?</li>
            <li>Servicios</li>
            <li>Contacto</li>
        </ul>
    </nav>
    <section>
        <article>
            Primera entrada
        </article>
        <article>
            Segunda entrada
        </article>
    </section>
    <aside>
        Complemento de información
    </aside>
    <footer>
        Nos encontramos en Cancún México, Número 4, Avenida Yucatán.

    </footer>
```

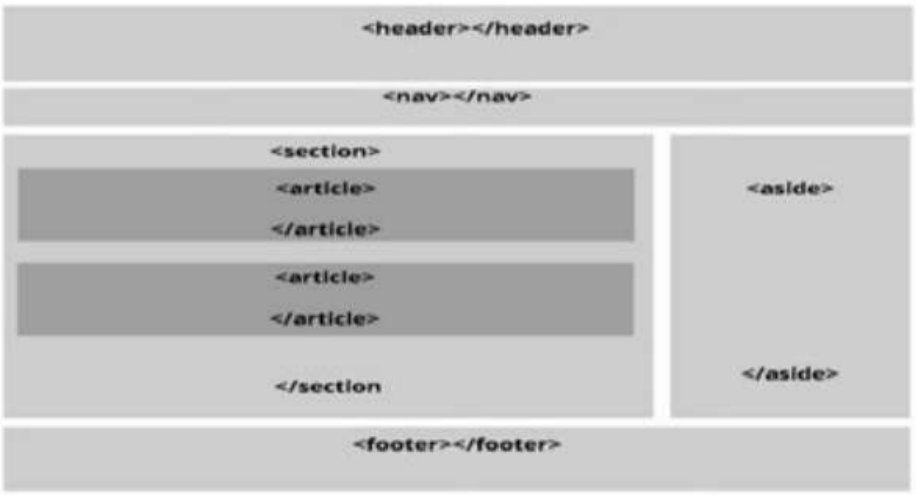
```
</body>
</html>
```

Semántica de HTML5

Si tuviéramos que definir de manera concreta los cambios en la última versión de HTML, podría resumirlos en la palabra semántica. Las etiquetas ahora no solo van a servir para colocar en un espacio determinado una imagen, un texto, enlaces e información. La realidad es que al utilizar HTML5 estamos también comunicándonos con el navegador de una forma diferente y totalmente nueva, en donde le decimos tanto el estilo y la posición de las cosas, como le damos las jerarquías a la información y esto implica que se tiene un grado de importancia para el usuario y nosotros en cada sección del sitio.

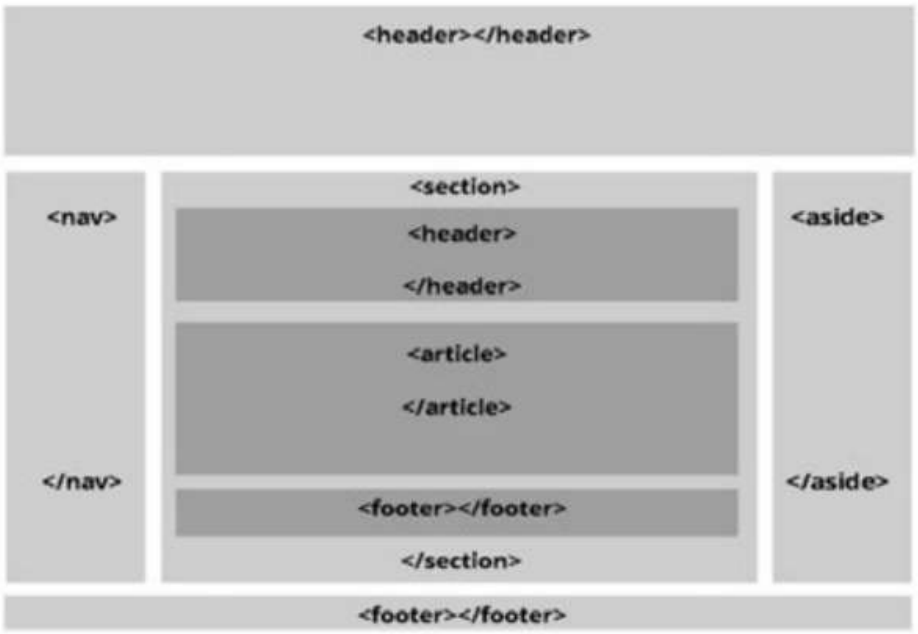
Flexibilidad de HTML5

La flexibilidad de HTML5 es puesta a prueba todos los días por los desarrolladores de sitios web, para lo que hemos visto hasta este punto del libro usamos una estructura básica como la de la siguiente imagen:



Pero en realidad si damos un paseo por internet vamos a observar que muchos de los sitios no tienen la misma estructura y esto no significa que no estén usando HML5 o que estén implementando de forma incorrecta el estándar. En realidad es que la

flexibilidad es mucha y el posicionamiento puede variar, por ejemplo la siguiente imagen es una web HTML5 que respeta el estándar y tiene una estructura diferente:



Las diferencias que podemos notar son:

- La etiqueta *<nav>* se encuentra en la izquierda.
- La etiqueta *<section>* contiene las etiquetas *<header>, <article>, <footer>*.

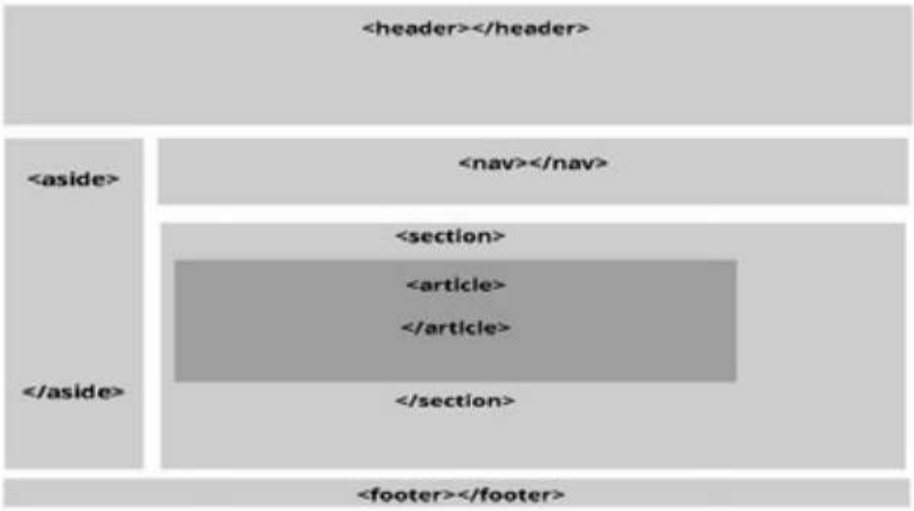
Es importante que entendamos lo siguiente: lo más importante es la semántica de la web, por ejemplo: que la etiqueta *<nav>* este arriba o abajo, izquierda o derecha es irrelevante hasta cierto punto, es obvio que si la colocamos en un lugar de difícil acceso, esto se verá reflejado en la navegación de nuestro sitio, pero es más importante entender que al ser la etiqueta *<nav>*, estamos colocando información que ayudará a la navegación de un sitio web.

La etiqueta *<nav>*, de ser necesario, podría ir colocada en cualquier otra sección.

En la imagen anterior podemos observar la etiqueta *<section>* y que tiene tres elementos, si ya entendimos el significado de la semántica podremos deducir lo siguiente: contamos con un encabezado de la sección, un artículo y un cierre (footer). Después de este ejemplo ya empezamos a ver que podemos jugar con las etiquetas, lo importante es que tenga sentido la semántica que estamos armando y podemos

ver que no es exclusiva la posición dentro del sitio web, lo relevante de este ejemplo es que la etiqueta tenga sentido semántico, dependiendo de lo que queremos colocar en el documento HTML.

La siguiente imagen nos muestra un último ejemplo de posición de elementos dependiendo de la semántica:



Otras etiquetas

Hay muchas etiquetas además de las que ya hemos visto, en esta sección nos dedicaremos a estudiarlas para poder implementarlas.

Etiqueta <hgroup>

Esta etiqueta permite agrupar las etiquetas *<h1>,<h2>,<h3>,<h4>,<h5>,<h6>.* La idea es que se forme un bloque con las relacionadas y así tendremos un formato para nuestros títulos.

Recordemos que las etiquetas *<h>* tienen una jerarquía, la más importante es *<h1>* que sería el título, un subtítulo sería la *<h2>* y así hasta llegar a la de menor importancia que es *<h6>,* si lo hacemos bien podemos redefinir la jerarquía de este grupo de etiquetas. HTML5 nos da la posibilidad de escribir las etiquetas *<h>* teniendo además la opción de no solamente agruparlas, sino también de poder jerarquizarlas por cada grupo ***<hgroup>*** que creemos.

Un ejemplo sería el siguiente:

```
<!DOCTYPE html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>Título de la web</title>
</head>
<body >
    <header>
        <hgroup>
            <h1>Título de la página</h1>
            <h2>Subtítulo de la página</h1>
        </hgroup>
    </header>
    <nav>
        <ul>
            <li>Inicio</li>
            <li>¿Quiénes somos?</li>
            <li>Servicios</li>
            <li>Contacto</li>
        </ul>
```

```
</nav>
<section>
    <article>
        <header>
            <hgroup>
                <h1>Título</h1>
                <h2>Subtítulo</h2>
            </hgroup>
        </header>
    </article>
    <article>
        <header>
            <hgroup>
                <h1>Título</h1>
                <h2>Subtítulo</h2>
            </hgroup>
        </header>
    </article>
</section>
<aside>
    Complemento de información
```

```
        </aside>
        <footer>
                Nos encontramos en Cancún México, Número 4, Avenida Yucatan.
        </footer>
    </body>
    </html>
```

Al crear diferentes *<hgroup>* observamos que cada grupo tendrá sus jerarquías dependiendo del nivel de la *<h>* que estemos utilizando.

Etiqueta <hr/>

Esta etiqueta es una forma de separar la información, tiene la característica de que podemos modificarle si es necesario el tamaño, el ancho y los atributos que tenga, además de que se cierra en la misma etiqueta que se abre, solo debemos de colocar la diagonal al final del nombre de la etiqueta.

El código sería el siguiente:

```
<!DOCTYPE html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>Titulo de la web</title>
</head>
<body >
```

```
<header>
    <hgroup>
        <h1>Título de la página</h1>
        <h2>Subtítulo de la página</h1>
    </hgroup>
</header>
<nav>
    <ul>
        <li>Inicio</li>
        <li>¿Quiénes somos?</li>
        <li>Servicios</li>
        <li>Contacto</li>
    </ul>
</nav>
<section>
    <article>
        <header>
            <hgroup>
                <h1>Título</h1>
                <h2>Subtítulo</h2>
            </hgroup>
```

```
                    <p>Primer parrafo</p>
                    <hr/>
                    <p>Segundo parrafo</p>

                </header>
            </article>
            <article>
                <header>
                    <hgroup>
                        <h1>Título</h1>
                        <h2>Subtítulo</h2>
                    </hgroup>
                </header>
            </article>
        </section>
        <aside>
            Complemento de información
        </aside>
        <footer>
            Nos encontramos en Cancún México, Número 4, Avenida Yucatan.
```

```
        </footer>
    </body>
    </html>
```

Etiquetas <abbr> y <title>

La etiqueta *<abbr>* nos permite representar las abreviaturas que colocaremos en los párrafos, pueden ser las siglas de una entidad federativa, organización, instituto, etc. La etiqueta *<title>* en este caso es un complemento para entender el significado de la abreviatura.

Si se coloca el ratón sobre la palabra o las palabras que se encuentran dentro de la etiqueta *<abbr>*, se mostrará el texto colocado en *(title="")*.

El código sería así:

```
<!DOCTYPE html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>Titulo de la web</title>
</head>
<body >
        <header>
                <hgroup>
                        <h1>Título de la página</h1>
                        <h2>Subtítulo de la página</h1>
```

```
            </hgroup>
        </header>
        <nav>
            <ul>
                <li>Inicio</li>
                <li>¿Quiénes somos?</li>
                <li>Servicios</li>
                <li>Contacto</li>
            </ul>
        </nav>
        <section>
            <article>
                <header>
                    <hgroup>
                        <h1>Título</h1>
                        <h2>Subtítulo</h2>
                    </hgroup>
                    <p>Primer parrafo</p>
                    <hr/>
                    <p>Segundo parrafo</p>
```

```
                    </header>
                </article>
                <article>
                    <header>
                        <hgroup>
                            <h1>Título</h1>
                            <h2>Subtítulo</h2>
                        </hgroup>
                    <p> Hablamos del grupo <abbr title="Significado " >SIGLAS
</abbr>
                        </p>
                    </header>
                </article>
        </section>
        <aside>
            Complemento de información
        </aside>
        <footer>
            Nos encontramos en Cancún México, Número 4, Avenida Yucatan.

        </footer>
    </body>
    </html>
```

**Etiqueta **

Esta etiqueta, a diferencia de muchas etiquetas en HTML5, no tiene ningún valor semántico, su función es la de poder dar un estilo al texto que se encuentre dentro de la etiqueta. Podemos ver esta etiqueta como una oportunidad de colocar un estilo y lo más importante, sin afectar la semántica de nuestro documento.

Ejemplo:

```
<p> Este texto no tiene ningún estilo <span id=”cambiocolor”>Pero este sí tendrá </span></p>
```

Etiqueta <blockquote>

Esta etiqueta nos permite hacer una cita textual de una porción de texto o de un párrafo completo para notar que es diferente. Lo que se hace es que se cambia la sangría tanto del lado derecho y del lado izquierdo del texto colocado dentro de la etiqueta.

El código sería el siguiente:

```
<!DOCTYPE html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>Titulo de la web</title>
</head>
<body >
    <header>
```

```
            <hgroup>
                <h1>Título de la página</h1>
                <h2>Subtítulo de la página</h1>
            </hgroup>
        </header>
        <nav>
            <ul>
                <li>Inicio</li>
                <li>¿Quiénes somos?</li>
                <li>Servicios</li>
                <li>Contacto</li>
            </ul>
        </nav>
        <section>
            <article>
                <header>
                    <hgroup>
                        <h1>Título</h1>
                        <h2>Subtítulo</h2>
                    </hgroup>
```

```
                                        <p>Primer parrafo</p>
                                        <hr/>
                                        <p>Segundo parrafo</p>

                        </header>
                </article>
                <article>
                        <header>
                                <hgroup>
                                        <h1>Título</h1>
                                        <blockquote>Este texto es una cita</blockquote>
                                        <h2>Subtítulo</h2>
                                </hgroup>
                        </header>
                </article>
        </section>
        <aside>
                Complemento de información
        </aside>
```

```
        <footer>
                Nos encontramos en Cancún México, Número 4, Avenida Yucatán.

        </footer>
    </body>
    </html>
```

Etiqueta <cite>

Esta etiqueta sirve para citar el origen de una cita de una manera muy específica; por ejemplo, un libro, una canción, etc. El texto dentro de esta etiqueta se muestra en letra cursiva.

```
    <article>
        <header>
                <hgroup>
                <h1>Título</h1>
                <h2>Subtítulo</h2>
                   </hgroup>
                <p>Primer párrafo</p>
                <hr/>
                  <p>Segundo párrafo, hablemos de la película <cite> La teoría del
todo </cite> </p>
        </header>
    </article>
```

Etiqueta <q>

Esta etiqueta se usa para citas pequeñas o más cortas, por ejemplo: cuando señalamos solo un elemento de una lista, el texto dentro de la etiqueta quedará entre comillas dobles.

El código sería así:

```
<article>
    <header>
        <hgroup>
            <h1>Título</h1>
            <h2>Subtítulo</h2>
        </hgroup>
        <p>Primer párrafo</p>
        <hr/>
        <p>Segundo párrafo, hablemos de la película <cite> La teoría del todo </cite> </p>

        <p>Mostramos el texto por <q>José Luján</q></p>

    </header>
</article>
```

Etiqueta <figure>

Esta etiqueta nos ayuda a marcar elementos como diagramas, imágenes, ilustraciones, fotos, etc. El objetivo es poder identificar estos elementos como parte del contenido principal pero que pueden eliminarse o mover sin que este afecte al documento. En otras palabras podemos pensar en elementos visuales y ponerles una etiqueta para diferenciarlos del resto que tenemos.

El código sería así:

```
<article>
    <header>

                <h1>Título</h1>
                <blockquote>Este texto es una cita</blockquote>
                <figure>
                        <img src="imagen.png" />
                </figure>
                <h2>Subtítulo</h2>
    </header>
</article>
```

Etiqueta <figcaption>

El objetivo de esta etiqueta está relacionado con la etiqueta <figure>, usándola se enmarca un titular para esta última. Se puede ver como el método de agregar una

leyenda o texto, no es obligatorio su uso, pero se complementan de manera perfecta.

Etiqueta <dfn>

Esta etiqueta tiene el objetivo de marcar o resaltar una palabra o un contexto que es descrito en el mismo párrafo que lo contiene. Depende del navegador el estilo que toma esta etiqueta, en la mayoría de los navegadores se puede ver con el formato de itálica.

El código es el siguiente:

```
<article>
    <header>
        <h1>Título</h1>
            <p> El <dfn>formato HTML </dfn> es el lenguaje que se utiliza para a realizar cualquier sitio web.
            </p>
    </header>
</article>
```

El código sería el siguiente:

```
<article>
    <header>
        <h1>Título</h1>
            <blockquote>Este texto es una cita</blockquote>
            <figure>
            <img src="imagen.png" />
```

```
                    <figcaption>
                        Esta imagen trata de programación.
                    </figcaption>
                    </figure>
                <h2>Subtítulo</h2>
        </header>
    </article>
```

Etiqueta <mark>

Esta etiqueta la utilizamos para resaltar texto que es relevante en un contexto específico. Para entenderlo bien pensemos en cuando se marca texto en un libro, lo que deseas marcar debería de ser encerrado en la etiqueta *<mark>,* así como el marcador común de color amarillo, el texto encerrado en esta etiqueta queda de ese color amarillo.

El código es el siguiente:

```
    <p>Primer párrafo</p>

    <hr/>

    <p>Segundo párrafo, hablemos de la <mark>película</mark> <cite> La teoría del
todo </cite> </p>

    <p>Mostramos el texto por <q>José Luján</q></p>
```

Etiqueta <small>

En un principio se piensa que la etiqueta sirve para hacer que el texto se vea más pequeño, pero ya existe una forma de realizar esto desde el archivo CSS:

```
    font-size:smaller
```

El verdadero objetivo de la etiqueta es el de presentar la información legal, como decimos generalmente “leer la letra pequeña”.

El código es el siguiente:

```
<p>Primer párrafo</p>

<hr/>

<p>Segundo párrafo, hablemos de la <mark>película</mark> <cite> La teoría del todo </cite> </p>

<p>Mostramos el texto por <q>José Luján</q><small>Copyright 2015</small></p>
```

**Etiqueta **

Esta etiqueta se utiliza para indicar énfasis, el texto encerrado en la etiqueta se va a mostrar en letra cursiva.

Su código es:

```
<p>Primer párrafo</p>

<hr/>

<p>Pensemos en <em>algo</em> que queremos destacar </p>
```

**Etiqueta **

Esta etiqueta la utilizamos para indicar la parte más importante de un texto. El texto que es encerrado en la etiqueta se muestra con el estilo de letras “negritas”.

El código sería así:

```
<p>Primer párrafo</p>

<hr/>

<p>Pensemos en <em>algo</em> que queremos destacar, algo más <strong>importante </strong> </p>
```

Etiqueta <b>

La etiqueta *<b>* tiene el efecto de resaltar texto como muchas de las etiquetas que hemos analizado, pero no se recomienda usarla. Solamente que sea el último recurso, esto se indica en la documentación de HTML5. Como opciones podemos utilizar las etiquetas *, , <mark>.*

Etiqueta <i>

La etiqueta *<i>* tiene el efecto de resaltar lo que se encuentre dentro de la etiqueta, lo coloca en el formato de itálica, pero al igual que la etiqueta *<b>* ya no se recomienda implementarla, para eso tenemos otras opciones. Si se desea aplicar el formato de cursiva, se puede realizar desde la hoja de estilos sin ninguna complicación.

Etiqueta <address>

Esta etiqueta nos permite definir la información de contacto del documento HTML, le podríamos dar también un uso dentro de secciones como *<article>* y *<section>*, por defecto los navegadores cambian el texto dentro de estas etiquetas en formato de cursiva.

El código sería el siguiente:

```
<aside>
        Complemento de información
</aside>
<footer>
        <address>
        Autor: José Luján, Cancún México, Número 4, Avenida Yucatan.
        </address>
```

```
        </footer>
    </body>
    </html>
```

No debemos olvidar que en la etiqueta *<footer>* es en donde habitualmente vamos a colocar la información del autor, entonces es normal encontrar en un documento HTML por lo menos una etiqueta de apertura y de cierre de *<address>.*

Etiquetas <details> y <summary>

Estas etiquetas trabajan en conjunto para poder mostrar al usuario cierta información dependiendo de un clic. Durante mucho tiempo este efecto se realizó con Javascript o CSS y aunque existen estas etiquetas, no todos los navegadores las han implementado, por ejemplo en Firefox de Mozilla aún no funciona, pero si la pruebas en Google Chrome funcionará perfectamente.

<details> nos permite crear una porción de texto a mostrar cuando este reciba un clic, *<summary>* contiene la leyenda que se va a mostrar siempre. El código es el siguiente:

```
    <article>
        <header>
        <hgroup>
                    <h1>Título</h1>
                    <h2>Subtítulo</h2>
            </hgroup>
            <p>Primer parrafo</p>
            <hr/>
```

```
            <p>Segundo parrafo</p>
            <details>
                <summary>Expandir</summary>
                <p>Mostramos el texto</p>
            </details>
            </header>
    </article>
```

El resultado en Firefox es el siguiente:

En la imagen anterior podemos ver que no tiene efecto y el texto siempre se va mostrar, es decir, que ignora las etiquetas.

En Chrome el resultado es el siguiente:

En el momento de hacer clic despliega el texto, el resultado es el de la siguiente imagen:

Etiqueta <time>

Esta etiqueta nos permite colocar la fecha y hora. Lo interesante de esta etiqueta es que nos permite colocar diferentes formatos que solamente tenemos que especificar. Antes se colocaba la fecha y hora dentro de la etiqueta *<p>,* ahora lo podemos usar solamente utilizando *<time>.*

Dentro de la etiqueta contamos con el atributo *datetime*, en ese atributo es donde colocamos la fecha para que la entiendan otros programas.

Este es el código:

```
<article>
    <header>
        <hgroup>
            <h1>Título</h1>
            <h2>Subtítulo</h2>
        </hgroup>
        <p>Primer párrafo</p>
        <hr/>
          <p>Segundo párrafo</p>

         <p>Mostramos el <em>texto</em> por <q>José Luján</q></p>
       <p> La fecha el día de hoy es:</p>
          <time datetime="2015-05-30">30 de Mayo del 2015 </time>
    </header>
</article>
```

También podemos colocar la hora si es necesario, primero tenemos que saber la zona horaria, podemos encontrarla en Wikipedia:

http://en.wikipedia.org/wiki/Time_zone

Ya que tenemos la zona de horario correcta, sabemos qué letras son las que vamos a usar.

El código básico para la hora es:

```
<article>
    <header>
        <hgroup>
            <h1>Título</h1>
            <h2>Subtítulo</h2>
        </hgroup>
        <p>Primer párrafo</p>
        <hr/>
          <p>Segundo párrafo</p>
         <p>Mostramos el <em>texto</em> por <q>José Luján</q></p>
       <p> La fecha el día de hoy es:</p>
          <time datetime="2015-05-30">30 de Mayo del 2015 </time>
       <time datetime="12:00">Las 12 horas</time>
    </header>
</article>
```

Este es el código con zona horaria:

```
<article>
    <header>
        <hgroup>
```

```
                    <h1>Título</h1>
                    <h2>Subtítulo</h2>
                </hgroup>
                <p>Primer párrafo</p>
                <hr/>
                    <p>Segundo párrafo</p>
                  <p>Mostramos el <em>texto</em> por <q>José Luján</q></p>
               <p> La fecha el día de hoy es:</p>
                  <time datetime=”2015-05-30T12:00">Las 12h del 30 de Mayo del 2015</time>
        </header>
    </article>
```

Etiqueta <data>

Esta etiqueta nos sirve para trabajar con datos que no tienen relación alguna con la hora y la fecha, se utiliza el atributo “value” como en otras etiquetas y se coloca el valor que queremos representar.

El código sería así:

```
<data value=”9”>Nueve</data>
```

3 FORMULARIO

Los formularios son una forma de poder comunicarnos con el usuario. Siendo más específicos sobre este tema, el usuario, al utilizar un formulario, logra comunicarse con el sitio web con un objetivo definido, como por ejemplo: realizar una compra, solicitar información, realizar una búsqueda, etc.

En HTML5 contamos con nuevos elementos y atributos para poder trabajar mejor con este importante elemento.

Crear un formulario

Para crear un formulario utilizamos el elemento *<form>* que como la mayoría de las etiquetas en *HTML5* se abre y se cierra. En el momento de escribir la etiqueta *<form>* tendríamos que definir al menos 3 componentes (name, method y action) para el funcionamiento correcto.

Atributo name

Este atributo nos permite colocarle un nombre al formulario para después poder acceder a ciertos elementos que contenga.

Atributo method

Este atributo determina cómo se va enviar la información que contiene el formulario. Tenemos 2 formas disponibles para enviar los datos: Post y Get.

El método *Post* se utiliza cuando estamos pensando en enviar la información de manera no visible, esto quiere decir que no se puede ver de primera mano por el usuario.

El método *Get* se utiliza para enviar datos por medio de la URL, a la cual se le añade lo que nosotros estemos interesados en enviar.

Podemos resumir que si enviamos información delicada deberíamos usar el método *Post* y cuando usamos información común y corriente podemos utilizar *Get*.

Atributo action

En el atributo *action* colocamos la URL en la que se va a procesar la información, por ejemplo si tenemos un archivo que es “procesar_formulario.php” en nuestro *action* tendríamos que colocar esto para hacer llegar lo que estamos mandando del formulario a ese archivo y ahí procesar lo que necesitamos y queremos.

El código de un formulario básico sería el siguiente:

```
<!DOCTYPE html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>Titulo de la web</title>
</head>
<body >
```

```
        <section>

                <form name="cuestionario" method="get"
action="procesar_formulario.php">

                </form>

        </section>

        <footer>



        </footer>

    </body>

    </html>
```

Atributo target

Este atributo no es de los elementales pero sí hay que implementarlo en algunos casos ya que nos permite indicar dónde vamos a mostrar la respuesta de lo que se recibe. Tenemos estos valores:

_self

Lo mostrará en el mismo lugar (este viene por defecto si no colocamos un valor).

_blank

Lo mostrará en una nueva ventana.

_parent

Sirve para el nivel de navegación, si contamos con solo un nivel el resultado de _parent y _top van a coincidir.

_top

Lo muestra en la ventana de nivel superior.

Elemento <input>

Este elemento es el que permite introducir datos en el formulario para después procesarlos. Este elemento cuenta con un atributo que es *"type",* este atributo nos sirve para definir el tipo de dato que esperamos en la entrada, no es lo mismo introducir un nombre de usuario o una contraseña, sabemos que en la contraseña por convención no se muestra lo que se introduce, se muestran asteriscos en lugar de los caracteres que se introducen. En el atributo *type* tenemos los siguientes valores:

Valor "text"

Se coloca cuando esperamos recibir un texto.

Valor "password"

Se utiliza cuando estamos esperando recibir una contraseña, con esto nos aseguramos de que lo que se introduce en este campo se oculta a la vista con el carácter de asterisco.

Valor "hidden"

Con este valor estamos ocultando el campo de entrada.

Valor "radio"

Se utiliza cuando nos interesa generar un botón del tipo radio.

Valor "checkbox"

Se utiliza para generar una casilla de selección, se complementa utilizando el atributo *value* que nos permite especificar el valor que vamos a enviar si es que está

seleccionada la casilla, en caso de no colocar un valor, se manda el valor *"on"* para indicar que está seleccionada.

Valor "button"

Este valor crea un botón, a este botón le podemos colocar acciones o diseño dependiendo de nuestras necesidades, en muchos casos se utiliza en conjunto con código de *javascript* para las acciones.

Valor "submit"

Este valor crea un botón con el objetivo de enviar el formulario, cuenta con un atributo *value* en donde se coloca un texto.

Este es un ejemplo simple en código aplicando todos los valores anteriores:

```
<!DOCTYPE html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>Título de la web</title>
</head>
<body >
    <section>
        <form name="cuestionario" method="get" action="procesar_formulario.php">
        <label>Usuario:</label>
        <input type="texto" name="usuario">
```

```
            <label>Contraseña:</label>
            <input type="texto" name="usuario">

            <input type="hidden" name="oculto" value="oculto">

            <label>País:</label>
            <input type="radio" name="opciones" value="mx" checked>México
            <input type="radio" name="opciones" value="es">España
            <input type="radio" name="opciones" value="eu">USA
            <label>Check</label>
            <input type="checkbox" name="casilla" value="1">
            <input type="submit" value="Enviar">

            </form>
        </section>
        <footer>

        </footer>
    </body>
    </html>
```

Otros elementos

Elemento <textarea>

La característica principal es que este elemento crea un campo para introducir texto con diferentes líneas, el tamaño lo podemos definir colocándole un valor a los atributos fila y columna, *rows* y *cols* en inglés, respectivamente.

Elementos <select> y <option>

El elemento *<select>* nos permite colocar elementos dentro de él para agrupar las opciones, el elemento *<option>* nos permite definir un elemento para colocar dentro del elemento *<select>*.

El código de ejemplo quedaría de la siguiente forma:

```
<!DOCTYPE html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>Titulo de la web</title>
</head>
<body >
    <section>
        <form name="formulario2" method="post" action="procesar_formulario.php">
        <label>Ejemplo textarea:</label>
        <textarea name="areadetexto" rows="6" cols="25"> </textarea>
```

```
            <br />
            <label>Opciones:</label>
            <select name="opciones">
            |       <option value="1">Opcion 1</option>
                    <option value="2">Opcion 2</option>
                    <option value="3">Opcion 3</option>
            </select>
                    <br />
            <input type="submit" value="Enviar">


            </form>
        </section>
        <footer>


        </footer>
    </body>
    </html>
```

Formularios HTLM5

En HTML5 una de las primeras cosas que fueron evidentes fue la cantidad de opciones nuevas con las que contaba el elemento *<input>,* en esta sección vamos a conocer las opciones que tenemos disponibles, en general decimos que el elemento *<input>* en su atributo *<type>* ahora puede recibir otro tipo de entradas.

Color

Nos sirve para poder seleccionar un color, al tratarse de colores el valor que se espera es un hexadecimal #XXXXXX.

El código sería el siguiente:

```
<input type=”color” name=”color”>
```

Date

Se utiliza cuando trabajamos con las fechas (año, mes, día), la interfaz depende de cada navegador, se usa comúnmente en los sitios de reservas, el código sería el siguiente:

```
<input type=”date” name=”fecha” >
```

Datetime

Nos permite manejar la fecha y la hora (incluye la zona horaria).

El código es así:

```
<input type=”datetime” name=”fechayhora”>
```

Datetime-local

También nos permite manejar la fecha y la hora pero la diferencia con el elemento anterior es que no tiene una zona horaria.

```
<input type=”datetime-local” name=”fechayhora”>
```

Email

Este tipo de campo sirve para introducir correos electrónicos, con esto aseguramos que el campo reciba un correo electrónico.

El código es:

```
<input type=”email” name=”correoelectronico”>
```

Month

Con este tipo de campo podemos colocar el año y el mes.

```
<input type=”month” name=”mes”>
```

Number

En este tipo de campo recibimos un número pero cuenta con unos atributos que nos permiten definir por ejemplo el máximo, el mínimo y para un rango de valores.

max colocamos el valor máximo aceptable.

min colocamos el valor mínimo aceptable.

step nos permite definir intervalos válidos para la entrada.

El código es el siguiente:

```
<input type=”number” name=”numero” min=”0” max =”50” step=”5”>
```

Range

Nos permite seleccionar un valor de un rango de números. Debemos de utilizar los atributos *min* y *max* para el funcionamiento correcto (se puede combinar con *step* también).

El código es el siguiente:

```
<input type=”range”  name=”rango” min=”0” max=”50”>
```

Search

Nos sirve para realizar validaciones en un campo de búsqueda.

```
<input type=”search” name=”buscar”>
```

Time

Se utiliza para trabajar con el formato de horas y minutos, puede tener 2 opciones, horas-minutos-segundos u horas-minutos, dependiendo del navegador.

El código es:

```
<input type=”time” name=”tiempo”>
```

Url

Se utiliza cuando deseamos aceptar una dirección url (absoluta) y devuelve un error en caso de no aceptarla.

```
<input type=”url” name=”unaurl”>
```

Week

Se utiliza cuando nos interesa trabajar con las semanas, nos indica el número de semana.

```
<input type=”week” name=”semana”>
```

Formularios de atributos HTML5

autocomplete

Este atributo puede tomar 2 valores, si no le colocamos valor, el valor por defecto es *on*, el otro valor que podemos colocar es *off*. Se recomienda colocar en el campo de búsquedas.

Este es el código:

```
<input type=”search” name=”busqueda” autocomplete=”on”>
```

autofocus

Este atributo selecciona por defecto el elemento al que se le coloca este atributo, el objetivo es que hagamos notar la importancia de un elemento al usuario.

Ejemplo:

```
<input type=”search” name=”busqueda ” autocomplete=”off” autofocus>
```

form

Este atributo nos permite declarar elementos de un formulario fuera de las etiquetas *<form> </form>.*

Este es el código:

```
<!DOCTYPE html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>Titulo de la web</title>
</head>
<body >
    <section>
        <form name="formulario2" method="post" action="procesar_formulario.php">
```

```
                <label>Ejemplo textarea:</label>
                <textarea name="areadetexto" rows="6" cols="25"> </textarea>
                <br />
                <label>Opciones:</label>
                <select name="opciones">
                |       <option value="1">Opcion 1</option>
                        <option value="2">Opcion 2</option>
                        <option value="3">Opcion 3</option>
                </select>
                        <br />
                <input type="submit" value="Enviar">

                </form>

                <input type="search" name="busqueda" form="formulario2">
        </section>
        <footer>

        </footer>
</body>
</html>
```

formnovalidate

El atributo **formnovalidate** nos sirve para NO utilizar la validación integrada que tenemos en HTML5. La validación siempre va a suceder ya que viene activada por defecto, en caso de no querer que se realice, tenemos que colocar el atributo *formnovalidate* en el botón de envío.

El código quedaría así:

```
<!DOCTYPE html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>Título de la web</title>
</head>
<body >
    <section>
        <form name="formulario2" method="post" action="procesar_formulario.php">
        <label>Ejemplo textarea:</label>
        <textarea name="areadetexto" rows="6" cols="25"> </textarea>
        <br />
        <label>Opciones:</label>
        <select name="opciones">
        |    <option value="1">Opcion 1</option>
```

```
                    <option value="2">Opcion 2</option>
                    <option value="3">Opcion 3</option>
                </select>
                    <br />
                <input type="submit" value="Enviar" formnovalidate>

                </form>

                <input type="search" name="busqueda" form="formulario2">
        </section>
        <footer>

        </footer>
    </body>
    </html>
```

multiple

El objetivo de este atributo es colocar múltiples valores en un mismo campo. Este se activa como cualquier tipo boolean colocándolo en donde nos interesa que se active. Para ingresar múltiples valores tenemos que colocar una coma entre ellos. No se puede utilizar en todos los tipos de entradas.

El código sería el siguiente:

```
<input type="email" name="correos" multiple>
```

patern

Este atributo es muy útil en el momento de querer una validación ya que nos permite definir expresiones para crear reglas de validación, por ejemplo si necesitamos recibir únicamente 2 letras y no queremos números o caracteres especiales. El código es el siguiente:

```
<!DOCTYPE html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>Título de la web</title>
</head>
<body >
    <section>
        <form name="formulario2" method="post" action="procesar_formulario.php">
        <label>Ejemplo textarea:</label>

        <label>2 letras</label>
        <input type="text" name="letras" pattern="[A-Za-z]{2}">

            <br />
        <input type="submit" value="Enviar">
```

```
            </form>



        </section>
        <footer>


        </footer>
    </body>
    </html>
```

Si introducimos datos que no son los que solicitamos en el atributo *pattern*, veremos lo siguiente:

En caso de colocar 2 caracteres que sean letras de la (A-Z) y (a-z) simplemente continúa con las acciones que indicamos en el formulario.

placeholder

Este atributo nos sirve para poder entregar una pista respecto a la entrada esperada. El valor que coloquemos se mostrará dentro de la entrada.

El código es el siguiente:

```
<input type=”search” name=”buscar” placeholder= "indica que buscar">
```

required

Este atributo, al colocar una entrada, esta se vuelve indispensable y esto quiere decir que el formulario no será procesado si falta este dato.

El código es el siguiente:

```
<input type=”text” name=”nombre” required >
```

4 CSS

Las siglas CSS son un acrónimo del inglés “Cascading Style Sheets”, en español lo podemos traducir como “Hojas de estilo en cascada”.

Hasta este capítulo hemos trabajado con HTML y las famosas etiquetas que este nos proporciona, pero existen otros tipos de problemas cuando trabajamos con HTML, por ejemplo el estilo.

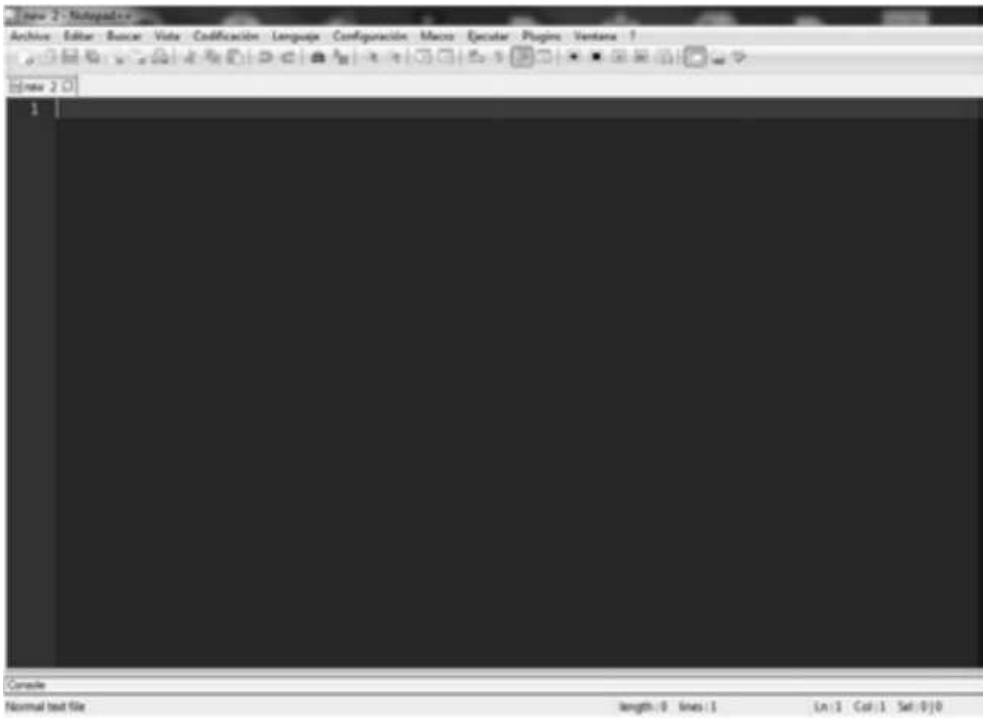
Cuando hablamos de estilo queremos decir el color, el tamaño, la posición y todo lo que tiene que ver con el estilo “visual”. CSS es un lenguaje que sirve para definir y jugar con el estilo en HTML. Debemos dejar claro que HTML es un lenguaje de etiquetas y funciona por sí solo. CSS es un lenguaje o mejor dicho: “Hojas de estilos” que trabajan sobre HTML.

En CSS al igual que HTML se cuentan con versiones, actualmente nos encontramos en la versión CSS3. A estas alturas el desarrollador de sitios web en su labor diaria es casi imposible que solamente use alguna de estas tecnologías, podríamos decir que el 99% de los desarrolladores combinan la implementación de CSS y HTML para generar los sitios que visitamos todos los días en internet.

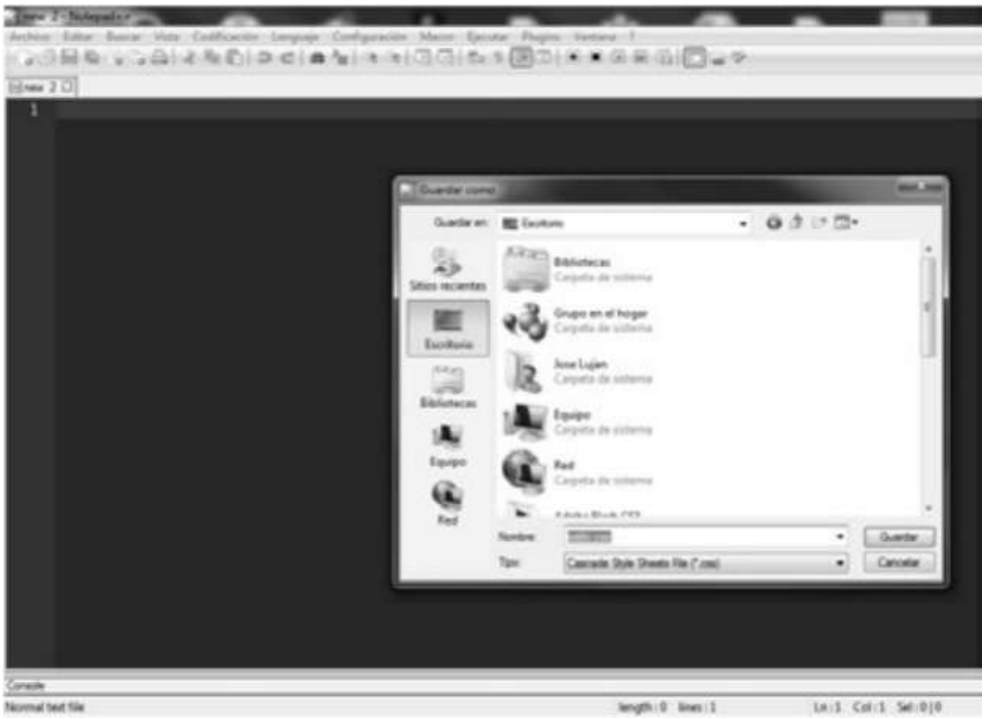
El objetivo principal de CSS es que podamos mejorar y controlar la presentación de nuestro HTML.

Crear un archivo CSS

Para crear un archivo CSS abrimos el editor que usamos para escribir archivos HTML y simplemente creamos un archivo nuevo como en la imagen:



Después seleccionamos la opción “Guardar como” y colocamos el nombre “estilo” (no tiene que ser este nombre por defecto, pero en español se coloca este de forma habitual) y seleccionamos el tipo: Cascade Style Sheets File. Se vería como la siguiente imagen:



Por ahora no importa que el archivo esté vacío, lo importante es que ya hemos creado un archivo CSS que contendrá nuestros estilos.

A continuación realizaremos la implementación de un estilo utilizando CSS y no utilizándolo.

Ejemplo de estilo sin CSS

Lo que vamos a hacer ahora es mostrar un ejemplo de cómo se definía un estilo sin utilizar CSS, debemos recordar que CSS es una tecnología que nos permite presentar de mejor forma nuestros archivos HTML, pero significa que es obligatoria o la única opción, aunque no se puede concebir actualmente el desarrollo de un sitio sin implementar esta tecnología.

Supongamos que tenemos un simple archivo HTML con una etiqueta *<h1>* y un párrafo *<p>*.

Este el código:

```
<!DOCTYPE html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>Titulo de la web</title>
</head>
<body >
    <section>
        <h1>Mi web</h1>
        <p>Esto es solo un párrafo </p>
    </section>
    <footer>

    </footer>
</body>
</html>
```

Si analizamos el código anterior observamos que es un código HTML normal como el de cualquier sitio web, supongamos que queremos colocar el párrafo de color rojo y de un tamaño específico.

Para lograr eso tendríamos que utilizar <font> como en el siguiente código:

```
<!DOCTYPE html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>Titulo de la web</title>
</head>
<body >
    <section>
        <h1>Mi web</h1>
        <p><font color="red" size="2em">Esto es solo un párrafo </font></p>
    </section>
    <footer>

    </footer>
</body>
</html>
```

En el código anterior colocamos el color utilizando *<font>* y esto en HTML5 ya no es recomendable, pero así es como se hacía.

Si agregamos un segundo párrafo, en realidad el código que colocamos en *<font>* lo tendríamos que volver a colocar ya que en el momento de cerrar la etiqueta *<font>* termina el efecto implementado y quedaría el código como el siguiente:

```
<!DOCTYPE html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>Titulo de la web</title>
</head>
<body >
    <section>
        <h1>Mi web</h1>
        <p><font color="red" size="2em">Esto es solo un párrafo </font></p>
        <p><font color="red" size="2em">Esto es un párrafo nuevo</font></p>
    </section>
    <footer>

    </footer>
</body>
</html>
```

Si examinamos el código anterior, podemos observar que el código se comienza a volver sucio, repetitivo y poco práctico, esta es una de las varias razones por las que se implementa una hoja de estilo CSS.

A continuación, haremos que nuestro texto sea de color rojo pero utilizando una hoja de estilos.

Ejemplo de estilo con CSS

Ya sabemos cómo crear un archivo CSS, así que comenzamos desde que ya tenemos un archivo CSS vacío. Ahora lo que importa es indicarle a nuestro archivo HTML que tiene una hoja de estilos que tomar en cuenta en el momento de desplegar una vista de nuestro archivo HTML.

Colocaremos la siguiente línea:

```
<link rel=”stylesheet” href=”estilo.css”>
```

El significado de la línea es: le indicamos que tenemos un archivo que se llama estilo.css y que es una hoja de estilo de nuestro archivo. Esto se coloca dentro de la etiqueta “head” de nuestro archivo HTML y se vería de la siguiente forma:

```
<!DOCTYPE html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>Titulo de la web</title>
<link rel="stylesheet" href="estilo.css">
</head>
<body >
```

```
        <section>
                <h1>Mi web</h1>
                <p><font color="red" size="2em">Esto es solo un párrafo
</font></p>
                <p>Esto es solo un párrafo </p>
        </section>
        <footer>


        </footer>
    </body>
    </html>
```

Una buena práctica es crear un directorio exclusivo para los archivos con un formato CSS, ya que así podremos detectar de manera más rápida dónde se encuentran los archivos con ese formato.

En ese caso creamos un directorio que podemos llamar CSS o ESTILOS. En nuestro caso creamos un directorio que se llama CSS. El archivo estilo.css lo colocaremos dentro de este directorio y el código lo tendríamos que cambiar de la siguiente forma:

```
    <!DOCTYPE html>
    <html lang="es">
    <head>
    <meta charset="utf-8">
    <title>Titulo de la web</title>
```

```
    <link rel="stylesheet" href="css/estilo.css">
    </head>
    <body >
        <section>
            <h1>Mi web</h1>
            <p><font color="red" size="2em">Esto es solo un párrafo
</font></p>
            <p>Esto es solo un párrafo </p>
        </section>
        <footer>


        </footer>
    </body>
    </html>
```

Ahora no vamos a explicar con detalle cómo utilizar CSS ya que tendremos una explicación profunda de toda la sintaxis y la estructura de CSS, lo que nos interesa es dejar los párrafos de color rojo para mostrar la ventaja de implementar CSS.

Nuestro archivo estilo.css debe de contener el siguiente código:

```
    p{
    color:#ff0000;
    font-size:2em;
    }
```

Podemos traducir el código anterior de la siguiente forma: las etiquetas *<p>* que halla en nuestro archivo HTML van a tener el color en hexadecimal #ff0000 y el tamaño de la fuente será de 2em.

La forma común de utilizar los colores en HTML y CSS es con su valor hexadecimal, así es como lo hace la mayoría del software de diseño en el formato RGB.

Nuestro archivo HTML contendría este código:

```
<!DOCTYPE html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>Titulo de la web</title>
<link rel="stylesheet" href="css/estilo.css">
</head>
<body >
    <section>
        <h1>Mi web</h1>
        <p>Esto es solo un párrafo </p>
        <p>Esto es solo un párrafo </p>
    </section>
    <footer>

    </footer>
```

```
</body>

</html>
```

En el código anterior podemos ver una de las ventajas de implementar una web con hojas de estilos, vemos que el código se mantiene limpio, no es repetitivo y en el archivo HTML solo nos tenemos que seguir preocupando por la semántica y la estructura, así dejamos que CSS sea el archivo en donde nos preocupamos por la vista y el estilo.

Usar CSS y HTML juntos es una realidad, actualmente HTML5 contempla la implementación de CSS3, las personas que manejan HTML y desarrollan sitios web no pueden trabajar sin combinar estos dos estándares definidos para la creación de sitios web.

Selector de un estilo CSS

Una estructura general de lo que declaramos en nuestros archivos con formato CSS es el siguiente ejemplo:

```
p{color: #FF0000; }
```

p - es la etiqueta a la que vamos aplicar el estilo, se le conoce como selector.

{ - la llave es la apertura de la definición.

color - es la propiedad que queremos modificar.

: - es el indicado de que lo que viene a continuación es un valor.

#FF0000 - es el valor que le asignamos a la propiedad seleccionada.

; - es el separador de la propiedad.

} - con la llave de cierre terminamos la declaración.

Para una mejor lectura de los estilos se recomienda separar de la siguiente forma los saltos de línea:

```
p{
color: #FF0000;
 }
```

Si queremos modificar más de una propiedad, se recomienda colocar una propiedad por línea, como el siguiente ejemplo:

```
p{
color: #FF0000;
font-size:2em;
 }
```

Los comentarios en el archivo CSS se colocan entre los caracteres /* */, sería algo así:

```
p{
color:#ff0000; /* esto es el color rojo*/
font-size:2em;
}
/*
 Estas
son
muchas
```

```
líneas
de
comentarios
*/
```

Referencias

La referencia es la forma en la que establecemos la relación entre el estilo y a qué le aplicaremos ese estilo, en los ejemplos anteriores la referencia era a la etiqueta *<p>,* pero esto cambiaría si lo queremos aplicar a un título, a un párrafo únicamente, a una imagen u otro elemento.

Para crear una referencia, tenemos 3 opciones básicas que son las siguientes:

- Por etiqueta
- Por un id
- Por una clase

Referencia por etiqueta

La referencia por etiqueta es la que hemos estado utilizando en el caso de *<p>,* esta nos sirve cuando queremos que afecte a todos los elementos con la misma etiqueta, si decimos que afecte a la etiqueta <p>, todas las etiquetas <p> declaradas sin excepción se verán afectadas por el estilo que definamos.

Cuando usamos etiquetas para referirnos en un archivo CSS no es necesario colocar algún carácter especial, solamente colocamos el nombre de la etiqueta, en el caso de ser <p> o <h1>, por ejemplo sería algo así:

```
p{
color:#ff0000;
}
```

```
h1{
color:#ffff00;
}
```

Referencia por id

El id es un identificador que podemos colocar a los elementos de nuestro archivo HTML para poder acceder a él de forma directa, una regla al colocar un id es que este es único, ya que es un identificador y no lo podemos repetir para otro elemento aunque no sea del mismo tipo. Vamos a crear dos párrafos y que uno de ellos tenga un identificador.

El código quedaría de la siguiente forma:

```
<!DOCTYPE html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>Titulo de la web</title>
<link rel="stylesheet" href="css/estilo.css">
</head>
<body >
    <section>
        <h1>Mi web</h1>
        <p id="cambio">Esto es solo un párrafo </p>
        <p>Esto es solo un párrafo </p>
```

```
        </section>
        <footer>


        </footer>
    </body>
    </html>
```

Para crear una referencia por id en un archivo CSS utilizamos el carácter “#”, que se conoce en algunos países como el de gato o número, se encuentra en la tecla del número 3 en el teclado de un PC.

El archivo estilo.css contendría un código como el siguiente:

```
#cambio{
color:#ff0000;
font-size:2em;
}
```

Podemos ver que el estilo solo afecta al párrafo que contiene el id=”cambio” y como el id tiene que ser único, ese estilo solamente puede afectar a un elemento si lo hacemos bien.

Referencias por clase

La referencia por clase nos permite ser más flexibles ya que una clase se puede asignar a varios elementos y con esto podemos lograr efecto en más de un elemento. Para definir una clase en un archivo CSS necesitamos colocar el carácter de punto “.” antes del nombre de la clase. El código de nuestro archivo estilo.css sería el siguiente:

```
.cambio{
color:#ff0000;
font-size:2em;
}
```

El código del archivo HTML es el siguiente:

```
<!DOCTYPE html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>Titulo de la web</title>
<link rel="stylesheet" href="css/estilo.css">
</head>
<body >
    <section>
        <h1 class="cambio">Mi web</h1>
        <p class="cambio">Esto es solo un párrafo </p>
        <p>Esto es solo un párrafo </p>
    </section>
    <footer>

    </footer>
```

```
</body>

</html>
```

Como vemos en el código anterior la clase a diferencia del id se le puede asignar a diferentes elementos aunque sean de distinta naturaleza.

Información relevante de las referencias

Dentro de la definición de referencias es importante conocer las otras formas que tenemos para implementarlas.

Selectores en conjunto

Si por algún motivo necesitamos que dos elementos tengan el mismo estilo, los podemos colocar juntos, este sería el código:

```
p h1{

color:#FF0000;

font-size:2em;

}
```

Las etiquetas se deben separar por un espacio.

Selectores por descendencia

Si tenemos casos en donde un elemento contiene a otro y queremos que ese elemento tenga un estilo, no es necesario declarar una clase o un id, pongamos el ejemplo de una etiqueta **<a>**

Tenemos este ejemplo:

```
<p>Si te interesa abre el <a href=”http://pagina.com”> enlace </a> </p>
```

Podemos ver que tenemos un enlace en la etiqueta *<p>*.

Si queremos que esa etiqueta *<a>* sea la que debe tener un estilo entonces podemos en nuestro CSS colocar lo siguiente:

```
p a {
font-size:10em;
}
```

El detalle que no debemos olvidar es que TODAS las etiquetas *<a>* que se encuentren dentro de una etiqueta *<p>* van a tomar ese estilo.

Otras referencias

Además de las referencias ya estudiadas, podremos encontrar otras formas de señalar los estilos en nuestros sitios web.

Referencia de clases con elementos

Primero analizaremos el código HTML de nuestro sitio:

```
<!DOCTYPE html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>Título de la web</title>
<link rel="stylesheet" href="css/estilo.css">
</head>
<body >
    <section>
```

```
            <h1 class="cambio">Mi web</h1>
            <p class="cambio">Esto es solo un párrafo </p>
            <p>Esto es solo un párrafo </p>
        </section>
        <footer>


        </footer>
    </body>
    </html>
```

Podemos observar que tenemos un elemento *<h1>* y dos elementos *<p>,* la clase "cambio" está asignada a dos elementos, al elemento *<h1>* y a un elemento *<p>.* Entonces nuestra clase se aplica a ellos dos, supongamos que aplicar el estilo pero solamente a los *h1*, esto quiere decir que aunque el elemento *<p>* tenga el texto: *class="cambio"* no va a tomar en cuenta el estilo, quedaría el código de nuestro archivo estilo.css de la siguiente forma:

```
h1.cambio{
color:#FF0000;
font-size:10em;
}
```

El resultado se verá de la siguiente forma:

Podemos ver que aunque en el HTML tenemos una etiqueta *<h1>* y otra etiqueta *<p>* con el estilo asignado, en realidad solo la etiqueta *<h1>* lo toma en consideracion, esto es debido a que en el estilo colocamos elementos antes del nombre de la clase:

```
h1.cambio{
```

Con eso le indicamos que solo las etiquetas _h1 que tengan asignada la clase cambio son las que tendrán ese estilo.

Referencias con selector de atributos

Otra forma que tenemos de utilizar el estilo es aprovechándonos de la posibilidad de utilizar selectores pero más detallados, esto lo podemos lograr con expresiones regulares.

Es importante mencionar que estos selectores nos van a servir para cualquier atributo, pero para el ejemplo utilizaremos el atributo *name*.

Analicemos 3 expresiones regulares que nos van a servir:

- ^=
- $=
- *=

Este es nuestro código HTML:

```
<!DOCTYPE html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>Título de la web</title>
<link rel="stylesheet" href="css/estilo.css">
</head>
<body >
    <section>
        <h1 class="cambio">Mi web</h1>
        <p name="cambio">Esto es solo un parrafo </p>
        <p name="rojo">Esto es solo un parrafo </p>
        <p name="rojomini">Esto es solo un parrafo </p>
        <p name="azulgrande">Esto es solo un parrafo </p>
        <p name="miniazul">Esto es solo un parrafo </p>
    </section>
    <footer>
```

```
        </footer>
    </body>
    </html>
```

Este es el código de nuestro estilo.css:

```
h1.cambio{
color:#FF0000;
font-size:10em;

}
p[name^="rojo"]{
color:#00FF00;
}

p[name$="grande"]{
color:#0000FF;
}

p[name*="nia"]{
font-size:5em;
}
```

Este es el resultado del código:

Vamos a centrarnos en examinar el código CSS. Tenemos los 3 selectores:

```
p[name^="rojo"]{
color:#00FF00;
}

p[name$="grande"]{
color:#0000FF;
}

p[name*="nia"]{
```

```
font-size:5em;
}
```

El selector “^=” lo que hace es asignar el estilo a todo elemento que en el atributo *name* empiece con “rojo”, el código es el siguiente:

```
p[name^="rojo"]{
color:#00FF00;
}
```

El selector “$=” lo que hace es asignar el estilo a todo elemento que en el atributo *name* termine con “grande”, el código es el siguiente:

```
p[name$="grande"]{
color:#0000FF;
}
```

El selector “*=” lo que hace es asignar el estilo a todo elemento que contenga en el atributo *name* la cadena “nia”, el código es el siguiente:

```
p[name*="nia"]{
font-size:5em;
}
```

Hagamos un paréntesis para hablar del carácter asterisco “*” que se le considera también como selector universal, este nos va a permitir en el CSS asignar un valor a todos los elementos del documento si los colocamos por ejemplo:

```
*{
```

```
margin:0px;

}
```

La línea anterior entonces lo que hace es colocar el margen de todos los elementos en 9 píxeles.

Selector hover

Este selector lo usamos cuando nos interesa que un elemento actúe o cambie de cierta forma en el momento que pasemos el ratón sobre él. Para usarlo colocamos primero la referencia del elemento y después del carácter de dos puntos ":" se coloca la palabra *hover*.

Este es el código del archivo html:

```
<!DOCTYPE html>

<html lang="es">

<head>

<meta charset="utf-8">

<title>Titulo de la web</title>

<link rel="stylesheet" href="css/estilo.css">

</head>

<body >

    <section id="contenedor">

        <p class="texto">Probando los estilos</p>

    </section>

    <footer>
```

```
        </footer>
    </body>
    </html>
```

Este es el código del archivo estilo.css:

```
body{
padding:0px;
margin:0px;
}

#contenedor{
text-align:center;
padding:10px;
border:5px solid ;
width:400px;
}

#contenedor:hover{
width:800px;
}
```

```
.texto{
font-size:30px;
}
```

Así es cómo se ve el elemento sin poner el ratón sobre el elemento:

Así es cómo se ve el elemento al colocar el ratón sobre él:

Referencia por pseudo-clase

Las pseudo-clases nos permiten aprovechar las posiciones de los elementos HTML, las asociaciones que hay entre ellos y los estados de los elementos.

:nth-child (número)

Esta pseudo-clase selecciona los elementos con la condición de que sea el hijo.

:nth-last-child(número)

Sirve igual que la anterior pero empieza a contar desde el último hijo.

:first-child

Sirve para seleccionar únicamente el primer hijo del elemento padre.

:last-child

Sirve para seleccionar únicamente el último hijo del elemento padre.

:nth-of-type(número)

Selecciona el elemento indicado pero con la condición de que sean del mismo tipo.

:nth-last-of-type(número)

Lo mismo que el anterior pero seleccionando el último elemento.

:empty

Selecciona el elemento pero con la condición de que no tenga hijos.

Propiedades

Vamos a conocer algunas de las propiedades que tenemos disponibles en CSS y qué significan.

margin-top

Permite colocar un margen en la parte superior del elemento, código de ejemplo:

```
p{
margin-top:10px
}
```

margin-right

Coloca un margen en la parte derecha del elemento, código de ejemplo:

```
p{
margin-right:10px;
}
```

margin-bottom

Crea un margen en la parte inferior del elemento, código de ejemplo:

```
p{
margin-bottom:10px;
}
```

margin-left

Añade un margen en la parte izquierda del elemento, código de ejemplo:

```
p{
margin-left:10px;
}
```

padding-top

Permite colocar un relleno en la parte superior del elemento, código de ejemplo:

```
#cuadro{
padding-top:10px
}
```

padding-right

Nos permite colocar un relleno en la parte derecha del elemento, código de ejemplo:

```
#cuadro{
padding-right:10px
}
```

padding-bottom

Coloca un relleno en la parte inferior del elemento, código de ejemplo:

```
#cuadro{
padding-bottom:10px
}
```

padding-left

Nos permite colocar un relleno en la parte izquierda del elemento, código de ejemplo:

```
#cuadro{
padding-left:10px
}
```

border-top-width

Permite colocar un ancho en el borde superior del elemento, código de ejemplo:

```
p{
border-top-width:5px;
}
```

border-right-width

Nos permite colocar un ancho en el borde derecho del elemento, código de ejemplo:

```
p{
border-right-width:5px;
}
```

border-bottom-width

Nos permite colocar un ancho en el borde inferior del elemento, código de ejemplo:

```
p{
border-bottom-width:5px;
}
```

border-left-width

Permite colocar un ancho en el borde izquierdo del elemento, código de ejemplo:

```
p{
border-left-width:5px;
}
```

border-top-color

Nos permite colocar un color en el borde superior del elemento, código de ejemplo:

```
p{
border-top-color:#FF00FF;
}
```

border-right-color

Añade un color en el borde derecho del elemento, código de ejemplo:

```
p{
border-right-color:#FF00FF;
}
```

border-bottom-color

Coloca un color en el borde inferior del elemento, código de ejemplo:

```
p{
border-bottom-color:#FF00FF;
}
```

border-left-color

Nos permite colocar un color en el borde izquierdo del elemento, código de ejemplo:

```
p{
border-left-color:#FF00FF;
}
```

border-top-style

Nos permite añadir un estilo al borde superior del elemento, código de ejemplo:

```
p{
border-top-style:dotted;
}
```

border-right-style

Permite colocar un estilo al borde derecho del elemento, código de ejemplo:

```
p{
border-right-style:dotted;
}
```

border-bottom-style

Añade un estilo al borde inferior del elemento, código de ejemplo:

```
p{
border-bottom-style:dotted;
}
```

border-left-style

Permite colocar un estilo al borde izquierdo del elemento, código de ejemplo:

```
p{
border-left-style:dotted;
}
```

width

Esta propiedad permite determinar el ancho del elemento, puede ser por una medida o por porcentaje.

height

Permite determinar el alto del elemento, puede ser por una medida o por porcentaje.

min-width

Establece un ancho mínimo de los elementos del bloque, puede ser por una medida o por porcentaje.

min-height

Establece un alto mínimo de los elementos del bloque, puede ser por una medida o por porcentaje.

max-width

Establece un ancho máximo de los elementos del bloque, puede ser por una medida o por porcentaje.

max-height

Establece un alto máximo de los elementos del bloque, puede ser por una medida o por porcentaje.

line-height

Establece la altura entre las bases del texto, puede ser por número, longitud o porcentaje.

vertical-align

Establece una alineación vertical del texto, puede ser por posición, porcentaje o longitud.

color

Establece el color de primer plano, el valor se acostumbra colocar en valor hexadecimal.

backgroud-color

Establece el color de fondo del elemento que lo contiene.

backgroud-image

Establece una imagen de fondo del elemento que lo contiene, se coloca la url de la ubicación de la imagen.

background-repeat

Establece la repetición de la imagen de fondo, se coloca el valor del eje que queremos se repita o no se repita, eje X-eje Y.

background-position

Establece la posición de la imagen de fondo, puede ser por posición, porcentaje y longitud.

text-align

Establece la alineación del texto, se pueden colocar los valores izquierda, derecha, centro y justificado.

text-decoration

Establece un estilo o efecto al texto, puede ser el valor de subrayado, tachado o parpadeo.

letter-spacing

Establece el espacio entre los caracteres, se coloca el valor de la longitud del espacio.

word-spacing

Establece el espacio entre las palabras, se coloca el valor de la longitud del espacio.

text-transform

Establece la transformación del texto entre mayúsculas y minúsculas.

font-family

Establece la familia de fuentes que vamos a utilizar en el texto.

font-style

Establece el estilo de la fuente, puede ser *normal, italic* y *oblique.*

font-variant

Convierte las minúsculas a mayúsculas pero se mantiene un tamaño inferior al de las mayúsculas.

font-weight

Establece la intensidad de la fuente, puede ser el valor de normal, bold, bolder, lighter y otras unidades.

font-size

Establece el tamaño de la fuente, puede ser uno de los siguientes valores: xx-small, x-small, small, médium, large, x-large, xx-large, larger, smaller.

float

Establece un posicionamiento del tipo flotante, los valores pueden ser izquierda, derecha o ninguno.

clear

Establece un lado en que no se podrá tener elementos flotantes, los valores que se pueden usar son absolute, fixed, relative y static.

5 FIREBUG

El objetivo de compartir la información sobre una herramienta como Firebug es que en los siguientes capítulos del libro sea de utilidad para el lector en el momento de buscar errores y conocer mejor el funcionamiento de lo que se va leyendo.

Firebug es una extensión gratuita o complemento que es de gran utilidad cuando estamos desarrollando sitios web, ya que nos permite ver el código de nuestro sitio web en el navegador y podemos ver las referencias de los estilos, esto quiere decir que si seleccionamos un párrafo podemos ver cuál es el estilo que está haciendo efecto sobre el elemento, si se cuenta con más de un estilo también podemos ver la referencia, además si hacemos un cambio en Firebug podemos ver en tiempo real cuál es el efecto y no tenemos que ir al archivo inicial, guardar el cambio y actualizar, aunque obviamente este cambio no se hace de forma directa en el código fuente, solo queda como una vista previa con el cambio que realizamos, suele ser de gran ayuda cuando necesitamos hacer cambios de forma rápida.

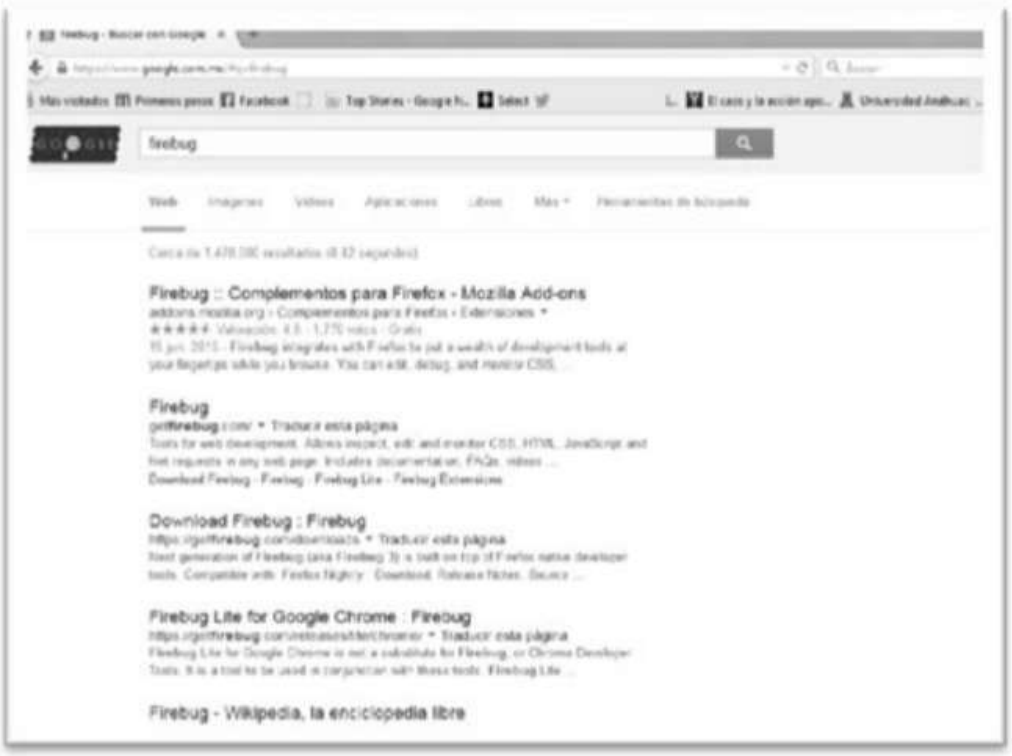
Otras de las funciones que nos permite examinar son Javascript, DOM, scripts y otros.

Los navegadores como Chrome actualmente cuentan con herramientas similares, aunque Firebug fue pionera en este tipo de herramientas.

La fundación Mozilla es uno de los pilares más importantes en los últimos avances de HTML y CSS ya que está muy involucrada en proyectos de este tipo, la mayoría de la información en español fiable que podemos encontrar sobre estas tecnologías también es desarrollada por Mozilla.

Obtener Firebug

Para obtener Firebug primero vamos abrir el navegador Firefox y después un navegador, luego tecleamos Firebug.



Después nos manda a una ventana de instalación de complementos y hacemos clic en el botón “Add to Firefox”.

Después de la descarga e instalación, nos solicita que confirmemos la instalación de Firebug.

Finalmente, después de cumplir los pasos de la instalación y aceptar las ventanas, nos va a pedir reiniciar el navegador, al hacerlo vamos a ver en nuestro navegador el icono de un insecto y ese es el de Firebug.

Abramos un sitio web de los que ya hemos creado.

Este es el código:

```
<!DOCTYPE html>

<html lang="es">

<head>

<meta charset="utf-8">

<title>Titulo de la web</title>

<link rel="stylesheet" href="css/estilo.css">

</head>
```

```
<body >
    <section>
        <h1 class="cambio">Mi web</h1>
        <p name="cambio">Esto es solo un parrafo 1 </p>
        <p name="rojo">Esto es solo un parrafo 2 </p>
        <p name="rojomini">Esto es solo un parrafo 3</p>
        <p name="azulgrande">Esto es solo un parrafo 4 </p>
        <p name="miniazul">Esto es solo un parrafo 5 </p>
    </section>
    <footer>

    </footer>
</body>
</html>
```

El archivo estilo.css es este:

```
h1.cambio{
color:#FF0000;
font-size:10em;
}
```

```
p:nth-child(2){

color:#00FF00;

}
```

En el momento de hacer clic en el icono de Firebug vamos a ver algo como lo siguiente:

Podemos ver que en la parte de abajo del navegador aparece el código HTML del documento que se está mostrando en el navegador, en el lado derecho podemos ver por defecto el CSS de las etiquetas, las clases o div que estén seleccionados.

Para conocer la herramienta recomiendo que pruebes con sitios que visites habitualmente y vas a ver los estilos que usan para cada uno de ellos, esto nos permite conocer y saber cómo se logran ciertos estilos o efectos que no sabemos cómo se crean.

CSS3 es la última versión de CSS, debemos recordar que CSS tenía como objetivo dar un formato a los archivos HTML. Actualmente se utiliza para muchas más cosas y no solamente como formato, con la combinación de HTML5 + JavaScript + CSS3 las tres tecnologías proveen una base sólida para poder desarrollar sitios y plataformas web.

Los aspectos que se destacan por la mayoría de los desarrolladores al hablar del nuevo estándar CSS son elementos como las sombras, los bordes, las animaciones, las transparencias y otros efectos de este tipo.

Un recordatorio de que es una tecnología nueva es que en algunos casos tendremos que utilizar los prefijos webkit o moz dependiendo del navegador que se utilice ya que muchos aspectos no se encuentran definidos en su totalidad en el estándar de CSS3 y es la única forma de lograr el mismo estilo en diferentes navegadores.

Propiedades CSS3

Vamos a estudiar las propiedades destacables de la última versión de CSS.

border-radius

Esta propiedad permite generar esquinas redondeadas en el elemento que se está aplicando este estilo.

El código del archivo HTML es el siguiente:

```
<!DOCTYPE html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>Titulo de la web</title>
<link rel="stylesheet" href="css/estilo.css">
</head>
<body >
    <section id="contenedor">
        <p class="texto">Probando los estilos</p>
    </section>
    <footer>

    </footer>
</body>
</html>
```

El código del estilo.css es el siguiente:

```
body{
padding:0px;
margin:0px;
}

#contenedor{
text-align:center;
padding:10px;
border:1px solid #000000;
width:400px;
margin:5px;
}

.texto{
font-size:30px;
}
```

El resultado de este código es la siguiente imagen:

Podemos ver que tenemos un borde en el contenedor y que este tiene las esquinas sin redondear, si queremos redondear el elemento, necesitamos aplicar la característica de *border-radius*.

El código del HTML es el mismo, pero el del CSS sería de esta forma:

```
body{
padding:0px;
margin:0px;
}

#contenedor{
text-align:center;
padding:10px;
border:1px solid #000000;
```

```
    width:400px;
    margin:5px;
    border-radius:30px;
    }

    .texto{
    font-size:30px;
    }
```

Cuanto mayor sea el número que se coloca en la propiedad *border-radius,* más notorio será el borde redondeado del elemento.

El resultado se ve de la siguiente forma:

A diferencia de la primera imagen podemos observar las esquinas con un redondeo. Otra forma de aplicar redondeo es asignarle valores independientes a cada esquina del elemento. El código del archivo estilo.css sería el siguiente:

```
body{
padding:0px;
margin:0px;
}

#contenedor{
text-align:center;
padding:10px;
border:1px solid #000000;
width:400px;
margin:5px;
border-radius:10px 20px 30px 40px;
}

.texto{
font-size:30px;
}
```

Como vemos tiene cuatro valores, cada valor es una esquina de aplicación al redondeo, este sería el orden:

border-radius: superior-izquierda, superior-derecha, inferior derecha, inferior izquierda.

El resultado es el siguiente:

Los valores comienzan por el valor de la esquina que se encuentra marcado con el número 1 en la imagen y termina con la esquina marcada con el valor 4.

border-image

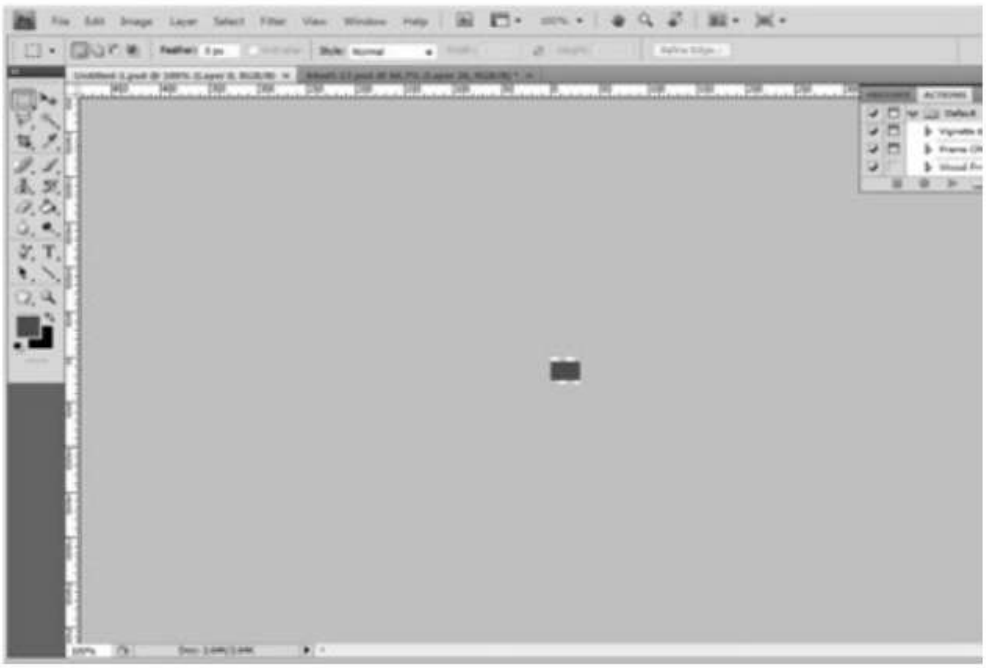
Esta es una de las nuevas propiedades que a simple vista podemos notar como un gran avance con CSS, con este atributo podemos colocar una imagen como borde, la imagen se vuelve una especie de borde al rodear al elemento.

La recomendación para las imágenes que vamos a usar en *border-image* es el formato *png*.

La imagen que vamos a usar para el ejemplo es un cuadrado de color rojo y en formato *png*, está almacenada en la ruta:

```
images/img.png
```

Así es cómo se ve:



El código del archivo HTML es:

```
<!DOCTYPE html>

<html lang="es">

<head>

<meta charset="utf-8">

<title>Titulo de la web</title>

<link rel="stylesheet" href="css/estilo.css">

</head>
```

```
<body >
    <section id="contenedor">
        <p class="texto">Probando los estilos</p>
    </section>
    <footer>

    </footer>
</body>
</html>
```

El código del archivo estilo.css es:

```
body{
padding:0px;
margin:0px;
}

#contenedor{
text-align:center;
padding:10px;
border:5px solid ;
width:400px;
```

```
    margin:5px;
    border-image: url("../images/img.png") 7 strech;
    }

    .texto{
    font-size:30px;
    }
```

Cuando usamos la propiedad *border-image* tenemos tres parámetros que colocar: el primero es la ruta de la imagen, el segundo es el tamaño y el último parámetro es la forma de distribuir la imagen. Podemos colocar los valores: *stretch, round* y *repeat*.

El resultado es el siguiente:

box-shadow

Las sombras son un efecto que se ha buscado durante mucho tiempo, en la mayoría de los casos se recurría a scripts para lograr este efecto, en la actualidad utilizamos esta propiedad para lograrlo.

Nuestro archivo HTML es el siguiente:

```
<!DOCTYPE html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>Titulo de la web</title>
<link rel="stylesheet" href="css/estilo.css">
</head>
<body >
    <section id="contenedor">
        <p class="texto">Probando los estilos</p>
    </section>
    <footer>

    </footer>
</body>
</html>
```

El código del archivo estilo.css es el siguiente:

```
body{
padding:0px;
margin:0px;
}

#contenedor{
text-align:center;
padding:10px;
border:1px solid #000000;
width:400px;
margin:5px;
box-shadow:rgb(150,150,150)10px 10px;
}

.texto{
font-size:30px;
}
```

En el atributo *box-shadow* podemos ver que se tiene el valor de un color usando el formato RGB, ahí interpretamos rgb (150,150,150), los dos valores a continuación, 10px 10px, son los que nos permiten colocar un desplazamiento de la sombra e indican el valor horizontal y vertical. Se pueden tener valores positivos o negativos.

El resultado de aplicar este estilo es la siguiente imagen:

Si colocamos los valores negativos, podemos ver cómo se desplaza la sombra en posición contraria, este es el código CSS con la modificación:

```
body{
padding:0px;
margin:0px;
}

#contenedor{
text-align:center;
padding:10px;
border:1px solid #000000;
```

```
width:400px;
margin:5px;
box-shadow:rgb(150,150,150)-10px -10px;
}

.texto{
font-size:30px;
}
```

Esta es la imagen del resultado:

Podemos agregar otro valor para lograr un efecto más parecido a una sombra, a este valor se le conoce como la distancia de desenfoque. El archivo HTML sigue siendo el mismo, el archivo CSS cambia en el parámetro, sería el siguiente código:

```
body{
padding:0px;
margin:0px;
}

#contenedor{
text-align:center;
padding:10px;
border:1px solid #000000;
width:400px;
margin:5px;
box-shadow:rgb(150,150,150)10px 10px 10px;
}

.texto{
font-size:30px;
}
```

El resultado lo podemos ver en la siguiente imagen:

text-shadow

El atributo anterior que vimos solo se aplicaba para las cajas, pero *text-shadow* es exclusivo para el texto.

El archivo HTML es el siguiente:

```
<!DOCTYPE html>

<html lang="es">

<head>

<meta charset="utf-8">

<title>Titulo de la web</title>

<link rel="stylesheet" href="css/estilo.css">

</head>
```

```
<body >
    <section id="contenedor">
        <p class="texto">Probando los estilos</p>
    </section>
    <footer>

    </footer>
</body>
</html>
```

El código del archivo estilo.css es el siguiente:

```
body{
padding:0px;
margin:0px;
}

#contenedor{
text-align:center;
padding:10px;
border:1px solid #000000;
width:400px;
```

```
    margin:5px;
    text-shadow:rgb(150,150,150)5px 5px 5px;
    }

    .texto{
    font-size:30px;
    }
```

Al igual que la propiedad anterior, colocamos el valor del color en el formato RGB, los otros tres valores van en este orden: distancia horizontal desde la sombra al objeto, distancia vertical y radio de desenfoque.

El resultado de este código lo vemos en la siguiente imagen:

column

Antes teníamos que crear contenedores y dentro de un contenedor colocar dos o más dependiendo de la distribución que necesitábamos para simular el efecto de columnas, aunque el término como tal no exista. Actualmente podemos trabajar con la propiedad “columna” para dividir de una forma sencilla al elemento que se la coloquemos.

Este es el código del archivo HTML:

```
<!DOCTYPE html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>Titulo de la web</title>
<link rel="stylesheet" href="css/estilo.css">
</head>
<body >
    <section id="contenedor">
        <p class="texto">Probando los estilos</p>
    </section>
    <footer>

    </footer>
</body>
</html>
```

Este es el código del archivo estilo.css:

```
body{
padding:0px;
margin:0px;
}

#contenedor{
text-align:center;
padding:10px;
border:5px solid ;
width:400px;

border: 1px solid #000000;
-webkit-column-count: 2; /* Chrome, Safari, Opera */
-webkit-column-gap: 20px;
-webkit-column-rule: 1px solid #000000;

-moz-column-count: 2; /* Firefox */
-moz-column-gap: 20px;
-moz-column-rule: 1px solid #000000;
```

```
column-count: 2;

column-gap: 20px;

column-rule: 1px solid #000000;



}



.texto{

font-size:30px;

}
```

Podemos ver que estamos usando el prefijo para los diferentes navegadores en cada una de las características que estamos definiendo; comenzamos con la propiedad:

```
column-count: 2;

-moz-column-count: 2;

-webkit-column-count: 2;
```

Esta propiedad nos sirve para decidir cuántas columnas vamos a crear, en este caso son 2.

En el siguiente código definimos la diferencia de las columnas:

```
column-gap: 20px;

-moz-column-gap: 20px;

-webkit-column-gap: 20px;
```

Finalmente con el último atributo definimos una línea entre las columnas:

```
column-rule: 1px solid #000000;

-moz-column-rule: 1px solid #000000;

-webkit-column-rule: 1px solid #000000;
```

El resultado es el siguiente:

filter

Los filtros son un efecto que le podemos aplicar a un elemento, se puede aplicar a casi cualquier elemento, no es exclusivo de las imágenes como se puede llegar a pensar. Para el ejemplo trabajaremos con *blur*.

Este es el código de HTML:

```
<!DOCTYPE html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>Titulo de la web</title>
<link rel="stylesheet" href="css/estilo.css">
</head>
<body >
    <section id="contenedor">
        <p class="texto">Probando los estilos</p>
    </section>
    <footer>

    </footer>
</body>
</html>
```

Este es el código del archivo estilo.css:

```
body{
padding:0px;
```

```
margin:0px;
}

#contenedor{
text-align:center;
padding:10px;
border:5px solid ;
width:400px;

-moz-filter:blur(10px);
-webkit-filter:blur(10px);
filter:blur(10px);



}

.texto{
font-size:30px;
}
```

Este es el resultado:

Podemos ver que tenemos que aplicar el prefijo de cada uno de los navegadores para que sea tomado en cuenta por la mayoría de los navegadores:

```
-moz-filter:blur(10px);

-webkit-filter:blur(10px);

filter:blur(10px);
```

El efecto que estamos utilizando en este ejemplo es *blur*, este efecto produce el efecto visual de un desenfoque, el valor mínimo es 1px y el valor máximo es 10px.

También tenemos los efectos: *brightness, contrast, grayscale, hue-rotate, invert, opacity, saturate, sepia.*

brightness es el efecto sobre la luminosidad del elemento, los valores van del 0 al 1, el código sería así:

```
-moz-filter:brightness(0.5);

-webkit-filter: brightness (0.5);

filter: brightness (0.5);
```

contrast es el efecto sobre el contraste del elemento, los valores van del 0 al 1. Por defecto se tiene el contraste al 100% y se puede aumentar en porcentaje.

```
-moz-filter:contrast(170%);

-webkit-filter:contrast (170%);

filter: contrast (170%);
```

grayscale es el efecto de una escala de grises, los valores que podemos colocar van del 0 al 1. Se pueden tomar valores por porcentaje.

```
-moz-filter:grayscale(100%);

-webkit-filter:grayscale (100%);

filter: grayscale(100%);
```

hue-rotate, con este efecto se aplica un giro en la tonalidad, el valor que debemos colocar va de 1 a 360.

```
-moz-filter:hue-rotate(180deg);

-webkit-filter:hue-rotate (180deg);

filter: hue-rotate(180deg);
```

invert, con este efecto se invierte el color del elemento produciendo un negativo, los valores se pueden colocar por porcentaje.

```
-moz-filter:invert(100%);

-webkit-filter: invert(100%);

filter:invert(100%);
```

opacity es el efecto que produce la opacidad, los valores se pueden colocar por porcentaje.

```
-moz-filter:opacity(100%);

-webkit-filter:opacity(100%);

filter:opacity(100%);
```

saturate, este efecto satura los colores del elemento, el valor puede ser por porcentaje.

```
-moz-filter:saturate(100%);

-webkit-filter:saturate(100%);

filter:saturate(100%);
```

sepia, este efecto le coloca el tono de sepia al elemento, el valor puede ser por porcentaje.

```
-moz-filter:sepia(100%);

-webkit-filter:sepia(100%);

filter:sepia(100%);
```

linear-gradient

Esta propiedad nos permite jugar con el degradado, algo que era impensable lograr y que solamente con imágenes se podía hacer, ahora lo podemos realizar con una línea.

El archivo HTML es el siguiente:

```
<!DOCTYPE html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>Titulo de la web</title>
<link rel="stylesheet" href="css/estilo.css">
</head>
<body >
    <section id="contenedor">
        <p class="texto">Probando los estilos</p>
    </section>
    <footer>

    </footer>
</body>
</html>
```

El código del archivo estilo.css es el siguiente:

```
body{
padding:0px;
```

```
margin:0px;
}

#contenedor{
text-align:center;
padding:10px;
border:1px solid #000000;
width:400px;
margin:5px;
background: linear-gradient(top,#FFFFFF,#000000);
}

.texto{
font-size:30px;
}
```

El resultado es el siguiente:

Como podemos ver en la imagen, no se nota ningún degradado aunque nuestro código es correcto, este es uno de los casos que ya se mencionó anteriormente en este libro en donde el prefijo, dependiendo del navegador, habilita la propiedad. En nuestro caso en particular si estamos utilizando Firefox agregamos el prefijo ”moz”, si usamos Opera, agregamos “o”, si usamos Safari, “webkit” y todos los que ya toman en cuenta al estándar, ya no usan el prefijo.

El código del archivo estilo.css queda entonces así:

```
body{
padding:0px;
margin:0px;
}
```

```
#contenedor{
text-align:center;
padding:10px;
border:1px solid #000000;
width:400px;
margin:5px;
background: linear-gradient(top,#FFFFFF,#000000);
background: -webkit-linear-gradient(top,#FFFFFF,#000000);
background: -o-linear-gradient(top,#FFFFFF,#000000);
background: -moz-linear-gradient(top,#FFFFFF,#000000);
}

.texto{
font-size:30px;
}
```

Los tres parámetros de este atributo son los siguientes: posición inicial, color inicial, color final. El degradado se genera a partir de esos tres parámetros.

Este sería el resultado:

Vamos a hacer la prueba de cambiar la dirección del degradado, aplicando el degradado desde la posición derecha. El código del archivo estilo.css es el siguiente:

```
body{
padding:0px;
margin:0px;
}

#contenedor{
text-align:center;
padding:10px;
border:1px solid #000000;
width:400px;
```

```
margin:5px;
background: linear-gradient(right,#FFFFFF,#000000);
background: -webkit-linear-gradient(right,#FFFFFF,#000000);
background: -o-linear-gradient(right,#FFFFFF,#000000);
background: -moz-linear-gradient(right,#FFFFFF,#000000);
}

.texto{
font-size:30px;
}
```

El resultado sería el siguiente colocando la posición de inicio derecha:

Además de colocar una posición habitual como derecha, izquierda, abajo y arriba, podemos combinar las posiciones como en el siguiente ejemplo, este es el código del archivo estilo.css:

```
body{
padding:0px;
margin:0px;
}

#contenedor{
text-align:center;
padding:10px;
border:1px solid #000000;
width:400px;
margin:5px;
background: linear-gradient(top right,#FFFFFF,#000000);
background: -webkit-linear-gradient(top right,#FFFFFF,#000000);
background: -o-linear-gradient(top right,#FFFFFF,#000000);
background: -moz-linear-gradient(top right,#FFFFFF,#000000);
}

.texto{
```

```
font-size:30px;

}
```

En el código anterior le indicamos que la posición es la esquina superior derecha, también lo podemos indicar por ángulo, por ejemplo: 45deg. El resultado se ve de la siguiente forma:

Además de indicarle la posición, podemos agregar colores parar generar un efecto de degradado más completo, vamos a hacer un degradado en nuestro elemento en el que la mitad en el eje vertical sea la parte más clara y las orillas lo más oscuro. El código del archivo estilo.css es el siguiente:

```
body{

padding:0px;

margin:0px;
```

```
}

#contenedor{
text-align:center;
padding:10px;
border:1px solid #000000;
width:400px;
margin:5px;
background: linear-gradient(top, #000000,#FFFFFF,#000000);
background: -webkit-linear-gradient(top, #000000 ,#FFFFFF,#000000);
background: -o-linear-gradient(top, #000000 ,#FFFFFF,#000000);
background: -moz-linear-gradient(top, #000000 ,#FFFFFF,#000000);
}

.texto{
font-size:30px;
}
```

El resultado se ve de la siguiente forma:

Podemos colocar el valor de "*transparent*" como valor inicial en un degradado, este es el código del archivo estilo.css:

```
body{
padding:0px;
margin:0px;
}

#contenedor{
text-align:center;
padding:10px;
border:1px solid #000000;
```

```
    width:400px;
    margin:5px;
    background: linear-gradient(top, transparet,#000000);
    background: -webkit-linear-gradient(top, transparent,#000000);
    background: -o-linear-gradient(top, transparent,#000000);
    background: -moz-linear-gradient(top, transparent,#000000);
    }

    .texto{
    font-size:30px;
    }
```

radial-gradient

Este atributo trata también sobre el degradado pero se centra en los degradados radiales, su uso es un poco más complejo porque tenemos más cantidad de parámetros, pero con un ejemplo lo entenderemos. Comenzamos mencionando que tenemos un parámetro que nos permite colocar la posición, tenemos los valores:

center, top, bottom, left, right

Como en el degradado anterior también podemos combinar las palabras. Dentro de los nuevos parámetros tenemos que indicar la forma del degradado, puede ser círculo o elipse.

Este es el código del archivo HTML:

```
<!DOCTYPE html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>Titulo de la web</title>
<link rel="stylesheet" href="css/estilo.css">
</head>
<body >
    <section id="contenedor">
        <p class="texto">Probando los estilos</p>
    </section>
    <footer>

    </footer>
</body>
</html>
```

Este es el código del archivo estilo.css:

```
body{
padding:0px;
margin:0px;
```

```
}

#contenedor{
text-align:center;
padding:10px;
border:1px solid #000000;
width:400px;
margin:5px;
background: radial-gradient(center, ellipse, #ffffff,#000000);
background: -webkit-radial-gradient(center, ellipse, #ffffff,#000000);
background: -o-radial-gradient(center, ellipse, #ffffff,#000000);
background: -moz-radial-gradient(center, ellipse, #ffffff,#000000);
}

.texto{
font-size:30px;
}
```

El código anterior nos demuestra la implementación básica del degradado radial, en donde le pasamos el primer parámetro “center” que es la posición de partida, después “ellipse” que es la forma del degradado, tenemos “ellipse” y “circle”. Finalmente le pasamos el color inicial y el color final para realizar el degradado.

El resultado de este código se ve de la siguiente forma:

outline

La propiedad *outline* nos permite aplicar un segundo borde o lo podemos describir como un borde separado del primer borde que definimos previamente.

Este es el archivo HTML:

```
<!DOCTYPE html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>Titulo de la web</title>
<link rel="stylesheet" href="css/estilo.css">
</head>
```

```
<body >
    <section id="contenedor">
        <p class="texto">Probando los estilos</p>
    </section>
    <footer>

    </footer>
</body>
</html>
```

El código del archivo estilo.css sería este:

```
body{
padding:0px;
margin:0px;
}

#contenedor{
text-align:center;
padding:10px;
border:5px solid ;
width:400px;
margin:50px;
```

```
    border: 1px solid #000000;



    outline: 4px dashed #000000;

    outline-offset:15px;

    }



    .texto{

    font-size:30px;

    }
```

Primero definimos un borde como los que ya conocemos:

```
    border: 1px solid #000000;
```

Después, utilizamos la propiedad *outline* y definimos el estilo del segundo borde que estamos declarando, la definición es muy similar al primer borde; en realidad son idénticas, colocamos el grueso del borde, el estilo y el color.

```
    outline: 4px dashed #000000;
```

Finalmente, utilizamos la propiedad *outline-offset* para indicar la distancia entre el primer borde y el segundo, en nuestro caso colocamos el valor de 15px:

```
    outline-offset:15px;
```

Este es el resultado:

transform

Esta propiedad es una de las más llamativas visualmente, ya que un elemento que tenemos puede sufrir cambios dependiendo del tipo de transformación que apliquemos, tenemos 4 tipos disponibles: *rotate*, *scale*, *skew*, *translate*.

El código HTML para los ejemplos que vamos a ver es el siguiente:

```
<!DOCTYPE html>

<html lang="es">

<head>

<meta charset="utf-8">

<title>Titulo de la web</title>

<link rel="stylesheet" href="css/estilo.css">

</head>
```

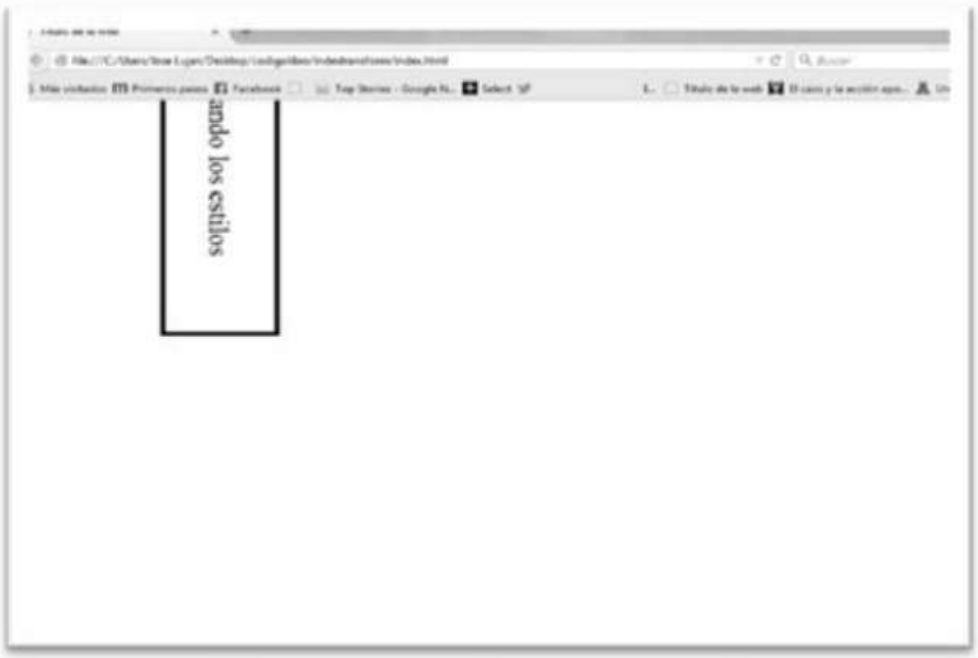
```
<body >
    <section id="contenedor">
        <p class="texto">Probando los estilos</p>
    </section>
    <footer>

    </footer>
</body>
</html>
```

rotate es el efecto que nos ayuda a rotar el elemento, el valor lo debemos especificar en grados, el código es el siguiente:

```
body{
padding:0px;
margin:0px;
}

#contenedor{
text-align:center;
padding:10px;
border:5px solid ;
width:400px;
```

```
-moz-transform:rotate(90deg);

-webkit-transform:rotate(90deg);

transform:rotate(90deg);

}


.texto{

font-size:30px;

}
```

Así se vería el resultado:



scale, este efecto nos permite escalar el elemento dependiendo del valor que le pasamos, le tenemos que pasar dos valores (X,Y). Podemos decir que el primer valor es una escala horizontal y el segundo valor es sobre la escala vertical, en caso de solo

pasarle un valor por defecto se le aplica a los dos ejes; el código del archivo estilo.css es el siguiente:

```
body{
padding:0px;
margin:0px;
}

#contenedor{
text-align:center;
padding:10px;
border:5px solid ;
width:400px;
-moz-transform:scale(2,2);
-webkit-transform:scale(2,2);
transform:scale(2,2);
}

.texto{
font-size:30px;
}
```

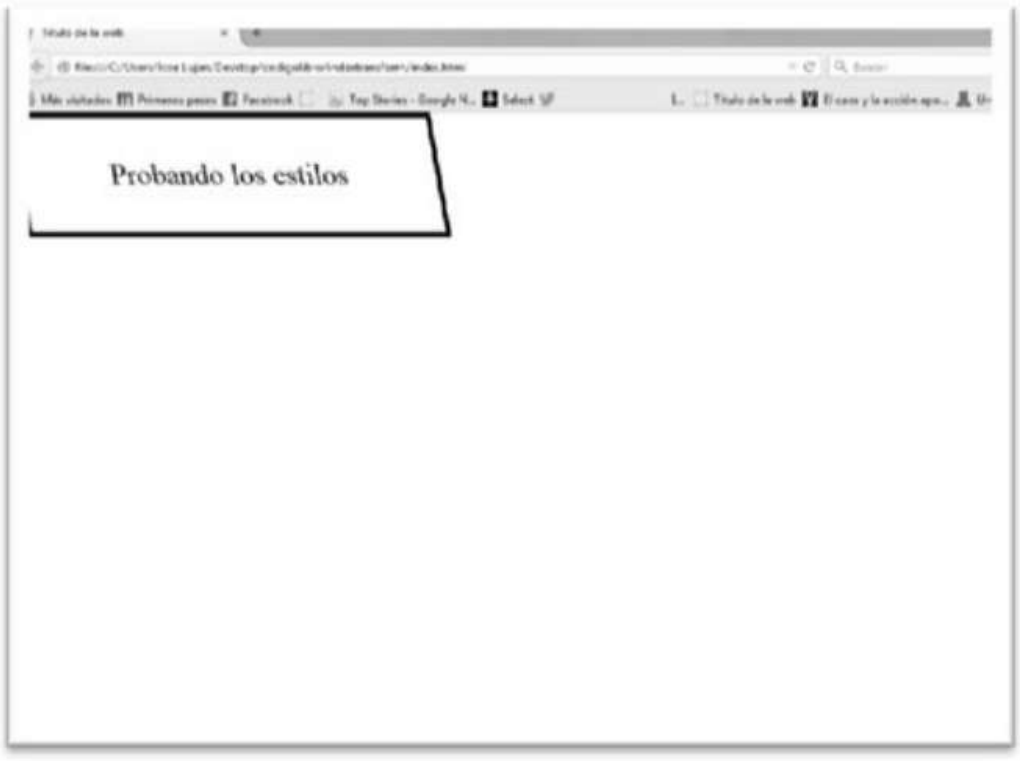
Este es el resultado de aplicar *scale:*

skew es el efecto que cambia la simetría de un elemento, los valores que utilizamos tienen que ser en grados; el código del estilo.css sería el siguiente:

```
body{
padding:0px;
margin:0px;
}

#contenedor{
text-align:center;
padding:10px;
border:5px solid ;
width:400px;
```

```
-moz-transform:skew(10deg);

-webkit-transform:skew(10deg);

transform:skew(10deg);

}



.texto{

font-size:30px;

}
```

Así se ve el efecto:

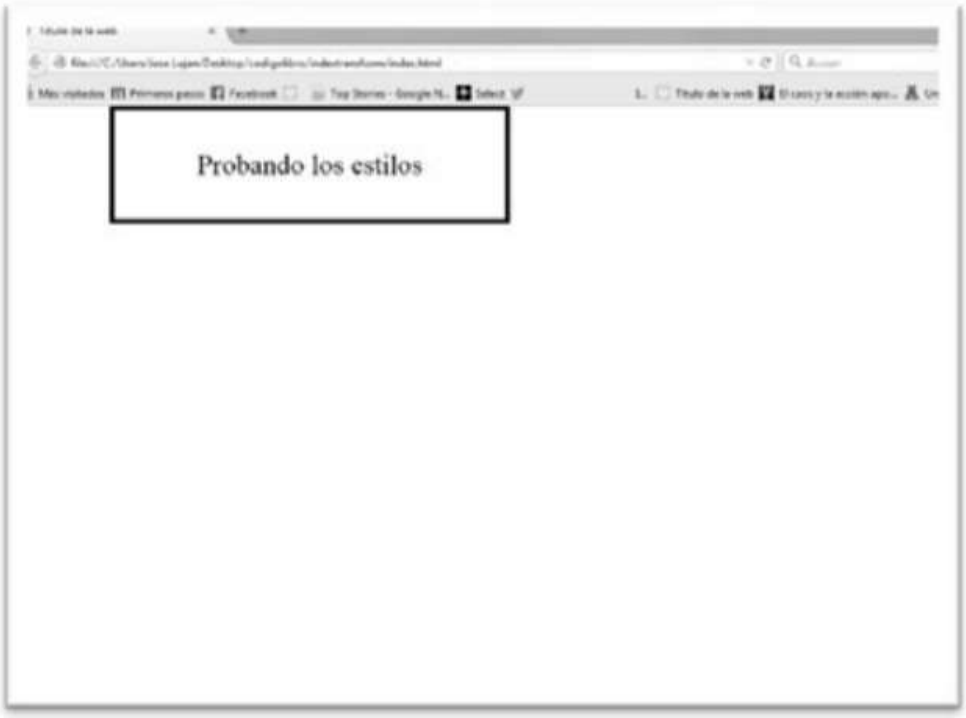


translate es el efecto que tenemos disponible para mover de la posición inicial al elemento. La posición 0,0 es el lado superior e izquierdo de la pantalla, siendo el máximo valor el lado derecho inferior de la pantalla. Se debe indicar dos valores, el del eje X y el del Y, para las X utilizamos translateX y para las Y utilizamos translateY.

El código en el eje de la X es el siguiente:

```
body{
padding:0px;
margin:0px;
}

#contenedor{
text-align:center;
padding:10px;
border:5px solid ;
width:400px;
-moz-transform:translateX(100px);
-webkit-transform:translateX(100px);
transform:translateX(100px);
}

.texto{
font-size:30px;
}
```

El resultado se ve así:



El código para aplicar el efecto en el eje de la Y es el siguiente:

```
body{
padding:0px;
margin:0px;
}

#contenedor{
text-align:center;
padding:10px;
border:5px solid ;
```

```
    width:400px;
    -moz-transform:translateY(100px);
    -webkit-transform:translateY(100px);
    transform:translateY(100px);
    }

    .texto{
    font-size:30px;
    }
```

Y este el resultado:

transition

Esta propiedad nos permite hacer efectos que visualmente se vean como una animación. El objetivo principal es suavizar estos efectos, para este ejemplo vamos a utilizar el selector *hover* para tener un tamaño inicial y después asignarle un tamaño final con este selector, lamentablemente este tipo de efectos no se pueden observar en un material impreso como este libro, pero el lector va a poder probar el código que colocaremos en este apartado por su cuenta.

El código del archivo html es:

```
<!DOCTYPE html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>Titulo de la web</title>
<link rel="stylesheet" href="css/estilo.css">
</head>
<body >
    <section id="contenedor">
        <p class="texto">Probando los estilos</p>
    </section>
    <footer>

    </footer>
```

```
</body>
</html>
```

El código del archivo estilo.css es este:

```
body{
padding:0px;
margin:0px;
}

#contenedor{
text-align:center;
padding:10px;
border:5px solid ;
width:400px;
-moz-transition: all 2s ease;
-webkit-transition: all 2s ease;
transition: all 2s ease;

}

#contenedor:hover{
```

```
width:800px;
}

.texto{
font-size:30px;
}
```

Con *transition* tenemos la siguiente estructura de 4 parámetros, no siempre tenemos que utilizarlos todos, depende de lo que necesitemos:

transition: propiedad, tiempo, función de tiempo, retardo.

El primer parámetro es la propiedad, que en este caso es el cambio, el segundo parámetro es el tiempo en el que se ejecutará, el tercer parámetro pueden ser varios valores: *ease, ease-in, ease-out, ease-in-out, linear*. Cada palabra es una curva de Bézier, yo recomiendo probarlas ya que será más fácil determinar cuál es la conveniente para cada ejercicio o efecto que deseamos. El último parámetro es un retardo.

7 JAVASCRIPT

En la actualidad la mayoría de los sitios web en su estructura más básica cuentan con tres tecnologías: HTML, CSS y JavaScript, en otras palabras decimos que HTML es la estructura, CSS es el estilo y JavaScript es la programación.

JavaScript es un lenguaje de programación ya algo viejo dentro de la historia de web, la sintaxis de este lenguaje es muy similar a la del lenguaje C, esto es algo común en la programación hoy en día. Todos los navegadores actuales soportan JavaScript: Firefox, Chrome, Explorer, Opera, Safari, entre otros.

Javascript es un lenguaje que utiliza la filosofía de la programación orientada a objetos, existe una eterna discusión sobre esto ya que muchos autores argumentan que no es un lenguaje orientado a objetos y otros tantos argumentan que sí. Nosotros no entraremos en esa discusión, recomendaría en todo caso consultar fuente oficiales como la de Mozilla para obtener información teórica del lenguaje; Por nuestra parte analizaremos, estudiaremos y ejecutaremos el ejercicio sin mayores complicaciones.

Una de las principales preguntas es: ¿qué relación hay entre Java y JavaScript?

La respuesta es simple: ninguna. Java es un lenguaje completamente diferente a Javascript, puede ser que en la sintaxis en algunos casos se parezcan, pero es porque ambos son lenguajes que derivan de C como muchos otros, pero solamente eso. Java y JavaScript no tienen nada que ver.

JavaScript es un lenguaje conocido por "ser un lenguaje del lado del cliente", esto se entiende mejor cuando explicamos que en el mundo del desarrollo web tenemos dos lados: el lado del cliente y el lado del servidor. El lado del servidor es donde físicamente están nuestros archivos, bases de datos y toda la infraestructura que da forma a nuestro proyecto; el lado del cliente es el navegador por el que el usuario accede a nuestro sitio web, JavaScript es un lenguaje que se ejecuta en el navegador, por eso decimos que es un lenguaje de programación del lado del cliente.

Una observación más que podemos hacer después de leer el párrafo anterior es que entonces JavaScript utiliza el navegador como su entorno, ya que ahí es donde se ejecuta.

JavaScript, a diferencia de HTML y CSS, sí es un lenguaje de programación, ya que tenemos conceptos como el de variables, ciclos, condicionales y otros que iremos viendo a lo largo de este capítulo.

Cómo escribir JavaScript en nuestro sitio web

Vamos a comenzar con una forma sencilla, pero no la correcta. Esto es solo para comprender un poco mejor cómo funciona la lógica de ejecución de javascript. Comenzamos colocando en nuestro *index* una etiqueta que es <script>, el código de nuestro archivo html se ve así:

```
<!DOCTYPE html>

<html lang="es">

<head>

<meta charset="utf-8">

<title>Titulo de la web</title>
```

```
<link rel="stylesheet" href="css/estilo.css">


</head>
<body >
      <section id="contenedor">
      </section>
      <footer>


      </footer>

<script>

 document.write("Hola!");


</script>
</body>
</html>
```

Durante mucho tiempo se colocaba la etiqueta *<script>* en la etiqueta *<head>*, pero en la actualidad como parte de las buenas prácticas se considera mejor colocarla al final del archivo por una razón: el código que tengamos se va ejecutar al acabar de cargar todos los elementos, aunque existen otras formas de asegurarnos que se ejecute el código Javascript hasta que finalicemos la descarga de todo el archivo HTML y asegurar así su funcionamiento correcto.

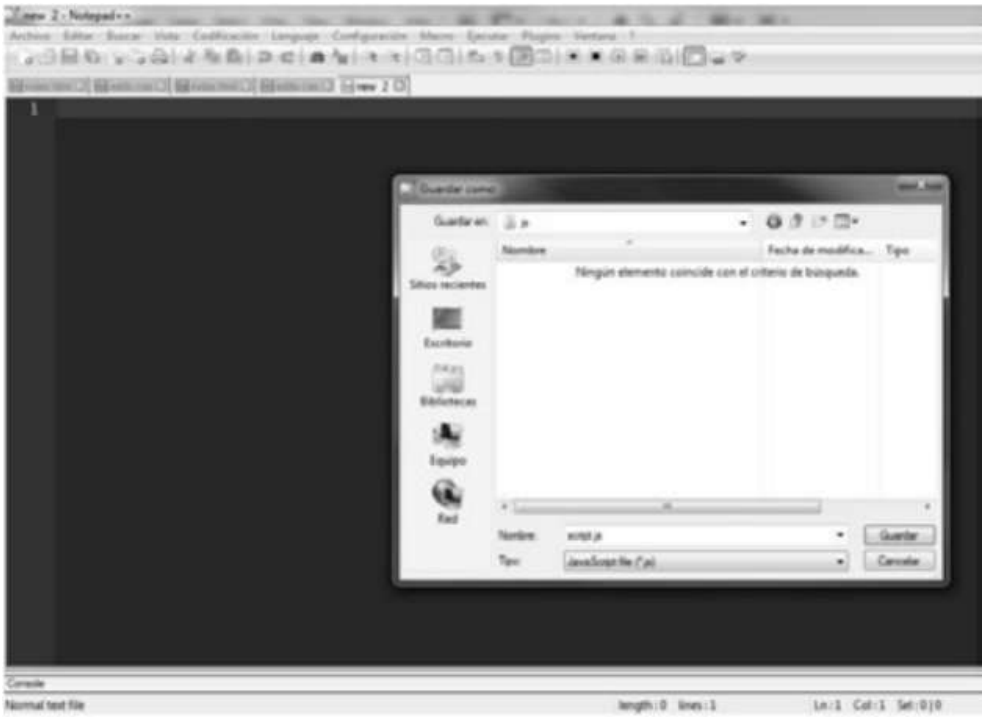
Para averiguar si estamos ejecutando el código Javascript abrimos el archivo en un navegador y deberíamos de ver un resultado similar al siguiente:

Si vemos escrita la línea de código escrita entre las etiquetas *<script>*, entonces está funcionando sin problemas.

Separar el código Javascript de nuestro archivo html

Así como en su momento separamos el estilo del archivo HTML, vamos a crear un archivo para el script y así tener separados los archivos que involucran a nuestro sitio web, el archivo HTML, el archivo CSS y el archivo Javascript.

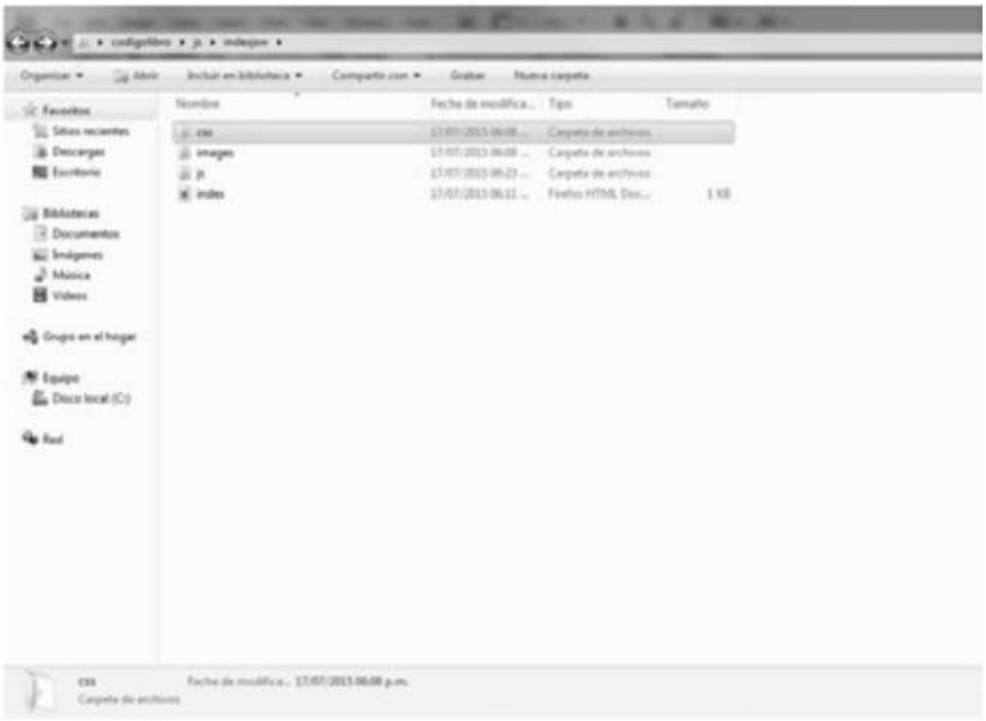
Creamos un archivo con el editor de texto que estemos usando, después seleccionamos “guardar como” y seleccionamos “Javascript” como en la siguiente imagen:



El nombre que yo le coloqué es script, pero puede ser cualquier otro. Además, lo coloqué en un directorio de nombre “js” (es una forma corta de escribir Javascript), así puedes encontrarlo también en internet.

Se considera buena práctica separar por directorios los tipos de archivos, en nuestro directorio principal los archivos HTML, en un directorio de nombre “css” los archivos de estilos y en un directorio “js” los archivos de Javascript, también podemos crear un directorio “images” para las imágenes.

Se vería como la siguiente imagen:



Ahora finalmente vamos a indicarle a nuestro archivo HTML dónde se encuentra el archivo JS que queremos utilice.

Agregaremos en el archivo HTML la fuente del archivo de Javascript:

```
<!DOCTYPE html>

<html lang="es">

<head>

<meta charset="utf-8">

<title>Título de la web</title>

<link rel="stylesheet" href="css/estilo.css">
```

```
</head>
<body >
    <section id="contenedor">

    </section>
    <footer>

    </footer>
 <script src="../js/script.js"></script>
 </body>

</html>
```

No está limitado a un archivo, podríamos incluir más archivos Javascript en caso de ser necesario, solo tendríamos que agregar otra línea para la referencia:

```
<script src="../js/script.js"></script>
<script src="../js/scriptnuevo.js"></script>
<script src="../js/scriptmasnuevo.js"></script>
```

Sintaxis

La sintaxis es un conjunto de reglas o normas que debemos de seguir cuando estamos escribiendo el código de programación. Cada lenguaje de programación tiene su propia sintaxis, aunque muchas de las reglas se pueden parecer en realidad debemos de tener cuidado en el momento de escribir, ya que un error o la falta de un carácter puede generar un comportamiento no adecuado en nuestro programa.

En Javascript tenemos las siguientes reglas:

- Siempre que terminamos una instrucción se escribe el carácter de punto y coma ";" (Javascript no va a generar un error si no lo haces, pero es una recomendación hacerlo).
- Los comentarios en el código se escriben con el carácter de diagonal 2 veces o con la diagonal seguida de un asterisco y así de nuevo para cerrar. Veamos los ejemplos:

```
//Esto es un comentario de una línea

/*Estos

Son

Muchos

Comentarios*/
```

- Javascript es sensible a mayúsculas y minúsculas, es decir, que no es lo mismo Arbol, aRbol, arBol, arbOl, arboL. Para Javascript son 5 palabras diferentes.

Estas son las principales reglas de sintaxis que se usan en Javascript, algunas otras las vamos a ir viendo poco a poco, pero además haremos el comentario cuando sea conveniente en donde se señalan las buenas prácticas.

Variables

Uno de los primeros conceptos que debemos de entender en la programación es el de "variable". Una variable es un espacio en memoria que estamos reservando y al que le debemos asignar un nombre. Las variables las utilizamos cuando el programa se está ejecutando, por ejemplo una computadora tiene en ejecución 10 programas y cada programa tiene sus variables, el problema es que solo tenemos un disco duro, una memoria ram y en general un acceso a los dispositivos de memoria, por eso creamos variables para reservar esa memoria y que así nadie la pueda usar.

Si no reserváramos la memoria, tendríamos el problema de que la información guardada se sobrescribiría por otros programas y entonces en el momento de volver a leer encontraríamos información que simplemente no serviría para lo que estamos haciendo.

El concepto de variable es la base fundamental de cualquier lenguaje de programación.

En Javascript declaramos una variable de la siguiente forma:

```
var numero = 10;
```

Para declarar una variable se coloca primero la palabra reservada "var", esta palabra lo que hace es indicar que lo que sigue es la declaración de una variable, continuamos con el nombre de la variable, en nuestro ejemplo la palabra que usamos es número, pero se puede usar cualquier palabra. Para colocar el nombre de una palabra se recomienda que usemos letras minúsculas; además, el nombre no puede llevar espacios entre letras o palabras.

Finalmente le estamos asignando el valor al mismo tiempo que la declaramos, podemos también separar el proceso en dos líneas:

```
var numero;

numero = 10;
```

La primera línea es la declaración y la segunda línea es la asignación de valor.

Como vemos se puede hacer en dos pasos o en uno, eso lo decide el desarrollador dependiendo de la ocasión. Algo que debemos resaltar también es que en Javascript las líneas terminan con un carácter de punto y coma (“;”).

Tipos de variables

En Javascript tenemos distintos tipos de variables, esto quiere decir que son de diferente naturaleza y eso nos sirve porque en cada caso podemos utilizarlas y manipularlas dependiendo de nuestra necesidad.

Tipo entero

Las variables de tipo entero son las que son números y no están fraccionados, por ejemplo:

```
var numero = 7;
```

Tipo flotante

Las variables de tipo flotante son los números que están fraccionados o con decimales, por ejemplo:

```
var temperatura = 32.5;
```

Tipo cadena de texto

Las variables de tipo cadena de texto pueden ser cualquier cadena o conjunto de caracteres, lo importante es limitar el inicio y final de la cadena entre comillas dobles o comillas simples, aunque la cadena contenga un número si este número está entre comillas, deja de ser una variable de tipo número y automáticamente se vuelve una cadena de caracteres, por ejemplo:

```
var cadena = “José Dimas Luján Castillo”;

var cadena_nueva = ‘ Hola #&/() Mundo’;
```

Esta es una variable del tipo número:

```
var numero = 998;
```

Esta es una variable del tipo cadena:

```
var numero = “998”;
```

Debemos de observar que la diferencia es que los números están entre las comillas en el del tipo cadena.

Tipo booleano

El tipo booleano, en inglés *boolean*, es un tipo de variable que solo puede tener dos valores, el valor de falso o el valor de verdadero. Es muy importante comentar que las palabras falso o verdadero tienen que estar escritas exactamente iguales, es decir, en minúsculas para que sea ese valor, ya que son palabras reservadas, esto quiere decir que son exclusivamente para referir a un tipo booleano, el ejemplo de estas variables sería el siguiente:

```
var variablefalso = false;

var variableverdadero = true;
```

Tipo arreglo

Las variables de tipo arreglo también se conocen como vectores, o “array” en inglés. Estas variables funcionan como un conjunto o una colección de datos que pueden ser del mismo tipo o pueden ser de diferentes tipos.

Supongamos que queremos tener los números del 1 al 10 juntos, lo que haríamos es lo siguiente:

```
var num1 = 1;

var num2 = 2;
```

```
var num3 =3;

var num4 =4;

var num5 = 5;

var num6 = 6;

var num7 = 7;

var num8=8;

var num9=9;

var num10=10;
```

El código anterior no está mal, pero en realidad lo que necesitamos no son variables por separado, necesitamos un conjunto de datos que en este caso sean del mismo tipo, del tipo entero. Para estos casos se utilizan los arreglos. Para declarar un arreglo de números, haríamos lo siguiente:

```
var numeros = [1,2,3,4,5,6,7,8,9,10];
```

Cada elemento que colocamos en el arreglo tiene una posición, tenemos 10 números, por lo tanto tenemos 10 posiciones.

Para seleccionar una posición del arreglo en Javascript, hacemos lo siguiente:

```
document.write(numeros[6]);
```

Para acceder a una posición, colocamos el nombre del arreglo y entre corchetes el número de la posición a la que queremos acceder. En el código anterior hemos mandado a imprimir la posición 6 del arreglo números. Entonces el resultado es el siguiente:

```
7
```

Te preguntarás: ¿por qué el número 7?

Los arreglos comienzan desde la posición 0, esto quiere decir que en nuestro ejercicio sería algo así:

Posición	Número
0	1
1	2
2	3
3	4
4	5
5	6
6	7
7	8
8	9
9	10

En la tabla anterior podemos ver más fácil lo que comentamos, el número 1 que pusimos se encuentra en la posición 0, el número 2 se encuentra en la posición 1 y así sucesivamente.

Si quisiéramos crear un arreglo de números, haríamos lo siguiente:

```
var nombres = [“José”,”Juan”,”Pedro”,”Hugo”,”Quique”, “Miguel”];
```

Como vemos en el código anterior lo único que cambia es que los nombres están entre comillas, ya que son cadenas de texto.

Operadores matemáticos

Los operadores son los que nos van a comenzar a enseñar la utilidad de la programación, hasta ahora solo hemos creado variables y asignados valores, pero la mayoría de las operaciones que se realizan tienen que ver con los operadores matemáticos.

Operador de asignación

Ya lo vimos pero no le dimos la importancia como operador, el operador de asignación es “=”. El signo de "=" es un operador que se utiliza para asignar un valor, este sería el ejemplo:

```
var numero = 10;
```

Operador de suma

El operador de suma (+) nos sirve para realizar la operación de adición. Por ejemplo vamos a sumar dos números:

```
document.write(5+5);
```

El resultado de esto sería 10 como vemos en la siguiente imagen:

También podemos sumar 2 variables:

```
var numA = 10;

var numB = 11;

var resultado = numA + numB;

document.write(resultado);
```

El resultado es el siguiente:

Operador de resta

El operador de resta (-), que es el guion, se puede utilizar en las mismas operaciones que con el operador de suma, obviamente realizando la operación contraria, este es un ejemplo:

```
var numA = 10;

var numB= 9;

var resultado = numA-numB;

document.write(resultado);
```

Este es el resultado de ejecutar el código:

Operador de multiplicación

El operador de multiplicación es el asterisco (*), este sería un ejemplo de una multiplicación en Javascript:

```
var numA = 10;
var numB= 9;
var resultado = numA*numB;
document.write(resultado);
```

Así se vería el resultado:

Operador de división

En la división contamos con 2 operadores, el operador (/) y el (%). En realidad es una división, pero estamos obteniendo otro tipo de valor.

En el caso de la diagonal tenemos lo siguiente:

```
var numA = 12;

var numB= 5;

var resultado = numA/numB;

document.write(resultado);
```

El resultado es 2.4 como la imagen, este operador es el que realiza una división como normalmente la conocemos.

Ahora utilizamos el siguiente operador (%), este operador representa una división pero en la que nos interesa el resultado del resto, en nuestro caso estamos haciendo la división de 12/5 en la que el resultado es 2.4, pero cuando queremos saber el resto es que nos interesa saber lo que sobró, decimos entonces:

¿Cuántas veces cabe el 5 en el 12?

Sabemos que cabe 2 veces y sobra el número 2. Entonces el resultado será lo que "sobró", a este operador se le conoce como el operador de módulo o división resto.

El código es el siguiente:

```
var numA = 12;
var numB= 5;
var resultado = numA%numB;
document.write(resultado);
```

Operador de incremento

El operador de incremento (++) nos sirve para ir realizando un incremento paulatino de una variable, por ejemplo:

```
var numero=0;

numero++;//es equivalente a decir numero=numero+1;

numero++;

numero++;



document.write(numero);
```

Si analizamos el código anterior podemos observar lo siguiente, la primera línea solo inicializa la variable con un valor de 0:

```
var numero=0;
```

A continuación le decimos a la variable número que incremente 1.

```
numero++;
```

Esto lo hacemos dos veces más:

```
numero++;

numero++;
```

Entonces al indicarle que incremente 3 veces le decimos que del valor 0 pase a 3. El resultado de estas instrucciones es el siguiente:

Operador de decremento

Este operador (--) es similar al operador de incremento pero lo contrario, lo que hace es decrementar el valor de la variable, este es el código de ejemplo:

```
var numero=0;

numero--;//es equivalente a decir numero=numero-1;

numero--;

numero--;



document.write(numero);
```

El resultado es el siguiente:

Operadores relacionales

Los operadores relaciones son los que se utilizan en instrucciones como las condicionales, el objetivo es evaluar un dato y a partir de esa evaluación tomar una decisión dentro del código.

Operador	Nombre	Cómo se usa
>	Mayor que	A > B
<	Menor que	A < B
>=	Mayor o igual	A >= B
<=	Menor o igual	A <= B
==	Igual que	A == B
¡=	Distinto de	A ¡= B

Operadores lógicos

Los operadores lógicos nos sirven para realizar la combinación de evaluaciones y así realizar operaciones más complejas.

Cuando hablamos de una evaluación nos referimos a lo siguiente:

```
10 > 1
```

El resultado de esa evaluación es VERDADERO. ¿Por qué?

Estamos evaluando si 10 es mayor que 1, como la sentencia se cumple, es decir, SÍ, 10 es mayor que el número 1.

Si utilizamos otro operador esto cambia:

```
10<1
```

El resultado en este caso es FALSO. ¿Por qué?

¿10 es menor que 1? NO, por lo tanto el resultado de esta evaluación es FALSO.

Operador AND

Este operador nos sirve para combinar dos evaluaciones y se utilizan los siguientes caracteres para usarlo (&&).

```
EVALUACION1 && EVALUACION2
```

Lo representaríamos de la siguiente forma:

```
Variable1 && Variable2
```

Al tener dos evaluaciones ya sabemos que existen estas posibles combinaciones:

Evaluacion1	Evaluacion2	Resultado
True	True	True
True	False	False
False	True	False
False	False	False

Como vemos solo se obtiene el valor de verdadero en el momento de que la evaluación 1 y la evaluación 2 arrojen el valor de verdadero.

Operador OR

El operador OR se ejecuta con el símbolo (||). La tabla de resultados es diferente, con que al menos una evaluación sea verdadera el resultado final siempre será verdadero.

Evaluacion1	Evaluacion2	Resultado
True	True	True
True	False	True
False	True	True
False	False	False

Operador negación

Es un operador (!) que se utiliza para obtener el valor contrario.

Variable	Valor negado
Verdadero	Falso
Falso	Verdadero

Estructuras de control

Cuando programamos una de las primeras cosas que tenemos que saber es manejar el control de flujo de un programa, en Javascript y en cualquier lenguaje de programación contamos con instrucciones específicas para manejar diferentes secuencias de instrucciones dependiendo de la lógica de nuestro programa.

Estructura de control IF

Esta es la estructura más común en los lenguajes de programación, casi siempre es la misma sintaxis. Esta estructura de control nos sirve cuando queremos evaluar algo y a partir de si se cumple la instrucción, tomamos un camino; en caso de que no se cumpla la instrucción, tomamos otro camino.

Esta sería la lógica:

```
SI (CONDICION) {

    EJECUTAMOS ESTE CODIGO
```

```
} SI NO SE CUMPLE LA CONDICION {
        EJECUTAMOS ESTE CODIGO
}
```

Ahora lo vamos a hacer con algo que sea más fácil de entender:

```
SI( tienes 18 años){
        Puedes pedir una cerveza
}En caso de que no los tengas {
        NO puedes pedir una cerveza
}
```

Como podemos ver, en la estructura IF siempre tenemos 2 caminos:

- El camino en caso de cumplir la condición.
- El camino en caso de no cumplir la condición.

En código Javascript sería de la siguiente forma:

```
var edad=20;


if(edad>18){
        document.write("Puedes pedir");
}else{
        document.write("No puedes pedir");
}
```

El resultado es el siguiente:

Vamos a cambiar entonces el valor de edad para probar que toma el camino contrario; el código es el siguiente:

```
var edad=17;


if(edad>18){
        document.write("Puedes pedir");
}else{
        document.write("No puedes pedir");
}
```

El resultado es el siguiente:

Como vemos, dependiendo de la evaluación se toma el camino 1 o el camino 2. Algo que en un principio no se tiene mucho en cuenta es un caso como en el siguiente código:

```
var edad=18;



if(edad>18){

      document.write("Puedes pedir");

}else{

      document.write("No puedes pedir");

}
```

Podemos ver que la variable edad vale 18, en términos matemáticos lo podemos escribir así: "edad=18" pero nosotros estamos preguntando si edad es mayor que 18, en ese caso se ejecutará el segundo camino como podemos ver en la siguiente imagen:

Lo que sucede es que la condición no se cumple, edad no es mayor que 18, por lo tanto se ejecuta el camino dos. Si queremos incluir el valor 18 como uno permitido, necesitamos entonces cambiar el operador por el siguiente:

```
var edad=18;



if(edad>=18){
        document.write("Puedes pedir");
}else{
        document.write("No puedes pedir");
}
```

Por eso, es importante tener claro el uso de los operadores para realizar una condición.

Estructura IF simple

En algunas ocasiones solo queremos tomar un camino y no es necesario un segundo, para estos casos utilizamos un IF simple, es decir, en el código vamos a ver un IF sin ELSE; veamos un ejemplo:

```
var edad=18;



if(edad>=18){

        document.write("Puedes pedir");

}
```

Como podemos ver en el código anterior no contamos con la instrucción *else*, el resultado es el siguiente:

Estructura de control FOR

Cuando tenemos una instrucción y queremos que esta se ejecute más de una vez, probablemente nuestra primera opción sea la estructura de control *for*. Esta nos sirve para ejecutar un número determinado de veces "algo", el resultado lógico sería el siguiente:

```
for(VALOR INICIAL, CONDICION, INCREMENTO){

INSTRUCCIONES QUE EJECUTAREMOS

}
```

El valor inicial: es el punto de partida de las repeticiones, podemos comenzar desde 0 como se hace habitualmente, pero si necesitamos iniciar desde el número 3, 15, 10000 o desde cualquier otro número lo podemos hacer.

Condición: es la condición que vamos a verificar cada vez que terminemos de ejecutar lo que se encuentra dentro del *for*, en caso de cumplir la condición *for* simplemente continuamos con las siguientes líneas y esto implica que en el momento en que ya no cumplamos la condición, simplemente saltamos a las siguientes líneas porque ya hemos terminado las repeticiones.

Incremento: si el valor inicial siempre fuera igual, significa que entonces nunca acabaríamos de ejecutar el ciclo, entonces siempre que se ejecuta el *for* sucede un incremento y eso significa que estamos más cerca de terminar, este sería el código de la instrucción *for*:

```
for(var i = 0; i <10;i++){

        alert(i);

}
```

Ahora vamos a ver el resultado de la ejecución:

La instrucción:

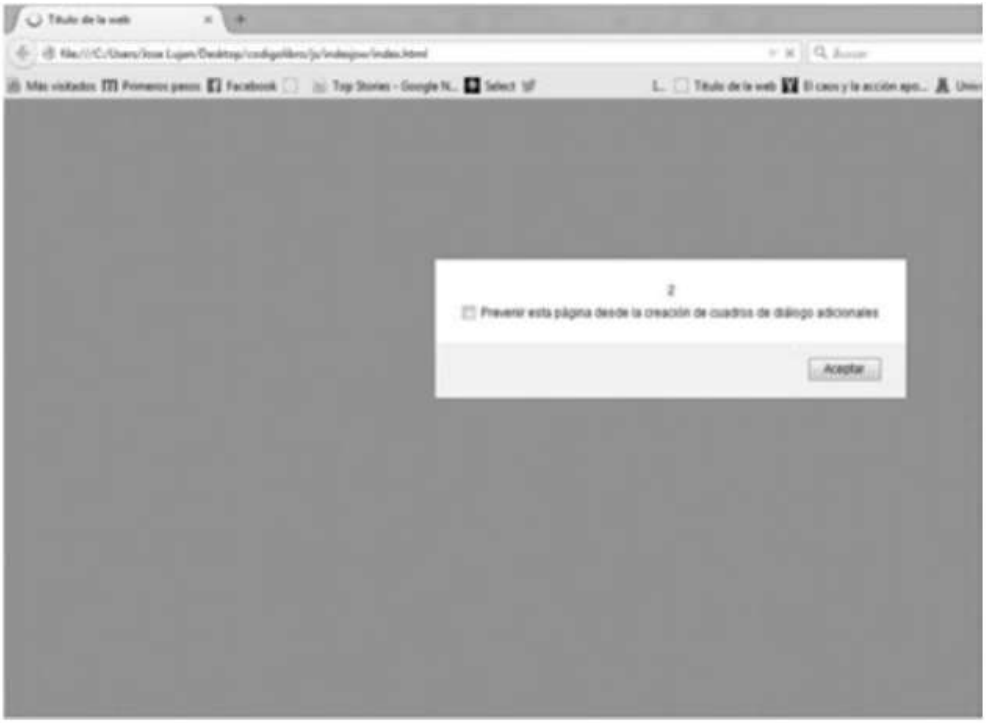
```
alert(i);
```

Lo que hace es imprimir un pequeño campo de alerta con un texto que nosotros le indiquemos, en nuestro caso no es un texto, es una variable… la variable *i*.

La primera vez que ejecutamos el código, la variable (i) tiene el valor de 0. Cuando hacemos clic en el botón *Aceptar* sucede lo siguiente:

Como podemos ver, se repite la creación del cuadro de texto pero ahora con un valor diferente, el número es 1, esto es debido a que ya se ejecutó la instrucción *for*, entonces realiza un incremento sobre la variable.

Si hacemos clic, veremos el siguiente resultado:



Como vemos, siempre va a ir incrementando, hasta que no cumplamos la condición que en nuestro caso es:

```
i <10;
```

Mientras *i* sea menor que 10, seguiremos poniendo *for* y ejecutando la instrucción.

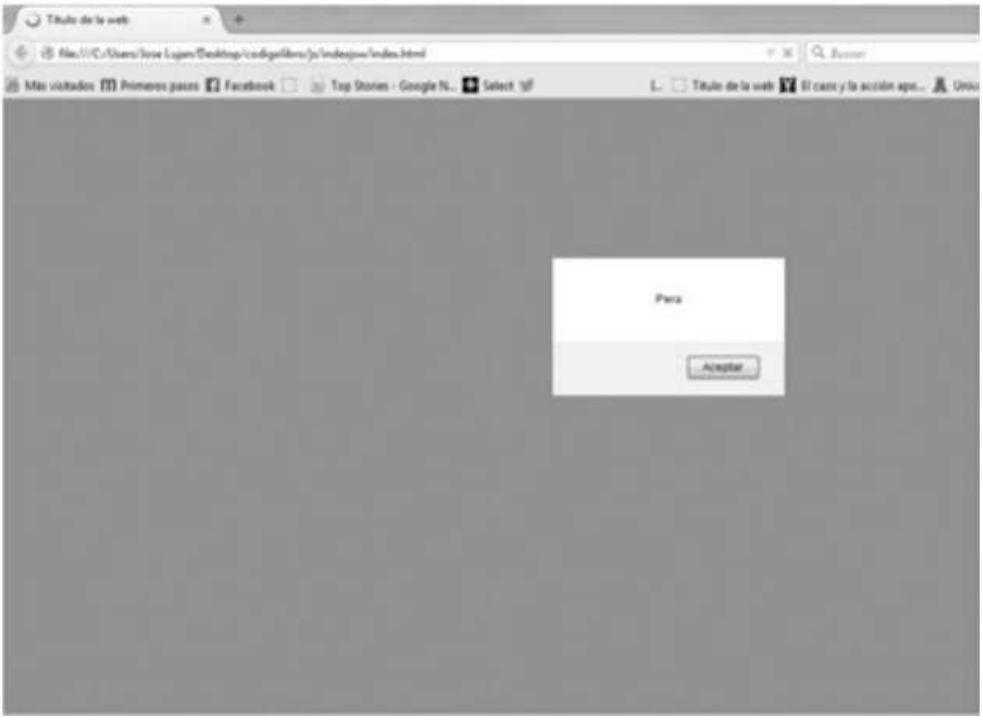
Estructura for/in

La estructura *for/in* es una forma más sencilla de recorrer elementos, además de que nos sirve para recorrer, por ejemplo, los arreglos, que son un tipo de dato más complejo.

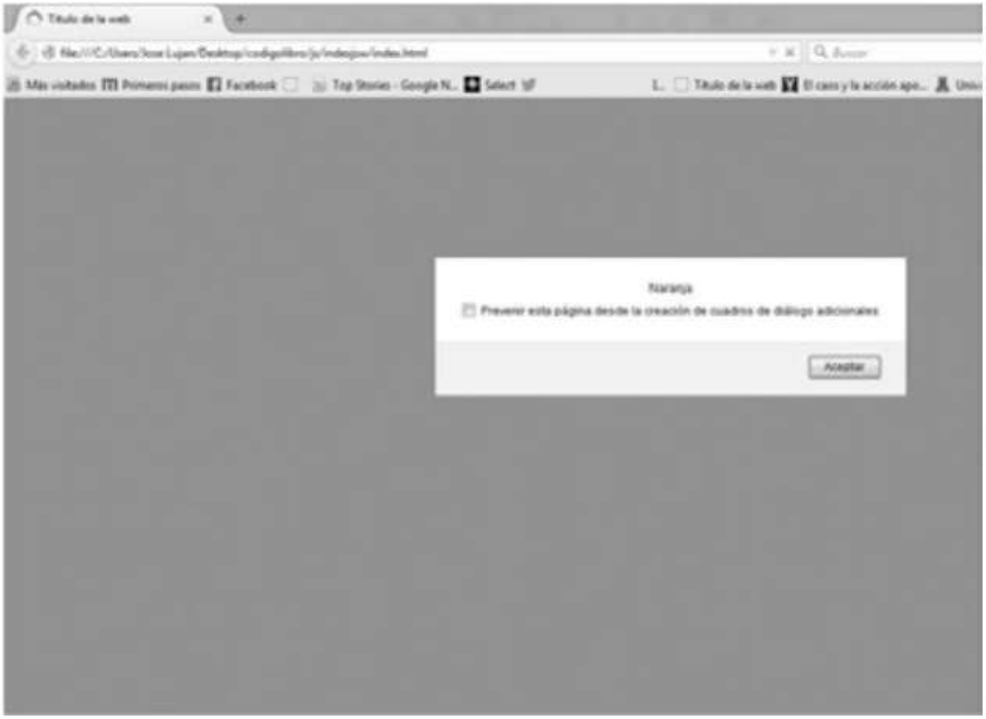
Este es nuestro código:

```
var frutas = ["Pera", "Naranja", "Sandía", "Melon", "Guayaba", "Limón"];



for(i in frutas) {

  alert(frutas[i]);

}
```

El resultado de ejecutar esto la primera vez es el siguiente:



Si hacemos clic en el botón *Aceptar* y observamos la pantalla, veremos lo siguiente:



Estructura de control WHILE

La estructura de control *while* es muy parecida a la de *for*, nos sirve para repetir la ejecución de código, con la diferencia que con *while* no necesitamos saber el número determinado de veces, solamente debemos saber cuándo queremos que se deje de repetir la ejecución.

La lógica es la siguiente:

```
While(CONDICION){

Código a ejecutar

INCREMENTO

}
```

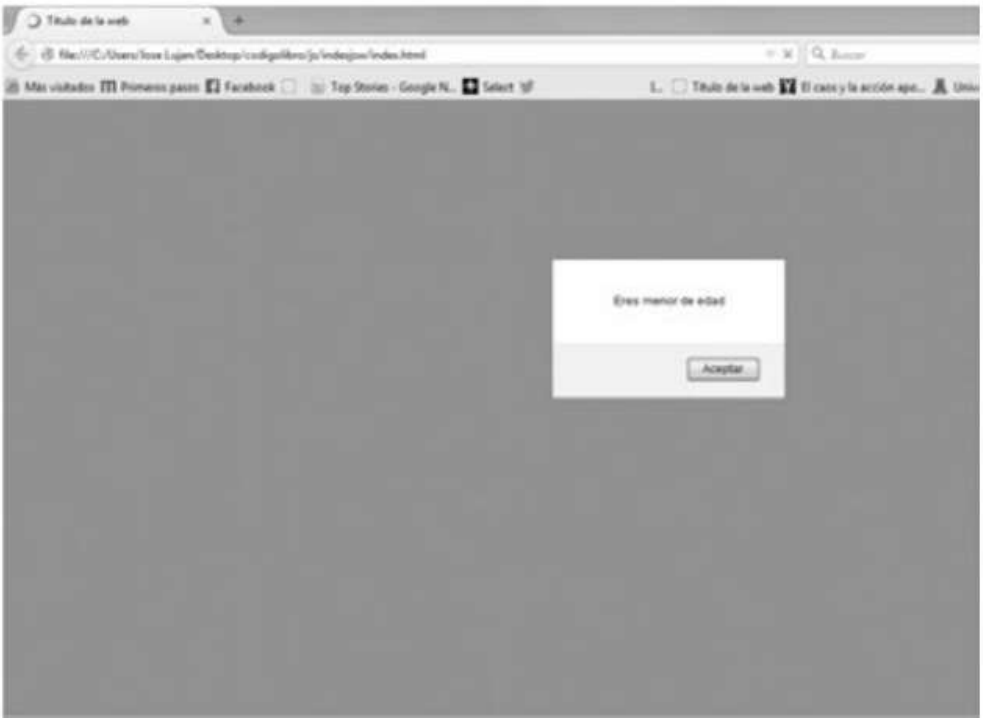
Condición: es lo que vamos evaluar, en caso de cumplir la condición entonces ejecutamos el código, en caso de que no la cumplamos, se pasa de largo.

Incremento: a diferencia del *for* el incremento no se hace en la misma parte en donde se declara la condición, el incremento se hace antes de cerrar la llave, que es parte del *while*.

Este es un ejemplo con la variable edad, mientras sea menor de edad vamos a imprimir el mensaje: “Eres menor de edad”:

```
edad=0;

while(edad<18){

alert("Eres menor de edad");

edad++;

}
```

El resultado es el siguiente:



La imagen anterior se va a repetir hasta que la variable edad deje de cumplir la condición:

```
while(edad<18){
```

Estructura de control *DO WHILE*

La estructura de control *DO* es muy similar al *WHILE*, pero tiene una diferencia primordial, en el *WHILE* siempre necesitamos cumplir la condición aunque sea 1 vez, si no la cumplimos nunca entramos al ciclo WHILE.

En el caso del *DO* siempre entramos al ciclo una primera vez, aunque no se cumpla la condición se entra al ciclo y se ejecuta una vez el código, la lógica es la siguiente:

```
Do{
Código a ejecutar
INCREMENTO
}while(CONDICION)
```

Incremento: aquí colocamos la instrucción de incremento que vamos a realizar sobre la variable que verificaremos en la condición.

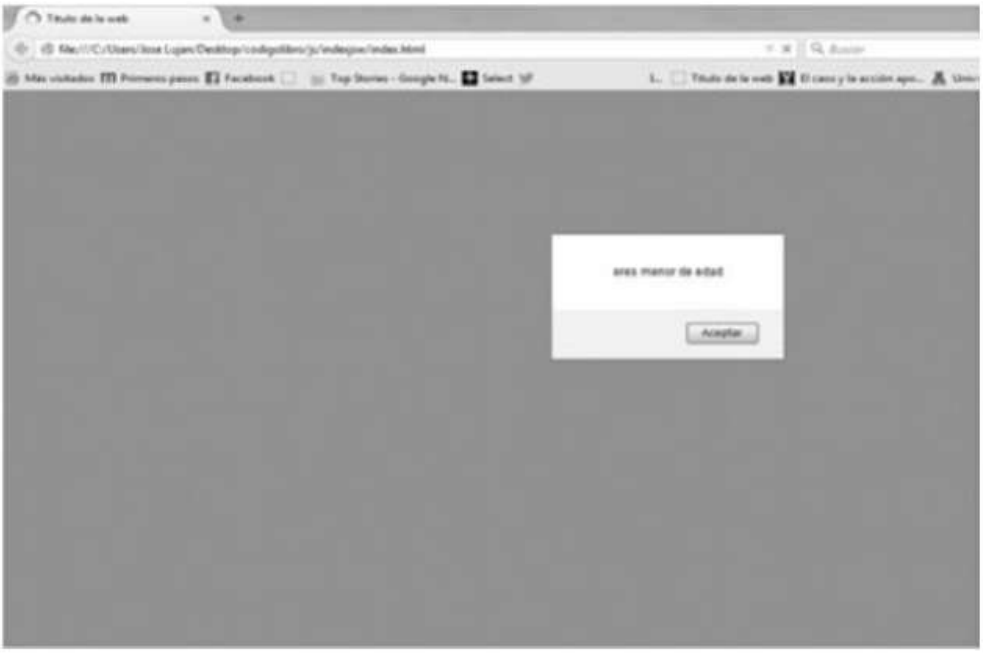
Condición: colocamos los operadores para realizar la validación de la condición.

El código en un caso real quedaría de la siguiente forma:

```
edad=20;


do{
alert("eres menor de edad");
```

```
edad++

}while(edad<18)
```

El resultado es el siguiente:



Como podemos ver, la edad tiene el valor de 20:

```
edad=20;
```

Aunque la condición dice:

```
}while(edad<18)
```

Podemos ver que sí entra al ciclo y además ejecutamos el código, esto es debido a que el ciclo *DO* siempre entra la primera vez sin verificar la condición, así que aunque no se cumpla, entra al ciclo y ejecuta el código, después de esta primera vez ya solo se va ejecutar el código cuando se cumpla la condición.

Funciones

Las funciones en la programación son una forma de agrupar ciertas instrucciones de código y así poder llamarlas en bloque. Lo importante de una función es colocarle un nombre para poder identificarla dentro del código.

Ahora vamos a declarar una función que se llama saludo:

```
function saludo(){

alert("Esta es una funcion");

}

saludo();
```

Para declarar una función colocamos la palabra reservada *function*, después colocamos el nombre de la función que en este caso es saludo, abrimos y cerramos paréntesis y colocamos unas llaves:

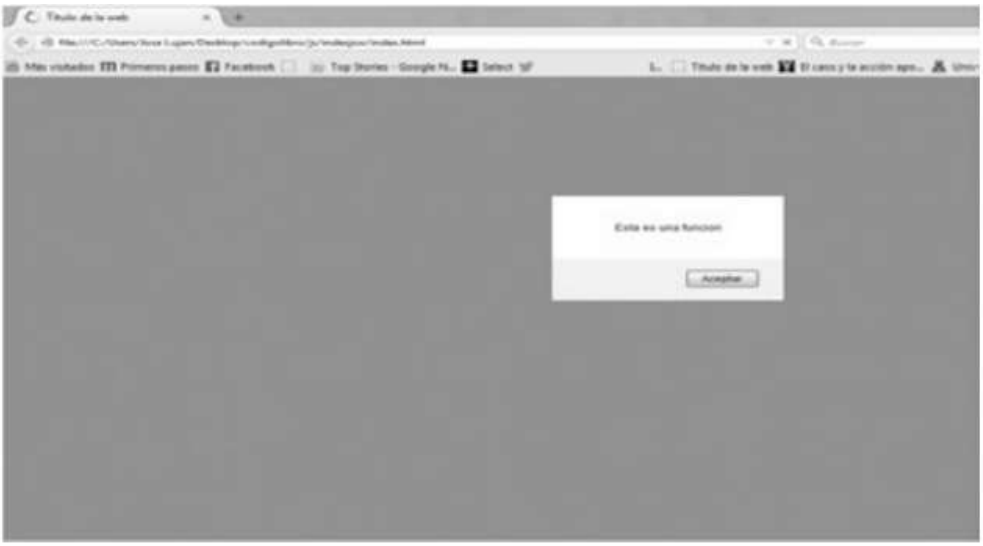
```
Function NOMBREFUNCION(){

Código que vamos a ejecutar;

}
```

En nuestro ejemplo le colocamos un simple alert que lo que hará es desplegar un cuadro con un saludo.

Para poder ejecutar una instrucción la llamamos por su nombre en cualquier parte del código que la necesitemos:

```
saludo();
```

Este es el resultado de la ejecución:



Funciones con parámetros

Cuando trabajamos con funciones tenemos la posibilidad de pasarle parámetros y utilizar estos para realizar operaciones, a este tipo de funciones se les conoce por el nombre de funciones con parámetros. Los parámetros se colocan entre los paréntesis.

```
function saludo(nombre){

var nombreR = nombre;

alert("Hola "+nombreR);

}



saludo("José");
```

Vamos a analizar el código, primero vamos a declarar una función que se llama saludo y le vamos a pasar un parámetro al que llamaremos nombre:

```
function saludo(nombre){
```

Ahora declaramos una variable nombreR que va almacenar la variable nombre que recibimos por parámetro:

```
var nombreR = nombre;
```

Finalmente dentro de la función mandamos a imprimir un mensaje al que le estamos concatenando la variable “nombreR” que llenamos con el parámetro que recibimos.

```
alert("Hola "+nombreR);
```

Para mandar a llamar a la función y que esta se ejecute, colocamos su nombre y le pasamos el parámetro que enviaremos.

```
saludo("José");
```

Este es el resultado:

Funciones que devuelven valores

Las funciones, como ya vimos anteriormente, tienen la posibilidad de trabajar por sí solas, también pueden trabajar recibiendo parámetros, tendremos una tercera

opción en la que podemos devolver valores. En la programación en general la forma técnica de definir una función es la siguiente: una porción de código que por defecto debe devolver un valor. Esto quiere decir que para ser considerada de forma general una función, la porción de código debe devolver un valor.

Para devolver un valor utilizamos la palabra reservada *return*, con esta palabra le especificamos cuál es el dato que vamos a devolver, el código sería el siguiente:

```
function saludo(nombre){

var cadena = "hola"+nombre;

return cadena;

}



alert(saludo("José"));
```

Primero declaramos una función que se llama saludo, a la cual le pasamos el parámetro “nombre”.

```
function saludo(nombre){
```

Ahora declaramos una variable cadena, a la cual le colocamos una cadena de texto “hola” y le concatenamos la variable nombre que recibimos.

```
var cadena = "hola"+nombre;
```

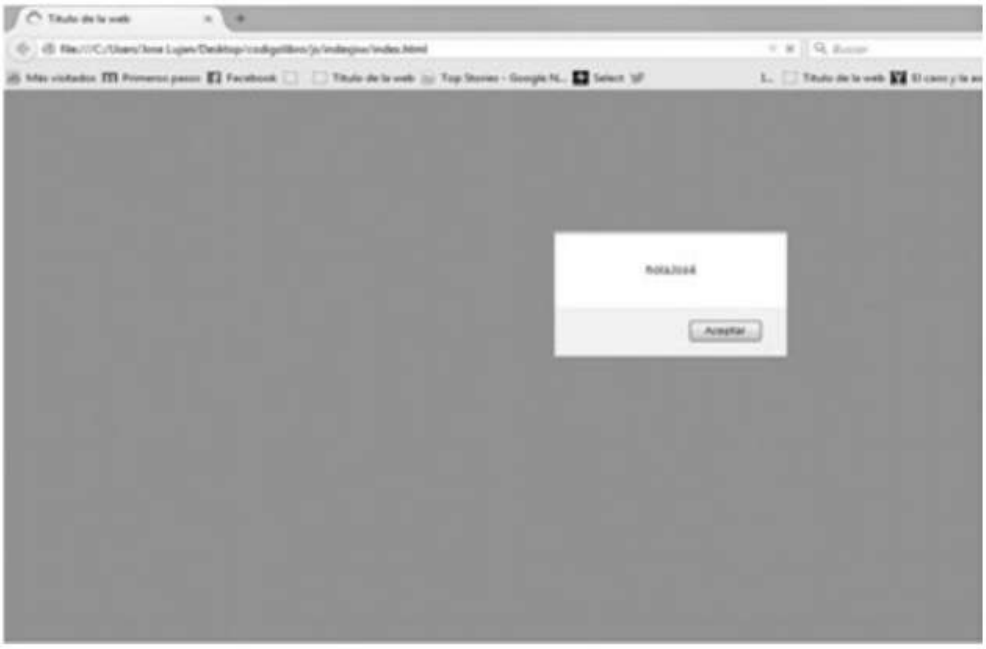
Finalmente utilizamos la palabra *return* para devolver un dato, el dato que estamos asignando es la variable cadena.

```
return cadena;
```

Para ejecutar la función lo que hacemos es llamarla por su nombre, pero en este caso la colocamos dentro de un alert para poder ver el resultado de la ejecución de la función.

```
alert(saludo("José"));
```

El resultado se ve de la siguiente forma:



Uno de los motivos principales por los que devolvemos una variable al ejecutar una función es porque necesitamos utilizar ese valor en el programa principal; para estos casos hacemos lo siguiente:

```
function saludo(nombre){

var cadena = "hola"+nombre;

return cadena;

}
```

```
var dato= saludo("Juan");




alert(dato);
```

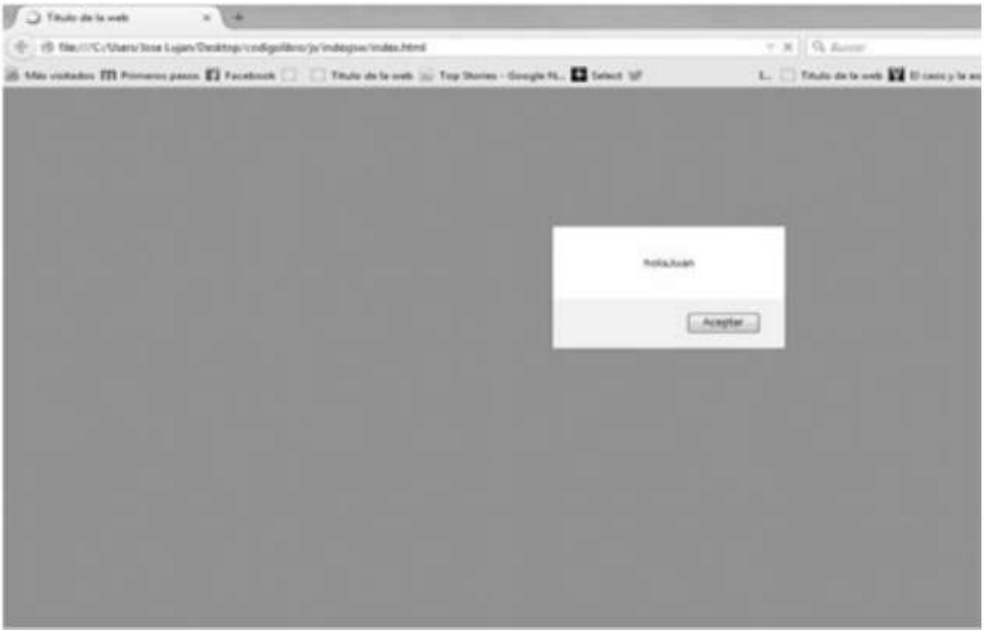
Como podemos ver, si comparamos los dos códigos que hemos realizado, la diferencia es la siguiente:

```
var dato= saludo("Juan");
```

Estamos llamando a la función “saludo” pasándole el parámetro “juan” que es el parámetro nombre, además estamos almacenando lo que devuelve esa función en una variable que llamamos saludo.

Finalmente ya tenemos la variable *dato* para acceder de primera mano a lo que arrojó la función saludo.

El resultado se ve de la siguiente forma:



Objetos en JavaScript

Ya conocemos las variables y las funciones que nos proporciona Javascript, además de estos conceptos Javascript nos ofrece el concepto de Objeto, este concepto es la base del paradigma de programación orientada a objetos. La teoría nos dice que los objetos pueden ser cualquier cosa, una televisión, un teléfono, un coche, todo lo que tenemos en nuestra vida es un objeto. Como esta definición puede generar un poco de confusión, voy a compartir una definición menos teórica pero con más imaginación para ayudar al lector a entender qué es un objeto.

Un objeto lo podemos ver como una variable, hasta ahora sabemos:

```
Las variables pueden ser de diferentes tipos: enteros, flotantes, etc.
```

El objeto es algo más grande, algo más complejo que una variable, pero por ahora lo vemos como una variable ya que en el momento de declarar un objeto, también le estamos diciendo a la computadora cuánto va ser el espacio que necesitamos reservar de memoria si queremos utilizar un objeto.

La diferencia más grande es que un objeto es más complejo, las variables tienen tipos, los objetos tienen que ser definidos y cuentan con métodos, atributos y otras propiedades.

Veamos cómo se define un objeto sencillo en Javascript:

```
var objeto = {
        atributo1:0,
        atributo2:"cadena",
        atributo3:true
}
```

Para declarar un objeto también utilizamos la palabra reservada “var” y después colocamos el nombre seguido del signo igual y abrimos llaves:

```
var nombreobjeto = {
```

Después podemos colocar cualquier cantidad de atributos, en este caso el atributo son las propiedades y las características del objeto, el atributo1 es del tipo entero:

```
    atributo1:0,
```

El atributo2 es del tipo string o cadena:

```
    atributo2:"cadena",
```

El atributo3 es del tipo booleano, los atributos van separados con el carácter coma “,” y el último atributo no lleva este carácter:

```
    atributo3=true
```

Cuando acabamos de definir el objeto cerramos llaves:

```
}
```

Para poder acceder a cada atributo, en Javascript tenemos 2 opciones.

```
var objeto = {

    atributo1:0,

    atributo2:"cadena",

    atributo3:true

}



alert(objeto.atributo1);
```

La primera opción es acceder colocando el nombre del objeto, después el carácter de punto seguido del nombre del atributo:

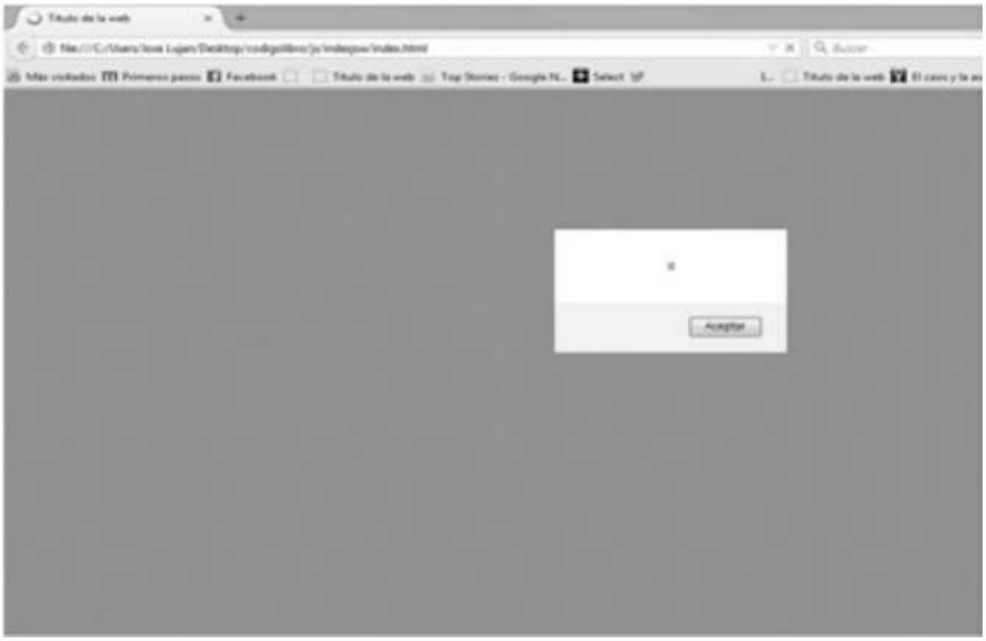
```
alert(objeto.atributo1);
```

La otra opción es colocar el nombre del objeto y entre corchetes y comillas el nombre del atributo:

```
alert(objeto["atributo1"]);
```

Las dos formas son válidas.

El resultado de ejecutar el código es el siguiente:



Cuando queremos acceder a un atributo del objeto y cambiar su valor, tenemos que hacer lo siguiente:

```
var objeto = {
    atributo1:0,
    atributo2:"cadena",
```

```
        atributo3:true
    }


    objeto.atributo1=10;
    alert(objeto.atributo1);
```

Accedemos al atributo colocando el nombre del objeto y después indicamos el atributo. Para asignar el valor utilizamos el operador de asignación:

```
objeto.atributo1=10;
```

Este es el resultado de ejecutar el código:

Métodos

Los métodos son las acciones que puede realizar un objeto, por ejemplo si tuviéramos un objeto persona, los métodos podrían ser: nadar, comer, correr, respirar, etc. Cualquier acción que puede realizar un objeto se define en un objeto.

Esta sería la forma de declarar un método:

```
var objeto = {
        atributo1:0,
        atributo2:"cadena",
        atributo3:true,

        cambiar: function(){
                objeto.atributo1=100;
        }
}

objeto.cambiar();
alert(objeto.atributo1);
```

Los métodos se definen como las variables, se coloca el nombre, el carácter dos puntos (:) y seguido de la palabra reservada *function*, dentro del método colocamos todo el código que deseamos ejecutar, en nuestro método estamos accediendo al atributo1 a cambiar el valor que tiene de 0 por el valor de 100:

```
cambiar: function(){
                objeto.atributo1=100;
        }
```

Para ejecutar el método tenemos que llamarlo, primero colocamos el nombre del objeto seguido del carácter punto (“.”) y a continuación el nombre del método:

```
objeto.cambiar();
```

Constructor

Cuando vamos a crear un objeto lo primero que se ejecuta es el constructor, cuando creamos un objeto utilizamos la palabra reservada “new”, en ese momento se ejecuta el constructor.

Finalmente vamos a ver la sintaxis final de cómo se declara un objeto y cómo se crea un objeto:

```
function objeto(nombre){
        this.atributo1=0;
        this.atributo2="cadena";
        this.atributo3=true;

        this.cambiar = function(nombre){
                this.atributo1=100;
                this.atributo2=nombre;
        }

}

var objetoX = new objeto();

objetoX.cambiar("José");
```

```
alert(objetoX.atributo2);
```

Estrictamente la creación de un objeto se hace con una función, en nuestro ejemplo le colocamos el nombre “objeto”:

```
function objeto(nombre){
```

Para acceder a los atributos de un objeto utilizamos la palabra reservada “this”. Esta le está indicando que estamos accediendo a nuestros propios atributos, ya que en relación con esta clase vamos a crear distintos objetos y cada objeto tiene sus propios atributos:

```
this.atributo1=0;

this.atributo2="cadena";

this.atributo3=true;
```

Para declarar un método colocamos la palabra reservada “this” seguida del carácter punto “.”, luego colocamos el nombre del método, utilizamos después la palabra reservada *function* y declaramos el contenido del método, si necesitamos recibir parámetros los colocamos entre los paréntesis y los utilizamos como en cualquier función.

```
this.cambiar = function(nombre){

            this.atributo1=100;

            this.atributo2=nombre;

    }
```

Para crear un objeto utilizamos la palabra reservada “new” seguida de lo que ya definimos previamente:

```
var objetoX = new objeto();
```

Si queremos definir más objetos lo que tenemos que hacer es únicamente asignarles un nombre diferente, por ejemplo:

```
var objetoY = new objeto();

var objetoZ = new objeto();
```

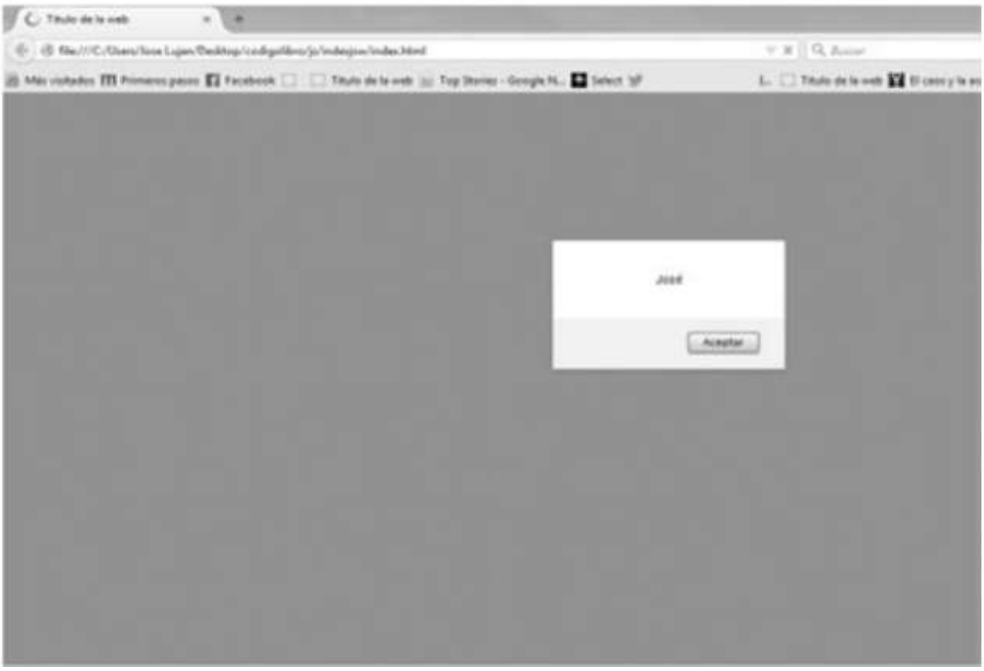
Si queremos ejecutar un método del objeto, colocamos el nombre del objeto seguido del carácter punto "." Y el nombre del método con sus respectivos parámetros si son necesarios:

```
objetoX.cambiar("José");
```

Para acceder a los atributos del método, solo colocamos el nombre del método seguido del carácter punto (".") y el nombre del atributo:

```
alert(objetoX.atributo2);
```

Este es el resultado de lo que hicimos:



Relación JavaScript-HTML

Javascript es el lenguaje de programación que nos permite comunicarnos de forma directa con el navegador, esto hace que la relación de Javascript y HTML sea muy estrecha, ya que los dos son interpretados por el navegador y conviven en el mismo sistema.

La relación se vuelve mucho más estrecha cuando descubrimos y entendemos que Javascript nos permite modificar e interactuar con casi cualquier elemento de un navegador.

Objeto window

Cuando hacemos clic en el navegador para abrirlo se crea un objeto *window*, este objeto es el navegador que estamos utilizando, como cualquier objeto contamos con atributos y métodos que podemos modificar.

El objeto *window* es al que accedemos aunque en muchos casos lo vamos a tener que hacer desde la propiedad *window*, que es la forma en que accederemos al objeto.

El método más famoso del objeto *window* es *alert(),* hasta este apartado del libro siempre que queremos utilizar el método *alert* hacemos lo siguiente:

```
alert("colcamos texto");
```

La forma estricta de escribirlo sería:

```
window.alert("colocamos texto");
```

En ambos casos el resultado es el siguiente:

Propiedades y métodos de window

Vamos a describir las propiedades y los métodos de *window*.

Propiedad/Método	Qué hace
alert()	Despliega una caja de texto con la información que le colocamos
confirm	Despliega una caja de texto con 2 opciones: SÍ y NO
history	Lo utilizamos para poder navegar por el historial de la ventana
location	Podemos acceder a la información de la url
setInterval	Nos permite llamar funciones por una frecuencia que definimos
setTimeout	Ejecuta una función que indiquemos al transcurrir los milisegundos que le especifiquemos

Objeto Document

Cada vez que abrimos un sitio web sucede algo muy importante para un desarrollador web: se crea el DOM. DOM es una estructura que crea nodos, es decir, que crea un árbol y crea una relación entre los nodos, tenemos nodos padres y nodos hijos.

Por ejemplo:

```
<!DOCTYPE html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>Titulo de la web</title>
<link rel="stylesheet" href="css/estilo.css">



</head>
<body >
<p>Lo que sea</p>

 <script src="./js/script.js"></script>
 </body>

</html>
```

Este código HTML crea los siguientes nodos:

- Document es el nodo raíz.
- Document tiene 2 nodos, nodo head y nodo body.
- El nodo head tiene dentro dos nodos: el nodo meta y el noto title.
- El elemento title tiene el texto = Título de la web.
- El nodo body tiene dentro el nodo de <p>.
- El elemento p contiene texto = “Lo que sea”.

Como podemos ver se va creando una estructura de nodos y esta nos permite poder acceder y manipular todos los elementos si sabemos las relaciones que existen o la posición que ocupa.

DOM se inicia con el objeto Document que mencionamos anteriormente y accedemos a él desde la propiedad “document”, por ejemplo podemos utilizar:

```
document.write("Texto");
```

El resultado es el siguiente:

Acceder a nodos

Vamos a conocer los métodos que nos proporciona el DOM para poder acceder a los nodos, el objetivo de acceder a los nodos puede ser la modificación de los mismos o la lectura de información contenida.

getElementsByName()

Esta función nos permite buscar elementos por el atributo “name” y el nombre que tenga asignado.

Este es el código HTML:

```
<!DOCTYPE html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>Titulo de la web</title>
<link rel="stylesheet" href="css/estilo.css">



</head>
<body >
    <p name="nombre">José Luján </p>
    <p name="edad">28</p>
```

```
<script src="./js/script.js"></script>

</body>



</html>
```

Para seleccionar el nodo que tiene lo siguiente:

```
<p name="nombre">José Luján </p>
```

En Javascript tendríamos que hacer lo siguiente:

```
var texto = document.getElementsByName("nombre");
```

getElementById()

Otra forma que tenemos para seleccionar nodos es utilizando el *id* que se le puede asignar. Esta función podemos decir que accede de forma directa al nodo.

El código HTML es este:

```
<!DOCTYPE html>

<html lang="es">

<head>

<meta charset="utf-8">

<title>Titulo de la web</title>

<link rel="stylesheet" href="css/estilo.css">
```

```
</head>
<body >
    <p id="nombre">José Luján </p>
    <p name="edad">28</p>

 <script src="./js/script.js"></script>
 </body>

</html>
```

El código en Javascript para acceder al *id* es el siguiente:

```
var texto = document.getElementById("nombre");
```

getElementsByTagName()

Esta función nos permite obtener todos los elementos de la página que sean de la etiqueta que indicamos.

```
var parrafos = document.getElementsByTagName(“p”);
```

La instrucción anterior obtiene todas las etiquetas “p” que tenemos en nuestro documento. Como pueden ser 10,000 elementos, puede ser ninguno, así que el valor que nos llega es un arreglo.

Para acceder a un elemento del arreglo hacemos lo siguiente.

```
var primerap = párrafos[0];
```

Crear nodos

En algunas ocasiones nos puede interesar crear los nodos y debemos de entender que esto varía dependiendo del tipo de nodo que vamos a crear. El manejo del DOM se puede considerar como algo avanzado en la programación de Javascript. Ya que el tema es muy amplio en este apartado solo nos centraremos en crear un nodo del tipo *Element* como lo es la etiqueta <p>.

Lo que vamos a hacer es lo siguiente:

- Crear el nodo p.
- Crear el nodo texto.
- Añadir el texto al nodo p.
- Añadir el nodo p al documento.

Este es el código HTML:

```
<!DOCTYPE html>

<html lang="es">

<head>

<meta charset="utf-8">

<title>Titulo de la web</title>

<link rel="stylesheet" href="css/estilo.css">




</head>

<body >
```

```
<script src="./js/script.js"></script>
</body>

</html>
```

En el código anterior podemos ver que no tenemos ningún párrafo en el documento, el código del archivo Javascript es el siguiente:

```
var parrafo = document.createElement("p");
var texto = document.createTextNode("José Luján");
parrafo.appendChild(texto);
document.body.appendChild(parrafo);
```

Primero creamos un párrafo y le indicamos que es un elemento:

```
var parrafo = document.createElement("p");
```

Continuamos creando el texto, para eso usamos un nodo del tipo texto y colocamos texto dentro de él:

```
var texto = document.createTextNode("José Luján");
```

Al nodo párrafo le agregamos el nodo texto:

```
parrafo.appendChild(texto);
```

Al documento principal le agregamos un nodo hijo:

```
document.body.appendChild(parrafo);
```

El resultado de la ejecución de este código es el siguiente:

Eliminar nodos

Para eliminar nodos solamente tenemos que utilizar la función *removeChild().*

Tenemos el siguiente código HTML:

```
<!DOCTYPE html>

<html lang="es">

<head>

<meta charset="utf-8">

<title>Titulo de la web</title>

<link rel="stylesheet" href="css/estilo.css">
```

```
</head>
<body >

      <p id="texto">José Luján</p>
 <script src="./js/script.js"></script>
 </body>

</html>
```

Ahora vamos a eliminar desde Javascript el nodo del párrafo:

```
var parrafo = document.getElementById("texto");
parrafo.parentNode.removeChild(parrafo);
```

Una regla para eliminar un nodo es hacerlo desde el nodo padre, por eso utilizamos *parentNode.*

Acceder a los atributos desde Javascript

Al entender que todo documento HTML es un árbol de nodos, ya sabemos que podemos manipularlos, crearlos o destruirlos. También podemos acceder a los atributos de los nodos y modificarlos.

En este ejemplo vamos a acceder al margen de un elemento.

Tenemos el código HTML de nuestro archivo *index.html*:

```
<!DOCTYPE html>
<html lang="es">
```

```
<head>
<meta charset="utf-8">
<title>Titulo de la web</title>
<link rel="stylesheet" href="css/estilo.css">



</head>
<body >

      <p id="texto">José Luján</p>
 <script src="./js/script.js"></script>
 </body>

</html>
```

Este es el archivo estilo.css:

```
*{
padding:0px;
margin:0px;
}
```

Podemos ver que todos los elementos tienen un *padding* y *margin* en 0px.

Ahora vamos a analizar nuestro archivo Javascript:

```
document.getElementById("texto").style.margin = "10px";
```

En el código anterior lo primero que hacemos es acceder al elemento de forma directa seleccionándolo por ID, después entramos a su estilo y específicamente a la propiedad de margen, cambiando el valor que teníamos en un principio en 0px a 10px.

El resultado se ve de la siguiente forma:

En la imagen anterior podemos ver que con la herramienta Firebug podemos corroborar el cambio del margen desde Javascript.

Eventos

Los eventos son las acciones que pueden ocurrir en nuestro sitio web, por ejemplo: hacer clic, cargar una página, entre otras. Tenemos varios eventos en Javascript que podemos utilizar.

Evento	Descripción del evento	Elementos para los que sirve
onblur	Deseleccionar elemento	Button, inputs, label, select, textarea, body
onchange	Deselecciona un elemento que cambio	Input, select, textarea
onclick	Al hacer clic con el ratón	Todos
ondbclick	Al hacer doble clic con el ratón	Todos
onfocus	Seleccionar elemento	Buton, input, label, select, textarea, body
onkeydown	presionar una tecla	Formulario y body
onkeypress	Presionar una tecla y soltar	Formulario y body
onkeyup	Soltar una tecla presionada	Formulario y body
onload	Cargar una página	Body
onmousedown	Presionar el botón del ratón	Todos
onmousemove	Mover el ratón	Todos
onmouseout	Sacar el ratón de un elemento	Todos

onmouseover	Colocar el ratón sobre un elemento	Todos
onmouseup	Soltar el botón del mouse (ratón) que estaba presionado	Todos
onreset	Reiniciar formulario	Form
onresize	Modificar el tamaño de la ventana	Body
onselect	Seleccionar	Input, textarea
onsubmit	Enviar	Form
onunload	Salir de página	body

Solo para hacer una prueba con algunos de estos eventos podemos colocar este código:

```
<!DOCTYPE html>

<html lang="es">

<head>

<meta charset="utf-8">

<title>Titulo de la web</title>

<link rel="stylesheet" href="css/estilo.css">




</head>

<body >
```

```
        <p id="texto" onClick=alert("CLIC");>José Luján</p>
    <script src="./js/script.js"></script>
    </body>


</html>
```

Podemos ver que en el párrafo habilitamos el evento *onclick,* esto quiere decir que cuando se haga clic en el párrafo realizará una "acción" y le colocamos la acción de mostrar una ventana *alert* con el mensaje "CLIC", así se ve el resultado:

addEventListener()

En el apartado anterior utilizamos el evento *onClick* pero en la actualidad esa forma de implementarlo no es la correcta, en HTML5 se utiliza *addEventListener* para implementar métodos correctamente.

¿Por qué se utiliza *addEventListener*?

Como vimos en el apartado anterior, el código se comienza a ensuciar ya que estamos colocando en el mismo archivo HTML y JAVASCRIPT, además tenemos que colocar las instrucciones en el mismo apartado y esto ensucia aún más el código, siguiendo la recomendación debemos utilizar *addEventListener* para poder enlazar el evento y el código que se va ejecutar.

La sintaxis del *addEventListener* es la siguiente:

addEventListener(evento,función,captura)

El primer atributo es el nombre del evento, el segundo atributo es la función que vamos a ejecutar cuando suceda el evento, el tercer atributo es un booleano, es decir, que es falso o verdadero, por ahora nos quedaremos con que lo usaremos colocando el valor de falso. El tercer atributo es para indicar el tipo de propagación del evento si es de orden superior o inferior, para nuestros ejemplos nos va servir en false.

En el siguiente ejemplo vamos a implementar de forma correcta un evento, este es el código HTML:

```
<!DOCTYPE html>

<html lang="es">

<head>

<meta charset="utf-8">

<title>Titulo de la web</title>

<link rel="stylesheet" href="css/estilo.css">
```

```
</head>
<body >

      <p id="nombre"> José Luján </p>
 <script src="./js/script.js"></script>
 </body>

</html>
```

En el archivo HTML solo tenemos una etiqueta <p> y esta tiene un identificador, este es el que vamos a utilizar en Javascript para enlazarlo con una función.

Este es el código de nuestro archivo de Javascript:

```
function mostrarsaludo(){
alert("Hola!");
}

window.onload = function(){

var texto = document.getElementById("nombre");
texto.addEventListener("click",mostrarsaludo,false);

}
```

Primero definimos la función “mostrarsaludo” que es una simple función que solo va a mostrar un mensaje que dice “Hola!”:

```
function mostrarsaludo(){

alert("Hola!");

}
```

El siguiente código es importante ya que con él aseguramos la correcta ejecución, utilizamos el evento *load*, este evento nos asegura que hasta que se realice la carga de todo, se ejecutará el código que coloquemos dentro.

```
window.onload = function(){
```

Dentro del *onload* colocamos lo que queremos que se ejecute en el momento que se realiza la carga, primero creamos un elemento texto que va a ser enlazado al elemento que tenga el id=”nombre”.

```
var texto = document.getElementById("nombre");
```

Ahora vamos a utilizar el *addEventListener* y le pasamos los parámetros correctos, le pasamos como primer parámetro cuál es el evento que vamos a esperar, en este caso es el evento de click, después colocamos el nombre de la función que vamos a ejecutar cuando suceda este evento, finalmente colocamos el parámetro en false.

```
texto.addEventListener("click",mostrarsaludo,false);
```

Ahora ya realizamos la implementación correcta de un evento en Javascript.

En el momento de hacer clic al elemento <p> sucederá lo siguiente:


Título de la web
Hola!
Aceptar

ÍNDICE ANALÍTICO

D

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

V

W