



UNIVERSIDAD NACIONAL
AVENIDA DE
MÉXICO

**UNIVERSIDAD NACIONAL AUTÓNOMA
DE MÉXICO**

**FACULTAD DE ESTUDIOS SUPERIORES
CUAUTITLÁN**

**IMPLEMENTACIÓN DE BOOTSTRAP EN EL
DESARROLLO WEB.**

T E S I S

QUE PARA OBTENER EL TÍTULO DE:

LICENCIADA EN INFORMÁTICA

P R E S E N T A :

ABRIL IRAIS ALBA VILLA

ASESOR:

L.I. ROSALBA NANCY ROSAS FONSECA

**CUAUTITLÁN IZCALLI, ESTADO DE MÉXICO
2019**



Universidad Nacional
Autónoma de México



UNAM – Dirección General de Bibliotecas
Tesis Digitales
Restricciones de uso

DERECHOS RESERVADOS ©
PROHIBIDA SU REPRODUCCIÓN TOTAL O PARCIAL

Todo el material contenido en esta tesis esta protegido por la Ley Federal del Derecho de Autor (LFDA) de los Estados Unidos Mexicanos (México).

El uso de imágenes, fragmentos de videos, y demás material que sea objeto de protección de los derechos de autor, será exclusivamente para fines educativos e informativos y deberá citar la fuente donde la obtuvo mencionando el autor o autores. Cualquier uso distinto como el lucro, reproducción, edición o modificación, será perseguido y sancionado por el respectivo titular de los Derechos de Autor.



UNIVERSIDAD NACIONAL
AUTÓNOMA DE
MÉXICO

FACULTAD DE ESTUDIOS SUPERIORES CUAUTITLÁN
SECRETARÍA GENERAL
DEPARTAMENTO DE EXÁMENES PROFESIONALES

U. N. A. M.
FACULTAD DE ESTUDIOS
ASUNTO: VOTO APROBATORIO



DEPARTAMENTO DE EXÁMENES PROFESIONALES
de la FES Cuautitlán.

M. en C. JORGE ALFREDO CUÉLLAR ORDAZ
DIRECTOR DE LA FES CUAUTITLÁN
PRESENTE

ATN: I.A. LAURA MARGARITA CORTAZAR FIGUEROA
Jefa del Departamento de Exámenes Profesionales

Con base en el Reglamento General de Exámenes, y la Dirección de la Facultad, nos permitimos comunicar a usted que revisamos el: **Trabajo Profesional**

Implementación de Bootstrap en el Desarrollo Web

Que presenta la pasante: ABRIL IRAIS ALBA VILLA

Con número de cuenta: 41406077-9 para obtener el Título de la carrera: Licenciatura en Informática

Considerando que dicho trabajo reúne los requisitos necesarios para ser discutido en el EXAMEN PROFESIONAL correspondiente, otorgamos nuestro VOTO APROBATORIO.

ATENTAMENTE

"POR MI RAZA HABLARÁ EL ESPÍRITU"

Cuautitlán Izcalli, Méx. a 21 de mayo de 2019.

PROFESORES QUE INTEGRAN EL JURADO

	NOMBRE	FIRMA
PRESIDENTE	M.I. Gerardo Vigil Sanabria	
VOCAL	L.I. Rosalba Nancy Rosas Fonseca	
SECRETARIO	L.I. Marco Alberto Silva Reyes	
1er. SUPLENTE	Lic. Julio César Ramón Maldonado Rodríguez	
2do. SUPLENTE	L.I. Luis Manuel Romero García	

NOTA: los sinodales suplentes están obligados a presentarse el día y hora del Examen Profesional (art. 127).

DEDICATORIAS

Gracias a Dios, a la vida y al tiempo, por darme la familia más perfecta del mundo, por darme los pilares más importantes en mi vida, porque sin ellos no me encontraría en donde estoy, mucho menos lograría lo que tengo hasta ahora, gracias por permitirme despertar cada mañana y tener la oportunidad de aprender más, de dar cada día un nuevo paso.

A mi papi José Luis Alba Garduño y mi mami Ma. de la luz Villa Álvarez, les quiero dar las gracias por darme la oportunidad de tener una vida tan bonita, de haberme educado y darme los valores pertinentes para convertirme en esta persona, les debo todo lo que soy, porqué las experiencias de vida que he tenido a su lado, han sido una gran lección de vida para mí, no se imaginan cuanto valoro el esfuerzo, dedicación y amor que me dan en cada uno de los pasos que doy, muchas gracias por impulsarme a ser cada día mejor, gracias por cobijarme y apoyarme en mis decisiones, lamento ser un dolor de cabeza de vez en cuando, pero saben que los amo con todo mi ser, este trabajo es para ustedes, para que conozcan mi lado artístico y profesional, yo sé que lo van a disfrutar, pues esto es logro de ustedes, me siento enormemente orgullosa de los papás que tengo. Los amo infinito más uno por siempre.

A mi hermanita Bel, te quiero dar las gracias por convertirte en mi ejemplo de vida, por ser mi mentora, mi camino y mi luz, siempre me he sentido orgullosa de la hermana que tengo, por lo valiente e inteligente, aguerrida y cariñosa, desde chiquita haz sido mi mayor ídolo, mi mejor confidente y mi ángel guardián, pues juntas hemos pasado muchas cosas, de las que ahora nos reímos, pero que en esos momentos me sentía protegida por ti, este trabajo también se inspira en ti, porque si te das cuenta, seguí bien tus pasos, pues me enseñaste bien, yo sé que lo sabes, pero es necesario recordarlo, te amo inmensamente y siempre estaré contigo, en las buenas y en las peores, siempre juntas de la mano. Te amo corazóna.

A mi amorcito Francisco Islas, tengo que reconocer que nunca paso por mi cabeza encontrarme con una persona tan maravillosa como tú, en muy poquito tiempo te convertiste en mi persona favorita, mi compañero de vida, llegaste en la etapa menos esperada y la convertiste en la más increíble de mi vida, llegaste para quedarte, y quiero que sepas que te agradezco, el apoyo, la confianza, los desvelos, las mal pasadas y los momentos divertidos e inesperados que compartes conmigo, eres pieza fundamental de este trabajo, pero sobre todo eres pilar fundamental en mi vida, te volviste mi protector, mi cómplice, mi guía, mi amor más bonito, gracias por comenzar a caminar a mi lado, siempre juntos de la mano, a donde sea que vayamos, por lo que sea nos espere, Te amo.

A mi querida asesora Profesora Nancy, mil gracias por acompañarme en este camino tan largo, por darme la bienvenida a la Universidad, acompañarme en el proceso y llevarme de la mano hasta mi gran clausura. Gracias por las risas, por las críticas constructivas, por su apoyo, por su optimismo y sobre todo por el cariño que me brindo en todo este proceso. La quiero mucho.

Gracias a todos ustedes por apoyarme, confiar y creer en mí, esto no fue fácil, pero su compañía siempre estuvo a mi lado, gracias por convertirme en una persona de bien, gracias por amarme. Esto es para todos ustedes. Gracias.

ÍNDICE

1. OBJETIVOS DEL TRABAJO	1
2. INTRODUCCIÓN	2
3. ANTECEDENTES	3
4. JUSTIFICACIÓN DEL TRABAJO	4
5. HIPÓTESIS	5
6. METODOLOGÍA POR EMPLEAR	6
7. INTRODUCCIÓN AL DESARROLLO WEB	7
7.1. HTML5	8
7.2. ESTRUCTURA	10
7.3. ETIQUETAS Y ATRIBUTOS	17
7.4. ESTRUCTURA SEMÁNTICA	48
8. INTRODUCCIÓN AL DISEÑO WEB CSS	51
8.1. CSS	51
8.2. ESTRUCTURA	52
8.3. COLORES	56
8.4. MEDIDAS DE CSS	65
8.5. PROPIEDADES	67
8.6. CSS RESPONSIVE	121
9. INTRODUCCIÓN A JAVASCRIPT	162
9.1. JAVASCRIPT	162
9.2. HOLA MUNDO	163
9.3. ESTRUCTURA	164
9.3.1. OPERADORES	170
9.3.2. COMENTARIOS	173
9.4. ESTRUCTURAS DE CONTROL DE FLUJO	173
9.5. DOM	179
10. FRAMEWORKS	183
10.1. BOOTSTRAP	184
10.2. INCORPORACIÓN DE BOOTSTRAP EN DISEÑO WEB	184
10.3. DISEÑO DE PANTALLA	188
11. CASO PRÁCTICO - APLICACIÓN DE BOOTSTRAP EN EL DESARROLLO WEB	229
12. ANÁLISIS DE RESULTADOS	233
13. CONCLUSIÓN	234
14. BIBLIOGRAFÍA	235

1. OBJETIVOS DEL TRABAJO

El objetivo de este trabajo, es mostrar una manera diferente de enseñar Desarrollo web, distinguiendo los elementos que competen a cada lenguaje que construye un sitio web.

- También busca distinguir este trabajo con la forma de explicar los elementos, sus funciones, el uso que se les puede dar y las combinaciones que se pueden lograr, fortaleciendo el conocimiento con teoría concreta e ilustraciones alusivas al tema, a su función y el código que se utiliza, de esa manera, brinda la oportunidad al usuario de experimentar con los temas involucrados.
- Por otra parte, justifica la importancia de implementar **BOOTSTRAP** en la creación de las páginas, pues el aporte que comparte el Framework aminoriza el trabajo y recursos.
- Finalmente realiza una comparación tangible y visual de un desarrollo antiguo entre **HTML, CSS, JS** vs el desarrollo implementando la nueva herramienta **BOOTSTRAP**.

2. INTRODUCCIÓN

Este trabajo es motivado por necesidad escolar, profesional y laboral, la implementación del desarrollo web, hoy en día es imprescindible para la sociedad, pues la vida literalmente gira alrededor de las tecnologías, ahora bien, el desarrollo web se implementó al mundo tecnológico hace unas décadas atrás, pues la necesidad en ese momento era cambiar la información que se tenía escrita en papel a un sitio web sencillo.

Con el paso del tiempo el concepto, función y aspecto del desarrollo web fue evolucionando, hasta conocerlo como la construcción de sitios web dinámicos con un contenido multimedia y su principal función es ser amigable, versátil, optimo y estilizado para el usuario. Y algo muy importante para el desarrollo web que se conoce ahora, es que necesita ser visualmente atractivo para el usuario, y eso propicia un gran problema para los desarrolladores, pues la mayoría carecen de paciencia y creatividad apropiada para la demanda del público.

Para ello, surgieron nuevas tecnologías “Frameworks”, que brindan la oportunidad de solucionar esas necesidades de los desarrolladores y complacen al usuario final. Aunque se tenga una nueva herramienta, es necesario entender cómo funciona **HTML**, **CSS**, y **JS**, pues con esos lenguajes se crea, implementa y mantiene el Desarrollo y Diseño web, es esencial tener las bases sólidas de los elementos que componen una página web.

De esta manera, surgió el interés de compartir los conocimientos necesarios, explicados con detalle sobre la función de cada elemento de todos los lenguajes, los ¿Por qué? ¿Cuándo? ¿Cómo? Y ¿Dónde? Son respondidos en este trabajo, pues no solo brinda conocimiento teórico y acontecido, sino también el elemento principal que es el conocimiento práctico y visual del código, así como también detallar su funcionamiento, como usarlo y conjugarlo en el plano web.

3. ANTECEDENTES

La manera en que el almacenamiento y el análisis de información ha sido uno de los grandes problemas en la informática. El descubrimiento de la computadora ha sido un gran progreso para el hombre, pues satisface las necesidades del cálculo en la información organizada.

Hoy en día la sociedad tiene acceso a la información con más facilidad, pues se ha convertido en algo de vital importancia. El avance tecnológico ha tenido gran progreso en corto tiempo, por tanto, el control y actualización de los datos y la tecnología es fundamental.

En la actualidad cualquier organización construye sistemas capaces de satisfacer las exigencias de su público, por ello, desarrollan entornos web que facilitan la accesibilidad de la información desde cualquier lugar del mundo.

4. JUSTIFICACIÓN DEL TRABAJO

El manejo de lenguajes del Desarrollo y Diseño web, es ahora un requisito laboral en la mayoría de las empresas, es decir, en este momento una persona como desarrolladora web (Back-End) ya es obsoleta, al igual que un diseñador web (Front-End), y si aún no lo son, el sueldo que se obtiene por el trabajo es mal pagado y las condiciones de trabajo no son agradables, hoy en día las empresas buscan acortar gastos y recurso humano, con el fin de obtener mejores resultados y beneficiarse de cualquier manera, para eso, en el campo laboral del desarrollo de tecnologías de cualquier software, es preciso que una sola persona sepa desarrollar e implementar ambas tareas (Full-Stack), donde el trabajo incrementa pero de igual manera el sueldo.

Este trabajo tiene como finalidad enseñar bases sólidas y concretas sobre los lenguajes del Desarrollo y la implementación de Diseño en las páginas web, ya que su contenido se encuentra ampliamente explicado de la tarea que desempeña cada lenguaje, su importancia, su función, su uso, estructura y cómo manejarlos y combinarlos, aparte de contener información teórica concreta y sustentada por autores reconocidos y sitios web certificados en el desarrollo y diseño de los sitios virtuales.

Por otro lado, también brinda de forma paralela, la importancia de implementar Frameworks de diseño en el contenido web, pues en este caso, crear un sitio web con **HTML, CSS y JS**, es un trabajo complicado y pesado, porque se invierte mucho tiempo, lógica y creatividad estructural. Por eso, la intervención de **BOOTSTRAP**, como herramienta de apoyo en el diseño web, permite agilizar la construcción de una página web sin problema, aminoriza tiempo y trabajo, pues la facilidad que ofrece al trabajarlo a la par del desarrollo web, permite obtener un trabajo completo, estructurado, organizado, diseñado, responsivo, óptimo y dinámico.

Teniendo en cuenta el contenido de este trabajo, van a existir 3 tipos de personas interesadas en este trabajo:

- El primer tipo de persona, será la que tenga un conocimiento de Back-End o Front-End podrá tener la posibilidad de complementar y afinar su profesión.
- El segundo tipo de persona, será la que no tenga ningún conocimiento, y al leer este trabajo, obtendrá el conocimiento y noción de lo que implica crear contenido web.
- El tercer tipo de persona, será la que tenga interés de involucrarse, pero no entienda nada del tema, con este trabajo podrá entenderlo y emprenderlo a la perfección.

De esta manera, se beneficiarán del tema para poder emprender la labor de crear sitios web.

5. HIPOTESIS

La falta de conocimiento y enseñanza escolar, en una de las ramas más importantes del Desarrollo web, como lo es el “Diseño web”, dentro de la carrera de Informática en la FESC, propicia el desinterés de los alumnos por aprender a dar presentación y organización a sus trabajos. Aunque, ese no es el peor de los casos, dentro del ambiente laboral, la falta de herramientas y conocimientos más actuales y utilizados dentro de las empresas, genera alumnos desempleados sin oportunidad de enfrentarse al mundo de los adultos, por ende, genera frustración e inseguridad.

Este trabajo busca demostrar una manera diferente de enseñar Desarrollo web, explicando ampliamente la función y el uso de los elementos que componen cada uno de los lenguajes que construyen un sitio web, mejorando así el aprendizaje con teoría necesaria y concreta, imágenes ilustrativas y muestra explícita del código utilizado, de esta manera el alumno podrá despertar la iniciativa de aprender, experimentar, investigar y culminar con un nuevo aprendizaje que le permita encontrar un buen trabajo.

Cumpléndose lo anterior, también se demostrará una comparación tangible y visual del desarrollo anticuado que ofrece **HTML**, **CSS** y **JS**, contra la implementación de **BOOTSTRAP** como herramienta funcional y precisa del Diseño web, sin menospreciar la labor que ejercen los lenguajes anteriores, pues en conjunto, se convierten en una herramienta eficaz y eficiente para la creación de páginas web.

6. METODOLOGÍA POR EMPLEAR

La necesidad de este trabajo involucra muchos objetivos, pero lo principal y más importante, se rige por lograr que las personas comprendan la importancia de cada elemento que compone el Desarrollo y Diseño web, pero no solo eso, también es de vital importancia conocer sus características, sus funciones, los usos que se les puede dar y sobre todo lo que se puede lograr con ellos en conjunto, todo esto sustentado con fuentes fieles y reconocidas, pero sobre todo actualizadas.

Para lograr el objetivo se hace uso de la **metodología de investigación descriptiva**, la cual aporta un camino objetivo: la recolección de los elementos necesarios de la investigación para ser descritos y observados mediante su uso y exposición, categorizados en la importancia que desempeñan y lograr e, estadísticamente hablando, de una conclusión positiva ante el desempeño, progreso y culminación del contenido de este trabajo.

7. INTRODUCCIÓN AL DESARROLLO WEB

Para crear un Sitio Web, se necesita de ciertos conocimientos y elementos base para poder planear, estructurar, desarrollar y diseñar un entorno web que cubra las necesidades de la persona u organización que lo solicite. Para lograr un trabajo excepcional existe algo muy importante para elaborar los proyectos, hoy en día el Desarrollo Web suele dividirse en dos partes “Front-End” y “Back-End”, lo que significa que un proyecto suele hacerse en partes por equipo en cualquier nivel escolar, laboral o profesional. Gran parte de la comunidad de Desarrollo Web no cuentan con un conocimiento al 100% para realizar ambas partes, pues, aunque es el ideal, la gran mayoría suele inclinarse por alguna de las dos etapas del desarrollo y las personas que conocen ambas partes se les llama Full-Stack.

Pero bueno tú te preguntarás... ¿Y qué significa Front-End y Back-End?, como se comentaba anteriormente, un proyecto se compone de Estructura, Funcionalidad y Vista en cualquier plataforma web. Como bien se sabe, para que una persona tenga interés por una Aplicación Web en cualquier dispositivo móvil, necesita tener una interfaz llamativa, innovadora y moderna, a esto se le conoce como Front-End, es decir, lo que un usuario percibe visualmente al interactuar en el sitio.

El Back-End tiene como objetivo hacer que el proyecto desempeñe las funciones correctamente, esto se logra con programación y bases de datos que genera el administrador del sitio y por ende gestiona la información del sitio, en pocas palabras, es lo que está oculto a la vista del usuario final, y solo lo puede manipular quien lo desarrollo.

Para un Desarrollador es imprescindible entender el propósito del sitio, esto se refiere a la audiencia que se espera tenga el sitio web, los cuales se distinguen por los aspectos geográficos, tecnológicos y demográficos, al igual que conocer el tipo de producto o servicio que se ofrece al público, así como todas las incógnitas que surgen al momento de planear y organizar el desarrollo de cualquier aplicación.

Contemplando lo anterior, cuando se comience a plantear la estructura y diseño del entorno, se necesita de mucha información para poder esclarecer cualquier tipo de duda y con ello poder adaptarse para ambas situaciones, es decir, la satisfacción del cliente y del usuario.

En el desarrollo de alguna aplicación web, se cuenta con un mundo de herramientas tecnológicas que se pueden utilizar para su implementación, las cuales cumplen la demanda que se tiene hoy en día, pues son necesarias para tener una aplicación web optima, eficiente y atractiva al usuario, aunque esto no quiere decir que se le asignen todas a un solo sitio web, pues cada una tiene un objetivo, que es resolver un problema en particular.

7.1.HTML5

Vértice (2009) afirma: Las páginas web de los primeros tiempos de vida de Internet eran tan simples como un documento de texto o un simple folleto en el que el desarrollo más complicado consistía en las fuentes y colores. El diseño solía realizarse de arriba abajo y de izquierda a derecha, intercalando secuencias de imágenes y textos, con muchos saltos de línea y otros elementos separadores como barras horizontales y las viñetas de lista. Se construían en lenguaje **HTML**. No obstante, este tipo de páginas generalmente funcionaban bien, los programadores se encontraron con problemas como la superposición de colores y efectos de texto demasiado resaltados y sobredimensionados. (p.7)

Con el paso del tiempo, el desarrollo de comercio electrónico y las necesidades comunicativas, las páginas web comenzaron a tener estructuras más complejas y otros lenguajes de programación y metodologías entraron en juego, como JavaScript, que permitía crear pequeños programas muy útiles en la web. Cuando se comenzó el poder de la **www** (World Wide Web), sitios web y en general el internet, surgió el convencimiento a los desarrolladores de que había que cambiar la apariencia de las páginas por completo, hacerlas visualmente más atractivas para el usuario.

Vértice (2009) menciona: En 1994 fue creado el consorcio W3C, para crear objetivos y estándares en el desarrollo de lenguaje **HTML**. Desde entonces, este lenguaje ha tenido varias versiones hasta la actual 5.0. Con la intención de ofrecer un contenido dinámico a través de la red. (p8) Para esto surgieron nuevas generaciones de páginas web cuyas principales características de diseño eran los iconos que reemplazaban a palabras, fondos de pantalla que se formaban a partir de mosaicos, botones con bisel y relieve, banners que sustituían a las cabeceras y el uso de despleables. Por otra parte, estos diseños tendían a ser muy recargados con tecnologías que no respetaban el objetivo de la página. Introducían muchos iconos llamativos, gifs animados y colores primarios.

Pero, aun así, había diseñadores que conseguían utilizar el código **HTML** de maneras innovadoras consiguiendo el efecto más deseado. Así fue como surgieron las etiquetas **<TABLE>**, aunque más pensada para introducir datos estadísticos que para el propio diseño”.

De esta manera se conseguía un mejor control del posicionamiento de los elementos y se solucionaban muchos problemas estéticos. Cuando la tecnología fue cambiando, las páginas aún tenían que adaptarse a los cambios de monitores, ya fuera por los pixeles o nuevas profundidades de colores, aunado a esto, que los navegadores web contenían reglas propias y que no permitían visualizar de igual manera una página web de navegador a otro.

Más tarde llegó un periodo en el que los desarrolladores habían aprendido de todos estos errores del pasado y comenzaron a realizar diseños agradables que permitían al visitante focalizar su atención en los principales productos y servicios que ofertaban las páginas. Los websites llegaron a convertirse en el “punto de información” de los negocios de las empresas. La mayoría de los desarrolladores se adentraron a la interactividad, el lenguaje **DHTML** (HTML dinámico) y los clips flash para impresionar a la audiencia. Las tablas eran limitantes, así que tenían que solventarlo escribiendo gran cantidad de código e implementando estilos,

lo cual era un trabajo duro y complicado. Para deshacerse de este problema, introdujeron el lenguaje **CSS** (hojas de estilo en cascada), el cual se hizo muy popular en poco tiempo entre los desarrolladores. (Vértice, 2009, p.9)

En este punto los programadores habían alcanzado un nivel de madurez en el uso de la tecnología muy altos. Aquí surgieron la optimización de las páginas para ser encontradas y posicionadas por los motores de búsqueda, y los layouts, la última tendencia de diseño en esos días. Pues permitían estructurar una página y se utilizan para dirigir la atención del usuario al principal contenido de ésta, no del diseño. Motivados por esta técnica, empezó a mantenerse el fondo (background) con un diseño muy simple y a dejar la parte exterior del layout en blanco para evitar la distracción.

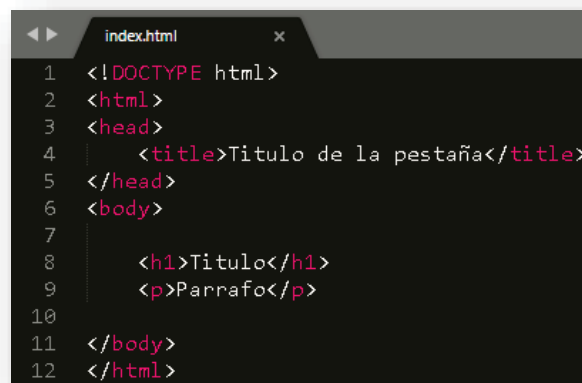
El desarrollo web comenzó a mezclarse con lenguajes de programación avanzados que permitían importantes acciones como (**PHP**, **C#**) para conectar una página web con una base de datos, esto era primordial en las tiendas online que surgían en esos momentos. El diseño web también prospero aún más pues integraba elementos multimedia como el sonido, los clips de video, la animación, bases de datos, los mundos 3d y virtualización. Aunque los diseñadores eran conscientes de que el principal obstáculo de estos websites era la velocidad de descarga, ya que su demanda por el contenido multimedia creció. (Vértice, 2009, p.10)

De repente y gracias al desarrollo de ciertas herramientas, los usuarios se dieron cuenta de que podían hacer muchas más cosas en Internet que simplemente mirar páginas, pues también podían participar. Así nació la web 2.0, cuyos patrones de diseño se caracterizan porque el usuario puede manejar aspectos visuales y la presentación de tecnologías como **AJAX** y con un conjunto de combinados, facilitan el desarrollo de interfaces ricas y adaptables". (Vértice, 2009, p.11)

7.2. ESTRUCTURA

HTML5 es un lenguaje de marcado de hipertextos, es decir, son bloques de comandos que generan la construcción de un sitio web, para poder realizar su construcción **HTML5** utiliza etiquetas o marcas, que consisten en breves instrucciones con una apertura y cierre, donde contienen la información que se desea transmitir al usuario final, estos comandos se generan en editores de texto, los más utilizados hoy en día son: TextEdit Open, Sublime Text, Brackets, etc. por mencionar algunos, cuando el código está concluido, se guarda con extensión (.HTML) y al ejecutarlo en un navegador web, el compilador del navegador traduce el código y muestra el contenido con la apariencia comúnmente conocida. (W3schools, 2018-2019)

Las etiquetas en **HTML5** se identifican con los nombres de los elementos del lenguaje de marcado, entre paréntesis angulares o mejor conocidos como corchetes, las etiquetas se componen normalmente en pares como **<HTML>** (Etiqueta de apertura) y **</HTML>** (Etiqueta de cierre), la etiqueta final se escribe igual que la anterior, pero añadiendo un slash comúnmente denominado (Diagonal) antes de la palabra, lo que nos indica que ahí termina el contenido de esa etiqueta.



```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title>Titulo de la pestaña</title>
5 </head>
6 <body>
7
8   <h1>Titulo</h1>
9   <p>Parrafo</p>
10
11 </body>
12 </html>
```

Ilustración 1- Imagen propia

Hernández (2014) menciona una de las ventajas principales de **HTML5** está en la facilidad para realizar una estructura básica de la página, esto se debe al paso del tiempo, pues se han acortado las declaraciones, son aún más legibles, con mayor significado”. (p55) Por lo tanto, son más fáciles de entender e implementar. Todo documento **HTML5** debe empezar con la etiqueta **<DOCTYPE>**, esta declaración tiene la función de determinar al editor de texto que ese archivo que se está creando pertenece a **HTML5**.

Para entenderlo mejor, **<!DOCTYPE>** nos sirve para indicar a nuestro espacio de trabajo, es decir, a nuestro editor de textos que el documento que se está creando, está descrito siguiendo la estructura determinada por un DTD concreto. Un DTD es la “Definición del Tipo de Documento”, por tanto, nuestro archivo ya es declarado en extensión y sintaxis como **HTML5**. Así que, el DTD es dónde se define la estructura que debe tener el documento. Por tanto, se utiliza **<!DOCTYPE>** para informar qué DTD es un proyecto **HTML5** para trabajar, aclarado esto, **HTML5** es la versión de lenguaje marcado más actual que tenemos hoy en día. (W3schools, 2018-2019)

Después se declara la etiqueta **<HTML>** dentro de **<!DOCTYPE>** y fuera como segunda etiqueta, porque de esta forma se está declarando que es el elemento raíz de una página web. Cada etiqueta tiene un cierre, por tanto, la etiqueta principal siempre va a ser **<HTML>** siendo así la que guarda todo el contenido de una página web.

El siguiente elemento o etiqueta que se nos presenta en la estructura es **<HEAD>**, usualmente es utilizado como contenedor de metadatos. Los metadatos suelen definir el título del documento, el conjunto de caracteres, estilos, enlaces, guiones, entre otros. Es decir, define la información acerca del documento, pero no es algo que se muestre al usuario final. Como, por ejemplo: **<TITLE>**, **<STYLE>**, **<META>**, **<LINK>**, **<SCRIPT>**, y **<BASE>**.

La etiqueta **<TITLE>** es el elemento que define el título de la pestaña del navegador

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <title>Titulo del documento y la pestaña del navegador</title>
5 </head>
```

Ilustración 2 - Imagen propia

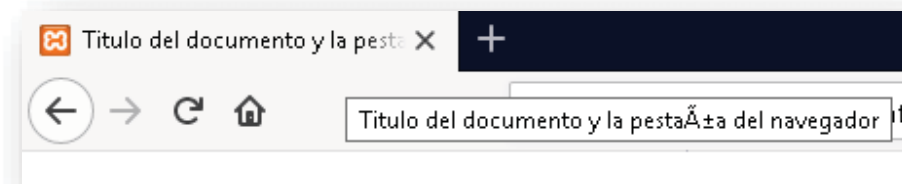


Ilustración 3 - Imagen propia

La etiqueta **<STYLE>** se utiliza para definir el estilo de una sola página HTML, dentro de esta etiqueta, se puede estilizar la sección de **HTML5** que se elija, más adelante se hablará sobre los atributos relacionados al diseño de un sitio web.

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4
5 <style>
6   body{background-color: salmon;}
7 </style>
8
9 </head>
```

Ilustración 4 - Imagen propia

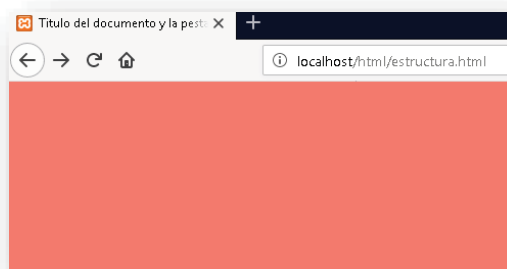


Ilustración 5 - Imagen propia

La etiqueta **<META>** es un elemento que se utiliza para especificar un conjunto de caracteres como: descripción de páginas, palabras clave, autor, y otros metadatos. Los metadatos se utilizan principalmente para los navegadores, es decir, dentro de esa sección del código, el navegador identifica el idioma en el que se desenvuelve la página, archivos externos, etc.

Para definir un conjunto de caracteres, en el desarrollo web se utiliza **UTF-8**.

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4
5 <meta charset="UTF-8">
6
7 </head>
8 <body>
9 <h1>Información</h1>
10 </body>
11 </html>
```

Ilustración 6 - Imagen propia

Esto es importante porque identifica cuales letras del contenido contienen caracteres diferentes a los que un lenguaje de desarrollo comúnmente maneja. Por ejemplo, si no existiera ese elemento, la palabra “Información” aparecería de esta manera:

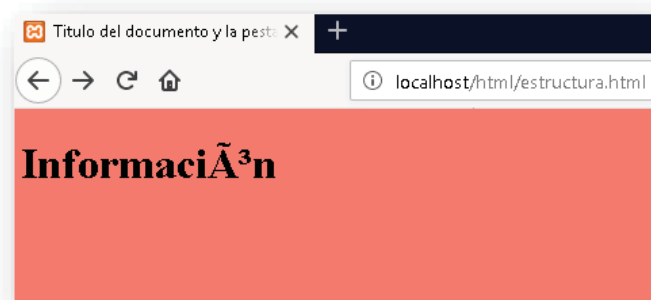


Ilustración 7 - Imagen propia

En cambio, si recurrimos la etiqueta **<META>** con el atributo que identifica los caracteres, veríamos de esta forma la página:

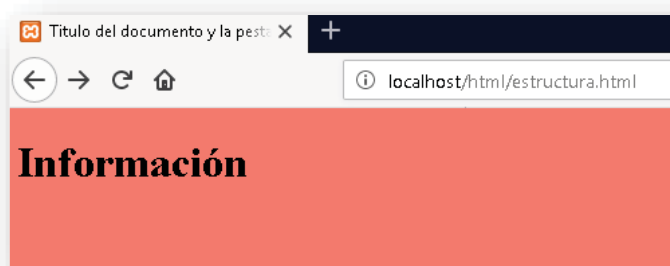


Ilustración 8 - Imagen propia

UTF-8 (Formato de Transformación UCS/Unicode de 8 bits) es un sistema de codificación de caracteres de longitud variable. Puede representar cualquier carácter en el estándar de Unicode, y aun así es compatible con ASCII. Un beneficio de UTF-8 es su habilidad para manejar idiomas que tienen cientos o miles de caracteres.

Existen otros atributos sobre la etiqueta **<META>** que realmente se tienen solo a la vista del desarrollador, sus objetivos son meramente información que no se representa gráficamente al usuario.

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4
5 //Definir una descripción de su página web
6 <meta name="description" content="DesarrolloWeb">
7
8 //Palabras clave para motores de búsqueda
9 <meta name="keywords" content="HTML, CSS, JavaScript">
10
11 //Autor de la Página
12 <meta name="author" content="Alba">
13
14 //Actualización de la página
15 <meta http-equiv="refresh" content="5">
16
17 </head>
```

Ilustración 9 - Imagen propia

La etiqueta **<LINK>** por otra parte, es un elemento que se utiliza para enlazar hojas de estilo externas al documento que se está trabajando.

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4
5 <link rel="stylesheet" href="estilo.css">
6
7 </head>
```

Ilustración 10 - Imagen propia

El atributo **REL** especifica la relación entre el documento actual y el documento vinculado, mientras que el otro elemento **HREF** se utiliza como atributo para definir la dirección del enlace externo. La siguiente etiqueta que puede contener **<HEAD>**, es **<SCRIPT>**, que se utiliza para definir un código de naturaleza **JS**, es decir, **JAVASCRIPT**.

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4
5 <script>
6 function myJS() {
7     document.getElementById("prueba").innerHTML = "Hola JavaScript";
8 }
9 </script>
10 </head>
11 <body>
12
13 <h1>Hi primer JS</h1>
14 <p id="prueba">Cargando...</p>
15 <button type="button" onclick="myJS()">Saluda</button>
16
17 </body>
18 </html>
```

Ilustración 11 - Imagen propia



Ilustración 12 - Imagen propia



Ilustración 13 - Imagen propia

Y por último de lo que puede contener **<HEAD>**, es la etiqueta **<BASE>**, donde su único objetivo es especificar la URL o las URLs que se utilizaron para conformar la página.

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4
5   <base href="https://www.w3schools.com" target="_blank">
6
7 </head>
```

Ilustración 14 - Imagen propia

Se continua con la etiqueta **<BODY>**, que abarca todo el contenido de un sitio web que puede ser visible para el usuario, comúnmente se comienza con una etiqueta de encabezado, estas son definidas por las etiquetas **<H1>** a **<H6>**.

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4
5 </head>
6 <body>
7   <h1>Parrafo</h1>
8   <h2>Parrafo</h2>
9   <h3>Parrafo</h3>
10  <h4>Parrafo</h4>
11  <h5>Parrafo</h5>
12  <h6>Parrafo</h6>
13 </body>
14 </html>
15
```

Ilustración 15 - Imagen propia



Ilustración 16 - Imagen propia

Como se puede observar en las imágenes anteriores, el objetivo de cada etiqueta es establecer el tamaño que le corresponde al contenido que tengan dentro de ellas, según la etiqueta que se disponga a ocupar.

Para introducir texto al sitio web, se ocupa la etiqueta `<P>`.

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4
5 </head>
6 <body>
7   <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do
8   eiusmod
9   tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim
10  veniam,
11  quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea
12  commodo
13  consequat. Duis aute irure dolor in reprehenderit in voluptate velit
14  esse
15  cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat
16  cupidatat non
17  proident, sunt in culpa qui officia deserunt mollit anim id est
18  laborum.</p>
19 </body>
20 </html>
```

Ilustración 17 - Imagen propia

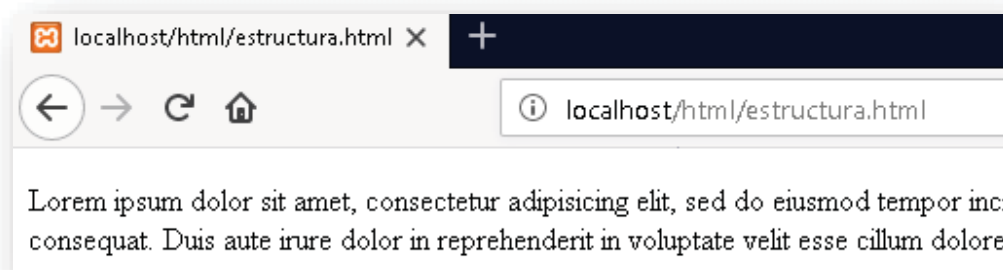


Ilustración 18 - Imagen propia

La estructura principal, se ve de esta forma.

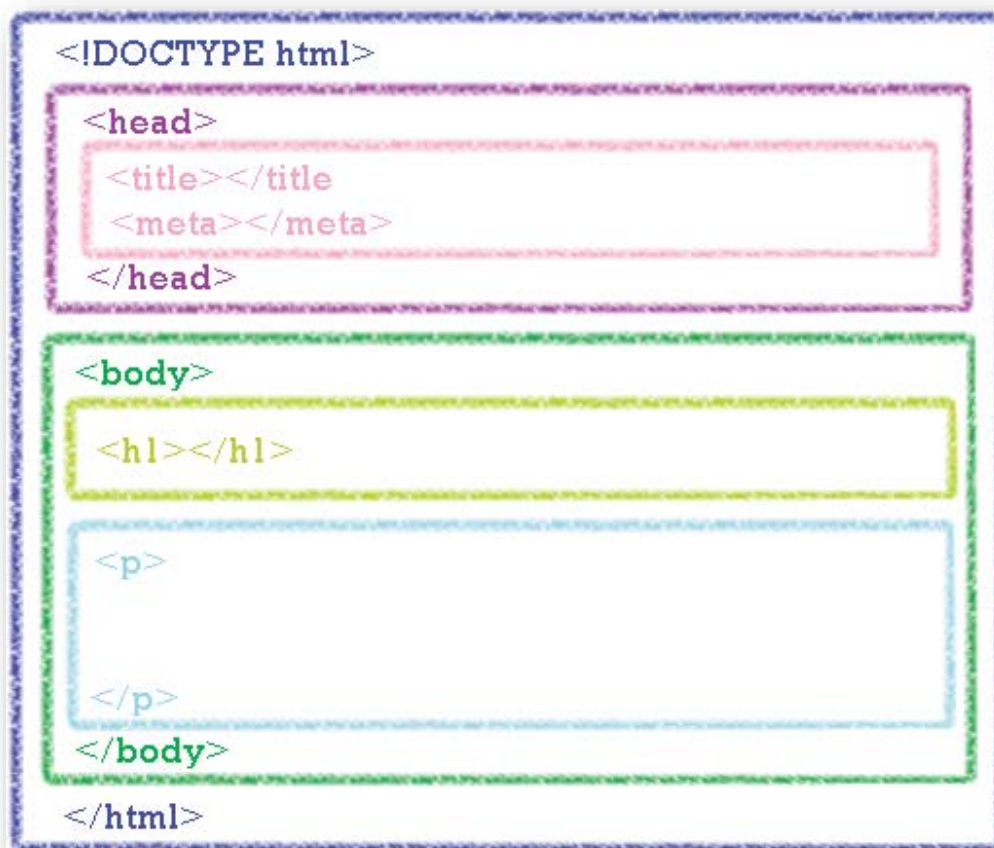


Ilustración 19 - Imagen propia

Las etiquetas se distribuyen de esta manera, abarcando toda la página y distribuyéndose de una manera absoluta.

7.3. ETIQUETAS Y ATRIBUTOS

En **HTML5**, es importante conocer la estructura anterior, pues es la base de una página web y parte importante del inicio de su formación, pues conforme se avance en este proyecto, se irán conociendo los cambios que conocemos en la actualidad.

Ahora bien, existen muchas más etiquetas de **HTML5** que se conocerán a lo largo de este capítulo, pero lo importante es que se conocerán los atributos que contienen cada una de las etiquetas que comúnmente se utilizan en el Desarrollo Web. Los atributos proporcionan información adicional a los elementos de **HTML5**, estos se definen en la etiqueta de apertura, después del nombre del elemento a emplear, más no afecta a la etiqueta de cierre, porque sería un error de sintaxis.

En la etiqueta **<HTML>** el atributo **LANG** puede ser utilizado para indicar el idioma de una página web o una parte de una página Web. Esto facilita la ayuda a los motores de búsqueda y a los navegadores. Si tiene cualquier contenido en la página que esté en un idioma diferente del declarado en el elemento **HTML5**, se puede usar este atributo de idioma en los elementos que alberguen a ese contenido. Esto le permite estilizarlo o procesarlo de manera diferente. La ISO 639-1 define las abreviaturas para los idiomas.

```
1 <!DOCTYPE html>
2 <html lang="es">
```

Ilustración 20 - Imagen propia

Los enlaces en las páginas web son realmente indispensables, su función es ser un puente entre páginas web, pues nos ayudan a saltar de una página a otra sin tener que teclear su URL, de esta forma se puede conformar un sitio web, a estos enlaces se les conoce como Hipervínculos. Los hipervínculos en **HTML5** se definen con la etiqueta **<A>**. La dirección o ruta del destino del enlace se especifica en el atributo **HREF**, mencionado anteriormente.

Existen dos tipos de rutas de enlaces que se pueden vincular al código que se está generando, un enlace “Externo”, es aquel que escribe la dirección o ruta completa de un sitio web en internet.

```
<body>
  <a href="https://www.google.com/">Google</a>
</body>
```

Ilustración 21 - Imagen propia

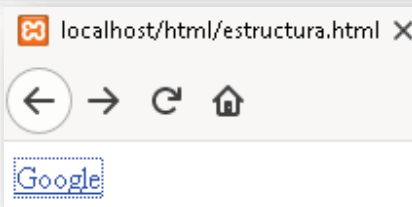


Ilustración 22 - Imagen propia

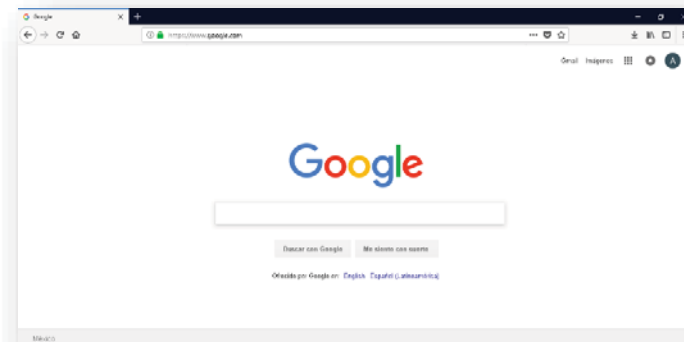


Ilustración 23 - Imagen propia

El segundo tipo de ruta se le conoce como “Interno”, su objetivo es mostrar una página web pero localizada en el servidor del desarrollador, es decir, el archivo se ubica en la URL exacta de donde se encuentra en la computadora.

```
<body>
  <a href="pres_abril/index.html">Sitio Web</a>
</body>
```

Ilustración 24 - Imagen propia

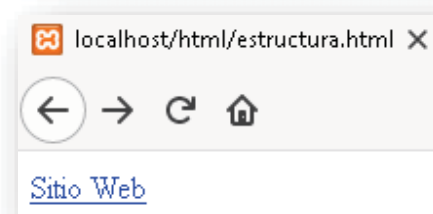


Ilustración 25 - Imagen propia

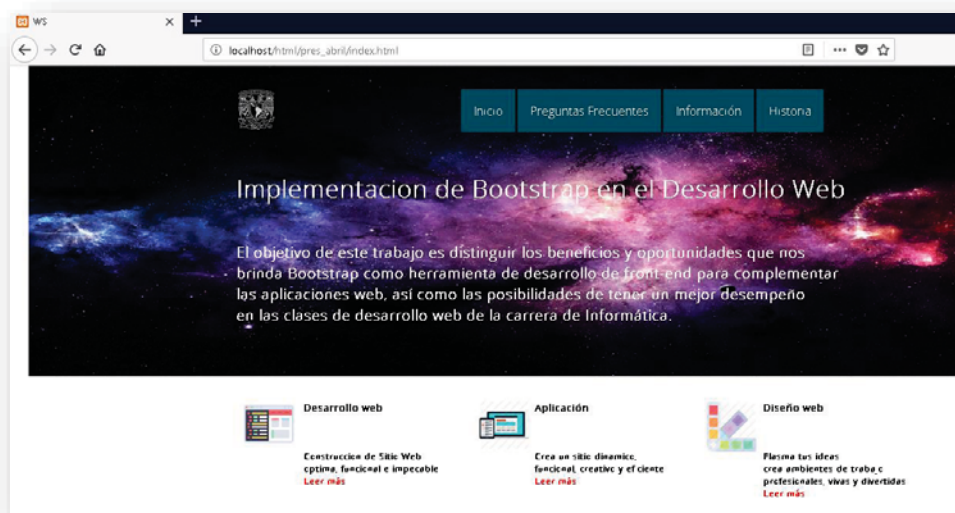


Ilustración 26 - Imagen propia

Entrados en el tema de una ventana nueva por medio de los enlaces, también se pueden encontrar diferentes atributos que nos permiten hacer más fácil el manejo del sitio web, por ejemplo, el atributo **TARGET**, ofrece diferentes modos de abrir un enlace, con el valor **_BLANK** abre en una nueva página o pestaña el enlace que se está seleccionando.

```
<body>
  <a href="pres_abril/index.html" target="_blank">Sitio web</a>
</body>
```

Ilustración 27 - Imagen propia

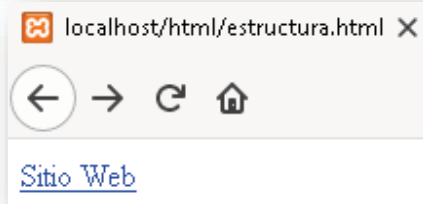


Ilustración 28 - Imagen propia

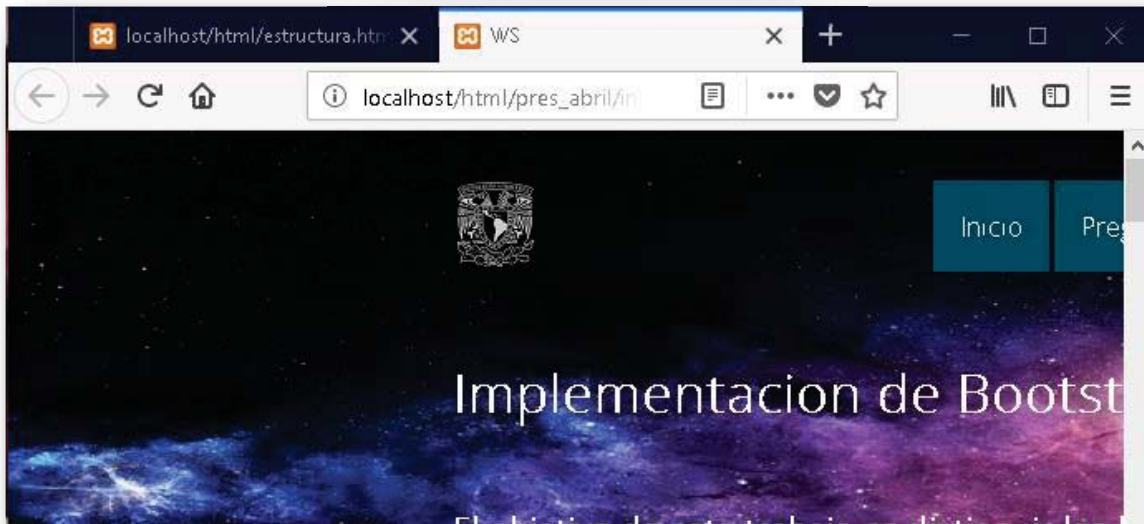


Ilustración 29 - Imagen propia

Con **_SELF** se obtiene el modo predeterminado que tiene **HTML5** para abrir un enlace, es decir, que abre el enlace en la misma página que se está visualizando.

```
<body>
  <a href="pres_abril/index.html" target="_self">Sitio web</a>
</body>
```

Ilustración 30 - Imagen propia

Ahora bien, se pueden incluir diferentes tipos de enlaces que complementen un Sitio Web para el usuario final, por ejemplo, crear un enlace para un correo electrónico:

```
<body>
  <a href="mailto:info@gmail.com">Para más información: desarrolloweb@gmail.com </a>
</body>
```

Ilustración 31 - Imagen propia

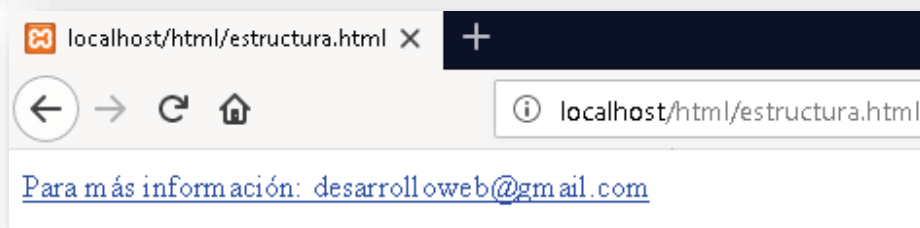


Ilustración 32 - Imagen propia

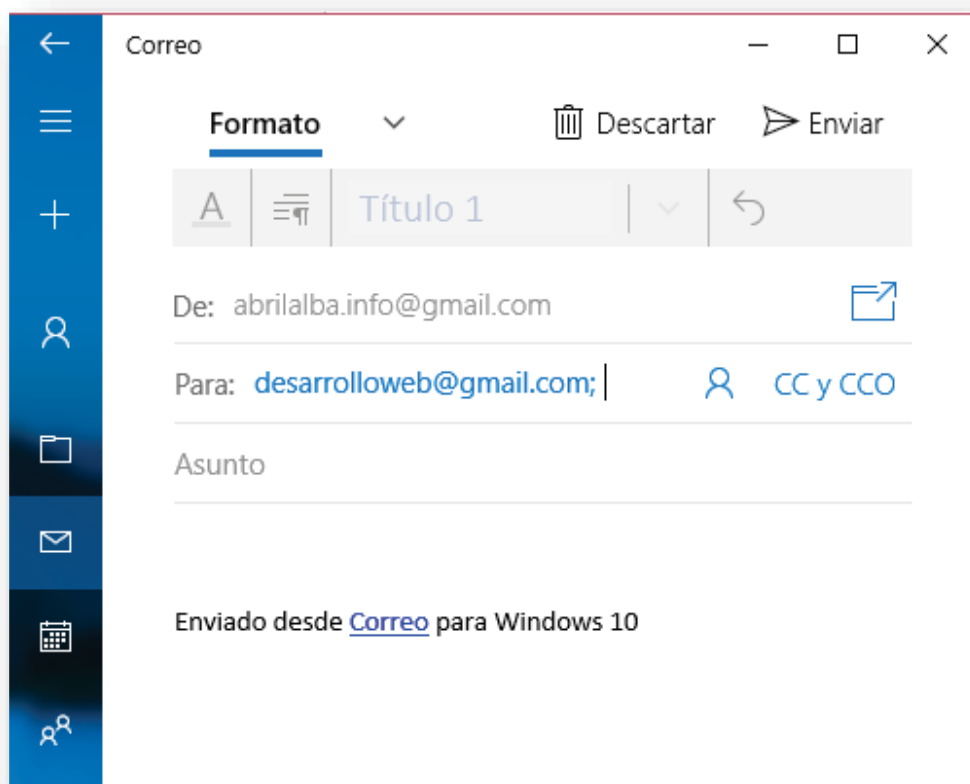


Ilustración 33 - Imagen propia

La función que hace **MAILTO** es enlazar un acceso directo a una página de correo electrónico de tu sistema operativo predeterminado, es decir, abre una ventana externa de correo electrónico con la cual se accede a tener contacto con la administración del sitio web.

También se pueden hacer enlaces que descarguen archivos o imágenes, basta con tener en el servidor el archivo y agregar el atributo en el código y el mismo navegador identificará que ese archivo o imagen es descargable.

```
<body>
  <a href="documentos.zip">Archivo zip</a>
</body>
```

Ilustración 34 - Imagen propia

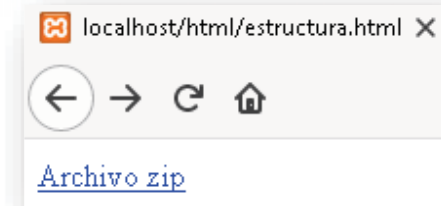


Ilustración 35 - Imagen propia

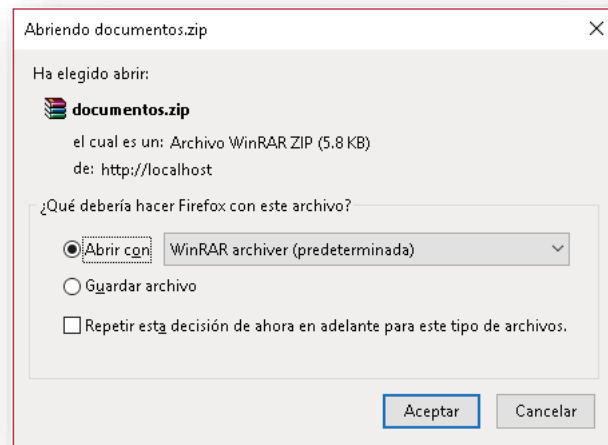


Ilustración 36 – Imagen propia

Como se muestra en la imagen, el texto que está entre las etiquetas de apertura y cierre **<A>**, se encuentra el texto que ve el usuario para identificar que es un enlace.

Ahora bien, algo muy importante dentro de los enlaces, es no confundirlos con botones, ya que, un enlace es una conexión entre diferentes páginas web y un botón se asocia más con acciones como “envío, cambio o eliminación de datos”, esto pertenece al ambiente de un formulario para base de datos, por lo cual no “re direcciona a otro contenido web” como un enlace, para poder crear un botón se cuenta con dos etiquetas, la primera es **<BUTTON>** aunque no se utilice frecuentemente para el desarrollo de páginas web, no está por demás mencionarla, pues su primordial función es agregar texto e imágenes sin problema dentro de la etiqueta.

```
<button> Click me!</button>
```

Ilustración 37 - Imagen propia



Ilustración 38 - Imagen propia

Como se puede observar, la etiqueta **<BUTTON>** no tiene la gran función, pues solo es un botón que no realiza ninguna acción, por ello no se suele utilizar, pues prácticamente es obsoleta, por supuesto que existe una función que hará que el botón realice una acción y una redirección, es decir, dentro del campo de JS existe un evento que permite hacer dichas hazañas, pues **ONCLICK** tiene por objetivo realizar una acción por si solo y re direccionar o abrir ventanas emergentes según lo que se desee con el botón, todo esto será posible si se le dicta dentro de la etiqueta el atributo **ROLE**, con esto el navegador sabrá que esa etiqueta es un enlace más no un botón, aclarado ese punto lo más recomendable es no alterar las funciones de las etiquetas de **HTML5**, porque los navegadores web suelen no ser compatibles con dichas funciones, por otra parte, esta regla esta dictada por los estándares de desarrollo web W3C.

```
<button name="facebook" role="link" onclick="window.location='http://www.facebook.com'"> Click me!</button>
```

Ilustración 39 - Imagen propia

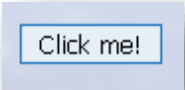


Ilustración 40 -
Imagen propia



Ilustración 41 - Imagen propia

Por otro lado, **<INPUT>** solo permite texto y es el elemento más recomendable para crear botones, como ya se dijo anteriormente, un botón solo es utilizado para fines administrativos de información, por tanto, suelen utilizarse dentro de un formulario, esto se da más cuando se trabaja con bases de datos y servidores.

```
<input type="button" value="Click me">
```

Ilustración 42 - Imagen propia

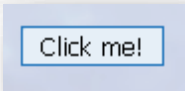


Ilustración 43 -
Imagen propia

También los usos de la multimedia dentro de las páginas web son fundamentales para la interactividad del usuario final con el sitio web, las imágenes, los videos y audios ayudan a que sea realmente llamativo e interesante el entorno. En **HTML5** se definen las imágenes con la etiqueta ****, esta etiqueta es vacía pues no tiene etiqueta de cierre y solo puede contener atributos.

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4
5 </head>
6 <body>
7 
8 
9 </body>
10
11 </html>
12
```

Ilustración 44 - Imagen propia

El atributo **SRC** especifica la URL de donde se encuentra el archivo, ya sea dentro del servidor y/o carpeta, el atributo **ALT** proporciona la opción de escribir un título o pie de la imagen, esto con el objetivo de si el usuario no puede acceder a la visualización de la imagen, el texto les indique de que trata la ilustración, si una imagen dentro de **HTML5** no tiene el atributo **ALT**, esa página web no es válida.

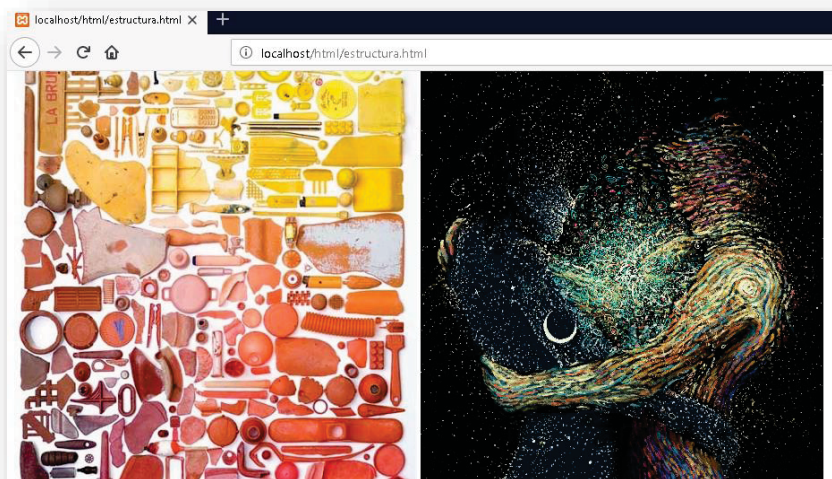


Ilustración 45 - Imagen propia, imágenes ilustrativas tomadas de https://www.taringa.net/miajuli/el-encuentro-de-dos-personas-es-como-el-contacto-de-dos-sustancias-qui_q1t4t, <https://www.pinterest.com.mx/pin/108790147218180336/>

Dentro de **HTML5** se pueden utilizar imágenes de cualquier extensión png, jpeg, jpg, incluyendo gif's, cuando las imágenes se encuentran en otra carpeta o servidor, solo basta con iniciar la URL desde raíz y completar la guía de donde se encuentre la imagen que se busca exponer.

También se tiene la opción de volver una imagen a un enlace, solo se necesita abrir una etiqueta **<A>** con su respectivo atributo **HREF**, y dentro de esta etiqueta, insertar la etiqueta

****, con eso el cursor al pasar por encima de la imagen se mostrará una manita en lugar del apuntador. Puede ser una imagen normal, o un icono que enlace a hacer más grande la imagen, enlazar a otra página web, a una dirección de correo o simplemente abrir la imagen para tener una mejor vista de ella, esto se puede realizar como se expusieron anteriormente los enlaces.

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4
5 </head>
6 <body>
7
8 <a href="www.google.com">
9 
10 </a>
11
12 </body>
13
14 </html>
```

Ilustración 46 - Imagen propia

También se pueden anexar los atributos **STYLE** dentro de la etiqueta ****, esto es con el uso particular para las imágenes, pues tenemos la opción de señalar el tamaño que se desea que tenga la imagen en el sitio web.

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4
5 </head>
6 <body>
7
8 <a href="www.google.com">
9 
10 </a>
11
12 </body>
13
14 </html>
```

Ilustración 47 - Imagen propia

Alternativamente se pueden utilizar las propiedades **WIDTH** y **HEIGHT** sin estar anidado al atributo **STYLE**, aunque esta segunda opción es más recomendable, pues evitamos que el código se confunda y altere las imágenes.

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4
5 </head>
6 <body>
7
8 <a href="www.google.com">
9 
10 </a>
11
12 </body>
13
14 </html>
```

Ilustración 48 - Imagen propia

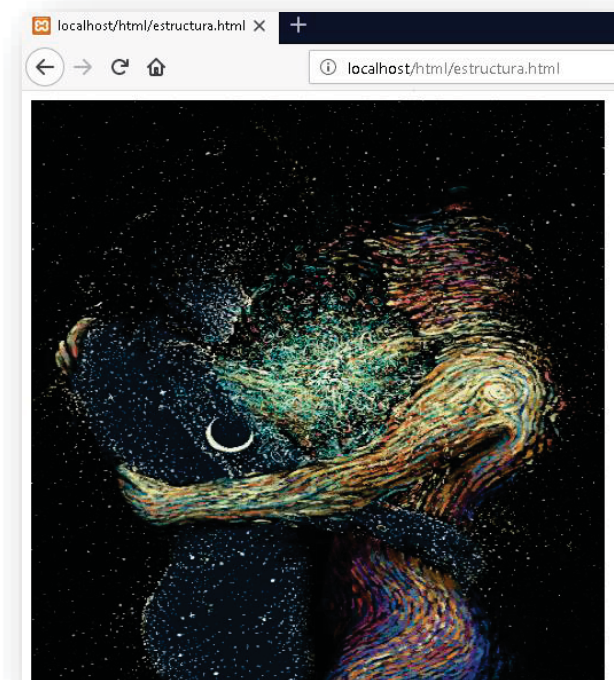


Ilustración 49 - Imagen propia, imagen ilustrativa tomada de <https://www.taringa.net/miajuli/el-encuentro-de-dos-personas-es-como-el-contacto-de-dos-sustancias-qui-q1t4t>

Dentro del ambiente de multimedia, también se encuentran los videos, para declararlo se puede hacer de diversas maneras, la principal es citar la etiqueta **<VIDEO>**, esto le indica a **HTML5** que se va a adjuntar un archivo con extensión mp4, es decir, un video clip, una película u otros flujos de video con esa extensión, los atributos que permite contener esta etiqueta son **STYLE** pues **WIDTH** y **HEIGHT** como se mencionó anteriormente, indican el tamaño que se les asigne en sus respectivos parámetros y **CONTROLS** facilita la tarea de asignar los controles de “continuar” y “pausa” en el video sin agregar funciones en **JS**.

Dentro de la etiqueta **<VIDEO>** se coloca la siguiente etiqueta **<SOURCE>** la cual indica el origen del video y cuáles son los recursos que necesita el navegador para que se ejecute el video en la página, de lo contrario aparecerá un mensaje de error en el navegador, pues la extensión del video puede ser incompatible con el navegador, usualmente se añaden dos videos de diferente extensión para que se permita ejecutar en cualquier navegador alguno de los dos videos, en este caso las extensiones MP4, MOV y OGG son compatibles con los navegadores más demandados, es decir, con FIREFOX, CHROME, OPERA, pero MP4 es universal incluso en SAFARI e INTERNET EXPLORER.

```
<video width="320" height="240" controls>
  <source src="unpocoloco.mp4" type="video/mp4">
</video>
```

Ilustración 50 - Imagen propia

Dentro de la etiqueta **<SOURCE>** encontramos atributos que se utilizan para enlazar el video de los archivos multimedia a la página web, en este caso hablamos de **SRC**, el cual nos permite hacer la misma función que **HREF**, pero en este caso **SRC** es específicamente para Audios o Videos dentro de **HTML5**. Otro de los atributos es **TYPE**, su objetivo es definir la extensión del video.

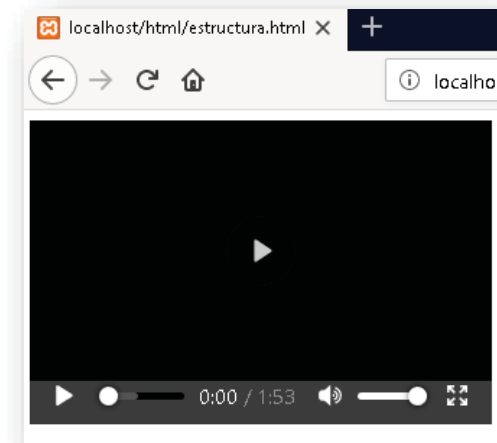


Ilustración 51 - Imagen propia

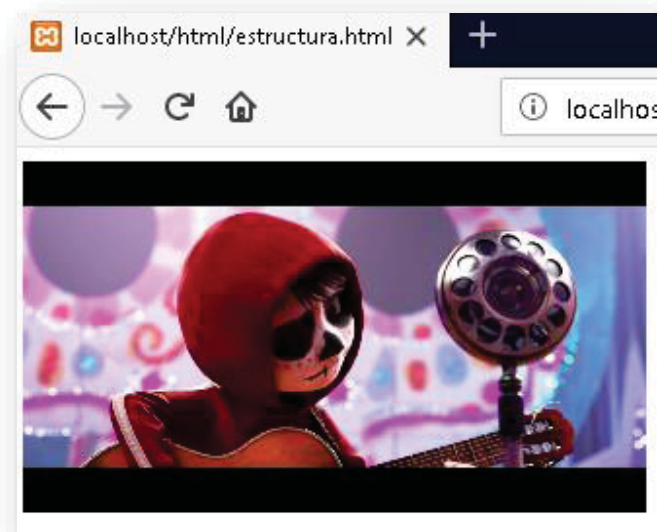


Ilustración 52 - Imagen propia, video tomado de <https://www.youtube.com/watch?v=yZ7cBunq8xo>

Otra forma de enlazar un video en una página web, es por medio de la etiqueta **<IFRAME>**, es un enlace en bloque que permite una visualización diferente de cualquier elemento en una página web, es decir, un **<IFRAME>** es una página web dentro de otra lo que permite doble contenido de diferente índole, pero lo interesante de esto es que el video se enlaza directamente de la plataforma **YOUTUBE**, y este se obtiene de la misma.

```
<iframe width="560" height="315" src="https://www.youtube.com/embed/yg8116aeD7E"
frameborder="0" allow="autoplay; encrypted-media" allowfullscreen></iframe>
```

Ilustración 53 - Imagen propia

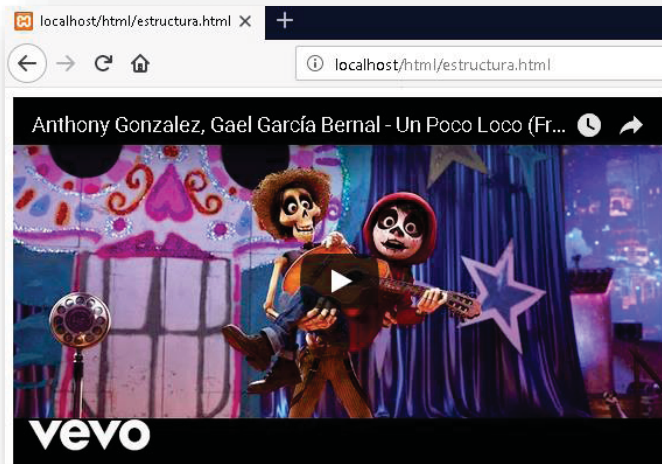


Ilustración 54 - Imagen propia, video tomado de <https://www.youtube.com/watch?v=yZ7cBunq8xo>

Así se es procedimiento:

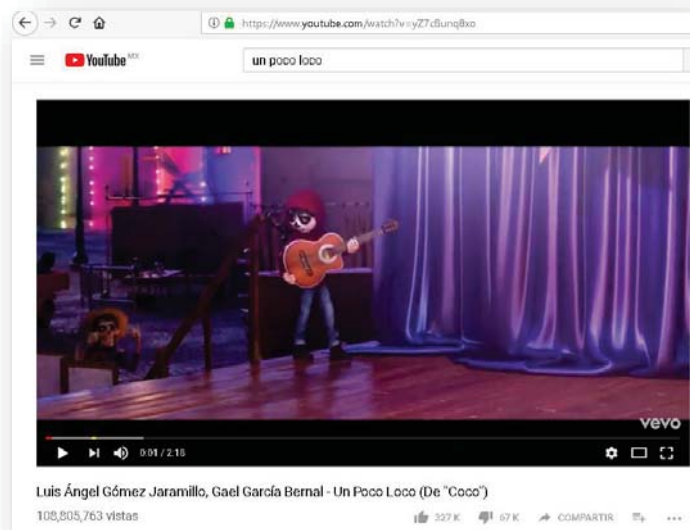


Ilustración 55 - Imagen propia, video tomado de <https://www.youtube.com/watch?v=yZ7cBunq8xo>

Se busca el video que se enlazara a la página web.



Ilustración 56 - Imagen propia

En la parte inferior derecha del video, encontraras unos enlaces donde se pueden realizar diferentes dinámicas con respecto al video, el enlace que interesa es el botón “Compartir”.

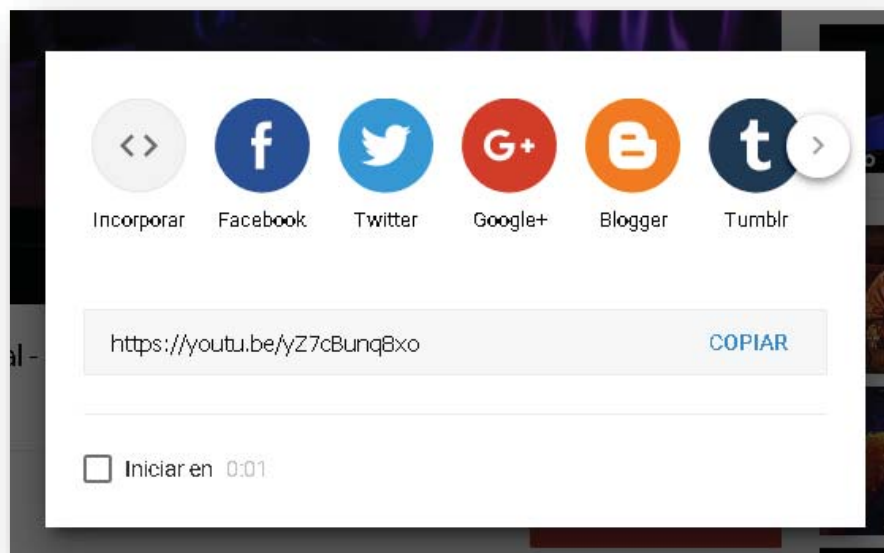


Ilustración 57 - Imagen propia

Cuando se da clic, aparece esta ventana, donde tenemos que seleccionar el enlace “Incorporar”, como se puede observar, se encuentran los corchetes que se utilizan para las etiquetas en HTML5.

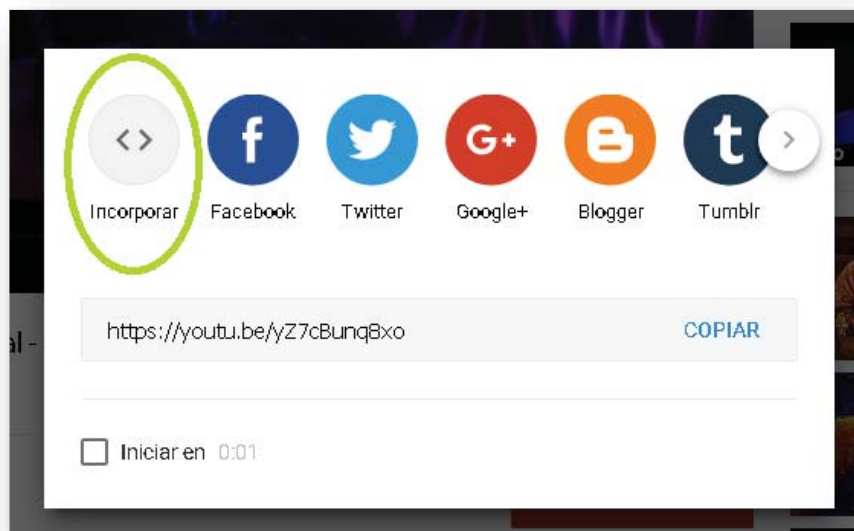


Ilustración 58 - Imagen propia

Ahora bien, cuando se selecciona, aparecerá el código que se necesita para que el video aparezca dentro de la página web, cabe recalcar que este código ya está configurado con los requisitos que necesita el video y los recursos para que pueda presentarse el video de manera natural.

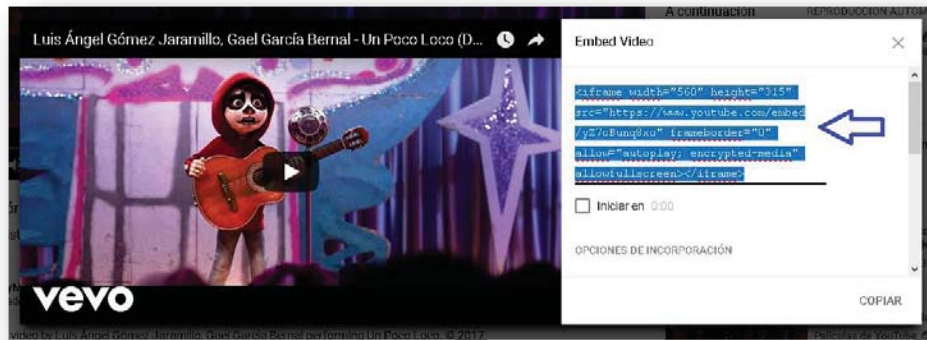


Ilustración 59 - Imagen propia, video tomado de <https://www.youtube.com/watch?v=yZ7cBunq8xo>

De esta manera, copiando el código y pegándolo en el editor de textos donde se encuentra tu código, aparecerá de esta forma.

```
<body>
  <iframe width="560" height="315" src="https://www.youtube.com/embed/yg8116aeD7E"
  frameborder="0" allow="autoplay; encrypted-media" allowfullscreen></iframe>
</body>
```

Ilustración 60 - Imagen propia

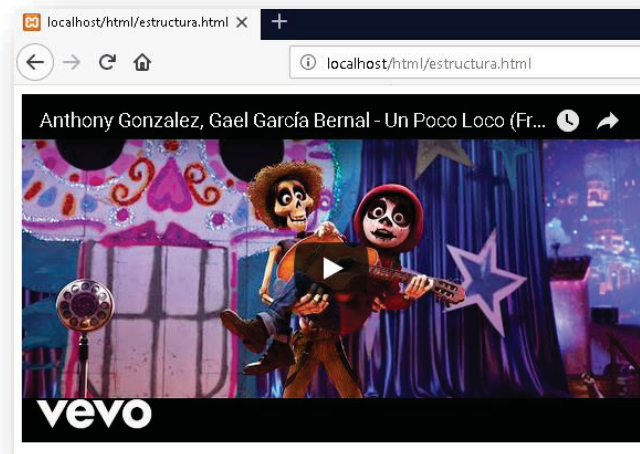


Ilustración 61 - Imagen propia, video tomado de <https://www.youtube.com/watch?v=yZ7cBunq8xo>

Otro tipo de imágenes que se pueden incluir dentro de **HTML5**, son los iconos, no son precisamente imágenes ilustrativas, pero si más indicadoras de lo que confiere a la acción de algún enlace, una lista o elementos que precisan de un modo más gráfico y descriptivo a su función, es decir, si existe un enlace que tiene como nombre “Buscar” se le puede anexar una imagen a escala representativa del nombre que contiene, por ejemplo, una “Lupa”.

Para poder entender un poco más sobre los iconos, estos son pequeñas imágenes vectoriales que permiten moldearse por medio de **CSS**, ya que permiten reducir o aumentar su tamaño, color, sombra, etc., sin perder calidad de imagen. Hoy en día un icono es más representativo para las nuevas generaciones, pues ahora se está aprendiendo y conociendo el mundo cibernético de manera visual, por tanto, es más funcional una imagen que una palabra o texto.

Ahora bien, para acceder a los iconos, es necesario anexar una biblioteca de iconos a **HTML5**, esto se anexa como una hoja de estilo de **CSS**, en la parte de **<HEAD>** se incluye un **<LINK>**, pero te preguntas ¿Qué es una biblioteca y cómo la vas a utilizar? Bueno, no te espantes, una biblioteca de iconos es el simple conjunto de todos ellos con su respectivo nombre, pues para poder solicitarlo dentro de **HTML5** debe ser por medio del nombre y obviamente con la biblioteca declarada dentro del código correctamente, dentro de la internet se pueden encontrar infinidad de sitios web que contengan bibliotecas de iconos, inclusive algunos con temática estacional o simplemente alusiva a algún evento en particular, esto brinda la oportunidad de escoger para complementar la página con respecto a su concepto.

En este caso se utilizarán los iconos del sitio FONT AWESOME, lo primero es teclear la página oficial.



Ilustración 62 - Imagen propia

Después se selecciona el enlace para comenzar a utilizar los iconos que nos ofrece de manera gratis, algo importante dentro de estos sitios es que tienen bibliotecas “free” y bibliotecas “pro”, esta última es de cobro, la diferencia entre ambas es que probablemente la biblioteca “pro” tenga mayor diversidad de iconos.

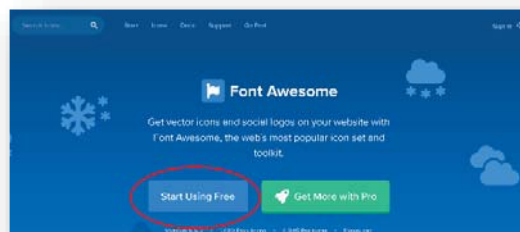


Ilustración 63 - Imagen propia

Se abrirá una nueva página donde nos arroja el link que se tiene que colocar dentro de **<HEAD>**.

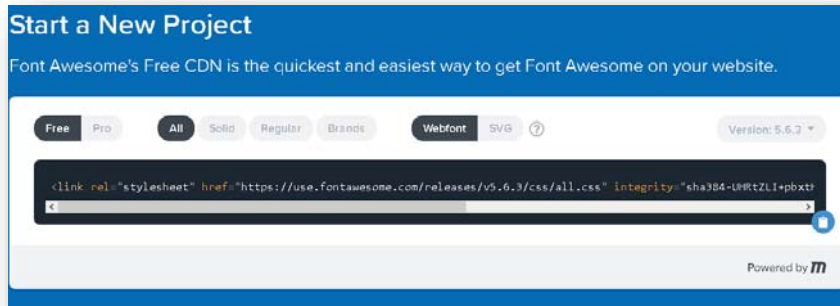


Ilustración 64 - Imagen propia

```
<head>
<meta charset="utf-8">
<link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.6.3/css/all.css" integrity="
sha384-UhRTzLI+pbxtH...>
<link rel="stylesheet" type="text/css" href="css/estilo.css">
</head>
```

Ilustración 65 - Imagen propia

De esta manera se puede acceder al repertorio de iconos que contenga este sitio, así que cuando se desee algún icono se va al buscador del sitio y se escribe lo que se busca, aparecerán todos los iconos relacionados a lo que se busca, por tanto, aparecerán iconos opacos, lo cual significa que es un icono de pago.

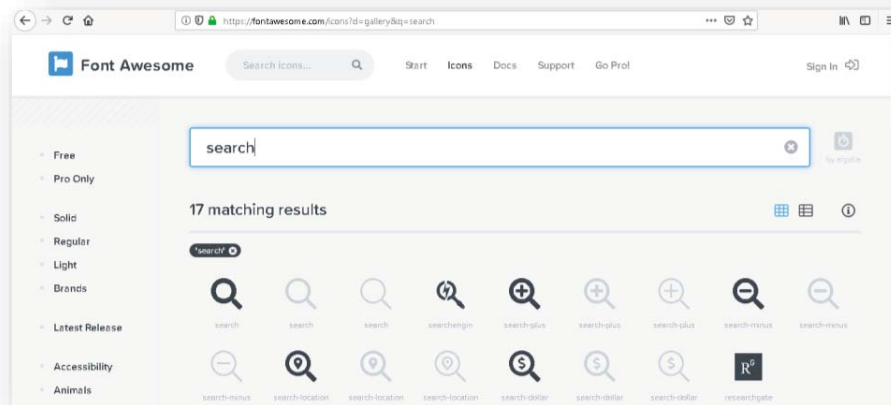


Ilustración 66 - Imagen propia

Pero te preguntarás ¿Cómo hago para que aparezcan en mi página web?, bueno para poder hacerlos visibles, se debe tener en cuenta que **HTML5** cuenta con dos tipos de etiquetas para anclar los iconos al elemento que se desea, que son **<I>** y ****, en este caso, el mismo sitio nos arroja la línea de código que se necesita para llamar a ese icono, solo basta con seleccionar el icono deseado y abrirá una nueva página.

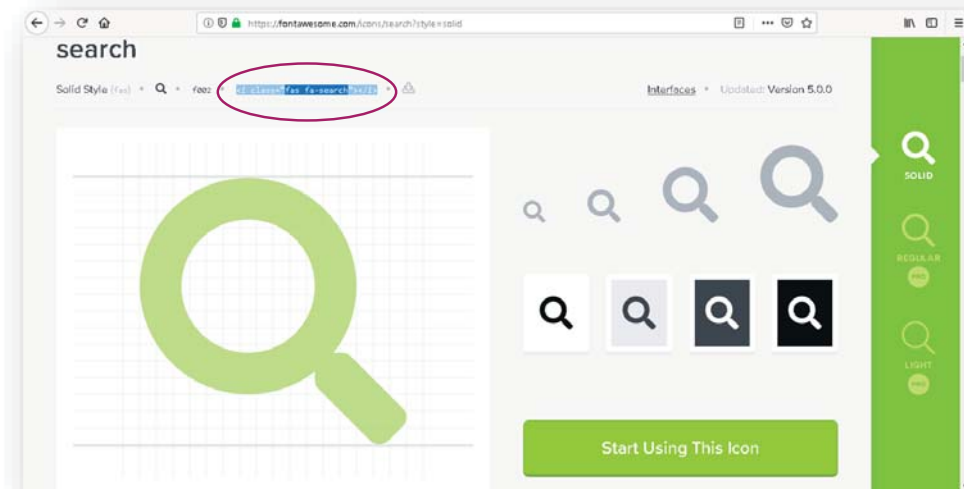


Ilustración 67 - Imagen propia

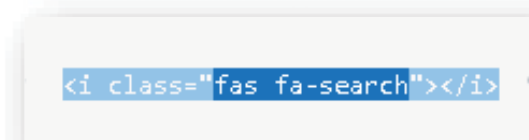


Ilustración 68 - Imagen propia

Y con esa línea de código se anexa al código que se tiene.

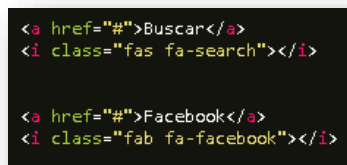


Ilustración 69 - Imagen propia

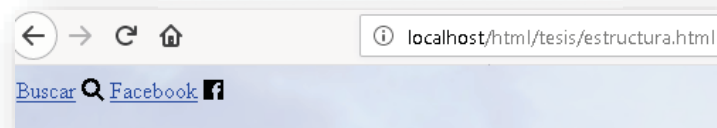


Ilustración 70 - Imagen propia

Solo como dato curioso, si observas la línea de código, esta contiene una clase y para aclarar la duda, esa clase no la puede contener el elemento **<A>**, en primera porque no es correcta la ubicación, segunda el navegador no lo identificara y tercer al momento de querer dar estilo por medio de **CSS** no se podrá hacer por separado, entonces ahí interviene un conflicto estético, no tanto de sintaxis, pero no sería lo ideal, esto sucede de igual manera con la etiqueta ****, pues habrá sitios donde arrojen alguna de las dos etiquetas con su respectiva clase.

Ahora bien, poder insertar un audio en la página web, se hace de similar forma que con un video, en este caso para llamar al elemento lo hacemos por medio de la etiqueta **<AUDIO>**, esta nos permite declarar una función similar a la de video, pues declarada esta etiqueta de **HTML5**, entiende que se está insertando un audio, dentro de esta etiqueta solicitamos la misma etiqueta de video como es **<SOURCE>** lo único que cambia es el atributo **SRC** por un audio con extensión mp3, ya que es la extensión más común entre audios y que en cualquier navegador lo puede soportar y leer.

```
<audio controls>
  <source src="wwwy.mp3" type="audio/mpeg">
</audio>
```

Ilustración 71 - Imagen propia, audio tomado de DVBBBS (WE WERE YOUNG)

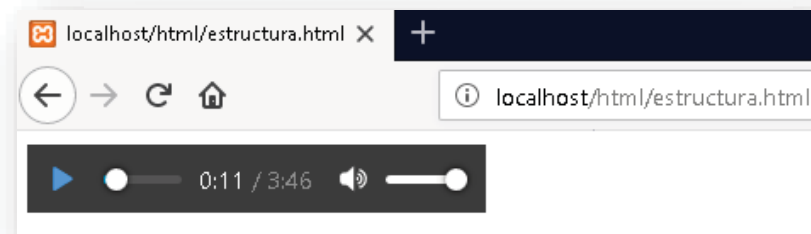


Ilustración 72 - Imagen propia, audio tomado de DVBBBS (WE WERE YOUNG)

HTML5 también cuenta con tablas que permiten organizar la información, estas pueden aplicarse con la etiqueta **<TABLE>**, dentro de esta etiqueta tendrá filas las cuales se declaran con la etiqueta **<TR>**, y para los encabezados de cada fila, se utiliza la etiqueta **<TH>** lo cual permite que se le identifique por las palabras en negritas que da por default la etiqueta, por otro lado, se tiene la etiqueta **<TD>**, con esta última, se concluye el cuerpo de una tabla, pues su objetivo es declarar las celdas de cada fila.

```
<body>
  <table>
    <tr>
      <th>Nombre</th>
      <th>Semestre</th>
      <th>No. Cuenta</th>
    </tr>
    <tr>
      <td>Rosa Aguilar</td>
      <td>4</td>
      <td>384956832</td>
    </tr>
    <tr>
      <td>Ricardo Cordoba</td>
      <td>8</td>
      <td>234521345</td>
    </tr>
  </table>
</body>
```

Ilustración 73 - Imagen propia

Nombre	Semestre	No. Cuenta
Rosa Aguilar	4	384956832
Ricardo Cordoba	8	234521345

Ilustración 74 - Imagen propia

De esa manera se visualizaría una tabla en **HTML5**, sin embargo, existen diferentes formas de adaptar mejor una tabla para que luzca más presentable, cuando nos encontramos con más datos que van dentro de un requisito o parámetro de la tabla, lo recomendable es que se realice un combo de datos, en este caso existe el atributo **COLSPAN**.

```

<table>
  <tr>
    <th>Nombre</th>
    <th>Semestre</th>
    <th>No. Cuenta</th>
    <th colspan="2">Telefono</th>
  </tr>
  <tr>
    <td>Abril Irais Alba Villa</td>
    <td>9</td>
    <td>414060779</td>
    <td>58611186</td>
    <td>5519592651</td>
  </tr>

```

Ilustración 75 - Imagen propia

Nombre	Semestre	No. Cuenta	Telefono
Abril Irais Alba Villa	9	414060779	58611186 5519592651
Juan Francisco Islas Aguilar	5	309104726	

Ilustración 76 - Imagen propia

De esta manera, **HTML5** permite guardar más de un dato en una sola columna, también existe la manera de crear un combo con las filas, de igual forma se puede realizar este resultado, pero verticalmente, afectando a las filas. Para esto se utiliza el atributo **ROWSPAN**.

```

<table>
  <tr>
    <th>Nombre</th>
    <td>Abril Irais Alba Villa</td>
  </tr>
  <tr>
    <th>Semestre</th>
    <td>9</td>
  </tr>
  <tr>
    <th>No. Cuenta</th>
    <td>414060779</td>
  </tr>
  <tr>
    <th rowspan="2">Telefono</th>
    <td>58611186</td>
  </tr>
  <tr>
    <td>5519592651</td>
  </tr>
</table>

```

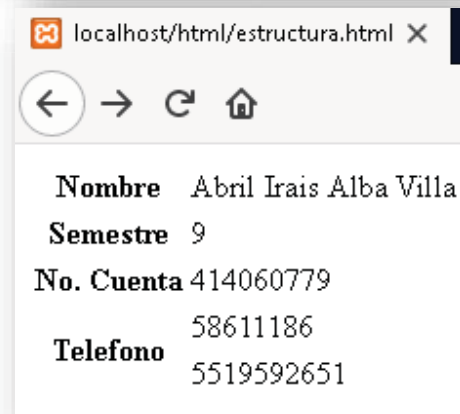


Ilustración 78 - Imagen propia

Se puede añadir una leyenda o un título a la tabla para complementarla aún más. Esto lo podemos lograr con la etiqueta **<CAPTION>**, esta etiqueta se debe colocar siempre después de la etiqueta **<TABLE>**, pues si no se declara así puede ser un error de sintaxis.

```

<table>
  <caption>Datos de Alumnos</caption>
  <tr>
    <th>Nombre</th>
    <td>Abril Irais Alba Villa</td>
  </tr>

```

Ilustración 79 - Imagen propia

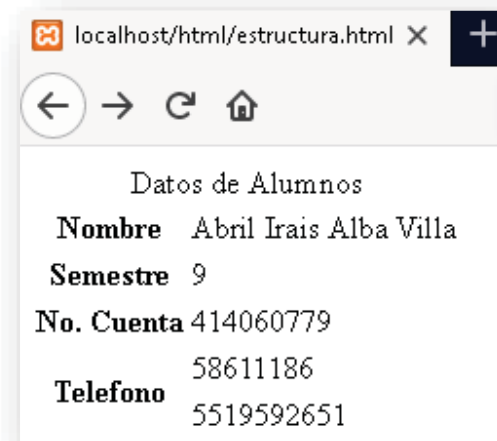


Ilustración 80 - Imagen propia

Al igual que las tablas para mantener un orden de datos dentro de ellas, también existen listas que nos permiten hacer una organización de manera ordenada o desordenada, para ello se utiliza la etiqueta `` el objetivo de esta etiqueta es crear una lista desordenada, cada elemento de la lista se distinguirá por viñetas y se declararán con la siguiente etiqueta ``.

```
<body>
<ul>
  <li>Diseño</li>
  <li>Desarrollo</li>
  <li>Dinamismo</li>
  <li>Optimización</li>
</ul>
</body>
```

Ilustración 81 - Imagen propia

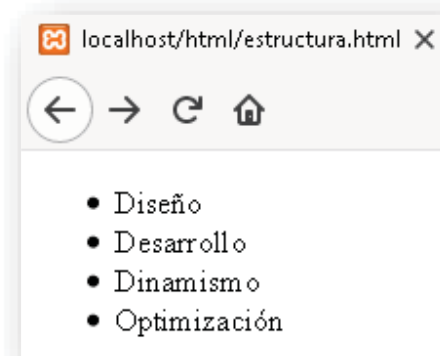


Ilustración 82 - Imagen propia

Y las listas ordenadas comienzan con la etiqueta ``, de esta manera la lista se organizará jerárquicamente, dependiendo de lo que el diseñador determine al igual que en las listas

desordenadas. También se puede agregar el atributo **START**, solamente a las listas ordenadas ya que se permite elegir de que numero comience a contar los elementos de la lista.

```
<body>
  <ol>
    <li>Planeacion</li>
    <li>Desarrollo</li>
    <li>Diseño</li>
    <li>Integración</li>
  </ol>
</body>
```

Ilustración 83 - Imagen propia

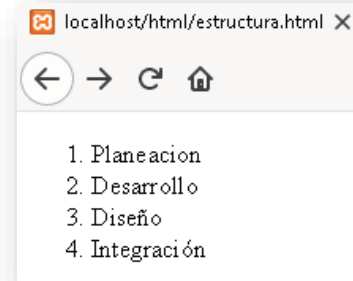


Ilustración 84 - Imagen propia

```
<body>
  <ul>
    <li>Imagenes</li>
    <ol start="10">
      <li>PNG</li>
      <li>JPEG</li>
      <li>JPG</li>
      <li>GIF</li>
    </ol>
  </ul>
</body>
```

Ilustración 85 - Imagen propia

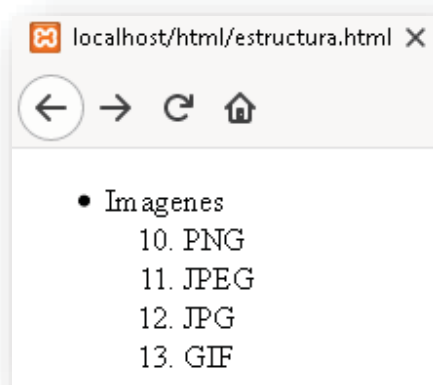
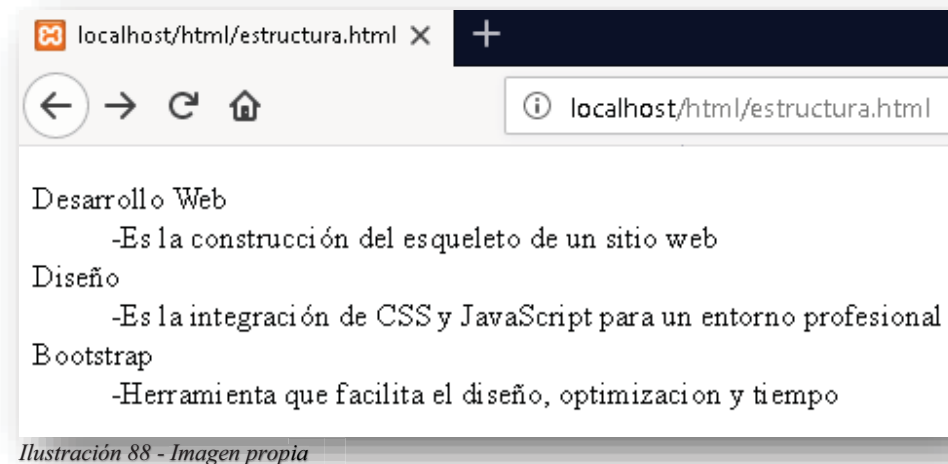


Ilustración 86 - Imagen propia

Las listas contienen al igual que las tablas una descripción, la diferencia es que en las listas se pueden describir todos los términos de la lista, para crearla, se comienza con la etiqueta **<DL>** su objetivo es declarar una lista descriptiva y para sus elementos se declara el título dentro de la lista con la etiqueta **<DT>** y la descripción con la etiqueta **<DD>**.

```
<body>
  <dl>
    <dt>Desarrollo Web</dt>
    <dd>-Es la construcción del esqueleto de un sitio web</dd>
    <dt>Diseño</dt>
    <dd>-Es la integración de CSS y JavaScript para un entorno profesional</dd>
    <dt>Bootstrap</dt>
    <dd>-Herramienta que facilita el diseño, optimización y tiempo</dd>
  </dl>
</body>
```

Ilustración 87 - Imagen propia



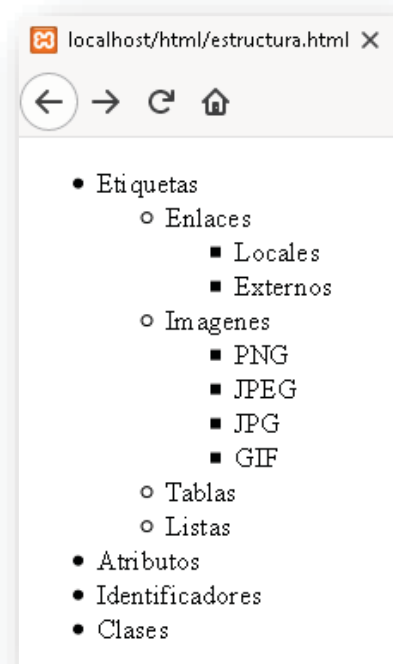
Las listas también se pueden anidar dependiendo de la información que se tenga que mostrar en un sitio web.

```

<body>
  <ul>
    <li>Etiquetas</li>
    <ul>
      <li>Enlaces</li>
      <ul>
        <li>Locales</li>
        <li>Externos</li>
      </ul>
      <li>Imágenes</li>
      <ul>
        <li>PNG</li>
        <li>JPEG</li>
        <li>JPG</li>
        <li>GIF</li>
      </ul>
      <li>Tablas</li>
      <li>Listas</li>
    </ul>
    <li>Atributos</li>
    <li>Identificadores</li>
    <li>Clases</li>
  </ul>
</body>

```

Ilustración 89 - Imagen propia



Cabe mencionar que cuando se crea una tabla o una lista, estos son elementos demasiado pesados de leer, porque su contenido son solo palabras, por eso a estos elementos se les puede agregar imágenes, enlaces, iconos llamativos o elementos que permitan digerir de manera más rápida y satisfactoria la información que se presenta.

Las etiquetas más comunes que podemos encontrar en **HTML5**, son todas aquellas que describen y muestran pequeños detalles que pasan desapercibidos pero que son importantes conocer. Dentro del parámetro párrafos, se tiene la etiqueta **<P>** ya antes mencionada, pero dentro de esta etiqueta existen otras que le brindan un formato diferente a un párrafo, la etiqueta **<PRE>** tiene por objetivo dar un formato al texto y representarlo tal cual se escribe en el código, pues de ninguna manera eso podría ocurrir con **<P>** porque su función es otra.

```
<body>
  <p>Lorem ipsum dolor sit amet,
    consectetur adipisicing elit,
    sed do eiusmod tempor incididunt
    ut labore et dolore magna aliqua.
  </p>
  <pre>
  Lorem ipsum dolor sit amet,
    consectetur adipisicing elit,
    sed do eiusmod tempor incididunt
    ut labore et dolore magna aliqua.
  </pre>
</body>
```

Ilustración 91 - Imagen propia

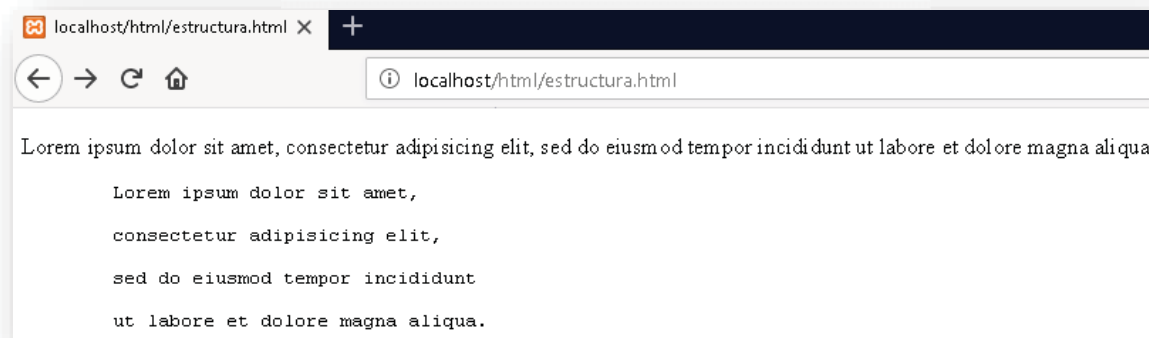


Ilustración 92 - Imagen propia

Se puede observar que la misma etiqueta **<PRE>** le brinda una fuente diferente a la letra en comparación a **<P>**, pero para esta última etiqueta, podemos encontrar otra etiqueta que permita hacer saltos de línea para evitar que el texto se vaya de corrido por la página.

```

<body>
  <p>Lorem ipsum dolor sit amet, <br>
    consectetur adipiscing elit, <br>
    sed do eiusmod tempor incididunt <br>
    ut labore et dolore magna aliqua. <br>
  </p>
</body>

```

Ilustración 93 - Imagen propia

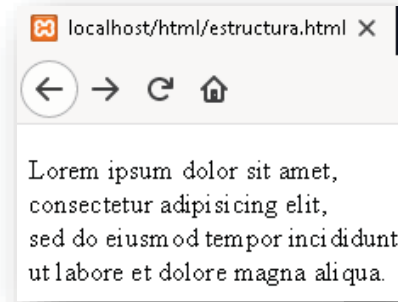


Ilustración 94 - Imagen propia

Existen muchos formatos que se le pueden dar a los textos que contenga la página web, dentro de estas etiquetas, permiten resaltar aquello que es importante o de lo contrario hacer ver que elemento del texto no es relevante. Dentro de ellas se pueden encontrar las etiquetas: **** Marca el texto en negrita, **** Marca el texto importante, **<I>** Marca el texto en cursiva, **** Hace enfatizado el texto, **<MARK>** Realiza un texto marcado, **<SMALL>** Convierte en pequeño el texto, **** Muestra el texto eliminado, **<INS>** Subraya el texto, **<SUB>** Declara el texto en subíndice, **<SUP>** Declara el texto en superíndice.

```

<body>
  <p>Lorem <b>ipsum dolor</b> sit amet, <br>
    consectetur <strong>adipiscing</strong> elit, <br>
    sed <i>do</i> eiusmod <em>tempor</em> incididunt <br>
    ut <mark>labore</mark> et <small>dolore</small> magna
    <del>aliqua.</del> Ut <ins>enim ad</ins> minim <br>
    veniam, <sub>quis</sub> nostrud <sup>exercitation</sup> <br>
  </p>
</body>

```

Ilustración 95 - Imagen propia

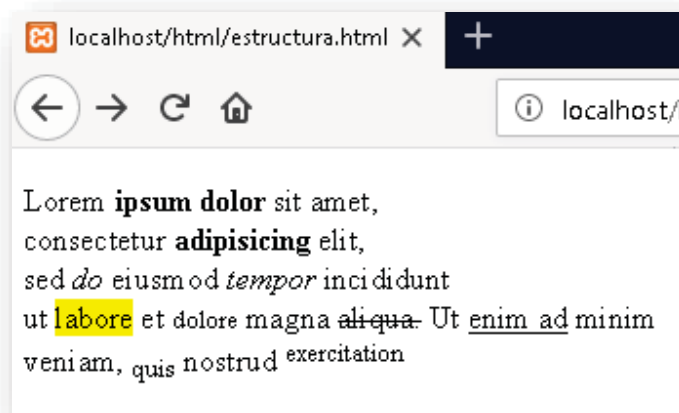


Ilustración 96 - Imagen propia

Para realizar citas dentro del texto o en determinado espacio del sitio web, existen etiquetas que facilitan ese tipo de tareas, como por ejemplo **<Q>** es una etiqueta que interpreta una pequeña cita dentro del texto con comillas dobles.

```
<p>Lorem ipsum dolor sit amet, <q>consectetur <br>
adipiscing elit</q>, sed do eiusmod tempor <br>
incididunt ut labore et dolore magna aliqua.<br>
</p>
```

Ilustración 97 - Imagen propia

Lorem ipsum dolor sit amet, “consectetur
adipiscing elit”, sed do eiusmod tempor
incididunt ut labore et dolore magna aliqua.

Ilustración 98 - Imagen propia

La etiqueta **<BLOCKQUOTE>** hace una cita de referencia de la otra fuente de información de donde provienen los datos.

```
<p>
<blockquote cite="https://www.google.com">Ut enim ad minim veniam, quis
nostrud <br>
exercitation ullamco laboris nisi ut <br>
aliquip ex ea commodo consequat.</blockquote> <br>
</p>
```

Ilustración 99 - Imagen propia

Lorem ipsum dolor sit amet, “consectetur
adipiscing elit”, sed do eiusmod tempor
incididunt ut labore et dolore magna aliqua.

Ut enim ad minim veniam, quis nostrud
exercitation ullamco laboris nisi ut
aliquip ex ea commodo consequat.

Colonia Valle de las flores

Ilustración 100 - Imagen propia

La etiqueta **<ABBR>** tiene la función de englobar una abreviatura que, al posar el apuntador sobre la abreviatura, mostrara su significado en una etiqueta emergente.

```
<abbr title="WORLD WIDE WEB (RED INFORMATICA MUNDIAL)">WWW</abbr>
```

Ilustración 101 - Imagen propia



Ilustración 102 - Imagen propia

La etiqueta **<ADDRESS>** por su parte, tiene como objetivo resaltar un texto de manera sutil con una dirección de domicilio, o cualquier texto que contenga esta etiqueta.

```
<address>  
Colonia Valle de las flores<br>  
Calle Jacarnadas <br>  
Numero 320 Interior 3<br>  
Cuautitlan México  
</address>
```

Ilustración 103 - Imagen propia

*Colonia Valle de las flores
Calle Jacarnadas
Numero 320 Interior 3
Cuautitlan México*

Ilustración 104 - Imagen propia

La etiqueta **<CITE>** se encarga de marcar una parte del texto que sea importante de manera cursiva.

```
<p>  
<cite>Duis aute irure dolor</cite> 1345 in reprehenderit in<br>  
voluptate velit esse cillum dolore eu <br>  
fugiat nulla pariatur. Excepteur sint<br>  
occaecat cupidatat non proident, sunt in <br>  
</p>
```

Ilustración 105 - Imagen propia

*Duis aute irure dolor 1345 in reprehenderit in
voluptate velit esse cillum dolore eu
fugiat nulla pariatur. Excepteur sint
occaecat cupidatat non proident, sunt in*

Ilustración 106 - Imagen propia

Por último, se encuentra una etiqueta **<BDO>** con su respectivo atributo, la cual hace que el texto de invierta, puede utilizarse de manera seria o creativa en un sitio web.

```
<p>  
<bdo dir="rtl">culpa qui officia deserunt mollit anim id est laborum.</  
bdo>  
</p>
```

Ilustración 107 - Imagen propia

```
.murobal tse di mina tillom tnuresed aiciffo iuq apluc
```

Ilustración 108 - Imagen propia

Dentro de las etiquetas existen otro tipo de atributos que nos posibilitan nombrar a los elementos para tareas futuras, es decir, nos permiten llamarlas en alguna hoja de estilo **CSS** o en una función de **JS**. Con esto estamos hablando de las pseudoclasas **ID** y **CLASS**, una pseudoclase **ID**, es un identificador único de elementos **HTML5**, este atributo solo puede pertenecer a una etiqueta, pues si existen otros identificadores con el mismo nombre, el código no se compilará de manera adecuada y existirán errores de sintaxis.

```
<p>
  <h1 id="Header">IDENTIFICADORES</h1>
</p>
```

Ilustración 109 - Imagen propia



Ilustración 110 - Imagen propia

Para que un identificador tenga la funcionalidad que le corresponde, tenemos que llamarlo en una hoja de estilo, la forma correcta es el carácter (#) acompañado del nombre del identificador.

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <link rel="stylesheet" type="text/css" href="css/estilo.css">
</head>
<body>
  <p>
    <h1 id="Header">IDENTIFICADORES</h1>
  </p>
</body>
</html>
```

Ilustración 111 - Imagen propia

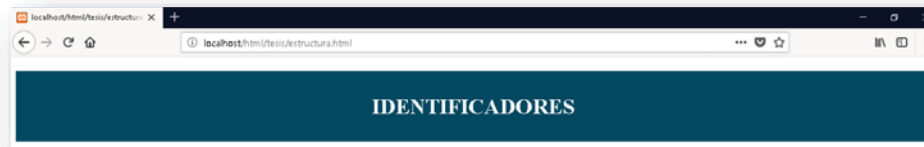


Ilustración 112 - Imagen propia

Y para JS se necesita citarlo de esta manera se necesita el método “`getElementById()`”.

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <link rel="stylesheet" type="text/css" href="css/estilo.css">
</head>
<body>

  <p>

    <h1 id="Header">IDENTIFICADORES</h1>
    <button onclick="displayResult()">Acción</button>

  </p>

  <script>
    function displayResult() {
      document.getElementById("Header").innerHTML = "Presente!!!";
    }
  </script>

</body>
</html>
```

Ilustración 113 - Imagen propia

Es una manera sencilla de representar un JS, pero puede adaptarse de acuerdo a las necesidades y requerimientos del sitio web.

Ahora que se entiende el funcionamiento de **ID** y como se puede trabajar, se puede continuar con **CLASS**, es muy similar el trabajo que se practica con **ID**, pero la diferencia entre ambos es que **CLASS** puede especificar a uno o más elementos. Y para ser citado en una hoja de estilo CSS, lleva un (.) antes del nombre de la clase.

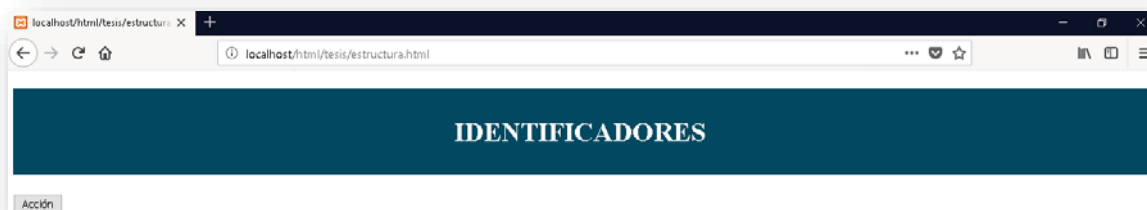


Ilustración 114 - Imagen propia

Como se puede ver, en las siguientes imágenes, existen cuatro elementos que son nombrados con el mismo nombre y al momento de editar con una hoja de estilo, altera a los 4 elementos de la misma manera. Aparte de entender que un **ID** (identifica un elemento único) y un **CLASS** (nombra a uno o varios elementos). Ahora bien, para citar en un **JS** se necesita el método `getElementsByClassName()`.

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <link rel="stylesheet" type="text/css" href="css/estilo.css">
</head>
<body>

  <div class="img">
    
  </div>
  <div class="img">
    
  </div>
  <div class="img">
    
  </div>
  <div class="img">
    
  </div>

</body>
</html>
```

Ilustración 115 - Imagen propia

```
.img{
margin-left: 0px;
margin-top: 00px;
display: inline flex;
float: left;
padding: 20px;
background-color: rgb(195,195,195);
height: 50px;
width: 50px;
text-align: center;
border: 5px solid white;
}
```

Ilustración 116 - Imagen propia

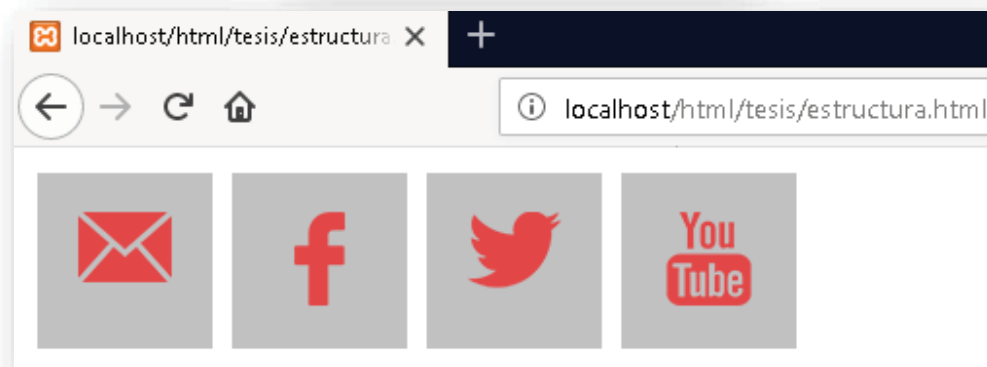


Ilustración 117 - Imagen propia

```

<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <link rel="stylesheet" type="text/css" href="css/estilo.css">
</head>
<body>
  <button onclick="myFunction()">Sombra</button>

  <script>
    function myFunction() {
      var x = document.getElementsByClassName("img main");
      x[0].style.backgroundColor = "black";
    }
  </script>

  <div class="img">
    
  </div>
  <div class="img">
    
  </div>
  <div class="img main">
    
  </div>
  <div class="img">
    
  </div>

</body>
</html>

```

Ilustración 118 - Imagen propia

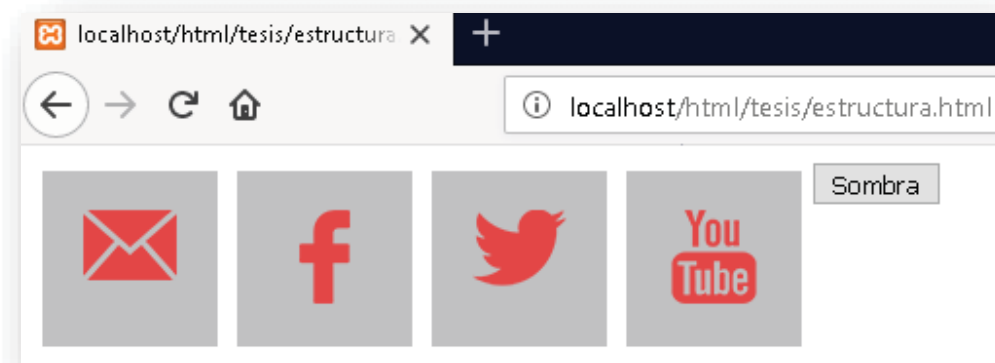


Ilustración 119 - Imagen propia

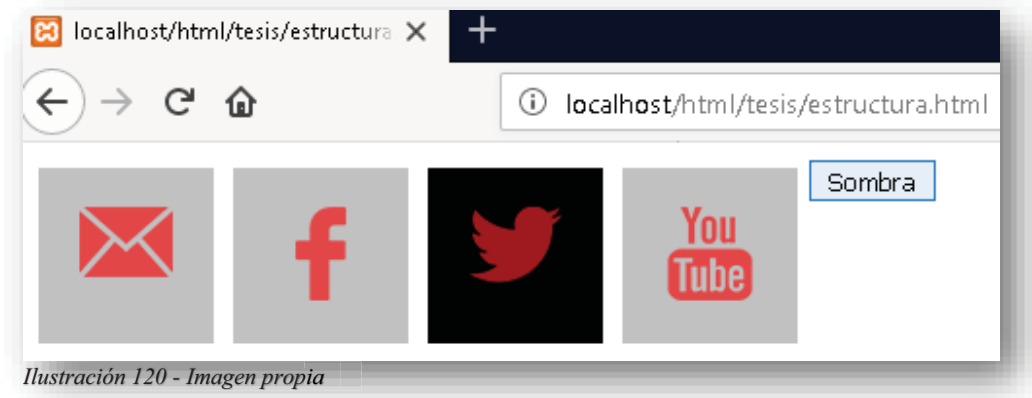


Ilustración 120 - Imagen propia

También se puede observar el uso de una nueva etiqueta que es **<DIV>**, su objetivo es realizar contenedores dentro de **HTML5** y permite editarse con **CSS** o generar funciones con **JS**, lo que permite hacer separaciones o saltos de línea entre los elementos y moldear y/o alterar de forma “Responsive” la página web, es decir, que la página web puede ajustarse a los dispositivos móviles sin perder su diseño.

Si hablamos de **HTML5** y todo lo que puede abarcar con sus elementos, nunca se acabaría este tema, pues existe una infinidad de etiquetas que buscan resolver un problema en específico y aún más infinito es el número de atributos, hasta este momento el aprendizaje que se obtuvo es la base suficiente para poder brincar al siguiente escalón, ahora puedes darte cuenta que **HTML5** sirve para crear un esqueleto de tu página web de manera sencilla y con los elementos correctos para que sea autosuficiente pero sobre todo que tenga la información adecuada para nutrir la página.

7.4. ESTRUCTURA SEMANTICA

Cuando hablamos de una estructura semántica, no es más que la descripción de las etiquetas que con su propio nombre dicen lo que tienen por objetivo, con esto obtenemos el diseño de una página web, pues se dedica aparte de hacer más claro el código, también organiza de una manera más ordenada y distribuida los elementos que se tienen en el código, por supuesto esto ayuda a que el usuario final tenga una mayor facilidad de entender la información que se le está presentando en el sitio web.

Ahora bien, cuando se estructura una página web, se entendió que existe una cabecera y un cuerpo, estamos bien hasta el momento, ya aprendimos a desarrollarlo, pero existen hoy en día más elementos y espacios de un sitio web que son utilizados y explotados al máximo, pues las demandas de los sitios web dentro de internet tienen que evolucionar e innovar.

Para lograr una página del estilo que hoy conocemos y para desarrollarlo en **HTML5** nada más, hacemos uso de la siguiente estructura.

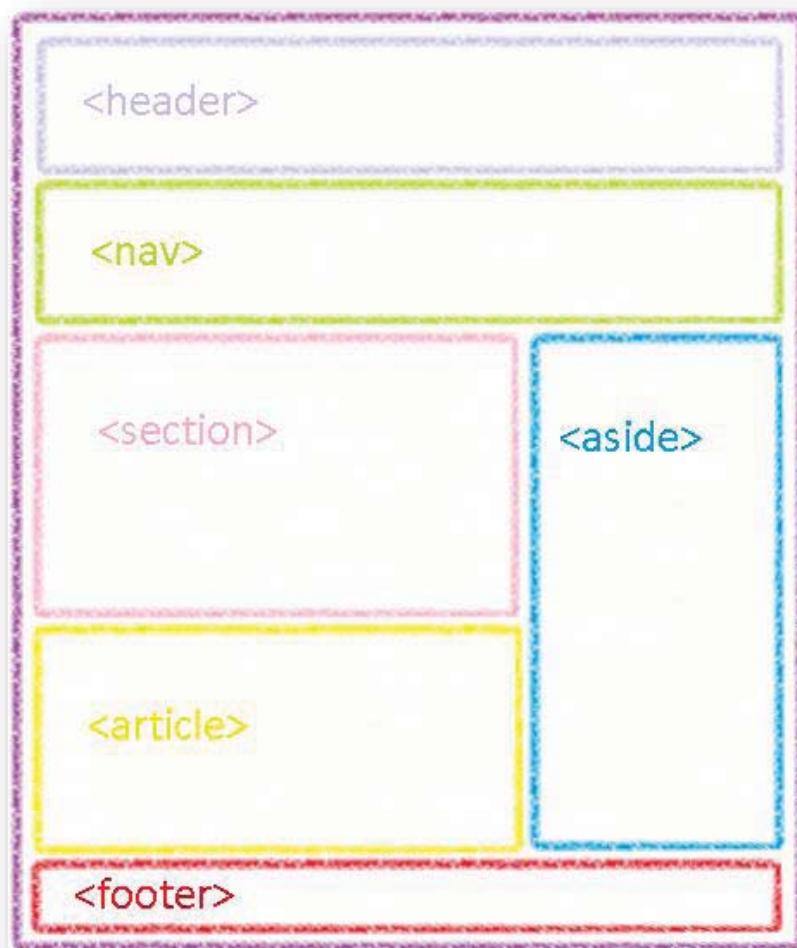


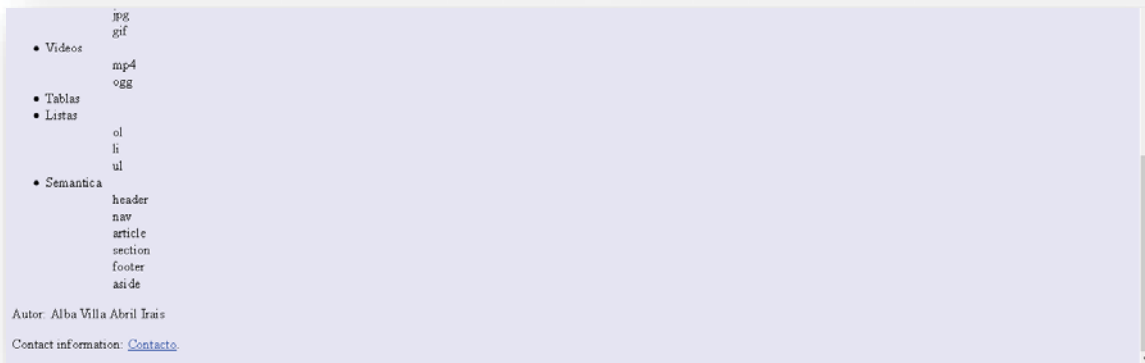
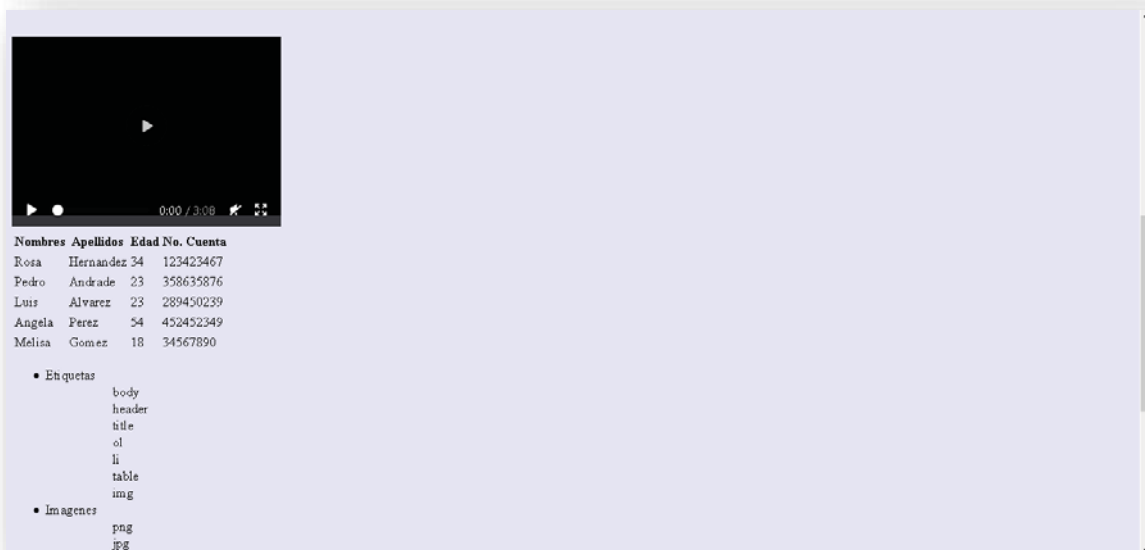
Ilustración 121 - Imagen propia

La etiqueta **<HEADER>** es el elemento que especifica una cabecera, **<NAV>** es el elemento que contiene los enlaces de una página, normalmente se maneja como un menú, **<SECTION>** es el elemento que se dedica a desarrollar el contenido de la página, aquí se pueden involucrar todos los elementos que conocimos anteriormente, **<ARTICLE>** es el elemento que engloba información independiente a la página web, posiblemente para sitios de debate, cuestión de comentarios, e incluso se puede involucrar artículos publicitarios, **<ASIDE>** es un elemento versátil, puede mantenerse como un segundo menú, también se puede incluir el tema de la multimedia para una mayor organización, **<FOOTER>** es elemento que especifica el pie de la página y el que contiene la información importante de la página, como puede ser el autor de la página, enlaces de contacto, términos y usos del contenido, etc.

La organización que nos brinda la estructura semántica tiene un objetivo principal, clave y sumamente importante, cuando se creaban o desarrollaban páginas web, en **HTML4** se utilizaban los identificadores y las clases para poder ordenar los elementos correspondientes a la distribución de la página web, por ende, esto representaba bastante trabajo, mucho código y posiblemente muchos errores para colocarlos donde se debían con ayuda de **CSS**. Con estos y otros problemas de antigüedad **HTML5** se reinventó con esta estructura que ayudó a formar de manera más limpia y corta una página.

Dentro de estos nuevos elementos se tiene la etiqueta **<FIGURE>** que permite adjuntar un pie de imagen y **<FIGCAPTION>** tiene por objetivo encabezar una imagen, **<DETAILS>** define detalles adicionales que el usuario puede ver u ocultar, **<MAIN>** tiene por objetivo generar un espacio para un menú, pero regularmente se utiliza **<NAV>** para esos casos, aunque no es menos importante, puede ocuparse de igual manera, **<MARK>** este elemento se utiliza para resaltar algún tipo de texto, párrafo, frase o palabra dentro del contenido, teniendo básicamente una tarea similar a las etiquetas de texto que se mencionaron anteriormente en los tipos de texto, **<SUMMARY>** define un encabezado en la etiqueta **<DETAILS>** pero esto pertenece un poco más al desarrollador que al usuario final.

Existen también otras etiquetas que permiten pulir detalles dentro de la página web, como **<DIALOG>** que permite hacer un diálogo o una ventana, **<TIME>** ofrece la facilidad de enlazar la hora en la página web, **<WBR>** realiza saltos de línea sobre el texto.



8. INTRODUCCIÓN AL DISEÑO WEB CSS

Así como se leyó en **HTML5**, el desarrollo web evolucionó para mejorar todos sus aspectos, de este modo se creó y adaptó un elemento externo que permitió resolver un problema que apareció en esa época, pues **HTML**, no se podía permitir cambiar a un estilo por sí solo, pues su objetivo solo es estructurar una página web, y por ende la presentación como no era la ideal para el público, pues su presencia estética era nula, después apareció la etiqueta de estilo para que dentro de las etiquetas se permitiera ejercer cambios limitados como el tamaño y color del elemento, siendo una aplicación muy pobre para **HTML** surgió la idea de crear hojas de estilo, que permitía que se aplicaran en los distintos archivos electrónicos del momento.

Cuando **CSS** apareció en el ámbito de desarrollo web, causó un gran impacto, pues mostró algo diferente, una presencia gráfica, aunque ya existía una solución surgieron muchos problemas con los recursos de los navegadores web, ya que el soporte que se le brindaba era prácticamente nulo al que tenía **HTML** en ese momento, ahora con los avances y evoluciones de **HTML**, **CSS** se vio con un camino libre para también progresar y hacer cambios que mejorarían por mucho a su predecesor como fue **CSS nivel 1**, **CSS nivel 2**, **CSS2.1**, hasta el que tenemos hoy en día que es **CSS3**, que para su tiempo de presentación al mundo, tuvo un cambio dramático y positivo para el Diseño Web. (Uniwebsidad, 2018-2019)

Las hojas de estilo en cascada a traducción de (Cascading Style Sheets), ha cambiado la vida de un diseñador web positivamente, pues cuando se habla de efectos especiales, fuentes externas, bordes, colores y formas, eran un gran dolor de cabeza, ya que el trabajo que generaba declarar los elementos de **HTML** e incluir el estilo con **CSS** y cargarlos en un navegador web era sumamente pesado, porque al subirlo, el tráfico de datos en la red era demasiado para un espacio tan pequeño e insuficiente, por ese motivo también tuvo que evolucionar lo que conocemos hoy en día como **INTERNET** y en conjunto con **HTML** y **CSS** con el tiempo adaptaron sus recursos para tener un desempeño del Desarrollo Web al máximo. (W3C, 2018-2019)

8.1.CSS

Hoy en día disfrutamos de páginas que tienen funcionalidad, innovación, tecnología y presencia para el usuario final, con esto cabe destacar que los navegadores web que se tienen como: Firefox, Chrome, Safari y Opera, gozan de todas las propiedades, recursos y soportes para los diferentes sitios web que se suban en red, su funcionalidad ahora ya no tiene problemas, ya que la evolución y corrección de errores se quedaron en el pasado.

Ahora bien, **CSS** es el formato recomendado por default para las páginas escritas en **HTML5**, ya que su uso permite un mejor diseño y permite un trabajo con mejor apariencia. Se conoce como un lenguaje de diseño gráfico que permite describir la presentación de páginas web. Lo que posibilita adaptar la presentación a diferentes tipos de dispositivos, como pantallas grandes, pantallas pequeñas o móviles y tabletas. **CSS** es independiente de **HTML** y se puede usar con cualquier lenguaje de marcado basado en XML. La separación de **HTML** de **CSS** facilita el mantenimiento de sitios, el intercambio de hojas de estilo entre páginas y la personalización de páginas en diferentes entornos.

8.2. ESTRUCTURA

Anteriormente en **HTML5** se mencionaba que un cambio de estilo, se podía insertar dentro del mismo **HTML5** sin ningún problema en **<HEAD>** o en las mismas etiquetas y es algo realmente cierto, el único impedimento para llevarlo a cabo es cuando el desarrollo de **HTML5** es muy extenso, eso mismo ocurre en **CSS**, es decir, si tu código es muy ambicioso y por ende tiene demasiadas líneas de código, entonces se corre el riesgo de tener un error de estética, porque, así como un código de **HTML5** se cuida su indentación, sus espacios y las declaraciones de etiquetas de manera correcta, también se cuida el hecho de no revolver códigos de un lenguaje con otro, el ideal es crear una etiqueta **<LINK>** externos dentro de **<HEAD>** para enlazar cualquier archivo que modifique **HTML5**, esto sirve para que el código tenga una mejor vista y sea más práctico de entender. Pero también se corre el riesgo que, si se tiene estilos de las tres diferentes formas, **HTML5** se registrará por el estilo que existe dentro de un elemento o etiqueta, pues para **HTML5** tiene prioridad por sus elementos que por algo externo o implícito en el código.

Teniendo en claro lo anterior, ahora se conocerá la manera de desarrollar un Front-End, retomando un poco este paso para no olvidarlo, al enlazar un estilo **CSS** a un archivo **HTML5**, lo realizamos de la siguiente manera.

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <link rel="stylesheet" type="text/css" href="css/estilo.css">
</head>
```

Ilustración 125 - Imagen propia

Con la etiqueta **<LINK>** se enlaza el archivo externo con extensión “.CSS”, de esta manera se le está especificando a **HTML5** que se adjuntó un archivo externo de tipo **CSS**, de esta forma se podrá describir y presentar de manera diferente en pantalla para el usuario. **CSS** se utiliza para definir el diseño, el estilo, las variaciones de tamaño a los elementos que se están afectando en **HTML5**.

Ahora bien, para trabajar dentro del documento **CSS** y poder aprender lo que hace, se alterará el documento **HTML5** que se mostró en el capítulo anterior. De principio es muy importante saber cómo se compone la sintaxis de **CSS**:

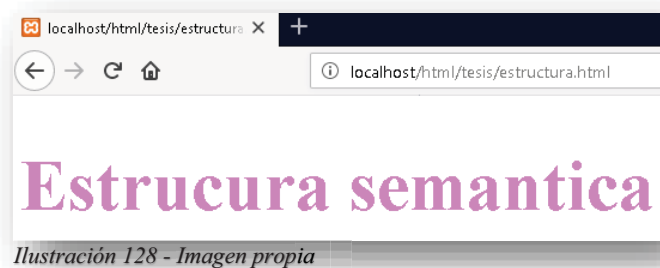
```
h1{
  color:violet;
  font-size: 60px;
}
```

Ilustración 126 - Imagen propia

La primera línea de código funge como “Selector”, es decir, **CSS** identifica el elemento de **HTML5** al que se quiere cambiar su estilo, específicamente reconoce solo a las etiquetas,

identificadores y clases de **HTML5**, después vienen dos llaves quienes contienen el bloque de declaración, dentro de estas llaves se emplean las propiedades de **CSS** y su valor separándose con dos puntos y terminando la línea con punto y coma. Esta relación de **CSS** es muy fácil de utilizar pues es lo único que se tiene que saber y entender para ponerlo en práctica.

Si se ejecuta esa declaración de **CSS**, obtenemos algo así:



Aplicando el ejemplo anterior de **CSS**, se puede observar que altera la única etiqueta **<H1>** que se tiene en **HTML5**, pues el selector contiene el nombre **H1** lo que indica que todo lo que se encuentre en el bloque de declaración transformará a la etiqueta de **HTML5** y se visualizara en el navegador web.

En este caso, en **HTML5** se tiene una sola etiqueta **<H1>** por tanto **CSS** solo la modifica a ella, pero existen casos que cuando se tienen muchas etiquetas del mismo nombre, por ejemplo, que se tengan muchos párrafos dentro del archivo **HTML5** y **CSS** tiene un selector con el nombre **P**, a todos y cada uno de los elementos que tengan la etiqueta **<P>** se modificaran, esto en algunos casos funciona y facilita el trabajo, ya que no se tiene que hacer un bloque de declaración por cada elemento que posea **HTML5**, de lo contrario si solo se quieren alterar unos y otros no, se puede especificar con los identificadores **ID** o con las clases **CLASS** que tenga cada etiqueta.

```

<p id="inicio">Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod
tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam,
quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo
consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse
cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non
proident, sunt in culpa qui officia deserunt mollit anim id est laborum.</p>

<p>Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod
tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam,
quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo
consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse
cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non
proident, sunt in culpa qui officia deserunt mollit anim id est laborum.</p>

<p>Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod
tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam,
quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo
consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse
cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non
proident, sunt in culpa qui officia deserunt mollit anim id est laborum.</p>

```

Ilustración 129 - Imagen propia

```

p{
    color: indianred;
    text-align: center;
}

/*identificador*/
#inicio{
    color: lawngreen;
    text-align: left;
}

```

Ilustración 130 - Imagen propia



Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Ilustración 131 - Imagen propia

Pero CSS también necesita que le especifiquen los selectores, como se puede observar en la imagen anterior, existe un selector con el signo “#”, este se les asigna a aquellos elementos que son identificadores **ID** y para los elementos que son clases **CLASS** se necesita un “.” antes del selector.


```

/*identificador*/
#inicio{
  color: lawngreen;
  text-align: left;
}

/*clase*/
.parrafo{
  color: deepskyblue;
  text-align: right;
}

```

Ilustración 132 - Imagen propia



Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Ilustración 133 - Imagen propia

Por supuesto, también se pueden encontrar dificultades cuando diferentes elementos de **HTML5** se quieren tener con el mismo estilo, a esto se conoce como agrupación de selectores, de esta manera se aminora el código y se obtiene un el mismo resultado. Para agruparlos solo basta con separarlos con una coma a un selector de otro.



```

h1, h2, p.parrafo{
  color: darkgoldenrod;
  text-align: center;
  text-decoration: underline;
}

```

Ilustración 134 - Imagen propia

Estructura semantica

- Inicio
- HTML5
- CSS
- JavaScript
- Bootstrap

Tema

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Ilustración 135 - Imagen propia

Antes de comenzar a conocer todas las propiedades de **CSS**, es importante saber qué tipos de medidas se utilizan, pues como la estructura/sintaxis son fundamentales para poder diseñar una página web.

8.3.COLORES

Para hablar de diseño, personalización o apariencia, es muy importante conocer de una pieza fundamental, como en todo, el color es muy importante en la presentación de cualquier trabajo, ya que se está viviendo una etapa en la que la visualización es la principal fuente de información y comunicación, pues sin esto, haciendo referencia a la tecnología, el internet no sería solicitado para redes sociales o páginas de cualquier tipo si no tienen una apariencia llamativa, profesional y por supuesto innovadora, para ello es muy importante aplicar presencia con el color en nuestra pantalla.

Para entender un poco sobre el color, es importante saber ¿Qué es?, ¿De dónde proviene?, ¿Cuál es su objetivo?, para que un humano detecte color, primero debe visualizar la luz blanca. Todo cuerpo iluminado absorbe una parte de las ondas electromagnéticas y refleja las restantes. Las ondas reflejadas son captadas por el ojo e interpretadas en el cerebro como distintos colores según las longitudes de ondas correspondientes. Para tener un concepto formal sobre el color, depende de la manera en que se quiera tener el resultado, por ejemplo: para verlo de una percepción física, el color es una propiedad de la luz en los objetos, para la química el color es una reacción de elementos, para la filosofía y psicología el color es un elemento que porta simbolismo de los elementos.

El objetivo del color es caracterizar a un elemento cual sea, no importa tamaño, textura ni forma, brinda apariencia y significado a todo aquello que lo porte. Este proviene de la descomposición de la luz, ya sea solar o fuente artificial, el color varía de acuerdo a la naturaleza de la posición de luz que tenga y de la cual provienen 7 colores espectrales que identificamos en el cielo cuando hay luz de sol y está lloviendo: rojo, naranja, amarillo, verde, azul, azul oscuro y violeta, a estos colores los conocemos como “Arcoíris”, pero existen dos colores más: el color blanco proviene de la presencia de todos los rayos y el color negro es la ausencia todo rayo de luz. (W3schools, 2018-2019)

Por supuesto que existe la gran pregunta ¿Cuántos colores existen?, teniendo en cuenta lo anterior, esos 10 colores desencadenan una gran reacción, pues la exposición de los siete colores con el color blanco o negro que, en pocas palabras, es decir, presencia o ausencia de luz, se obtiene una infinidad de colores, todos derivados de la descomposición de luz. Realmente no existe una cantidad exacta de cuantos son los colores que existen hasta el día de hoy, pero algo que es muy claro, es que las diferentes variables que se tienen desatan una cantidad infinita de colores.








Nombre	Color
Rojo	
Naranja/Anaranjado	
Amarillo	
Verde	
Azul	
Añil/Azul oscuro	
Violeta	

Ilustración 136 - Tabla propia

Ahora bien, para entender como nacen los colores en sí, viéndolo de forma en que ya se puede jugar y crear con ellos, se tiene que comenzar con los colores primarios, estos colores son los únicos que no pueden ser obtenidos por medio de mezclas con otros colores: Amarillo, Rojo y Azul, estos colores son provenientes de los colores de la luz.

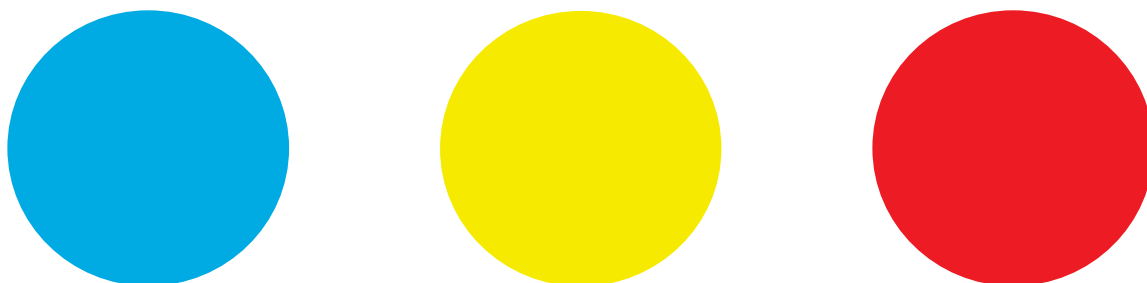


Ilustración 137 - Imagen propia

Los colores secundarios son aquellos que obtenemos con la mezcla de los colores primarios.

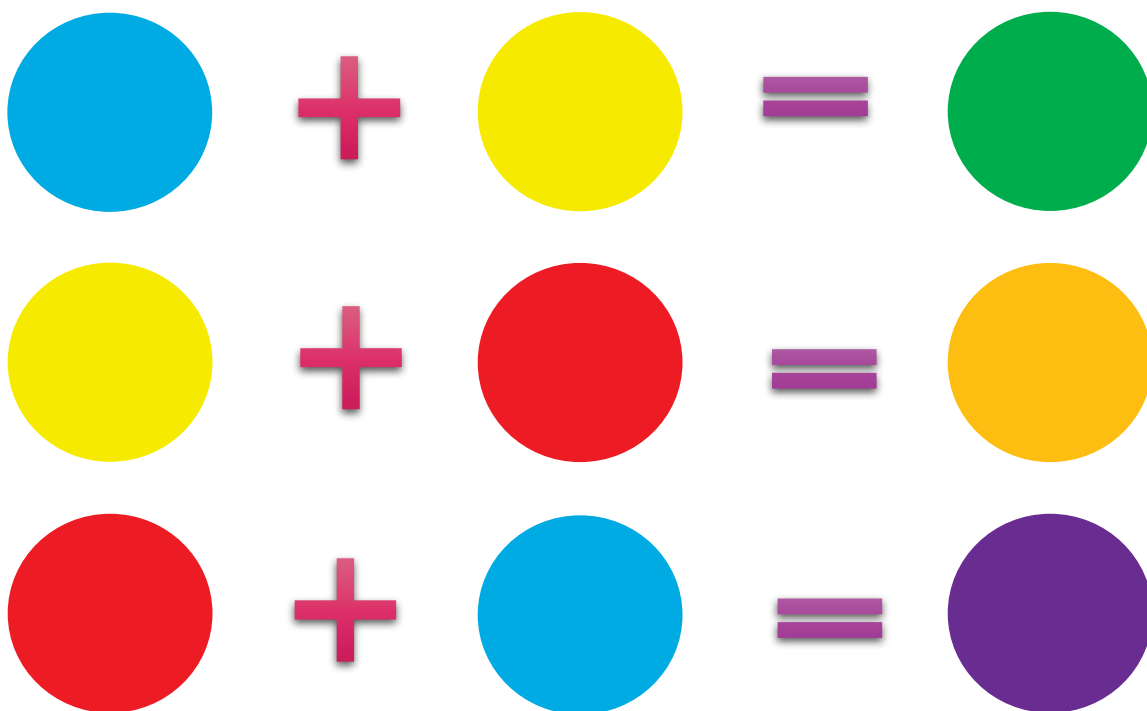


Ilustración 138 - Imagen propia

Pero esto no aplica en **CSS**, pues los colores primarios provienen de RGB que maneja los colores Rojo, Verde y Azul, de estos surgen colores secundarios y terciarios, donde también influyen los elementos de opacidad y luminosidad, lo que provoca una infinidad de colores y estos son compatibles con todos los navegadores web.



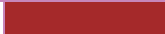
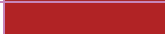
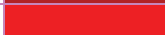

















































































Ilustración 139 - Imagen propia (Círculo cromático)

En CSS el color se puede especificar por:

- **Colores predeterminados:** Son todos aquellos colores que son compatibles en todos los navegadores web que se tienen hoy en día, para esto, se habla alrededor de 140 colores.
- **Colores con valores RGB:** Son todos aquellos colores que se realizan por medio de mezclas entre tres colores primarios (**RRR, GGG, BBB**), como lo son Red (Rojo), Green (Verde) y Blue (Azul), su intensidad se mide de 0 – 255 puntos y las medidas de los tres por ejemplo se miden de (0, 0, 0) se obtiene el negro y si tenemos lo contrario, es decir el valor más alto (255, 255, 255) se obtiene el color blanco.
- **Colores con valores hexadecimal:** Son todos aquellos colores que se obtienen con dígitos hexadecimales (00-ff) y se representa de esta manera **#rrggbb**, donde se consideran los mismos parámetros de color: rojo, verde y azul. El valor más alto lo ocupa #ffffff se obtiene un color blanco y el más bajo que es #000000 se obtiene el color negro.
- **Colores con valores HSL:** Son todos aquellos colores que especifican la tonalidad, saturación y luminosidad (**000, 100%, 100%**), los valores se manejan de 0 – 360, 0% - 100%, 0% - 100%, los valores menores, es decir, 0 en el caso de la tonalidad o matiz es rojo, 120 llega a verde y 240 parte del azul, en la saturación el 0 parte de un tono gris y el 100 significa el color completo y el ultimo donde 0 significa el color más oscuro y el 100 el color más claro de la tonalidad que dependa el primer parámetro.
- **Colores con valor RGBA:** Se trata de los mismos colores que tienen los valores RGB, la diferencia radica en que el ultimo estándar que se llama **Alfa** especifica la opacidad del color, es decir, la transparencia y su valor es decimal, donde 0.0 es completamente transparente y 1.0 es el color completo.

- **Colores con valor HSLA:** Son todos los colores que se encuentran con los valores HSL, la diferencia radica en que el ultimo estándar que se llama **Alfa** especifica la opacidad del color, es decir, la transparencia y su valor es decimal, donde 0.0 es completamente transparente y 1.0 es el color completo.

Nombre del color	RGB	HEX	HSL	
maroon	(128, 0, 0)	#800000	(0, 100%, 25%)	
darkred	(139, 0, 0)	#8b0000	(0, 100%, 27%)	
brown	(165, 42, 42)	#a52a2a	(0, 59%, 41%)	
firebrick	(178, 34, 34)	#b22222	(0, 68%, 42%)	
red	(255, 0, 0)	#ff0000	(0, 100%, 50%)	
crimson	(220, 20, 60)	#dc143c	(348, 83%, 47%)	
indianred	(205, 92, 92)	#cd5c5c	(0, 53%, 58%)	
lightcoral	(240, 128, 128)	#f08080	(0, 79%, 72%)	
salmon	(250, 128, 114)	#fa8072	(6, 93%, 71%)	
lightsalmon	(255, 160, 122)	#ffa07a	(17, 100%, 74%)	
darksalmon	(233, 150, 122)	#e9967a	(15, 72%, 70%)	
saddlebrown	(139, 69, 19)	#8b4513	(25, 36%, 31%)	
sienna	(160, 82,45)	#a0522d	(19, 56%, 40%)	
chocolate	(210, 105, 30)	#d2691e	(25, 75%, 47%)	
peru	(205, 133, 63)	#cd853f	(30, 59%, 53%)	
sandybrown	(244, 164, 96)	#f4a460	(28, 87%, 67%)	
burlywood	(222, 184, 135)	#deb887	(34, 57%, 70%)	
orangered	(255, 69, 0)	#ff4500	(16, 100%, 50%)	
tomato	(255, 99, 71)	#ff6347	(9, 100%, 64%)	
coral	(255,127,80)	#ff7f50	(16, 100%, 66%)	
darkgoldenrod	(184, 134, 11)	#b8860b	(43, 89%, 38%)	
goldenrod	(218, 165, 32)	#daa520	(43, 74%, 49%)	
darkorange	(255, 140, 0)	#ff8c00	(33, 100%, 50%)	
orange	(255, 165, 0)	#ffa500	(39, 100%, 50%)	
gold	(255, 215, 0)	#ffd700	(51, 100%, 50%)	
yellow	(255, 255, 0)	#ffff00	(60, 100%, 50%)	
khaki	(240, 230, 140)	#f0e68c	(54, 77%, 75%)	
lightyellow	(255, 255, 153)	#ffff99	(60, 100%, 80%)	
greenyellow	(173, 255, 47)	#adff2f	(84, 100%, 59%)	
chartreuse	(127, 255, 0)	#7fff00	(90, 100%, 50%)	
lawngreen	(124, 252, 0)	#7cfc00	(90, 100%, 49%)	
lime	(0, 255, 0)	#00ff00	(120, 100%, 50%)	
limegreen	(50, 205, 50)	#32cd32	(120, 61%, 50%)	
palegreen	(152, 251, 152)	#98fb98	(120, 93%, 79%)	
lightgreen	(144, 238, 144)	#90ee90	(120, 73%, 75%)	
mediumspringgreen	(0, 250, 154)	#00fa9a	(157, 100%, 49%)	
springgreen	(0, 255, 127)	#00ff7f	(150, 100%, 50%)	
mediumseagreen	(60, 179, 113)	#3cb371	(147, 50%, 47%)	
seagreen	(46, 139, 87)	#2e8b57	(146, 50%, 36%)	

forestgreen	(34, 139, 34)	#228b22	(120, 61%, 34%)	
green	(0, 128, 0)	#008000	(120, 100%, 25%)	
darkgreen	(0, 100, 0)	#006400	(120, 100%, 20%)	
yellowgreen	(154, 205, 50)	#9acd32	(80, 61%, 50%)	
darkseagreen	(143, 188, 143)	#8fbc8f	(120, 25%, 65%)	
olivedrab	(107, 142, 35)	#6b8e23	(80, 60%, 35%)	
olive	(128, 128, 0)	#808000	(60, 100%, 25%)	
darkolivegreen	(85, 107, 47)	#556b2f	(82, 39%, 30%)	
teal	(0, 128, 128)	#008080	(180, 100%, 25%)	
darkcyan	(0, 139, 139)	#008b8b	(180, 100%, 27%)	
cadetblue	(95, 158, 160)	#5f9ea0	(182, 25%, 50%)	
lightseagreen	(32, 178, 170)	#20b2aa	(177, 70%, 41%)	
mediumaquamarine	(102, 205, 170)	#66cdaa	(160, 51%, 60%)	
darkturquoise	(0, 206, 209)	#00ced1	(181, 100%, 41%)	
mediumturquoise	(72, 209, 204)	#48d1cc	(178, 60%, 55%)	
turquoise	(64, 224, 208)	#40e0d0	(174, 72%, 56%)	
aquamarine	(127, 255, 212)	#7fffd4	(160, 100%, 75%)	
light cyan/aqua	(224, 255, 255)	#e0ffff	(180, 100%, 94%)	
paleturquoise	(175, 238, 238)	#afeeee	(180, 65%, 81%)	
powderblue	(176, 224, 230)	#b0e0e6	(187, 52%, 80%)	
lightsteelblue	(176, 196, 222)	#b0c4de	(214, 41%, 78%)	
skyblue	(135, 206, 235)	#87ceef	(197, 71%, 73%)	
lightskyblue	(135, 206, 250)	#87cefa	(203, 92%, 75%)	
cyan/aqua	(0, 255, 255)	#00ffff	(180, 100%, 50%)	
deepskyblue	(0, 191, 255)	#00bfff	(195, 100%, 50%)	
dodgerblue	(30, 144, 255)	#1e90ff	(210, 100%, 56%)	
cornflowerblue	(100, 149, 237)	#6495ed	(219, 79%, 66%)	
steelblue	(70, 130, 180)	#4682b4	(207, 44%, 49%)	
royalblue	(65, 105, 225)	#4169e1	(225, 73%, 57%)	
blue	(0, 0, 255)	#0000ff	(240, 100%, 50%)	
mediumblue	(0, 0, 205)	#0000cd	(240, 100%, 40%)	
darkblue	(0, 0, 139)	#00008b	(240, 100%, 27%)	
navy	(0, 0, 128)	#000080	(240, 100%, 25%)	
midnightblue	(25, 25, 112)	#191970	(240, 64%, 27%)	
darkslateblue	(72, 61, 139)	#483d8b	(248, 39%, 39%)	
mediumslateblue	(123, 104, 238)	#7b68ee	(249, 80%, 67%)	
mediumpurple	(147, 112, 219)	#9370db	(260, 60%, 65%)	
blueviolet	(138, 43, 226)	#8a2be2	(271, 76%, 53%)	
rebeccapurple	(102, 51, 153)	#663399	(270, 50%, 40%)	
indigo	(75, 0, 130)	#4b0082	(275, 100%, 25%)	
purple	(128, 0, 128)	#800080	(300, 100%, 25%)	
darkmagenta	(139, 0, 139)	#8b008b	(300, 100%, 27%)	
darkviolet	(148, 0, 211)	#9400d3	(282, 100%, 41%)	
darkorchid	(153, 50, 204)	#9932cc	(280, 61%, 50%)	
mediumorchid	(186, 85, 211)	#ba55d3	(288, 59%, 59%)	


































orchid	(218, 112, 214)	#da70d6	(302, 59%, 65%)	
plum	(221, 160, 221)	#dda0dd	(300, 47%, 75%)	
thistle	(216, 191, 216)	#d8bfd8	(300, 24%, 80%)	
lavander	(230, 230, 250)	#e6e6fa	(240, 67%, 94%)	
pink	(255, 192, 203)	#ffc0cb	(350, 100%, 88%)	
lightpink	(255, 182, 193)	#ffb6c1	(351, 100%, 86%)	
hotpink	(255, 105, 180)	#fb69bb4	(330, 100%, 71%)	
palevioletred	(219, 112, 147)	#db7093	(340, 60%, 65%)	
fucsia/magenta	(255, 0, 255)	#ff00ff	(300, 100%, 50%)	
deeppink	(255, 20, 147)	#ff1493	(328, 100%, 54%)	
mediumvioletred	(199, 21, 133)	#c71585	(322, 81%, 43%)	

Ilustración 140 - Tabla tomada de https://www.w3schools.com/colors/colors_groups.asp

En la tabla anterior, se pueden observar solo algunos de los colores que se suelen ocupar en CSS, pues las cuatro medidas de color que se encuentran son la combinación de los colores primarios de CSS, existen muchos más, por supuesto, pues todos estos derivan directamente de los colores primarios, secundarios y terciarios, mezclándolos más colores, con luminosidad, oscuridad, opacidad o transparencia, saturación o pureza del color.

La siguiente tabla, muestra los colores que se pueden conseguir solo con la mezcla de blanco y negro.

RGB	HEX	COLOR
(0, 0, 0)	#000000	
(8, 8, 8)	#080808	
(16, 16, 16)	#101010	
(24, 24, 24)	#181818	
(32, 32, 32)	#202020	
(40, 40, 40)	#282828	
(48, 48, 48)	#303030	
(56, 56, 56)	#383838	
(64, 64, 64)	#404040	
(72, 72, 72)	#484848	
(80, 80, 80)	#505050	
(88, 88, 88)	#585858	
(96, 96, 96)	#606060	
(104, 104, 104)	#686868	
(105, 105, 105)	#696969	
(112, 112, 112)	#707070	
(120, 120, 120)	#787878	
(128, 128, 128)	#808080	
(136, 136, 136)	#888888	
(144, 144, 144)	#909090	
(152, 152, 152)	#989898	
(160, 160, 160)	#a0a0a0	

(168, 168, 168)	#a8a8a8	
(169, 169, 169)	#a9a9a9	
(176, 176, 176)	#b0b0b0	
(184, 184, 184)	#b8b8b8	
(190, 190, 190)	#bebebe	
(192, 192, 192)	#c0c0c0	
(200, 200, 200)	#c8c8c8	
(208, 208, 208)	#d0d0d0	
(211, 211, 211)	#d3d3d3	
(216, 216, 216)	#d8d8d8	
(220, 220, 220)	#dcdcdc	
(224, 224, 224)	#e0e0e0	
(232, 232, 232)	#e8e8e8	
(240, 240, 240)	#f0f0f0	
(245, 245, 245)	#f5f5f5	
(248, 248, 248)	#f8f8f8	
(255, 255, 255)	#ffffff	

Ilustración 141 - Tabla propia

Como se puede observar, la ausencia de luz al momento de aumentarla se puede degradar un color a otro y notar la transición, esto puede pasar con cualquier color y se obtendrán muchos tonos de colores.












COLOR	RGBA-ALFA/OPACIDAD
	(255, 20, 147, 0.0)
	(255, 20, 147, 0.1)
	(255, 20, 147, 0.2)
	(255, 20, 147, 0.3)
	(255, 20, 147, 0.4)
	(255, 20, 147, 0.5)
	(255, 20, 147, 0.6)
	(255, 20, 147, 0.7)
	(255, 20, 147, 0.8)
	(255, 20, 147, 0.9)
	(255, 20, 147, 1.0)

Ilustración 142 - Imagen propia

Existe una gran diferencia entre degradar un color de más claro a oscuro y en aplicar opacidad a un color, como el ejemplo de la tabla de negro a blanco, ahí solo se hizo la degradación de un color a otro y sus tonalidades, pero en esta última tabla, la opacidad lo único que consigue es que el color se desvanezca, es decir, que se vuelva transparente conforme a su parámetro.

Ahora bien, cuando se intenta crear una página web, por lo regular se utilizan colores con combinaciones que atraigan al usuario, sea vistosa e innovadora, moderna, pero a la vez profesional, lo único que pasa es que también existen reglas en los colores, pues los colores necesitan llevarse bien y crear un ambiente armónico entre ellos, para ello existen diferentes combinaciones que facilitan este trabajo como son:

- ❖ **Combinación monocromática:** Los esquemas de colores monocromáticos utilizan un solo color. Se pueden utilizar diferentes valores de un mismo color para hacer que parezcan diferentes colores. Los esquemas de colores monocromáticos son armoniosos y relajantes, y proporcionan una sensación de sofisticación.

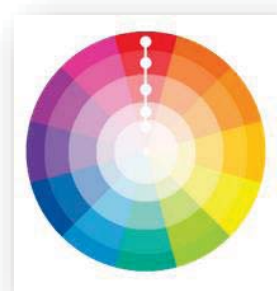


Ilustración 143 - Imagen tomada de https://www.w3schools.com/colors/colors_schemes.asp

- ❖ **Combinación análoga:** Los esquemas de colores análogos utilizan colores contiguos del círculo cromático. Suelen combinar bien, y crean diseños serenos y cómodos. Estas combinaciones se encuentran a menudo en la naturaleza, y son armoniosas y agradables a la vista.

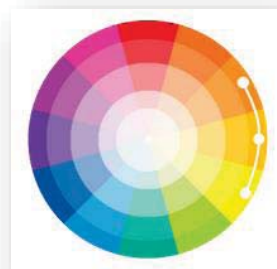


Ilustración 144 - Imagen tomada de https://www.w3schools.com/colors/colors_schemes.asp

- ❖ **Combinación complementaria:** Los colores opuestos crean esquemas de colores contrastantes. Por ejemplo, amarillo y morado, o rojo y verde, son colores contrastantes. Los esquemas de colores contrastantes suelen utilizar un color cálido y un color frío.



Ilustración 145 - Imagen tomada de https://www.w3schools.com/colors/colors_schemes.asp

- ❖ **Combinación triádica:** Las combinaciones de colores triádicas se componen de tres colores opuestos. Un esquema de colores triádico utiliza colores separados por la misma distancia en el círculo cromático. Suelen dar un contraste armonioso.



Ilustración 146 - Imagen tomada de https://www.w3schools.com/colors/colors_schemes.asp

8.4.MEDIDAS DE CSS

Para poder dar estilo a un elemento de **HTML5**, es fundamental lo que se conoció anteriormente, pues la sintaxis y las medias de color conforman gran parte del manejo de **CSS**, aun, no obstante, existe otro elemento muy básico, pero no menos importante, que son las Medidas de Unidad, con este último elemento, se completan al 100% los elementos esenciales para desarrollar un diseño y/o estilo de un elemento.

Ahora bien, las unidades de medida permiten desplazar un elemento dentro de la pantalla y no solo eso también admite definir la extensión o longitud de algún valor, ya sea altura y/o anchura. Contemplando lo anterior, para poder declarar las medidas se deben de tener en cuenta dos puntos muy importantes; las medidas que se indican dentro de estos parámetros son declaradas como valores numéricos enteros y decimales, estos valores se dividen en dos partes dentro del lenguaje **CSS**, (Valores enteros para Valores Absolutos) y (Valores decimales para Valores Relativos), después de esto se encuentra la unidad que determina el valor sin dejar espacio en blanco del número al parámetro.

Quizás suena confuso, pero para dejar más claro el tema de las medidas, se puede explicar de la siguiente manera:

- **Una medida absoluta:** Es todo aquel valor completamente definido, el cual no depende de ninguna otra medida, es decir, son valores que permiten dar un valor fijo a un elemento y nada lo puede modificar, por lo cual no se pueden adaptar a diferentes tipos de pantallas o dispositivos móviles, pues siempre mantendrán la medida que tienen asignadas.
 - **Cm:** Centímetros
 - **Mm:** Milímetros
 - **In:** Pulgadas (1in = 96px = 2.54cm)
 - **Pt:** Puntos (1pt = 1/72in = 0.35mm)
 - **Pc:** Pica (1pc = 4.23mm = 12pt)

- **Una medida relativa:** Definen su valor en relación con otra medida, es decir, son flexibles para adaptarse a diferentes tamaños de dispositivos, por lo tanto, son los elementos más usados dentro del diseño web.
 - **Px:** Píxeles.
 - **Em:** Definen el tamaño de la fuente (2em = 2 veces su tamaño original)
 - **Ex:** Definen la altura de la fuente.

Para detallar aún más, cuando se refiere a medidas escaladas de algún elemento, por lo regular se utilizan más PX y EM, por un lado, los píxeles se permiten ajustar a cualquier pantalla donde se proyecte la página web, pero por otro lado EM se permite un poco más de profesionalismo al momento de escalar el tamaño de algún elemento, pues no pierde calidad al elemento.

Cabe recalcar que el tamaño default que maneja cualquier tipo de fuente es de 16px, por lo tanto, esos píxeles equivalen a 1em al igual equivale a 12pt, pero claro que te preguntarás que significa “em”, bueno pues “em” no fue desarrollada por **CSS**, esta unidad de medida

fue definida y creada para la tipografía desde hace mucho tiempo y aunque no es una definición exacta, el tamaño que se le adjudica a ese elemento es el ancho de una “eme mayúscula” y, por ende, de ese motivo tiene su nombre.

Ahora bien, para la época moderna que se está viviendo, una página web ya no se mira solo en un computador, estas también son consumidas por medio de dispositivos móviles como celulares y tabletas, por lo tanto, la actualización de las tecnologías debe y tienen que aplicarse, porque la demanda de calidad y cantidad es exorbitante, para ello px y em son por ahora muy indispensables en el diseño de un sitio web.

Dejando en claro estos puntos tan esenciales para poder actuar, ahora sí se puede comenzar a conocer las propiedades que vienen a complementar el cuerpo de **CSS**, pues en el próximo tema, se podrán aplicar las unidades de color y medida sin ningún problema y así poder entender las utilidades de estos elementos.

8.5.PROPIEDADES

Teniendo una base sobre la estructura de **CSS** y como se indican los colores, continuar con las propiedades es la tercera parte para aprender a utilizar **CSS**, ahora bien, algo muy importante dentro el diseño de front end, son los fondos de pantalla, son indispensables para dar un efecto de personalidad, dentro de esta propiedad se pueden utilizar diferentes opciones.

Para declarar un fondo se hace por medio de la propiedad **BACKGROUND**, esta propiedad maneja solo dos elementos visuales: color e imágenes, para declararlos se pueden hacer dentro de background como declaración abreviada o declarándolo en las líneas de código correspondientes sin ningún problema, pues al final se obtiene el mismo resultado, **BACKGROUND** tiene muchas opciones para implementarlo:

```
body {  
  background: mediumspringgreen;  
}
```

Ilustración 147 - Imagen propia

```
body{  
  background-color: #00fa9a;  
}
```

Ilustración 148 - Imagen propia

Cabe destacar que es el mismo color, pero declarado de diferente manera. Ahora el resultado será este en ambos ejemplos:

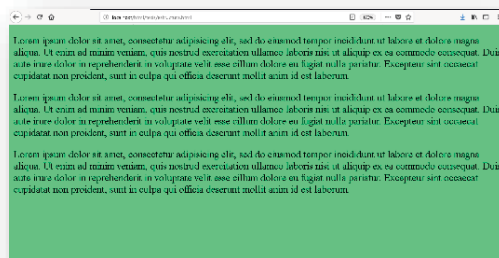


Ilustración 149 - Imagen propia

Al utilizar todos los tipos de medidas de los colores se visualiza así:

```
#rgb{  
  background: rgb(220, 20, 60);  
}  
  
#rgba{  
  background: rgba(0,255,0,0.5);  
}  
  
#hex{  
  background: #008080;  
}  
  
#hsl{  
  background: hsl(300, 100%, 25%);  
}  
  
#hsla{  
  background: hsla(351, 100%, 86%, 0.3);  
}  
  
#nombre{  
  background: gold;  
}
```

Ilustración 150 - Imagen propia

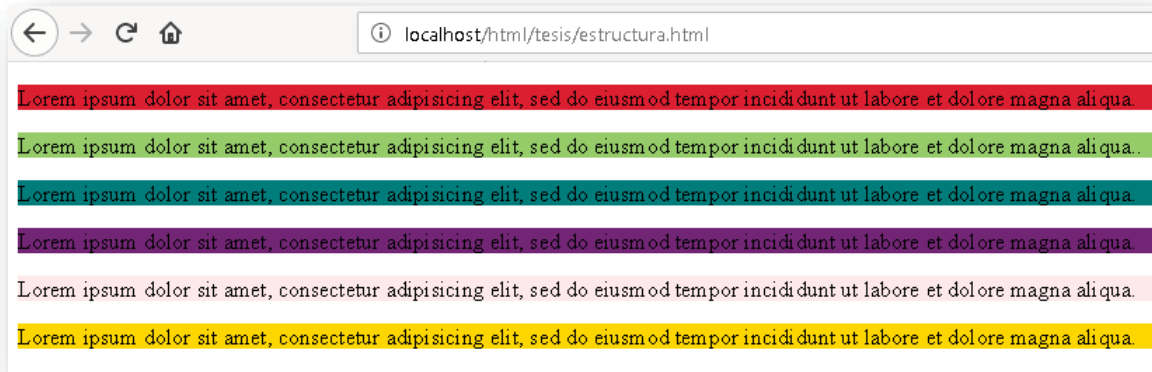


Ilustración 151 - Imagen propia

Por otro lado, siendo un poco más creativo el asunto del fondo de los colores, también se cuenta con un valor que permite jugar con los colores de manera que se pueda forjar un aspecto armónico en la pantalla, con esto se quiere decir que los colores se pueden combinar y degradar en diferentes tonos y formas.

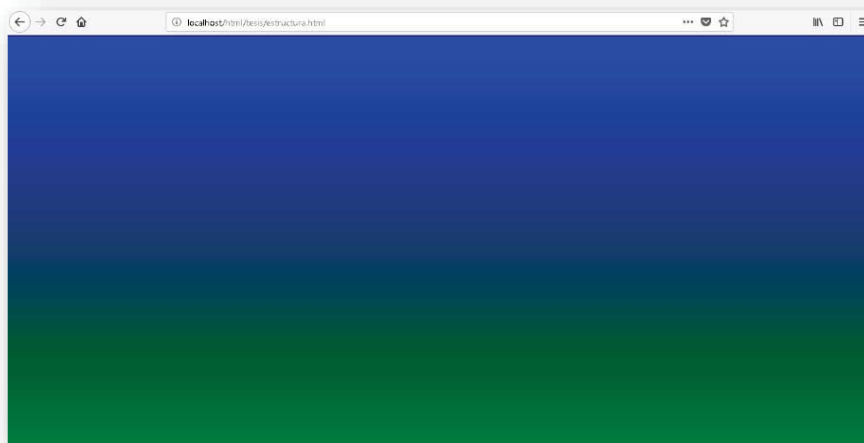


Ilustración 152 - Imagen propia

Para lograrlo se debe tener como mínimo dos colores para poder hacer el desvanecimiento gradual dentro de ellos, en CSS se tienen diferentes tipos de direcciones de gradientes.

LINEAL-GRADIENT: Dentro de este valor, se tiene la siguiente sintaxis.

```
background-image: linear-gradient(direccion de color, color%, color%, ...);
```

Ilustración 153 - Imagen propia

Para ser más específicos, linear-gradient determina la dirección de los colores a donde se quiera dirigir el degradado, se puede jugar de diferentes maneras con este valor, pues se presta para hacer diferentes fondos, se pueden utilizar a partir de 2 a más colores según sea el gusto, dentro de los colores podemos especificar que tanto porcentaje se quiere visualizar cierto color, se pueden utilizar todos los modos de color que existen para aplicarlo.

```
html{  
  height: 650px;  
  background-color: #2cbaa6;  
  background-image: linear-gradient(#2cbaa6, #29d90d);
```

Ilustración 154 - Imagen propia



Ilustración 155 - Imagen propia

Como se puede observar en el ejemplo, este valor se puede utilizar dentro de cualquier elemento, pero se tiene que determinar la altura del elemento para que el gradiente se pueda observar bien, si no tendríamos algo como esto:

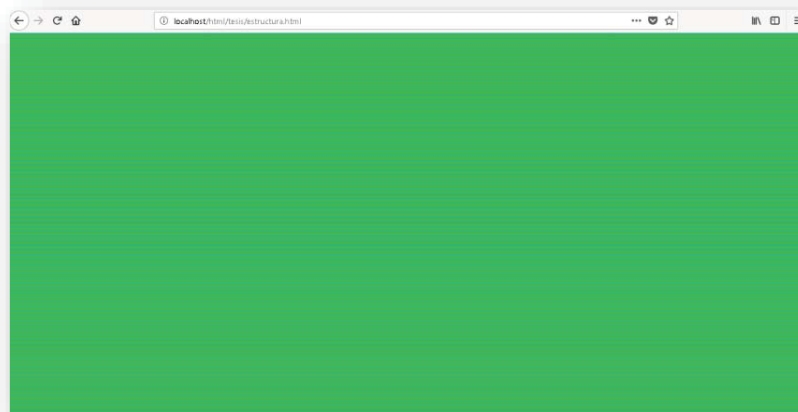


Ilustración 156 - Imagen propia

No tiene el espacio suficiente para lucir, por lo cual es importante señalar el espacio del fondo, por otro lado, para lograr este efecto se necesita de otra cosa más, poner un fondo de color **BACKGROUND-COLOR** como base antes de la propiedad que genera el efecto, esto para que en dado caso que no funcione, pueda aparecer un color default en la pantalla.

Los siguientes ejemplos, son de las diferentes direcciones y maneras de aplicarlo:

```
html{
  height: 650px;
  background-color: rgba(206, 0, 101, 0);
  background-image: linear-gradient(to right, rgba(206, 0, 101, 0), rgba(206, 0, 101, 1.0));
}
```

Ilustración 157 - Imagen propia



Ilustración 158 - Imagen propia

```
html{
  height: 650px;
  background-color: #009096;
  background-image: linear-gradient(to bottom right, #009096, #bc0083, #998600);
}
```

Ilustración 159 - Imagen propia



Ilustración 160 - Imagen propia

```
#gradiente1{
  height: 150px;
  background-color: #1a0469;
  background-image: linear-gradient(0deg, #1a0469, #b20090);
  margin-bottom: 5px;
}

#gradiente2{
  height: 150px;
  background-color: #1a0469;
  background-image: linear-gradient(90deg, #1a0469, #b20090);
  margin-bottom: 5px;
}

#gradiente3{
  height: 150px;
  background-color: #1a0469;
  background-image: linear-gradient(180deg, #1a0469, #b20090);
  margin-bottom: 5px;
}

#gradiente4{
  height: 150px;
  background-color: #1a0469;
  background-image: linear-gradient(-90deg, #1a0469, #b20090);
}
```

Ilustración 161 - Imagen propia



Ilustración 162 - Imagen propia


```
html{
  height: 650px;
  background-color: violet;
  background-image: linear-gradient(to left bottom, black, violet, indigo, blue, cyan, green,
  yellow, orange, red, deeppink, pink, white);
  margin-bottom: 5px;
}
```

Ilustración 163 - Imagen propia



Ilustración 164 - Imagen propia

- ✚ **REPEATING-LINEAR-GRADIENT:** Este degradado tiene el objetivo de crear lienzos de color desvanecido repetitivamente, donde se pueden utilizar de diversa manera para la presentación en pantalla. Se puede aplicar de igual manera en todos los tipos de gradientes lineales que se mostraron anteriormente.

```
html{
  height: 200px;
  background-color: violet;
  background-image: repeating-linear-gradient(gold, magenta, cyan);
}
```

Ilustración 165 - Imagen propia



Ilustración 166 - Imagen propia

🚦 **RADIAL-GRADIENT:** Dentro de este valor se tiene la siguiente sintaxis:

```
background-image: radial-gradient(forma, tamaño at posición, color%, ..., color%);
```

Ilustración 167 - Imagen propia

Este valor es muy similar al de un gradiente lineal en cuanto a función, su única diferencia es que este es en forma circular, para entender la sintaxis el primer parámetro es la forma que dará el gradiente, es decir, su forma predeterminada del valor es una elipse u ovalo, el tamaño se define por medio de cuatro parámetros y la posición será central.

```
html{  
  height: 650px;  
  background-color: #00c2b4;  
  background-image: radial-gradient(#00c2b4, #bc54d6, #261ccf);  
}
```

Ilustración 168 - Imagen propia

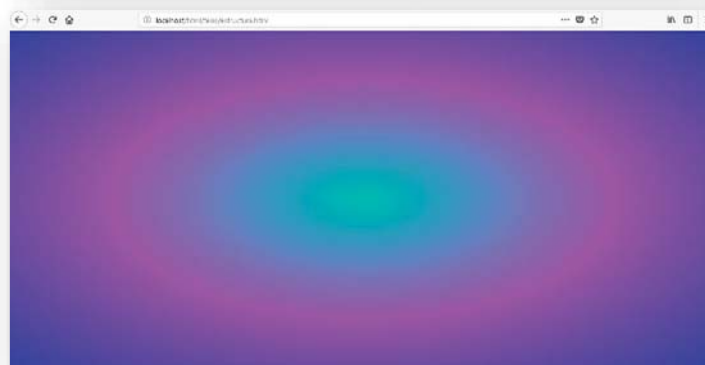


Ilustración 169 - Imagen propia

Como se puede observar, declara la instrucción se obtiene una elipse, para este tipo de degradado se puede utilizar de igual forma la opacidad, la combinación multicolor, solo que aquí se tienen solo dos formas, elipse o circular:

```
html{  
  height: 650px;  
  background-color: #8b0065;  
  background-image: radial-gradient(circle, #8b0065, #ff4100 10%, #e054ba);  
}
```

Ilustración 170 - Imagen propia



Ilustración 171 - Imagen propia

Para los tamaños se tienen los siguientes parámetros:

```
html{  
  height: 650px;  
  background-color: deeppink;  
  background-image: radial-gradient(closest-side at 60% 55%, deeppink, green, gold);  
}
```

Ilustración 172 - Imagen propia



Ilustración 173 - Imagen propia

```
html{  
  height: 650px;  
  background-color: white;  
  background-image: radial-gradient(farthest-side at 10% 55%, white, violet, grey);  
}
```

Ilustración 174 - Imagen propia



Ilustración 175 - Imagen propia

```
html{
  height: 650px;
  background-color: red;
  background-image: radial-gradient(closest-corner at 80% 40%, red, plum, deeppink);
}
```

Ilustración 176 - Imagen propia

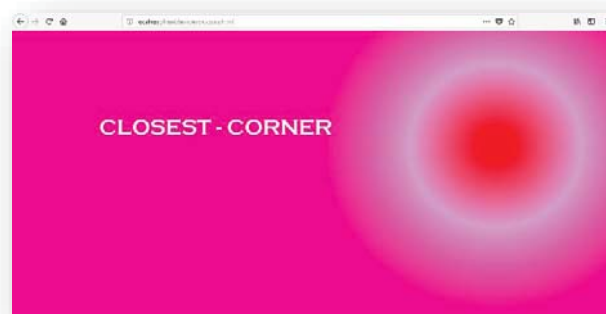


Ilustración 177 - Imagen propia

```
html{
  height: 650px;
  background-color: black;
  background-image: radial-gradient(farthest-corner at 60% 85%, black, lime, blue);
}
```

Ilustración 178 - Imagen propia

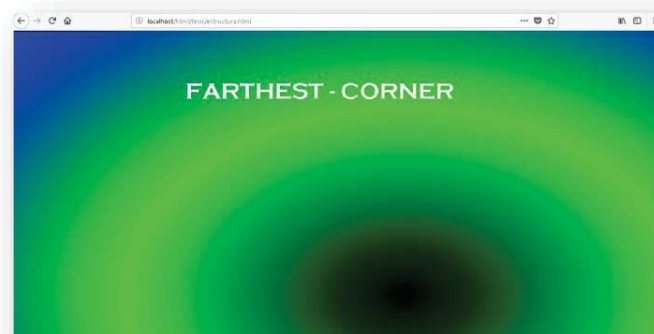


Ilustración 179 - Imagen propia

Como se puede observar el degradado de colores puede variar entre la forma circular o elíptica, dependiendo de los porcentajes y su posicionamiento variará de acuerdo a los parámetros indicados, de esta manera se puede jugar con los valores y elementos que se tienen para crear un fondo atractivo de manera sencilla.

- ✚ **REPEATING-RADIAL-GRADIENT:** Este valor representa un gradiente radial pero repetitivo, lo cual permitiría hacer de este elemento otra herramienta para formar un fondo diferente.

```
#gradiente1{  
  height: 650px;  
  width: 800px;  
  background-color: red;  
  background-image: repeating-radial-gradient(deeppink, plum 10%, white 10%);  
}
```

Ilustración 180 - Imagen propia

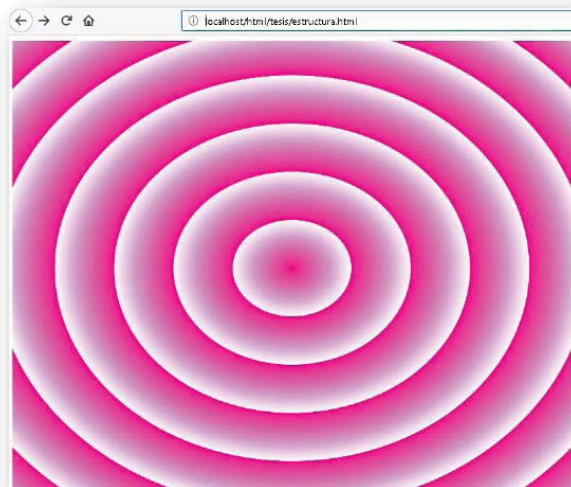


Ilustración 181 - Imagen propia

Para poder aprender a conocer CSS y sus elementos, se tiene que conocer cada uno de ellos y sobre todo a jugar, experimentar que puede salir con los colores, con las formas, con sus valores de propiedades, pues de esa manera es como se aprende y descubre la manera de crear piezas auténticas. Los colores no solo nos sirven para colorear, también ayudan a dar dimensión, personalidad, ilustración, matiz, sentido a lo que se está plasmando y cuando se conoce la manera de poder combinar, contrastar, enfatizar, profundizar, se conoce un mundo nuevo.

De tal manera, tanto simple o complejo, la declaración de propiedades permite involucrar no solo colores, también en el ámbito de fondos, se permite el anexo de imágenes, con diferentes maneras de involucrarlas, trabajando solamente con imágenes o combinar color e imagen.

Así como los colores son muy representativos, el tener una imagen dentro de una página web, promete ser un sitio agradable a la vista, pues como se mencionó anteriormente hoy en día el involucrarse con la tecnología resulta ser un aprendizaje visual, por ello es de suma importancia ser lo más llamativos e ilustrativos posibles.

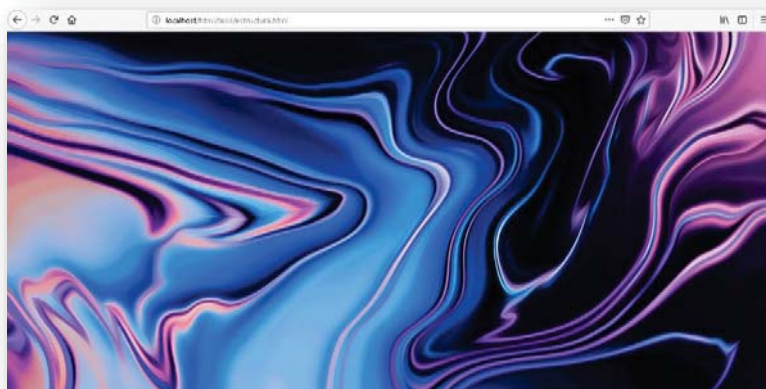


Ilustración 182 – Imagen propia, fondo tomado de <http://www.xperiaplay.com/wallpaper-de-la-semana-203-fondo-de-pantalla-de->

```
body{  
  background: url(../img/fondo.jpg);  
}
```

Ilustración 183 - Imagen propia

```
body{  
  background-image: url(../img/fondo.jpg);  
}
```

Ilustración 184 - Imagen propia

Una diferencia importante entre color e imágenes es que **BACKGROUND** con los colores es un valor absoluto, es decir, que abarca todo el tamaño de la pantalla sin ningún problema, de lo contrario las imágenes tienen un tamaño determinado, así como puede ser una imagen grande que cubra toda el área de la pantalla, como el ejemplo anterior de imagen, se cuenta con una imagen grande que cubre perfectamente el área de la pantalla, pues es una imagen de 1500 x 800 o sea pequeña y al extenderse genera deformación de imagen y corre el riesgo de pixelearse, para este problema en las imágenes, **BACKGROUND-IMAGE** se encarga de evitar esos problemas, pues cuando una imagen se manda de fondo y es pequeña, suele repetirse en toda la pantalla.

```
body{  
  background-image: url(../img/fondopequeño.jpg);  
}
```

Ilustración 185 - Imagen propia



Ilustración 186 - Imagen propia, fondo tomado de <http://photoshopeducacion.blogspot.com/>

Esta propiedad tiende a repetir la imagen hasta cubrir todo el espacio, por eso es recomendable conocer el tamaño de la imagen para que pueda adecuarse de mejor manera al tamaño de la pantalla, por otro lado, si las imágenes son pequeñas pueden ser sin fondo que permitan decorar un poco más la página.

```
body{
background-image: url(../img/fondo.png);
background-repeat: no-repeat;
background-position: right top;
}
```

Ilustración 187 - Imagen propia

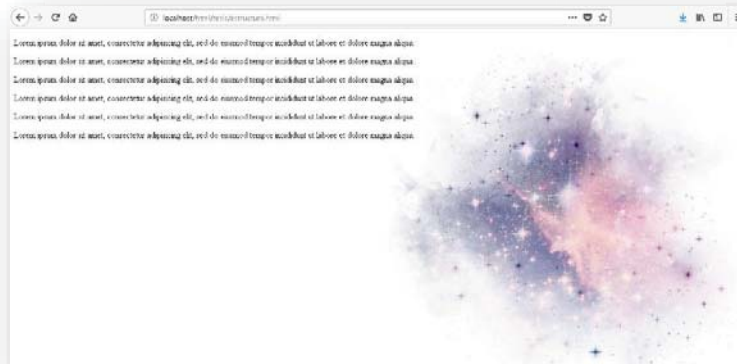


Ilustración 188 - Imagen propia, fondo tomado de <https://www.pinterest.com.mx/pin/711357703617755966/>

Como se puede observar, se tienen tres puntos a relucir, en primer punto la imagen que se presenta se maneja como una decoración, es realmente una imagen pequeña, el segundo punto son los elementos extras que se tienen en el código, las propiedades **BACKGROUND-REPEAT** y **BACKGROUND-POSITION**, sin estas dos propiedades se tendría el mismo resultado que el ejemplo erróneo anterior y como tercer punto los valores que se le asignan a cada propiedad.

La propiedad **BACKGROUND-REPEAT** tiene por objetivo definir hacia qué dirección repetir la imagen o evitar que se repita, sus valores son:

🚩 **REPEAT**: Es el valor predeterminado que arroja la propiedad.

```
body{
background-image: url(../img/fondopequeño.png);
background-repeat: repeat;
}
```

Ilustración 189 - Imagen propia

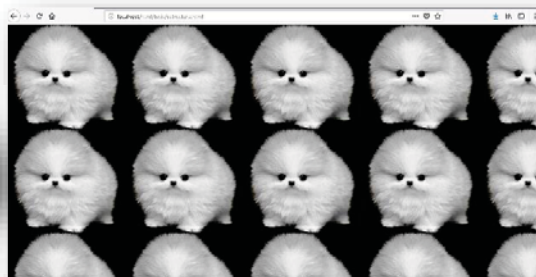


Ilustración 190 - Imagen propia, fondo tomado de <https://comoadiestraraunperro.com/razas-perros-miniatura-ninos/>

✚ **REPEAT-X**: Realiza la repetición de la imagen horizontalmente.

```
body{  
  background-image: url(../img/fondopequeño.png);  
  background-repeat: repeat-x;  
}
```

Ilustración 191 - Imagen propia

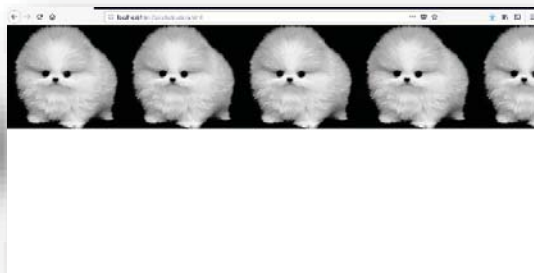


Ilustración 192 - Imagen propia, fondo tomado de <https://comoadiestraraunperro.com/razas-perros-miniatura-ninos/>

✚ **REPEAT-Y**: Realiza la repetición de la imagen verticalmente.

```
body{  
  background-image: url(../img/fondopequeño.png);  
  background-repeat: repeat-y;  
}
```

Ilustración 193 - Imagen propia

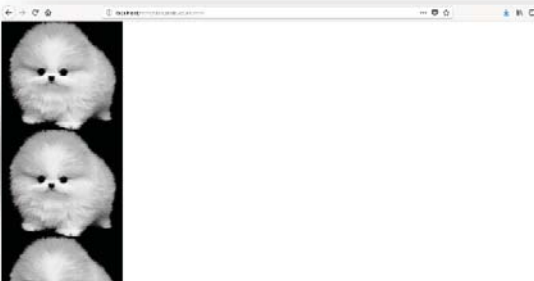


Ilustración 194 - Imagen propia, fondo tomado de <https://comoadiestraraunperro.com/razas-perros-miniatura-ninos/>

✚ **NO-REPEAT**: Deja una sola imagen a su tamaño original en la parte superior izquierda de la pantalla.

```
body{  
  background-image: url(../img/fondopequeño.png);  
  background-repeat: no-repeat;  
}
```

Ilustración 195 - Imagen propia



Ilustración 196 - Imagen propia, fondo tomado de <https://comoadiestraraunperro.com/razas-perros-miniatura-ninos/>

- ✚ **SPACE**: Se extiende en la pantalla sin perder forma, pero no rellena todo el espacio, entre cada repetición de imagen se genera un espacio en blanco.

```
body{  
  background-image: url(../img/fondopequeño.png);  
  background-repeat: space;
```

Ilustración 197 - Imagen propia



Ilustración 198 - Imagen propia, fondo tomado de <https://comoadiestraraunperro.com/razas-perros-miniatura-ninos/>

- ✚ **ROUND**: Su función es estirar y deformar la imagen para cubrir todo el espacio.

```
body{  
  background-image: url(../img/fondopequeño.png);  
  background-repeat: round;
```

Ilustración 199 - Imagen propia

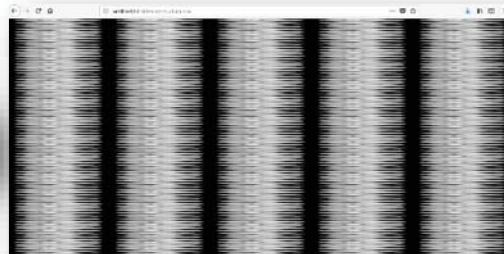


Ilustración 200 - Imagen propia, fondo tomado de <https://comoadiestraraunperro.com/razas-perros-miniatura-ninos/>

Pero esta propiedad también necesita de otras propiedades, el objetivo de estas tres propiedades es brindarle apoyo a la imagen pequeña para poderla posicionar de una manera que sirva de fondo, dando un toque personal o simplemente decorar de manera discreta la página web, pues al final son válidas todas las maneras en que se puede adaptar una página web.

Se comienza con la propiedad **BACKGROUND-ATTACHMENT**, se dedica a establecer de fondo la pantalla, atándola a la página o desplazarla en ella.

- ✚ **SCROLL**: Es el valor predeterminado de la propiedad, pues la imagen se desplaza junto con la página.
- ✚ **FIXED**: La función del valor es que la imagen se queda estática en el fondo, no se mueve, aunque la página sea larga y al desplazarse arriba o abajo, esta no se moverá.
- ✚ **LOCAL**: La imagen se desplaza con los elementos que contenga la página.

Conociendo esta propiedad, la segunda propiedad se llama **BACKGROUND-REPEAT** esta es importante para poder realizar un efecto de fondo más confortable, pues cuando se tiene una imagen pequeña suele ser un dolor de cabeza, pero ocupándola de una manera de “detalle” se puede lograr una armonía en la pantalla, la función de esta propiedad es posicionar la imagen de acuerdo a los valores que se le otorguen. Para ello se presentan las

diferentes posibilidades que se tienen, por supuesto solo se ocupa una, pero a manera de ejemplo se puede ver mejor visualmente:

✚ LEFT-TOP/ LEFT-CENTER/ LEFT-BOTTOM/ LEFT

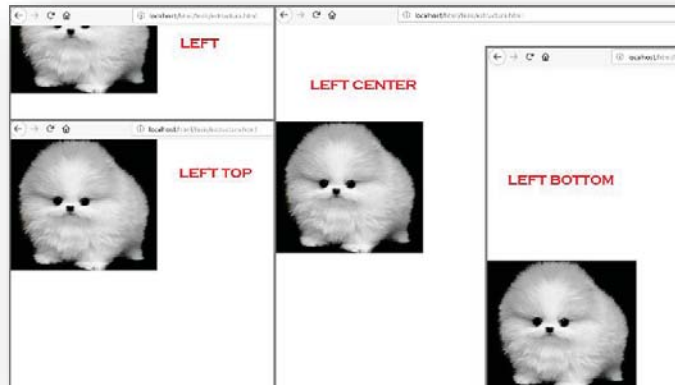


Ilustración 201 - Imagen propia, fondo tomado de <https://comoadiestraraunperro.com/razas-perros-miniatura-ninos/>

```
body{  
background-image: url(../img/fondopequeño.png);  
background-repeat: no-repeat;  
background-attachment: fixed;  
background-position: left / left top / left center / left bottom;
```

Ilustración 202 - Imagen propia

✚ CENTER-TOP/ CENTER-BOTTOM/ CENTER

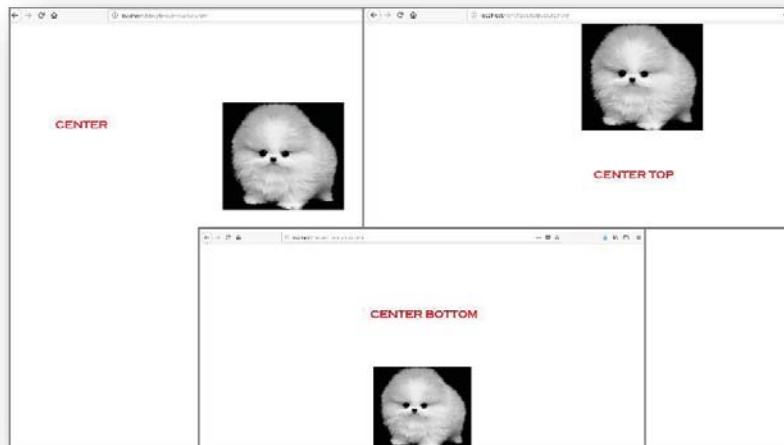


Ilustración 203 - Imagen propia, fondo tomado de <https://comoadiestraraunperro.com/razas-perros-miniatura-ninos/>

```
body{  
background-image: url(../img/fondopequeño.png);  
background-repeat: no-repeat;  
background-attachment: fixed;  
background-position: center / center top / center bottom;
```

Ilustración 204 - Imagen propia

✚ RIGHT-TOP/ RIGHT-CENTER/RIGHT-BOTTOM/ RIGHT

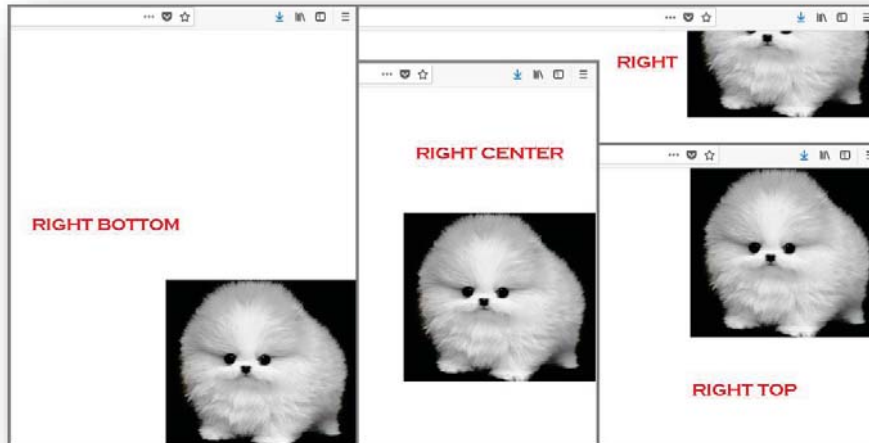


Ilustración 205 - Imagen propia, fondo tomado de <https://comoadiestraraunperro.com/razas-perros-miniatura-ninos/>

```
body{  
  background-image: url(../img/fondopequeño.png);  
  background-repeat: no-repeat;  
  background-attachment: fixed;  
  background-position: right / right top / right center / right bottom;
```

Ilustración 206 - Imagen propia

✚ X0% Y100%: Se posiciona mediante porcentaje.

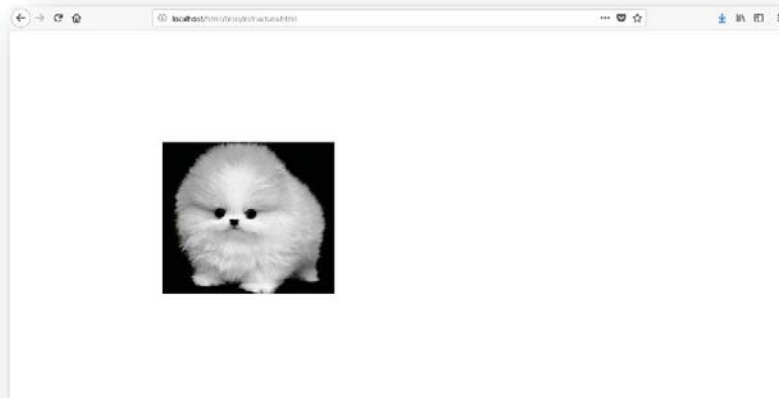


Ilustración 207 - Imagen propia, fondo tomado de <https://comoadiestraraunperro.com/razas-perros-miniatura-ninos/>

```
body{  
  background-image: url(../img/fondopequeño.png);  
  background-repeat: no-repeat;  
  background-attachment: fixed;  
  background-position: 25% 50%;
```

Ilustración 208 - Imagen propia

✚ **XPOS YPOS**: Se rigen por cualquier unidad de medida en CSS.

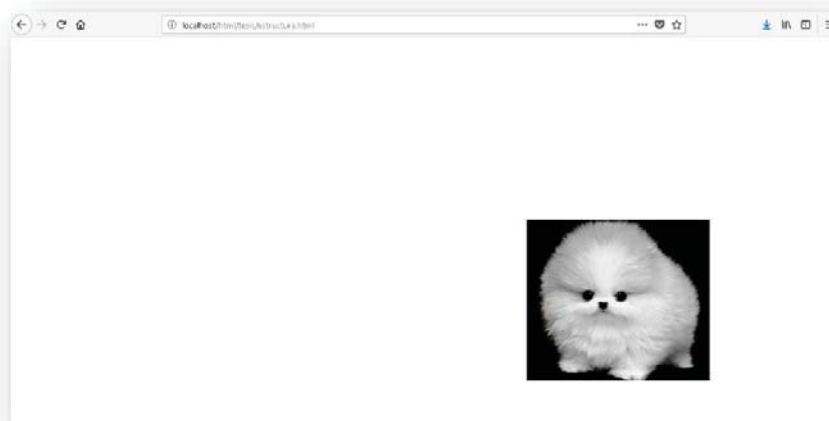


Ilustración 209 - Imagen propia, fondo tomado de <https://comoadiestraraunperro.com/razas-perros-miniatura-ninos/>

```
body{  
  background-image: url(../img/fondopequeño.png);  
  background-repeat: no-repeat;  
  background-attachment: fixed;  
  background-position: 850px 300px;
```

Ilustración 210 - Imagen propia

Todas las maneras de acomodar o posicionar una imagen son ideales para el trabajo que sea, pues al final no es una imagen complementaria, es solamente ilustración de fondo. Por supuesto también existen otras tres propiedades como **BACKGROUND-SIZE**, pues su función es indicarle a la imagen de fondo su tamaño utilizando las propiedades:

- ✚ **AUTO**: Este valor es predeterminado por la propiedad, pues arroja la imagen con el tamaño que tiene de origen al fondo.
- ✚ **WIDTH% LEPTH%**: Su función es otorgarle ancho y alto a la imagen de acuerdo al porcentaje que se le indique.
- ✚ **COVER**: Tiene el objetivo de ajustarse al elemento que esté dando fondo, puede extenderse sin deformarse, pero no se apreciaría la imagen por completo.
- ✚ **CONTEIN**: Este valor se asegura de cubrir el espacio y que la imagen se pueda apreciar por completo.

La segunda propiedad es **BACKGROUND-CLIP**, esta propiedad se encarga de que el fondo se extienda en un elemento, sus valores especifican hasta que limite se extenderá:

- ✚ **BORDER-BOX**: Este valor es el predeterminado que tiene la propiedad, pues su cobertura de fondo es después de la frontera que marca.
- ✚ **PADDING-BOX**: Este valor se extiende hasta un margen interno, es decir, no toca la frontera y se mantiene en un margen aún menor.
- ✚ **CONTENT-BOX**: Es el valor que cubre hasta la frontera del elemento que esté trabajando.

Y por último se tiene la propiedad **BACKGROUND-ORIGIN**, su función es especificar la posición de origen, es decir, toda imagen que se inserte dentro de una página web se ubicará en la esquina superior izquierda, esta propiedad bien podría ser utilizada para algún logo o imagen representante de alguna organización, pues su función es relativamente similar en todos sus valores.

- ✚ **PADDING-BOX**: Es el valor por defecto que brinda la propiedad, solo ubica a la imagen en la esquina superior izquierda de la pantalla con un borde de relleno o espacio en blanco.
- ✚ **BORDER-BOX**: Asigna a la imagen en la esquina superior izquierda del borde del contenido.
- ✚ **CONTENT-BOX**: Ubica a la imagen en la esquina superior izquierda del contenido.

Así como se puede observar las últimas dos propiedades, se manejan con cajas o bloques, como en **HTML5** todos sus elementos pueden ser muy utilizables dentro de cualquier elemento y asignarle un fondo especial.

Aprendido lo anterior, se puede mencionar que involucrar uno o más propiedades y elementos para crear un fondo auténtico, profesional y atractivo. Se pueden realizar con diferentes efectos con la propiedad de fondos:

```
body {
  background: plum url("../img/fondocorto.jpg") no-repeat fixed center;
}
```

Ilustración 211 - Imagen propia



Ilustración 212 - Imagen propia, fondo tomado de <https://www.elspectador.com/noticias/actualidad/los-mandalas-son-un-tema-religioso-articulo-657707>

```
#gifcolor {
  width: 300px;
  height: 200px;
  background: violet;
  animation: mymove 5s infinite;
}

@keyframes mymove {
  50% {background-color: deeppink;}
}
```

Ilustración 213 - Imagen propia

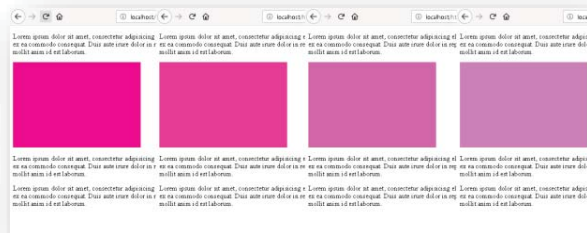


Ilustración 214 - Imagen propia

```
body {
  height:650px;
  background: linear-gradient(to right, rgba(206, 87, 173, 0), rgba(206, 87, 173, 1.0), rgba(206, 87, 173, 0)),
  url(..img/fondouno.png) no-repeat fixed right bottom, url(..img/fondodos.png) no-repeat fixed left center;
}
```

Ilustración 215 - Imagen propia



Ilustración 216 - Imagen propia, fondo tomado de <https://www.avon.mx/avon-mx/fashion/mandalas-mas-paz-menos-estres.html> y <https://www.kisspng.com/png-mandala-graphic-design-clip-art-drawing-image-clip-6729546/>

Ahora bien, el lado más artístico de **CSS** ya se conoció, aunque es importante mencionar que sin un fondo una página web no puede ser un éxito asegurado, para continuar con las demás propiedades se tiene que entender que **CSS** se encarga de transformar los elementos de **HTML5**, por tanto, cada elemento es una caja, en **CSS** existe la propiedad **BORDER**, su esencial propósito es envolver a cada elemento de **HTML5** en una caja, esto permite que por medio de otras propiedades se le asigne diseño a cada uno.

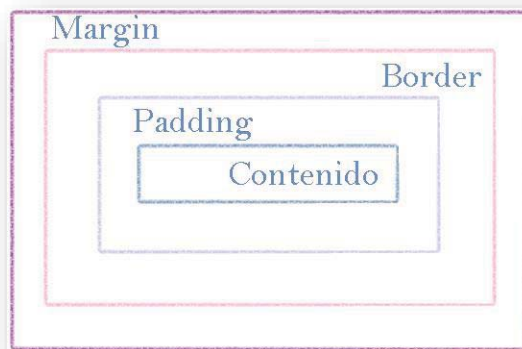


Ilustración 217 - Imagen propia

Para entenderlo un poco más, la imagen anterior representa el **MODELO DE CAJA**, un elemento de **HTML5** lo reflejamos como un contenido, todo contenido de **HTML5** tiene un margen, un borde y un relleno “invisible”, pues para **HTML5** no es visible ese espacio alrededor del elemento, pero con **CSS** si lo es, pues al momento de transformarlo con **CSS** el elemento se envuelve en una caja que permite que el margen, el borde y el relleno se

vuelvan visibles en **CSS** y poder trabajar con esos elementos para convertir al contenido en un elemento presentable en pantalla:

```
<div class="contenedor">
<p>Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod
tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam,
quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo
consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse
cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non
proident, sunt in culpa qui officia deserunt mollit anim id est laborum.</p>
</div>
```

Ilustración 218 - Imagen propia

Aquí se presenta un elemento **<DIV>** con un párrafo **<P>** y un texto cualquiera, para **HTML5** no existe algo más que el texto:

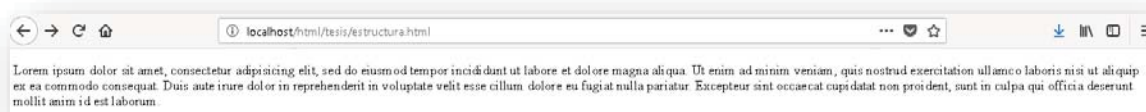


Ilustración 219 - Imagen propia

Pero en **CSS**, ocurre algo diferente, desata muchas propiedades y valores que se le pueden asignar:

```
div{
background-color: plum;
}
```

Ilustración 220 - Imagen propia

Se le asigna un color de fondo para identificar la caja:

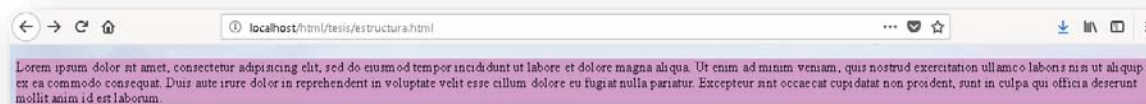


Ilustración 221 - Imagen propia

Hasta este momento ya se puede identificar el contenido marcado por el fondo en color plum y ya se encuentra dentro de una caja, como **CSS** ya identifico el margen y el relleno, se pueden modificar:

```
div{
background-color: plum;
margin: 20px;
padding: 20px;
}
```

Ilustración 222 - Imagen propia

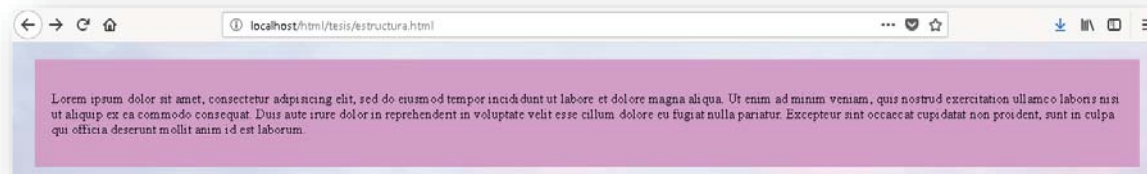


Ilustración 223 - Imagen propia

De esta forma, se puede identificar aún mejor los espacios que surgen en CSS, **PADDING** es el relleno que existe entre el contenido ya sea imagen o texto y la frontera de la caja, es decir, en el ejemplo que se muestra, el texto ahora está centrado en la misma distancia que existe entre el largo y ancho de la caja y su margen **MARGIN** es el que se refleja de la frontera de la caja al exterior, es decir, el contenido ahora se encuentra centrado en la parte externa como si flotara, esto sucede porque se le brindo un parámetro para que no se pegara a las paredes de la página.

```
div{  
  background-color: plum;  
  border: 10px solid black;  
  margin: 20px;  
  padding: 20px;  
}
```

Ilustración 224 - Imagen propia

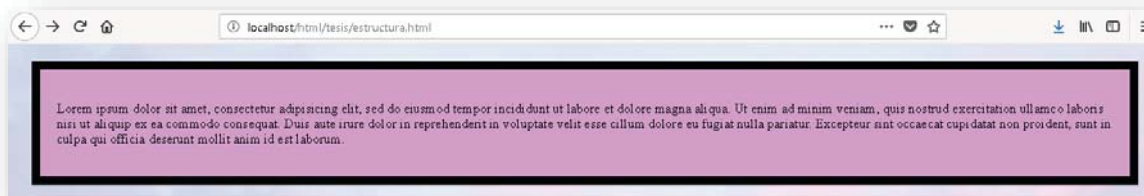


Ilustración 225 - Imagen propia

Ahora bien, para entender más a fondo el **MODELO DE CAJA**, se comenzará a conocer cada elemento que la compone, para poder identificar la caja donde se encuentra el contenido, se tiene la propiedad **BORDER** la cual permite hacer un grosor del borde de la caja, declarando su valor en sólido hará que sea visible y permitirá colorearlo y editarlo para fines visuales. De esta manera es como CSS trabaja en todos los elementos de **HTML5**, pero la propiedad **BORDER** desata muchos valores que permiten mejorar el aspecto de cualquier elemento, al igual que **MARGIN** y **PADDING**.

La propiedad **BORDER** puede ser modificada en tamaño, forma, color y estilo, es necesario entender que esta propiedad tiene explícitamente que comenzar con el estilo, pues de esta manera será visible en pantalla, de otra manera no aparecerá. Para conocerlo se debe mostrar de esta forma:

```
.contenedor {  
    border-style: solid;  
}
```

Ilustración 226 - Imagen propia

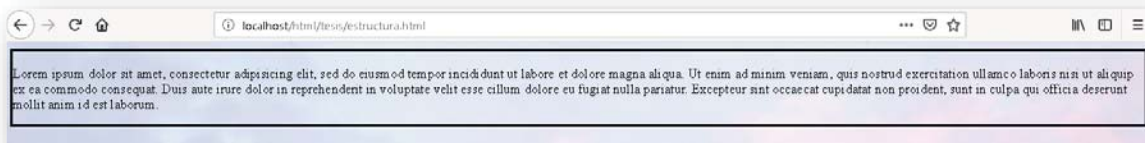


Ilustración 227 - Imagen propia

Esa es la forma más sencilla y básica de declarar un borde, por supuesto que existen diferentes opciones para poder observar un borde de caja, se muestran de la siguiente manera:

- **DASHED**: Aplica un borde de guiones.

```
.contenedor {  
    border-style: dashed;  
}
```

Ilustración 228 - Imagen propia

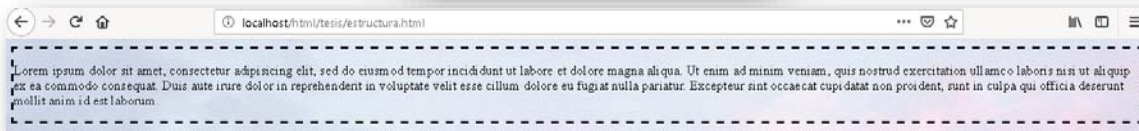


Ilustración 229 - Imagen propia

- **DOTTED**: Aplica un borde de puntos.

```
.contenedor {  
    border-style: dotted;  
}
```

Ilustración 230 - Imagen propia

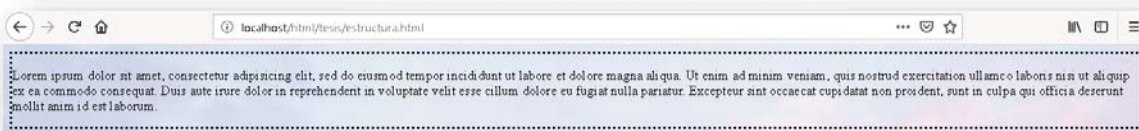


Ilustración 231 - Imagen propia

- **DOUBLE:** Aplica un borde doble.

```
.contenedor {  
  border-style: double;  
}
```

Ilustración 232 - Imagen propia

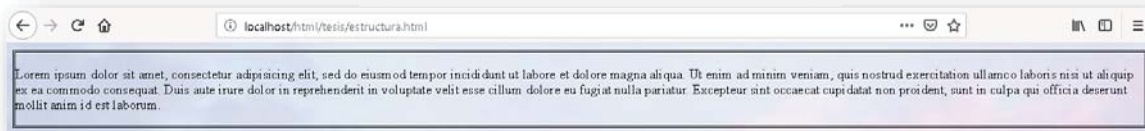


Ilustración 233 - Imagen propia

- **GROOVE:** Aplica un borde en 3D, simulado en pico con luz de lado inferior derecho diagonal.

```
.contenedor {  
  border-style: groove;  
}
```

Ilustración 234 - Imagen propia

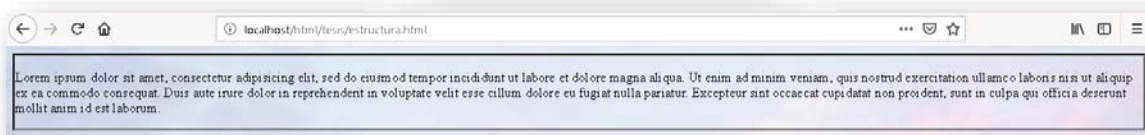


Ilustración 235 - Imagen propia

- **HIDDEN:** Aplica un borde oculto.

```
.contenedor {  
  border-style: hidden;  
}
```

Ilustración 236 - Imagen propia

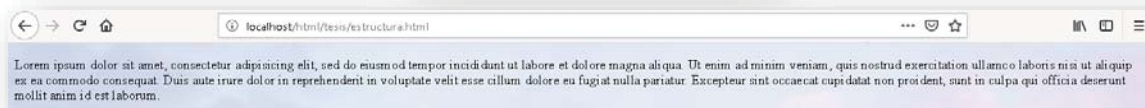


Ilustración 237 - Imagen propia

- **INSET:** Aplica un borde en 3D, define luz en plano de lado inferior derecho diagonal.

```
.contenedor {  
  border-style: inset;  
}
```

Ilustración 238 - Imagen propia

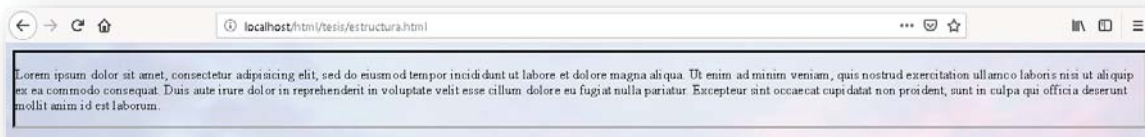


Ilustración 239 - Imagen propia

- **NONE:** No aplica borde.

```
.contenedor {  
  border-style: none;  
}
```

Ilustración 240 - Imagen propia

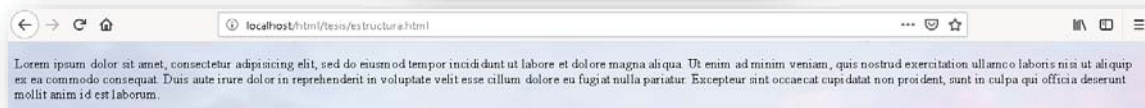


Ilustración 241 - Imagen propia

- **RIDGE:** Aplica un borde en 3D, simulado en pico con luz de lado superior izquierdo diagonal.

```
.contenedor {  
  border-style: ridge;  
}
```

Ilustración 242 - Imagen propia

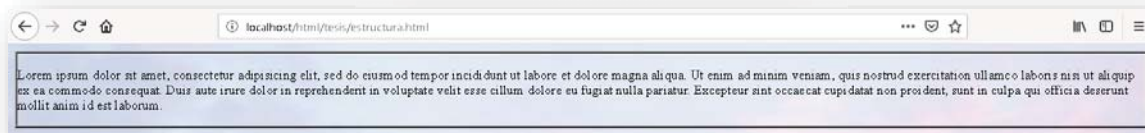


Ilustración 243 - Imagen propia

- **OUTSET:** Aplica un borde en 3D, define luz en plano de lado superior izquierdo diagonal.

```
.contenedor {  
  border-style: outset;  
}
```

Ilustración 244 - Imagen propia

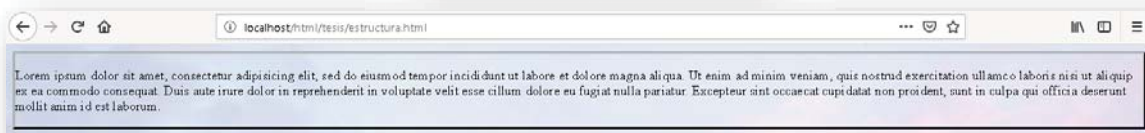


Ilustración 245 - Imagen propia

En este valor se puede hacer un mix de los valores que se presentaron anteriormente:

```
.contenedor {  
  border-style: groove double dotted dashed;  
}
```

Ilustración 246 - Imagen propia

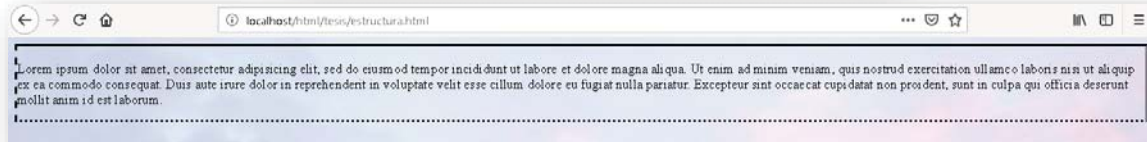


Ilustración 247 - Imagen propia

Ahora bien, para hacer lucir un borde, se necesita la siguiente propiedad **BORDER-WIDTH**, como su nombre lo dice, aplica grosor o anchura al borde, para así apreciar mejor el diseño del borde, se puede manejar de 3 diferentes maneras:

- Tamaño específico: En PX, EM, PT, CM, MM, etc.

```
.contenedor {  
  border-style: groove double dotted dashed;  
  border-width: 10px;  
}
```

Ilustración 248 - Imagen propia

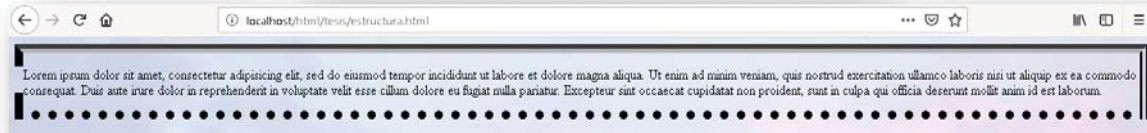


Ilustración 249 - Imagen propia

- Valores predefinidos: Fino (thin), Medio (medium), Grueso (thick).



Ilustración 250 - Imagen propia

- Mixto: Permite definir cuatro valores.

```
.contenedor {
  border-style: groove double dotted dashed;
  border-width: 15px 10px 8px 6px;
}
```

Ilustración 251 - Imagen propia

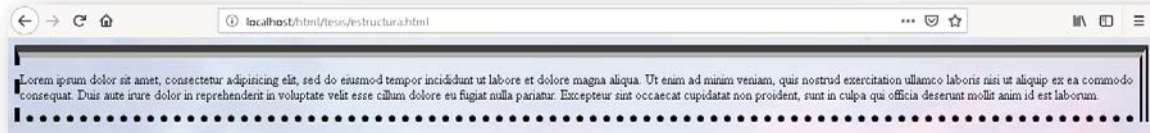


Ilustración 252 - Imagen propia

Ahora si se puede decir, que se aprecian mucho más los diseños, aunque no lucen del todo bien, pues falta algo más, falta ese elemento que les da vida a los objetos, eso le corresponde a la siguiente propiedad; **BORDER-COLOR**, esta puede adaptarse a cualquier medida de color:

```
.contenedor {
  border-style: groove double dotted dashed;
  border-width: 15px 10px 8px 6px;
  border-color: #a31359;
}
```

Ilustración 253 - Imagen propia

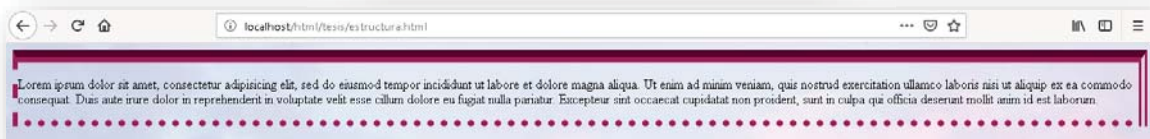


Ilustración 254 - Imagen propia

Y hacer combinaciones con los cuatro bordes que se tienen:

```
.contenedor {
  border-style: groove double dotted dashed;
  border-width: 15px 10px 8px 6px;
  border-color: #a31359 turquoise rgb(186,85,211) palevioletred;
}
```

Ilustración 255 - Imagen propia

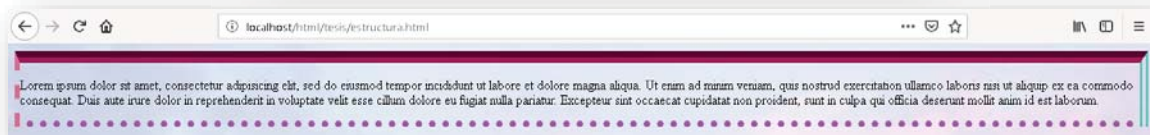


Ilustración 256 - Imagen propia

Para los bordes con estilo, es muy importante saber su dinámica, para no caer en conflicto con la siguiente propiedad, pues bien, cuando se declara un solo valor a **BORDER-STYLE**, este se aplica a los cuatro bordes o fronteras que se tienen (borde superior, borde derecho, borde inferior, borde izquierdo), cuando se declaran dos valores, estos se aplican como primer valor para borde superior e inferior y el segundo para borde derecho e izquierdo, si se declaran tres valores, el primero se aplica para borde superior, el segundo para borde derecho

e izquierdo y el tercero para borde inferior, para cuando se declaren cuatro valores, cada uno se aplica a cada uno de los bordes.

Entendido lo anterior, ahora se nos presentan propiedades aún más específicas como: **BORDER-TOP-STYLE** (Estilo de Borde Superior), **BORDER-RIGHT-STYLE** (Estilo de Borde Derecho), **BORDER-BOTTOM-STYLE** (Estilo de Borde Inferior), **BORDER-LEFT-STYLE** (Estilo de Borde Izquierdo), estas propiedades nos permiten asignar con mayor facilidad los estilos que se quieren para cada uno de los bordes, no existe ningún problema en utilizar cualquiera de las dos formas que se tienen para declarar un borde, pues ambas ofrecen la misma tarea.

Este elemento dentro de CSS también permite abreviar las declaraciones para evitar en sí tanto código, como se puede observar:

```
.contenedor {  
  border: 10px dotted #a31359;  
}
```

Ilustración 257 - Imagen propia

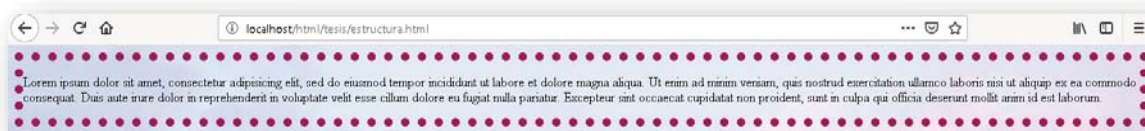


Ilustración 258 - Imagen propia

De esa manera se puede evitar declarar tanta propiedad y obtener un resultado muy agradable visualmente hablando. Pero un borde también puede ser modificado, con la siguiente propiedad **BORDER-RADIUS**, lo que permite eliminar las puntas de los bordes y redondearlas de acuerdo a la medida que se le indique.

```
.contenedor {  
  border: 10px dotted #a31359;  
  border-radius: 5px;  
}
```

Ilustración 259 - Imagen propia



Ilustración 260 - Imagen propia

Una parte muy esencial y complementaria de los bordes y de cualquier elemento, es la propiedad **MARGIN**, esta se utiliza para dar espacio a los elementos a su alrededor externo, se puede convertir en una propiedad abreviada sin ningún problema porque CSS identifica

como válidas ambas maneras de trabajar, por tanto, es importante saber las propiedades completas y como poder abreviarlas:

Las propiedades de **MARGIN** por separado permiten hacer más fácil la tarea, pues se tiene el control de la ubicación del margen que se quiere modificar:

- **MARGIN-TOP**: Asigna el margen en la parte superior del contenido.
- **MARGIN-BOTTOM**: Asigna el margen en la parte inferior del contenido.
- **MARGIN-LEFT**: Asigna el margen en la parte izquierda del contenido.
- **MARGIN-RIGHT**: Asigna el margen en la parte derecha del contenido.

Todas estas propiedades tienen los siguientes valores:

- El navegador calcula el margen y arroja uno default.
- Longitud especificada con medidas PX, EM, PT, CM etc.
- En porcentaje.

El que exista la opción de abreviar, ayuda mucho a reducir y optimizar código para cualquier modificación, ahora bien, para entender como abreviar el código se debe conocer la dinámica que mantiene **MARGIN**; se declaran 4 valores de medida, el primero se asigna al margen superior, el segundo al margen derecho, el tercero al margen inferior y el cuarto se asigna al margen izquierdo.

```
.contenedor {  
  border: 10px dotted #a31359;  
  border-radius: 5px;  
  margin: 20px 20px 20px 20px;  
}
```

Ilustración 261 - Imagen propia

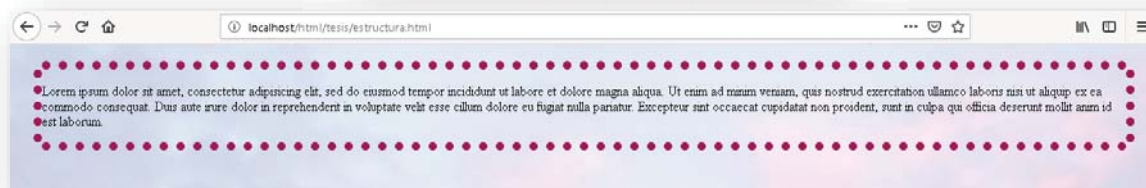


Ilustración 262 - Imagen propia

Si se declaran tres valores, el primero se aplica al margen superior, el segundo a los márgenes derecho e izquierdo y el tercero al margen inferior, en caso de declararse solo dos valores, el primero se aplica a los márgenes superior e inferior y el segundo a los márgenes derecho e izquierdo.

Como **MARGIN** existe otra propiedad que va mucho de la mano con esta, pues si hay un margen exterior debe existir uno interior, para ello se tiene **PADDING** esta propiedad es el relleno de los elementos, pues el valor que se aplique genera un espacio entre el borde de un elemento y el contenido.

```

.contenedor {
  border: 10px dotted #a31359;
  border-radius: 5px;
  margin: 20px 20px 20px 20px;
  padding: 30px;
}

```

Ilustración 263 - Imagen propia

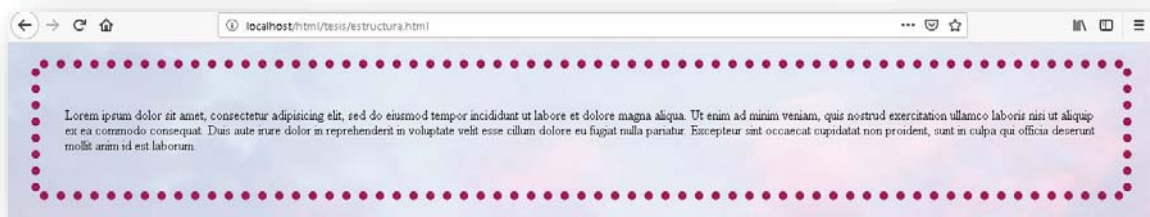


Ilustración 264 - Imagen propia

Esta propiedad dispone de las mismas reglas y elementos que **MARGIN**:

- **PADDING-TOP**: Asigna el margen en la parte superior del contenido.
- **PADDING-BOTTOM**: Asigna el margen en la parte inferior del contenido.
- **PADDING-LEFT**: Asigna el margen en la parte izquierda del contenido.
- **PADDING-RIGHT**: Asigna el margen en la parte derecha del contenido.

Todas estas propiedades tienen los siguientes valores:

- El navegador calcula el margen y arroja uno default.
- Longitud especificada con medidas PX, EM, PT, CM etc.
- En porcentaje.

Y si en su caso, cuando se declare esta propiedad abreviada, aplica la misma regla que se vio anteriormente en **BORDER** y **MARGIN**, si se declara un solo valor, este se aplica a los cuatro lados de **PADDING**, si se declaran dos valores, el primer valor se aplica al relleno superior e inferior y el segundo valor se aplica al relleno derecho e izquierdo, cuando exista una declaración de tres valores, se aplicará el primer valor para el relleno superior, el segundo para el relleno derecho e izquierdo y el tercero para el relleno inferior y cuando exista una declaración de cuatro valores, estos se aplicaran uno por uno a cada relleno.

Ya que se entendió la manera en que se manejan los elementos dentro de las cajas de **CSS**, se puede continuar con algo que probablemente es muy importante para cualquier Diseñador Web, y es que cuando se trata del texto, es hablar de una parte muy fundamental de una página web, pues es lo que la nutre a la misma, pero cuidado, porque si un texto es sumamente extenso, se puede cometer un error que cueste mucho, pues se vuelve muy tedioso y aburrido leer tanto texto sin chiste y la página no será consumida nunca más por el público, por tanto es muy importante hacer del texto llamativo y divertido para la audiencia.

Y se comienza con la propiedad **COLOR**, su función es asignarle un color al texto, se puede utilizar la medida de color que sea.

```
p{
  color: #01444e;
}
```

Ilustración 265 - Imagen propia



Ilustración 266 - Imagen propia

Después de aplicar color, se tiene que alinear el texto, este puede ser alineado a la izquierda, a la derecha, centrado o justificado dependiendo del estilo que lleve la página y eso se consigue con la propiedad **TEXT-ALIGN**.

```
p{
  color: #01444e;
  text-align: justify;
}
```

Ilustración 267 - Imagen propia



Ilustración 268 - Imagen propia

Como se pudo apreciar en el **HTML5**, con la etiqueta **<A>**, al introducir un texto, esta etiqueta automáticamente vuelve el texto a un hipervínculo o enlace, por lo cual adorna siempre el texto como si estuviese subrayado:

```
<a href="">Volver al Inicio</a>
```

Ilustración 269 - Imagen propia



Ilustración 270 - Imagen propia

Y como se está hablando de cómo editar el texto en CSS, esto incluye a los enlaces, por lo tanto, existe una propiedad que permite anular o mejorar la decoración de un enlace, o hacer que el texto resalte de entre los párrafos, y con eso se refiere a **TEXT-DECORATION**, cuenta con los siguientes valores:

- **NONE** (anula decoración del texto)
- **OVERLINE** (decora el texto con una línea arriba)
- **LINE-THROUGH** (tacha el texto)
- **UNDERLINE** (Subraya el texto)

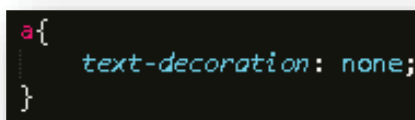


Ilustración 271 - Imagen propia



Ilustración 272 - Imagen propia

También se cuenta con la propiedad **TEXT-TRANSFORM**, esta permite cambiar el texto a mayúsculas o minúsculas dependiendo del diseño y motivo del texto, cuenta con 3 valores; **UPPERCASE** (Todo el texto en mayúsculas), **LOWERCASE** (Todo el texto en minúsculas) y **CAPITALIZE** (A toda palabra le asigna la inicial en mayúscula):

```

P{
  color: #01444e;
  text-align: justify;
  text-decoration: none;
  text-transform: uppercase;
}

```

Ilustración 273 - Imagen propia



Ilustración 274 - Imagen propia

Cuando se trata de un texto más formal, se cuenta con una propiedad **TEXT-INDENT** permite realizar una sangría en cada inicio de párrafo.

```

P{
  color: #01444e;
  text-align: justify;
  text-decoration: none;
  text-transform: uppercase;
  text-indent: 30px;
}

```

Ilustración 275 - Imagen propia

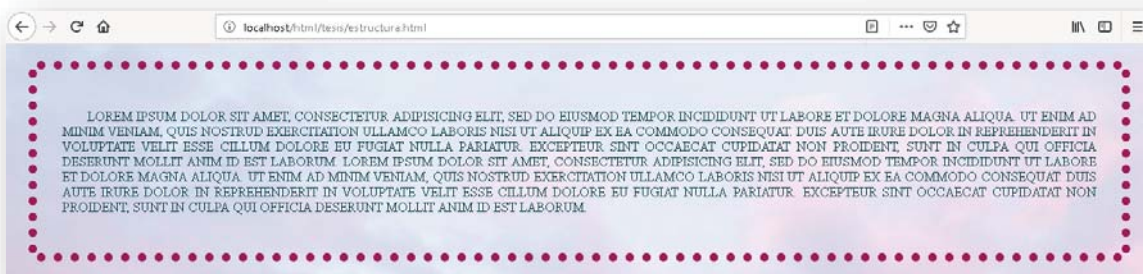


Ilustración 276 - Imagen propia

Para un formato de texto también se puede modificar el espaciado entre las letras, **LETTER-SPACING**, como nota extra, se permite utilizar números negativos para cerrar los espacios, dependiendo de lo que se busque tener como diseño de la página web.

```
P{
  color: #01444e;
  text-align: justify;
  text-decoration: none;
  text-transform: uppercase;
  text-indent: 30px;
  letter-spacing: 5px;
}
```

Ilustración 277 - Imagen propia

```
P{
  color: #01444e;
  text-align: justify;
  text-decoration: none;
  text-transform: uppercase;
  text-indent: 30px;
  letter-spacing: -3px;
}
```

Ilustración 278 - Imagen propia

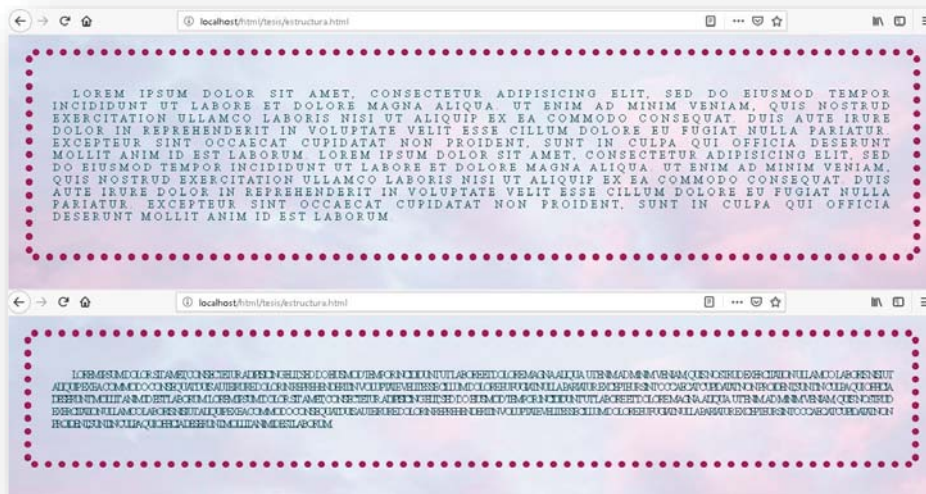


Ilustración 279 - Imagen propia

Y la propiedad **LINE-HEIGHT**, altera la altura del espacio entre los renglones.

```
P{
  color: #01444e;
  text-align: justify;
  text-decoration: none;
  text-transform: uppercase;
  text-indent: 30px;
  letter-spacing: 5px;
  line-height: 10px;
}
```

Ilustración 280 - Imagen propia

```
P{
  color: #01444e;
  text-align: justify;
  text-decoration: none;
  text-transform: uppercase;
  text-indent: 30px;
  letter-spacing: 5px;
  line-height: -10px;
}
```

Ilustración 281 - Imagen propia

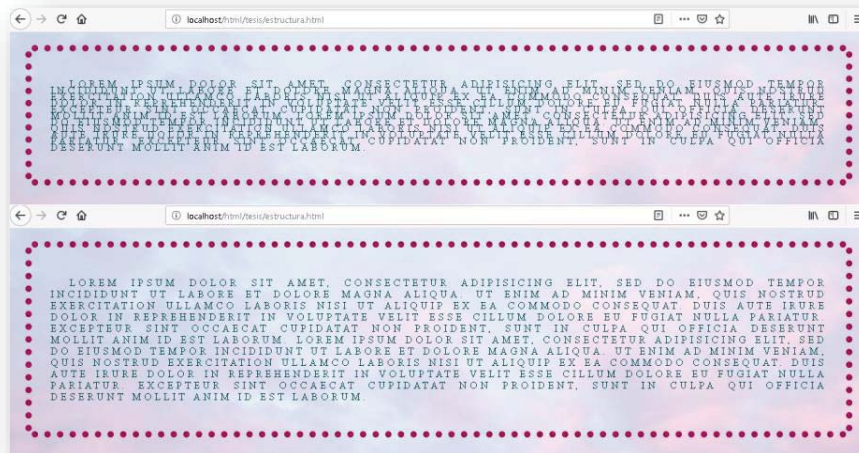


Ilustración 282 - Imagen propia

La propiedad **WORD-SPACING** permite hacer espacios, pero esta vez, entre cada palabra.

```

p{
  color: #01444e;
  text-align: justify;
  text-decoration: none;
  text-transform: uppercase;
  text-indent: 30px;
  letter-spacing: 5px;
  line-height: -10px;
  word-spacing: 5px;
}

```

Ilustración 283 - Imagen propia

```

p{
  color: #01444e;
  text-align: justify;
  text-decoration: none;
  text-transform: uppercase;
  text-indent: 30px;
  letter-spacing: 5px;
  line-height: -10px;
  word-spacing: -5px;
}

```

Ilustración 284 - Imagen propia

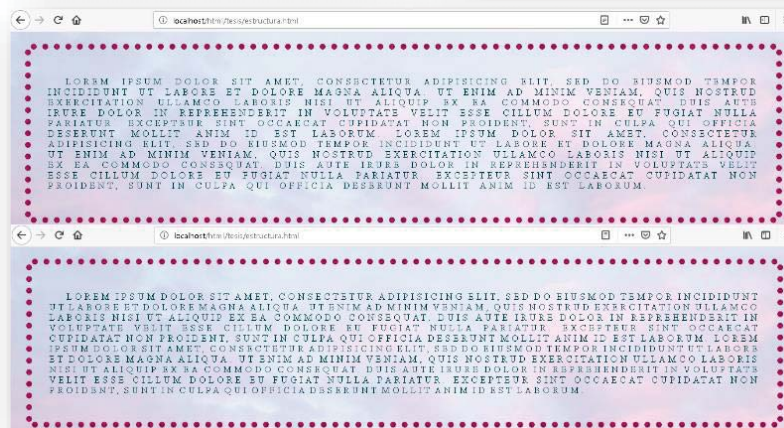


Ilustración 285 - Imagen propia

Y, por último, pero no menos importante, cuando se trata de un tema de diseño web austero, es decir, que aún no se conoce de tanta producción, existen efectos en el texto que le brinda un poco más de visualización. Para ello se tiene la propiedad **TEXT-SHADOW**, la cual permite adherirles sombras a las letras para causar más profundidad, esto puede ocuparse mejor en un título o subtema para resaltar.

```
h1{
  text-align: center;
  text-shadow: 3px 2px #a31359;
}
```

Ilustración 286 - Imagen propia



Ilustración 287 - Imagen propia

Como se puede observar en el código, se encuentran dos medidas, la primera especifica el movimiento de izquierda a derecha y el segundo de arriba abajo, por supuesto aquí también existe la regla de medidas negativas, pues se puede ubicar la sombra en donde convenga según el diseño, y como tercer y último punto, es importante declarar el color que se le quiera dar a la sombra, para así dejar un poco más de personalidad al sitio web.

Entendiendo el conjunto de propiedades que existen para el texto, ahora se puede comenzar a cambiar la fuente del texto, es decir, la tipografía que se utilizará para cambiar la apariencia en pantalla. Dentro de CSS se rigen por dos familias de fuentes:

- Familia tipográfica: Es la letra que tiene un nombre específico, por ejemplo "Arial", "Verdana", "Time New Roman" etc.
- Familia genérica tipográfica: Son las fuentes que hacen referencia al estilo del tipo de letra. Como, por ejemplo: serif (tipo de letra similar a Times New Roman), sans-serif (tipo Arial), cursive (tipo Comic Sans), fantasy (tipo Impact) y monospace (tipo Courier New).

Y para aplicarle un tipo de fuente diferente al texto, se hace de la siguiente manera:

```
p{
  color: #01444e;
  text-align: justify;
  text-indent: 30px;
  font-family: "Comic Sans MS", cursive, sans-serif;
}
```

Ilustración 288 - Imagen propia

Se utiliza la propiedad **FONT-FAMILY**, esta nos permite ingresar de uno a tres valores, por supuesto que te preguntas ¿Por qué?, bueno, lo que sucede es que todos los navegadores son compatibles con estos tipos de fuentes, pero por ciertas razones desconocidas el navegador no reconoce la fuente y muestra la que el navegador da default, por ello se utilizan en orden, la primera es la que se desea proyectar, la segunda y tercera con fuentes genéricas que permiten dar un cambio visual a la letra en caso de que la primera no sea reconocida, esto es importante por si existe cualquier error del navegador.

Ahora bien, los cambios técnicos que puede tener una fuente, son el tamaño de la fuente esto se aplica con la propiedad **FONT-SIZE**, tiene valores absolutos como:

- **XX-SMALL**
- **X-SMALL**
- **SMALL**
- **MEDIUM**
- **LARGE**
- **X-LARGE**
- **XX-LARGE**

Y también se cuenta con las medidas de unidad que se conocieron anteriormente. Ahora bien, aunque se tienen muchas medidas, cabe destacar que para el cambio de tamaño puede ser en la unidad de medida que sea de la preferencia del diseñador, pero algo muy importante es mantener la fidelidad de la página web, por tanto, sería recomendable utilizar la unidad **EM**, ya que esta se encuentra basada en el tamaño de la letra, y, por ende, mantendrá el tamaño de la letra en caso de que exista un zoom en pantalla de computador o dispositivo celular y la calidad de imagen será igual que la original.

```
p{
  color: #01444e;
  text-align: justify;
  text-indent: 30px;
  font-family: "Comic Sans MS"
  font-size: 2em;
}
```

Ilustración 289 - Imagen propia

Lorem ipsum

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum. Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Ilustración 290 - Imagen propia

Por otro lado, se tiene el cambio de volumen o grosor de la fuente y para ello se necesita la propiedad **FONT-WEIGHT**, en esta propiedad solo aplican dos valores: **NORMAL** y **BOLD**, aunque se podría decir que solo es un valor, ya que **NORMAL** es el valor default que tiene la fuente en el navegador, el único valor que realmente aplicaría un cambio es **BOLD**, pero no está por demás mencionar ambos valores.

```
P{
  color: #01444e;
  text-align: justify;
  text-indent: 30px;
  font-family: "Comic Sans MS", cursive, sans-serif;
  font-size: 2em;
  font-weight: bold;
}
```

Ilustración 291 - Imagen propia

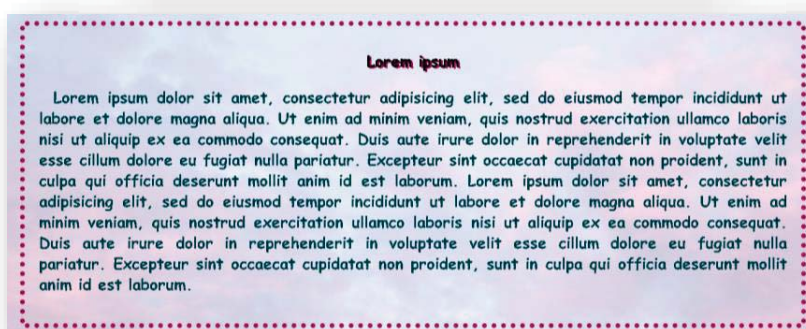


Ilustración 292 - Imagen propia

Y por supuesto, existe también estilo para la fuente, esta tarea le pertenece a la propiedad **FONT-STYLE**, al igual que la propiedad anterior, tiene solo dos valores: **CURSIVA (ITALIC)** y **NORMAL**, pero como ya se mencionó, el término **NORMAL** significa el valor default de una fuente.

```
P{
  color: #01444e;
  text-align: justify;
  text-indent: 30px;
  font-family: "Comic Sans MS", cursive, sans-serif;
  font-size: 2em;
  font-weight: bold;
  font-style: italic;
}
```

Ilustración 293 - Imagen propia

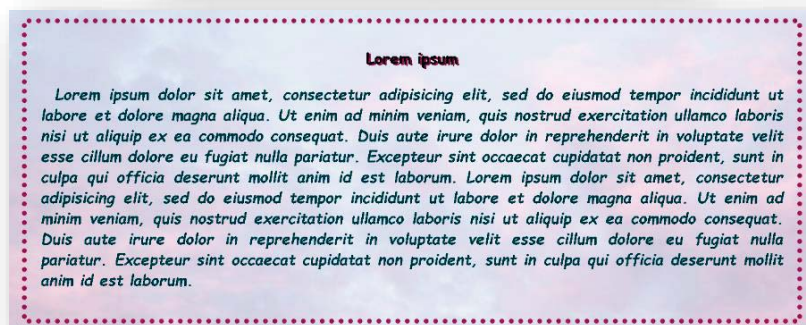


Ilustración 294 - Imagen propia

Algo muy increíble de las fuentes es que no se cierran a las que tiene instaladas el ordenador, es decir, aquellas fuentes que se encuentran en WORD, son las que CSS contiene dentro de su código, por ello es que los navegadores reconocen o no la fuente que se le está aplicando, aclarado ese punto, las fuentes pueden adaptarse incluso a fuentes web, con esto se refiere a que se puede descargar una fuente de algún sitio web, basta con tener el archivo con extensión (TTF, OTF Y WOFF), guardarla en una carpeta de fuentes y con el siguiente código poder disponer de la fuente.

```
@font-face{
  font-family: miFuente;
  src: url(../fonts/letra.ttf);
}
```

Ilustración 295 - Imagen propia

La regla de la fuente es que primero se debe señalar que se creará una fuente nueva, por ello se utiliza dentro de CSS **@FONT-FACE**, en la primera propiedad, se declarará el nombre de la fuente, cual sea porque tú la estas adjuntando y en la segunda propiedad se declara la dirección de árbol que tiene el archivo.

```
p{
  color: #01444e;
  text-align: justify;
  text-indent: 30px;
  font-family: miFuente;
  font-size: 2em;
  font-weight: bold;
  font-style: italic;
}
```

Ilustración 296 - Imagen propia

Después en el elemento que vas a ocupar la fuente, basta con poner el nombre en la propiedad **FONT-FAMILY**, como se mostró en el inicio de la declaración de fuente, con eso se obtendrá el resultado que elegiste.



Ilustración 297 - Imagen propia

Es importante saber que esta regla solo puede contener una fuente, así que, si se quieren adjuntar más de una fuente, se tendrá que repetir la misma regla.

Y cuando de reglas se trata en **CSS**, existe una muy importante que no se puede pasar por nada del mundo, a lo largo de este trabajo se ha mencionado que **CSS** edita elementos, pero que pasaría si un elemento se encuentra dentro de otro y son varios, ¿Qué se haría en ese caso?, se supone que acá se está hablando de hacer más óptimo el código, haciéndolo más ligero de leer y comprender para futuras actualizaciones, por ende, si comenzamos a editar uno por uno los elementos se harían miles de líneas de código y jamás se terminaría.

Para ese problema existen combinaciones de elementos, ¿A qué se refiere esto?, bueno, cuando hay un elemento dentro de otro, para poder realizar la edición de ese elemento, se formula una combinación de dichos elementos, para ello existen cuatro tipos de combinaciones:

- ✓ **Combinador descendiente:** Como su nombre lo dice, esta combinación permite nombrar un selector **A**, este es el contenedor de muchos otros elementos y cuando esos elementos son de la misma categoría, al nombrarlo permite hacer un cambio no en el primer selector, si no en el segundo.

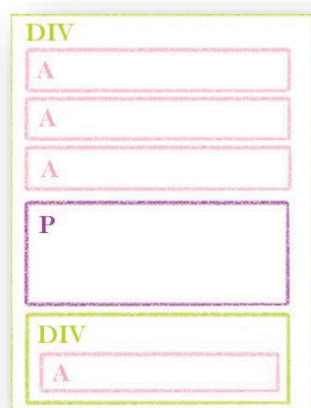


Ilustración 298 - Imagen propia

Para comprenderlo aún mejor, se puede explicar con el ejemplo de la imagen anterior, imagina que tienes un **<DIV>** y este funge como contenedor de seis elementos más, tres de ellos serán enlaces directos **<A>**, el cuarto será un párrafo directo **<P>** y hay otro **<DIV>** con un enlace **<A>** dentro, ahora bien, fuera de ese **<DIV>** se tienen otros enlaces y otros elementos complementarios de la página web, entonces, el objetivo es editar los enlaces que están dentro del **<DIV>** contenedor, aquí es donde surge un gran problema, porque si dentro de **CSS** se nombra al elemento **A** que es el respectivo elemento de enlaces, entonces todo lo que se elija de diseño, se aplicará a todos los enlaces, tanto los que están fuera como dentro del **<DIV>** contenedor, para evitar eso, existe la combinación descendente, pues si nombras dentro de **CSS** al elemento **DIV** y con un espacio nombras al elemento **A**, obtendrás un resultado exitoso, pues lo único que tendrá un cambio, serán los enlaces que se encuentran dentro de **<DIV>** contenedor, incluido el que está dentro del otro **<DIV>**.

```
div a{
  color: black;
  font-family: verdana;
}
```

Ilustración 299 - Imagen propia

- ✓ Combinador selector Hijo: Esta combinación tiene por objetivo seleccionar al elemento contenedor, es decir, el contenedor “padre” y el elemento hijo, es todo aquel elemento directo que se encuentra dentro del contenedor padre, y ambos son separados por el signo “>”.

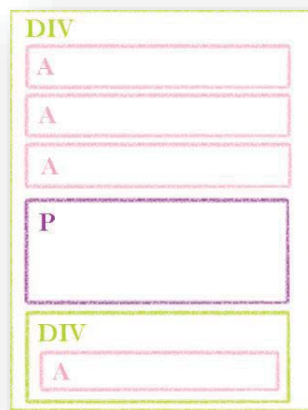


Ilustración 300 - Imagen propia

Para comprenderlo aún mejor, se puede explicar con el ejemplo de la imagen anterior, imagina que tienes un **<DIV>** y este funge como contenedor de seis elementos más, tres de ellos serán enlaces directos **<A>**, el cuarto será un párrafo directo **<P>** y hay otro **<DIV>** con un enlace **<A>** dentro, ahora bien, el objetivo es editar los enlaces hijos que están dentro del **<DIV>** contenedor, entonces si nombras dentro de **CSS** al elemento **DIV** y con el signo “>” nombras al elemento **A**, obtendrás un cambio en los primeros tres enlaces, pues ellos tienen el papel hijo, y si te preguntas porque el ultimo enlace no cambio, es porque es enlace se encuentra dentro de otro elemento, y funge como nieto, por tanto, no funcionan los cambios en ese elemento.

```
div > a{
  color: yellow;
  font-family: verdana;
}
```

Ilustración 301 - Imagen propia

- ✓ Combinador hermano adyacente: Esta combinación funciona seleccionando al elemento hermano y con el signo “+” se selecciona al elemento hermano adyacente que se quiere afectar.



Ilustración 302 - Imagen propia

Para comprenderlo aún mejor, se puede explicar con el ejemplo de la imagen anterior, imagina que tienes un **<DIV>** que funge como el contenedor de todo, evidentemente pasa a ser padre y después tienes otros tres elementos que fungen como hijos hablando jerárquicamente y a la vez como hermanos hablando de igualdad de posición, cuando se aplica la combinación de hermano adyacente, en primera instancia los hermanos deben pertenecer al mismo padre por obvias razones, y cuando se nombra, en este caso a **DIV HERMANO**, con esa acción, valga la redundancia, está seleccionando a ese primer hermano, con el signo “+” enlaza al siguiente elemento que es **P** porque es el hermano siguiente, es decir, el hermano adyacente, y por ende, es el elemento que va a salir afectado.

```
div#papa div#hermano + p{
  color: yellow;
  font-family: verdana;
}
```

Ilustración 303 - Imagen propia

- ✓ Combinador hermano general: Este combinador funciona seleccionando a todos los elementos que sean hermanos de un elemento en general.

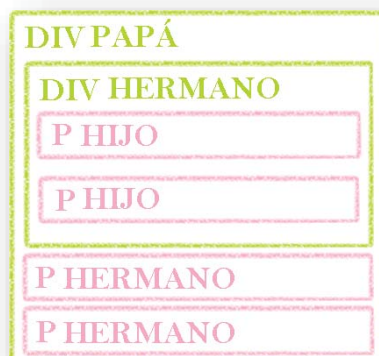


Ilustración 304 - Imagen propia

Para comprenderlo aún mejor, se puede explicar con el ejemplo de la imagen anterior, imagina que tienes un **<DIV>** que funge como el contenedor de todo, evidentemente pasa a ser padre y después tienes otros tres elementos que funcionan como hijos hablando jerárquicamente y a la vez como hermanos hablando de igualdad de posición, cuando se aplica la combinación de hermano general, en primera instancia los hermanos deben pertenecer al mismo padre por obvias razones, y cuando se nombra, en este caso a **DIV HERMANO**, con esa acción, valga la redundancia, está seleccionando a ese primer hermano, con el signo “~” enlaza al siguiente elemento que es **P**, esa acción afectará a todo elemento hermano de **<DIV HERMANO>** excepto a él.

```
div#papa div#hermano ~ p{  
  color: yellow;  
  font-family: verdana;  
}
```

Ilustración 305 - Imagen propia

Hasta este punto, ya se aprendió sintaxis, medidas de unidad y color, fondos, bordes, espacios, texto y fuentes, precisamente con esos conocimientos ya se puede avanzar con el resto de propiedades sin tener que explicar nuevamente a lo que se dedica cada una de ellas, lo cual nos permite jugar con todo eso para obtener resultados novedosos y creativos, por ello, se continuará con enlaces, tablas, listas e imágenes, que son parte fundamental dentro de un sitio web, pues bien un enlace, tiene el objetivo de transportarnos de un sitio a otro con el alcance de un solo clic, y su comportamiento dentro del mundo en **CSS** es muy amigable al cambio.

Se mantendrá con un tipo de fuente, tamaño y alineación.

```
#enlace1, #enlace2, #enlace3, #enlace4{  
  font-family: miFuente;  
  font-size: 2em;  
  text-align: center;  
  margin: 10px;  
}
```

Ilustración 306 - Imagen propia



Ilustración 307 - Imagen propia

Cuando se le otorga un color diferente al texto de un enlace, ya existe un cambio por sí solo.

```
#enlace1 a{
  color: #1b957d;
}
```

Ilustración 308 - Imagen propia



Ilustración 309 - Imagen propia

Se puede convertir en un botón con cuatro arreglitos.

```
#enlace2 a{
  border: 3px solid #fff;
  background-color: #36df00;
  color: white;
  text-decoration: none;
}
```

Ilustración 310 - Imagen propia



Ilustración 311 - Imagen propia

Y de manera sencilla, puede culminar en un cambio importante, pero no solo existe cambio de apariencia, para un enlace permite ver más allá, pues su función es enlazar, es más dinámico, pues eso es lo que también se puede lograr.

Existen estados interactivos para los enlaces, los cuales permiten que el enlace tenga una apariencia, pero al poner el apuntador encima cambia su apariencia, esto permite hacer más llamativa la página web.

- ❖ **A:LINK** – Indica que el enlace no ha sido visitado (valor default).
- ❖ **A:VISITED** – Indica que el enlace ya fue visitado.
- ❖ **A:HOVER** – Indica que el apuntador se encuentra sobre el enlace sin acción.
- ❖ **A:ACTIVE** – Indica que el apuntador hizo clic en el enlace.

Y en el orden que se presentan, es recomendable aplicarlos así dentro del código.



Ilustración 312 - Imagen propia

Si recordarás en el inicio de este proyecto, en **HTML5** se habló de la diferencia que existe entre un enlace y un link, pero dentro de **CSS** en la parte estética eso no importa, puesto que un link lo puedes construir y transformar en un botón sin ningún problema, mientras que un botón sigue luciendo como un botón, es importante saber que para temas de diseño, un link no mantiene un solo patrón o estilo, pues se permite ser versátil y adaptarse a cualquier estilo de maquetación web.

Ahora bien, dentro del campo de las listas en **CSS**, es relativamente poco, pues este elemento no se presta tanto para una transformación del elemento, pues no permite una adaptación como tal en una página web, si se quiere ver un poco más específico, solo se estiliza la lista. Para comenzar un cambio visualmente hablando a una lista, ya sea ordenada o no, se puede observar los valores default que da el elemento **** y **** en **HTML5**, es decir, los marcadores que arroja, pues son aquellas figuras/signos/símbolos que se prestan para identificar los elementos que conforman la lista, por ende, cambiar ese signo es uno de los objetivos que tienen **CSS** para ese elemento, para poder asignarle un nuevo marcador a las listas se necesita de la siguiente propiedad **LIST-STYLE-TYPE**, la cual contiene 24 valores distintos.

✓ **NONE:**

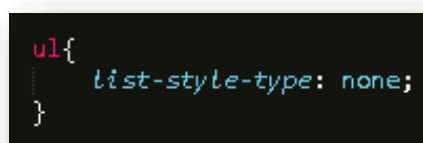


Ilustración 313 - Imagen propia

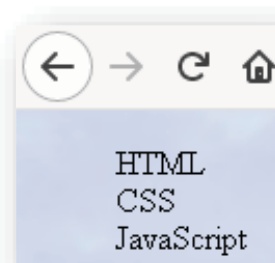


Ilustración 314 - Imagen propia

✓ **CIRCLE:**

```
ul{  
  list-style-type: circle;  
}
```

Ilustración 315 - Imagen propia



Ilustración 316 - Imagen propia

✓ **DISC:**

```
ul{  
  list-style-type: disc;  
}
```

Ilustración 317 - Imagen propia



Ilustración 318 - Imagen propia

✓ **SQUARE:**

```
ul{  
  list-style-type: square;  
}
```

Ilustración 319 - Imagen propia



Ilustración 320 - Imagen propia

✓ **DECIMAL:**

```
ul{  
  list-style-type: decimal;  
}
```

Ilustración 321 - Imagen propia

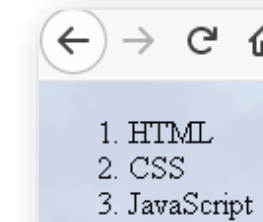


Ilustración 322 - Imagen propia

✓ **DECIMAL-LEADING-ZERO:**

```
ul{  
  list-style-type: decimal-leading-zero;  
}
```

Ilustración 323 - Imagen propia

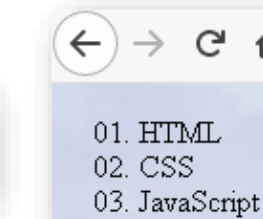


Ilustración 324 - Imagen propia

✓ **LOWER-ALPHA / LOWER-LATIN:**

```
ul{  
  list-style-type: lower-alpha;  
}
```

Ilustración 325 - Imagen propia

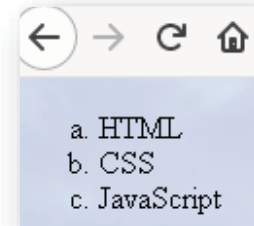


Ilustración 326 - Imagen propia

✓ **UPPER-ALPHA / UPPER-LATIN:**

```
ul{  
  list-style-type: upper-alpha;  
}
```

Ilustración 327 - Imagen propia

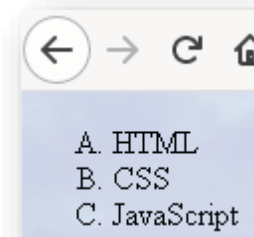


Ilustración 328 - Imagen propia

✓ **LOWER-GREEK:**

```
ul{  
  list-style-type: lower-greek;  
}
```

Ilustración 329 - Imagen propia

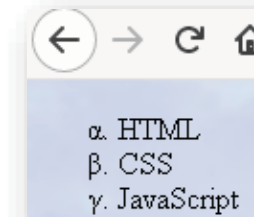


Ilustración 330 - Imagen propia

✓ **LOWER-ROMAN:**

```
ul{  
  list-style-type: lower-roman;  
}
```

Ilustración 331 - Imagen propia



Ilustración 332 - Imagen propia

✓ **UPPER-ROMAN:**

```
ul{  
  list-style-type: upper-roman;  
}
```

Ilustración 333 - Imagen propia

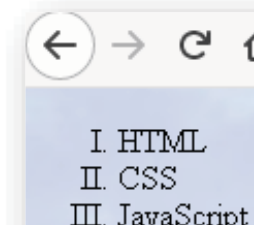


Ilustración 334 - Imagen propia

✓ **ARMENIAN:**

```
ul{  
  list-style-type: armenian;  
}
```

Ilustración 335 - Imagen propia

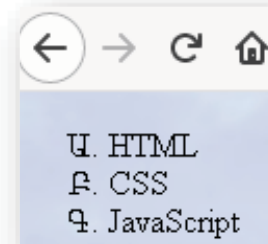


Ilustración 336 - Imagen propia

✓ **GEORGIAN:**

```
ul{  
  list-style-type: georgian;  
}
```

Ilustración 337 - Imagen propia



Ilustración 338 - Imagen propia

✓ **HEBREW:**

```
ul{  
  list-style-type: hebrew;  
}
```

Ilustración 339 - Imagen propia



Ilustración 340 - Imagen propia

✓ **CJK-IDEOGRAPHIC:**

```
ul{  
  list-style-type: cjk-ideographic;  
}
```

Ilustración 341 - Imagen propia

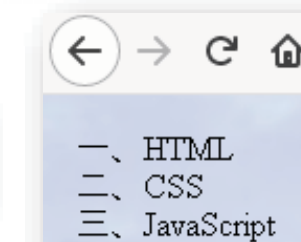


Ilustración 342 - Imagen propia

✓ **CJK-EARTHLY-BRANCH:**

```
ul{  
  list-style-type: cjk-earthly-branch;  
}
```

Ilustración 343 - Imagen propia

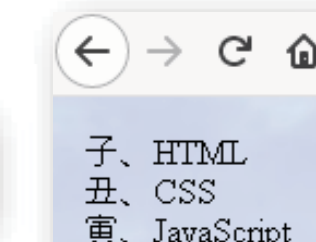


Ilustración 344 - Imagen propia

✓ **CJK-HEAVENLY-STEM:**

```
ul{  
  list-style-type: cjk-heavenly-stem;  
}
```

Ilustración 345 - Imagen propia



Ilustración 346 - Imagen propia

✓ **HIRAGANA:**

```
ul{  
  list-style-type: hiragana;  
}
```

Ilustración 347 - Imagen propia

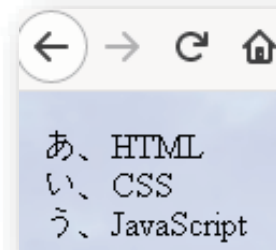


Ilustración 348 - Imagen propia

✓ **KANNADA:**

```
ul{  
  list-style-type: kannada;  
}
```

Ilustración 349 - Imagen propia

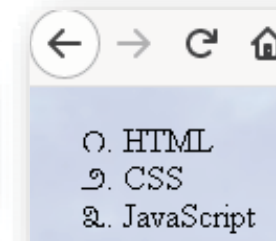


Ilustración 350 - Imagen propia

✓ **KATAKANA:**

```
ul{  
  list-style-type: katakana;  
}
```

Ilustración 351 - Imagen propia



Ilustración 352 - Imagen propia

✓ **HIRAGANA-IROHA:**

```
ul{  
  list-style-type: hiragana-iroha;  
}
```

Ilustración 353 - Imagen propia

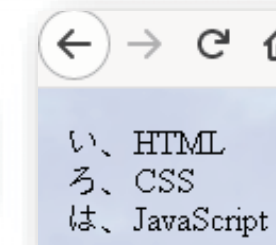


Ilustración 354 - Imagen propia

✓ **KATAKANA-IROHA:**

```
ul{  
  list-style-type: katakana-iroha;  
}
```

Ilustración 355 - Imagen propia

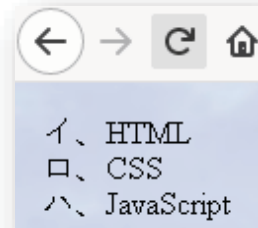


Ilustración 356 - Imagen propia

Para este tipo de propiedad, se le puede aplicar cualquier valor a ambas listas, ya que es indistinto y por ello, el modo en que se quiera aplicar funciona de igual forma, eso ya depende del diseñador.

Sin embargo, para CSS es primordial tener una visualidad ilustrativa, más allá de lo que nos permitan los símbolos, también se permiten adjuntar imágenes como marcadores, para ello se hace uso de la propiedad **LIST-STYLE-IMAGE** su valor hace referencia al URL de la imagen.

```
ul{  
  list-style-image: url('../img/corazon.png');  
}
```

Ilustración 357 - Imagen propia

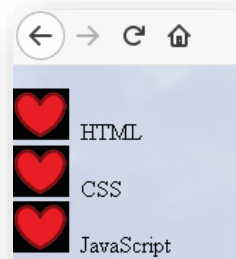


Ilustración 358 - Imagen propia

Otra opción que se tiene dentro de los marcadores es definir su posición, como se puede observar en los ejemplos anteriores, los marcadores siempre se encuentran fuera del cuadro del texto, es decir, que están alineados al exterior del texto, pero esto puede cambiar, depende del gusto del diseñador, pero eso se consigue con la propiedad **LIST-STYLE-POSITION**.

```
ul{  
  list-style-image: url('../img/corazon.png');  
  list-style-position: outside;  
}
```

Ilustración 359 - Imagen propia

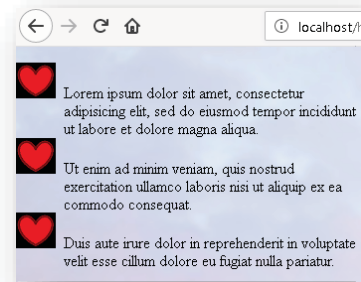


Ilustración 360 - Imagen propia

Si se puede observar, en el ejemplo anterior se encuentra el valor default que nos brinda **HTML5**, por lógica la única manera de poder brindar un cambio a la lista sería cambiando la posición del marcador, para ello se aplica el valor **INSIDE**.

```
ul{
  list-style-image: url('../img/corazon.png');
  list-style-position: inside;
}
```

Ilustración 361 - Imagen propia

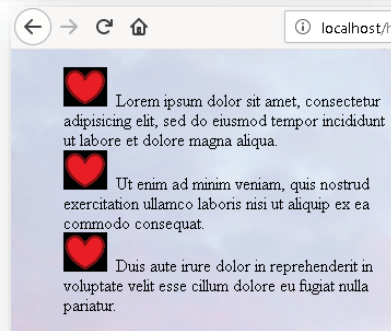


Ilustración 362 - Imagen propia

De esa manera se puede lograr un orden diferente dentro de las listas. Ahora bien, como en todo cambio de **CSS**, se puede generar una nueva vista si se añade color, por ello, embellecer una lista provoca frescura a un elemento que no convive tanto con un cambio como tal.

```
ul{
  background-color: #ab4086;
  list-style-image: url('../img/corazon.png');
  list-style-position: inside;
  padding: 10px;
}

li{
  background-color: #c28caf;
  color: white;
  margin: 5px;
}
```

Ilustración 363 - Imagen propia

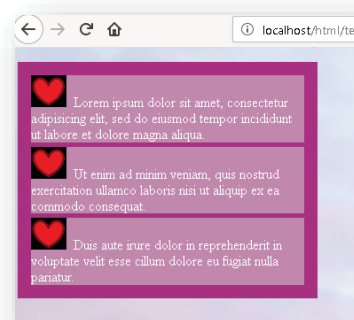


Ilustración 364 - Imagen propia

Por otro lado, tenemos a un elemento que permite registrar muchos datos, por ende, es un factor que suele volverse aburrido y tedioso de visualizar y entender, con esto se habla de las tablas, por ello, se permite mucho más cambio que el que brinda una lista, para conocer cómo cambiar el aspecto de una tabla, comenzamos plasmando una en **HTML5**, de manera muy sencilla.



Como primer indicio, se detallará la tabla con la propiedad **BORDER**, en este caso se emprenderá paso a paso para generar el cambio, por ello, primero se enmarca la tabla.



Pero bueno, también se necesita que se delinee las celdas y el encabezado, para eso aplicaremos la agrupación de selectores.



Muy padre ¿no crees?, pero creo que se genera un problema porque se nota sumamente saturado, para eso existe una propiedad que nos permite resolver ese problema **BORDER-COLLAPSE**, como su nombre lo dice, colapsara o unirá las líneas para que no se mire de esa manera.

```
table, th, td{
  border-style: solid;
  color: black;
}

table{
  border-collapse: collapse;
}
```

Ilustración 370 - Imagen propia

Nombre	Telefono	Ciudad
Abril Alba	56789432	Cuautitlan Izcalli
Omar Gutierrez	58345612	Tultitlan
Juan Carlos Loaza	56345132	Naucalpan
David Carrillo	79683421	Cuautitlan
Alan Cedeño	68752344	Ciudad de Mexico
Olimpia Garduño	34555234	Cuautitlan Izcalli
Julia Alvarez	58677743	Toluca
Victoria Garcia	54334423	Nuevo Leon
Susana Gonzales	78695040	Playa del Carmen
Pedro Cardona	43237645	Ciudad de Mexico

Ilustración 371 - Imagen propia

Seguramente te preguntaras cuales son los valores de la propiedad anterior, bueno para empezar esta propiedad consta de dos valores, **COLLAPSE** que indica que unirá los bordes y **SEPARATE**, si queremos un ejemplo visual solamente pongamos el ejemplo pasado al revés, supongamos que tenemos una tabla donde no se mira ningún borde doble, entonces lo que haría la propiedad sería doblar los bordes de la tabla del principio.

Pero algo que salta a la vista es que todo se ve súper amontonado, por lo cual utilizaremos las propiedades de medida que serían **HEIGHT** y **WIDTH**.

```
table, th, td{
  border-style: solid;
  color: black;
}

table{
  border-collapse: collapse;
  width: 50%;
}

th{
  height: 40px;
}

td{
  height: 20px;
}
```

Ilustración 372 - Imagen propia

Nombre	Telefono	Ciudad
Abril Alba	56789432	Cuautitlan Izcalli
Omar Gutierrez	58345612	Tultitlan
Juan Carlos Loaza	56345132	Naucalpan
David Carrillo	79683421	Cuautitlan
Alan Cedeño	68752344	Ciudad de Mexico
Olimpia Garduño	34555234	Cuautitlan Izcalli
Julia Alvarez	58677743	Toluca
Victoria Garcia	54334423	Nuevo Leon
Susana Gonzales	78695040	Playa del Carmen
Pedro Cardona	43237645	Ciudad de Mexico

Ilustración 373 - Imagen propia

Ahora si se nota como una tabla, algo muy importante dentro de una tabla es la alineación del texto contenido, para eso se puede utilizar la propiedad **TEXT-ALIGN** la cual alinea de manera horizontal el contenido en **IZQUIERDA (LEFT)**, **CENTRO (CENTER)** o **DERECHA (RIGHT)**, también existe la alineación **JUSTIFICADO (JUSTIFY)** pero en este caso no se utiliza, pero no está por demás mencionarlo para que no se olvide.


```

table, th, td{
border-style: solid;
color: black;
}

table{
border-collapse: collapse;
width: 50%;
}

th{
height: 40px;
}

td{
height: 20px;
}

.numero{
text-align: center;
}

```

Ilustración 374 - Imagen propia

Nombre	Telefono	Ciudad
Abril Alba	56789432	Cuautitlan Izcalli
Omar Gutierrez	58345612	Tultitlan
Juan Carlos Loiza	56345132	Naucalpan
David Carrillo	79683421	Cuautitlan
Alan Cedeño	68752344	Ciudad de Mexico
Olimpia Garduño	34555234	Cuautitlan Izcalli
Julia Alvarez	58677743	Toluca
Victoria Garcia	54334423	Nuevo Leon
Susana Gonzales	78695040	Playa del Carmen
Pedro Cardona	43237645	Ciudad de Mexico

Ilustración 375 - Imagen propia

Para estudiar detenidamente las alineaciones, se nota que el elemento **TH** que es la cabecera de la tabla, mantiene un valor por default que es alineación central, pero haciendo uso de las clases, se puede afectar a una sola fila donde pueda alinear el texto, esto con el objetivo de hacer un poco más estética la tabla, pero también se cuenta con otra alineación de texto que es **VERTICAL-ALIGN**, como su nombre lo dice alineará cualquier texto que se encuentre en **TH** y/o **TD** en la parte **SUPERIOR (TOP)**, **CENTRAL (CENTER)** o **INFERIOR (BOTTOM)** de la celda donde se encuentre.

```

table, th, td{
border-style: solid;
color: black;
}

table{
border-collapse: collapse;
width: 50%;
}

th{
height: 35px;
}

td{
height: 30px;
vertical-align: bottom;
}

.numero{
text-align: center;
}

```

Ilustración 376 - Imagen propia

Nombre	Telefono	Ciudad
Abril Alba	56789432	Cuautitlan Izcalli
Omar Gutierrez	58345612	Tultitlan
Juan Carlos Loiza	56345132	Naucalpan
David Carrillo	79683421	Cuautitlan
Alan Cedeño	68752344	Ciudad de Mexico
Olimpia Garduño	34555234	Cuautitlan Izcalli
Julia Alvarez	58677743	Toluca
Victoria Garcia	54334423	Nuevo Leon
Susana Gonzales	78695040	Playa del Carmen
Pedro Cardona	43237645	Ciudad de Mexico

Ilustración 377 - Imagen propia

Entendido lo anterior, ya se puede obtener una tabla, más definida, estilizada y organizada, pero eso no es lo único que se puede lograr en una tabla, aquí también se juega con las propiedades.

```

th, td{
  border-bottom: 1px solid purple;
  color: black;
}

table{
  border-collapse: collapse;
  width: 40%;
}

th{
  height: 35px;
}

td{
  height: 30px;
  vertical-align: bottom;
  padding: 5px 20px;
}

.numero{
  text-align: center;
}

```

Ilustración 378 - Imagen propia

Nombre	Telefono	Ciudad
Abril Alba	56789432	Cuautitlan Izcalli
Omar Gutierrez	58345612	Tultitlan
Juan Carlos Loaiza	56345132	Naucalpan
David Carrillo	79683421	Cuautitlan
Alan Cedeño	68752344	Ciudad de Mexico
Olimpia Garduño	34555234	Cuautitlan Izcalli
Julia Alvarez	58677743	Toluca
Victoria Garcia	54334423	Nuevo Leon
Susana Gonzales	78695040	Playa del Carmen
Pedro Cardona	43237645	Ciudad de Mexico

Ilustración 379 - Imagen propia

Lo único que aquí cambio, fue el tipo de borde, pero también podemos utilizar el estado interactivo que se conoció en los enlaces, como lo es **HOVER**.

```

tr:hover{
  background-color: #e1c3d7;
}

```

Ilustración 380 - Imagen propia

Nombre	Telefono	Ciudad
Abril Alba	56789432	Cuautitlan Izcalli
Omar Gutierrez	58345612	Tultitlan
Juan Carlos Loaiza	56345132	Naucalpan
David Carrillo	79683421	Cuautitlan
Alan Cedeño	68752344	Ciudad de Mexico
Olimpia Garduño	34555234	Cuautitlan Izcalli
Julia Alvarez	58677743	Toluca
Victoria Garcia	54334423	Nuevo Leon
Susana Gonzales	78695040	Playa del Carmen
Pedro Cardona	43237645	Ciudad de Mexico

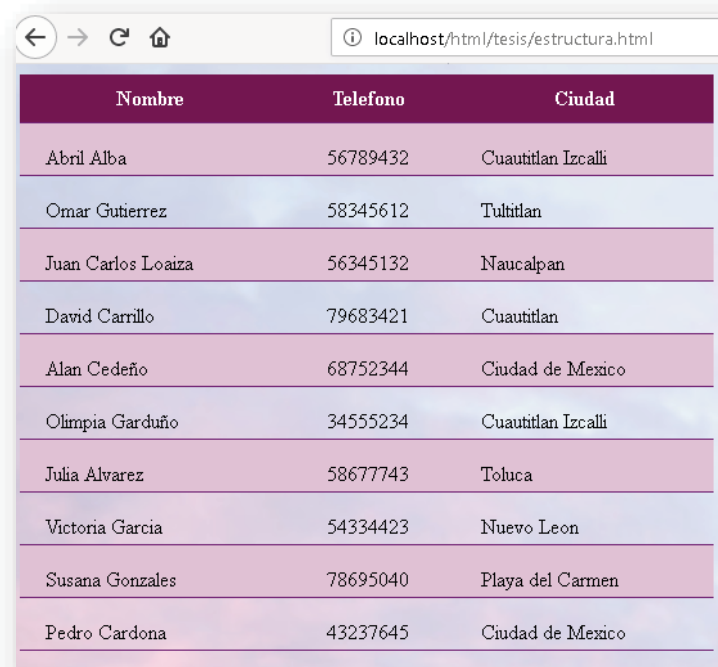
Ilustración 381 - Imagen propia

Y de esta manera, en cada una de las columnas se pintarán de color cuando se posicione el apuntador sobre ellas. Pero solo para aumentar más detalles, en las siguientes imágenes se muestra solo un pequeño cambio en el elemento **<TR>**, se utiliza una de las pseudoclasas que permiten hacer un efecto cebra, su comportamiento dentro de **CSS** es trabajar con los hijos de un elemento padre, por lo que hace una alteración de seleccionar imparmente los elementos.

```
tr:nth-child(even){
    background-color: #e1c3d7;
}

th{
    height: 35px;
    color: white;
    background-color: #7d1257;
}
```

Ilustración 382 - Imagen propia



Nombre	Telefono	Ciudad
Abril Alba	56789432	Cuautitlan Izcalli
Omar Gutierrez	58345612	Tultitlan
Juan Carlos Loaiza	56345132	Naucalpan
David Carrillo	79683421	Cuautitlan
Alan Cedeño	68752344	Ciudad de Mexico
Olimpia Garduño	34555234	Cuautitlan Izcalli
Julia Alvarez	58677743	Toluca
Victoria Garcia	54334423	Nuevo Leon
Susana Gonzales	78695040	Playa del Carmen
Pedro Cardona	43237645	Ciudad de Mexico

Ilustración 383 - Imagen propia

8.6. CSS RESPONSIVE

Más allá de darle color, forma, organización y estilo a una página web, **CSS** se distingue aún más por su labor en ofrecer un mejor producto al usuario final, pues de unos años en adelante la tecnología avanza a tal grado que arrasó con todo lo conocido, para el desarrollo y diseño web fue una oportunidad de crecimiento y madurez, pues ya no se encontraba estancado con la gran novedad que era tener imágenes gifs dentro de una página, vio más la perspectiva de otorgar mucha más funcionalidad al diseño, permitió adaptarse a lo nuevo y estar preparado para el futuro, para eso, se tuvo que involucrar con los dispositivos móviles, las grandes pantallas, las magníficas vistas de alta resolución, todo el proceso que ha llevado a cabo **CSS** de la mano con muchos otros elementos que involucran una plataforma culmina con un trabajo responsivo.

Y a que me refiero con **RESPONSIVO**, pues bien, desde luego que es la adaptación a las nuevas tecnologías, más allá de eso es el proceso por el cual pasan los elementos de **HTML5** por **CSS** para volverse adaptables a cualquier Smartphone, Tablet, PC. Con esto se concluye que la página se vuelve óptima, consumible por el público, moderna y sobretodo compatible con lo que se tenga a la mano tecnológicamente hablando.

Para poder comenzar a trabajar una página responsiva, se necesita saber de unos cuantos elementos indispensables para poder entender el manejo de las cajas en **CSS**, con esto lo primero es el diseño de la visualización, dentro de este tema se dispone de la propiedad “**DISPLAY**” en **CSS** es lo más importante para el control del diseño web, esta propiedad especifica si muestra y cómo va a mostrar un elemento, para entender esto, podremos entender el tema de las cajas, por ende, entendemos que cada elemento que contenga **HTML5** es un bloque o caja, pues estos bloques así como los visualizamos en el código, de igual manera se presentan en el navegador, pero con **CSS** podemos distinguir su comportamiento antes de moldearlos, por tanto, es importante saber qué elementos son **BLOCK** o **INLINE**.

Los elementos **BLOCK** son todos aquellos que en el código empiezan una nueva línea de código, cuando se representan en pantalla, se acomodan uno debajo del otro, aunado a que ocupan todo el ancho que dispongan de la línea, esos elementos son: **<DIV>**, **<H1>**, **<H2>**, **<H3>**, **<H4>**, **<H5>**, **<H6>**, **<P>**, **<HEADER>**, **<FOOTER>**, **<SECTION>** por dar un ejemplo.

Y los elementos en línea **INLINE**, son aquellos que se encuentran dentro de otro elemento y al presentarse en pantalla se colocan uno a lado del otro, por ende, solo ocupan el espacio que le es necesario para estar presente: ****, **<A>**, ****.

Ahora bien, comprendido el punto anterior, pasemos a la sustitución de valores predeterminados, los elementos que se mencionaron anteriormente con su respectiva descripción, son los valores default que les da **HTML5**, sin embargo, dentro de los estándares de **CSS**, ese valor predeterminado puede anularse de manera sutil, así que un elemento bloque puede convertirse con la propiedad **DISPLAY** un elemento en línea con el valor **INLINE** o viceversa, todo dependerá de la organización y diseño que se le otorgue a la página web.

Pero también se encuentra una combinación de ambos valores como es **INLINE-BLOCK**, lo que hace este valor es poner en línea los elementos, pero manteniendo el largo y ancho de un bloque, aparte de que no genera saltos de línea entre los elementos, este valor los mantiene juntos a los demás elementos para evitar tener algún espacio muerto, el uso más común de este valor son los enlaces de navegación, es decir, los menús.

Teniendo en cuenta ese punto, continuamos con la posición de la propiedad, los navegadores como tal le adjudican de forma automática una posición dentro de la página a todas las cajas que contenga el código **HTML5**, pero contando con la astucia de **CSS**, estos valores se pueden modificar utilizando la propiedad **POSITION** y uno de sus 5 valores.

- ✓ **NORMAL O ESTÁTICO (STATIC)**: Esta posición es el valor default que designa el navegador al código **HTML5**, por lo tanto, designa la posición al objeto en la esquina superior izquierda de la pantalla, cabe mencionar que todos los objetos se posicionaran debajo uno del otro sucesivamente hasta mostrarse todos dentro de la pantalla.

```
.estatica{
  border: 5px solid #e1c3e7;
  padding: 40px;
  font-size: 3em;
  position: static;
}
```

Ilustración 384 - Imagen propia

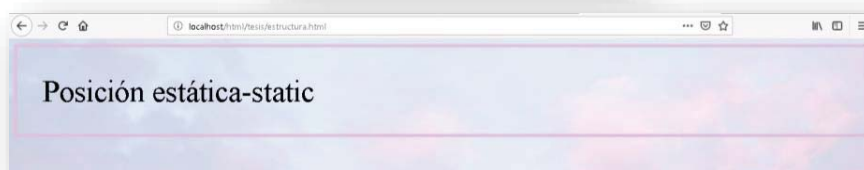


Ilustración 385 - Imagen propia

- ✓ **RELATIVO (RELATIVE)**: La posición relativa le permite al objeto cambiar su posición con respecto a los valores (**TOP**, **LEFT** y **BOTTOM**), no se menciona la alineación **RIGHT** porque la posición relativa está relacionada con la posición normal, es decir, no es necesario decirle al objeto que se recorra cierta distancia, cuando su origen proviene de lado izquierdo, por tanto, se puede ajustar a cualquier espacio de la pantalla.

```
.relativa{
  border: 5px solid #e1c3e7;
  padding: 40px;
  margin-top: 10px;
  font-size: 3em;
  position: relative;
  left: 150px;
}
```

Ilustración 386 - Imagen propia



Ilustración 387 - Imagen propia

- ✓ **ABSOLUTO (ABSOLUT)**: La posición absoluta vuelve independiente al objeto de la posición donde se encuentra, por lo tanto, se rige por los valores (**TOP, LEFT, RIGHT, BOTTOM**), es decir, el objeto queda por encima de los demás y es libre de colocarse donde sea sin afectar a los demás elementos.

```
.absoluta{
  border: 5px solid #69ebf1;
  padding: 40px;
  font-size: 3em;
  margin: 10px;
  position: absolute;
  left: 750px;
  top: -2px;
}
```

Ilustración 388 - Imagen propia

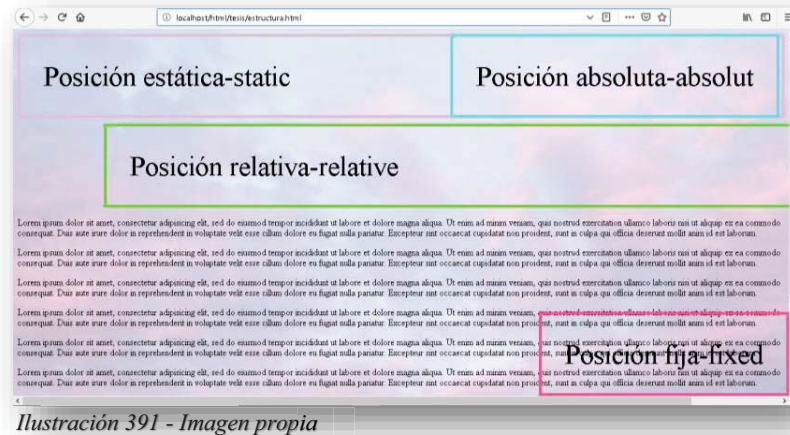


Ilustración 389 - Imagen propia

- ✓ **FIJO (FIXED)**: La posición fija mantiene el mismo objetivo que la posición absoluta, la única diferencia que hace en la posición fija es que como su nombre lo dice, el elemento se vuelve fijo en pantalla y no se mueve con el resto del contenido.

```
.fija{
  border: 5px solid #f959a1;
  padding: 40px;
  font-size: 3em;
  position: fixed;
  bottom: 0px;
  right: 0px;
}
```

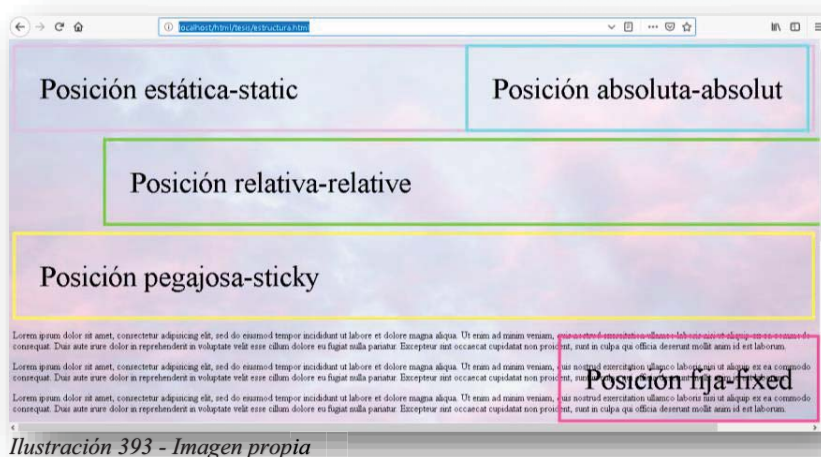
Ilustración 390 - Imagen propia



- ✓ **PEGAJOSO (STICKY)**: Esta posición se alterna entre la posición relativa y fija, pues su objetivo es dependiente del desplazamiento de la página web, es decir, primero se posiciona el elemento como si se utilizara la posición relativa, pero al momento de que la página se desplace hacia abajo, el elemento se mantendrá fijo en la pantalla y no desaparecerá hasta volver a su posición original.

```
.pegajosa{
border: 5px solid #fff95b;
margin-top: 10px;
padding: 40px;
font-size: 3em;
position: sticky;
top: 0px;
}
```

Ilustración 392 - Imagen propia



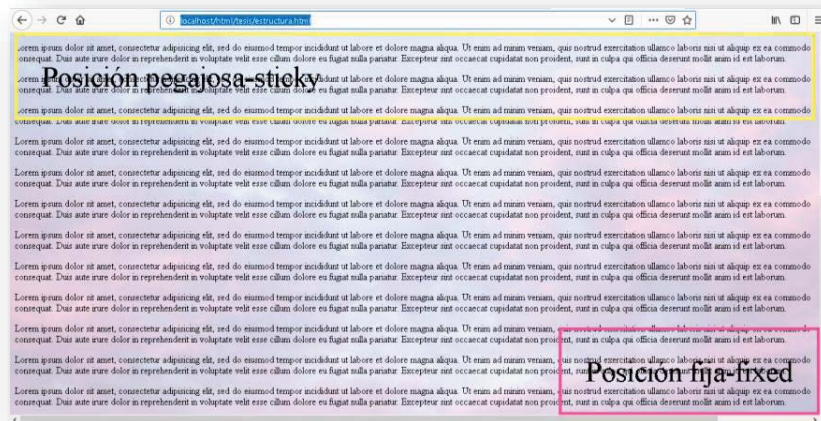


Ilustración 394 - Imagen propia

También se cuenta con otro tipo de posicionamientos que la propiedad **POSITION** no puede controlar, para ello son independientes por lo que se vuelven propiedades, una de ellas se llama superposición y en **CSS** se escribe **Z-INDEX** el objetivo de esta posición es situar un elemento en apilamiento, es decir, puede sobreponer un elemento en otros o ponerlo atrás de todos.

```

<div>
  Superposición
</div>
```

Ilustración 395 - Imagen propia

```
img {
  position: absolute;
  left: 0px;
  top: 0px;
  z-index: -1;
}
```

Ilustración 396 - Imagen propia



Ilustración 397 - Imagen propia, imagen de fondo tomada de https://www.freepik.es/fotos-premium/texturas-vintage-fondos-madera_2013955.htm

Como se puede observar en las imágenes anteriores, se tienen dos elementos completamente distintos, por lo cual debería encontrarse primero la imagen y debajo de ella el texto, pero

con la superposición se puede lograr unir o apilar ambos elementos y posicionarlos en un solo lugar.

Y por último se encuentra la posición más complicada de **CSS**, pero siendo honestos la más ocupada, que es la propiedad **FLOAT**, es difícil porque la comprensión de utilizarla suele complicarse un poquito, los valores que maneja son **RIGHT** y **LEFT**, cuando un elemento se vuelve flotante deja de pertenecer al flujo de organización que mantiene **HTML5**, con esto el elemento se vuelve independiente y por consecuencia afecta a los demás objetos, he de ahí el dolor de cabeza que provoca, pero para entenderlo, si un elemento se vuelve flotante los demás elementos que no lo sean se adecuaran y adaptaran al espacio que ocupe ese elemento, suena sencillo y es sencillo cuando se tienen pocos elementos, pero cuando la página web se encuentra completa, un elemento flotante afecta todo.

Para dar una idea de cómo funciona se colocarán dos elementos en **HTML5**, un párrafo con una imagen, de esta manera sin hacer ningún cambio en **CSS**, se visualizan de esta manera.

```

<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod
tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam,
quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo
consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse
cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non
proident, sunt in culpa qui officia deserunt mollit anim id est laborum.</p>
```

Ilustración 398 - Imagen propia

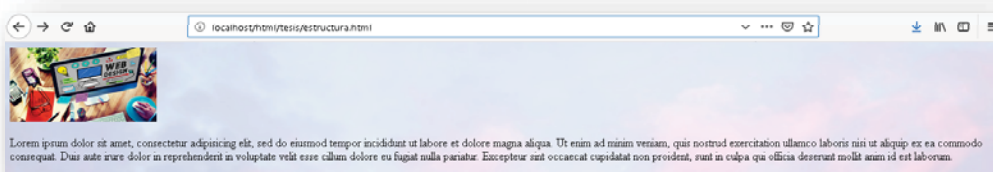


Ilustración 399 - Imagen propia , imagen ilustrativa tomada de <https://www.sanpablostereo.com/dise%C3%B1o-web/>

Ahora, se aplica en **CSS** la propiedad y esto sucede.

```
img{
float: right;
}
```

Ilustración 400 - Imagen propia

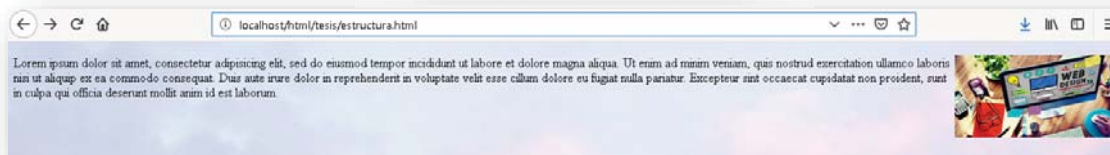


Ilustración 401 - Imagen propia, imagen ilustrativa tomada de <https://www.sanpablostereo.com/dise%C3%B1o-web/>

Un punto importante a mencionar sobre la propiedad **FLOAT** es que necesita que se cumplan unas cuantas condiciones para la correcta funcionalidad de la propiedad, en primer lugar es importante especificar el ancho del elemento, ya que los navegadores suelen asignar una anchura determinada para poder acomodar todos los elementos, algo que caracteriza a un elemento flotante es que en el momento en el que es declarado, se alinea todo lo que pueda al valor que fue designado y también se pegara tan arriba como le sea posible.

En ese momento es cuando se complican las cosas porque el elemento literalmente arruina toda la formación de los elementos, aunque ocasione problemas es una de las propiedades más flexibles dentro de **CSS**, por otro lado, se cuenta con otra propiedad que permite aligerar los problemas y pueden aplicarse juntas en un solo elemento **CLEAR**, su principal objetivo es reestablecer el flujo organizacional de los elementos, esta propiedad puede adoptar los mismos valores que **FLOAT** (**LEFT**, **RIGHT** y **NONE**).

Dentro del mismo tema de las cajas, se encuentra otro problema, cuando en un elemento se tiene mucha información, es decir, muchas palabras y cuando se edita en **CSS** ese elemento, suele pasar que se deforma el texto, por ende, se necesita de una propiedad que permita establecer un orden dentro de ese elemento.

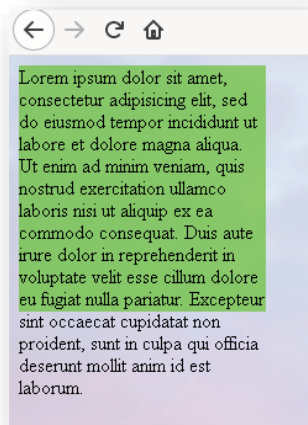


Ilustración 402 - Imagen propia

Para ello existe la propiedad **OVERFLOW** tiene por objetivo especificar al elemento si recorta el contenido o añade barras de desplazamiento. Cuenta con los siguientes valores:

- **VISIBLE**: Valor por defecto designado por **HTML5**, por lo tanto, como su nombre lo dice, permite que el texto se desborde del elemento y sea visible en pantalla.

```
div{
  background-color: rgba(24,255,0, .5);
  width: 200px;
  height: 200px;
  overflow: visible;
}
```

Ilustración 403 - Imagen propia

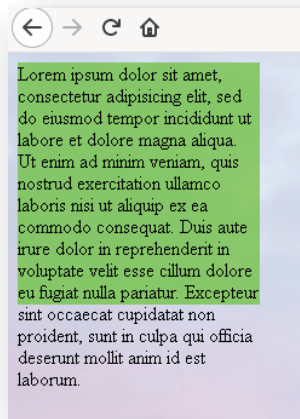


Ilustración 404 - Imagen propia

- **HIDDEN**: Recorta el texto y lo vuelve invisible en la pantalla, en mejores palabras para entenderlo, hace que desaparezca el texto y por lo tanto lo deja inconcluso.

```
div{  
  background-color: rgba(24,255,0, .5);  
  width: 200px;  
  height: 200px;  
  overflow: hidden;  
}
```

Ilustración 405 - Imagen propia

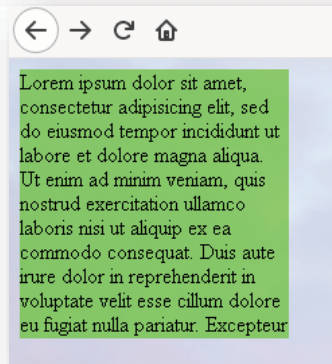


Ilustración 406 - Imagen propia

- **SCROLL**: Tiene la misma función que **HIDDEN**, pero la diferencia con este valor, radica en que establece barras de desplazamiento para poder visualizar el contenido restante, debe tenerse en cuenta que, así como **HIDDEN** no tiene compasión por recortar y desaparecer el texto restante, **SCROLL** establecerá barras de desplazamiento aun sin que el elemento las necesite.

```
div{
  background-color: rgba(24,255,0, .5);
  width: 200px;
  height: 200px;
  overflow: scroll;
}
```

Ilustración 407 - Imagen propia

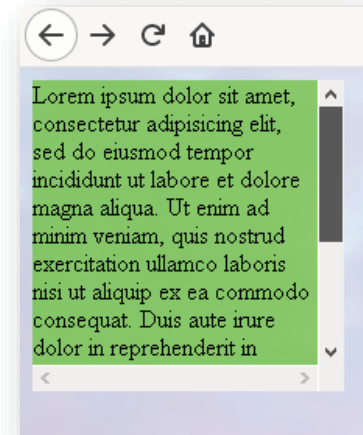


Ilustración 408 - Imagen propia

- **AUTO:** Concede la misma función que **SCROLL**, la única diferencia es que designa una barra de desplazamiento si lo ve necesario.

```
div{
  background-color: rgba(24,255,0, .5);
  width: 200px;
  height: 200px;
  overflow: auto;
}
```

Ilustración 409 - Imagen propia

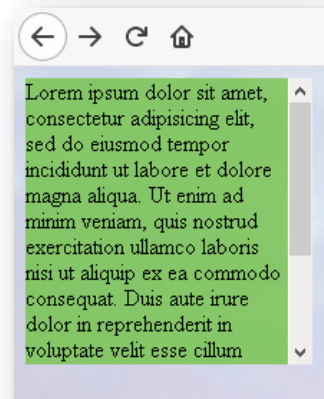


Ilustración 410 - Imagen propia

Solo como dato importante es que todos estos valores se pueden utilizar solo si el elemento tiene designada una altura, si esto no existe será en vano asignar la propiedad de desbordamiento al elemento.

A partir de este momento se puede comenzar a hacer magia dentro de **CSS**, pues ya se conocieron todas las propiedades y cada uno de sus valores, ya se conocieron las unidades de medida y color, también se tiene el conocimiento sobre cómo se tienen que visualizar los elementos como cajas, podemos comenzar a construir una página como se debe.

Uno de los elementos más conocidos en los diferentes sitios web o plataformas y que son indispensables, son las barras de navegación o menús, normalmente se comienza a trabajar con una base de **HTML5**, esta base comúnmente es una lista, si, así como lo lees, una lista puede manejarse como una barra de navegación y se puede transformar de esta manera:

```
<body>
  <ul>
    <li>Inicio</li>
    <li>HTML</li>
    <li>CSS</li>
    <li>JS</li>
    <li>Bootstrap</li>
    <li>Acerca</li>
    <li>Ayuda</li>
  </ul>
</body>
```

Ilustración 411 - Imagen propia



Ilustración 412 - Imagen propia

En primer lugar, se deben de quitar todos los elementos que caracterizan a una lista, como quitar las viñetas y reasignar la configuración por defecto de **HTML5**.

```
ul{
  list-style-type: none;
  margin: 0px;
  padding: 0px;
}
```

Ilustración 413 - Imagen propia

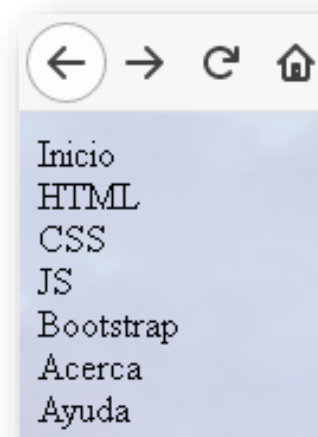


Ilustración 414 - Imagen propia

Después se asignará un fondo de color para hacer visible el área que se está trabajando y la anchura deseable a la barra de navegación.

```
ul{
  list-style-type: none;
  margin: 0px;
  padding: 0px;
  width: 250px;
  background-color: rgba(255,201,250,0.6);
}
```

Ilustración 415 - Imagen propia

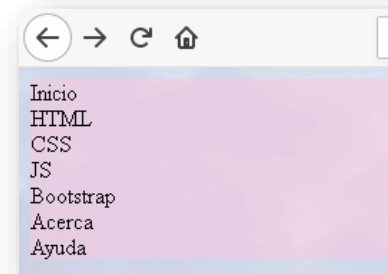


Ilustración 416 - Imagen propia

Ahora se le dará cuerpo a los botones de la barra volviendo al elemento **** bloque, dándole un relleno y una mejor apariencia al texto.

```
li{
  display: block;
  color: white;
  padding: 10px 12px;
  font-family: menu;
  text-shadow: 2px 2px 5px black;
  font-size: 2em;
}
```

Ilustración 417 - Imagen propia

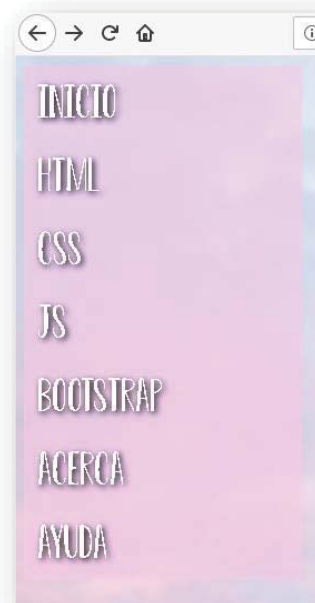


Ilustración 418 - Imagen propia

Para brindarle un diseño especial, se le activará uno de los estados interactivos para elementos y se le dará color diferente al primer botón para brindar visualmente armonía.


```
li:hover:not(.active){
    background-color: rgba(125,87,172,0.5);
}
.active{
    background-color: #b91bab;
}
```

Ilustración 419 - Imagen propia

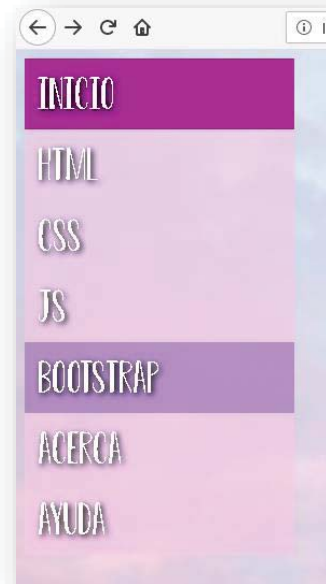


Ilustración 420 - Imagen propia

Sin duda alguna se completó una barra de navegación, ahora bien, para completar una visualización de la página, se le puede agregar una posición fija a la barra de navegación y uno que otro detalle.

```
ul{
    list-style-type: none;
    margin: 0px;
    padding: 0px;
    width: 250px;
    background-color: rgba(255,201,250,0.6);
    height: 100%;
    position: fixed;
}
```

Ilustración 421 - Imagen propia

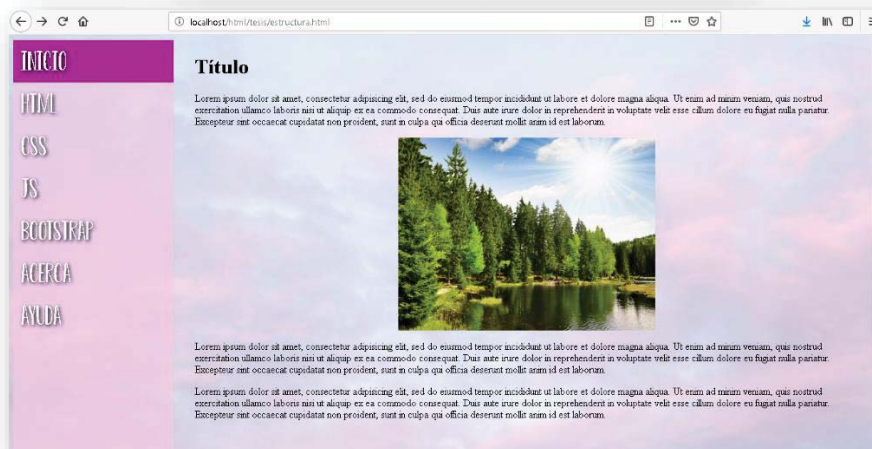


Ilustración 422 - Imagen propia, imagen ilustrativa tomada de <https://miviaje.com/7-parques-nacionales-de-alemania/>

Por supuesto también se puede colocar de forma horizontal y lo único que cambia es que se declaran flotantes los elementos **** y alineados a la derecha.

```
ul{
  list-style-type: none;
  margin: 0px;
  padding: 0px;
  background-color: rgba(255,201,250,0.6);
  overflow: hidden;
  position: fixed;
  width: 100%;
  top:0;
}

li{
  display: block;
  float: left;
  color: white;
  padding: 10px 12px;
  font-family: menu;
  text-shadow: 2px 2px 5px black;
  font-size: 2em;
}
```

Ilustración 423 - Imagen propia

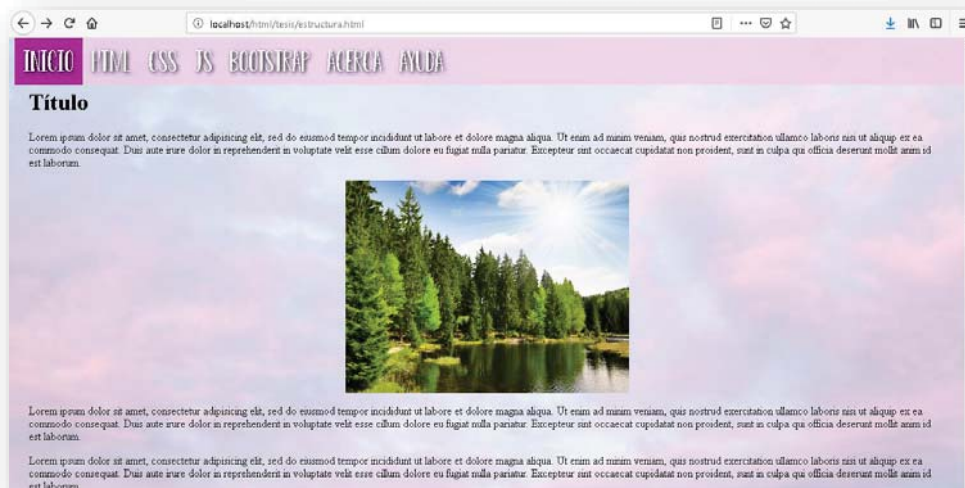


Ilustración 424 - Imagen propia, imagen ilustrativa tomada de <https://miviaje.com/7-parques-nacionales-de-alemania/>

Hasta este punto, ya se tiene el ejemplo de los dos tipos de barras que existen en páginas web, obviamente el modelo y estilo varía según el diseñador. Pero ¿Qué pasa cuando la barra se tiene que visualizar en diferentes dispositivos?, bueno para eso se tiene que convertir en responsivo el elemento y eso se logra de la siguiente manera:

En primera instancia se le asignaran a la barra tres diferentes posiciones como mínimo, si te preguntas porque tres, esto es para asignarle una posición a cada dispositivo con el que se cuenta hoy en día, como lo es una PC o laptop, Tablet y por último Smartphone o teléfono celular, cada una de ellas llevara diferentes valores para que se adapte a su respectivo tamaño de dispositivo, para poder hacerlo dentro de CSS se necesita de una herramienta responsiva **@MEDIA SCREEN**, la cual permite asignarle los diferentes valores al elemento con su

respectiva medida de cabecera, es decir, cuando se declara el elemento, en realidad estás declarando la herramienta y le asignas el tamaño que debe tener para mostrar cierto comportamiento de la página web, cuando esta se reduzca, cambiara de nuevo a otra posición, ahora bien, los valores para que se muevan van a colocarse dentro de la herramienta.

```
@media screen and (max-width: 900px) {
  ul {
    width: 100%;
    height: auto;
    position: relative;
  }

  ul li {
    float: left;
    padding: 15px;
  }

  div {margin-left: 0;}
}
```

Ilustración 425 - Imagen propia

Lo recomendable en estos casos es, asignar principalmente el estilo más grande, me refiero al que se le asigna al tamaño de un monitor de computadora, sin utilizar la herramienta y después colocar la posición del resto de los tamaños ahora si utilizando la herramienta. Por supuesto que existirán muchísimas medidas para los diferentes dispositivos que se tienen actualmente, pero asignando una medida estándar para cada uno puede funcionar perfectamente con tres posiciones, de esa manera se puede lograr lo siguiente:

```
ul{
  list-style-type: none;
  margin: 0px;
  padding: 0px;
  background-color: rgba(255,201,250,0.6);
  overflow: auto;
  position: fixed;
  width: 25%;
  height: 100%
}
li{
  display: block;
  color: white;
  padding: 8px 16px;
  font-family: menu;
  text-shadow: 2px 2px 5px black;
  font-size: 2em;
}
li:hover:not(.active){
  background-color: rgba(125,87,172,0.5);
}
.active{
  background-color: #b91bab;
}
```

Ilustración 426 - Imagen propia



Ilustración 427 - Imagen propia, imagen ilustrativa tomada de <https://miviaje.com/7-parques-nacionales-de-alemania/>

```

@media screen and (max-width: 900px) {
  ul {
    width: 100%;
    height: auto;
    position: relative;
  }

  ul li {
    float: left;
    padding: 15px;
  }

  div {
    margin-left: 10%;
  }
}

```

Ilustración 428 - Imagen propia



Ilustración 429 - Imagen propia

```

@media screen and (max-width: 700px) {
  ul li {
    text-align: center;
    float: none;
  }
}

```

Ilustración 430 - Imagen propia



Ilustración 431 - Imagen propia, imagen ilustrativa tomada de <https://miviaje.com/7-parques-nacionales-de-alemania/>

Entendido el tema anterior, ahora sí se puede decir que comenzamos a innovar las páginas web a dispositivos móviles, pero eso no lo es todo, en una barra de navegación siempre existirán menús desplegables, para ello se necesita de una nueva lista, exacto, así como lo lees, todo esto se desarrolla a partir de una lista, apuesto que se está volviendo muy fácil este tema.

Para tener una mayor facilidad de comportamiento y manejo de los elementos, comenzaremos con un **<DIV>**, esta caja contendrá todos los elementos de la lista. Y lo identificaremos con el nombre “vertical” esto se explicará más adelante.

```
<div id="vertical"></div>
```

Ilustración 432 - Imagen propia

Dentro de este **<DIV>**, se insertará la lista base, esta se identificará con la clase **MENU**.

```
<div id="vertical">  
  <ul class="menu">  
    <li><a href="#">Inicio</a></li>  
    <li><a href="#">HTML</a></li>  
    <li><a href="#">CSS</a></li>  
    <li><a href="#">JS</a></li>  
    <li><a href="#">Acerca</a></li>  
    <li><a href="#">Ayuda</a></li>  
  </ul>  
</div>
```

Ilustración 433 - Imagen propia

Hasta este momento ya se puede visualizar algo en pantalla.

- [Inicio](#)
- [HTML](#)
- [CSS](#)
- [JS](#)
- [Bootstrap](#)
- [Acerca](#)
- [Ayuda](#)

Ilustración 434 - Imagen propia

Después se incluirán los elementos correspondientes a cada uno de la lista.

```

<div id="vertical">
  <ul class="menu">
    <li><a href="#">Inicio</a></li>
    <li><a href="#">HTML</a>
      <ul>
        <li><a href="#">Historia</a></li>
        <li><a href="#">Concepto</a></li>
        <li><a href="#">Función</a></li>
        <li><a href="#">Estructura</a></li>
        <li><a href="#">Etiquetas</a></li>
        <li><a href="#">Atributos</a></li>
        <li><a href="#">Diseño Responsive</a></li>
      </ul>
    </li>
    <li><a href="#">CSS</a>
      <ul>
        <li><a href="#">Historia</a></li>
        <li><a href="#">Concepto</a></li>
        <li><a href="#">Función</a></li>
        <li><a href="#">Estructura</a></li>
        <li><a href="#">Medidas</a></li>
        <li><a href="#">Propiedades</a></li>
        <li><a href="#">Diseño Responsive</a></li>
      </ul>
    </li>
    <li><a href="#">JS</a>
      <ul>
        <li><a href="#">Historia</a></li>
        <li><a href="#">Concepto</a></li>
        <li><a href="#">Función</a></li>
        <li><a href="#">Estructura</a></li>
      </ul>
    </li>
    <li><a href="#">Bootstrap</a>
      <ul>
        <li><a href="#">Historia</a></li>
        <li><a href="#">Concepto</a></li>
        <li><a href="#">Función</a></li>
        <li><a href="#">Estructura</a></li>
        <li><a href="#">Modelos</a></li>
      </ul>
    </li>
    <li><a href="#">Acerca</a></li>
    <li><a href="#">Ayuda</a></li>
  </ul>
</div>

```

Ilustración 435 - Imagen propia

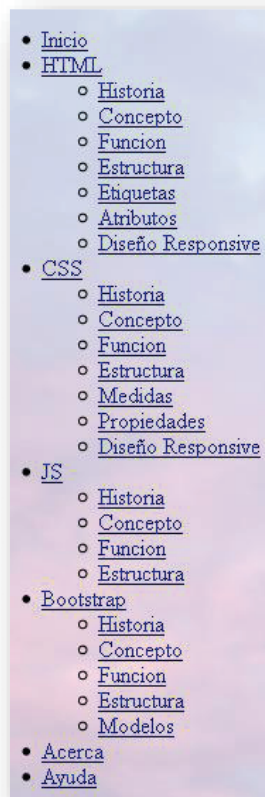


Ilustración 436 - Imagen propia

Por supuesto, que se pueden poner todos los niveles que se quieran para realizar un menú más ambicioso, todo depende del gusto, del contenido o de la organización de la información.

```

<li><a href="#">CSS</a>
  <ul>
    <li><a href="#">Historia</a></li>
    <li><a href="#">Concepto</a></li>
    <li><a href="#">Función</a></li>
    <li><a href="#">Estructura</a></li>
    <li><a href="#">Medidas</a>
      <ul>
        <li><a href="#">Unidades</a>
          <ul>
            <li><a href="#">Absolutas</a></li>
            <li><a href="#">Relativas</a></li>
          </ul>
        </li>
        <li><a href="#">Colores</a>
          <ul>
            <li><a href="#">RGB</a></li>
            <li><a href="#">HX</a></li>
          </ul>
        </li>
      </ul>
    </li>
    <li><a href="#">Propiedades</a></li>
    <li><a href="#">Diseño Responsive</a></li>
  </ul>
</li>

```

Ilustración 437 - Imagen propia



Ilustración 438 - Imagen propia

Ahora viene el momento de mejorar todo en **CSS** y lo primordial es reubicar al elemento de origen, es decir, quitarle márgenes y rellenos a la lista, para ello, se aplica un elemento “*” que significa que todo lo que esté dentro de la página web no tendrá si márgenes ni rellenos. Al igual que eliminar cualquier elemento de decoración de texto.

```
* {
  margin: 0px;
  padding: 0px;
  text-decoration: none;
  list-style-type: none;
}
```

Ilustración 439 - Imagen propia



Ilustración 440 - Imagen propia

Como se puede observar, todos los elementos de la lista se pegaron de lado de origen, es decir, de lado izquierdo de la pantalla. A continuación, se utilizará al identificador **VERTICAL**, es decir, a **<DIV>** se le asignara un margen de 20px y detalles de fuente.

```
#vertical {
  margin: 20px;
  width: 250px;
  font-family: menu;
  text-align: center;
  font-size: 2em;
  text-shadow: 2px 2px 5px black;
}
```

Ilustración 441 - Imagen propia



Ilustración 442 - Imagen propia

Aún no se puede visualizar un orden de los elementos, se necesita un poco más de dedicación, para ello, lo que sigue es recortar la lista, es decir, decirles a todos los elementos que no correspondan a la lista principal que desaparezcan.


```
.menu li ul{
  display: none;
}
```

Ilustración 443 - Imagen propia



Ilustración 444 - Imagen propia

Ahora daremos un mejor aspecto al menú ¡pero ojo! aún no es el resultado final. Se le aplicará un display **BLOCK**, eliminaremos la decoración de los enlaces, se le dará un relleno, color de fuente y fondo al igual que se le otorgará la posición relativa para movimientos futuros.

```
.menu li a {
  display: block;
  position: relative;
  padding: 8px;
  color: #fff;
  background-color: rgba(255,201,250,0.6);
}
```

Ilustración 445 - Imagen propia

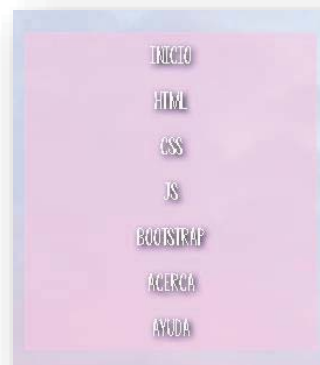


Ilustración 446 - Imagen propia

El aspecto sigue siendo el de una lista en una caja, pero poco a poco eso cambiará, la posición de esta lista se volverá relativa para permitir a los elementos ocultos moverse con más facilidad, ahora que también se hace un llamado a la clase que le dará vida al menú y es **HOVER**.

```
.menu li a: hover{
  background-color: rgba(125,87,172,0.5);
  color: white;
  position: relative;
}
```

Ilustración 447 - Imagen propia

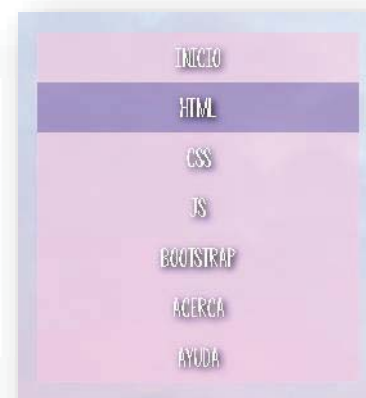


Ilustración 448 - Imagen propia

En este momento se puede hacer aparecer el segundo nivel de la lista, volviendo su visibilidad en **BLOCK** con una posición absoluta y un margen izquierdo, este último se utiliza para que la posición absoluta no se empalme con el menú principal.

```
.menu li:hover > ul{
  display: block;
  position: absolute;
  left: 280px;
}
```

Ilustración 449 - Imagen propia

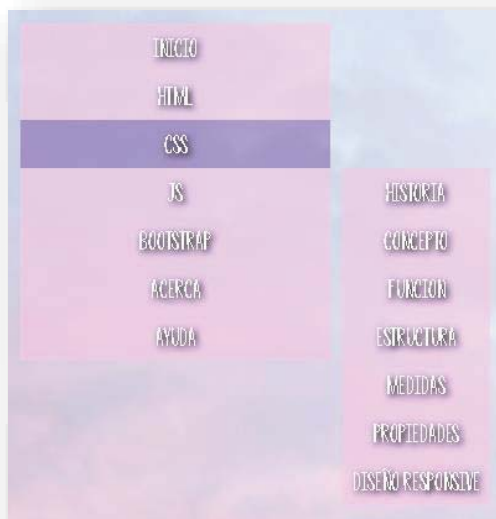


Ilustración 450 - Imagen propia

Como se puede observar, se tiene un error en la visualización del segundo nivel, pero eso tiene arreglo y se soluciona con la siguiente declaración, asignando márgenes de derecha y altura para adecuar la posición donde se requiera.

```
.menu li ul li{
  position: relative;
  background-color: #6CC;
  color: #FFF;
  right: 10px;
  top: -62px;
}
```

Ilustración 451 - Imagen propia



Ilustración 452 - Imagen propia

También se asigna un ancho determinado y color.

```
.menu li ul li a {  
  width: 254px;  
  background-color: #6CC;  
  color: #FFF;  
}
```

Ilustración 453 - Imagen propia



Ilustración 454 - Imagen propia

Con esos parámetros, se puede seguir aumentando listas dentro de las listas y todas formaran parte de las medidas y espacios que se les asignaron a las principales.



Ilustración 455 - Imagen propia

Y solo por último se le asignaran detalles de color.

```

.menu li ul li a:hover{
    background-color: #399;
}

.menu ul li ul li a:hover {
    background-color: #399;
    color: white;
}

.active{
    background-color: #b91bab;
}

```

Ilustración 456 - Imagen propia

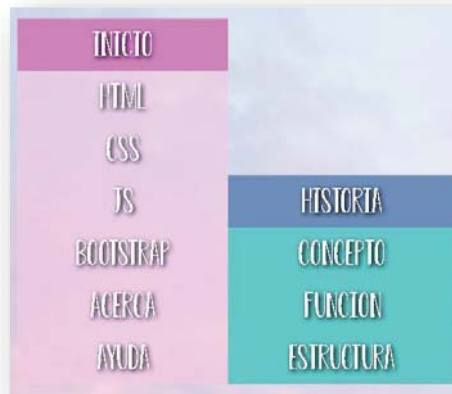


Ilustración 457 - Imagen propia

Por supuesto que también se puede mantener un menú horizontal y para ello no existen grandes cambios, pues lo único que sucede es que los elementos se vuelven flotantes desde un principio y son orientados a la izquierda en bloques.

```

*{
    padding: 0px;
    margin: 0px;
    list-style-type: none;
    text-decoration: none;
}

#vertical{
    margin: auto;
    width: 900px;
    font-family: menu;
    text-align: center;
    font-size: 1.5em;
    text-shadow: 2px 2px 5px black;
}

.menu li a{
    background-color: rgba(255,201,250,0.6);
    color: white;
    padding: 8px 15px;
    display: block;
}

.menu > li{
    float: left;
}

.menu li a:hover{
    background-color: rgba(125,87,172,0.5);
    color: white;
}

.menu li ul{
    display: none;
    position: absolute;
    min-width: 170px;
}

.menu li:hover > ul{
    display: block;
    background-color: #6CC;
}

.menu li ul li{
    position: relative;
    background-color: #399;
    color: white;
}

.menu li ul li ul{
    right: -170px;
    top: 0px;
}

```

Ilustración 458 - Imagen propia



Ilustración 459 - Imagen propia

Por otro lado, se tienen las imágenes, las famosas ilustraciones que dan vida a una página web, claro que con su sola presencia da una mejor vista a la pantalla, pero hoy en día ya no es suficiente una imagen, el desarrollo de diseño que ha tenido con el tiempo ha dejado obsoleta la simple ilustración, ahora se utiliza editar la imagen en color, dimensión, animación, galerías interactivas y por supuesto moldear la imagen al gusto. Para ello se puede comenzar con una imagen, insertándola dentro del sitio de **HTML5**.

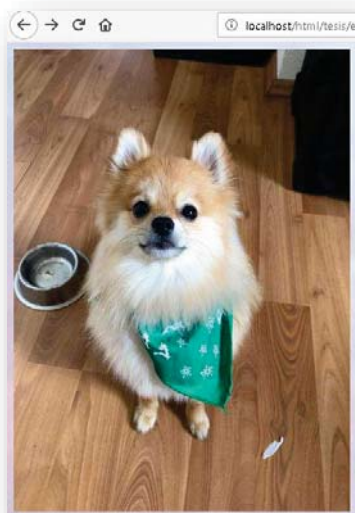


Ilustración 460 - Imagen propia

A una imagen se le pueden dar muchos cambios estéticos dentro de **CSS**, muy en particular se puede utilizar un borde redondeado, con la cual se puede adaptar para ilustraciones antiguas, modernizar el contorno de la fotografía, etc.

```
img{  
  border-radius: 50%;  
}
```

Ilustración 461 - Imagen propia



Ilustración 462 - Imagen propia

Una imagen también puede volverse un enlace, estos se manejan en miniatura como tipo carrito de imágenes o galería de imágenes, lo cual permite seleccionar la fotografía y expandirla para poder apreciarle aún más.

```
<a target="_blank" href="img/rocket/FT1.jpg">  
    
</a>
```

Ilustración 463 - Imagen propia

```
img{  
  border:3px solid #383;  
  border-radius: 20px;  
  padding: 6px;  
  width: 150px;  
}  
  
img:hover{  
  box-shadow: 6px 6px 6px 6px rgba(0, 147, 0, 0.5);  
}
```

Ilustración 464 - Imagen propia



Ilustración 465 - Imagen propia

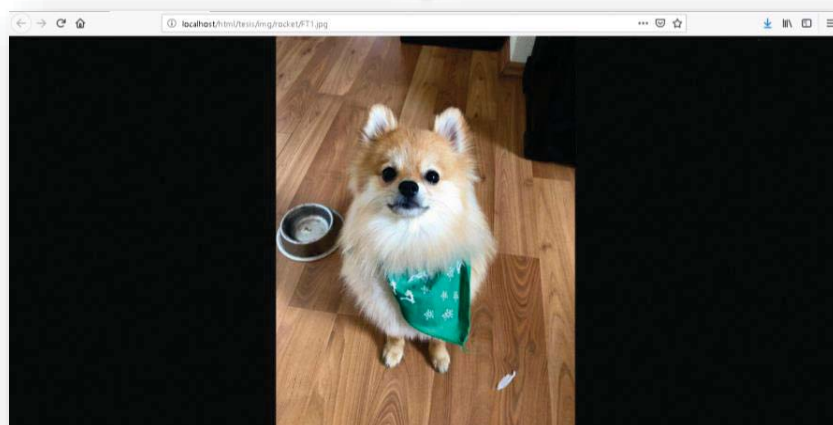


Ilustración 466 - Imagen propia

Algo muy importante dentro del tema responsivo, es que las imágenes también se encuentran compatibles con estos cambios, y si bien es que es un paso imprescindible para las nuevas tecnologías, CSS tampoco hace magia, pues el modo responsivo se aplica para reducir a escala la imagen, de lo contrario podemos encontrarnos con el tamaño original de la imagen y para de contar.

```
img{
  max-width: 100%;
  height: auto;
}
```

Ilustración 467 - Imagen propia



Ilustración 468 - Imagen propia



Ilustración 469 - Imagen propia

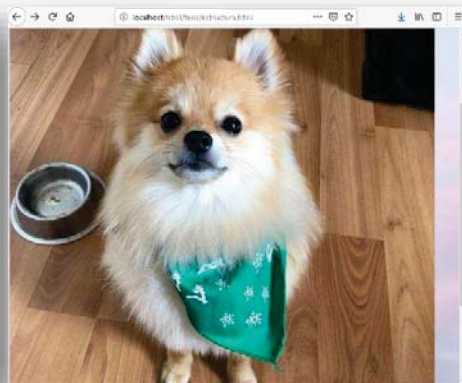


Ilustración 470 - Imagen propia

Como se puede observar, a medida que se hace grande la imagen o se abre más la ventana web, aparece la barra de desplazamiento y la imagen comienza a expandirse hasta su tamaño original, sin perder calidad ni distorsión.

Uno de los problemas comunes en una imagen es poder centrarla dentro de la pantalla, pero a veces suele causar dolor de cabeza con el hecho de colocar todos los márgenes y ver si se le atina al centro, ahora puede ser más simple volviendo la imagen un **BLOCK**, esta cualidad le permitirá moverse sin ningún problema.

```
img{
  display: block;
  margin-left: auto;
  margin-right: auto;
}
```

Ilustración 471 - Imagen propia

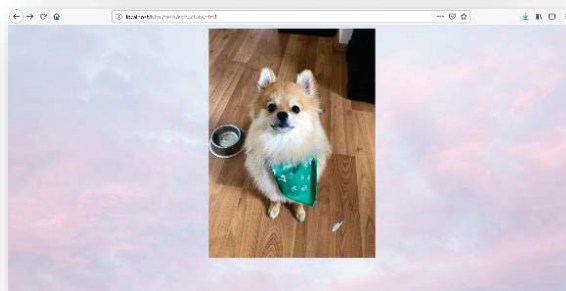


Ilustración 472 - Imagen propia

Algunas de las opciones para las imágenes responsivas, son las leyendas o pie de imagen que contienen.

```
<div class="polaroid">
  
  <div class="container">
    <p>Rocket atento!!!</p>
  </div>
</div>
```

Ilustración 473 - Imagen propia

```
div.polaroid {
  width: 40%;
  background-color: white;
  box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.2), 0 6px 20px 0 rgba(0, 0, 0, 0.19);
  margin-bottom: 25px;
}

div.container {
  text-align: center;
  padding: 3px 5px;
}
```

Ilustración 474 - Imagen propia



Ilustración 475 - Imagen propia

En una imagen también se puede utilizar la propiedad **OPACITY**, es decir, que una imagen se puede volver transparente, esto funciona de igual forma con la opacidad de los colores, pues sus valores varían de 1.0 que es el valor de mayor exposición de la imagen y el 0 que es el valor de mayor transparencia de imagen.

```
img {
  opacity: 1.0;
}
```

Ilustración 476 - Imagen propia

```
img {
  opacity: 0.5;
}
```

Ilustración 478 - Imagen propia

```
img {
  opacity: 0.0;
}
```

Ilustración 480 - Imagen propia



Ilustración 477 - Imagen propia



Ilustración 479 - Imagen propia

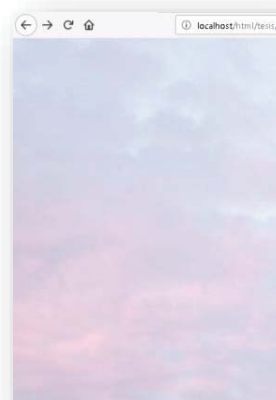


Ilustración 481 - Imagen propia

Al igual que se puede colocar texto sobre una imagen, esto se logra partiendo de posiciones relativas y absolutas, por supuesto que todo depende del gusto y elección del usuario final o diseñador web, puesto que las medidas y posiciones tienen mayor probabilidad de variar.

```
<div class="contenedor">
  
  <div class="centro">¡¡¡CUIDADO!!!</div>
</div>
```

Ilustración 482 - Imagen propia

```
.centro {
  position: absolute;
  top: 50%;
  left: 50%;
  transform: translate(-50%, -50%);
  font-size: 2em;
  font-family: menu;
}

img {
  opacity: 0.3;
  display: block;
  margin-left: auto;
  margin-right: auto;
}
```

Ilustración 483 - Imagen propia



Ilustración 484 - Imagen propia

Y como buena edición de imágenes, no pueden faltar los filtros, no es Photoshop ni nada por el estilo, se trata de los contrastes, luces y sombras, matices y tintes que se le dan a la imagen, con los que se puede jugar para convertir a la fotografía o imagen en algo más especial.

```











```

Ilustración 485 - Imagen propia

```

img {
  float: left;
  max-width: 500px;
}

.blur {-webkit-filter: blur(4px);filter: blur(4px);}
.brightness {-webkit-filter: brightness(250%);filter: brightness(250%);}
.contrast {-webkit-filter: contrast(180%);filter: contrast(180%);}
.grayscale {-webkit-filter: grayscale(100%);filter: grayscale(100%);}
.huerotate {-webkit-filter: hue-rotate(180deg);filter: hue-rotate(180deg);}
.invert {-webkit-filter: invert(100%);filter: invert(100%);}
.opacity {-webkit-filter: opacity(50%);filter: opacity(50%);}
.saturate {-webkit-filter: saturate(7);filter: saturate(7);}
.sepia {-webkit-filter: sepia(100%);filter: sepia(100%);}
.shadow {-webkit-filter: drop-shadow(8px 8px 10px green);filter: drop-shadow(8px 8px 10px green);}

```

Ilustración 486 - Imagen propia

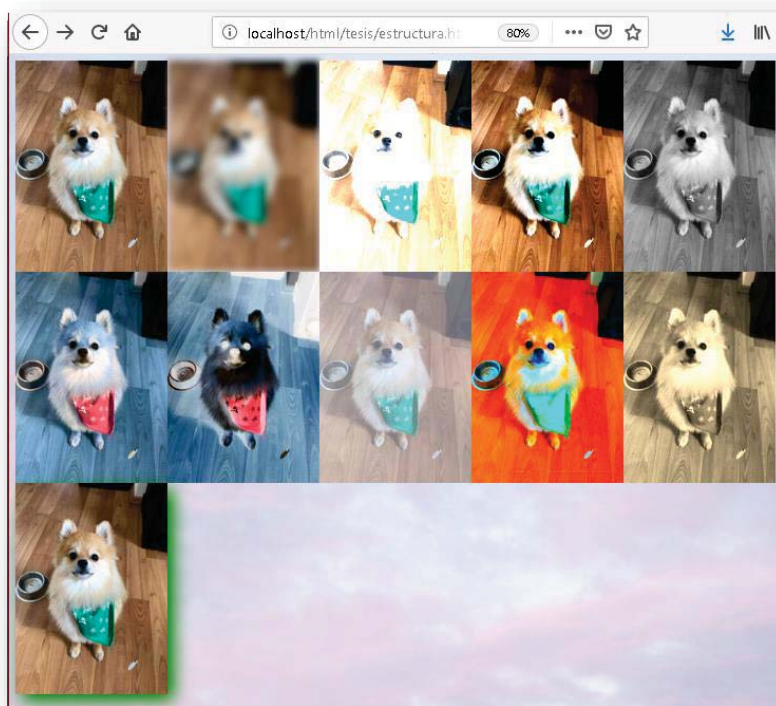


Ilustración 487 - Imagen propia

Para una imagen, dentro de CSS tiene un mundo de posibilidades para lucir increíble y por supuesto, hacer que el sitio web luzca muy moderno y estilizado, para ello, las imágenes pueden volcarse a tener personalizaciones dinámicas como se vio anteriormente en los colores tipo gif's.

Para que una imagen pueda ser editada dinámicamente, siempre va a permanecer dentro de un **<DIV>**, a la cual se le va a definir por un nombre para poder ser editable dentro de CSS, ahora bien, para aplicar los efectos también se insertará otro **<DIV>** dentro del “**<DIV>** padre”, este servirá para declarar el efecto y un texto, porque de esta manera es como se están presentando las nuevas imágenes. Y los efectos que se tienen son los siguientes.

✓ DESVANECIMIENTO

```
<div class="contenedor">
  
  <div class="overlay">
    <div class="texto">GUAU!!! GUAU!!!</div>
  </div>
</div>
```

Ilustración 488 - Imagen propia

Ahora bien, dentro de **CSS CONTENEDOR** se le asignaran los valores de posición relativa y con una anchura del 50%, esto posicionará a la imagen en el 50% de la pantalla por la anchura y se podrá mover si así se desea por la posición.

```
.contenedor {
  position: relative;
  width: 50%;
}
```

Ilustración 489 - Imagen propia

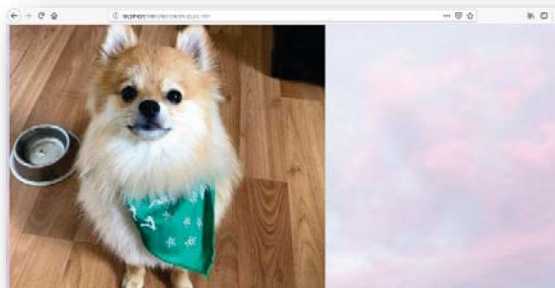


Ilustración 490 - Imagen propia

Después la imagen solo se declarará su visualización en bloque y sus márgenes se mantendrán fieles a la imagen, por supuesto, esto puede variar de acuerdo al gusto del diseñador.

```
.image {
  display: block;
  width: 100%;
  height: auto;
}
```

Ilustración 491 - Imagen propia

En este momento se toma el segundo **<DIV>** y se le aplican los efectos para la imagen. Este contendrá una posición absoluta, no tendrá margen de ningún tipo (superior, inferior, izquierdo, derecho), pero es realmente necesario declararlos en cero para que el navegador no les asigne por default algún margen, se le asignara el 100% de alto y ancho para que cubra por completo la imagen, se le declarara un valor de partida de opacidad y una transición lenta de la cantidad deseada de segundos a la transición, por último, se le asignara un color.

```
.overlay {
  position: absolute;
  top: 0;
  bottom: 0;
  left: 0;
  right: 0;
  height: 100%;
  width: 100%;
  opacity: 0;
  transition: 6s ease;
  background-image: radial-gradient(farthest-side at 10% 55%, white, violet, grey);
}
```

Ilustración 492- Imagen propia

Como se puede seguir el código, aun no existe ningún cambio, bueno pues aún falta **HOVER** como ya se ha entendido a lo largo del tema, esta pseudoclase nos permite realizar dinámicas transiciones dentro de los elementos.

```
.contenedor:hover .overlay {
  opacity: 1;
}
```

Ilustración 493 - Imagen propia

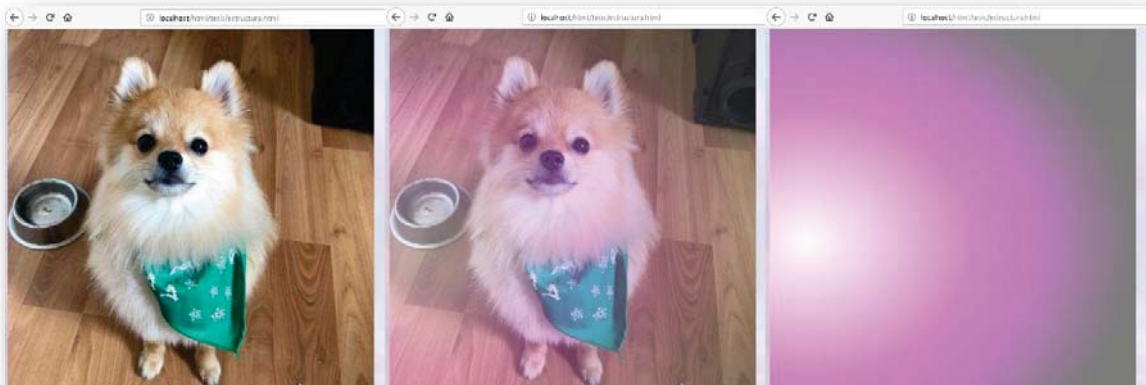


Ilustración 494 - Imagen propia

Así como se presenta en la imagen, durante 6 segundos se realizará una transición de desvanecimiento con el color de elección, y como se puede ver en el **<DIV>** de la interacción se le agregó un texto que aparecerá dentro de la transición, esto puede utilizarse para realizar una colaboración informativa alusiva a la imagen, ya depende de la manera que se empleen los elementos.

```

.texto {
  color: white;
  font-size: 20px;
  position: absolute;
  top: 50%;
  left: 50%;
  transform: translate(-50%, -50%);
  -ms-transform: translate(-50%, -50%);
}

```

Ilustración 495 - Imagen propia

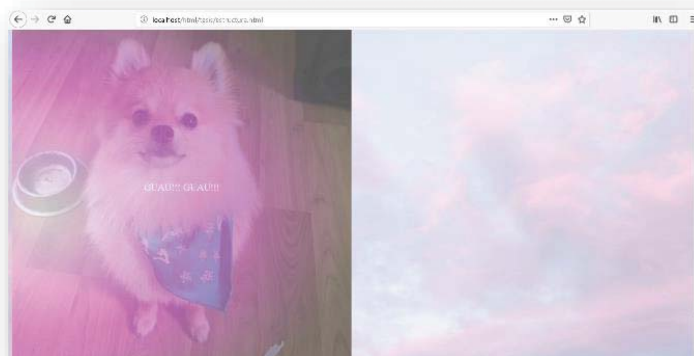


Ilustración 496 - Imagen propia

Los elementos raros que se notan en la declaración del texto **TRANSFORM: TRASLATE (X,Y)** se utiliza principalmente para reubicar al elemento en su punto de origen sobre otros elementos y poderlo ubicar donde se desea, su espacio será su 100% dentro del elemento en el que se encuentre y la siguiente línea es para que otros navegadores web entiendan la misma regla.

✓ DESLIZAMIENTO SUPERIOR

Para permitirse un deslizamiento como diapositiva de power point, se va a seguir trabajando con el código CSS anterior, la única diferencia va a radicar en este y los siguientes tres movimientos de la siguiente manera.

```

.overlay {
  position: absolute;
  bottom: 100%;
  left: 0;
  right: 0;
  height: 0;
  width: 100%;
  overflow: hidden;
  transition: 6s ease;
  background-image: linear-gradient(to bottom right, #009096, #bc0083, #998600);
}

```

Ilustración 497 - Imagen propia


```

.contenedor:hover .overlay {
  bottom: 0;
  height: 100%;
}

```

Ilustración 498 – Imagen propia

Para que exista un desplazamiento superior se necesita quitar **MARGIN TOP**, a **BOTTOM** se le asignara el 100% de esta manera cubrirá al 100% la imagen, por consiguiente, se le dará el ancho del 100% mientras que la altura se le retirará para que permita una transición sin tropiezos, también se le aplicará **OVERFLOW HIDDEN** para que no se pueda ver en primera instancia el color, aparte en **HOVER** se le agregará que **BOTTOM** estará en cero, pues así se generará la transición.

```

.texto {
  color: white;
  white-space: nowrap;
  font-size: 20px;
  position: absolute;
  overflow: hidden;
  top: 50%;
  left: 50%;
  transform: translate(-50%, -50%);
  -ms-transform: translate(-50%, -50%);
}

```

Ilustración 499 - Imagen propia

También se le agregará al texto un **OVERFLOW** invisible para que permanezca de esa manera junto con el color dinámico, al igual que la propiedad **WHITE-SPACE**, esta se dedica a limitar los espacios en blanco de un elemento. Y así se muestra el movimiento.

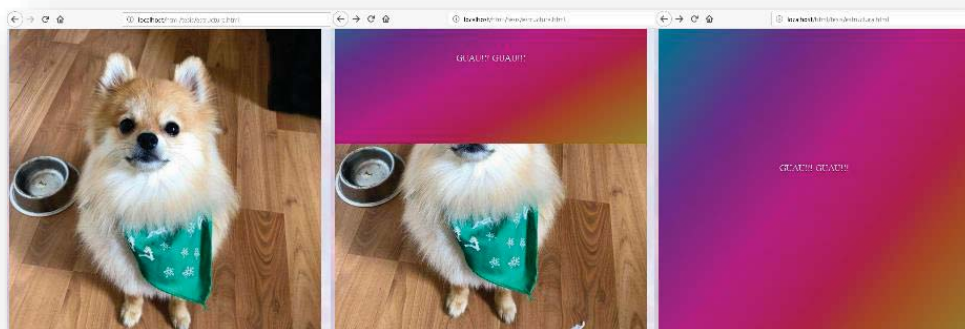


Ilustración 500 - Imagen propia

✓ DESLIZAMIENTO INFERIOR

En esta transición se declara en el elemento del movimiento en cero **BOTTOM** y se le retirará de igual manera **BOTTOM** de **HOVER**.

```
.overlay {  
  position: absolute;  
  bottom: 0;  
  left: 0;  
  right: 0;  
  height: 0;  
  width: 100%;  
  overflow: hidden;  
  transition: 6s ease;  
  background-image: linear-gradient(to bottom right, #009096,  
}
```

Ilustración 501 - Imagen propia

```
.contenedor:hover .overlay {  
  height: 100%;  
}
```

Ilustración 502 - Imagen propia

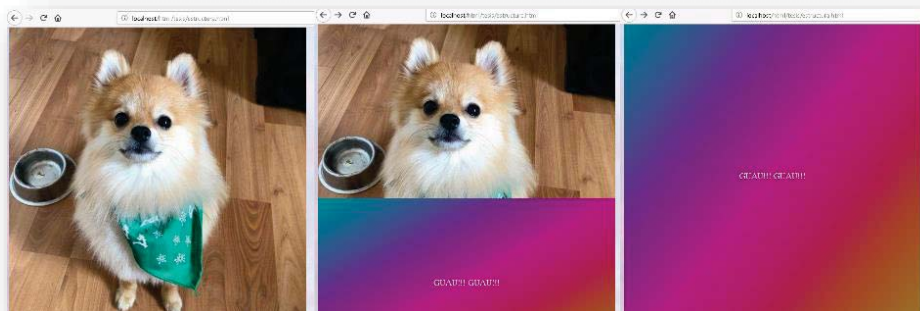


Ilustración 503 - Imagen propia

✓ DESLIZAMIENTO IZQUIERDO

Ahora bien, en esta parte todo se mantendrá de la misma manera, bajo los cambios correspondientes, pues dentro del movimiento **WIDTH** se mantendrá en cero y **HEIGHT** en 100%, mientras que en **HOVER** cambiará de **HEIGHT** a **WIDTH**.

```

.overlay {
  position: absolute;
  bottom: 0;
  left: 0;
  right: 0;
  height: 100%;
  width: 0;
  overflow: hidden;
  transition: 6s ease;
  background-image: linear-gradient(to bottom right, #009096, #bc0083, #998600);
}

.contenedor:hover .overlay {
  width: 100%;
}

```

Ilustración 504 - Imagen propia

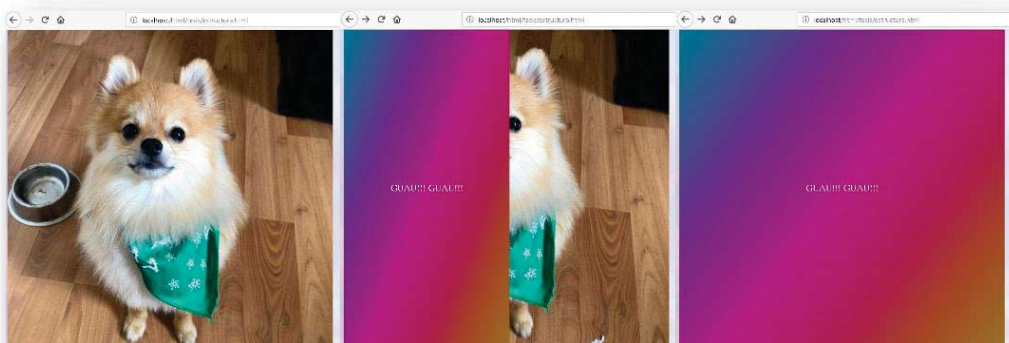


Ilustración 505 - Imagen propia

✓ DESLIZAMIENTO DERECHO

Y para el movimiento de desplazamiento derecho, lo único que cambia es que dentro del elemento movimiento, se le dará un valor de 100% al margen derecho y en **HOVER** se mantendrá **WIDTH** con el 100% que es el que genera la transición y se le colocara el margen **LEFT** en cero.

```

.overlay {
  position: absolute;
  bottom: 0;
  left: 100%;
  right: 0;
  height: 100%;
  width: 0;
  overflow: hidden;
  transition: 6s ease;
  background-image: linear-gradient(to bottom right, #009096, #bc0083, #998600);
}

.contenedor:hover .overlay {
  width: 100%;
  left: 0;
}

```

Ilustración 506 - Imagen propia

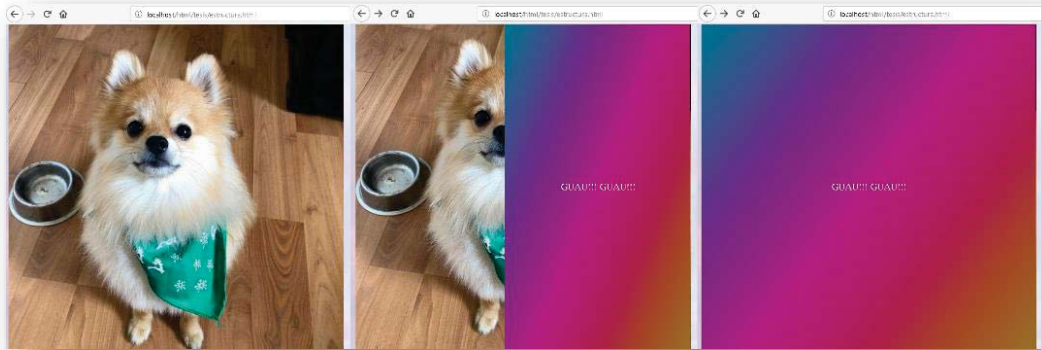


Ilustración 507 - Imagen propia

✓ DESVANECIMIENTO EN CAJA

Ahora bien, el movimiento en esta dinámica es similar, sólo varían unos elementos, pero es la misma función.

```
<div class="contenedor">
  
  <div class="middle">
    <div class="texto">GUAU!!! GUAU!!!</div>
  </div>
</div>
```

Ilustración 508 - Imagen propia

Como se puede observar aquí sólo cambia el nombre del movimiento. También cambiarán unas propiedades dentro de **IMAGE** y **MIDDLE**, dentro del primer elemento se le agregará una opacidad y una transición, esto es necesario porque el efecto permite resaltar el texto, entonces ya no se necesita color y el segundo elemento también tendrá una transición de tiempo, opacidad y la reubicación de los elementos.

```
.image {
  opacity: 1;
  display: block;
  width: 100%;
  height: auto;
  transition: 6s ease;
}

.middle {
  transition: 6s ease;
  opacity: 0;
  position: absolute;
  top: 50%;
  left: 50%;
  transform: translate(-50%, -50%);
  -ms-transform: translate(-50%, -50%)
}
```

Ilustración 509 - Imagen propia

```
.contenedor:hover .image {
  opacity: 0.3;
}

.contenedor:hover .middle {
  opacity: 1;
}

.texto {
  background-color: purple;
  color: white;
  font-size: 16px;
  padding: 16px 32px;
}
```

Ilustración 510 - Imagen propia

Y te preguntarás ¿Por qué dos transiciones? ¿Por qué la propiedad **TRANSFORM** está en otro lugar?, bueno, pues para el efecto que debe ejercer **CSS** necesita que sea por separado, pues como se puede observar en la segunda imagen, se tienen dos **HOVER**, esto es necesario para que exista una transición de imagen y otra del efecto para que en sincronía se armonice el asunto. Este es el resultado.

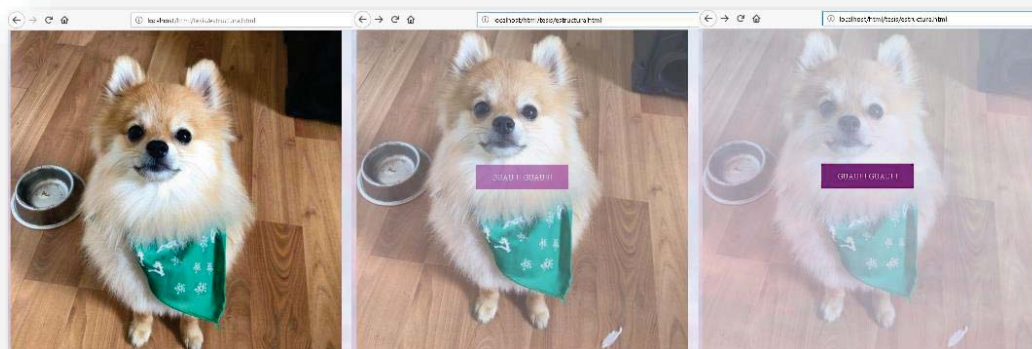


Ilustración 511 - Imagen propia

Pero con las imágenes aún no se acaba, pues **CSS** también permite crear slides o carruseles de imágenes, pues hoy en día las galerías de imágenes son muy importantes. Y para poder lograrlo, se hará uso de los elementos **<INPUT>** a los cuales se les asignará una imagen con **CSS**.

```
<div class="galeria">
  <input type="radio" name="navegacion" id="_1" checked>
  <input type="radio" name="navegacion" id="_2">
  <input type="radio" name="navegacion" id="_3">
  <input type="radio" name="navegacion" id="_4">
  <input type="radio" name="navegacion" id="_5">
  <input type="radio" name="navegacion" id="_6">

  
  
  
  
  
  
</div>
```

Ilustración 512 - Imagen propia

Declarado lo anterior, dentro de **CSS** se llama a la clase **GALERIA** a la cual se le asigna un ancho y alto a la o las imágenes que se quiere mostrar.

```
.galeria {
  height: 500px;
  width: 460px;
  margin: 10px;
  position: relative;
}
```

Ilustración 513 - Imagen propia

Después se llamará a **** anidada con el contenedor que es **GALERIA**, aquí se apilarán todas las imágenes con una posición absoluta, sin márgenes, con una transición de 3 segundos y todas serán declaradas invisibles para efectuar el movimiento, de igual manera se llamará a los **<INPUT>** anidados con **GALERIA**, a estos se les dirá que son de posición relativa para que permita el movimiento de la imagen que se seleccione con márgenes correspondientes a los márgenes del contenedor para que puedan ser visibles al público.

```
.galeria img {
  position: absolute;
  top: 0;
  left: 0;
  opacity: 0;
  transition: opacity 3s;
}

.galeria input[type=radio] {
  position: relative;
  bottom: -510px;
  left: 150px;
}
```

Ilustración 514 - Imagen propia

Ahora bien, para poder generar la transición, se llama al contenedor **GALERIA** anidado con **<INPUT>** y se hará uso de “~” para anclar la imagen, para dejar el tema más claro, lo que hará esa función es que al seleccionar cualquier **<INPUT>** aparecerá la imagen situada en esa posición y lo hará de manera delicada con la opacidad que se declaró a la imagen anteriormente.

```
.galeria input[type=radio]:nth-of-type(1):checked ~ img:nth-of-type(1) {
  opacity: 1;
}

.galeria input[type=radio]:nth-of-type(2):checked ~ img:nth-of-type(2) {
  opacity: 1;
}

.galeria input[type=radio]:nth-of-type(3):checked ~ img:nth-of-type(3) {
  opacity: 1;
}

.galeria input[type=radio]:nth-of-type(4):checked ~ img:nth-of-type(4) {
  opacity: 1;
}

.galeria input[type=radio]:nth-of-type(5):checked ~ img:nth-of-type(5) {
  opacity: 1;
}

.galeria input[type=radio]:nth-of-type(6):checked ~ img:nth-of-type(6) {
  opacity: 1;
}
```

Ilustración 515 - Imagen propia

Y de esta manera es como se obtiene un CARRUSEL, SLIDE o GALERIA.

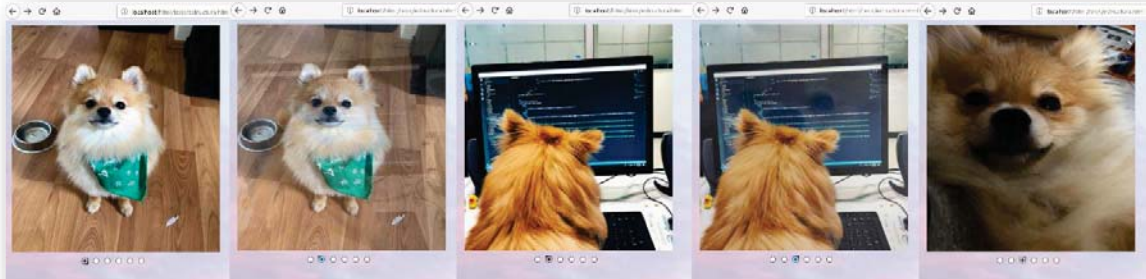


Ilustración 516 - Imagen propia

Ahora bien, dentro de CSS existe el Diseño Responsivo Web, que permite crear un **GRID**, este se encarga de volver sensibles a los elementos para que se ajusten al tamaño de pantalla que este visualizando la página web, por lo tanto, un **GRID** se rige por una cuadrilla donde al generar columnas se pueden lograr acomodar los elementos de manera rápida.

```
<head>
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link rel="stylesheet" type="text/css" href="css/estilo2.css">
</head>
<body>

  <div class="header">
    <h1>Header</h1>
  </div>

  <div class="row">
    <div class="col-3 col-s-3 menu">
      <h2>Article</h2>
      <ul>
        <li>Home</li>
        <li>Temass</li>
        <li>Ayuda</li>
        <li>Acerca de</li>
      </ul>
    </div>

    <div class="col-6 col-s-9">
      <h1>Contenido</h1>
      <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod
tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam,
quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo
consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse
cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non
proident, sunt in culpa qui officia deserunt mollit anim id est laborum.</p>
    </div>
  </div>
```

Ilustración 517 - Imagen propia

```
    <div class="col-3 col-s-12">
      <div class="aside">
        <h2>Aside</h2>
        <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod
tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam,
quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo
consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse
cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non
proident, sunt in culpa qui officia deserunt mollit anim id est laborum.</p>
      </div>
    </div>

    <div class="footer">
      <h2>Footer</h2>
      <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod
tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam,
quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo
consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse
cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non
proident, sunt in culpa qui officia deserunt mollit anim id est laborum.</p>
    </div>
```

Ilustración 518 - Imagen propia

Como se puede ver en la imagen de arriba se tiene un nuevo **<META>** dentro de esta etiqueta se declara que todo el contenido se volverá sensible a la adaptación responsiva de elementos, por supuesto esto se volverá real cuando se adapte con **CSS**, por otro lado, se encuentran “CLASES” donde se les declaran columnas, estas permiten hacer la rejilla como tal y, por lo tanto, se puede lograr hacer celdas con las mismas, de esta manera se puede facilitar el ajuste de los elementos sin problemas.

Dentro de **CSS**, se declaran los respectivos colores, rellenos, márgenes y decoraciones de lo que se incluyó en **HTML5**, pero sin duda, se encuentra algo nuevo, pues la clase **ROW** que contiene a las columnas se le establecen los valores correspondientes de una tabla, pues por defecto, está creando una tabla como tal con toda la página, después por medio de “**@MEDIA**” se les declara el tamaño correspondiente a las columnas para que se adapten a las tabletas y móviles. Pues con eso se logra algo similar a esto:

```

* {
  box-sizing: border-box;
}

.row::after {
  content: "";
  clear: both;
  display: table;
}

[class*="col-"] {
  float: left;
  padding: 15px;
}

html {
  font-family: temas;
  font-size: 14px;
}

.header {
  background-color: #9933cc;
  color: #ffffff;
  padding: 15px;
}

.menu ul {
  list-style-type: none;
  margin: 0;
  padding: 0;
}

.menu li {
  padding: 8px;
  margin-bottom: 7px;
  background-color: #33b5e5;
  color: #ffffff;
  box-shadow: 0 1px 3px rgba(0,0,0,0.12), 0 1px 2px rgba(0,0,0,0.24);
}

.menu li:hover {
  background-color: #0099cc;
}

.aside {
  background-color: #33b5e5;
  padding: 15px;
  color: #ffffff;
  text-align: center;
  font-size: 12px;
  box-shadow: 0 1px 3px rgba(0,0,0,0.12), 0 1px 2px rgba(0,0,0,0.24);
}

.footer {
  background-color: #0099cc;
  color: #ffffff;
  text-align: center;
  font-size: 12px;
  padding: 15px;
}

[class*="col-"] {
  width: 100%;
}

@media only screen and (min-width: 600px) {
  /* For tablets: */
  .col-s-1 {width: 8.33%;}
  .col-s-2 {width: 16.66%;}
  .col-s-3 {width: 25%;}
  .col-s-4 {width: 33.33%;}
  .col-s-5 {width: 41.66%;}
  .col-s-6 {width: 50%;}
  .col-s-7 {width: 58.33%;}
  .col-s-8 {width: 66.66%;}
  .col-s-9 {width: 75%;}
  .col-s-10 {width: 83.33%;}
  .col-s-11 {width: 91.66%;}
  .col-s-12 {width: 100%;}
}

@media only screen and (min-width: 768px) {
  /* For desktop: */
  .col-1 {width: 8.33%;}
  .col-2 {width: 16.66%;}
  .col-3 {width: 25%;}
  .col-4 {width: 33.33%;}
  .col-5 {width: 41.66%;}
  .col-6 {width: 50%;}
  .col-7 {width: 58.33%;}
  .col-8 {width: 66.66%;}
  .col-9 {width: 75%;}
  .col-10 {width: 83.33%;}
  .col-11 {width: 91.66%;}
  .col-12 {width: 100%;}
}

```

Ilustración 519 - Imagen propia

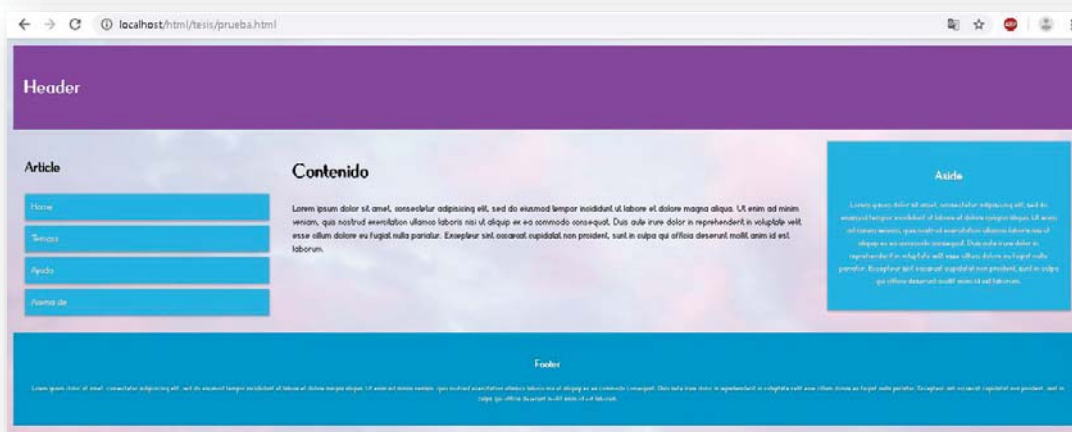


Ilustración 520 - Imagen propia

Por supuesto que la distribución de las cajas y el contenido variará dependiendo del concepto de cada página o sitio web. Comprendido todo lo anterior, no queda duda que se pueden generar miles y miles de estilos para un sitio web, del tema que sea, se pueden lograr muchas cosas, muy buenas, bien hechas, sofisticadas y fuera de lo común, no queda más que mencionar que con **CSS** a pesar de ser un contenido extenso, con muchas propiedades y un sinnúmero de combinaciones, es el elemento que más se utiliza dentro del Diseño Web, nada existe sin **CSS**.



Ilustración 521 - Imagen propia, imágenes ilustrativas tomadas de <https://co.pinterest.com/pin/344314333990534602/>

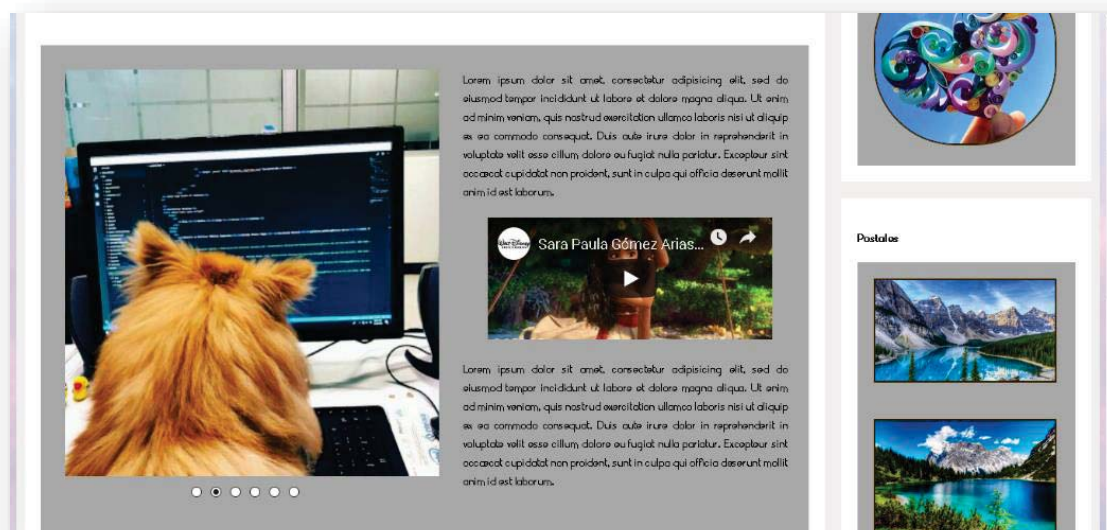


Ilustración 522 - Imagen propia, imágenes ilustrativas tomadas de <https://co.pinterest.com/pin/344314333990534602/>, <https://www.viajejet.com/paisajes-campo/>, <https://www.dzoom.org.es/guia-completa-fotografia-paisaje/>, video tomado de <https://www.youtube.com/watch?v=ftNOCfq0ljQ>

Poco texto representativo

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.



Nombre	N. Cuenta	Localidad
Ana Contreras	456732456	CDMX
Omar Rojas	232458900	Monterrey
Eduardo Ozuna	432533212	Toluca
Monica Pérez	313234565	Hidalgo
Juan Moreno	232415608	Estado de México

Ilustración 523 - Imagen propia, imagen ilustrativa tomada de <http://www.fondosni.com/wallpapers-gratis/Paisaje-de-agua/400x250.html>

Redes sociales



Hecho con  para todos.

©Copyright 2019 Abril Alba All rights reserved

Ilustración 524 - Imagen propia, iconos tomados de <https://fontawesome.com/>

9. INTRODUCCION A JAVASCRIPT

A lo largo del tiempo, **HTML5** ha permitido dotar a las páginas web de información visualmente atractiva al cliente, pero en su proceso, ha tenido que permitir la implementación de nuevas herramientas para ejercer una mejora dentro de su contenido, para ello el primer paso fue el comienzo de **CSS**, pues permitió darles esa visualización más encantadora a los sitios web, pero no obstante seguía existiendo un cierto desencanto, pues las páginas web no se permitían ninguna interacción solas o con el usuario, con el pasar del tiempo, se comenzó una nueva tecnología dentro de las empresas desarrolladoras de software, Netscape se dispuso a desarrollar ordenes fáciles de utilizar que en conjunto dio nacimiento a **LIVESCRIPT**, pero no se encontraba del todo desarrollada, por lo cual Sun Microsystems creadora de “JAVA” se unió en colaboración con Netscape y así consiguieron que LiveScript se adaptará a ser un lenguaje estándar de internet para realizar ordenes dentro de los sitios web, ahora bien, **LIVESCRIPT** tenía mucha semejanza al lenguaje “JAVA” y por esa razón fue renombrado como **JAVASCRIPT**, Sánchez (2012).

9.1.JAVASCRIPT

Pero te preguntaras ¿Qué es **JS**?, Con el tiempo **JAVASCRIPT** fue consolidándose como un lenguaje orientado a objetos, diseñando para realizar ordenes de comportamiento para aplicaciones web a través de internet. Para esos momentos el desarrollo web se cimentó dentro de internet con sus tres pilares, pues **HTML5** brinda los elementos que conforman un esqueleto web, **CSS** se encarga de organizar y brindar una apariencia a los sitios web y la interactividad de **JS** termina por brindarle en toque final, dándole un giro de 360 grados a las páginas web, desde ese momento todo internet se volvió loco por lo que ahora un sitio web podía ofrecer, Sánchez (2012).

Es muy importante resaltar que **JS** se utiliza “de lado del cliente”, esto se suscita porque **JS** se ejecuta en el navegador del usuario o del cliente, por lo cual no está dentro de un servidor como lo es **HTML5** para que se pueda ejecutar, otro dato interesante, es que **JS** es traducido por el navegador que este visualizando el sitio web. Por lo tanto, el lenguaje de programación script se inserta dentro del documento **HTML5** y se ejecuta en el navegador del usuario al cargar la página. Este lenguaje permite variar dinámicamente el contenido del documento, validar formularios, modificar el compartimento normal del navegador, etc. Lo que no permite es manejar bases de datos, ya que como se ha comentado anteriormente, los scripts se ejecutan en la máquina del usuario y no en el servidor en donde está alojada la página, Sánchez (2012).

Sin embargo, hoy en día **JS** ya no solo es el lenguaje de programación web, que permite crear pequeños programas que realizan interacciones en una página, ahora se ha vuelto tan importante dentro del desarrollo de tecnología lógicas que se abrió campo en el desarrollo de aplicaciones como Facebook, Google, Twitter, pues sus posibilidades de crear se explotan al máximo para lograr generar núcleos de código de **JS** para la interacción de lo que hoy se conoce como una Red Social, también se encuentra dentro del desarrollo de software para dispositivos electrónicos, Sánchez (2012).

Pero bueno, todo suena súper padre y que **JS** es increíble, pero ¿Cómo funciona **JS**? ¿Cómo se implementa?, para partir y crear conocimientos sólidos de **JS**, es importante recalcar que para un principiante **JS** significa que es un lenguaje de programación web, por lo tanto, en su proceso de conocimiento, forjará una idea del manejo de **JS** y volcará al mismo en una nueva dirección en donde podrá crear lo que quiera, partiendo de ello, es necesario tener conocimientos firmes sobre **HTML5** y **CSS** para poder interactuar con **JS**.

9.2.HOLA MUNDO

Dentro de la primera programación en **JS**, es muy importante para los desarrolladores lograr la primera visualización de código en pantalla, de alguna manera es la forma de presentarse. Por ello, dentro del mundo de **JS** se tienen tres maneras interesantes de mostrar el primer “HOLA MUNDO”, como se sabe que **JS** se ejecuta dentro del navegador donde se esté presentando la página web, esto brinda la oportunidad de interactuar dentro de la consola en el navegador, pues permite desarrollar muchas tareas en los navegadores, y una de ellas se permite mostrar los posibles errores que se tenga en el código y resolverlos de manera lógica dentro del navegador, bueno, pues así es como se logra un “HOLA MUNDO”.

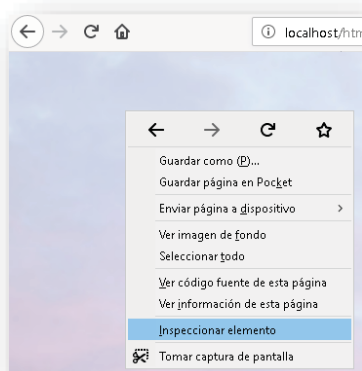


Ilustración 525 - Imagen propia

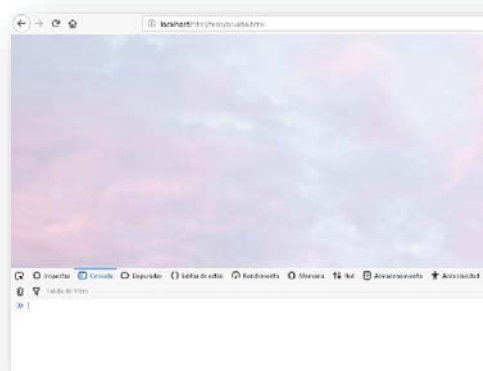


Ilustración 526 - Imagen propia

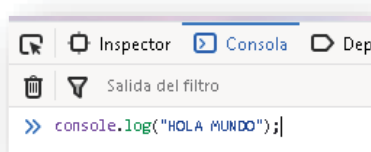


Ilustración 527 - Imagen propia

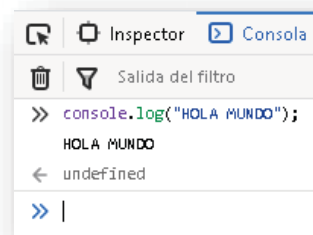


Ilustración 528 - Imagen propia

También es posible con:

```
<script type="text/javascript">
alert("Hola Mundo");
</script>
```

Ilustración 529 - Imagen propia

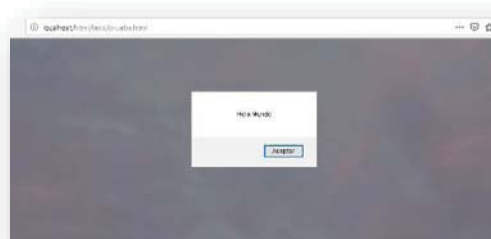


Ilustración 530 - Imagen propia

Y finalmente, interactuando con el DOM:

```
<script type="text/javascript">
document.write("Hola Mundo");
</script>
```

Ilustración 531 - Imagen propia

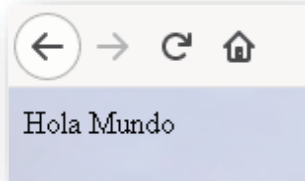


Ilustración 532 - Imagen propia

9.3. ESTRUCTURA

JS se caracteriza por declaración de pequeñas líneas de código que se llaman **SCRIPTS**, estos se incrustan o insertan dentro del código **HTML5**, de preferencia entre las etiquetas **<HEAD>** y **<BODY>** para que lo primero en cargarse sean los **SCRIPTS** de **JS**, de lo contrario también existe la posibilidad de anclar un documento con extensión **JS** como se hace con **CSS**, la única diferencia es que este se inserta por medio de un **<SCRIPT>**, pero dentro de este se utilizará **SRC** de este modo, la dirección del documento se colocará ahí, y se cargaran todos los **SCRIPTS** necesarios para una página web o para todo el sitio web, esto depende del diseñador.

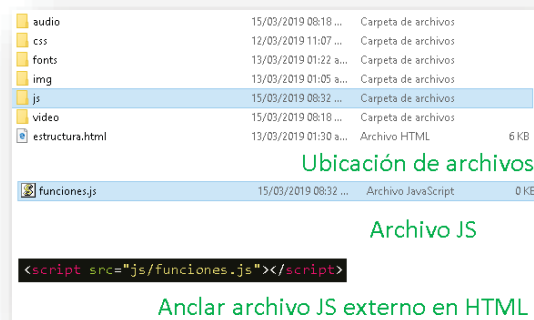


Ilustración 533 - Imagen propia

Antes de comenzar a teclear código, se debe tomar en cuenta las bases con las que se trabaja **JS**. Para poder construir un **SCRIPT**, se deben conocer la sintaxis de **JS**, para eso se tienen 3 bloques que lo conforman:

- ✓ **Variables:** Son contenedores que pueden albergar valores.

```
<script type="text/javascript">
var x, y, z; <- Como se declaran la(s) variables
x = 2; y = 4; <- Como se asignan lo(s) valores a las variables
z = x + y; <- Como se calculan los resultados de la(s) variables
</script>
```

Ilustración 534 - Imagen propia

- ✓ **Literales:** Son variables definidas de inicio, es decir, los valores que contengan las variables serán literales hasta una nueva asignación de valor, para ser más exactos el primer valor que se declara en cualquier variable, es una variable literal, pues su valor permanecerá así hasta tener un cambio.

```
<script type="text/javascript">
    var nombre = Alba, respuestaExamen = true, encuesta = null;
</script>
```

Ilustración 535 - Imagen propia

- ✓ **Constantes:** Estos contenedores de datos, son igual que las variables, su única diferencia es que el valor que contengan va a ser para siempre, nada ni nadie puede cambiar ese valor, es como si se declarará por nombre “PI” y el valor sea “3.1415”, por lo tanto, ese valor va a ser constante, no puede cambiar y para llamar a las constantes se hace por medio de la palabra reservada **CONST**.

Dentro de la sintaxis en **JS** define que existen diferentes tipos de valores que pueden contener las variables:

- ❖ **ENTERO:** Son números de cualquier naturaleza, es decir, números enteros, primos, naturales, positivos y negativos.
- ❖ **FLOTANTE:** Son los números que se encuentran de manera decimal, es decir, números que contengan decimales.
- ❖ **LÓGICO (BOOLEANO):** Este valor hace referencia al verdadero y falso de algún elemento o comparación del mismo, por lo tanto, un valor lógico identifica comparaciones entre diferentes valores.
- ❖ **NULOS:** Un valor nulo, es todo aquel valor no declarado, es decir, es un valor vacío.
- ❖ **INDEFINIDOS:** Existe una gran incógnita entre un valor nulo y un valor indefinido, pues ambos valores no son declarados dentro del código, la diferencia entre ambos es que un valor nulo es un valor literal, es decir, la variable contendrá un valor nulo de inicio, no tendrá un tipo (String o Booleano o Entero o Flotante) hasta una nueva reasignación, mientras que un valor indefinido es cuando se declara la variable pero no se escribe nada más, por lo tanto y por default el navegador declara a esa variable con un valor indefinido y sin tipo.
- ❖ **STRING** (cadena de caracteres): Son los valores en cadenas de letras o caracteres, es decir, esta variable contiene letras, palabras o concatenación de palabras.

Como es de esperarse, entender que es una variable dentro de la programación de cualquier lenguaje, es específicamente un pedacito de memoria en el que se registra con un nombre y se le asigna un valor, esto es para que el programa funcione mediante el contenido de las mismas, cuando se le asigna un nombre a una variable, se le denomina “IDENTIFICADOR”.

Como se mencionó anteriormente, aunque se esté hablando de un lenguaje de programación y en este momento se hable de variables y después de condicionales y bucles, etc., este tema no está relacionado al Back-End, ni a PHP, ni a bases de datos, este lenguaje y lo que se va a conocer es en referencia al Front-End, al diseño, al cliente, lo que se visualiza.

Aunque la declaración de una variable sea semejante a la de otros lenguajes de programación, JS permite declarar las variables sin especificar el tipo de valor que contendrá, pues para JS es indistinto ese detalle, porque el navegador en donde se ejecute la página web, se encargará de traducir o interpretar las variables y asignarles que tipo de valor son.

Para declarar una variable se hace como se muestra en la imagen anterior, si JS será insertado dentro de HTML5 se comienza abriendo el pequeño guion de JS con un **<SCRIPT>** y dentro se coloca el código, si no es así, en un documento externo con extensión JS se colocará todo el código sin **<SCRIPT>**, aclarado lo anterior, en la primera línea de código, se insertará la primera palabra **VAR** la cual simboliza la declaración de una variable.

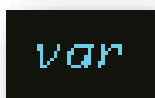
Una imagen que muestra el código 'var' escrito en un editor de código con un fondo oscuro y letras de color cian.

Ilustración 536 - Imagen propia

La continuación sugiere que dentro de JS, al momento de declarar una variable, se le asigne un nombre a la variable de motivo específico, concreto, lógico y único del valor que va a almacenar, es decir, tiene que declarar un nombre descriptivo de lo que contendrá, por ejemplo; si va a guardar un valor tipo **STRING** y recibirá el nombre de alguna persona, entonces ese es el identificador sugerente para la variable **NOMBRE**, pues describe perfectamente el valor que poseerá, si guarda el valor de un nombre de animal, el respectivo identificador sería **ANIMAL**.

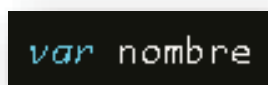
Una imagen que muestra el código 'var nombre' escrito en un editor de código con un fondo oscuro y letras de color cian.

Ilustración 537 - Imagen propia

Para términos más específicos, existen dos tipos de variables, las locales y globales, la primera es como se especifica en la ilustración anterior, pues al declarar la variable con la palabra reservada del código **VAR**, automáticamente se vuelve una variable local, esta solo existe específicamente para un método o función de JS, mientras que, si la variable se declara sin ella, es una variable global y puede utilizarse por cualquier función y/o método que se requiera.

Una imagen que muestra el código 'nombre' escrito en un editor de código con un fondo oscuro y letras de color cian.

Ilustración 538 - Imagen propia

Por otro lado, es que, para declarar el nombre de una variable, siempre tiene que comenzar por una letra, un “_” (guion bajo) o un signo de “\$” (dólar), cabe destacar que JS es sensible

a mayúsculas y minúsculas, por lo tanto, si se declara “nombre” y “Nombre” dentro de JS eso significa que son dos variables completamente diferentes.

```
var _$nombre  
var nombreApellido  
var _2nombre_Apellido
```

Ilustración 539 - Imagen propia

Lo que puede contener el identificador son letras, números, guiones y símbolos de dólar. Pero nunca puede comenzar con un número ni tener espacios en blanco.

Entendido lo anterior, seguido del identificador se coloca un signo “=” que especifica que le va a asignar un valor del tipo que sea.

```
var nombre = "Abril"
```

Ilustración 540 - Imagen propia

Para el momento de asignar un valor a la variable, es necesario tener en cuenta ciertas reglas, en el caso de declarar valores **STRING** es necesario que la cadena de caracteres se mantenga dentro de comillas simples o dobles.

```
var nombre = "Abril"  
var nombre = 'Abril'
```

Ilustración 541 - Imagen propia

Si un valor **STRING** se encuentra sin los dos tipos de comillas, por completo para a ser un valor indefinido, pues su valor no fue declarado como debe de ser o no se le otorgó valor, todo depende del manejo que se le dé al código. Por lo tanto, ese problema existirá en cualquier tipo de valor si no se le introduce algún número, o carácter al identificador.

Ahora bien, cuando se declaran muchas variables, es muy importante separarlas por medio de “,” (comas), de esta manera será más claro y acertado para el navegador, y al concluir la línea de código se coloca un “;” (punto y coma).

- ✓ **Objetos:** Es un contenedor de variables y métodos, cada valor refleja una propiedad individual de este objeto, para aterrizar el concepto, un objeto es la unión de propiedades (**VARIABLES**) y funciones (**MÉTODOS**) que lo caracterizan.

```
const celular = { marca: "Samsung",
                  color: "RoseGold",
                  precio: 10000}

console.log(celular);
```

Ilustración 542 - Imagen propia

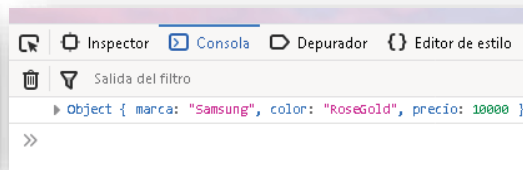


Ilustración 543 - Imagen propia

CELULAR	Propiedades (Variables)		Métodos(Funciones)	
	marcaCelular = Samsung		llamar.celular()	
	colorCelular = Dorado		navegar.celular()	
	precioCelular = 10000		chatear.celular()	

Ilustración 544 - Tabla propia

En lenguajes de programación orientados a objetos (POO), se tiene que programar siempre en base a objetos. Para ello, se crean **CLASES**, que son una especie de "moldes" a partir de los cuales se crean **OBJETOS**. La creación de una **CLASE** dentro de **JS** se observa de esta manera:

```
<script type="text/javascript">

function Celular (marca, color, precio) {
  this.marca = marca;
  this.color = color;
  this.precio = precio;

  this.llamar = function(){
    document.write("Llamar a mamá");
  }
  this.navegar = function(){
    document.write("Entrando a Facebook");
  }
  this.chatear = function(){
    document.write("Platicando con amigos");
  }
};

var myCelular = new Celular("Samsung", "RoseGold",
10000);

document.write(myCelular.color);

</script>
```

Ilustración 545 - Imagen propia

Se crea la clase por medio de una función, muy semejante a como se declara un objeto, solo que en este caso se añaden los métodos que realiza el objeto, por otro lado, se encuentra **THIS** que permite presentar los elementos del objeto y utilizar todas las funciones de las propiedades y métodos. Por medio de la sintaxis punto se puede acceder a los elementos del objeto.

- ✓ **Funciones:** Son los procedimientos de una aplicación a ejecutar, es decir, es el conjunto de declaraciones que ejecutarán una acción.

```
function nombre_de_la_funcion(parametros){
    declaraciones de la funcion;
}
```

Ilustración 546 - Imagen propia

Para términos más específicos, una función dentro de **JS**, es un bloque de código diseñado para realizar una tarea en concreto, dentro de su sintaxis, se encuentra la palabra clave **FUNCTION**, esta invoca el inicio de una función, seguido se le sentenciará un nombre alusivo a la función que desempeñará, cabe destacar, como nota importante, que el nombre que se le asigne a la función, contiene las mismas reglas que al declarar el nombre de una variable, después se tienen unos paréntesis que contendrán parámetros que serán utilizados por la función, estos pueden variar entre variables u otros valores, también puede y que en algún caso no contenga parámetros la función y solo se queden los paréntesis vacíos, no existe ningún error, pues esto dependerá del desarrollo del código e ideología que plante el programador, pero si es necesario que la sintaxis de la función este completa como se muestra en la imagen anterior, después se continua con la apertura de llaves, las cuales contendrán las acciones que deberá realizar la función.

```
<script type="text/javascript">
function saludo() {
    document.write("Hola a todos");
}
saludo();
</script>
```

Ilustración 547 - Imagen propia

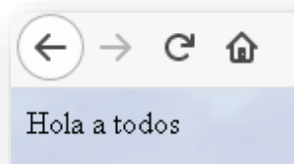


Ilustración 548 - Imagen propia

Como se muestra en el ejemplo anterior, se crea una función donde su principal objetivo es saludar al usuario, también se puede observar que al final se encuentra nombrada la función para poder visualizarla en el navegador, es decir, cuando se declaran funciones en **JS**, por si solas no se presentan en pantalla porque el navegador no las detecta como código presencial para el usuario, comúnmente se hacen aparecer por medio del nombramiento de las mismas, de esta manera cuando el navegador que este ejecutando la página detecte que se está solicitando a la función, busca dentro del código que tareas realiza esta función y las hace trabajar.

En el siguiente ejemplo, se tiene la **FUNCTION RESTA**, aquí se utilizan los parámetros con unas variables, dentro del contenido se encuentra una sola línea de código, esa palabra reservada funge como reflector del resultado de la función, de esa manera cuando se hace llamar a la función se ejecuta de igual manera que el ejemplo anterior.

```

<script type="text/javascript">
  function resta(a,b) {
    return a-b;
  }
  document.write(resta(20,10));
</script>

```

Ilustración 549 - Imagen propia

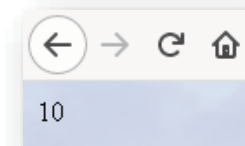


Ilustración 550 - Imagen propia

9.3.1. OPERADORES

Dentro del mundo de las variables y sus valores, existen operadores que permiten realizar operaciones aritméticas de datos, comparaciones y operaciones lógicas, en JS existen los siguientes operadores:

- Operadores de comparación

Igual	→	==
Distinto	→	!=
Mayor que	→	>
Menor que	→	<
Menor o igual a	→	<=
Mayor o igual a	→	>=

Ilustración 551 - Tabla propia

Dentro de estos operadores, se maneja el resultado lógico o booleano, por ende, como comparaciones, aquí se busca tener una respuesta positiva o negativa ante lo que se esté comparando.

```

<script type="text/javascript">
  var numeroUno = 20;
  var numeroDos = 40;

  var resultado = numeroUno < numeroDos;

  document.write(resultado);
</script>

```

Ilustración 552 - Imagen propia

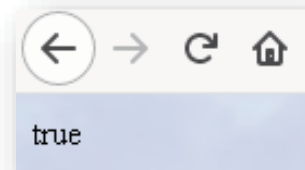


Ilustración 553 - Imagen propia

- Operadores de cadena

Los operadores en cadena consisten en concatenar caracteres, esto se realiza por medio del signo suma “+” aunque en esta situación, no suma a los caracteres, sino que los une, si se utiliza “**abril**” + “ ” + “**alba**”, el resultado será “**abril alba**”, las comillas que se encuentran

en medio de los dos signos de suma, permiten especificar al navegador un espacio en blanco dentro de la oración.

```
<script type="text/javascript">
    var nombre = "Abril";
    var apellido = "Alba";

    var dato = nombre + " " + apellido;

    document.write(dato);
</script>
```

Ilustración 554 - Imagen propia



○ Operadores aritméticos

+	Suma
-	Resta
*	Multiplicación
/	División
%	Residuo de división
++	Incremento
--	Decremento

Ilustración 556 - Tabla propia

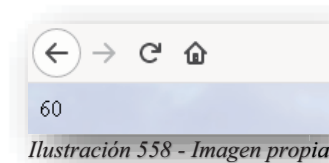
En los operadores aritméticos, solo se obtienen operaciones comunes que se conocen desde la primaria, por lo tanto, no existe gran ciencia, lo que sí es importante es que, al momento de utilizarse dentro de una condicional o un código elaborado de programación, su función se vuelve más esencial.

```
<script type="text/javascript">
    var numeroUno = 20;
    var numeroDos = 40;

    var resultado = numeroUno + numeroDos;

    document.write(resultado);
</script>
```

Ilustración 557 - Imagen propia



○ Operadores de asignación

=	Asigna el valor de la parte derecha del signo igual a la parte de la izquierda donde se encuentra la variable
+=	Realiza la suma del valor de la parte de la derecha con la de la izquierda (otro valor o una variable que tenga otro valor) y guarda el resultado en la parte de la izquierda (la variable designada para guardar ese dato).

-=	Realiza la resta del valor de la parte de la derecha con la de la izquierda (otro valor o una variable que tenga otro valor) y guarda el resultado en la parte de la izquierda (la variable designada para guardar ese dato).
*=	Realiza la multiplicación del valor de la parte de la derecha con la de la izquierda (otro valor o una variable que tenga otro valor) y guarda el resultado en la parte de la izquierda (la variable designada para guardar ese dato).
/=	Realiza la división del valor de la parte de la derecha con la de la izquierda (otro valor o una variable que tenga otro valor) y guarda el resultado en la parte de la izquierda (la variable designada para guardar ese dato).
%=	Realiza la deducción del resto de una división del valor de la parte de la derecha con la de la izquierda (otro valor o una variable que tenga otro valor) y guarda el resultado en la parte de la izquierda (la variable designada para guardar ese dato).

Ilustración 559 - Tabla propia

En estos operadores, su función es igual a los operadores aritméticos, la diferencia radica en que el resultado se le asigna a la variable que se encuentra de lado izquierdo, es decir, se le reasigna un valor nuevo a la primera variable.

```
<script type="text/javascript">
var numeroUno = 20;
var numeroDos = 40;

numeroUno *= numeroDos;

document.write(numeroUno);
</script>
```

Ilustración 560 - Imagen propia

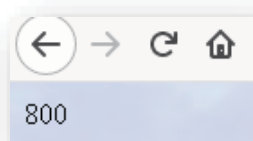


Ilustración 561 - Imagen propia

○ Operadores lógicos

Estos operadores sirven para realizar operaciones lógicas, son aquellas que dan como resultado un verdadero o un falso.

&&	Y
 	O
!	NO

Ilustración 562 - Tabla propia

9.3.2. COMENTARIOS

Cuando se comienza a programar en cualquier lenguaje, se suelen utilizar comentarios dentro del código, esto es con la finalidad de llevar un control del código, es decir, la organización y distribución del código suele ser muy extensa, por eso los archivos para diseño web con extensiones de **HTML5**, **CSS**, **JS**, porque si estuvieran unidos en uno solo, sería un gran dolor de cabeza tener que leer línea por línea en los diferentes entornos, por eso es imprescindible separar el código en sus 3 pilares, anclarlos al principal y dentro de cada uno de los archivos marcar trozos del código y especificar con detalle que función realiza ese código, de esta manera cuando exista actualización o mantenimiento del código, será más fácil entender su función y organización. Para hacer un comentario de una sola línea o un comentario corto, se hace uso de un doble slash “//”.

```
<script type="text/javascript">
    //comentarios cortos
</script>
```

Ilustración 563 - Imagen propia

Y cuando es necesario definir y ser preciso en la función del código, es decir, que se tiene que escribir con detalle más de una línea, existe la posibilidad de declarar un comentario extenso por medio de los siguientes signos “/**/”.

```
<script type="text/javascript">
    /*comentarios
       muy
       largos
    */
</script>
```

Ilustración 564 - Imagen propia

9.4. ESTRUCTURAS DE CONTROL DE FLUJO

Dentro de la programación no existe ningún programa con secuencia de líneas, también son complementados con estructuras de control de flujo que permiten condicionar al programa de cierta manera para cumplir con una tarea, de esta forma permiten variar la ejecución del código, es decir, al declarar una condición dentro del mismo programa, este al ejecutarse condicionará al código mientras obtenga decisiones del usuario para poder continuar con el resto del código, siendo más exactos, se le brinda la libertad al programa de tomar decisiones dependiendo de lo que el usuario introduzca en pantalla. Ahora bien, las estructuras de control se dividen en dos grupos, que son: Estructuras condicionales y estructuras repetitivas, también conocidas como los famosos bucles.

Las estructuras condicionales son aquellas sentencias que se aplican dentro del código para la toma de decisión, es decir, una condición dentro del código incita al mismo programa a

tomar una decisión con respecto a los límites y datos que contiene del usuario, esto solo se logra contemplando y cumpliendo con los parámetros que se le otorguen al programa. Las estructuras que se tienen para condicionar son:

- **IF**: Esta condición se encarga de hacer una evaluación de datos, por medio de eso, el programa arrojará una respuesta.

```
<script type="text/javascript">
    if (condicion) { //si se cumple esto
        //realiza estas instrucciones
    }
</script>
```

Ilustración 565 - Imagen propia

Un ejemplo claro es la respuesta de la siguiente imagen:

```
<script type="text/javascript">
var numeroUno = 20;
var numeroDos = 40;

if (numeroUno < numeroDos) { //si se cumple esto
    document.write("Si es menor");//realiza estas instrucciones
}
</script>
```

Ilustración 566 - Imagen propia

Para lenguaje natural y sea más comprensible, se puede traducir de la siguiente manera: “Si número uno es menor a número dos, entonces házmelo saber”, en dado caso que no se cumpla la condición simplemente no arrojará nada en pantalla.

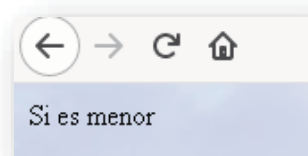


Ilustración 567 - Imagen propia

- **IF-ELSE**: Su función es similar a la anterior, su única diferencia es la complementación de la condición, pues solo se le anexa una segunda respuesta en caso de no cumplir con la primera condición.

```
<script type="text/javascript">
    if (condicion) { //si se cumple esto
        //realiza estas instrucciones
    } else { //sino
        //cumple estas instrucciones
    }
</script>
```

Ilustración 568 - Imagen propia

Para entenderse mejor, se supone un ejemplo, el ingreso de una contraseña, si esta contraseña es igual a la que se tiene registrada, se podrá ingresar a la cuenta.

```
<script type="text/javascript">
var contraseña = "alba1234";
if (contraseña == "alba1234") { //si se cumple esto
    document.write("Bienvenida Abril");//realiza estas instrucciones
}
else{ //sino
    document.write("Contraseña incorrecta");
}
</script>
```

Ilustración 569 - Imagen propia

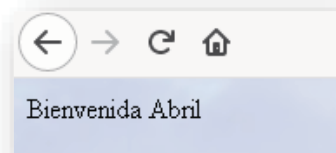


Ilustración 570 - Imagen propia

Sino es correcta, que lo haga saber.

```
<script type="text/javascript">
var contraseña = "alba1232";
if (contraseña == "alba1234") { //si se cumple esto
    document.write("Bienvenida Abril");//realiza estas instrucciones
}
else{ //sino
    document.write("Contraseña incorrecta");
}
</script>
```

Ilustración 571 - Imagen propia

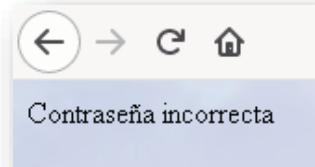


Ilustración 572 - Imagen propia

- **IF – ELSE IF – ELSE:** Esta condición es similar a la anterior, pero esta vez, permite hacer una mayor evaluación de datos y, por ende, es aún más explícita la respuesta.

```
<script type="text/javascript">

if (condicion) { //si se cumple esto
    //realiza estas instrucciones
}
else if (condicion) { //si se cumple esto
    //realiza estas instrucciones
}
else{ //sino
    //realiza estas instrucciones
}

</script>
```

Ilustración 573 - Imagen propia

A veces dentro de los datos que se manejar en una programación, suelen provocar problemas cuando son muy generales y lo que el programa busca es generar una respuesta más exacta, es por eso que se hace un tipo colador dentro de las condiciones de esta estructura, porque se permite especificar una respuesta con lo que realmente necesita.

```
<script type="text/javascript">

var edad = 21;

if (edad >= 18 ) { //si se cumple esto
    document.write("Eres mayor de edad");
}
else if (edad >= 60) { //sino se cumple, busca esto
    document.write("Eres de la tercer edad");
}
else{ //sino
    document.write("Eres menor de edad");
}

</script>
```

Ilustración 574 - Imagen propia

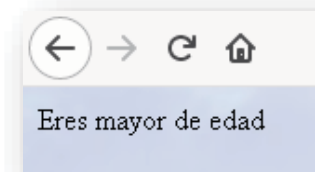


Ilustración 575 - Imagen propia

- **SWITCH:** Esta estructura es muy útil cuando se trata de poner bajo testeo a una expresión, pues su función es localizar la coincidencia que tenga la expresión con alguno de sus casos y poder generar las tareas que se tengan.

```
<script type="text/javascript">

switch (expresion) {
case opcion1:
//Sentencias ejecutadas cuando el resultado de la expresion coincide con opcion1.
break; //cierne de caso.
case opcion2:
//Sentencias ejecutadas cuando el resultado de la expresion coincide con opcion2.
break; //cierne de caso.
case opcionN:
//Sentencias ejecutadas cuando el resultado de la expresion coincide con opcionN.
break; //cierne de caso.
default:
//Sentencias ejecutadas cuando no ocurre una coincidencia con los anteriores casos.
break; //cierne de caso.
}

</script>
```

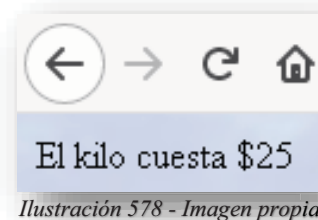
Ilustración 576 - Imagen propia

En caso de que la expresión no coincida con ninguno de los casos anteriores, se tiene una opción que permite generar una tarea alterna. Para comprender aún mejor cómo se desarrolla esta condición, por ejemplo, se quiere saber si la expresión, valor o elemento a evaluar se encuentra dentro de las opciones que tiene, si coincide con alguna de ellas, el programa arrojará a pantalla lo que sea que contenga ese caso, sino arrojará lo que tiene **DEFAULT**.

```
<script type="text/javascript">
var fruta = "platano";

switch (fruta) {
case "manzana":
    document.write("El kilo cuesta $20.50");
break;
case "uva":
    document.write("El kilo cuesta $10.50");
break;
case "pera":
    document.write("El kilo cuesta $12.50");
break;
case "guayaba":
    document.write("El kilo cuesta $6.50");
break;
case "platano":
    document.write("El kilo cuesta $25");
break;
default:
    document.write("Se acabo esa fruta. Hasta pronto.");
break;
}
</script>
```

Ilustración 577 - Imagen propia



Así como se mostraron las estructuras condicionales, toca el turno de presentar las estructuras loop o estructuras repetitivas, los bucles ofrecen una manera rápida y sencilla de hacer algo repetidamente, su finalidad es similar a un **IF**, la diferencia radica en que el bucle repetirá el código las veces que sea necesario hasta conseguir que la condición se cumpla, es muy importante tener cuidado con estas estructuras y saber poner los límites dentro de ellas, pues puede volverse un bucle infinito y provocar que las cosas se pongan feas, en las estructuras anteriores se tiene una respuesta a la condición de manera lógica o booleana, es decir, si la condición se cumple es verdadero sino es falso, así se maneja dentro de estas estructuras, pues la respuesta ante la condición es que si está se sigue cumpliendo el programa se repetirá hasta que sea falso. Dentro de estas estructuras se tienen las siguientes.

- **WHILE**: Esta estructura crea un bucle que ejecuta una sentencia especificada mientras la condición se evalúe como verdadera.

```
<script type="text/javascript">
while(condicion){ //si esto se cumple
    sentencia //haz esto
}
</script>
```

Ilustración 579 - Imagen propia

Para hacer gráfica y más explícito el concepto, se colocará una variable que contenga un número y a ese, se le aumentará un 1, dentro de la condición se declarará que la sentencia se va a repetir y va a estar sumando hasta que el número llegue a ser igual o menor que 10.

```
<script type="text/javascript">

var numeroInicial = 0;

while(numeroInicial < 10){
  numeroInicial = numeroInicial + 1;
  console.log(numeroInicial);
}

</script>
```

Ilustración 580 - Imagen propia



Ilustración 581 - Imagen propia

En esta estructura, también se complementa con una segunda palabra reservada, **DO - WHILE** hace la misma función que **WHILE** la única diferencia es que la sentencia o repetición que genera se coloca dentro de **DO** y después **WHILE** contendrá la condición para ejecutarse mientras sea verdadero, es esta manera se asegura el desarrollador que el bucle se cumpla al menos una vez, pues de no ser así, si la condición fuese falsa, **WHILE** por sí solo no leería la sentencia.

```
<script type="text/javascript">

var numeroInicial = 0;

do{
  numeroInicial = numeroInicial + 1;
  console.log(numeroInicial);
}while(numeroInicial < 10)

</script>
```

Ilustración 582 - Imagen propia

- **FOR**: En esta estructura su función es similar a la que desempeña **DO-WHILE**, solo que **FOR** permite tener más control sobre el mismo bucle, pues las condiciones que permite insertar dentro de su estructura, es más controlado.

```

<script type="text/javascript">
for (inicializacion; condicion; incremento) {
    sentencias
}
</script>

```

Ilustración 583 - Imagen propia

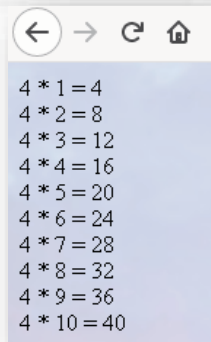
Un ejemplo fácil, puede lograrse una tabla de multiplicar.

```

<script type="text/javascript">
for (var numMultiplicado = 1; numMultiplicado <= 10; numMultiplicado++) {
    document.write( 4+" "+"*"+numMultiplicado+" "+"="+" "+numMultiplicado*4+"<br>");
}
</script>

```

Ilustración 584 - Imagen propia



A screenshot of a browser window displaying a multiplication table. The table consists of ten rows, each representing a multiplication of 4 by a number from 1 to 10. The browser's navigation bar is visible at the top, showing back, forward, refresh, and home icons. The text in the browser window is as follows:

```

4 * 1 = 4
4 * 2 = 8
4 * 3 = 12
4 * 4 = 16
4 * 5 = 20
4 * 6 = 24
4 * 7 = 28
4 * 8 = 32
4 * 9 = 36
4 * 10 = 40

```

Ilustración 585 - Imagen propia

Como se puede observar, dentro de **FOR** se creó una variable que se inicializó con un valor que es “1”, a su vez se le asigna que será menor o igual a “10” porque su siguiente tarea es aumentar uno. Por lo tanto, cada vez que el navegador lea ese **FOR** lo hará 10 veces para cumplir la condicional que tiene.

Cabe resaltar que el uso de condicionales y bucles es importante dentro de la programación en **JS**, pues no solo se utilizan para hacer operaciones aritméticas ni que devuelvan falso o verdadero, estas estructuras también son benéficas en la implementación de tareas más complicadas y detalladas de un programa. Por lo tanto, no es buena idea otorgarles poca atención.

9.5. DOM

Dentro de **JS** también se encuentra un nuevo pilar, que en conjunto le da vida al propio **JS**, los temas anteriores no son más que el conocimiento básico y esencial para comprender **JS**, pues si se puede dar un justo concepto o descripción de lo que es, por sí solo es un simple lenguaje de programación, pues no existe gran interactividad con un sitio web, bueno, pues para lograr eso se encuentra **DOM**.

La creación del Document Object Model (Modelo de Objeto de Documento) o **DOM** es una de las innovaciones que más ha influido en el desarrollo de las páginas web dinámicas y de las aplicaciones web más complejas. **DOM** permite a los programadores web acceder y manipular las páginas **HTML5**. Uniwebsidad (2018)

A pesar de sus orígenes, **DOM** se ha convertido en una utilidad disponible para la mayoría de lenguajes de programación (**Java**, **PHP**, **JavaScript**) y cuyas únicas diferencias se encuentran en la forma de implementarlo. Una de las tareas habituales en la programación de aplicaciones web con **JAVASCRIPT** consiste en la manipulación de las páginas web. De esta forma, es habitual obtener el valor almacenado por algunos elementos, por ejemplo: los elementos de un formulario, crear un elemento **<P>**, **<DIV>**, etc. de forma dinámica y añadirlo a la página, aplicar una animación a un elemento.

Todas estas tareas habituales son muy sencillas de realizar gracias a **DOM**. Sin embargo, para utilizar las utilidades de **DOM**, es necesario "TRANSFORMAR" la página original. Una página **HTML5** normal no es más que una sucesión de caracteres, por lo que es un formato muy difícil de manipular. Por ello, los navegadores web transforman automáticamente todas las páginas web en una estructura más eficiente de manipular.

Esta transformación la realizan todos los navegadores de forma automática y permite utilizar las herramientas de **DOM** de forma muy sencilla. **DOM** transforma todos los documentos **HTML5** en un conjunto de elementos llamados nodos, que están interconectados y que representan los contenidos de las páginas web y las relaciones entre ellos. Por su aspecto, la unión de todos los nodos se llama "árbol de nodos".

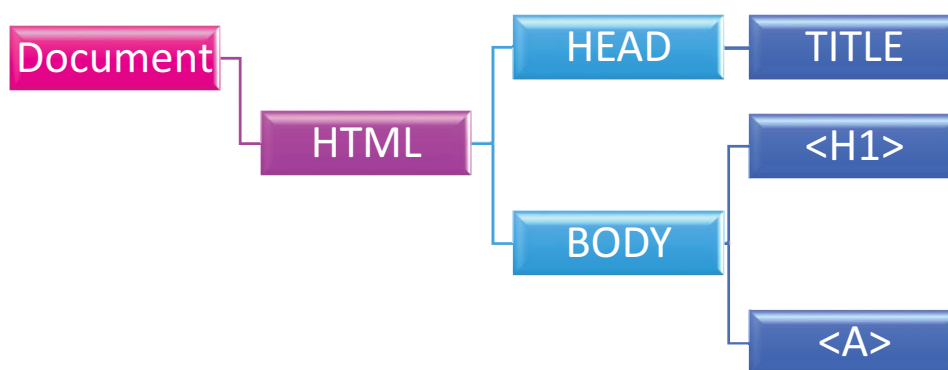


Ilustración 586 - Imagen propia

Con el modelo de objetos, JavaScript recibe toda la energía que necesita para crear **HTML5** dinámico:

- **JAVASCRIPT** puede cambiar todos los elementos **HTML5** en la página
- **JAVASCRIPT** puede cambiar todos los atributos de **HTML5** en la página
- **JAVASCRIPT** puede cambiar todos los estilos **CSS** en la página
- **JAVASCRIPT** puede eliminar elementos y atributos **HTML5** existente
- **JAVASCRIPT** puede añadir nuevos elementos y atributos **HTML5**
- **JAVASCRIPT** puede reaccionar a todos los eventos de **HTML5** existentes en la página
- **JAVASCRIPT** puede crear nuevos eventos en la página **HTML5**

El **DOM HTML5** es un estándar de objetos de modelo y de programación de interfaces para **HTML5**. Para poder tener un acceso real a los elementos de una página web por medio del **DOM**, se hace por medio de:

- Los elementos **HTML5** como objetos
- Las propiedades de todos los elementos **HTML5**
- Los métodos para acceder a todos los elementos **HTML5**
- Los eventos de todos los elementos **HTML5**

En otras palabras: El **DOM HTML5** es un estándar de cómo obtener, cambiar, añadir o eliminar elementos **HTML5**.

```
<p id="cambio">Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod
tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam,
quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo
consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse
cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non
proident, sunt in culpa qui officia deserunt mollit anim id est laborum.</p>

<script type="text/javascript">

    document.getElementById("cambio").innerHTML = "Nuevo texto";

</script>
```

Ilustración 587 - Imagen propia

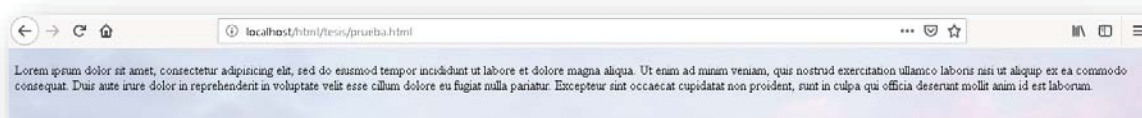


Ilustración 588 - Imagen propia

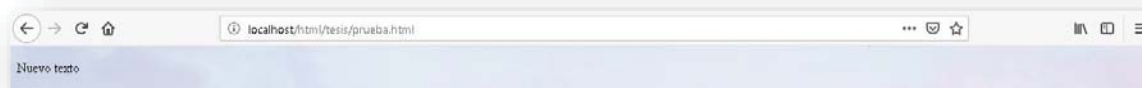


Ilustración 589 - Imagen propia

El cambio de texto que se realizó en la etiqueta **<P>** se hizo por medio del **ID** que contiene la etiqueta, de esa manera con **JS** el método **“getElementById”** llamo al nombre del **ID** que contenía **<P>** y al anclar a la propiedad **“innerHTML”** le redireccionó o redefinió el texto

que tendría la etiqueta, esta misma propiedad tiene solo dos funciones: conseguir el contenido de dicho **ID** o reemplazar el contenido. De esa manera el **DOM** funge como intermediario para que exista una interactividad entre cliente y página web, porque puede actuar en tiempo real, ocasionando cambios dentro de una página web mientras se lo permita al cliente.

Ubicación de elementos en HTML	
Método	Descripción
document.getElementById(id)	Encuentra al elemento por medio del "ID"
document.getElementsByTagName(name)	Encuentra al elemento por medio del nombre de la etiqueta
document.getElementsByClassName(name)	Encuentra al elemento por medio de la "CLASS"

Ilustración 590 - Tabla propia

Elementos que generan cambios	
Propiedades	Descripción
element.innerHTML = new html content	Cambia el contenido interno del elemento
element.attribute = new value	Cambia el valor del atributo del elemento
element.style.property = new style	Cambia el estilo del elemento
Método	Descripción
element.setAttribute(attribute, value)	Cambia el atributo de un elemento

Ilustración 591 - Tabla propia

Agregar o eliminar elementos	
Métodos	Descripción
document.createElement(element)	Crea un nuevo elemento de HTML
document.removeChild(element)	Remueve o elimina un elemento de HTML
document.appendChild(element)	Agrega un elemento HTML
document.replaceChild(new, old)	Reemplaza un elemento de HTML
document.write(text)	Agrega un texto dentro de la página web

Ilustración 592 - Tabla propia

Pero cuando se trata de la intervención de funciones para el dinamismo de una página web, se encuentran los eventos, que generan una acción automática ya programada, donde su única función es desempeñar animaciones dentro del sitio web.

Un dato importante es que existen dos modos de modificar un **HTML5** y que es lo más recomendable, pues las clases dentro de una etiqueta **HTML5** existe para que **CSS** trabaje sobre ella mientras que los identificadores existen para que permitan que **JS** actúe sobre ellos. Por supuesto que dentro de los componentes del **DOM**, así como en **HTML5** y **CSS** tiene un sinfín de elementos, propiedades y eventos que permiten tener un cambio aún más dinámico y profesional en un sitio web, y como en todo, con la práctica se comienza a familiarizar el uso y conocimiento de los nuevos y la interacción y combinación de todo lo que se conjuga para complementar **HTML5**, **CSS** Y **JS**.

10. FRAMEWORKS

Como se ha ido conociendo y aprendiendo a lo largo de este trabajo, el Desarrollo Web tuvo que ir evolucionando a medida que se creaban nuevas funcionalidades tecnológicas, estas cada vez eran más complejas porque su objetivo era satisfacer las necesidades del público y aumentar al mismo, a su vez la internet también solicitaba cierta demanda pues comenzaba a crecer el mundo tecnológico y el internet comenzó a ser indispensable en los hogares, organismos empresariales y consumidores en general, la sociedad comenzó a crecer a tal magnitud y al mismo tiempo que se comenzó a utilizar la web más que nunca, orillando así que el Desarrollo Web se volviera dinámico, llamativo, innovador, profesional y sobre todo un medio de comercio, por ende, **HTML5**, **CSS** y **JS** ya no podían solventar tanta demanda. Pues, aunque estos componentes del diseño web evolucionaran e hicieran mejoras, no era suficiente.

Hoy en día, existen muchos sitios que ofrecen crear páginas web de manera más fácil, más entretenida, sin necesidad de estar tecleando código, solo se le permite al creador moldear su página web a su gusto, por ello comenzaron a desarrollarse nuevas tecnologías, que permitieran tener un mejor desempeño, ahorrando tiempo y obteniendo mejores resultados. Una vez llegados a este punto, miles de programadores en todo el mundo comenzaron a hacer sus propias aplicaciones web para dar respuesta a la creciente demanda en complejidad de los sitios webs. Es así como comenzaron a surgir los Frameworks.

Dentro del concepto contundente de los Frameworks, es un software (entorno), aplicación, marco de trabajo o herramienta que brinda al programador o desarrollador ayuda para realizar aplicaciones profesionales, estables y dinámicas mediante un conjunto de paquetes (librerías **JS**), herramientas y utilidades que agilizan el proceso de desarrollo de un proyecto por lo que permiten realizar su trabajo en menor tiempo, pero con una complejidad y modernidad funcional de lo que esté desarrollando. (Desarrolloweb, 2018-2019)

Cabe destacar que, en el tema de los Frameworks, existen muchísimos y estos se dividen en muchos carriles, pues esta herramienta es pensada para todas las personas que desarrollan o programan tecnológicamente hablando, por ello, hay Frameworks para Back-End (Programación) como Front-End (Desarrollo y Diseño Web), es estos dos tipos se comienzan a derivar muchas ramitas en donde también se ocupan las nuevas herramientas, pues en este momento ya no se trata de crear algo cuando se pueda, sino que se trata de reinventar la web de manera más productiva, más eficiente y más rápida.

- **Frameworks para Desarrolladores de Front-End:** Son los Frameworks que facilitan el trabajo para las vistas o páginas que serán vistas por el usuario final o el público, este tipo de Frameworks se componen las tecnologías **JS**, **HTML5** y **CSS**.

Para esta categoría se pueden mencionar Frameworks como **Angular JS**, **React JS**, **Vue JS**, **Bootstrap**, **Material Design** entre otros.

- **Frameworks para Desarrolladores de Back-End:** Este tipo de Frameworks facilitan el trabajo con Lenguajes de Programación de lado del Servidor como

Python, Ruby, PHP, JavaScript del lado del servidor (**Node JS**), **Java**, etc. Ayudan a gestionar las Bases de Datos, el envío y procesamiento de datos, Cookies, Sesiones, etc.

Para esta categoría se pueden mencionar Frameworks como **Django, Ruby on Rails, Node JS, Laravel, Spring, React Native JS**, etc.

10.1. BOOTSTRAP

Bootstrap, originalmente llamado Blueprint de Twitter, fue desarrollado por Mark Otto y Jacob Thornton de Twitter, como un marco de trabajo (framework) para fomentar la consistencia entre las herramientas internas. Antes de **Bootstrap**, se usaron varias bibliotecas para el desarrollo de interfaces de usuario, las cuales generó inconsistencias y una gran carga de trabajo en su mantenimiento.

Según el desarrollador Mark Otto (2011, Twitter, Bootstrap, <https://twitter.com/mdo>) menciona:

"...un súper pequeño grupo de desarrolladores y yo nos reunimos para diseñar y construir una nueva herramienta interna y vimos la oportunidad de hacer algo más. A través de ese proceso, nos vimos construyendo algo mucho más sustancial que otra herramienta interna. Meses después terminamos con una primera versión de Bootstrap como una manera de documentar y compartir bienes y patrones de diseño comunes dentro de la compañía."

El primer desarrollo en condiciones reales ocurrió durante la primera "Semana de Hacking" (Hackweek) de Twitter." Mark Otto mostró a algunos colegas como acelerar el desarrollo de sus proyectos con la ayuda de la herramienta de trabajo. Como resultado, decenas de temas se han introducido en el marco de trabajo. En agosto del 2011, Twitter liberó a **Bootstrap** como código abierto. En febrero del 2012, se convirtió en el proyecto de desarrollo más popular de GitHub.

Para ser precisos con el concepto, **Bootstrap** es un marco de trabajo donde se incorporan muchas herramientas de código por parte de **CSS** y **JS** para diseño de sitios y aplicaciones web móviles. Contiene plantillas de diseño con tipografía, formularios, botones, cuadros, menús de navegación y otros elementos de diseño basado en **HTML5** y **CSS**, así como extensiones funcionales y dinámicas de JavaScript adicionales. A diferencia de muchos Frameworks web, **Bootstrap** solo se ocupa del desarrollo Front-End.

10.2. INCORPORACIÓN DE BOOTSTRAP EN DISEÑO WEB

Cuando se comienza la implementación de **Bootstrap** a **HTML5**, es muy importante comprender que este Framework está basado justamente en **HTML5** y **CSS**, otras funciones y animaciones se encuentran en **JS** y **JQuery** (Este último es una librería de **JS** en combinación con **CSS**), por lo tanto, cuando se agregue en el proyecto, no se instalara un programa o un software, nada de eso, solo serán anexados los enlaces de archivos externos

de **CSS** y **JS**, para poder acceder a todo su paquete de funciones se tiene que ingresar a su página oficial.

<https://getbootstrap.com>

Ilustración 593 - Imagen propia

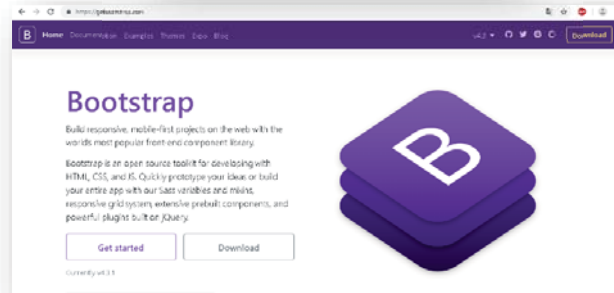


Ilustración 594 - Imagen propia

Como se puede mostrar en la imagen, al ingresar a su página oficial, presenta dos opciones de cómo integrar al Framework, la primera es descargarlo y tener los archivos de raíz, es decir, tener el archivo **CSS** y **JS** literalmente en las carpetas del proyecto, solo basta con descargar el paquete de archivos:

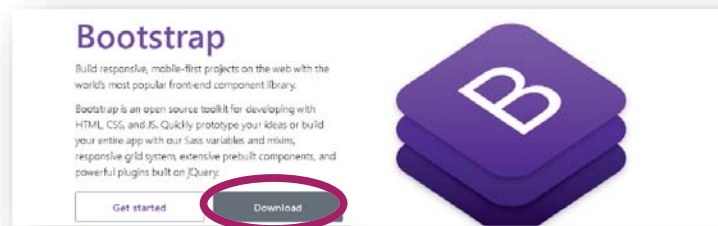


Ilustración 595 - Imagen propia



Ilustración 596 - Imagen propia

Cuando termine la descarga, se obtendrá un archivo **.zip**, donde se encontrarán comprimidas las carpetas **CSS** y **JS** que serán utilizadas dentro de **HTML5**, esos archivos ya tienen cargadas y registradas todas las herramientas, funciones y librerías para la función óptima y fiel de **Bootstrap**.



Ilustración 597 - Imagen propia



Nombre	Tamaño	Comprimido	Tipo	Modificado
..			Disco local	
css			Carpeta de archivos	13/02/2019 08:...
js			Carpeta de archivos	13/02/2019 08:...

Ilustración 598 - Imagen propia

Lo que se hará con estas carpetas, es descomprimirlas y exportarlas en la respectiva carpeta **CSS** y **JS** que se tiene ya creada para el proyecto, es decir, cuando se tiene un proyecto nuevo, se ordena por carpetas y el único archivo que se mantiene afuera es **HTML5**.

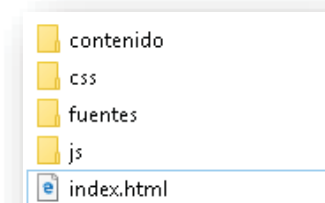


Ilustración 599 - Imagen propia

Ahora bien, para entender mejor y más ilustrado el ejemplo, cuando se abra el archivo.zip, el contenido que tenga la carpeta **CSS** se exportara a la carpeta del proyecto, si se puede observar mantienen el mismo nombre ambas carpetas, por tanto, no hay problema de equivocarse.

Nombre	Tamaño	Comprimido	Tipo	Modificado	Hash
..			Disco local		
bootstrap.css	192,348	25,178	Archivo CSS	13/02/2019 08:...	CCC143DD
bootstrap.css.m...	492,048	97,004	Archivo MAP	13/02/2019 08:...	28F28F08
bootstrap.min.css	155,758	23,054	Archivo CSS	13/02/2019 08:...	D315FFD1
bootstrap.min.c...	625,953	100,451	Archivo MAP	13/02/2019 08:...	D874D374
bootstrap-grid.css	64,548	6,855	Archivo CSS	13/02/2019 08:...	732A95EA
bootstrap-grid.c...	151,749	26,459	Archivo MAP	13/02/2019 08:...	F809828E
bootstrap-grid....	48,488	6,014	Archivo CSS	13/02/2019 08:...	BF571255
bootstrap-grid....	108,539	13,630	Archivo MAP	13/02/2019 08:...	50D36941
bootstrap-reboo...	4,897	1,699	Archivo CSS	13/02/2019 08:...	716E325F
bootstrap-reboo...	76,483	16,773	Archivo MAP	13/02/2019 08:...	1152A1F6
bootstrap-reboo...	4,021	1,591	Archivo CSS	13/02/2019 08:...	634CC80D
bootstrap-reboo...	32,461	7,979	Archivo MAP	13/02/2019 08:...	0A2BA8CF

Ilustración 600 - Imagen propia



Ilustración 601 - Imagen propia

Pero eso no es todo, el siguiente paso es anclar los archivos dentro del **HTML5** de la manera que se mencionó en los temas anteriores, **CSS** por medio de **<LINK>** y **JS** por **<SCRIPT>**.

Sin duda alguna es el camino más largo para tener acceso a **Bootstrap**, por suerte, las mejoras que se han realizado en los últimos años de este Framework, es que muestra una nueva forma de trabajar con él, pues no tiene mucho que **Bootstrap** se unió a MaxCDN para ser

patrocinado, esta empresa se dedica básicamente a incorporar de manera más rápida y práctica los desarrollos de aplicaciones para nuevas tecnologías.

CDN es la abreviatura de Content Delivery Network (Red de Entrega de Contenido), es un sistema de servidores distribuidos en la red que entrega páginas y otro contenido web a un usuario, según las ubicaciones geográficas del usuario, el origen de la página web y el servidor de entrega de contenido. Este servicio es eficaz para acelerar la entrega de un contenido de sitios web con alto tráfico de alcance global. En pocas palabras, se carga el contenido de Bootstrap en un servidor, cuando se ejecuta el sitio web el programa se cargará desde la memoria caché por lo que lleva a un tiempo de carga más rápida. Además, la mayoría de los CDN se asegurarán de que, una vez que el usuario solicite un archivo, se los envíe desde el servidor más cercano a ellos, lo que también lleva a un tiempo de carga más rápido.

Por ello, también se puede acceder a **Bootstrap** de la siguiente manera:



Ilustración 602 - Imagen propia

Basta con copiar y pegar esas líneas de código en el archivo **HTML5** y así de fácil las herramientas, librerías y funciones ya están cargadas en el documento **HTML5**.

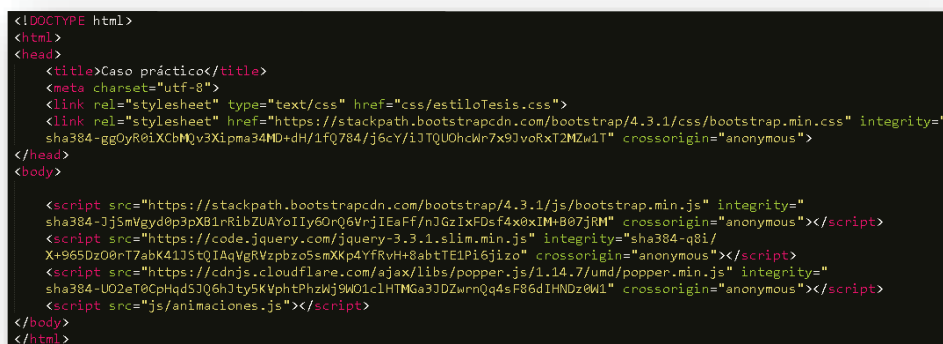


Ilustración 603 - Imagen propia

10.3. DISEÑO DE PANTALLA

Bootstrap 4 está diseñado para responder a dispositivos móviles. Para garantizar una representación adecuada y un zoom táctil, se agrega la siguiente **<META>** dentro de **<HEAD>**:

```
<head>
  <title>Caso práctico</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1 shrink-to-fit=no">
</head>
```

Ilustración 604 - Imagen propia

Cuando se habla de diseño de maquetación responsiva, **VIEWPORT** funge como contenedor adaptativo de la página web o contenido de HTML, por la naturaleza y objetivo de **VIEWPORT** permite adaptar la página en el navegador web que se esté visualizando, ya sea en dispositivos móviles o computadoras. También se cuenta con el atributo **CONTENT** donde explícitamente dice en su nombre que contendrá elementos responsables de que la página web se adapte a la pantalla que la ejecute, **WIDTH=DEVICE-WIDTH** que establece el ancho de la página con respecto al ancho de pantalla del dispositivo donde se visualice la página, por otro lado **INITIAL-SCALE=1** establece el nivel de zoom inicial cuando la página se carga por primera vez con el navegador y por último **SHRINK-TO-FIT=NO** establece que si la página no está responsiva, entonces que no se vuelva pequeña dentro del navegador.

Ahora bien, **Bootstrap** trabaja con **<DIV>** como ya se sabe, por medio de estos se generan las cajas, y **Bootstrap** lo que hace es incluirse dentro del código por medio de clases, en este caso va a existir un **<DIV>** padre, que contendrá a todo el contenido de la página, por darle un nombre sería el **BODY(Cuerpo)** de **Bootstrap**, y su principal clase sería **CONTAINER** quien brinda un margen sensible dentro del navegador y facilita el centrado de la página, por otro lado se tiene a **CONTAINER-FLUID** quien hace todo lo contrario, pues abarca completamente el área.

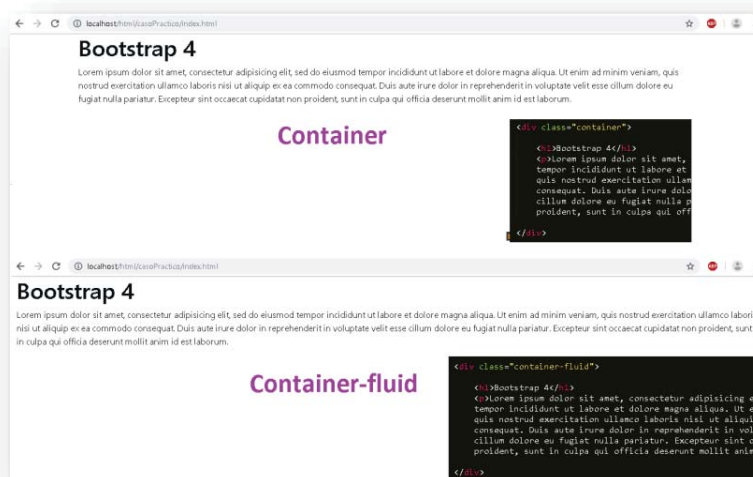


Ilustración 605 - Imagen propia

Partiendo de ese punto, ya se obtuvo una página responsiva, declarada desde el inicio y su contenido mantendrá o no un margen y una alineación, por lo tanto, el siguiente tema importante es la distribución del área de trabajo, sí, así como lo estás leyendo, **Bootstrap** trabaja el área completa de una página web dividida, es decir, para evitar convertir a los elementos en **BLOCK** o **IN-LINE** y poniendo posiciones **RELATIVE**, **ABSOLUTE** para acomodar a los elementos de manera ordenada dentro de la página.

Bootstrap trabaja en una rejilla que divide a la página y permite una mejor distribución de los elementos de **HTML5**, así como se mostró en **CSS** el **GRID**, pero en esta ocasión, es mucho más fácil de declarar y sin tanto código y cambio dentro de **CSS**, el sistema de cuadrícula de **Bootstrap** utiliza una serie de contenedores, filas y columnas para diseñar y alinear el contenido. Está construido con **FLEXBOX**, este es un elemento establecido de **CSS** como campo Layout, es la referencia a diseñar mejores estilos y aprovechamiento de espacio en los elementos y la página, por lo que es totalmente sensible a adaptarse en cualquier tipo de pantalla. Permite dividir a la página hasta 12 columnas, sino se requiere utilizar las 12 columnas individualmente, puede agrupar las columnas para crear columnas más anchas.

Este objetivo se logra declarando un **<DIV>** con clase **ROW** que funge como fila de la rejilla y los siguientes **<DIV>** que se encuentren dentro tendrán la clase **COL** estos declaran las columnas de la fila.

```
<div class="row">
  <div class="col" style="background-color: purple;">Una</div>
  <div class="col" style="background-color: cyan;">Dos</div>
  <div class="col" style="background-color: lightpink;">Tres</div>
</div>
```

Ilustración 606 - Imagen propia

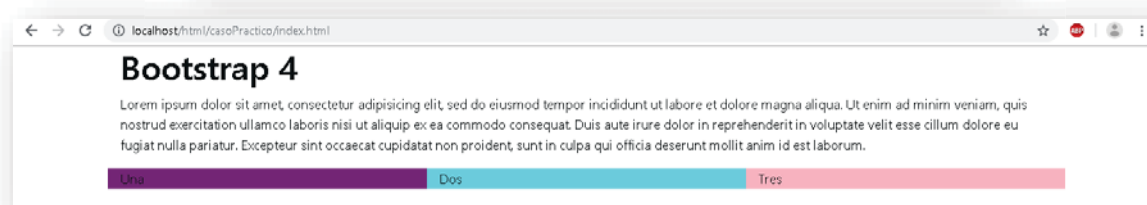


Ilustración 607 - Imagen propia

Como se muestra en el ejemplo anterior, un **<DIV>** se convirtió en una rejilla responsiva, lo cual permite ajustar a los elementos para que luzcan bien, de hecho, no se corre peligro que se deforme el contenido que tiene dentro de cualquier celda, pues esa es su mayor ventaja, **Bootstrap** siempre buscara resolver el ajuste y acomodo de los elementos de manera eficiente y rápida sin que el diseñador tenga que estar cambiando parámetros y porcentajes para que se vuelva a ajustar, también se observa dentro de la imagen que la creación de las columnas **Bootstrap** las adapta de igual anchura, por lo que con este Framework se cuenta con adaptación de tamaños y numero de rejillas, el sistema de rejilla de **Bootstrap** tiene cinco clases:

- **COL-** (Dispositivos extra pequeños - ancho de pantalla inferior a 576px)
- **COL-SM-** (dispositivos pequeños - ancho de pantalla igual o mayor que 576px)

- **COL-MD-** (dispositivos medianos - ancho de pantalla igual o mayor a 768px)
- **COL-LG-** (dispositivos grandes: ancho de pantalla igual o mayor que 992px)
- **COL-XL-** (dispositivos xlarge - ancho de pantalla igual o mayor que 1200px)

Estas mismas clases se pueden combinar para crear diseños más dinámicos y flexibles, por lo tanto, lo que más conviene, es que se le permita a **Bootstrap** generar los propios espacios y anchos de las rejillas, aunque si se quiere manejar con exactitud las divisiones, las columnas cuentan con otro parámetro, pues después de declarar el ancho que se requiere dentro de las columnas, es importante determinar que contenedor tendrá la página, pues de esto depende que los 12 espacios se distribuyan por el 100% del área que se determine.

En otras palabras, una rejilla está dividida en 12 espacios por fila, no más no menos:



Ilustración 608 - Imagen propia

Ahora cuando se declaran las columnas, estas pueden definirse del ancho que sea, pero al momento de declarar el tercer valor, se tiene que tomar en cuenta que los 12 espacios se deben de distribuir entre el número de columnas, por ejemplo, la distribución equitativa de las columnas se puede obtener las siguientes rejillas:

```
<!--Doce columnas-->
<div class="row">
  <div class="col" style="background-color: violet;">Uno</div>
  <div class="col" style="background-color: cyan;">Dos</div>
  <div class="col" style="background-color: lightpink;">Tres</div>
  <div class="col" style="background-color: violet;">Cuatro</div>
  <div class="col" style="background-color: cyan;">Cinco</div>
  <div class="col" style="background-color: lightpink;">Seis</div>
  <div class="col" style="background-color: violet;">Siete</div>
  <div class="col" style="background-color: cyan;">Ocho</div>
  <div class="col" style="background-color: lightpink;">Nueve</div>
  <div class="col" style="background-color: violet;">Diez</div>
  <div class="col" style="background-color: cyan;">Once</div>
  <div class="col" style="background-color: lightpink;">Doce</div>
</div>
```

Ilustración 609 - Imagen propia



Ilustración 610 - Imagen propia

```
<!--Seis columnas con 2 espacios-->
<div class="row">
  <div class="col-sm-2" style="background-color: violet;">Uno</div>
  <div class="col-sm-2" style="background-color: cyan;">Dos</div>
  <div class="col-sm-2" style="background-color: lightpink;">Tres</div>
  <div class="col-sm-2" style="background-color: violet;">Cuatro</div>
  <div class="col-sm-2" style="background-color: cyan;">Cinco</div>
  <div class="col-sm-2" style="background-color: lightpink;">Seis</div>
</div>
```

Ilustración 611 - Imagen propia



Ilustración 612 - Imagen propia

```
<!--Cuatro columnas con 3 espacios-->
<div class="row">
  <div class="col-sm-3" style="background-color: violet;">Uno</div>
  <div class="col-sm-3" style="background-color: cyan;">Dos</div>
  <div class="col-sm-3" style="background-color: lightpink;">Tres</div>
  <div class="col-sm-3" style="background-color: violet;">Cuatro</div>
</div>
```

Ilustración 613 - Imagen propia



Ilustración 614 - Imagen propia

```
<!--Dos columnas con 6 espacios-->
<div class="row">
  <div class="col-sm-6" style="background-color: violet;">Uno</div>
  <div class="col-sm-6" style="background-color: cyan;">Dos</div>
</div>
```

Ilustración 615 - Imagen propia



Ilustración 616 - Imagen propia

```
<!--Una columna con espacio completo-->
<div class="row">
  <div class="col" style="background-color: violet;">Uno</div>
</div>
```

Ilustración 617 - Imagen propia

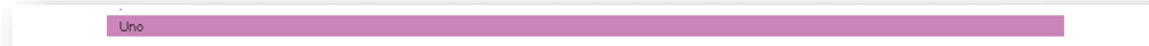


Ilustración 618 - Imagen propia

Por supuesto, que también se puede jugar con los espacios y generar una rejilla desigual de espacios, de esta manera se puede lograr una página web bien distribuida.

Si bien, se puede desde este punto adelantarse a las conclusiones, los ejemplos anteriores en comparación al tema y ejemplos de **CSS**, **Bootstrap** comienza a facilitar de un modo muy sutil la vida del diseñador, aunque **CSS** es muy importante e indispensable dentro del diseño, sin duda alguna la ayuda de una herramienta como **Bootstrap** es muy bien aceptada.



Ilustración 619 - Imagen propia

La siguiente atribución que tiene **Bootstrap** es con la tipografía, pues, aunque se vean que son elementos de **HTML5** como etiquetas y clases, todo está influenciado por **Bootstrap**, pues todo lo que este dentro de **HTML5** será sensible a la adaptación de dispositivos móviles, no se olvide que este Framework fue creado precisamente para móviles y tabletas. En los siguientes ejemplos se mostrará las funciones que desencadenan las siguientes etiquetas y clases.



Ilustración 620 - Imagen propia

Ilustración 621 - Imagen propia

<pre><h1 class="display-1">Hola</h1> <h1 class="display-2">Hola</h1> <h1 class="display-3">Hola</h1> <h1 class="display-4">Hola</h1></pre> <p><i>Ilustración 622 - Imagen propia</i></p>	<p>Se presenta una nueva forma de magnificar la fuente y ligereza de un encabezado</p>	 <p><i>Ilustración 623 - Imagen propia</i></p>
<pre><h1>Temas <small>texto anexo</small></h1> <h2>Temas <small>texto anexo</small></h2> <h3>Temas <small>texto anexo</small></h3> <h4>Temas <small>texto anexo</small></h4> <h5>Temas <small>texto anexo</small></h5> <h6>Temas <small>texto anexo</small></h6></pre> <p><i>Ilustración 624 - Imagen propia</i></p>	<p>Dentro de los encabezados es posible añadir un texto más pequeño extra al tema</p>	 <p><i>Ilustración 625 - Imagen propia</i></p>
<pre><p>Este texto es <mark>importante</mark></p></pre> <p><i>Ilustración 626 - Imagen propia</i></p>	<p>Se puede subrayar o marcar texto importante</p>	 <p><i>Ilustración 627 - Imagen propia</i></p>
<pre><p><abbr title="Responsive Web Desing">RWD</abbr></p></pre> <p><i>Ilustración 628 - Imagen propia</i></p>	<p>Funciona como mensaje emergente del significado de las siglas</p>	 <p><i>Ilustración 629 - Imagen propia</i></p>
<pre><dl> <dt>CSS</dt> <dd>-Diseño Web</dd> <dt>Bootstrap</dt> <dd>-Framework de RWD</dd> </dl></pre> <p><i>Ilustración 630 - Imagen propia</i></p>	<p>Una nueva manera de detallar una lista</p>	 <p><i>Ilustración 631 - Imagen propia</i></p>
<pre><p> Elementos HTML <code>head</code>, <code>body</code>,<code>input</code> </p></pre> <p><i>Ilustración 632 - Imagen propia</i></p>	<p>Permite resaltar un código dentro de texto normal</p>	 <p><i>Ilustración 633 - Imagen propia</i></p>

<pre><p>Teclea <kbd>Ctrl + P</kbd> para imprimir</p></pre> <p><i>Ilustración 634 - Imagen propia</i></p>	<p>Hace resaltar con estilo de manera increíble un texto en específico</p>	<p>Teclea Ctrl + P para imprimir</p> <p><i>Ilustración 635 - Imagen propia</i></p>
<pre><p class="font-weight-bold">Ejemplo de cambio</p></pre> <p><i>Ilustración 636 - Imagen propia</i></p>	<p>Convierte el texto en negritas</p>	<p>Ejemplo de cambio</p> <p><i>Ilustración 637 - Imagen propia</i></p>

Existen otras clases dentro de la tipografía de **Bootstrap** que favorecen un cambio dentro del texto, por supuesto que existen muchas, pero dentro de las más conocidas y usadas, se encuentran en la tabla anterior, otras comienzan con **TEXT-** y el siguiente valor, varía de entre texto centrado, alineado o justificado, negritas, cursivas o subrayadas, decoración, tamaño, etc., todo correspondiente al idioma Inglés, pues son muy parecidos los cambios que existen entre **HTML5** y **CSS**, sin embargo, **Bootstrap** busca mejorarlos de alguna manera.

También se puede contar dentro de **Bootstrap** colores por default, estos colores existen para una maquetación simple y rápida, para favorecer una rápida entrega de un proyecto, claro está, que si se quiere mejorar algún color se puede utilizar **CSS**, solo basta con llamar a la clase dentro de **CSS** y poder mejorar el color, sin embargo, los colores que proporciona el Framework no son tan malos cuando se trata de resaltar algún elemento.

```
<div class="row">
  <div class="col"><button class="bg-primary">Color primary</button></div>
  <div class="col"><button class="bg-success">Color success</button></div>
  <div class="col"><button class="bg-info">Color info</button></div>
  <div class="col"><button class="bg-warning">Color warning</button></div>
  <div class="col"><button class="bg-danger">Color danger</button></div>
  <div class="col"><button class="bg-secondary">Color secondary</button></div>
  <div class="col"><button class="bg-dark">Color dark</button></div>
  <div class="col"><button class="bg-light">Color light</button></div>
</div>
```

Ilustración 638 - Imagen propia

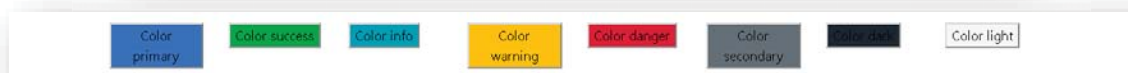


Ilustración 639 - Imagen propia

Después de todo lo que ha cambiado **HTML5** por medio de **Bootstrap**, con tan poco trabajo y en un tiempo record, sin duda poder trabajar una tabla no será tan difícil, para eso, se puede declarar una tabla como se ha hecho en los temas de **HTML5** y **CSS**, solo se anexa una clase **TABLE** en la etiqueta **<TABLE>**.


```

<table class="table">
  <thead>
    <tr>
      <th>Nombre</th>
      <th>Apellido</th>
      <th>No. Cuenta</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>Abril</td>
      <td>Alba</td>
      <td>646388569</td>
    </tr>
    <tr>
      <td>Luz</td>
      <td>Villa</td>
      <td>392533955</td>
    </tr>
    <tr>
      <td>Jose Luis</td>
      <td>Garduño</td>
      <td>858435678</td>
    </tr>
  </tbody>
</table>

```

Ilustración 640 - Imagen propia

Si se presta atención en la imagen anterior, se anexan dos etiquetas más, pues como su nombre lo menciona explícitamente, se detalla aún más la cabecera **<THEAD>** y cuerpo **<TBODY>** de la tabla, con solo eso se obtiene algo así:

Nombre	Apellido	No. Cuenta
Abril	Alba	646388569
Luz	Villa	392533955
Jose Luis	Garduño	858435678

Ilustración 641 - Imagen propia

Y bueno, si se puede hacer una comparación con la incrustación de una tabla en **HTML5** o **CSS** y la tabla de la imagen anterior, por mucho evita tanto trabajo para poder lograr ese diseño así sin más, en definitiva, se puede mejorar aún más esta tabla, con las siguientes clases.

- Tabla estilo cebra clara:

```

<table class="table table-striped">

```

Ilustración 642 - Imagen propia

Nombre	Apellido	No. Cuenta
Abril	Alba	646388569
Luz	Villa	392533955
Jose Luis	Garduño	858435678

Ilustración 643 - Imagen propia

- Tabla estilo bordeada:

```
<table class="table table-bordered">
```

Ilustración 644 - Imagen propia

Nombre	Apellido	No. Cuenta
Abril	Alba	646388569
Luz	Villa	392533955
Jose Luis	Garduño	858435678

Ilustración 645 - Imagen propia

- Tabla estilo default con efecto hover:

```
<table class="table table-hover">
```

Ilustración 646 - Imagen propia

Nombre	Apellido	No. Cuenta
Abril	Alba	646388569
Luz	Villa	392533955
Jose Luis	Garduño	858435678

Ilustración 647 - Imagen propia

- Tabla con estilo oscuro:

```
<table class="table table-dark">
```

Ilustración 648 - Imagen propia

Nombre	Apellido	No. Cuenta
Abril	Alba	646388569
Luz	Villa	392533955
Jose Luis	Garduño	858435678

Ilustración 649 - Imagen propia

- Tabla con estilos combinados entre oscura y cebra:

```
<table class="table table-dark table-striped">
```

Ilustración 650 - Imagen propia

Nombre	Apellido	No. Cuenta
Abril	Alba	646388569
Luz	Villa	392533955
Jose Luis	Garduño	858435678

Ilustración 651 - Imagen propia

- También se puede anexar a la combinación anterior el efecto hover:

```
<table class="table table-dark table-striped table-hover">
```

Ilustración 652 - Imagen propia

Nombre	Apellido	No. Cuenta
Abril	Alba	646388569
Luz	Villa	392533955
Jose Luis	Garduño	858435678

Ilustración 653 - Imagen propia

- Al igual que se puede tener una tabla desnuda:

```
<table class="table table-borderless">
```

Ilustración 654 - Imagen propia

Nombre	Apellido	No. Cuenta
Abril	Alba	646388569
Luz	Villa	392533955
Jose Luis	Garduño	858435678

Ilustración 655 - Imagen propia

- Algo curioso es que se puede mantener una tabla con el estilo default, pero hacer un efecto en el cuerpo de la tabla:

```

<tr class="table-success">
  <td>Abril</td>
  <td>Alba</td>
  <td>646388569</td>
</tr>
<tr class="table-danger">
  <td>Luz</td>
  <td>Villa</td>
  <td>392533955</td>
</tr>
<tr class="table-info">
  <td>Jose Luis</td>
  <td>Garduño</td>
  <td>858435678</td>
</tr>

```

Ilustración 656 - Imagen propia

Nombre	Apellido	No. Cuenta
Abril	Alba	646388569
Luz	Villa	392533955
Jose Luis	Garduño	858435678

Ilustración 657 - Imagen propia

- Y como es de esperarse, si se puede afectar al cuerpo de la tabla, también puede afectarse la cabecera, para eso se deben de colocar las clases dentro de las etiquetas correspondientes.

```

<table class="table table-striped">
  <thead class="thead-dark">

```

Ilustración 658 - Imagen propia

Nombre	Apellido	No. Cuenta
Abril	Alba	646388569
Luz	Villa	392533955
Jose Luis	Garduño	858435678

Ilustración 659 - Imagen propia

Solo existe un pequeño inconveniente dentro de las tablas, pues no se convierten en responsivas, lo que hace **Bootstrap** solo es asignarles un color y diseño, aunque este detalle puede solucionarse de una manera muy fácil y rápida, necesitaría anexarse un **<DIV>** padre, que envuelva completamente a la tabla y se le agregue la clase **TABLE-RESPONSIVE**, de esa manera la tabla se adaptará a cualquier tamaño de pantalla.

Para determinar diferentes estilos dentro de las imágenes, **Bootstrap** trabaja de igual manera por medio de clases, estas están inspiradas y formadas por medio de **CSS**, como en las tablas, las clases de las imágenes también pueden combinarse para crear mejores diseños y posiciones de imagen.

- Imagen con esquinas redondeadas:

```

```

Ilustración 660 - Imagen propia



Ilustración 661 - Imagen propia

- Imagen redondeada:

```

```

Ilustración 662 - Imagen propia

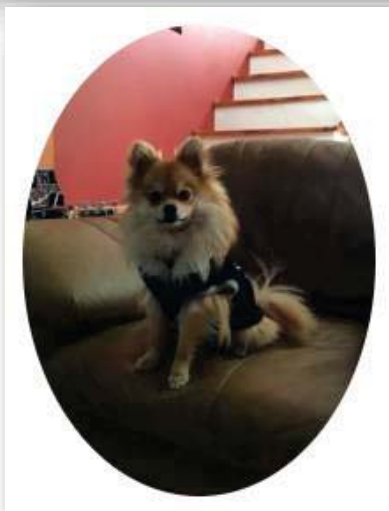


Ilustración 663 - Imagen propia

- Imagen enmarcada o bordeada:

```

```

Ilustración 664 - Imagen propia

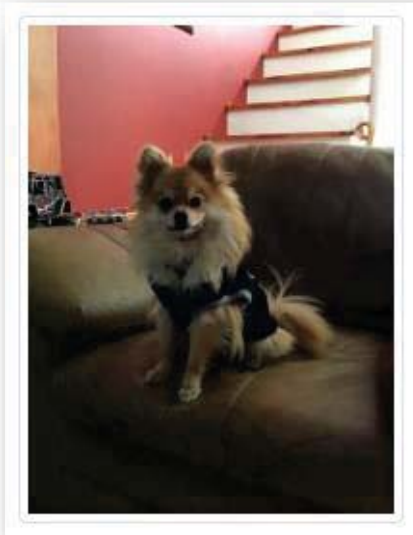


Ilustración 665 - Imagen propia

Ahora bien, **Bootstrap** también tiene la virtud de dar facilidad al alineamiento de imágenes:

- Alineación izquierda:

```

```

Ilustración 666 - Imagen propia



Ilustración 667 - Imagen propia

- Alineación derecha:

```

```

Ilustración 668 - Imagen propia

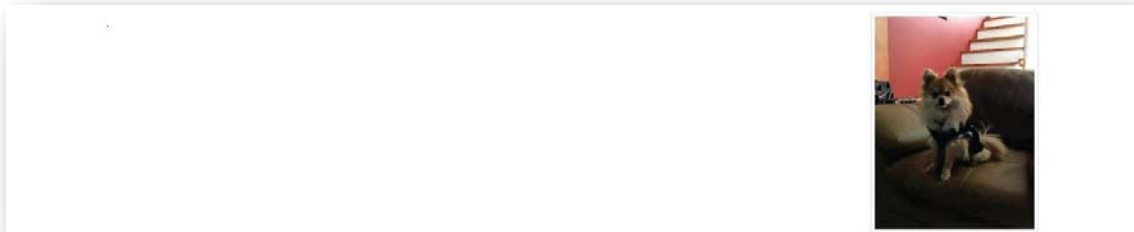


Ilustración 669 - Imagen propia

- Centrar imagen, **MX-AUTO** es el margen automático y **D-BLOCK** vuelve bloque a la imagen para poder centrarla:

```

```

Ilustración 670 - Imagen propia

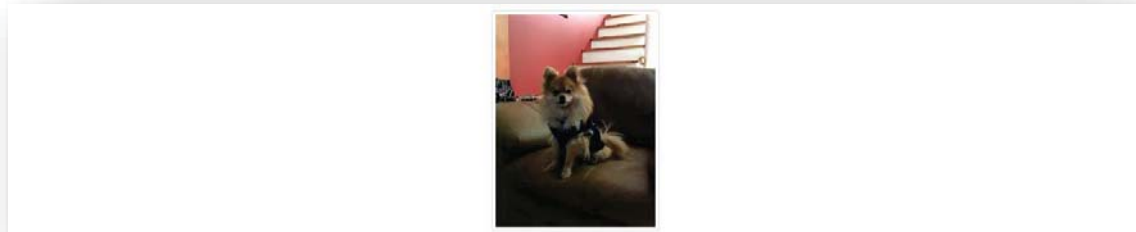


Ilustración 671 - Imagen propia

- Imagen responsiva:

```

```

Ilustración 672 - Imagen propia

Nombre	Apellido	No. Cuenta
Abril	Alba	646388569
Luz	Villa	392533955
Jose Luis	Garduño	858435678

A screenshot of a web browser showing the dog image from the previous illustrations, but now scaled to fit the width of the browser window, demonstrating a responsive image.

Ilustración 673 - Imagen propia

Para la presentación de una página principal, **Bootstrap** incorpora un nuevo elemento, donde crea un bloque grande en el inicio de una entrada, haciendo más llamativa a la página, a este elemento se le puede utilizar de muchas maneras, por supuesto eso dependerá del concepto de la página y el diseño que conlleve la misma.

```
<div class="jumbotron">
  <h1>Bootstrap</h1>
  <p>Aprende Diseño de Front-End con Bootstrap</p>
</div>
```

Ilustración 674 - Imagen propia

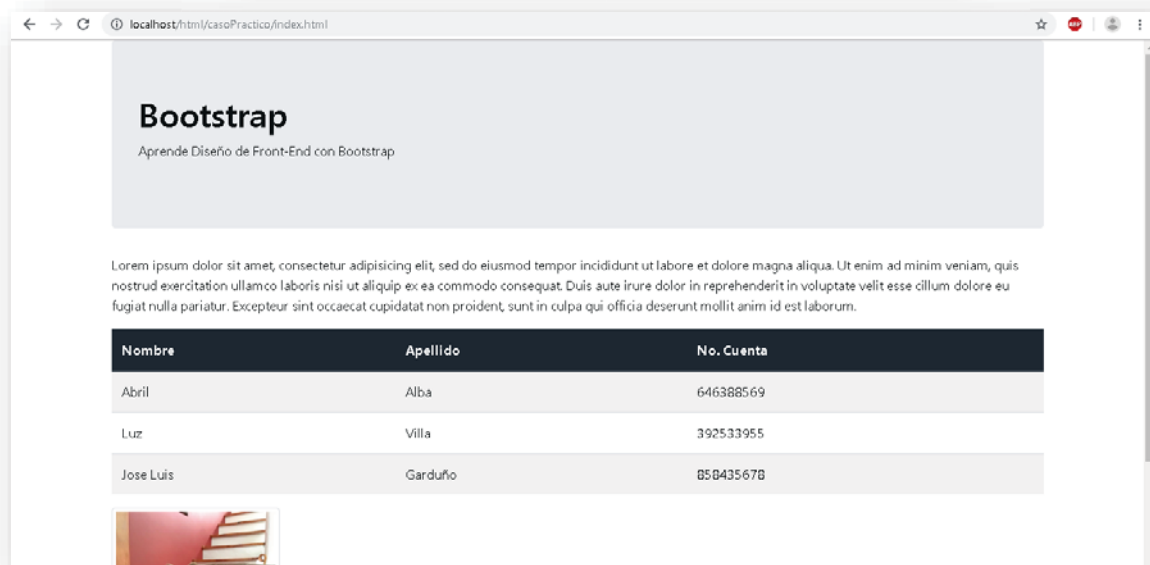


Ilustración 675 - Imagen propia

Para comenzar a dar cuerpo a una página, **Bootstrap** también permite el anexo de alertas, de esa forma se puede asegurar mediante un formulario o un aviso de alguna situación importante dentro de la página. Estos elementos se logran de la siguiente manera:

En primera instancia se declara el texto que se quiera volver una alerta, este texto estará dentro de un **<DIV>** con la clase **ALERT**, así:

```
<div class="alert">
  <strong>Aviso importante!</strong>
</div>
```

Ilustración 676 - Imagen propia



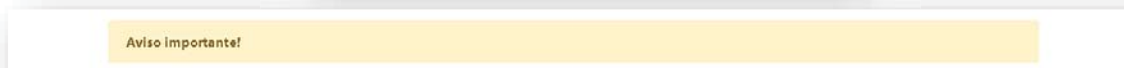
Ilustración 677 - Imagen propia

Como se puede observar en la imagen de la pantalla, no hay mayor cambio, más que un texto en negrita, pues para hacerlo más detallado, se le puede agregar la clase **ALERT-** después

del gui3n, se puede colocar alg3n nombre de colores que brinda **Bootstrap**, de esa manera la alerta cobra vida.

```
<div class="alert alert-warning">
  <strong>Aviso importante!</strong>
</div>
```

Ilustraci3n 678 - Imagen propia



Ilustraci3n 779 - Imagen propia

Claro que se puede seguir personalizando el aviso, pues si es una alerta, es porque necesita cierta importancia, para ello se agrega un link, pero se realiza un poquito diferente a lo que se tiene acostumbrado, pues se declara un enlace como en **HTML5** con **<A>** pero se le declarará una clase **ALERT-LINK**.

```
<div class="alert alert-warning">
  <strong>Aviso importante!</strong> Enterate de las 3ltimas actualizaciones <a href="#" class="alert-link">aqu3</a>.
</div>
```

Ilustraci3n 780 - Imagen propia



Ilustraci3n 781 - Imagen propia

Pero para lograr un efecto m3s interesante, puede permitirse una animaci3n, s3, as3 como lo lees, se puede lograr una animaci3n sin necesidad de a3adir **JS** o un **HOVER** de **CSS**, obviamente estos elementos nuevos est3n basados en **CSS** y **JS**, pero la gran ventaja es que s3lo se tienen que teclear un par de palabras reservadas para lograr hacer magia, pero bueno, para lograr hacer que se cierre esta alerta y desaparezca de pantalla se tomar3n en cuenta 3 diferentes clases, en primera instancia se le anexara una nueva clase **ALERT-DISMISSIBLE** al **<DIV>** padre, esa clase lo que har3 es meter un relleno a la derecha para que despu3s se coloque un bot3n, al que se le declarará una clase **CLOSE** y seguido el atributo **DATA-DISMISS** que llama un elemento **JS** para que permita la invisibilidad del elemento.

```
<div class="alert alert-warning alert-dismissible">
  <button type="button" class="close" data-dismiss="alert">&times;</button>
  <strong>Aviso importante!</strong> Enterate de las 3ltimas actualizaciones <a href="#" class="alert-link">aqu3</a>.
</div>
```

Ilustraci3n 782 - Imagen propia

Aqu3 esta:



Ilustraci3n 783 - Imagen propia

¡Y BUM! Ya no.



Ilustración 784 - Imagen propia

Y sólo por un detalle delicado que permite hacer que la alerta desaparezca con un sutil desvanecimiento, se le puede agregar las clases **FADE** y **SHOW** al **<DIV>** padre, para acceder al desvanecimiento.

Para una página web, son indispensables los botones y eso lo sabe **Bootstrap**, para ello, tiene clases específicas poder moldearlos y hacerlos muy presentables, lo primero que se tiene que hacer es declarar a los botones como se hace comúnmente en **HTML5**, después se le anexará el tipo que será **<BUTTON>** y la clase de Bootstrap **BTN**, aunada a esa clase también se le declarará un color para poder visualizarlo, por lo tanto, será **BTN-**.

```
<button type="button" class="btn btn-danger">Botón 1</button>
```

Ilustración 785 - Imagen propia

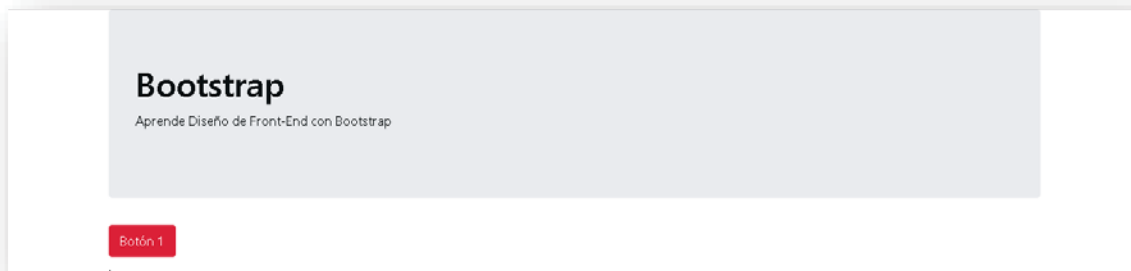


Ilustración 786 - Imagen propia

De esa manera ya se obtuvo un botón, pero la magia no acaba ahí, pues se puede declarar la misma clase a los enlaces **<A>** y formularios **<INPUT>**:

```
<a href="#" class="btn btn-info" role="button">Enlace-Botón</a>  
<input type="button" class="btn btn-success" value="Input Button">  
<input type="submit" class="btn btn-dark" value="Submit Button">
```

Ilustración 787 - Imagen propia

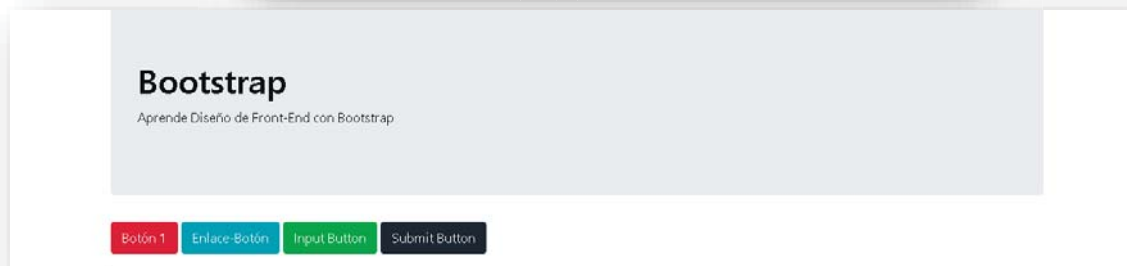


Ilustración 788 - Imagen propia

Los botones también pueden mostrar apariencia sin relleno con la clase **BTN-OUTLINE-COLOR**.

```
<button type="button" class="btn btn-outline-danger">Botón 1</button>
<a href="#" class="btn btn-outline-info" role="button">Enlace-Botón</a>
<input type="button" class="btn btn-outline-success" value="Input Button">
<input type="submit" class="btn btn-outline-dark" value="Submit Button">
```

Ilustración 789 - Imagen propia

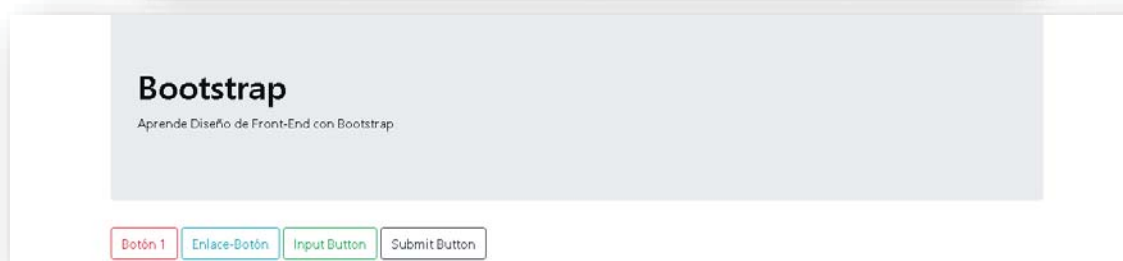


Ilustración 790 - Imagen propia

Como se puede alcanzar a notar, tanto el contorno o el borde como el texto adoptan el color que se les asignó y se ve muy agradable, muy combinado y sobre todo muy profesional. A los botones también se les puede asignar tamaños, solo varían por dos valores de clases como **BTN-LG** que hace el botón grande y **BTN-SM** que realiza lo contrario, pues hace miniatura al botón, por otra parte, existe un tercero, pero ese tamaño es el que otorga **Bootstrap** por default. Aunque está considerado como tamaño de botón, también se puede anexar a la categoría la clase **BTN-BLOCK**, el cual hace que el botón se extienda bastante y sea muy grande.

```
<button type="button" class="btn btn-danger btn-lg">Botón 1</button>
<a href="#" class="btn btn-info" role="button">Enlace-Botón</a>
<input type="button" class="btn btn-success btn-sm" value="Input Button">
<input type="submit" class="btn btn-dark btn-block" value="Submit Button">
```

Ilustración 791 - Imagen propia



Ilustración 792 - Imagen propia

Pero, aunque no lo creas, existen más mejoras para los botones y es aquí donde nos encontramos con las grandes animaciones, pues a los botones se les pueden anexar pequeños gif's de espera, estos se agregan en un **** con la clase **SPINNER-BORDER** y **SPINNER-GROW**.

```
<button type="button" class="btn btn-danger">
  <span class="spinner-border"></span>Botón 1
</button>

<a href="#" class="btn btn-info" role="button">
  <span class="spinner-grow"></span>Enlace-Botón
</a>
```

Ilustración 793 - Imagen propia

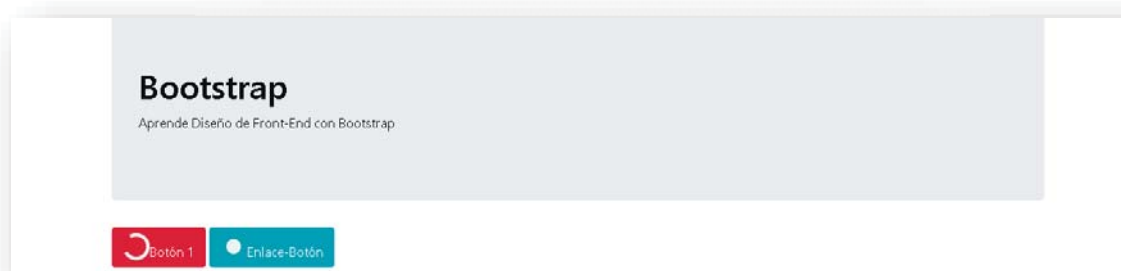


Ilustración 794 - Imagen propia

Ahora bien, dentro del tema de los botones también pueden unirse, es decir, generar un grupo de botones que generen un tipo de barra de navegación improvisada, pues cualquier elemento se puede utilizar a conveniencia del diseñador, para ello se crea un **<DIV>** padre que contendrá a los botones, ese **<DIV>** albergará una clase **BTN-GROUP**.

```
<div class="btn-group">
  <button type="button" class="btn btn-danger">Botón 1</button>
  <button type="button" class="btn btn-danger">Botón 1</button>
  <button type="button" class="btn btn-danger">Botón 1</button>
</div>
```

Ilustración 795 - Imagen propia

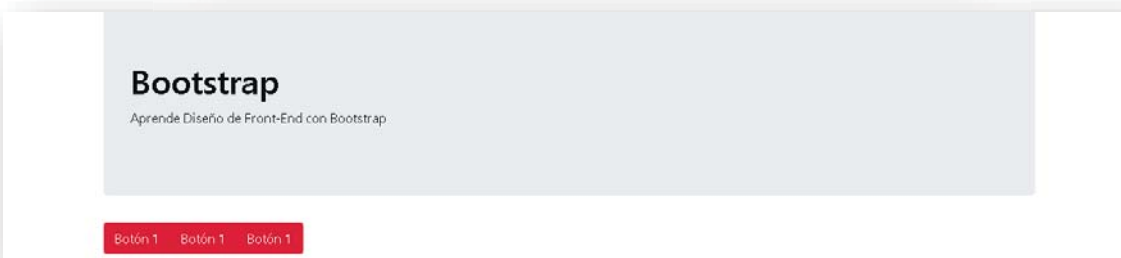


Ilustración 796 - Imagen propia

Partiendo de este punto, se puede moldear al grupo de botones verticalmente, para lograrlo, basta con incluir a la clase **BTN-GROUP-VERTICAL**.

```

<div class="btn-group btn-group-vertical">
  <button type="button" class="btn btn-danger">Botón 1</button>
  <button type="button" class="btn btn-danger">Botón 1</button>
  <button type="button" class="btn btn-danger">Botón 1</button>
</div>

```

Ilustración 797 - Imagen propia

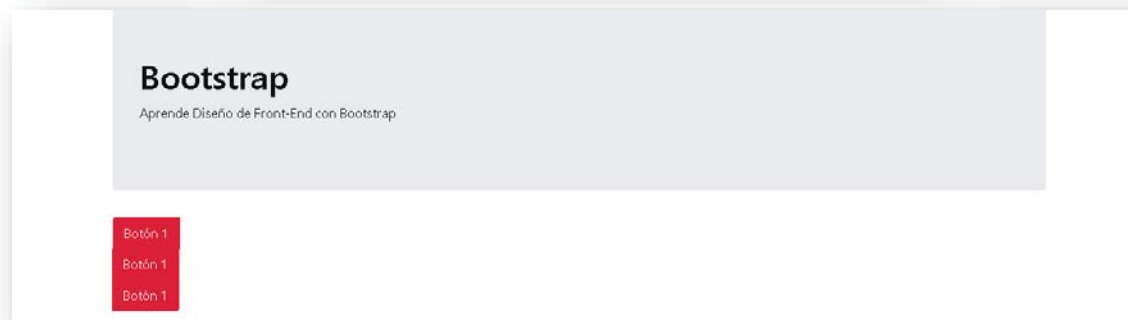


Ilustración 798 - Imagen propia

Ahora bien, un botón puede convertirse en un **DROPDOWN**, lo que significa que un botón o incluso grupo de botones se pueden adaptar a ser menús desplegables, esto se logra declarando un **<DIV>** padre con la clase **DROPDOWN**, dentro de ese **<DIV>** se creará al botón que se visualizará en pantalla, a ese botón se le declara la clase **BTN-COLOR** seguido de la siguiente clase **DROPDOWN-TOGGLE** la cual hará que el botón aparente un menú, mientras que el siguiente atributo que tendrá derivado de **JS** será **DATA-TOGGLE = DROPDOWN**, así se efectuará el efecto de deslizamiento de menú, por otro lado, después de la declaración de ese botón, se declarará un nuevo **<DIV>** donde tendrá la clase **DROPDOWN-MENU**, esta será la declaración del contenido del menú, por lo cual, dentro de ese **<DIV>** contendrá a los enlaces que culminarán al elemento como un menú desplegable, cabe resaltar que los enlaces tendrán incluida la clase **DROPDOWN-ITEM** esa clase es para que se coloquen de manera adecuada en la figura del menú.

```

<div class="dropdown">
  <button type="button" class="btn btn-danger dropdown-toggle" data-toggle="dropdown">Menú</button>
  <div class="dropdown-menu">
    <a href="#" class="dropdown-item">Opción 1</a>
    <a href="#" class="dropdown-item">Opción 2</a>
    <a href="#" class="dropdown-item">Opción 3</a>
  </div>
</div>

```

Ilustración 799 - Imagen propia

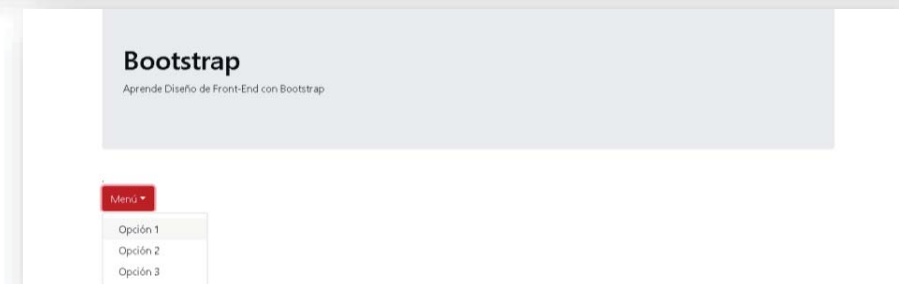


Ilustración 800 - Imagen propia

Para los elementos más detallados y complejos que maneja **Bootstrap**, se puede incluir las barras de progreso, estas barras son muy utilizadas dentro de cursos interactivos de internet, donde hacen la simulación de cuanto es el avance que se lleva en cierto curso o proceso que esté desarrollando el usuario, por lo tanto, dentro de todo el paquete de novedades que brinda **Bootstrap**, se pueden incluir estos elementos y se hace de la siguiente manera, creando un **<DIV>** con clase **PROGRESS**, dentro de ese elemento, se creará un nuevo **<DIV>** donde tendrá la clase **PROGRESS-BAR** y solo por visualizar como luciría una barra, se le anexa un ancho a la barra, por medio de un estilo **CSS**:

```
<div class="progress">
  <div class="progress-bar" style="width: 30%;"></div>
</div>
```

Ilustración 801 - Imagen propia

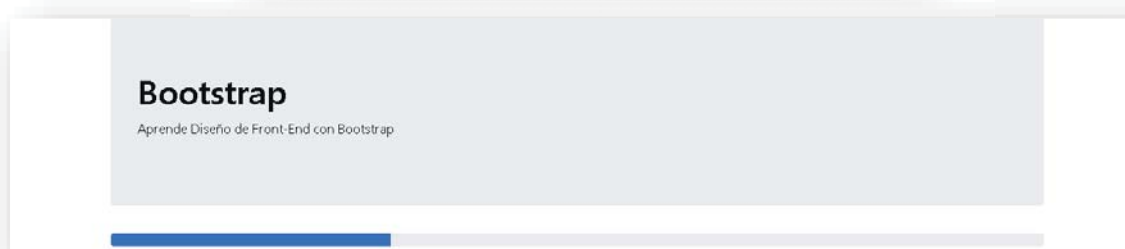


Ilustración 802 - Imagen propia

Como todo elemento que se ha visto a lo largo del tema, los tamaños se le asignan con las siguientes clases **LG** grande, **SM** pequeño, **MD** mediano, **LX** extra grande, por ende, los tamaños se anexarán a las clases con sus respectivas funciones, pues si es botón sería **BTN-SM**, para una tabla sería **TABLE-MD**, en este caso se utilizan las medias que brinda **CSS**, así que, para hacer un poco más alta a la barra de navegación, se hace desde el **<DIV>** padre, donde se le adjunta el estilo de altura.

```
<div class="progress" style="height:40px">
  <div class="progress-bar" style="width: 30%;"></div>
</div>
```

Ilustración 803 - Imagen propia

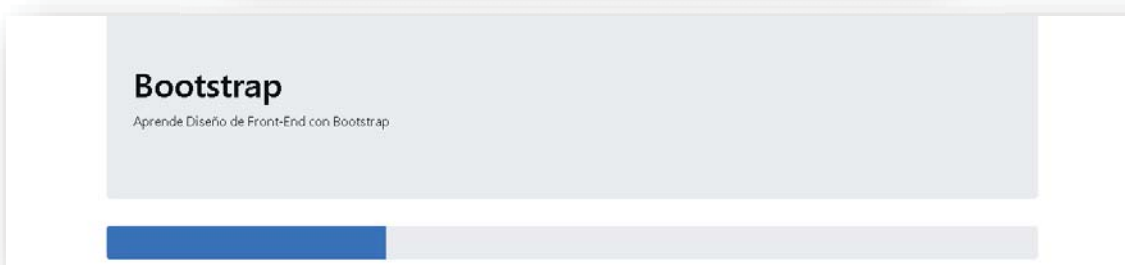


Ilustración 804 - Imagen propia

Cuando se trata de detallar al elemento, se puede iniciar por incluir el número del porcentaje que se indica en el color, de igual manera puede cambiarse el color de la barra de navegación por medio de la clase **BG-COLOR** lo cual declara que se le asigna un Background a la barra.

```
<div class="progress" style="height:40px">
  <div class="progress-bar bg-dark" style="width: 30%;">30%</div>
</div>

<div class="progress" style="height:40px">
  <div class="progress-bar bg-warning" style="width: 70%;">70%</div>
</div>

<div class="progress" style="height:40px">
  <div class="progress-bar bg-success" style="width: 5%;">5%</div>
</div>
```

Ilustración 805 - Imagen propia

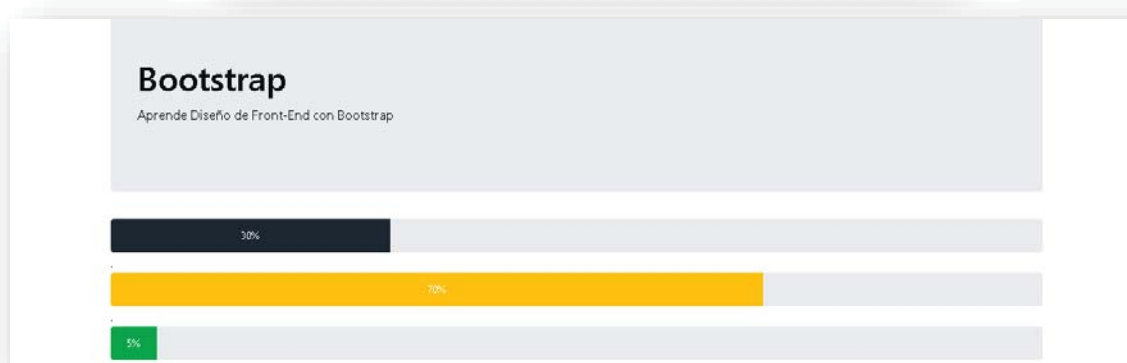


Ilustración 806 - Imagen propia

Una barra también puede tener estilo, para ello se le puede anexar el estilo cebra que se conoció en las tablas, por medio de la clase **PROGRESS-BAR-STRIPED**.

```
<div class="progress" style="height:40px">
  <div class="progress-bar bg-dark progress-bar-striped" style="width: 30%;">30%</div>
</div>

<div class="progress" style="height:40px">
  <div class="progress-bar bg-warning progress-bar-striped" style="width: 70%;">70%</div>
</div>

<div class="progress" style="height:40px">
  <div class="progress-bar bg-success progress-bar-striped" style="width: 5%;">5%</div>
</div>
```

Ilustración 807 - Imagen propia



Ilustración 808 - Imagen propia

Y como es de esperarse, también se pueden animar las barras de progreso, la importancia de eso, es brindarle una mejor apariencia a la página y hacer más interesante la estancia de la misma, sólo se necesita anexar la clase **PROGRESS-BAR-ANIMATED**.

Cuando se habla de un sitio web, se puede tener miles de diferentes conceptos, pero para todos aquellos que son tipo informativos, necesitan estar pasando de una a una las páginas web, tal cual, si fuese un libro, por lo tanto, se tiene el elemento **PAGINACIÓN**, que permite enumerar a los elementos que conformarán al sitio y de esa manera al pasar de una a otra página, este elemento apuntará donde nos encontramos. Para poder crearlo, se necesita de una lista, y en la etiqueta **** se le insertará la clase **PAGINATION**, previo a eso, los elementos **** que se encuentren dentro de la lista, necesitarán de la clase **PAGE-ITEM** que permite señalar el número de página que representará, dentro de esos elementos también existirá un enlace **<A>** el cual también tendrá que anexar una clase **PAGE-LINK** para poder efectuar el cambio de una página a otra, claro está, que en **HREF** se le insertará la dirección de la página que representa.

```
<ul class="pagination">
  <li class="page-item"><a class="page-link" href="#">Anterior</a></li>
  <li class="page-item"><a class="page-link" href="#">1</a></li>
  <li class="page-item"><a class="page-link" href="#">2</a></li>
  <li class="page-item"><a class="page-link" href="#">3</a></li>
  <li class="page-item"><a class="page-link" href="#">Siguiente</a></li>
</ul>
```

Ilustración 809 - Imagen propia

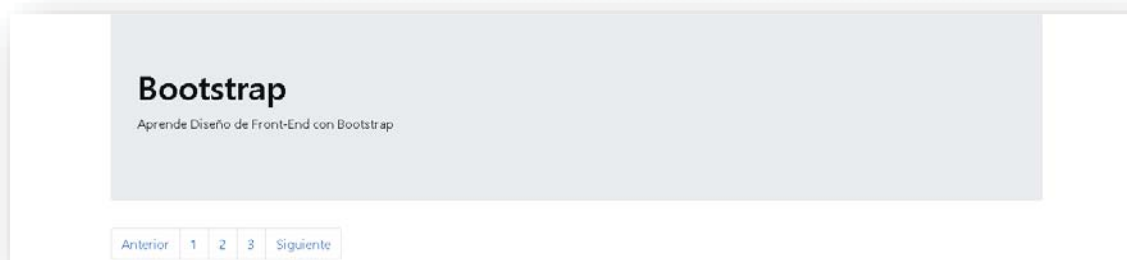


Ilustración 810 - Imagen propia

En esencia, ese es el estilo que arroja por default **Bootstrap**, pero sin duda, tiene varios trucos bajo la manga, pues los estilos se pueden cambiar de manera significativa, para poder ubicarse en la página donde se encuentre, es necesario que esa página mantenga la clase **ACTIVE**, así cuando se genere un clic a la página siguiente, se desactive la anterior y la siguiente se vuelva activa.

```
<ul class="pagination">
  <li class="page-item"><a class="page-link" href="#">Anterior</a></li>
  <li class="page-item active"><a class="page-link" href="#">1</a></li>
  <li class="page-item"><a class="page-link" href="#">2</a></li>
  <li class="page-item"><a class="page-link" href="#">3</a></li>
  <li class="page-item"><a class="page-link" href="#">Siguiente</a></li>
</ul>
```

Ilustración 811 - Imagen propia

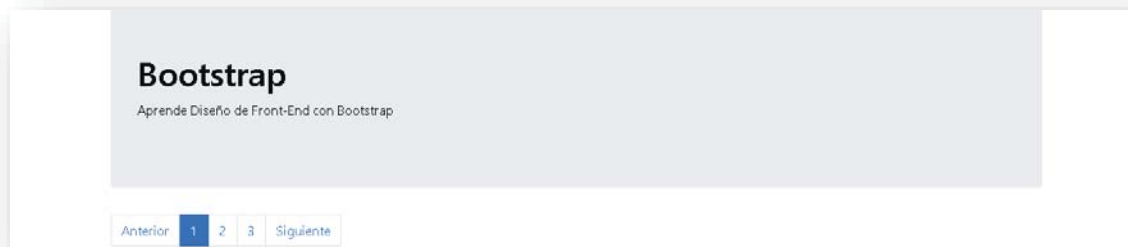


Ilustración 812 - Imagen propia

La orientación de alineación de este elemento también puede cambiar y se realiza por medio de dos clases **JUSTIFY-CONTENT-CENTER** lo cual permite centrar al elemento y **JUSTIFY-CONTENT-END** posiciona al elemento de lado derecho, esta clase se tiene que colocar en el elemento ****.

```
<ul class="pagination">
  <li class="page-item"><a class="page-link" href="#">Anterior</a></li>
  <li class="page-item active"><a class="page-link" href="#">1</a></li>
  <li class="page-item"><a class="page-link" href="#">2</a></li>
  <li class="page-item"><a class="page-link" href="#">3</a></li>
  <li class="page-item"><a class="page-link" href="#">Siguiente</a></li>
</ul>

<ul class="pagination justify-content-center">
  <li class="page-item"><a class="page-link" href="#">Anterior</a></li>
  <li class="page-item active"><a class="page-link" href="#">1</a></li>
  <li class="page-item"><a class="page-link" href="#">2</a></li>
  <li class="page-item"><a class="page-link" href="#">3</a></li>
  <li class="page-item"><a class="page-link" href="#">Siguiente</a></li>
</ul>

<ul class="pagination justify-content-end">
  <li class="page-item"><a class="page-link" href="#">Anterior</a></li>
  <li class="page-item active"><a class="page-link" href="#">1</a></li>
  <li class="page-item"><a class="page-link" href="#">2</a></li>
  <li class="page-item"><a class="page-link" href="#">3</a></li>
  <li class="page-item"><a class="page-link" href="#">Siguiente</a></li>
</ul>
```

Ilustración 813 - Imagen propia

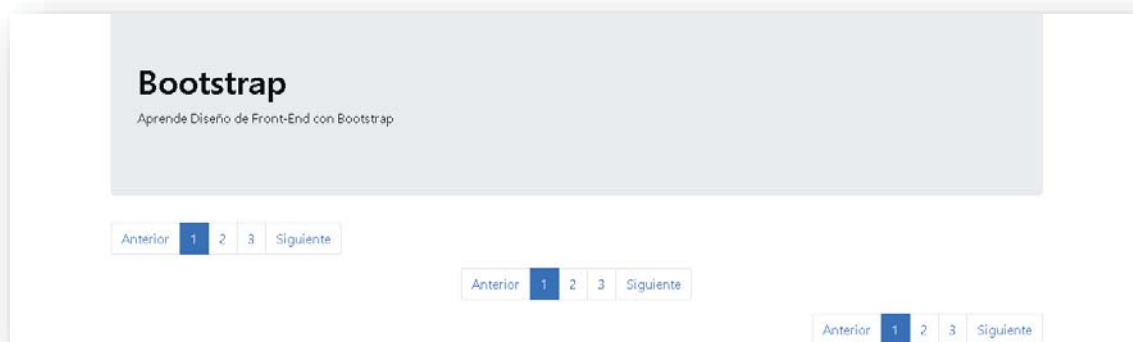


Ilustración 814 - Imagen propia

En este caso, no se tienen clases para cambios de color o forma, entonces es ahí cuando entra al rescate **CSS**, pues de cualquier manera que no se quiera tener el color o fuente, inclusive formas de los elementos, estos pueden ser reemplazados por medio de las clases en **CSS**.

También se cuenta con el famoso elemento **COLLAPSE** o **ACORDEÓN**, este es uno de los elementos que hacen favorecen la lectura de páginas informativas, pues la animación que contiene, ameniza la lectura.

Para poder crearlo se comienza por un botón, ahora este contendrá un atributo de **JS DATA-TOGGLE = COLLAPSE** lo cual, permitirá el efecto de deslizamiento y aparición del texto, después se creará un **<DIV>** donde se le incluirán una clase **COLLAPSE** que significa que ocultará el texto y también un identificador, ahora bien, regresando al botón se le incluirá un nuevo atributo de **JS DATA-TARGET** y se le asignara el nombre del identificador del **<DIV>**, de esa manera se podrá realizar el efecto Acordeón en pantalla.

```
<button data-toggle="collapse" data-target="#prueba">Acordeón</button>
<div id="prueba" class="collapse">
  Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod
  tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam,
  quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo
  consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse
  cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non
  proident, sunt in culpa qui officia deserunt mollit anim id est laborum.
</div>
```

Ilustración 815 - Imagen propia

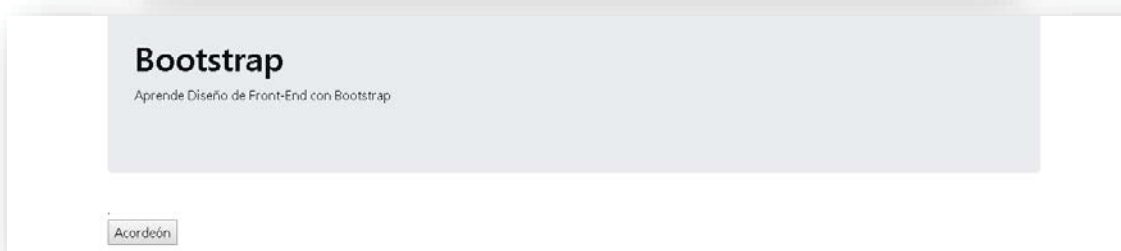


Ilustración 816 - Imagen propia



Ilustración 817 - Imagen propia

Dato curioso es que cuando se vuelve a dar clic en el botón, se vuelve a ocultar el contenido, también se puede cambiar la apariencia del botón, como se mostró en su espacio de este tema, los acordeones no solo se pueden generar con botones, también se puede hacer uso estilos carta, se crea un **<DIV>** abuelo, el cual será identificado con el nombre “Acordeon”.

Después se creará un nuevo **<DIV>** padre, ese tendrá la clase **CARD** y dentro de ese **<DIV>** se crearán otros dos **<DIV>** hermanos, el primero tendrá la clase **CARD-HEADER** su principal función es ser el elemento al que se le dé clic para que desplace el contenido, dentro tendrá un enlace **<A>** la cual albergará una clase **CARD-LINK** esta se declara como un enlace, después tendrá el atributo JS **DATA-TOGGLE=COLLAPSE**.

El segundo **<DIV>** hermano se identificará como **ACORDEON1** este identificador estará enlazado al enlace **<A>** del primer **<DIV>** hermano, y se colocará en **HREF=#ACORDEON1** de esa manera ya se estableció que cuando se dé clic al enlace, este desatará el contenido del segundo **<DIV>** hermano, continuando con el segundo **<DIV>** hermano contará con la clase **COLLAPSE-SHOW** esta funcionará de manera que permitirá que aparezca su contenido, no obstante también tendrá un atributo JS **DATA-PARENT=#ACORDEON** de esa manera se asegura que cuando se seleccione otro enlace, este guarde de nuevo su información y vuelva a su estado original.

Ahora bien, dentro del segundo **<DIV>** hermano, se creará un nuevo **<DIV>** el cual tendrá la clase **CARD-BODY** y se le agregará el texto deseado.

```

<div id="Acordeon">
  <div class="card">
    <div class="card-header">
      <a class="card-link" data-toggle="collapse" href="#Acordeon1">
        Tema 1
      </a>
    </div>
    <div id="Acordeon1" class="collapse show" data-parent="#Acordeon">
      <div class="card-body">
        Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.
      </div>
    </div>
  </div>
</div>

```

Ilustración 818 - Imagen propia

De esa manera se pueden agregar las cartas que sean necesarias para lograr esto:



Ilustración 819 - Imagen propia



Ilustración 820 - Imagen propia

Ya entrados en los temas complejos que contiene Bootstrap, también se pueden moldear los formularios, si bien se sabe, estos son utilizados para recibir, almacenar y manipular datos de entrada y salida, esto tiene que ver con programación, pero en lo que respecta a Bootstrap, tiene detalles significativos para estos elementos. Se cuenta con dos tipos de estilos, el primero se llama “forma apilada”, se llama así porque la clase **FORM-CONTROL** otorga el 100% de ancho a los elementos que lo contengan, para comenzar se declara un elemento **<FORM>** dentro de este se creará un **<DIV>** donde se le declara la clase **FORM-GROUP**, esto señala que creará un grupo de formulario, de esa manera, se agregaran dentro los elementos **<INPUT>** y **<LABEL>** el único elemento que mantendrá una clase de Bootstrap será **<INPUT>** como **FORM-CONTROL**.

```
<form>
  <div class="form-group">
    <label for="email">Usuario (Correo Electronico):</label>
    <input type="email" class="form-control" id="email">
  </div>
```

Ilustración 821 - Imagen propia

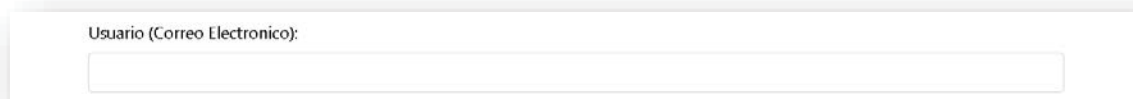
Un formulario con un campo de texto rectangular. Encima del campo, el texto "Usuario (Correo Electronico):" está etiquetado. El campo de texto está vacío.

Ilustración 822 - Imagen propia

De esa manera, se pueden ingresar los parámetros que sean necesarios para una base de datos, todos los elementos que se le incluyan tendrán el mismo diseño y estilo. Entendido ese punto, existen más elementos diferentes de un formulario que se le pueden agregar, como son los **CHECK-BOX**, dentro del **<DIV>** que define el grupo del formulario se anexará la clase **FORM-CHECK** de esa manera ya está definido, en este caso adentro contará con los elementos **<LABEL>** e **<INPUT>**, cada una tendrá la clase respectiva a su elemento **FORM-CHECK-LABEL** y **FORM-CHECK-INPUT**.

```
<div class="form-group form-check">  
  <label class="form-check-label">  
    <input class="form-check-input" type="checkbox"> Guardar contraseña  
  </label>  
</div>
```

Ilustración 823 - Imagen propia

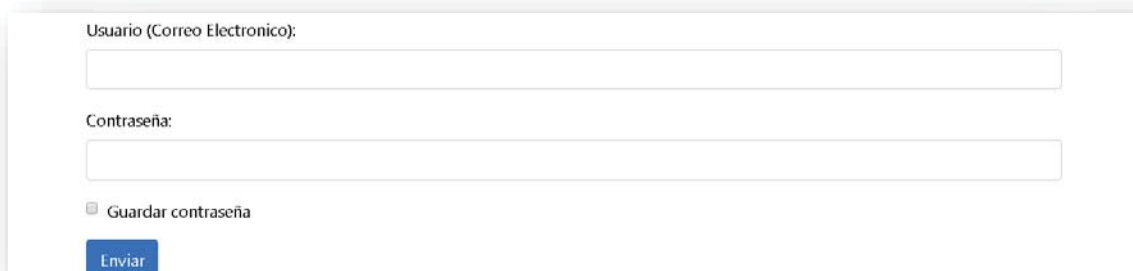
Un formulario con dos campos de texto. El primer campo está etiquetado como "Usuario (Correo Electronico):". El segundo campo está etiquetado como "Contraseña:". Debajo del segundo campo, hay un checkbox con el texto "Guardar contraseña" a su derecha. En la parte inferior izquierda del formulario, hay un botón azul con el texto "Enviar".

Ilustración 824 - Imagen propia

Después se tiene el segundo estilo que es “forma en línea”, su intención es reducir el espacio o comprimirlo, de manera que todos los elementos se coloquen en una sola fila. Esto se logra añadiendo la clase **FORM-INLINE** al elemento **<FORM>**, aquí se eliminarán los **<DIV>** que se encargan de agrupar a los elementos, excepto al de **CHECK-BOX**, para ese elemento solo se elimina la clase **FORM-GROUP**, después a todos los elementos de **<LABEL>** e **<INPUT>** incluido el **<DIV>** de **CHECK-BOX**, tendrán la clase **MR-SM-2** su objetivo es darles a los elementos un margen derecho, por lo cual, se alinearán de manera automática.


```

<form class="form-inline">
  <label for="email" class="mr-sm-2">Usuario (Correo Electronico):</label>
  <input type="email" class="form-control mr-sm-2" id="email">
  <label for="pwd" class="mr-sm-2">Contraseña:</label>
  <input type="password" class="form-control mr-sm-2" id="pwd">
  <div class="form-check mr-sm-2">
    <label class="form-check-label">
      <input class="form-check-input" type="checkbox"> Guardar contraseña
    </label>
  </div>
  <button type="submit" class="btn btn-primary">Enviar</button>
</form>

```

Ilustración 825 - Imagen propia

Ilustración 826 - Imagen propia

Cuando se trata de un formulario, su principal función es recaudar datos, pero ¿Qué pasa si no se ingresan datos en los elementos que reciben la información?, Bueno, Bootstrap cuenta con clases de validación, solo basta con incluir la clase **WAS-VALIDATED** y en los elementos **<INPUT>** agregar el atributo de JS **REQUIRED**.

```

<form class="form-inline was-validated">
  <label for="email" class="mr-sm-2">Usuario (Correo Electronico):</label>
  <input type="email" class="form-control mr-sm-2" id="email" required>
  <label for="pwd" class="mr-sm-2">Contraseña:</label>
  <input type="password" class="form-control mr-sm-2" id="pwd" required>
  <div class="form-check mr-sm-2">
    <label class="form-check-label">
      <input class="form-check-input" type="checkbox" required> Guardar contraseña
    </label>
  </div>
  <button type="submit" class="btn btn-primary">Enviar</button>
</form>

```

Ilustración 827 - Imagen propia

Y con eso, se distingue cuando introducen datos:

Ilustración 828 - Imagen propia

Y cuando no:

Ilustración 829 - Imagen propia

Un tema muy importante y que es imprescindible dentro de una página web, es una barra de navegación, y como se recordara en el tema de **CSS**, tiene su chiste para poder lograr una barra de navegación, por lo tanto, **Bootstrap** ofrece una manera más fácil y rápida de declarar a este elemento. Todo parte creando un elemento **<NAV>** este tendrá la clase **NAVBAR**

donde declara el inicio de una barra de navegación, también tendrá la clase **NAVBAR-EXPAND-SM** su función será colapsar a la barra en caso de que se abra la página en una pantalla pequeña, de igual manera, el tamaño puede cambiarse.

Después se agrega una lista, comenzando por el elemento **** contará con la clase **NAV** la cual declara que se realizará una barra de navegación, después a los elementos **** se les asignará la clase **NAV-ITEM** y dentro de estos elementos, se les añadirán enlaces **<A>** lo cuales albergarán las clases **NAV-LINK** y eso es todo.

```
<ul class="nav">
  <li class="nav-item">
    <a class="nav-link" href="#">Home</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="#">Temas</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="#">Ayuda</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="#">Acerca</a>
  </li>
</ul>
```

Ilustración 830 - Imagen propia

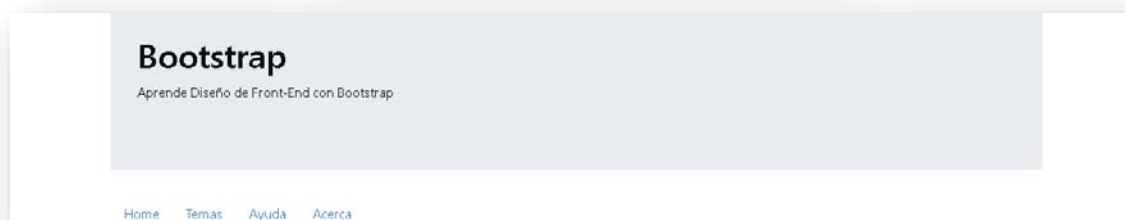


Ilustración 831 - Imagen propia

Se cuenta con el mismo sistema de alineación que en el elemento **PAGINACIÓN**, este se declara en los elementos ****, por otra parte, hablando de alineaciones, también se tiene una clase que permite hacer la barra de navegación horizontal **FLEX-COLUMN**.

```
<ul class="nav flex-column">
```

Ilustración 832 - Imagen propia



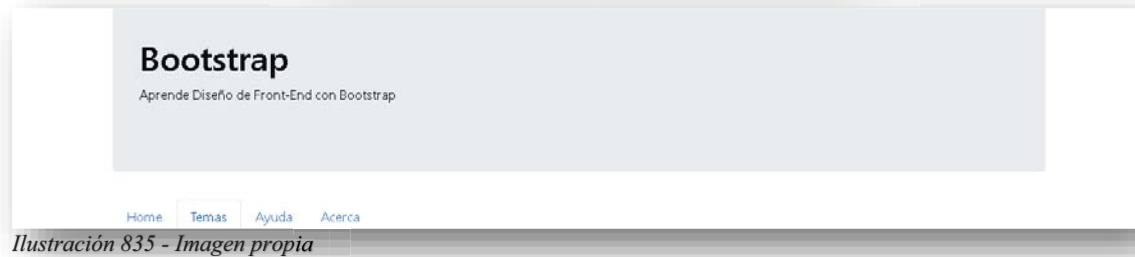
Ilustración 833 - Imagen propia

A continuación, se muestran los diferentes tipos de estilos que se tiene dentro de **Bootstrap** para las barras de navegación.

- Estilo pestaña:

```
<ul class="nav nav-tabs">
```

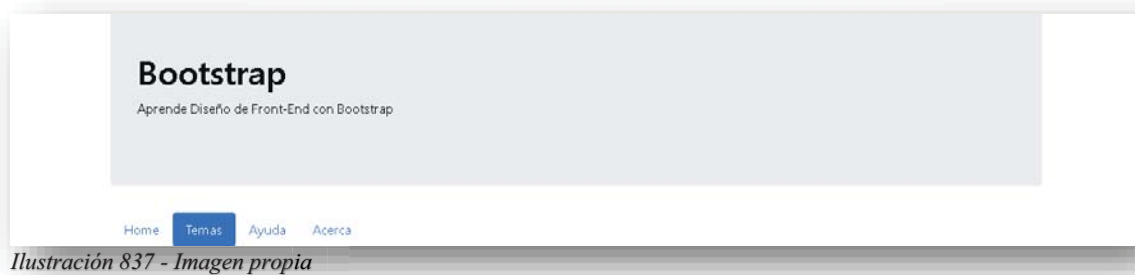
Ilustración 834 - Imagen propia



- Estilo pastillas:

```
<ul class="nav nav-pills">
```

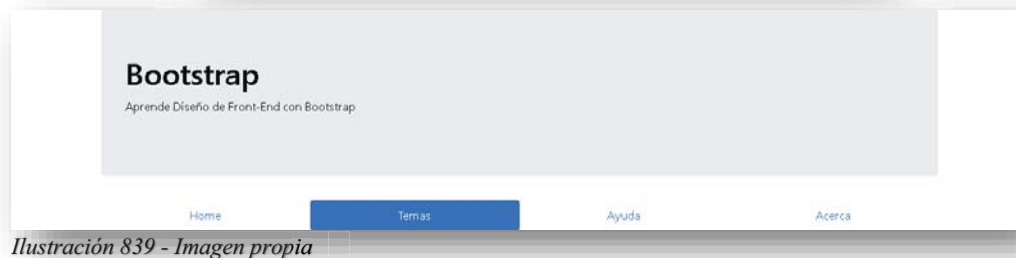
Ilustración 836 - Imagen propia



- Estilo pastilla justificada:

```
<ul class="nav nav-pills nav-justified">
```

Ilustración 838 - Imagen propia



Ese efecto, también se puede lograr en el estilo anterior, por otra parte, una barra de navegación suele tener opciones desplegables, para ello, lo que se tiene que alterar, son los elementos **** se les declara la clase **DROPDOWN** y al enlace **<A>** la clase **DROPDOWN-TOGGLE** y el atributo **JS DATA-TOGGLE=DROPDOWN** y así debajo del enlace, pero dentro del elemento **** se crea un **<DIV>** con la clase **DROPDOWN-MENU** y se le insertarán todos los enlaces que se deseen con su respectiva clase **DROPDOWN-ITEM**.

```
<ul class="nav nav-pills nav-justified">
  <li class="nav-item">
    <a class="nav-link" href="#">Home</a>
  </li>
  <li class="nav-item dropdown">
    <a class="nav-link dropdown-toggle active" data-toggle="dropdown" href="#">Temas</a>
    <div class="dropdown-menu">
      <a class="dropdown-item" href="#">HTML</a>
      <a class="dropdown-item" href="#">CSS</a>
      <a class="dropdown-item" href="#">JS</a>
      <a class="dropdown-item" href="#">BOOTSTRAP</a>
    </div>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="#">Ayuda</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="#">Acerca</a>
  </li>
</ul>
```

Ilustración 840 - Imagen propia

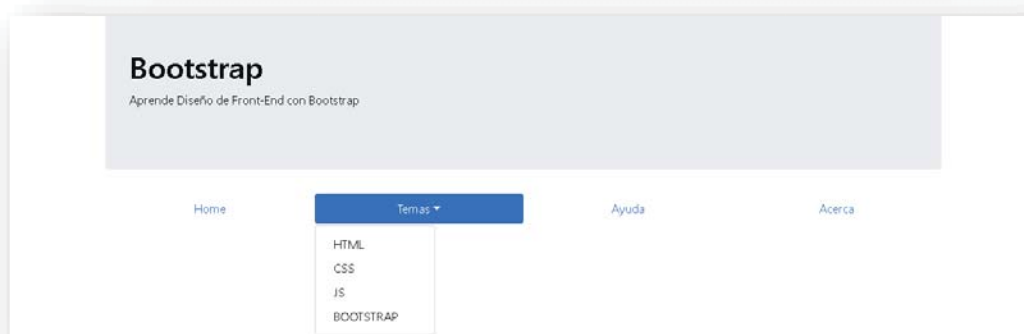


Ilustración 841 - Imagen propia

Dentro de **Bootstrap** también se cuenta con menús tabuladores, se asemejan mucho a los Acordeones, pero este sería en posición horizontal. Se comienza creando un **<DIV>** con la clase **TAB-CONTENT** de esta manera se está declarando el contenido de cada enlace del menú, después se crea un **<DIV>** que se identificará con un nombre alusivo al enlace que estará enlazado del menú, junto con las clases **CONTAINER**, **TAB-PANE**. De esa manera lo único que faltaría, es anexar el texto que se mostrará en pantalla.

```

<div class="tab-content">
  <div id="home" class="container tab-pane active"><br>
    <h3>HOME</h3>
    <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.</p>
  </div>
  <div id="temas" class="container tab-pane fade"><br>
    <h3>Temas</h3>
    <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.</p>
  </div>
  <div id="ayuda" class="container tab-pane fade"><br>
    <h3>Ayuda</h3>
    <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.</p>
  </div>
</div>

```

Ilustración 842 - Imagen propia

Cabe resaltar que, en la imagen anterior, las clases también cuentan con otras clases como **FADE** lo que indica que oculte el contenido si aún no se activa su correspondiente enlace y la clase **ACTIVE** será la visualización del menú en pantalla. Después de eso, en el menú, específicamente en los enlaces, su atributo **HREF** almacenará el nombre del identificador de su respectivo panel.

```

<ul class="nav nav-tabs">
  <li class="nav-item">
    <a class="nav-link active" data-toggle="tab" href="#home">Home</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" data-toggle="tab" href="#temas">Temas</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" data-toggle="tab" href="#ayuda">Ayuda</a>
  </li>
</ul>

```

Ilustración 843 - Imagen propia

Ahora sí, enlazados los contenidos con los respectivos enlaces, se obtiene algo así:



Ilustración 844 - Imagen propia

Considerando que oficialmente se tiene una barra de navegación, aún se puede detallar mucho más, pues se le puede incluir un logo **** en la parte izquierda de la barra, ese detalle se logra colocando al inicio de la barra de navegación un enlace **<A>** al cual se le añadirá la clase **NAVBAR-BRAND**, de esa manera declara que ese espacio de código se convirtió en un almacenamiento de logo, por lo cual, se le puede ingresar la imagen que sea.

```
<a class="navbar-brand" href="#">
  
</a>
```

Ilustración 845 - Imagen propia

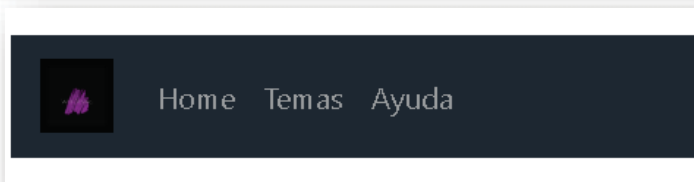


Ilustración 846 - Imagen propia

Y si no es aun suficiente, se tienen muchos más detalles para terminar de pulir una barra de navegación, como se ha conocido a lo largo de este tema, **Bootstrap** piensa en todo y para ello también permite anexar un buscador dentro de la barra, eso se logra con la declaración de un pequeño formulario **<FORM>** que contará con la clase **FORM-INLINE**, dentro tendrá elementos que constan de un **<BUTTON>** y un **<INPUT>** que tendrá las clases **FORM-CONTROL** y **MR-SM-2**.

```
<form class="form-inline">
  <input class="form-control mr-sm-2" type="text" placeholder="Search">
  <button class="btn btn-danger" type="button">Buscar</button>
</form>
```

Ilustración 847 - Imagen propia

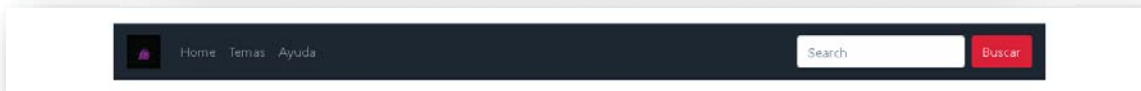


Ilustración 848 - Imagen propia

Aunque se tenga la opción de reducción o compresión de la barra de navegación para pantalla completa, esto también puede pulirse, pues puede crearse un elemento **<BUTTON>** con la función **COLLAPSE** de **Bootstrap** como se mostró anteriormente y después crear un **<DIV>** padre que envuelva a la lista y tenga los parámetros correspondidos de la función para que al momento de que la pantalla se haga pequeña, emerja el botón y al seleccionarlo, se despliegue la lista de la barra.

```
<nav class="navbar navbar-expand-lg navbar-dark bg-dark" style="margin:24px 0;">
  <a class="navbar-brand" href="#">
    
  </a>
  <button class="navbar-toggler navbar-toggler-right" type="button" data-toggle="collapse" data-target="#navb">
    <span class="navbar-toggler-icon"></span>
  </button>
```

Ilustración 849 - Imagen propia

```

<div class="collapse navbar-collapse" id="navb">
  <ul class="navbar-nav mr-auto">
    <li class="nav-item">
      <a class="nav-link" href="#">Home</a>
    </li>
    <li class="nav-item">
      <a class="nav-link" href="#">Temas</a>
    </li>
    <li class="nav-item">
      <a class="nav-link" href="#">Ayuda</a>
    </li>
  </ul>
  <form class="form-inline">
    <input class="form-control mr-sm-2" type="text" placeholder="Search">
    <button class="btn btn-danger" type="button">Buscar</button>
  </form>
</div>
</nav>

```

Ilustración 850 - Imagen propia

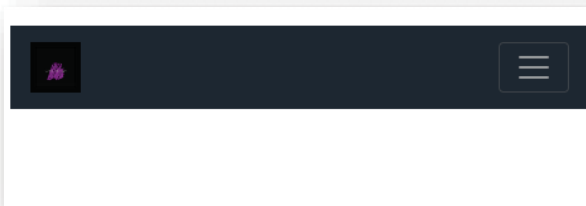


Ilustración 851 - Imagen propia



Ilustración 852 - Imagen propia

Aunado a esto también puede volverse fija la barra de navegación, pues como se conoció en CSS, cuando se desplace la pantalla por la página web, la barra no se perderá, todo lo contrario, se mantendrá fija para que se pueda tener acceso al resto del contenido, solo se añade la clase **FIXED-TOP** si se quiere fija a la barra en la parte superior o **FIXED-BOTTOM** para colocarse en la parte inferior.

También se tiene una nueva función para la barra de navegación y se llama **SCROLLSPY** su principal función es que cuando se desplace la página hacia abajo, actualice automáticamente a la barra e identifique en que apartado se encuentra de la misma, de esa manera se tiene mayor control de la posición en la que se encuentre el usuario, probablemente no sea la gran cosa, pero suele ser muy solicitado por el público.

Esta función se comienza anclando al elemento **<BODY>** a la barra de navegación, se logra por medio de los atributos de **JS DATA-SPY=SCROLL** declara que **<BODY>** será el lugar donde se establezca la función y honestamente es muy común que se declare a ese elemento en específico, **DATA-TARGET=.NAVBAR** con esta clase se ancla **<BODY>** con la barra **<NAV>**.

```

<body data-spy="scroll" data-target=".navbar">

```

Ilustración 853 - Imagen propia

Después será necesario anclar el contenido a la barra de navegación, será necesario declarar los **<DIV>** necesarios correspondientes al número de elementos o enlaces que tenga la barra de navegación, estos tendrán un identificador alusivo al tema correspondiente de la barra,

con ese identificador se enlazará a los enlaces que se encuentran en los elementos **** precisamente en el atributo **HREF**.

```
<ul class="navbar-nav">
  <li class="nav-item">
    <a class="nav-link" href="#html">HTML</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="#css">CSS</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="#bootstrap">BOOTSTRAP</a>
  </li>
</ul>
<form class="form-inline">
  <input class="form-control mr-sm-2" type="text" placeholder="Search">
  <button class="btn btn-success" type="button">Buscar</button>
</form>
</nav>

<div id="html" class="container-fluid bg-info" style="padding-top:70px;padding-bottom:70px">
  <h1>HTML</h1>
</div>
<div id="css" class="container-fluid bg-warning" style="padding-top:70px;padding-bottom:70px">
  <h1>CSS</h1>
</div>
<div id="bootstrap" class="container-fluid bg-danger" style="padding-top:70px;padding-bottom:70px">
  <h1>BOOTSTRAP</h1>
</div>
```

Ilustración 854 - Imagen propia

Ahora bien, solo faltaría agregar contenido, para observar el efecto.

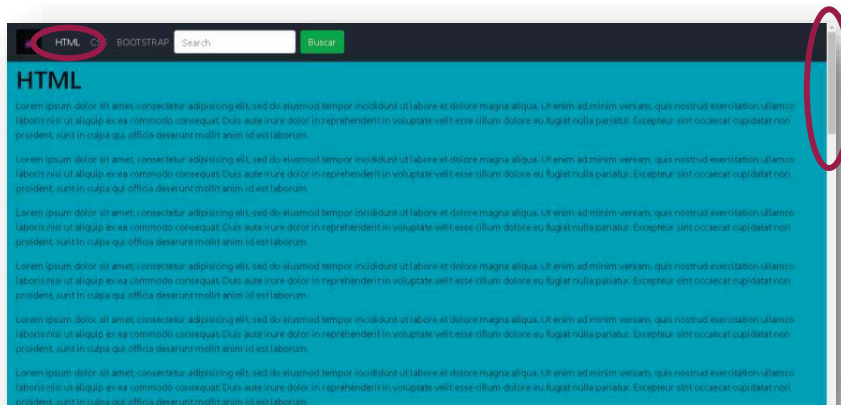


Ilustración 855 - Imagen propia

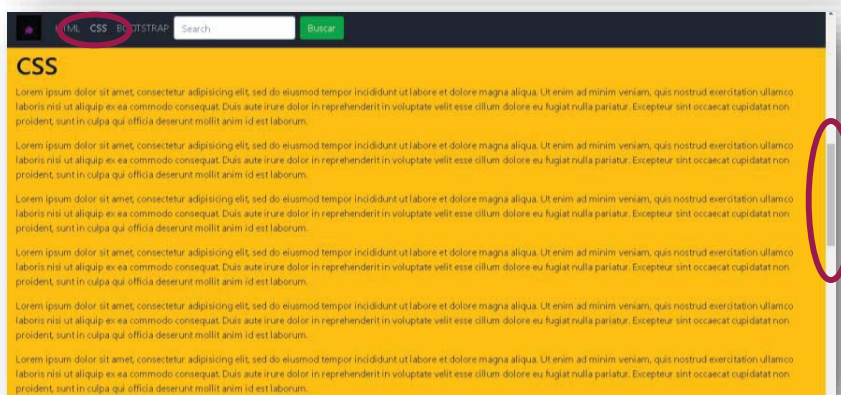


Ilustración 856 - Imagen propia

Y para aquellos pequeños detalles que necesita una página web, **Bootstrap** ofrece la opción de generar ventanas emergentes que pueden contener cualquiera de los elementos que se han visto con anterioridad, para poder aprovechar esta función se tiene que declarar un elemento **<BUTTON>** y como atributo **JS** se le añadirá **DATA-TOGGLE=MODAL** el cual declara al botón como una función de modal o ventana emergente, por otra parte, **DATA-TARGET=#NOMBRE** será el enlace que se tendrá con el contenido de **MODAL**.

```
<button type="button" class="btn btn-primary" data-toggle="modal" data-target="#nuevoModal">
  Seleccioname
</button>
```

Ilustración 857 - Imagen propia

Como un **MODAL** es semejante a una mini página web, por obvias razones, tendrá una cabecera, un cuerpo y un pie. Para poder crearlo, se declarará un elemento **<DIV>** donde contendrá una clase **MODAL** y un identificador **NOMBRE** ese será el que se coloca en **DATA-TARGET** del botón que se hizo anteriormente.

```
<div class="modal" id="nuevoModal">
```

Ilustración 858 - Imagen propia

Ahora bien, como se tiene claro que este elemento es una ventana emergente, pertenece a una categoría de alertas o diálogos, por lo tanto, se declarará otro **<DIV>** con la clase **MODAL-DIALOG** y dentro de ese mismo habrá otro **<DIV>** con la clase **MODAL-CONTENT** donde da apertura a desarrollar el cuerpo del **MODAL**.

```
<!-- Declaración del Modal -->
<div class="modal" id="nuevoModal">
  <div class="modal-dialog">
    <div class="modal-content">
```

Ilustración 859 - Imagen propia

Después dentro de los **<DIV>** anteriores, se anexarán 3 **<DIV>** más, cada uno será declarado por medio de clases como, cabecera **MODAL-HEADER**, cuerpo **MODAL-BODY** y pie **MODAL-FOOTER**, cada uno albergará un contenido diferente, todo depende del diseño y concepto de la información.

```
<!-- Cabecera Modal -->
<div class="modal-header">
  <h4 class="modal-title">Ventana emergente</h4>
  <button type="button" class="close" data-dismiss="modal">&times;</button>
</div>
```

Ilustración 860 - Imagen propia

```
<!-- Cuerpo Modal -->
<div class="modal-body">
  <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod
  tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam,
  quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo
  consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse
  cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non
  proident, sunt in culpa qui officia deserunt mollit anim id est laborum.</p>
</div>
```

Ilustración 861 - Imagen propia

```
<!-- Pie Modal -->
<div class="modal-footer">
  <button type="button" class="btn btn-danger" data-dismiss="modal">Cerrar</button>
</div>
```

Ilustración 862 - Imagen propia

Y de esa forma, se obtiene algo así:

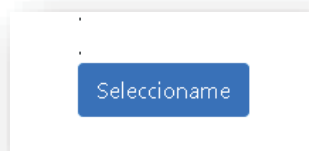


Ilustración 863 - Imagen propia



Ilustración 864 - Imagen propia

Después de todo este gran recorrido, se llega al final con el elemento “CARRUSEL”, como se puede aprender en las modificaciones de CSS con las imágenes, se realizó un tipo SLIDE o CARRUSEL de imágenes, bueno, Bootstrap también metió las manos en ese tema, por supuesto, de una manera rápida, fácil y mejorada. Como en todo se declara un <DIV> padre, este va a contener a todo el carrusel, por lo cual, tendrá la clase CAROUSEL SLIDE y el atributo de JS CAROUSEL, también tendrá un identificador alusivo a su función.

```
<div id="carousel" class="carousel slide" data-ride="carousel">
```

Ilustración 865 - Imagen propia

Después se declara una lista **** con la clase **CAROUSEL-INDICATORS**, este elemento albergará los indicadores de cuantas imágenes se van a presentar en el **CARRUSEL**, es importante mencionar que la cuenta de las imágenes comienza de **0-INFINITO** y los elementos **** tendrán el atributo JS **DATA-TARGET=#CAROUSEL**.

```
<ul class="carousel-indicators">
  <li data-target="#carousel" data-slide-to="0" class="active"></li>
  <li data-target="#carousel" data-slide-to="1"></li>
  <li data-target="#carousel" data-slide-to="2"></li>
  <li data-target="#carousel" data-slide-to="3"></li>
</ul>
```

Ilustración 866 - Imagen propia

A continuación, se declarará el cuerpo del **CARRUSEL** con un elemento **<DIV>** con la clase **CAROUSEL-INNER** como lo dice su nombre, se encargará de ejecutar el **CARRUSEL**, cada imagen estará dentro de otro **<DIV>** con la clase **CAROUSEL-ITEM**.

Por supuesto que el número de imágenes debe coincidir con el número de indicadores que se declararon anteriormente, sino no funcionará.

```
<div class="carousel-inner">
  <div class="carousel-item active">
    
  </div>
  <div class="carousel-item">
    
  </div>
  <div class="carousel-item">
    
  </div>
  <div class="carousel-item">
    
  </div>
</div>
```

Ilustración 867 - Imagen propia

Y, por último, es necesario insertar los controles del **CARRUSEL**, estos serán indicados por medio de enlaces **<A>**, serán dos elementos de ese tipo, ambos tendrán la clase **CAROUSEL-CONTROL-PREV** y **CAROUSEL-CONTROL-NEXT** respectivamente, también estarán enlazados al identificador del **<DIV>** padre por medio de **HREF:#CAROUSEL** y el atributo JS **DATA-SLIDE=PREV** y **DATA-SLIDE=NEXT** respectivamente con las clases que contenga cada enlace.

Dentro de los enlaces, almacenarán un elemento **** con la clase **CAROUSEL-CONTROL-PREV-ICON** y **CAROUSEL-CONTROL-NEXT-ICON**, estos solo servirán como iconos identificadores de las flechas de anterior y siguiente. Con todo esto se obtiene algo así:

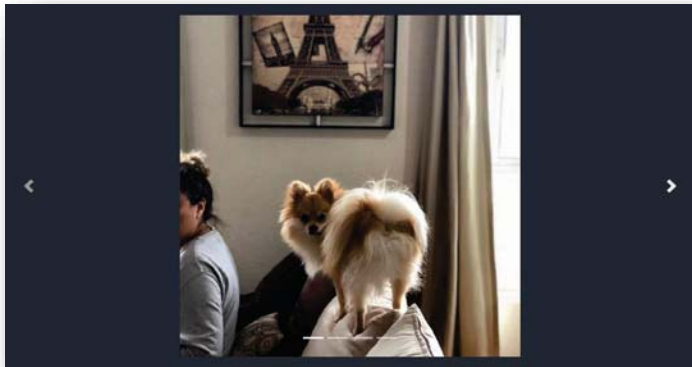


Ilustración 868 - Imagen propia

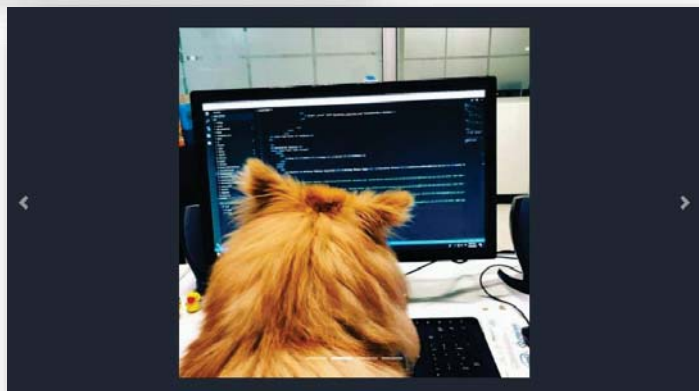


Ilustración 869 - Imagen propia

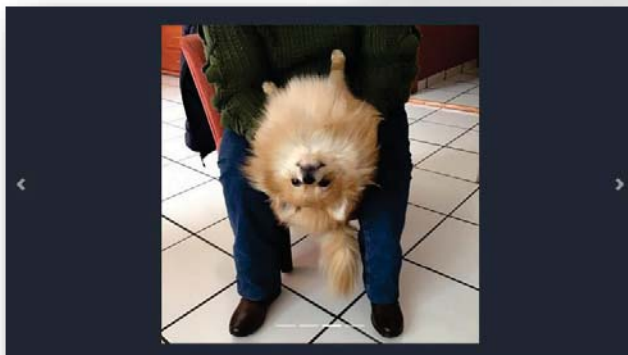


Ilustración 870 - Imagen propia

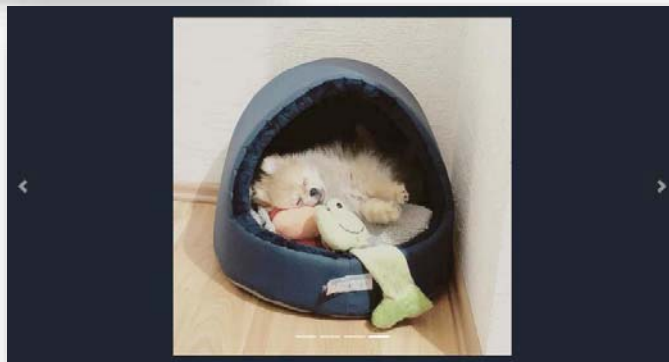


Ilustración 871 - Imagen propia

Después de todo, **Bootstrap** solo busca favorecer la manera de trabajo de un Diseñador, por ende, la facilidad que brinda, la rapidez y la buena vista que ofrece, permite no invertir demasiados recursos, en definitiva, cumple con su objetivo.

No obstante, ese Framework no sería posible, sino existirán los elementos principales, de los que, por supuesto está basado **Bootstrap**, **HTML5** brinda todo, pues sin ese lenguaje no se tendría un cimiento sólido donde enganchar la información, **CSS** termina por pulir aquellos detalles toscos y poco favorecedores de **HTML5**, aunque **JS** termina por poner un toque final.

No se desmerita ningún papel dentro de este proyecto, pues todos son esenciales, todos son importantes, todos buscan un fin en común, y por ello, se puede lograr lo menos impredecible, quizás todas las páginas o sitios web se vean similares, pues todas parten del mismo punto, todo varía dependiendo del diseño, de la orientación de los elementos y claro está, del contenido que alimenta al Desarrollo Web.

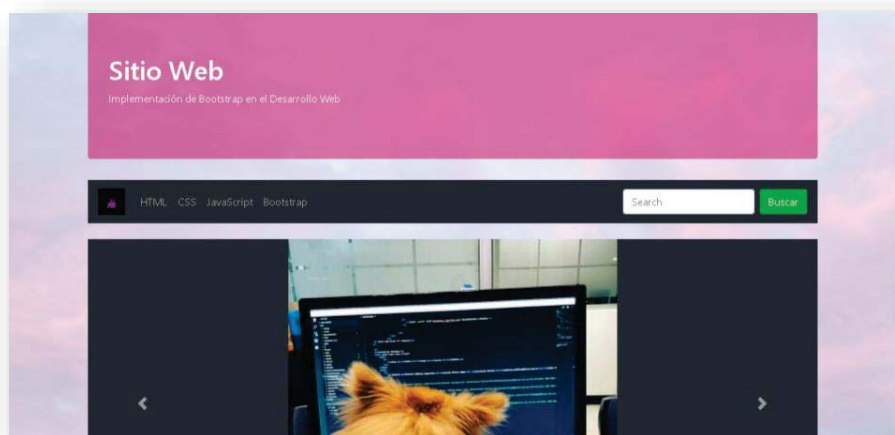


Ilustración 872 - Imagen propia

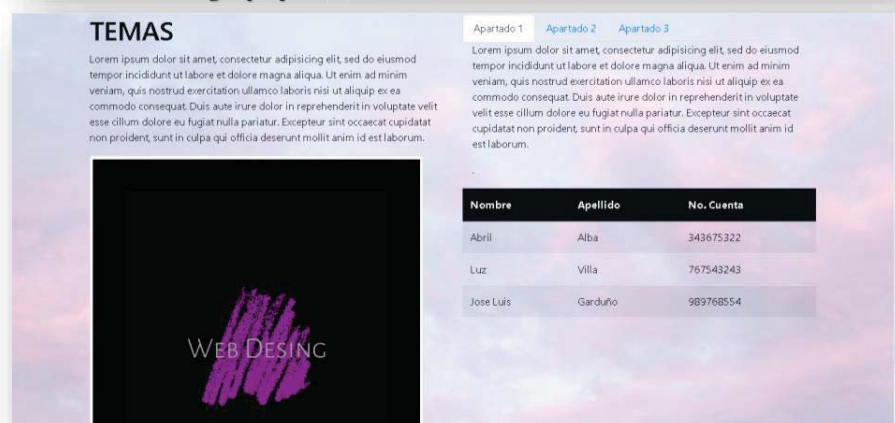


Ilustración 873 - Imagen propia

11. CASO PRÁCTICO - APLICACIÓN DE BOOTSTRAP EN EL DESARROLLO WEB

Para poder generar una perspectiva más explícita de lo que se trata el desarrollo y diseño web, basta con comenzar a implementar desde cero una página web, con una sola página, se encuentra con el gran problema de todos, la vista para el público.

En resumidas cuentas, cuando un cliente llega y dice “Necesito un sitio web que sea agradable a la vista, muy innovador, moderno y llamativo, que funcione muy bien y que sea rápido de cargar”, para un desarrollador comienza un nuevo reto, y esto acontece de estos tiempos, este problema se ha generado de mucho tiempo atrás.

Como se mencionó en los temas anteriores, la acelerada evolución de la tecnología trajo consigo la ambición de generar herramientas más interesantes de consumir, por ende, todo momento se solicitan y crean sitios dentro de la internet que generen un nuevo asombro para la audiencia.

Es muy importante resaltar que, para la mayoría de los desarrolladores web, crear una página o incluso un sitio web, no genera mucho problema, pues la funcionalidad interna, es más íntima, es decir, las funciones y la programación que existe detrás de la interfaz, depende solamente del desarrollador, por lo tanto, nadie critica esa parte, pues mientras realice su correcta función no existirá mayor problema.

Pero ¿Qué pasa cuando se trata de detallar un sitio web?, Bueno, ese si es un dolor de cabeza, pues tienen que sumergirse en los temas de color, forma, pulido de imágenes, decoración de diseño, organización, vista temática, etc. Existen muchos factores que determinan el gusto del cliente, y para la gran mayoría de los desarrolladores significa mucha fatiga, problemas y dolores de cabeza.

Sin embargo, existe una minoría que se dedica a esos detalles, personas que dedican su conocimiento y trabajo en transformar una página web simple en algo muy novedoso, y las herramientas con las que se va contando con el pasar del tiempo, mejoran aún más la posibilidad de obtener un resultado más impresionante.

Hoy en día se cuenta con muchas herramientas que permiten hacer la vida más fácil para el desarrollo web sin duda, pero el objetivo de este trabajo es para las personas que comienzan de cero y no tienen ni idea de cómo hacerlo, y al involucrarse y estar empapado de todo lo que conlleva realizar una página web, de esa manera está asegurado su brinco a un conocimiento más avanzado.

En las siguientes imágenes se mostrará un sitio web, inspirado en este trabajo, donde se involucra más el acierto de verlo en pantalla en vivo y a todo color, porque de eso se trata, la línea de este trabajo se enfoca en enseñar cómo se construyen las páginas web, como se puede escalar a generar un diseño y cuando ya se entiende ese punto, saltar de conocer una herramienta que facilita el trabajo de diseño, pero, sobre todo, enseña a valorar el tiempo y trabajo que se invirtió anteriormente, pues si no se tiene una idea de cómo funciona, no se sabe lo que se está creando.

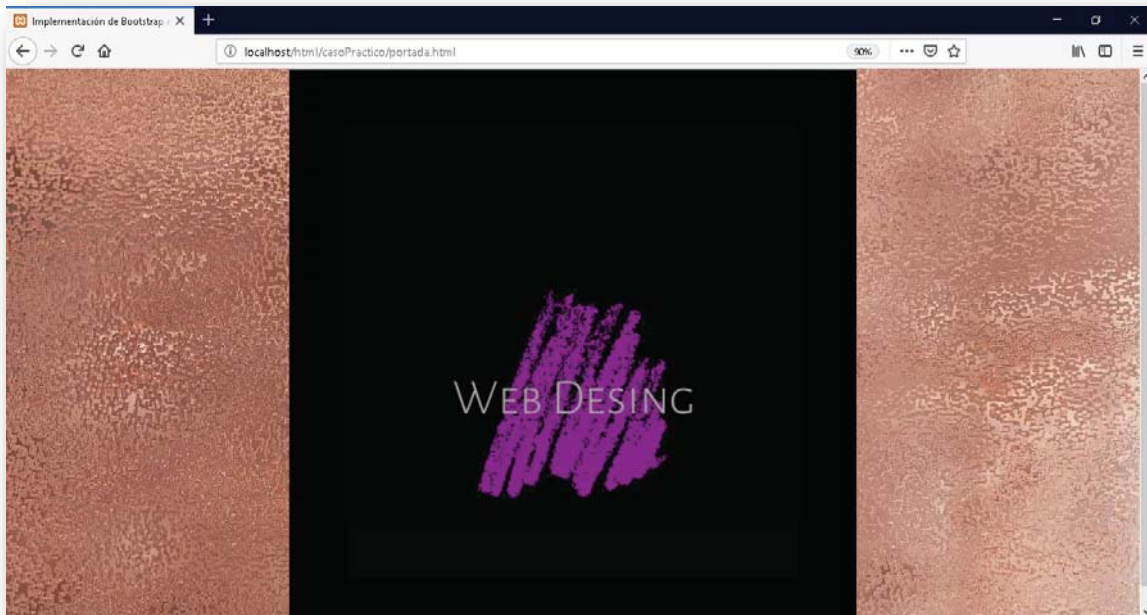


Ilustración 874 - Imagen propia



Ilustración 875 - Imagen propia

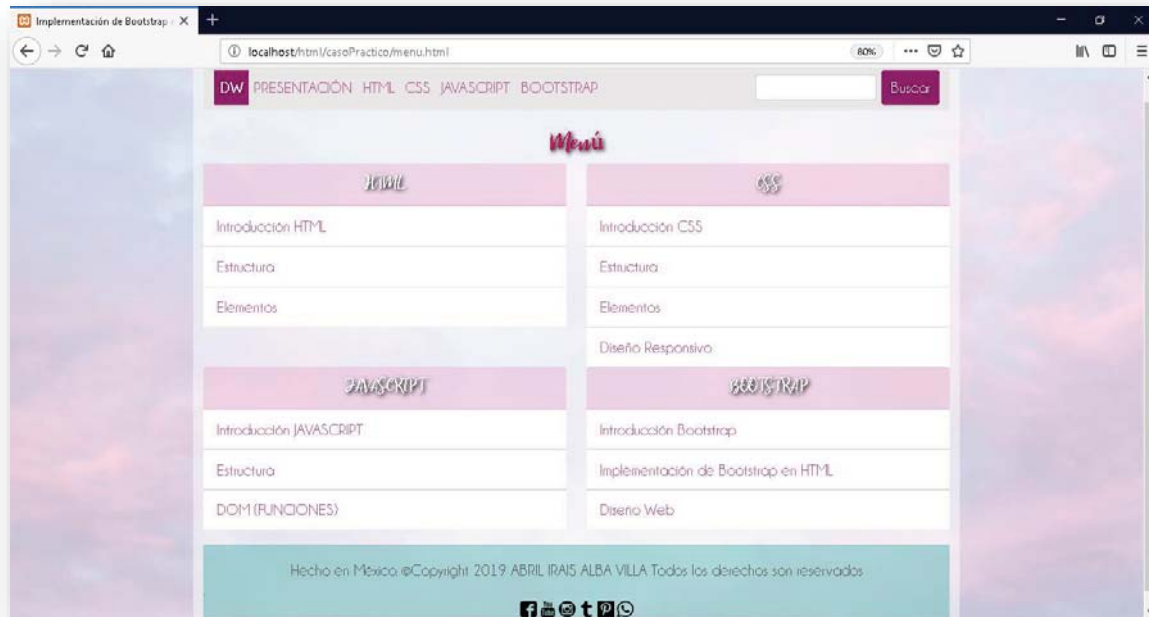


Ilustración 876 - Imagen propia



Ilustración 877 - Imagen propia

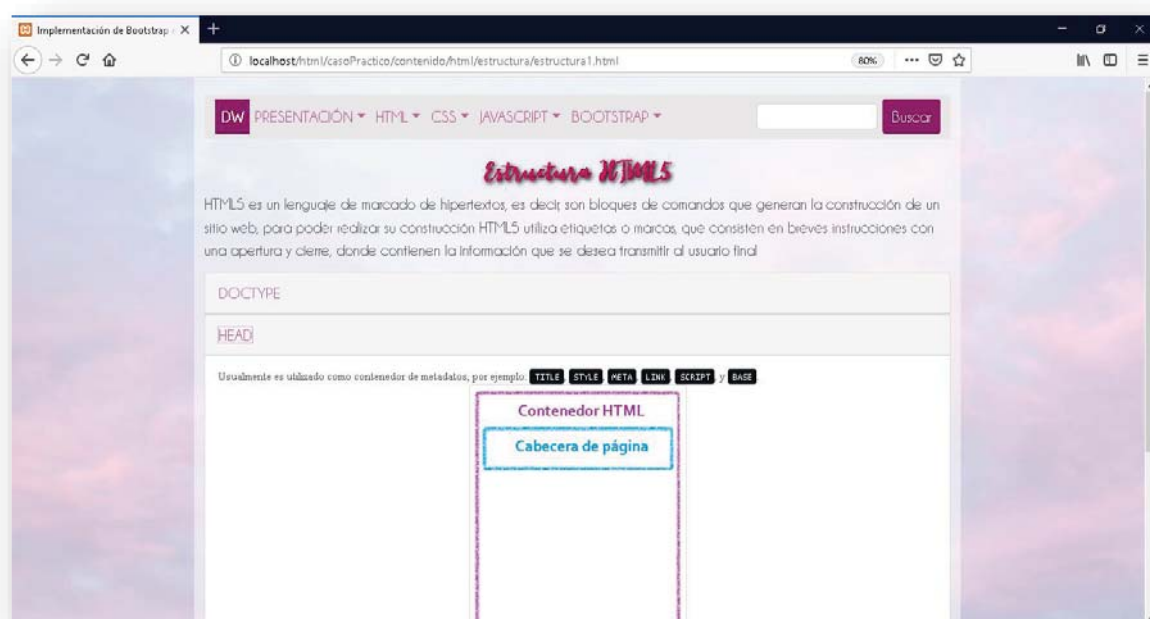


Ilustración 878 - Imagen propia

12. ANÁLISIS DE RESULTADOS

En fines prácticos, el objetivo de este trabajo se cumplió exitosamente, como se pudo conocer en todo el trabajo, se implementó un conocimiento sólido y concreto de la creación de una página web, porque involucró información teórica donde se buscó otorgar conocimiento general, también se encuentra información ilustrativa y descriptiva de la construcción y diseño de una página web, pero sobre todo culminar con tal de entender porque existen nuevas herramientas que facilitan el trabajo, como utilizarlas y explotarla para beneficio del diseñador.

Si no se entendiera la importancia de las herramientas como los Frameworks, no se cumpliría con el objetivo, puesto que no se tendría en mente lo que significa crear desde cero y entender las funciones reales de cada elemento que involucra el mundo del desarrollo web, de esa manera cuando se utilizan las herramientas se aprecia mucho más la facilidad de trabajo y el buen resultado que se obtiene.

CHECK-LIST DE CONOCIMIENTOS IMPLEMENTADOS		
	SI	NO
Antecedentes teóricos	X	
Teoría de elementos	X	
Explicación clara del tema	X	
Ejemplos ilustrativos	X	
Descripción clara de ejemplos	X	
Abarca todo el tema	X	

Ilustración 879 - Tabla propia

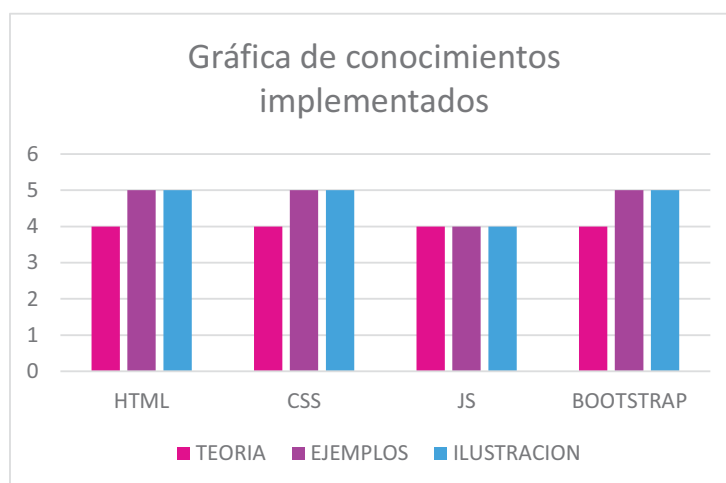


Ilustración 880 - Gráfica propia

13. CONCLUSIÓN

Al inicio de este trabajo, se planteó la problemática que se tiene con el Diseño web, pues se carece de interés e iniciativa educativa, por parte de docentes y alumnos, de modo que se presentó la oportunidad de exponer, una manera diferente de manejar el tema, con un método de investigación, que tuvo como fin, darle un giro y dirección diferente a una propuesta de aprendizaje al público interesado.

Para demostrarlo, se expuso la idea a muchas personas dentro de la carrera en informática, sobre el Diseño web y la importancia que se tiene dentro del conocimiento general de la carrera, en su gran mayoría se obtuvieron comentarios negativos sobre el tema, pues para todos es un tema, que debe ser autodidacta, es decir, que quien quiere aprenderlo, solo tiene que buscar en internet y desarrollarlo, pues no es algo que les quite el sueño.

Por otro lado, también se expuso la idea en el ámbito laboral, específicamente en el departamento de informática de diferentes dependencias, en donde su principal labor es el desarrollo de plataformas digitales, para sorpresa, el gran interés de su trabajo y culminación de un proyecto profesional, es la apariencia y el dinamismo que tenga el sitio, por lo tanto, es demandante la funcionalidad correcta de la página, así como el desempeño y diseño que lo complementan.

Por lo cual, se concluye que, en el ámbito laboral, es indispensable conocer e implementar el Diseño web, así como la función que deba desempeñar la página web, por lo tanto, es imperativo que los jóvenes se forjen un conocimiento sólido, amplio y concreto de lo que implica todo el Desarrollo web. Aunado que este trabajo brinda la posibilidad de obtener lo necesario para poder comenzar con un trabajo donde te permita explotar y experimentar aún más en conocimientos y experiencias.

Algo aún más importante, y que da un cambio a lo que se conocía como Desarrollo web, es que la implementación de las nuevas tecnologías también ya es necesario, de hecho, desarrollar paginas o sitios web con HTML5 puro, es completamente obsoleto, pues los Frameworks que se tienen hoy en día, son las herramientas principales de cada Desarrollador Full-Stack del mundo laboral, por lo cual también se concluye que este trabajo cubre las necesidades esenciales de las solicitudes laborales y cumple con los objetivos y la hipótesis que se planteó para construir este trabajo.

Por último, solo resta aclarar, que este trabajo fue motivado por la necesidad de cubrir una carencia personal, pues la falta de conocimiento e interés, propicia a tener problemas reales fuera de la escuela, dentro de nuestra carrera, nunca se dejara de aprender, pues nos encontramos en constante actualización de las buenas nuevas que surgen, por ende, es de gran importancia, que la investigación e implementación de la información sea constante y sobre todo compartida con los demás, porque de eso se trata nuestra carrera, sumarnos entre todos, ya que cualquier camino que se elija dentro de la informática, nos lleva a descubrir y trabajar en cosas nuevas y benéficas para todos.

14. BIBLIOGRAFÍAS

HTML

- DesarrolloWeb. (2019). HTML. 2019, de DesarrolloWeb Sitio web: <https://desarrolloweb.com/html/>
- Editorial Vértice. (2009). HTML. En Técnicas avanzadas de diseño web (192). España: Vértice.
- Hernández, J. (2014). Análisis y Desarrollo Web.
- Hernández, U. (2019). HTML. 2019, de CódigoFacilito Sitio web: <https://codigofacilito.com/cursos/HTML5>
- UNIWEBSIDAD. (2019). HTML. 2019, de UNIWEBSIDAD Sitio web: <https://uniwebsidad.com/libros/xhtml>
- W3C. (2018). Estándares de desarrollo y diseño web. ABRIL 2018, de W3C Sitio web: <https://www.w3.org/standards/webdesign/>
- W3Schools. (2019). HTML. 2019, de W3Schools Sitio web: <https://www.w3schools.com/html/default.asp>

CSS

- DesarrolloWeb. (2019). CSS a fondo. 2019, de DesarrolloWeb Sitio web: <https://desarrolloweb.com/css/>
- Hernández, Luis. (2019). CSS. 2019, de CódigoFacilito Sitio web: <https://codigofacilito.com/cursos/css>
- UNIWEBSIDAD. (2019). CSS. 2019, de UNIWEBSIDAD Sitio web: <https://uniwebsidad.com/libros/css>
- W3Schools. (2019). CSS. 2019, de W3Schools Sitio web: <https://www.w3schools.com/css/default.asp>

JAVASCRIPT

- DesarrolloWeb. (2019). JavaScript a fondo. 2019, de DesarrolloWeb Sitio web: <https://desarrolloweb.com/javascript/>
- Hernández, U. (2019). Curso profesional de JavaScript. 2019, de CódigoFacilito Sitio web: <https://codigofacilito.com/cursos/javascript-profesional>
- Sánchez, M. (2012). JavaScript. España: Innova.
- UNIWEBSIDAD. (2019). Introducción a JavaScript. 2019, de UNIWEBSIDAD Sitio web: <https://uniwebsidad.com/libros/javascript>
- W3Schools. (2019). JavaScript. 2019, de W3Schools Sitio web: <https://www.w3schools.com/js/default.asp>

BOOTSTRAP

- DesarrolloWeb. (2019). Framework. 2019, de DesarrolloWeb Sitio web: <https://desarrolloweb.com/wiki/framework.html>
- León, N. (2019). Curso gratuito de Bootstrap 4. 2019, de CódigoFacilito Sitio web: <https://codigofacilito.com/cursos/bootstrap>
- W3Schools. (2019). Bootstrap 4. 2019, de W3Schools Sitio web: <https://www.w3schools.com/bootstrap4/default.asp>