

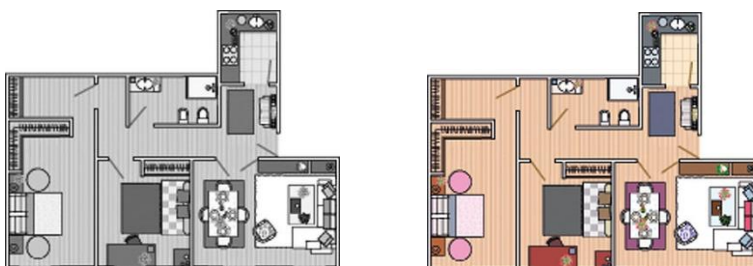
# UNIDAD 1. Fundamentos

## 1.1 Introducción

En esta guía se aprenderán sobre lo que es HTML y CSS. Las diferentes cosas que se pueden hacer con estos dos lenguajes de diseño web. Hay que tener en claro que estos tipos de lenguajes para la web son, como mencione antes, diseño web, no lenguajes de programación.

Por un lado, HTML es el lenguaje que nos da la estructura y la definición del contenido, por ejemplo, con HTML vamos a indicarle al navegador que esto es una imagen, un video, un párrafo, etc., por otro lado, CSS es un lenguaje de hojas de estilo, que es el que le da el aspecto al contenido. Con CSS podemos darle un color de fondo, cambiar el tamaño de fuente de algún texto e incluso distribuir de una mejor forma el contenido de una página web.

Hagamos una analogía: supongamos que una casa es un sitio web, la estructura y la definición que indicara que este ambiente es la sala o una cocina, será dada por HTML y la presentación como el color de las puertas y las paredes estaría dado por CSS.



Como podemos ver, estos dos lenguajes se complementan de una forma fantástica, pero cabe recalcar que son completamente diferentes, cada uno de ellos tiene su propia sintaxis y su propia forma de trabajar. En este curso se aprenderá desde cero como usar correctamente los lenguajes y las mejores prácticas que se llevan a cabo para realizar un sitio web.

## 1.2 ¿Cómo funciona la web?

En esta sección se verá cómo funciona la web y como HTML interviene en este proceso.



Lo primero que debemos saber, es que la web, se basa en consultas que hace una comunicación llamado HTTP o Protocolo de transferencia de Hipertexto. Todo inicia desde un navegador. Para acceder a un sitio web se escribirá en el navegador la dirección web, para este ejemplo se usará el sitio web del COJ: <http://www.coj.uci.cu/index>. Esta dirección, conocida como URL está dividida en algunos segmentos que dan información y ubicación del recurso que se consulta. Sus partes son las siguientes:

- **Negro:** El protocolo que se usa, en este caso HTTP.
- **Azul:** Nombre del host que hace la petición.
- **Naranja:** El recurso que se consulta antepuesto por una barra o "slash".

Con la URL el navegador realiza una solicitud y es enviada al servidor, es de esta forma que recibe la solicitud y procesa la información usando un lenguaje de programación, sea: Python, Ruby, PHP, Java, entre otro. En este procesamiento se puede dar el caso de que se consulte una base de datos, entre las más populares son: MySQL, PostgreSQL, Oracle o Microsoft SQL Server.

Una vez que el servidor está listo, da una respuesta, al navegador por lo general en el formato HTML que más que es más que nada texto plano. Una vez que el navegador recibe esto, internamente hace una representación en forma de "árbol" de cada elemento que se encuentra en el documento de HTML y posteriormente muestra el contenido al cliente.

Si solamente se mostró texto, se puede hacer uso de CSS para que cambie el aspecto de la página web, se pueda agregar colores, cambiar tipo de fuentes y estructurar contenido.

## 1.3 Flujo de trabajo

¿Cómo es el flujo de trabajo al momento de desarrollar un sitio web? En primer lugar, tenemos la necesidad de la empresa en que se trabaje o de un cliente, si sé es "**freelance**" o de nosotros mismos si estamos haciendo un proyecto personal.

Posteriormente ya que tenemos la necesidad, se piensa en la experiencia de usuario que se le va a dar a las personas que visitarán el sitio web. Para ello, existen una serie de pasos que llevan a cabo para desarrollar esta parte, que en este curso no se abordará.

Una que ya se tenga la experiencia de usuario que se llevará a cabo, se pasará a un documento, puede ser un papel, un programa de software o cualquier otro.

Ya teniendo lo que se va a implementar, se comenzará a "codear" y en esta parte hay que diferenciar los dos tipos de desarrollo que hay en un desarrollo web, uno es el **front-end** y el otro es el **back-end**.

- **Front-end:** Es la parte visual que ve el usuario y esto lo logramos haciendo uso de HTML, CSS y JavaScript.
- **Back-end:** Que utiliza los lenguajes que trabajan de lado del servidor, aquí tenemos a Python, Ruby, PHP, entre otros.

Estos son los pasos que hay que seguir al desarrollar sitios web. En este curso se cubrirán los lenguajes HTML y CSS.

# UNIDAD 2. HTML5

## 2.1 Sintaxis

La sintaxis, es la forma visible de un lenguaje, por ello, antes de crear nuestro proyecto HTML tenemos que tener en consideración lo siguiente:

### DOCTYPE

El DOCTYPE es una instrucción que va al inicio de nuestro documento HTML y que permite al navegador saber qué tipo de versión de HTML se está utilizando, esta información determinará la manera que se procesa el documento. Un **DOCTYPE** distinto puede provocar cambios en la visualización del documento. Con HTML5, que es la versión que se usará en este curso, el DOCTYPE es el siguiente: `<DOCTYPE html>`.

### Elementos

Los elementos nos ayudan a estructurar y dar significado a las partes de un documento html, se pueden crear encabezados, párrafos, textos con énfasis, subrayados o cursivas, listas de elementos, tablas, imágenes, formularios y más. Para definir y entender la estructura de un elemento debemos revisar los conceptos: contenido, modelo del contenido, texto de contenido, etiqueta y atributo. Veamos un ejemplo para entender mejor:

#### HTML

```
<p id = "parrafo">Contenido del elemento p</p>
```

- *Contenido*: El contenido de un elemento, puede ser caracteres, comentarios u otros elementos como se muestra a continuación:

#### HTML

```
<p id = "parrafo">Contenido del elemento p</p>
<p id = "parrafo"><!--Comentarios--></p>
<p id = "parrafo"><strong>Elementos</strong></p>
```

- *Modelo de contenido*: Define la estructura de un elemento si puede o no tener contenido, así como los atributos que puede o no tener. Un elemento, no puede contener atributos que no se encuentren en su modelo de contenido.
- *Contenido del texto*: Es el valor de atributo id del elemento. Este atributo se aprenderá más adelante en el curso de JavaScript.
- *Etiquetas*: Son utilizadas para delimitar el inicio y el fin de un elemento, por ello existen etiquetas de inicio y etiquetas de cierre `<p> ... </p>`.
  - *Etiquetas de inicio*: Consisten de la siguiente forma en el siguiente orden: un símbolo menor que, seguido el nombre de la etiqueta, seguido de uno o más atributos y el símbolo de mayor que.

#### HTML

```
<p id = "parrafo">Contenido del elemento p</p>
```

- *Etiquetas de cierre*: Consisten de las siguientes partes en el siguiente orden: un símbolo de menor que, seguido de un "slash", seguido del nombre de la etiqueta y el símbolo de mayor que.

#### HTML

```
<p></p>
```

**\*\*NOTA:** Cabe mencionar que las etiquetas no distinguen entre mayúsculas y minúsculas.

- *Atributos*: Siempre se escriben dentro de la etiqueta de inicio y contienen un valor.
  - *Atributos vacíos*: Son aquellos que se escriben dentro de la etiqueta con solo el nombre del atributo o el nombre del atributo seguido de un signo igual y dos comillas sin ningún valor:

#### HTML

```
<p id> o <p id="">
```

- *Atributos sin comillas*: Se puede indicar el valor sin uso de comillas siempre y cuando no contengan espacios en blanco:

#### HTML

```
<p id = parrafo>
```

- *Atributos con comillas simples y comillas dobles*: Se puede escribir el valor del atributo con comillas dobles o con comillas simples.

#### HTML

```
<p id = "parrafo"> o <p id = 'parrafo'>
```

**\*\*NOTA:** Cabe recalcar que existen elementos vacíos y no vacíos. Un elemento vacío es aquel que no le permite tener ningún contenido sin embargo puede tener atributos es por ello que este elemento solo tiene una etiqueta que es la de inicio, por ejemplo:

- `<br>`
- `<img src = "imagen.png">`

Una etiqueta no vacía es aquella que tiene una etiqueta de inicio y una de cierre, pudiendo o no tener contenido, por ejemplo:

- `<p>Un elemento no vacío</p>`
- `<span></span>`

## Comentarios

A veces, es necesario escribir comentarios en nuestro código, para facilitar la comprensión o entendimiento del mismo sin que sea visible para los navegadores, esto es posible con HTML. Un comentario consiste con un delimitador de inicio y uno de cierre, por ejemplo:

HTML

```
<!--Esto es un comentario-->
```

De la forma que se muestra en el ejemplo indicamos un comentario en HTML. Su delimitador inicial es `<!--`, seguido del comentario a escribir y cerramos con `-->`.

## 2.2 Estructura básica

Anteriormente se mencionó la sintaxis de HTML, en esta sección se aprenderá la estructura básica de un documento HTML, esa estructura contiene la declaración `<DOCTYPE html>` y algunos elementos más.

De aquí en adelante se realizarán algunas actividades para ser posteriormente realizadas, para ello se usará un editor de código para mayor comodidad para el estudiante y de esa manera hace uso de una herramienta de desarrollo.

Como recomendación para el estudiante se usará **Sublime Text 3** y un navegador, que será **Google Chrome**.

**\*\*NOTA:** Existen otros editores de código diferentes de Sublime Text, como lo son: Atom, Visual Code Studio, Notepad++, etc. De igual manera con los navegadores, existen muchos más, como son: FireFox, Opera, Microsoft Edge, etc.

Empecemos a ver en que consiste una estructura básica de HTML:

#### HTML

```
<DOCTYPE html>
<html>
<head>
  <title></title>
</head>
<body>

</body>
</html>
```

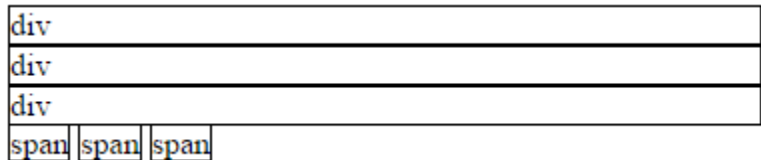
1. **DOCTYPE**: Como se mencionó antes, es para indicar la versión de HTML que se está usando y el que navegador lo interprete así, en este caso HTML5.
2. **<html>**: Encontramos esta etiqueta, con su inicio y final, este elemento es considerado como elemento raíz ya que contendrá a todos los demás elementos.
3. **<head>**: Contiene información sobre el documento que generalmente no es mostrado en el navegador, excepto la etiqueta **<title>** cuyo contenido se muestra en la pestaña del navegador.
4. **<body>**: Aquí se da forma al documento, contiene todos los demás elementos que se mostrarán en el navegador.

## 1.3 Elementos estructurales

En esta sección, se mencionarán los elementos estructurales que permiten ordenar y agrupar el contenido de una página web.

En HTML4 contiene dos elementos genéricos que son **<div>** y **<span>**. El elemento **<div>** sirve para poder agrupar secciones de contenido en bloque y el elemento **<span>** es un contenedor que sirve para agrupar contenido en línea. Se mostrará un ejemplo para ver la diferencia de contenido en bloque y contenido en línea:

#### WEB



```
div
div
div
span span span
```

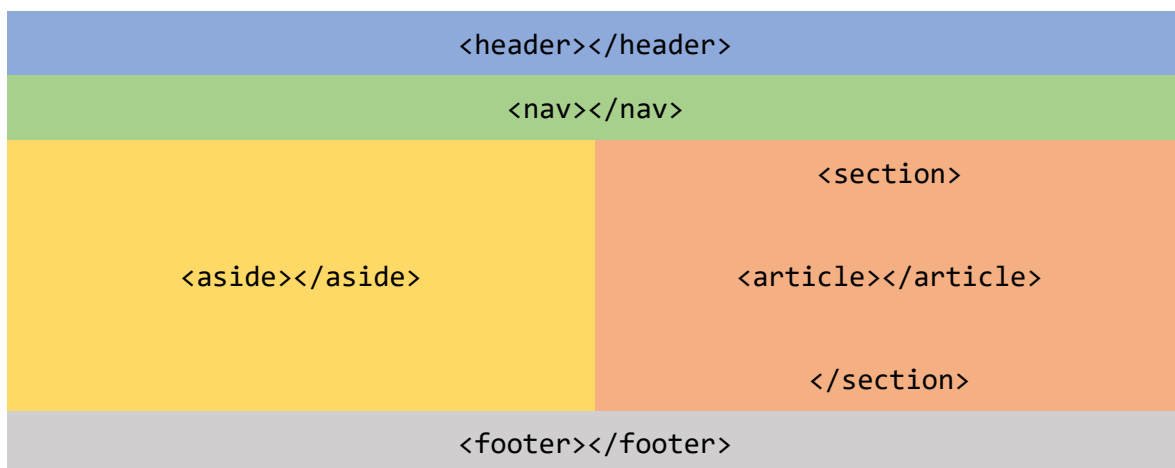
En ejemplo anterior vemos los elementos **<div>** y los elementos **<span>**. Como se ve el ejemplo, los elementos **<div>** se ubican uno debajo del otro abarcando todo el ancho de la página web, así mismo vemos como los elementos **<span>** se van ubicando uno delante del otro sin ocupar todo el ancho del navegador.

Así como los elementos anteriores, existen más elementos que por defecto tienen **display** en bloque, como son: `<header>`, `<section>`, `<article>`, `<aside>`, `<footer>`, y otros más. También existen más elementos que por defecto tienen **display inline**, tales como el elemento `<a>`, `<img>`, `<td>`, entre otros.

**\*\*NOTA:** La propiedad **display** podría variar depende al navegador.

Cambe mencionar que cada elemento `<div>` y `<span>` no tienen semántica, esto quiere decir que la única forma para identificarlos es agregada un atributo **class** o **id**.

Nosotros podemos estructurar una página web usando estos atributos y haciendo uso del atributo **id** para identificar cada uno de ellos, pero en este curso se enseña HTML5 y se mostrara la estructura de una página web haciendo uso del mismo:



- **<header>**: Contiene el encabezado de la página web.
- **<nav>**: Contiene el menú de navegación u otra funcionalidad de navegación de la página web.
- **<article>**: Posee piezas individuales de contenido, como post individuales o piezas de noticias individuales.
- **<section>**: Se encarga de agrupar diferentes áreas por funciones o diferentes artículos por tema, así mismo se puede utilizar dentro del elemento **<article>**.
- **<aside>**: Representa contenido relacionada de una forma sutil a su alrededor, pero que no es fundamental para el flujo de la página.
- **<footer>**: Contiene el pie de página.

Con los elementos mencionados, se puede realizar una página web con secciones bien definidas que nos permitan hacer un trabajo más ordenado y limpio.

## 1.4 Elementos para desplegar texto

En esta sección se mencionarán los elementos que nos permiten agregar en una página web, textos como encabezados y párrafos. Pero antes se debe de tener en claro algunos conceptos.

*Elementos de frase:* Son los que añaden información estructural y semántica al texto y se comportan como elementos en línea. Dentro de estos elementos, tenemos al elemento `<em>` que añade énfasis al texto y el elemento `<strong>` que añade un énfasis más fuerte. A nivel representativo, `<em>` hará que el texto se vea en cursiva y `<strong>` en negrita.

### HTML

```
<em>Enfasis</em>  
<strong>Más enfasis</strong>
```

### WEB

*Enfasis* **Más enfasis**

*Encabezados:* Estos elementos son `<h1>`, `<h2>`, `<h3>`, `<h4>`, `<h5>` y `<h6>`, estos ayudan a describir de una forma breve los elementos que presenten, siendo `<h1>` el más importante y `<h6>` el menos importante.

### HTML

```
<h1>Encabezado 1</h1>  
<h2>Encabezado 2</h2>  
<h3>Encabezado 3</h3>  
<h4>Encabezado 4</h4>  
<h5>Encabezado 5</h5>  
<h6>Encabezado 6</h6>
```

### WEB

**Encabezado 1**

**Encabezado 2**

**Encabezado 3**

**Encabezado 4**

**Encabezado 5**

**Encabezado 6**

**\*\*NOTA:** Evita saltar niveles de encabezados para mencionar un nivel del otro, utiliza h2 después de h1, h3 después de h2, así en ese orden. Si ejemplo, h2 no tiene el tamaño que deseas, usar un estilo CSS `font-size` para cambiar el tamaño ya que el fin de los encabezados no es brindar un tamaño determinado al texto si más bien, resaltar su importancia.



*Párrafos:* El elemento `<p>` representa un párrafo de texto en el navegador, en el ejemplo siguiente se crearon dos párrafos con sus respectivos elementos `<p>`.



**\*\*NOTA:** Cabe decir que un elemento `<p>` no puede contener dentro de él ningún elemento de bloque.

## 1.5 Enlaces

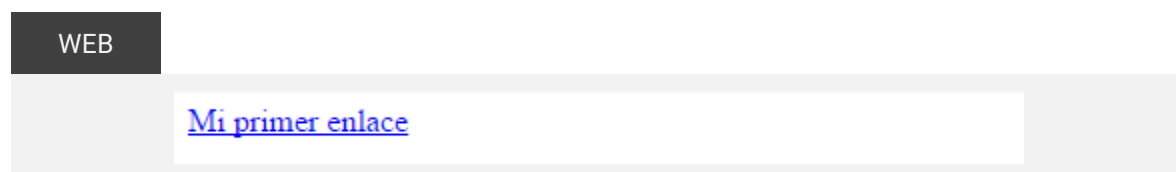
Los enlaces son importantes ya que permite moverse hacia otras páginas dentro de nuestro mismo sitio web también enlazar hacia una página de un sitio web externo, enlazar hacia ciertos elementos dentro de la misma página, enlazar a un correo electrónico, un recurso para descargar, entre otros.

Para crear estos enlaces se utilizará el elemento `<a>` también conocido como anchor que tiene la siguiente estructura:



Al inicio, se identifica una etiqueta de inicio (`<a>`) y una etiqueta de cierre (`</a>`), siguiente se identifica el atributo **href** cuyo valor es el enlace donde se direccionará al usuario para que interactúe con él y por último tendremos el texto del enlace que será lo que se visualizará en el navegador, en este caso "Mi primer enlace".

El ejemplo anterior se visualizará de la siguiente manera:



Un enlace por defecto aparece de color azul y subrayado, si el enlace ya ha sido "clicado" anteriormente aparecerá de color morado.

**\*\*NOTA:** Los colores de los enlaces pueden ser manipulados haciendo uso de CSS que se verá más adelante.

# UNIDAD 3. CSS3

## 3.1 Sintaxis

En esta sección, se aprenderá la sintaxis del lenguaje de estilos CSS. Este lenguaje se asocia mucho con HTML al momento de crear nuevos diseños. CSS tiene una sintaxis muy sencilla, está basado en reglas que se definen por selectores, propiedades y valores, tal y como se muestra en el siguiente ejemplo:

CSS

```
selector{
  propiedad1: valor1;
  propiedad2: valor2;
  propiedad3: valor3;
}
```

Se escribe el selector, seguido de un bloque de declaraciones encerrados en corchetes. Estas declaraciones a su vez tienen dos partes; una propiedad definida por unas palabras claves del lenguaje, seguida de dos puntos y un valor para esa propiedad. Existen varios valores y cada propiedad puede aceptar cada una de los distintos valores, además se tiene que tomar en cuenta que una regla tiene las siguientes características:

1. Siempre escribe las reglas en minúsculas (opcional).
2. El identificador del selector puede contener números y letras, así como guion y guion bajo.
3. El primer carácter de un identificador no puede ser un número.
4. El identificador de un selector puede contener caracteres especiales haciendo uso de la barra invertida "\".

### Propiedades

Una propiedad es una característica escrita en inglés a la que le asignamos un valor para poder cambiar el aspecto o estilo de un elemento, por ejemplo, tenemos las siguientes propiedades:

CSS


```
selector{
  color: valor;
  background-color: valor;
  font-family: valor;
  font-size: valor;
}
```

En el ejemplo tenemos el color de letra, color de fondo, tipo de letra y tamaño de letra, pero existen diferentes tipos de propiedades.

Veamos un ejemplo. Si queremos cambiar el color de letra a todos los elementos `<p>` en color rojo tendríamos que agregar una regla CSS, de la siguiente manera:

HTML	CSS
<pre>&lt;body&gt;   &lt;p&gt;Párrafo 1&lt;/p&gt;   &lt;p&gt;Párrafo 2&lt;/p&gt; &lt;/body&gt;</pre>	<pre>p{   color: red; }</pre>

Usando la regla CSS, se mostrará de la siguiente manera:

WEB


### Usar CSS dentro de un documento HTML

Dentro de un documento HTML podemos usar CSS para agregar estilos a nuestros elementos HTML, para ello existen 3 formas de poder realizarlo.

1. La primera forma de agregar un elemento CSS en un documento HTML es haciendo uso del elemento `<style>` dentro de nuestro elemento `<head>`, esta manera es la más sencilla de trabajar para agregar estilos a nuestra página, donde solo definimos el elemento `<style>` y dentro del elemento se puede comenzar a definir todas las reglas CSS.

HTML
<pre>&lt;head&gt;   &lt;style type = "text/css"&gt;     p{       color:red;     }   &lt;/style&gt; &lt;/head&gt;</pre>

2. Otra forma para añadir estilos a los elementos a HTML es hacerlo directamente al elemento con el uso de un atributo llamado `style`, de esta forma podemos agregar propiedades a un elemento específico.

HTML
<pre>&lt;h1 style = "color: blue;"&gt;¡Hola mundo!&lt;/h1&gt;</pre>

- La última manera es haciendo uso de un archivo externo con extensión **.css** y llamándolo en nuestro archivo HTML usando el elemento link dentro del elemento **<head>**.

HTML	WEB
<pre>&lt;head&gt;   &lt;link rel = "stylesheet"   type = "text/css" href =   "style.css"&gt; &lt;/head&gt;</pre>	<pre>p{   color: red; }</pre>

El atributo **rel** que indica el tipo de relación y su valor **stylesheet** que se utiliza para archivos **.css**. El atributo **type** que indica el tipo de atributo enlazado, usamos **text/css** para los archivos **.css** y el atributo **href** que es la ubicación del archivo.

Esta última forma es la que se recomienda usar, ya que con ella se tiene un archivo para todos los estilos que usará nuestra página y es más ordenado.

## 3.2 Selectores simples

Un selector representa una estructura. Esa estructura puede ser usada como una condición en una regla CSS, así mismo, un selector nos ayuda a referirnos a un elemento, clase o id, entre otros, dentro de nuestro documento HTML.

En CSS tenemos tres tipos de selectores, aquí se verán algunos de los selectores simples.

### Selector universal

El primer tipo es el selector universal, este selector aplica reglas a todos los elementos que se encuentran en un documento HTML.

HTML	CSS
<pre>&lt;h1&gt;Selector universal&lt;/h1&gt; &lt;p&gt;Esto es un ejemplo&lt;/p&gt;</pre>	<pre>*{   color: azul; }</pre>

Aquí tenemos un documento HTML, en donde se le agrega la siguiente regla: color: azul. Esto indicará que todos los textos de cada elemento se muestran de color azul.

WEB


A pesar de que tiene una forma de usar muy sencilla, es poco habitual que se use ya que es muy complicado que un mismo estilo se pueda aplicar a todos los elementos de una página, sin embargo, se pueden combinar con otros selectores.

### Selector de tipo

Este selector aplica los estilos CSS al elemento que nosotros elijamos, además, este selector se puede aplicar a múltiples elementos a la vez.

#### HTML

```
<h1>Selector universal</h1>
<p>Esto es un ejemplo</p>
```

#### CSS

```
p{
    color: orange;
}
```

Aquí se indica que solo el o los elementos `<p>` tendrán color de texto naranja.

#### WEB

## Selector universal

Esto es un ejemplo

### Selector de ID

Un atributo que poseen todos los elementos es el `id`, este atributo nos permite definir un nombre único en el cual podremos encontrar en nuestro documento HTML. Para empezar a usar un selector de `id`, se tiene que asignar el atributo `id` a un elemento, posteriormente usar la misma regla usando el mismo nombre de `id` precedido por el símbolo de almohadilla.

#### HTML

```
<section id = "noticia">
  <h1>Selector ID</h1>
  <p>Ejemplo de como funciona
    un selector ID</p>
</section>
```

#### CSS

```
#noticia{
    color: grey;
}
```

Como resultado, todos los elementos contenido dentro del elemento `<article>` son de color gris.

## WEB

### Selector ID

Ejemplo de como funciona un selector de id

#### Selector por clase

Otro de los atributos que nos permitirá asignar estilos, es el atributo **class**, el cual podemos asignarle a cualquier elemento para que se le apliquen estilos en común, con esto, se refiere a que se pueden crear estilos con un nombre de clase y esta clase podemos asignarla a cualquier elemento.

## HTML

```
<section>
  <h1 class = "clase">
    Selector ID</h1>
  <p>Ejemplo de como funciona
    un selector ID</p>
</section>
```

## CSS

```
.noticia{
  color: violet;
}
```

Ahora, como se puede ver el color de texto de `<h1>` afectado por la clase es de color violeta.

## WEB

### Selector de clase

Ejemplo de como funciona un selector de id

De esta forma, con los selectores nos ayuda a hacer más específicos para asignar un estilo CSS a un elemento.

## 3.3 Pseudoclases

El concepto pseudoclase se introdujo para permitir la selección basada en información que se encuentra fuera de la estructura del documento o que puede ser difícil o imposible de expresar con los otros selectores simples. Una pseudoclase contiene dos puntos, seguido del nombre de pseudoclase.

## Pseudoclases basadas en el historial del enlace

Nos permiten cambiar los estilos de un enlace según si han sido visitados o no anteriormente, estas pseudoclases son `link` y `visited`.

### HTML

```
<h1>Pseudoclases</h1>
<a target = "_blank" href =
"http://itver.edu.mx"> en este
Ejemplo veremos como funciona
Las pseudoclases :link y
:visited</a>
```

### CSS

```
a:link{
    color: yellow;
}
a:visited{
    color: olive;
}
```

Como se puede ver, el estilo del enlace ha cambiado a color amarillo y posteriormente a darle click cambia a un color oliva. Esto se debe a que agregamos un estilo a las pseudoclases `link` y `visited` para el elemento `<a>`.

### WEB

## Pseudoclases

en este Ejemplo veremos como funciona Las pseudoclases :link y :visited

## Pseudoclases

en este Ejemplo veremos como funciona Las pseudoclases :link y :visited

## Pseudoclases basadas en la acción del usuario

También existe pseudoclases basadas en la acción del usuario, como se muestra en el siguiente ejemplo:

### HTML

```
<h1> Datos personales </h1>
<form>
  <fieldset>
    <legend>Datos personales</legend>
    Nombres: <input type = "text"/>
    Apellidos: <input type = "text"/>
  </fieldset>
</form>
```

Tenemos un formulario creado con el elemento `<form>`, dentro de él, el elemento `<fieldset>` que sirven para agrupar elementos relacionados dentro de un formulario y el elemento `<legend>` que define una leyenda para la agrupación hecha con `<fieldset>`.

WEB

## Datos personales

Datos personales

Nombres:

Apellidos:

Todos los elementos son de color negro como se pueden ver en la imagen anterior. Al pasar el cursor arriba de la leyenda que dice "Datos personales" o dar clic sobre ese texto se muestra de color negro, e inclusive al escribir en los cuadros de texto se muestran de color negro por defecto. Ahora, se agregarán estilos con CSS.

CSS

```
legend:hover{
    color: red;
}
legend:active{
    color: azul;
}
input:focus{
    color: orange;
}
```

Al selector `<legend>` se le agrego la pseudoclase **hover** y como regla se escribió que se muestre el texto de color rojo. También, al selector `<legend>` se le agrego la pseudoclase **active** y el color que mostrará será azul y se agrego el selector `input` con la pseudoclase **focus** como color naranja. Ahora, en la siguiente imagen se muestran los resultados:

WEB

## Datos personales

Datos personales

Nombres:

Apellidos:



Cuando el mouse se encuentra ubicado encima del elemento **<legend>** este cambia a color rojo, dado que la pseudoclase **hover** que estilo le puede brindar a un elemento cuando el cursor se encuentra encima de él.

Cuando se da click al elemento **<legend>** cambiará a color azul tiempo que tengamos precisado, al dejar de presionar el estilo de color azul desaparecerá. Esto se debe a que la pseudoclase **active** nos permite brindar estilos al elemento cuando el usuario active el elemento, en este caso lo activa haciendo clic sobre él.

**\*\*NOTA:** Las pseudoclases **hover** y **active** apuntan a cualquier elemento, no necesariamente a enlaces.

En el elemento **input**, al hacer click y escribir, el texto cambia a color naranja y al salir de campo de texto regresa a color por defecto, que es negro. Esto se debe a que la pseudoclase **focus** nos permite brindar estilos a un elemento mientras tiene el foco.

**\*\*NOTA:** La pseudoclase **foco** es permitida en elementos que aceptan eventos de teclados como los eventos de un formulario. Se puede cambiar el foco usando la tecla **Tab**.

### Pseudoclases **first-child**, **last-child** y **nth-child**

Además de las pseudoclases que se mencionan anteriormente, existen las pseudoclases, **first-child**, **last-child** y **nth-child**.

#### HTML

```
<ul>
  <li>Primer elemento</li>
  <li>Segundo elemento</li>
  <li>Tercer elemento</li>
  <li>Cuarto elemento</li>
  <li>Último elemento</li>
</ul>
```

#### CSS

```
li:first-child{
  background: yellow;
}
li:last-child{
  background: red;
}
li:nth-child(2n){
  background: orange;
}
li:nth-child(3n){
  background: olive;
}
```

Tenemos una lista con varios elementos y se le agrego un estilo CSS. Se utilizó la pseudoclase **first-child** con **background** de color amarillo, posteriormente un la pseudoclase **last-child** de color rojo. Y la pseudoclase **nth-child** con notación **2n** y **3n** con color naranja y oliva.

- Primer elemento
- Segundo elemento
- Tercer elemento
- Cuarto elemento
- Último elemento

Con la pseudoclase **first-child** se indicará que el primer elemento hijo de una clase padre, en este caso **<ul>** tendrá color amarillo. La pseudoclase **last-child**, indica que el último hijo tendrá color rojo y la pseudoclase **nth-child** con notación  $2n$  y  $3n$ , estas pseudoclases nos permiten afectar a todos los elementos pares e impares respectivamente. Como se ve en la imagen anterior, el segundo y cuarto elemento en la lista de hermanos tiene fondo naranja y el tercer elemento tiene fondo oliva. Esto se debe a que la pseudoclase **nth-child** con notación **an+b** representa un elemento que tiene una posición **an+b** dentro de una lista de elementos hermanos, donde  $n$  es un entero mayor o igual que 0.

En esta sección se mostraron las diferentes pseudoclases que ayudan a brindar estilos CSS según estado o acción en un elemento.

### 3.4 Pseudoelementos

En CSS3 un pseudoelemento está formado por doble dos puntos (**::**) seguido del nombre de pseudoelemento. Los pseudoelementos te permiten acceder a información del documento que no es accesible de otra manera, tales y como lo hacen los pseudo elementos **first-letter** y **first-line**. Así mismo, pueden ayudar a los autores a referirse a contenidos que no existen en el documento tales como lo hacen los pseudoelementos **after** y **before**.

#### HTML

```
<h1 class = "titulo">Pseudo
Elementos</h1>
<p>
  Lorem ipsum dolor sit amet,
  consectetur adipiscing elit. Proin
  tincidunt faucibus dictum. Donec ex
  enim, laoreet eget congue at,
  semper eget ante.
</p>
```

#### CSS

```
body{
  background: #2c3e50;
  margin: 30px 30px;
  font-family: trebuchet ms;
  color: white;
}
.titulo{
  text-align: center;
}
```

WEB

# Pseudo Elementos

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Proin tincidunt faucibus dictum. Donec ex enim, laoreet eget congue at, semper eget ante.

Tenemos un encabezado `<h1>` seguido de un párrafo con algunos estilos definidos. Se le agregará un estilo más en CSS:

CSS

```
h1::before{
  content: "Before";
  background: orange;
}
```

Se agregó un estilo al elemento `<h1>` con el pseudo elemento `::before`, donde del estilo se agregó un **content** con el valor de "Before".

WEB

# BeforePseudo Elementos

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Proin tincidunt faucibus dictum. Donec ex enim, laoreet eget congue at, semper eget ante.

En el navegador, del lado izquierdo del elemento `<h1>` aparece el texto "Before" con su respectivo fondo de color naranja.

Posteriormente, se agregará otro estilo CSS al mismo selector `<h1>`, seguido esta vez de otro pseudoelemento.

## CSS

```
h1::after{  
  content: "After";  
  background: orange;  
}
```

Se muestra en el navegador que de lado derecho del elemento `<h1>` aparece el texto "After" con fondo de color naranja.

## WEB

**Before**Pseudo Elementos**After**

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Proin tincidunt faucibus dictum. Donec ex enim, laoreet eget congue at, semper eget ante.

Esto se debe que a estos pseudoelementos pueden ser utilizados para describir contenido antes y después de un elemento respectivamente.

Se agregará un estilo más, ahora al elemento `<p>` con el pseudo elemento **first-letter** con la propiedad **font-size** y de color dorado.

## CSS

```
p::first-letter{  
  font-size: 36px;  
  color: gold;  
}
```

El navegador se muestra que la primera letra del elemento `<p>`, es decir, en este caso la "f" mayúscula, adopto los estilos mencionados. Esto se debe a que **first-letter** representa al primer carácter del texto de un elemento.

## WEB

# BeforePseudo ElementosAfter

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Proin tincidunt faucibus dictum. Donec ex enim, laoreet eget congue at, semper eget ante.

Por último, se agregará un estilo más para el elemento `<p>` con el pseudoelemento **first-line**, agregando un color carmesí.

## CSS

```
p::first-line{
  color: crimson;
}
```

Se puede ver en la primera línea del texto cambio a color carmesí.

## WEB

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Proin tincidunt faucibus dictum. Donec ex enim, laoreet eget congue at

Esto se debe a que **frist-line** describe el contenido de la primera línea de un elemento.

## 3.5 Combinación de selectores

En esta sección, se hablará sobre la combinación de selectores, para ello se comenzará con unos ejemplos para que se entienda de una mejor forma.

## HTML

```
<ul>
  <li>Objeto 1</li>
  <li>Objeto 2
    <ol>
      <li>Objeto anidado 1</li>
      <li>Objeto anidado 2</li>
    </ol>
  </li>
</ul>
```

```
        </ol>
      </li>
      <li>Objeto 3</li>
    </ul>
```

Se tiene una lista desordenada `<ul>` con tres objetos, cuyo segundo objeto contiene una lista ordenada `<ol>` con dos objetos. Ahora, en con CSS se creará un combinado descendente, escribiendo el primer selector `<ul>`, un espacio y por último el segundo selector `<li>` con la propiedad `margin-left: 40px`.

#### CSS

```
ul li{
  margin-left: 40px;
}
```

En la siguiente imagen se muestra que todos elementos `<li>` descendientes de `<ul>` adoptan un margen hacia la izquierda de 40 pixeles.

#### WEB

- Objeto 1
- Objeto 2
  - 1. Objeto anidado 1
  - 2. Objeto anidado 2
- Objeto 3

Ahora se cambiará de combinado por un combinado hijo utilizando el signo mayor que entre el selector `ul` y el selector `li`.

#### CSS

```
ul > li{
  margin-left: 40px;
}
```

Se muestra que solo los elementos `<li>` hijos de `<ul>` adoptan un margen izquierdo de 40 pixeles, los demás descendientes, no.

## WEB

- Objeto 1
- Objeto 2
  - 1. Objeto anidado 1
  - 2. Objeto anidado 2
- Objeto 3

Esto se debe a que con un combinado descendente se puede afectar a cualquier elemento descendente de un elemento determinado mientras que con el combinado hijo solo afectaremos a los elementos hijo.

Veamos el siguiente ejemplo:

## HTML

```
<div>  
  <p>Párrafo 1</p>  
  <p>Párrafo 2</p>  
  <div>Caja</div>  
  <p>Párrafo 3</p>  
</div>
```

Se muestra un elemento `<div>` que contiene tres elementos `<p>` y un elemento `<div>`. Con CSS se agregará un combinado hermano adyacente `p + p` con fondo amarillo.

## CSS

```
p + p{  
  background: yellow;  
}
```

Y se muestra que solo el párrafo dos tiene color de fondo amarillo.

## WEB

Párrafo 1

Párrafo 2

Caja

Párrafo 3

Se modificará el combinado hermano adyacente por un combinado hermano general, reemplazando el símbolo "+" por una tilde.

#### CSS

```
p ~ p{  
    background: yellow;  
}
```

Ahora se muestra que ya no solo el párrafo dos tiene fondo de color amarillo, ahora también el párrafo tres tiene de color amarillo su fondo.

#### WEB

Párrafo 1

Párrafo 2

Caja

Párrafo 3

Esto se debe a que en un combinado hermano adyacente el elemento hermano ubicado a la derecha debe de estar el documento inmediatamente después del elemento hermano ubicado a la izquierda, es por ello que el primer elemento `<p>` no puede ser dado que no tiene a nadie que lo preceda y el tercer párrafo tampoco, ya que lo procede un elemento `<div>` y no un elemento `<p>`.

Mientras que, con el combinado hermano general, no es necesario que un elemento hermano este inmediatamente después del otro, es por ello que el párrafo tres, a pesar de estar después de un elemento `<div>` y no de un elemento `<p>` también aparece con un fondo amarillo.

## 3.6 Colores

En esta sección se verá entre las opciones que hay para asignar colores a los elementos dentro de nuestro sitio web.

La manera más sencilla de definir colores en CSS es usar el espacio de colores sRGB o también llamado Standard red, green and blue. Los colores en un espacio de colores RGB, son formadas por combinaciones de tres canales: rojo, verde y azul; cuyo rango de valores van desde 0 a 255, las combinaciones de esos tres colores nos permiten crear millones de colores, para ser exactos permiten crear 16 777 216 colores (255x255x255). En lo que podemos encontrar el color que más nos agrade o el que más se acomode a nuestras necesidades.

Actualmente existen cuatro maneras principales de representar colores en el standard red, green and blue, dentro de CSS, estos son:



































- Keywords
- Notación hexadecimal
- RGB y HSL

## Keywords

De los keywords se debe saber que estos valores son nombres en inglés que representan colores en la vida cotidiana, tales como pink (rosado), orange (naranja), yellow (amarillo), entre otros.

Estos colores tienen su correspondiente valor en su notación hexadecimal y algo que debemos tener en cuenta es que este método para definir colores es un poco limitado en cuanto la variedad de colores, para solucionar esta limitación se tienen otras opciones.

A continuación, se verá un ejemplo para mostrar los colores keywords.

Named	Numeric	Color name	Hex rgb	Decimal
		<b>black</b>	#000000	0,0,0
		<b>silver</b>	#C0C0C0	192,192,192
		<b>gray</b>	#808080	128,128,128
		<b>white</b>	#FFFFFF	255,255,255
		<b>maroon</b>	#800000	128,0,0
		<b>red</b>	#FF0000	255,0,0
		<b>purple</b>	#800080	128,0,128
		<b>fuchsia</b>	#FF00FF	255,0,255
		<b>green</b>	#008000	0,128,0
		<b>lime</b>	#00FF00	0,255,0
		<b>olive</b>	#808000	128,128,0
		<b>yellow</b>	#FFFF00	255,255,0
		<b>navy</b>	#000080	0,0,128
		<b>blue</b>	#0000FF	0,0,255
		<b>teal</b>	#008080	0,128,128
		<b>aqua</b>	#00FFFF	0,255,255

Tenemos un elemento `<p>` que contiene texto. Texto que se muestra por defecto de color negro en el navegador.

## HTML

```
<p>Representación de colores keywords</p>
```

Con CSS se creará una regla para el selector `<p>` para asignar un color de texto rojo, con la propiedad `color` y el valor `red` y se agregará un color de fondo con la propiedad `background` y el valor `yellow`.

## CSS

```
p{
  color: red;
  background: yellow;
}
```

Como se ve, esto funciona correctamente, tal y como se visualiza en el navegador.

## Representación de colores keywords

### Notación hexadecimal

Esta notación es la más usada y se define con el símbolo almohadilla (#) seguidos de números hexadecimales que pueden ser de un valor del 0 al 9 o una letra de la A a la F, se pueden utilizar tres o seis números hexadecimales para formar el canal de color adecuado.

En una notación de seis caracteres los primeros dos caracteres representan el canal rojo, el tercer y el cuarto representan el canal verde y los dos últimos caracteres representan el canal azul.

Color hexadecimal: #00ff00

Estos pares se obtienen de la conversión de 0 a 255 en un sistema hexadecimal. Veamos un ejemplo.

Se agregará con CSS un valor hexadecimal para el elemento `<p>`, este color será: #ff0000 que representa el color rojo.

#### HTML

```
<p>Valores hexadecimales</p>
```

#### CSS

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

- Color rojo: #ff0000.
- Color verde: #00ff00.
- Color azul: #0000ff.

Con el sistema hexadecimal podemos crear millones de posibilidades.

**\*\*NOTA:** Una notación de seis caracteres puede ser escrita como tres caracteres cuando cada canal contiene un carácter repetido, ejemplo:

- #ff0000 -> #f00

Vemos el primer canal repite la "f", el segundo canal el cero y el tercer canal también el cero, podemos hacerlo una notación de tres caracteres obteniendo el mismo resultado.

### RGB y RGBA

Ahora tocan a los colores RGB y RGBA, estas notaciones nos permitirán asignar colores usando dos funciones llamadas `rgb()` y `rgba()` utilizando un formato decimal.

RGB utiliza tres canales rojo verde y azul mientras que RGBA además utiliza un cuarto canal llamado canal alfa, el cual define la opacidad que oscila entre 0 y 1, donde 0 = pixel no tiene opacidad y 1 = pixel es totalmente opaco.

Tenemos dos párrafos, uno con nombre de clase **"colorrgb"** y otro con nombre de clase **"colorrgba"**.

#### HTML

```
<p class = "colorrgb">
  Color RGB
</p>
<p class = "colorrgba">
  Color RGBA
</p>
```

Agregamos color al texto de la clase **"colorrgb"**, para ello utilizaremos la función **rgb()** con tres valores separados con comas, estos valores van desde 0 a 255.

#### CSS

```
.colorrgb{
  color: rgb(255, 0, 0);
}
```

En el navegador el primer párrafo afectado por la clase **"colorrgb"** cambio a rojo.

#### WEB

Color RGB

Color RGBA

**\*\*NOTA:** La notación hexadecimal para los tres colores primarios son la siguiente:

- Rojo: color: **rgb(255,0,0);**
- Verde: color: **rgb(0,255,0);**
- Azul: color: **rgb(0,0,255);**

Se asignará un color a la clase **"colorrgba"** utilizando la función **rgba()** que constará de cuatro valores separados por comas. Los tres valores van de 0 a 255 y el cuarto valor indica la opacidad.

## CSS

```
.colorrgb{
  color: rgb(255, 0, 0);
}
.colorrgba{
  Color: rgba(0, 0, 255, .4);
}
```

## WEB

Color RGB

Color RGBA

## HLS y HSLa

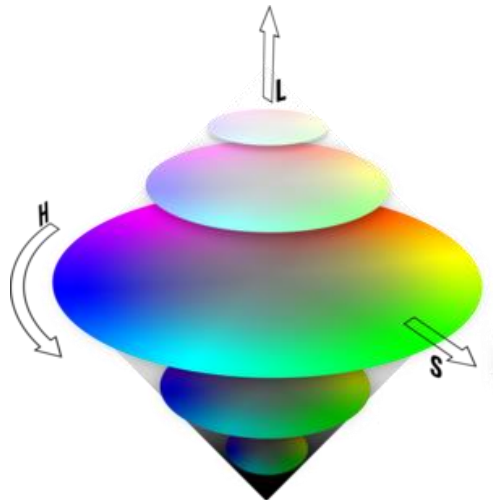
El modelo HLS se representa gráficamente como un cono doble o un doble hexágono. Los dos vértices en el modelo HSL se corresponden de la siguiente manera:

- H: El ángulo.
- S: Saturación.
- L: Luminancia.

Los colores HSL son asignados con la función `hsl()` del inglés Hue(matiz), Saturación(o saturación) y Lightness (o brillo).

Para usar esta función se necesita agregar unos valores entre los paréntesis, el primero es el matiz, este valor oscila entre 0 y 356, que puede ser mayor que a partir de 360 los matices se repiten. El segundo es la saturación y el tercero que es el brillo sus valores están basados en porcentajes entre 0% y 100%.

Tenemos dos párrafos, el primero con la clase `"colorhsl"` y el segundo con la clase `"colorhsla"`.



## HTML

```
<p class = "colorhsl">
  Color RGB
</p>
<p class = "colorhsla">
  Color RGBA
</p>
```

En CSS se agregará a la clase “**colorhsl**” un color utilizando la función **hsl()**, con los valores 0 de matiz, 100% de saturación y 50% de luminosidad.

#### CSS

```
.colorhsl{  
  color: hsl(0, 100%, 50%);  
}
```

Vemos que el primer párrafo cambia de color rojo.

#### WEB

Color HSL

Color HSLa

**\*\*NOTA:** Para los colores primarios de la notación HSL solamente varía el matiz:

- Rojo: `hsl(0, 100%, 50%)`.
- Verde: `hsl(120, 100%, 50%)`.
- Azul: `hsl(240, 100%, 50%)`.

Ahora se agregará el color a la clase “**colorhsla**” con la función **hsla()** que permite crear los primeros varios similar a la función **hsl()**, se usarán los valores anteriores, pero el cuarto valor nos ayuda conseguir el grado de opacidad, aquí tendrá 40% de opacidad poniendo el valor `.4`.

#### CSS

```
.colorhsl{  
  color: hsl(0, 100%, 50%);  
}  
.colorhsla{  
  color: hsla(0, 100%, 50%, .4);  
}
```

Estos son los resultados

Color HSL

Color HSLa

**\*\*NOTA:** Si deseas encontrar de manera rápida los colores para tus diseños se te recomiendan las siguientes páginas:

- Color Hex: <http://www.color-hex.com/>
- Flatuicolors: <http://www.flatuicolors.com/>
- Color Adobe: <http://www.color.adobe.com/>

## 3.7 Longitudes

Ahora se hablará sobre las longitudes en CSS y como estas ayudan a definir los tamaños de elementos estructurales, letras, bordes, entre otros.

Para establecer longitudes en CSS hay diferentes unidades de longitud:

- Absolutas
- Relativas

### Unidad de longitud absolutas

Estas están fijadas a relación de unidades físicas, estas unidades son centímetro (cm), milímetro (mm), pulgada (in), pixel (px), punto (pt) y pica (pc).

- 1 in = 2,54 cm
- 1 px = 1 / 96 in
- 1 pt = 1 / 72 in
- 1 pc = 12 pt

**\*\*Nota:** Si me dimos una pulgada en CSS esta no necesariamente medirá una pulgada, esta dependerá de la densidad de pixeles de la pantalla.

De estas unidades la más popular es el pixel (px), en la actualidad con la aparición de nuevos dispositivos con tamaños de pantalla diferentes los pixeles han perdido popularidad sin embargo los pixeles son confiables e ideales para empezar.

Veamos el siguiente ejemplo para ver las diferencias entre las diferentes unidades absolutas.

## HTML

```
<h1>Unidades absolutas</h1>
<div class = "pulgada">
  Pulgada
</div>
<div class = "centimetro">
  Centimetro
</div>
<div class = "picas">
  Picas
</div>
<div class = "milimetro">
  Milimetros
</div>
<div class = "punto">
  Puntos
</div>
<div class = "pixel">
  Pixeles
</div>
```

## CSS

```
div{
  height: 50px;
  font-family: Verdana;
  font-size: 16px;
  overflow: visible;
}
.pulgada{
  width: 7in;
  background: green;
}
.centimetro{
  width: 7cm;
  background: gold;
}
.picas{
  width: 7pc;
  background: red;
}
.milimetro{
  width: 7mm;
  background: blue;
}
.punto{
  width: 7pt;
  background: orange;
}
.pixel{
  width: 7px;
  background: olive;
}
```

Como se muestra, se tiene una lista de elementos `<div>`, el primero con fondo verde cuyo ancho de 7 pulgadas, luego un `<div>` de fondo amarillo con un ancho de 7 centímetros, seguido de un `<div>` con fondo rojo de 7picas, seguido de un `<div>` de fondo azul de 7 milímetros, luego se tiene un `<div>` de color naranja con un ancho de 7 puntos y por último un `<div>` de color oliva de un ancho de 7 pixeles.

## Unidades absolutas



### Unidades de longitud relativas

Estas, a diferencia de las absolutas no están completamente definidas ya que su valor esta referenciado respecto a otro valor. A pesar de su aparente dificultad, son las más utilizadas en el diseño web por la flexibilidad que se adaptan a los diferentes medios.

Entre estas unidades se tienen a em, rem, ex y ch, las cuales están relacionadas al tamaño de la fuente y de las cuales em es la más popular.

Tenemos un texto encerrado en su respectivo `<span>` con una clase, así mismo en CSS se tiene definido que `font-size` es de 40 px y esto afecta a todos los elementos contenidos dentro de él.

#### HTML

```
<html>
  <head>
    <title></title>
  </head>
  <body>
    <span class =
      "unidadrem">rem</span>
  </body>
</html>
```

#### CSS

```
html{
  font-size: 40px;
}
.unidadrem{
  color: crimson;
}
```

De esta manera se muestra con el tamaño de 40 pixeles.



## WEB

rem

Ahora, al selector de la clase unidad rem se le agregará la propiedad **font-size** igual a 4 rem.

## CSS

```
html{
  font-size: 40px;
}
.unidadrem{
  font-size: 4rem;
  color: crimson;
}
```

Ahora se muestra que el tamaño de la letra incremento su tamaño.

## WEB

rem

Esto se debe a que la unidad rem es relativa al tamaño de fuente del elemento raíz en todo momento que es el documento **html**. Dado que HTML tiene 40 pixeles como tamaño de fuente, ahora el texto rem tendrá un tamaño de fuente de 4 veces 40 pixeles.

A continuación, se mostrará otro ejemplo para profundizar en las unidades relativas em.

Se tiene dos elementos **<div>**. El primero con clase **"caja1"** y el segundo con clase **"caja2"**. El tamaño de fuente de cada uno es definido en la hoja de estilos y es de 50 pixeles.

## HTML

```
<html>
  <head>
    <title></title>
  </head>
  <body>
    <div class= "caja1">
      50</div>
    <div class = "caja2">
      <p>Tamaño de fuente
        Heredado</p>
    </div>
  </body>
</html>
```

## CSS

```
.caja1{
  background: green;
  font-size: 50px;
  height: 50px;
}
.caja2{
  font-size: 20px;
}
```

## WEB

50

Tamaño de fuente Heredado

Por ser un elemento de bloque y no tener definido un ancho se muestra que el `<div>` de color verde ocupa todo el ancho de la pantalla.

Posteriormente se agregará una propiedad `width` con `1em` a la hoja de estilos.

## CSS

```
.caja1{
  background: green;
  font-size: 50px;
  height: 50px;
  width: 1em;
}
.caja2{
  font-size: 20px;
}
```

Y se mostrar que ahora el ancho es de 50 pixeles.

50

Tamaño de fuente Heredado

Si se cambia el valor agregado anteriormente por 2em, lo que sucederá es que el ancho ahora será el doble, o sea, 50 pixeles. Esto se debe a que la unidad relativa em es equivalente al tamaño de fuente del elemento, en este caso el tamaño de fuente de 50 pixeles declarado en el **font-size**, por lo que al utilizar el valor de 2em el ancho de la caja cambia su tamaño a 100 pixeles.

### 3.8 Porcentajes

Esta sección se hablará sobre los porcentajes en CSS y como estos nos ayudan a cambiar la dimensión de los elementos estructurales, los tamaños de fuente, entre otros.

Los porcentajes se dejaron en un tema a parte ya que técnicamente no es una longitud a pesar de estar muy relacionados. Un porcentaje, consiste en un número seguido del símbolo porcentaje (%) cuyo valor es siempre relativa a otro valor.

Numero %

Se muestra el siguiente ejemplo en donde se tienen 3 elementos `<div>` uno dentro de otro. El primer elemento `<div>` contiene un ancho y alto de 400 pixeles, así como un tamaño de fuente de 80 pixeles.

#### HTML

```
<div class = "level1">
  <div class = "level2">
    <div class = "level3">
      20px
    </div>
  40px
</div>
80px
</div>
```

#### CSS

```
.level1{
  width: 400px;
  height: 400px;
  bakcground: red;
  font-size: 80px;
}
.level2{
  background: green;
}
.level3{
  background: blue;
}
```

De esta manera se muestra en el navegador:

## WEB



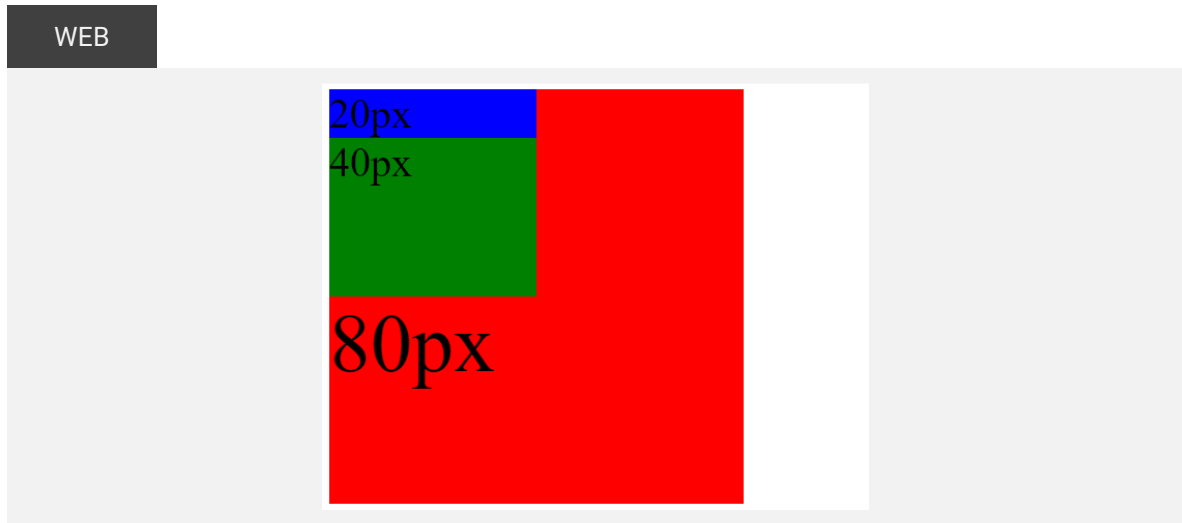
20px  
40px  
80px

Ahora, se agregarán algunas propiedades en la hoja de estilo. En la clase “**caja2**” se agregará un ancho y un alto del 50%, así mismo el tamaño de fuente se definirá al 50%.

## CSS

```
.level1{  
  width: 400px;  
  height: 400px;  
  background: red;  
  font-size: 80px;  
}  
.level2{  
  background: green;  
  width: 50%;  
  height: 50%;  
  font-size: 50%;  
}  
.level3{  
  background: blue;  
}
```

Se muestra que en el navegador que el elemento ahora posee el 50%, es decir, la mitad del alto, ancho y tamaño de fuente que el elemento padre.



Esto se debe a que los porcentajes hacen que estas propiedades adopten un valor proporcional a la longitud del elemento padre, quedando el ancho y el alto en 200 pixeles y un tamaño de fuente de 40 pixeles.

# UNIDAD 4. MODELO DE CAJA

## 4.1 Alto y ancho

Aquí se mostrará a como darle un alto y un ancho a un elemento usando CSS. Para modificar el tamaño de un elemento HTML tenemos dos propiedades llamadas **width** y **height**. La propiedad **width**, nos permite proporcionarle un ancho a un elemento HTML, así mismo la propiedad **height** nos permite dar un alto.

**\*\*NOTA:** Ambas propiedades puede tener como valores longitudes o porcentajes como las que se mencionaron anteriormente.

Tenemos tres párrafos a los cuales se les puede asignar la propiedad **width** y **height**.

### HTML

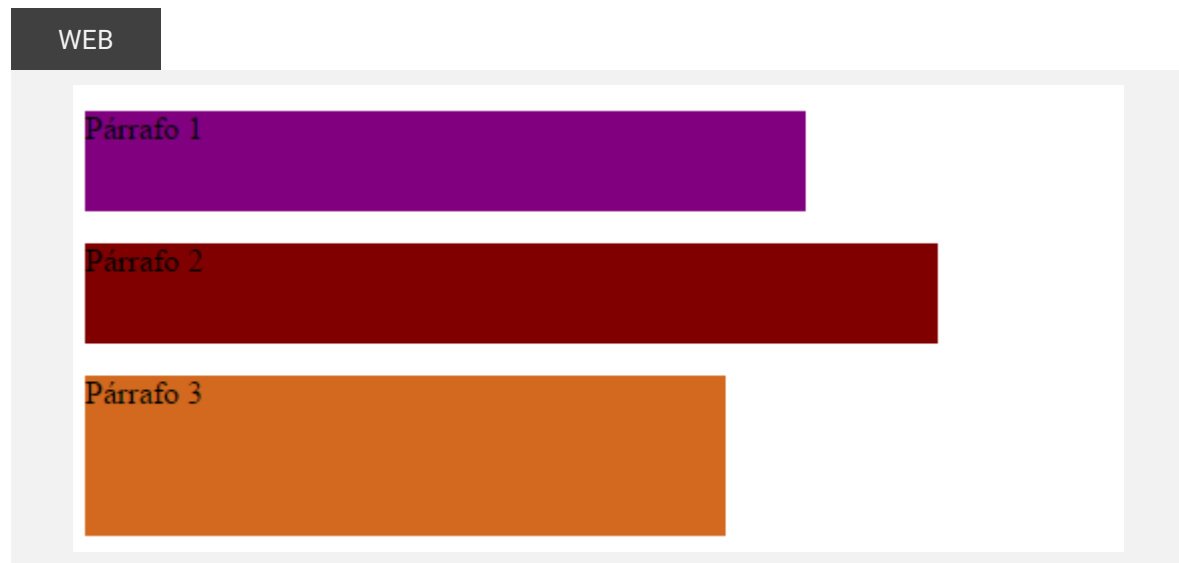
```
<p class = "parrafo1">Párrafo 1</p>
<p class = "parrafo2">Párrafo 2</p>
<p class = "parrafo3">Párrafo 3</p>
```

Al primer párrafo se asignará un color de fondo purpura así mismo un ancho de 360 pixeles y un alto de 50 pixeles. Al segundo párrafo le asignaremos un color de fondo marrón, un ancho de 80% y de alto 50 pixeles, aquí se ve que se puede usar más de una unidad de medida y al tercer párrafo pondremos un color de fondo chocolate, un tamaño de fuente de 16 pixeles, un ancho de 20em, unidad que es dado a base del tamaño de fuente, dando como valor  $20 \times 16 = 320$  pixeles de ancho y por último un alto de 80 pixeles.

### CSS

```
.parrafo1{
  background: purple;
  width: 360px;
  height: 50px;
}
.parrafo2{
  background: maroon;
  width: 80%;
  height: 50px;
}
.parrafo3{
  background: chocolate;
  font-size: 16px;
  width: 20em;
  height: 80px;
}
```

De esta manera vemos que los párrafos adoptaron el tamaño según las medidas que se les ha indicado.



De esta manera se hace uso de las propiedades **width** y **height** y para usar estas propiedades podemos hacer uso de longitudes relativas y absolutas, así como de porcentajes.

## 4.2 Display

En esta sección se hablará sobre una propiedad CSS que se llama **display** y sobre los valores más importantes que podemos asignarle.

Todo elemento tiene una propiedad llamada **display** por defecto, sin embargo, podemos darle a esta propiedad diversos valores, entre los más importantes tenemos los siguientes:

- **block.**
- **inline.**
- **inline-block**
- **none.**

Se muestra el siguiente ejemplo en el que se tienen tres elementos **<div>** con contenido de texto cada uno.

```
HTML
<div>div1</div>
<div>div2</div>
<div>div3</div>
```

En el navegador se muestra uno debajo del otro.

## WEB

```
div1  
div2  
div3
```

Se agregará una hoja de estilos en donde al elemento `<div>` se le asignará un borde de 1 pixel y la propiedad **solid**:

## CSS

```
div{  
  border: 1px solid;  
}
```

Y se muestra la verdadera magnitud del espacio que ocupa un elemento `<div>`.

## WEB

```
div1  
div2  
div3
```

El elemento `<div>` tiene por defecto la propiedad **display** con valor **block** lo cual hace que ocupe todo el ancho disponible, en este caso el ancho del navegador. Así mismo, el elemento bloque hace que se ubique debajo de cualquier elemento anterior.

Se le agrega un ancho de 100 pixeles y un alto de 100px.

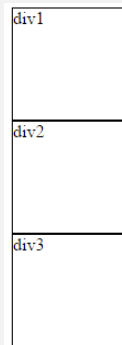
## CSS

```
div{  
  border: 1px solid;  
  width: 100px;  
  height: 100px;  
}
```

Como se muestra, nos damos cuenta que el elemento `<div>` si respeta las propiedades anteriormente mencionadas.



## WEB



El valor **block** de **display** es utilizado normalmente para grandes piezas de contenido, tales como los elementos estructurales **<header>**, **<body>**, **<footer>**, entre otros que lo tienen por defecto.

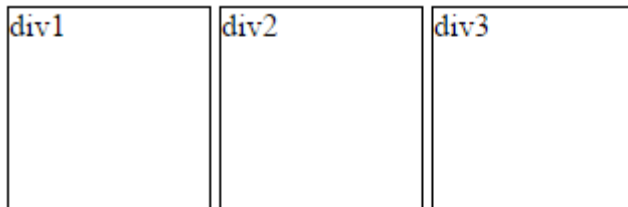
Siguiendo con el ejemplo, se agregará la propiedad **display** al elemento **<div>** con el valor de **inline-block**.

## CSS

```
div{  
  border: 1px solid;  
  width: 100px;  
  height: 100px;  
  display: inline-block;  
}
```

Ahora vemos que los elementos se ubican uno después de otro y también reconocer el alto y el ancho de cada uno de ellos. Si se borra el ancho y el alto, adoptarán en ancho y alto necesario, a diferencia que el **display-block** que ocupaba todo el ancho del navegador.

## WEB



Ahora, se cambiará el valor de **display** por **inline** y esto hará que se mantenga la misma visualización, un elemento después del otro.

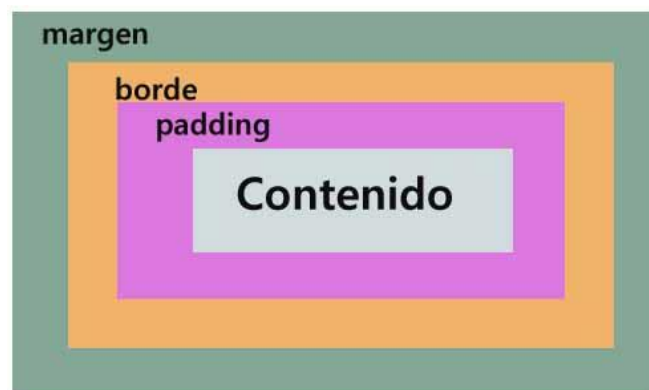
**\*\*NOTA:** Si un elemento tiene como valor `inline` en su propiedad `display` si se le agrega algún ancho y alto al elemento, este no reconocerá esas propiedades. El valor `inline` es usado generalmente por pequeñas piezas de contenido como palabras con énfasis.

El valor `none` de la propiedad `display` se verá que los elementos se ocultaron por completo. Esto se debe a que el valor `none` hace que el navegador los interprete como si no existieran, así mismo si se anida un elemento dentro de otro este elemento también será ocultado.

## 4.3 Modelo de caja

De acuerdo con el concepto de modelo de caja, cada elemento de la página es un elemento rectangular y puede tener ancho, alto, relleno, borde y margen, manifestados en las propiedades CSS `width`, `height`, `padding`, `border` y `margin` respectivamente. Todos los elementos se ajustan al modelo de caja por lo cual es importante conocerlo.

Cada elemento es un elemento es una caja rectangular y hay varias propiedades que determinan el tamaño de esa caja.



- El centro de la caja es conocido como área de contenido y está definido por el ancho y alto de un elemento que pueden ser determinados por el valor de la propiedad `display`, por el contenido del elemento o por ser especificados por las propiedades `width` y `height`. El rectángulo que rodea el contenido es conocido como filo de contenido o content edge.
- Alrededor del área de contenido está el área de relleno o padding area y el rectángulo que lo rodea es llamado filo del relleno.
- A fuera del `padding` está el área del borde o border area y el rectángulo que lo rodea es el filo del borde.
- Por último, afuera del borde está el área del margen o margin area por lo cual es delimitado por el filo del margen.

**\*\*NOTA:** Cabe resaltar que la propiedad `margin` no afecta el tamaño de la caja, pero genera un espacio hacia afuera del borde que rodea la caja y afecta la interacción con otras cajas por lo tanto es parte importante del modelo de caja.

De estas propiedades, **margin** posee un color de fondo transparente, siendo **border** la única propiedad a la que podemos asignarle un color de fondo. Así mismo, de estas tres propiedades, solo **margin** acepta valores negativos. Veamos el siguiente ejemplo.

HTML	CSS
<pre>&lt;!DOCTYPE html&gt; &lt;html lang="es"&gt; &lt;head&gt;   &lt;meta charset="UTF-8"&gt;   &lt;title&gt;&lt;/title&gt;   &lt;link rel="stylesheet" href="text/css" href="estilos.css"&gt; &lt;/head&gt; &lt;body&gt;   &lt;div class="caja1"&gt;     Caja1   &lt;/div&gt;   &lt;div class="caja2"&gt;     Caja2   &lt;/div&gt; &lt;/body&gt; &lt;/html&gt;</pre>	<pre>body{   width: 400px;   margin: 0 auto;   height: auto; } .caja1{   background: yellow;   width: 400px;   height: 100px; } .caja2{   background: tomato;   width: 400px;   height: 100px; }</pre>

En el archivo HTML tenemos dos elementos `<div>` el primero con clase `"caja1"` y el segundo con clase `"caja2"`. En el archivo CSS, se ve que ambos tienen un ancho de 400 pixeles y un alto de 100 pixeles, así mismo la clase `"caja1"` tiene un fondo de color amarillo y `"caja2"` un fondo de color tomate.



## 4.3 Padding

En esta sección se hablará sobre la propiedad **padding**, para ello se mostrará el siguiente ejemplo para su mejor entendimiento.

Tenemos tres elementos **<div>**, cuyas clases son **caja1**, **caja2** y **caja3**, todos con ancho de 200 pixeles y alto de 100 pixeles. El primero con color de fondo amarillo, el segundo azul y el tercero rojo.

### HTML

```
<div class = "caja">
  Caja 1
</div>
<div class = "caja2">
  Caja 2
</div>
<div class = "caja3">
  Caja 3
</div>
```

### CSS

```
body{
  margin: 0;
}
.caja1{
  background: yellow;
  width: 200px;
  height: 100px;
}
.caja2{
  background: blue;
  width: 200px;
  height: 100px;
}
.caja3{
  background: red;
  width: 200px;
  height: 100px;
}
```

De esta manera se ve en el navegador:

### WEB



Veremos cómo interactúa la propiedad **padding** con los elementos con la propiedad **display-block**. Esta propiedad no es necesaria declararla, dado que por defecto el elemento **<div>** posee **display-block**.

Se le agregaran algunas propiedades más CSS, las que son **padding-top**, **padding-left**, **padding-bottom** y **padding-right**.

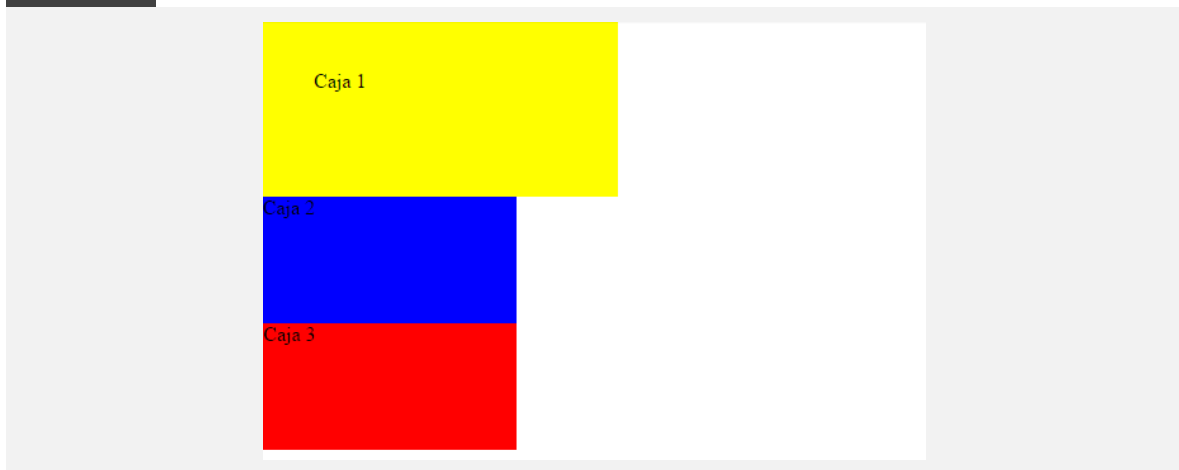
- **padding-top**: Aumenta el espacio en la parte superior de la caja.
- **padding-left**: Aumenta el espacio de la izquierdo de la caja.
- **padding-right**: Aumenta el espacio de la derecha de la caja.
- **padding-bottom**: Aumenta el espacio en la parte inferior.

#### CSS

```
.caja1{
  background: yellow;
  width: 200px;
  height: 100px;
  padding-top: 40px;
  padding-left: 40px;
  padding-bottom: 40px;
  padding-right: 40px;
}
```

Se muestra que en la parte superior, inferior, derecha e izquierda existe más espacio.

#### WEB



Esta notación, para cambiar el **padding** de un elemento, se puede abreviar usando solamente la propiedad **padding** que puede tener cuatro valores. El primero que representa el **padding** superior, el segundo el **padding** derecho, el tercero el **padding** inferior y por último el cuarto el **padding** izquierdo.

## CSS

```
.caja1{
  background: yellow;
  width: 200px;
  height: 100px;
  padding: 40px 40px 40px 40px;
}
```

**\*\*NOTA:** Así mismo se pueden reducir estos valores a solo dos donde el primer valor se aplica tanto al padding superior como inferior, mientras que el segundo se aplica al padding derecho e izquierdo. Además, se puede reducir todo esto a un solo valor, donde el valor indicado representa el mismo padding para los cuatro lados.

Ahora se verá como interactúa la propiedad **padding** con los elementos con propiedad **display inline-block**, para esto a los selectores caja 2 y caja 3 se le agregaran la propiedad **display inline-block**.

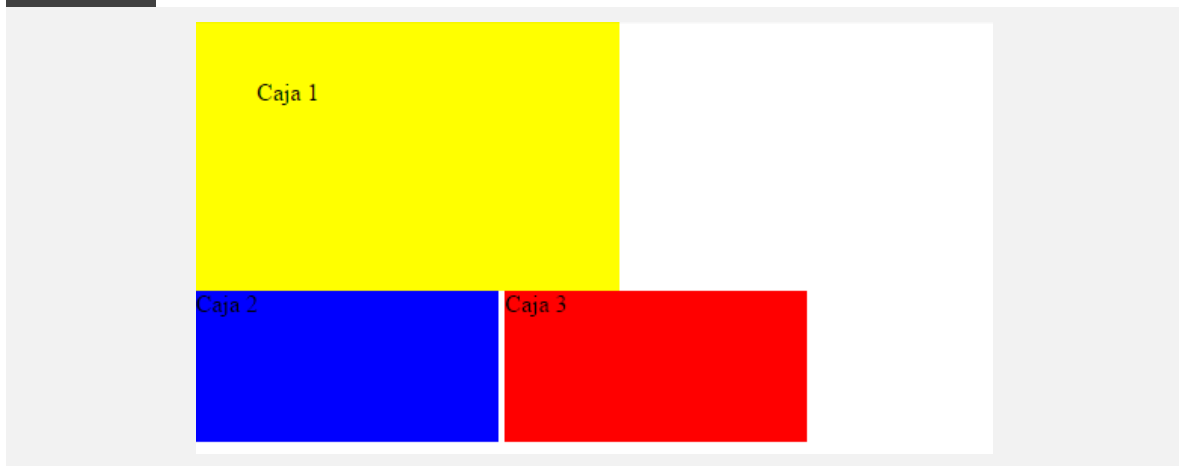
## CSS

```
.caja2{
  background: blue;
  width: 200px;
  height: 100px;
  display: inline-block;
}
.caja3{
  background: red;
  width: 200px;
  height: 100px;
  display: inline-block;
}
```

Dado que el primer elemento posee **display-block**, los demás elementos se ubican en la parte inferior y dado que el segundo y tercer elemento poseen **display inline-block** se ubica uno a costado del otro.

**\*\*NOTA:** Existe un espacio entre uno y otro. Esto no se debe a un margen, si no que los elementos tienen un pequeño salto de línea, lo que al igual que un espacio en blanco genera un espacio.

## WEB



Ahora, se mostrará la forma en la que interactúan los elementos que tienen como predeterminado **display-inline**, para ello, se cambiara el valor de **display** en la hoja de estilos por **inline**.

## CSS

```
.caja2{
  background: blue;
  width: 200px;
  height: 100px;
  display: inline;
}
.caja3{
  background: red;
  width: 200px;
  height: 100px;
  display: inline;
}
```

Como se muestra, las propiedades **width** y **height** no afectan a un elemento con **display-inline**.

## WEB



## 4.4 Bordes

En esta sección se hablará de la forma en cómo se aplican bordes a un elemento. Los bordes se encuentran ubicados entre el **margin** y el **padding** proporcionando un contorno alrededor del elemento.

Se tiene un elemento `<div>` con un color de fondo naranja, un ancho de 300 pixeles y un alto de 200 pixeles. Se utilizará la propiedad **border** que requiere de tres valores, estos son: ancho, estilo y color.

Para definir el ancho se puede utilizar unidades de medida que se vieron anteriormente, en este caso tiene 8 pixeles.

Para el estilo, los valores más comunes son: **solid**, **double**, **dashed** y **dotted** que brindan una apariencia distinta al borde, mientras que el valor **none** desaparece el borde, así mismo hay otros valores que se pueden elegir, en este ejemplo se usará **solid**.

En color, se puede hacer uso de cualquiera de los sistemas que se vieron anteriormente, en este caso se usará el sistema RGB aplicando un color azul.

### HTML

```
<div>
  border
</div>
```

### CSS

```
body{
  Margin: 0;
}
div{
  background: orange;
  width: 300px;
  height: 200px;
  border: 8px solid rgb(0,0,255);
}
```

De esta forma se muestra en el navegador:

### WEB



border



**\*\*NOTA:** Una forma de también expresar lo anterior visto con la declaración de un borde para un elemento puede ser de la siguiente manera:

- **border-width: 8px;**
- **border-style: solid;**
- **border-color: azul;**

De esta forma se obtiene el mismo resultado mostrado anteriormente.

Existen más propiedades que permiten asignar **border** específicamente a ciertos lados de un elemento.

Se tiene un elemento `<div>` con un ancho de 300 pixeles, un alto de 200 pixeles y color de fondo naranja. Ahora se agrega la propiedad **border-top** con un ancho de 10 pixeles, estilo **solid** y color rojo.

Ahora se asigna la propiedad **border-right** y se le asigna un ancho de 15 pixeles, estilo **dotted** y color azul.

Posteriormente la propiedad **border-bottom** con ancho de 10 pixeles, estilo **double** y color verde.

Y finalmente, la propiedad **border-left**, 15 pixeles de ancho, estilo **dashed** y color amarillo.

#### CSS

```
body{
  Margin: 0;
}
div{
  background: orange;
  width: 300px;
  height: 200px;
  border-top: 10px solid red;
  border-right: 15px dotted blue;
  border-bottom: 10px double green;
  border-left: 15px dashed yellow;
}
```

De esta manera se pueden asignar estilos a cada **border** por separado.



## 4.5 Margin

Esta sección se hablará sobre la propiedad **margin**, para ello se mostrará un ejemplo para conocerla más a profundidad.

Primero, se debe saber que la propiedad **margin** permite establecer una cantidad de espacio que rodea un elemento. Los márgenes de un elemento quedan fuera de cualquier frontera y son totalmente transparentes en color. Los márgenes pueden ser usados para posicionar los elementos en un lugar determinado de una página o para proporcionar espaciamiento, manteniendo todo lo demás elementos a cierta distancia.

Para lo anterior mencionado tenemos a tres elementos div que por defecto tienen la propiedad block. El primer y tercer elemento tienen un ancho de 400 píxeles y un alto de 100 píxeles, mientras que el segundo elemento tiene un ancho de 300 píxeles y un alto de 100 píxeles, así mismo se han utilizado los colores de la bandera de España.

Ahora, a la clase **caja2** se le asignará una propiedad **margin-top** con un valor de 20 píxeles, luego **margin-left** de 20 píxeles que generará un desplazamiento de 20 píxeles hacia su izquierda, así mismo se agrega **margin-bottom** igual a 20 píxeles y **margin-right** igual a 20 píxeles, generando márgenes a los cuatro lados del elemento.

### HTML

```
<div class="caja1">caja1</div>
<div class="caja2">caja2</div>
<div class="caja3">caja3</div>
```

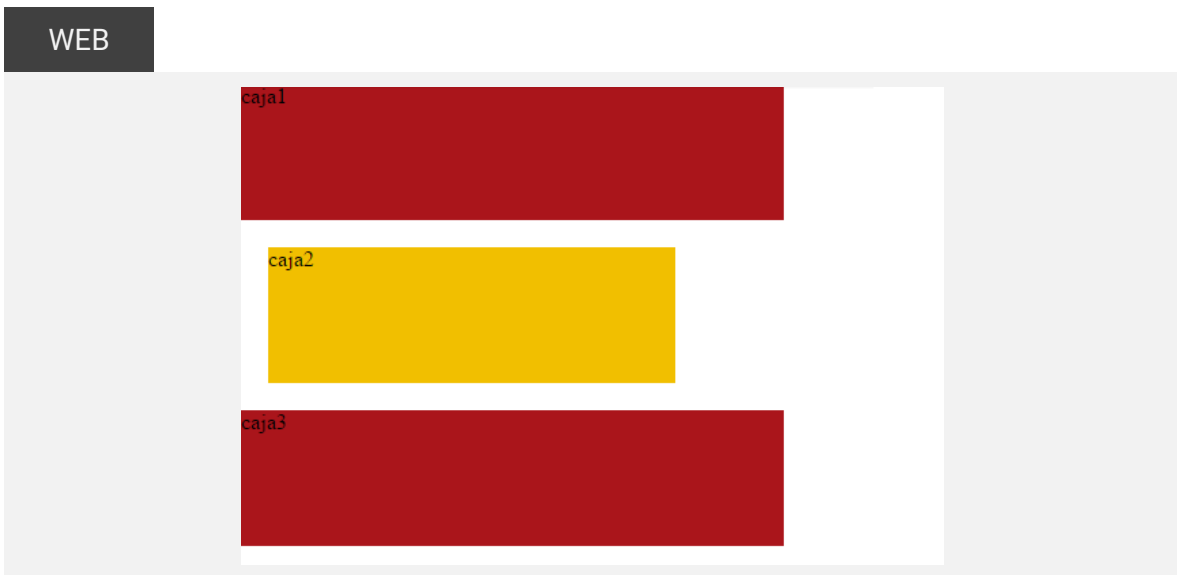
### CSS

```
body{
  margin: 0;
}
.caja1{
  background: #aa151b;
  width: 400px;
  height: 100px;
}
.caja2{
```



```
background: #f1bf00;  
width: 300px;  
height: 100px;  
margin-top: 20px;  
margin-left: 20px;  
margin-right: 20px;  
margin-bottom: 20px;  
}  
.caja3{  
background: #aa151b;  
width: 400px;  
height: 100px;  
}
```

De esta manera pueden ver los márgenes que se genera alrededor del elemento `<div>` y como hay un espacio en los diferentes lados indicados.



Así mismo, estas propiedades se pueden reducir en una sola, a la propiedad que tiene por nombre **margin**. Esta propiedad tiene cuatro valores, los cuales son el **margin-top**, **margin-right**, **margin-bottom** y **margin-left**.



**\*\*NOTA:** De igual forma los valores de margin se pueden reducir a dos, donde el primero representa el margen superior e inferior, y el segundo representa el margen derecho e izquierdo.

Si solamente se deja un solo valor, este representará los cuatro lados del margen, arriba, derecha, abajo e izquierda.

La propiedad margin también acepta valores negativos, para ello, a la propiedad margin agregada anteriormente se cambiarán los valores por -20px, 0px y 0px.



La interacción con la propiedad **margin** para los elementos que tienen como propiedad **display inline-block** no tienen la reacción anterior con los otros elementos, así que si para un elemento con **display inline-block** se le aplica un margen de -20 pixeles hacia arriba, este se quedará debajo del otro elemento, no se encimará, en cambio si se le aplica hacia un costado, derecho o izquierdo, si se encimará.