

Apuntes de CSS

migueluisV

Realizadas: Diciembre 2022

Índice

1 Sintaxis	8
1.1 Tipos de Selectores	8
1.2 Selectores descendentes	9
1.3 Comentarios	10
2 Formas de aplicar estilos CSS	10
2.1 Inline	10
2.2 Embedded o Internal	11
2.3 External	11
3 Herencia	12
4 Trabajando con texto	14
4.1 Fuentes	14
4.2 Tamaño de fuente	15
4.3 Estilo de fuente	16
4.4 Grosor de fuente	17
4.5 Variación de fuente	19
4.6 Alineando texto horizontalmente	19
4.7 Alineando texto verticalmente	21
4.8 Color de texto	23
4.9 Decoración del texto	23
4.10 Indentación del texto	25
4.11 Sombra del texto	25
4.12 Transformación del texto	26
4.13 Espaciado entre letras	27
4.14 Espaciado entre palabras	27
4.15 Manejo de espacios y saltos de línea	28
5 Posicionamiento	31
6 Floating	33
7 Propiedades para la visualización de elementos	35
7.1 display	35
7.2 visibility	37
7.3 clear	38
7.4 overflow	40
7.5 z-index	42
8 CSS3	44
8.1 Prefijos del proveedor o buscador	44
8.2 Esquinas redondeadas	44
8.3 Sombras	46

8.3.1	Técnicas para el sombreado	47
8.3.2	Sombras en texto	48
8.4	Transparencia	49
8.5	Pseudo clases	51
8.6	Pseudo elementos	52
8.7	word-wrap	54
8.8	@font-face	55
9	Degrados	57
9.1	Degrado lineal	57
9.1.1	Prefijos, múltiples escalas y presencia en el degradado	57
9.1.2	Dirección o ángulo del degradado	58
9.1.3	Repetir el degradado	60
9.2	Degrado radial	60
10	Fondos	62
10.1	background-size	62
10.2	background-clip	63
10.3	Bordes transparentes	65
10.4	Varias imágenes de fondo	65
10.4.1	background-position	66
10.5	opacity	67
11	Transiciones	69
11.1	Velocidad de la transición	70
11.2	Retraso de la transición	70
12	Transformaciones	71
12.1	rotate()	71
12.2	transform-origin()	72
12.3	translate()	73
12.4	skew()	74
12.5	scale()	76
13	Animaciones	77
13.1	Keyframes & animaciones	77
13.2	Propiedades de las animaciones	77
14	Filtros	80
14.1	Funciones	81
14.2	Usando múltiples filtros	85

Índice de Figuras

1	Orden de prioridad de estilos	9
2	Selectores descendentes	10
3	Uso de fuentes	15
4	Cambiando el tamaño de fuente	16
5	Cambiando el estilo de fuente	17
6	Cambiando el grosor de fuente	18
7	Cambiando la variación de fuente	19
8	Cambiando la alineación de texto	20
9	Cambiando la alineación vertical de texto 1	22
10	Cambiando la alineación vertical de texto 2	22
11	Cambiando el color del texto	23
12	Cambiando la decoración del texto	24
13	Cambiando la indentación del texto	25
14	Agregando sombra al texto	25
15	Cambiando las mayúsculas y minúsculas del texto	27
16	Cambiando el espaciado de caracteres del texto	27
17	Cambiando el espaciado de palabras del texto	28
18	Cambiando el manejo de espacios y saltos de líneas del texto	30
19	Diferencia entre valores de propiedad <i>position</i>	33
20	Diferencia entre valores de propiedad <i>float</i>	34
21	Aspecto de las visualizaciones <i>block</i> e <i>inline</i>	35
22	Diferencia entre valores de propiedad <i>display</i>	37
23	Diferencia entre valores de propiedad <i>visibility</i>	38
24	Diferencia entre valores de propiedad <i>clear</i>	39
25	Diferencia entre valores de propiedad <i>overflow</i>	41
26	Funcionamiento de la propiedad <i>z-index</i>	43
27	Diferencia entre valores de propiedad <i>border-radius</i>	45
28	Diferencia entre valores de propiedad <i>box-shadow</i>	46
29	Diferencia entre valores de propiedad <i>box-shadow</i>	48
30	Diferencia entre valores de propiedad <i>text-shadow</i>	49
31	Uso de RGBA, HSL y HSLA	51
32	Uso de las Pseudo clases	52
33	Uso de los Pseudo elementos	54
34	Diferencia entre valores de propiedad <i>word-wrap</i>	55
35	Uso de la regla <i>@font-face</i>	56
36	Utilizando el degradado lineal simple	57
37	Utilizando el degradado lineal múltiple	58
38	Utilizando el degradado lineal con dirección	59
39	Repetiendo un degradado lineal	60
40	Utilizando el degradado radial	62
41	Diferencia entre los valores <i>contains</i> y <i>cover</i>	63
42	Diferencia entre los valores <i>border-box</i> , <i>padding-box</i> y <i>content-box</i>	64
43	Consiguiendo un borde transparente	65

44	Múltiples fondos	66
45	Transparencia de imágenes	68
46	Método rotate() para rotar un elemento	72
47	Función transform-origin() para cambiar el eje de un elemento	73
48	Método translate() para mover un elemento	74
49	Método skew() para inclinar un elemento	75
50	Método scale() para cambiar escala de un elemento	77
51	Método drop-shadow() para agregar una sombra a un elemento	80
52	El círculo de colores	82
53	Conjunto 1 de ejemplos de filtros CSS	83
54	Conjunto 2 de ejemplos de filtros CSS	85
55	Uso de múltiples filtros CSS	86

Índice de Tablas

1	Tipos de familias de fuentes	14
2	Usos de la propiedad <i>white-space</i>	29
3	Nuevas herramientas para colores en CSS3	49

Este documento se hizo con **Overleaf** y los ejemplos fueron desarrollados y probados con **Visual Studio Code**, con su extensión **Live Server** en el buscador **Microsoft Edge** o **Brave**, por lo que algunas propiedades puede que lleguen a requerir un *Prefijo de buscador* para funcionar.

La estructura HTML de los ejemplos en este documento serán omitidos, dejando únicamente lo vital para que los ejemplos funcionen y para evitar que este trabajo sea muy largo, los ejemplos CSS si vendrán completos.

1 Sintaxis

La sintaxis de CSS está constituida por **selectores**, **propiedades** y **valores**. Veamos un simple ejemplo:

```
h1 {  
    color: orange;  
}
```

Vemos que:

- Hay una etiqueta HTML previo a la apertura de llaves, esta etiqueta es llamada **Selector**, que es la etiqueta destino a la cual se le aplicará un estilo.
- Una vez abiertas las llaves, hay una palabra, dos puntos y otra palabra, la primera es llamada **propiedad**, que básicamente es el atributo a estilizar de la etiqueta.
- Es llamado **valor** a la asignación de un dato o valor a la propiedad de un selector.
- **Un solo selector puede tener múltiples propiedades** a estilizar, cada una de ellas es separada mediante un **punto y coma** (;).

1.1 Tipos de Selectores

Primero, debemos aclarar dos conceptos o atributos de etiqueta HTML que se verán muy constantemente a lo largo del desarrollo de sitios web:

- El atributo **id** permite asignarle un identificador a una etiqueta, este identificador es utilizado por CSS para aplicar un estilo particular.
- El atributo **class** opera de la misma manera que **id**, la diferencia radica en que *id* se limita a un solo uso por página, mientras que *class* puede usarse múltiples veces, te preguntarás porqué, la respuesta está en que JavaScript utiliza también la palabra *id* en su sintaxis, por lo que, si utilizará un estilo en múltiples etiquetas, se recomienda utilizar **class**.
- El orden de prioridad de selectores es:
 1. id.
 2. class.
 3. type.
- **Regular:** se le aplica un estilo a una etiqueta.
- **id & class:** se le aplica un estilo a un **identificador (id)** o **class** específico mediante el carácter **gato (#)** y **punto (.)** respectivamente

Veamos un ejemplo (*Figura 1*):

```

estilos.css
p {
    color: white;
    background-color: gray;
}
#intro {
    background-color: red;
}
.cuerpo {
    background-color: green;
}

prueba.html
<p>Párrafo 1</p>
<p class="cuerpo">Párrafo 2</p>
<p class="cuerpo" id="intro">Párrafo 3</p>
<div class="cuerpo" id="intro">
    <p>Párrafo 4</p>
</div>

```

Figure 1: Orden de prioridad de estilos



Vemos que hay cuatro párrafos, a cada uno tiene un estilo diferente: el párrafo 1 no tiene ni identificador ni clase, por lo que el estilo a aplicar es el de las etiquetas **p**; el párrafo 2 tiene el identificador "intro", por lo que el estilo a aplicar es el de dicho identificador; el párrafo 3 tiene la clase "cuerpo", por lo que el estilo a aplicar es el de dicha clase; el párrafo 4 está contenido dentro de una etiqueta **div**, la cual tiene el identificador y clase anteriormente mencionados, por lo que debemos considerar el orden de prioridad de los selectores CSS y HTML, por lo que el estilo a aplicar es el de las etiquetas **p**.

1.2 Selectores descendentes

Podemos aplicar un estilo a una etiqueta contenida dentro de otra que posee un estilo definido (un identificador, una clase o estilo de etiqueta), es decir, clase dentro de clase, identificador dentro de un identificador, clase dentro de identificador o viceversa, miremos un ejemplo (*Figura 2*):

```
estilos.css
```

```
#intro .primero em {  
    color: pink;  
    background-color: gray;  
}  
  
prueba.html  
<div id="intro">  
    <p class="primero"><em>Párrafo 1</em> dentro del div</p>  
    </div>  
    <p class="primero">Párrafo 2 fuera del div.</p>  
    <p>Párrafo 3 fuera del div</p>
```

Figure 2: Selectores descendentes

Párrafo 1 dentro del div

Párrafo 2 fuera del div.

Párrafo 3 fuera del div

1.3 Comentarios

Para comentar una o varias líneas de código en CSS se debe escribir los caracteres /* al comienzo o primer línea y */ al final o última línea de código.

```
/* Esto es un comentario. */  
  
/* Todo este estilo está comentado.  
p {  
    color: red;  
} */  
  
#intro .primero em {  
color: pink;  
background-color: gray;  
}
```

2 Formas de aplicar estilos CSS

Los estilos CSS pueden ser aplicados de tres maneras sencillas:

2.1 Inline

Implica aplicar el estilo mediante la atributo HTML **style**, dentro de sus dos comillas, asignamos las propiedades CSS de estilo que la etiqueta u objeto HTML tendrá, únicamente esta etiqueta, no el resto del mismo tipo.

```
<html>
  <head>
    <title>Prueba</title>
  </head>
  <body>
    <!-- Estilo Inline. -->
    <p style="color: white; background-color: gray;">Hola </p>
    <p>mundo.</p>
  </body>
</html>
```

En el ejemplo anterior, solamente la primer etiqueta tendría un estilo definido.

2.2 Embedded o Internal

Contrario a la forma *Inline*, donde se le aplica un estilo a una sola etiqueta, la forma **Embedded** crea un estilo dentro de la estructura HTML y se le puede aplicar a cuantas etiquetas se desee. Estos estilos internos son creados en la cabecera de la estructura del documento, utilizando la etiqueta **style**.

```
<html>
  <head>
    <title>Prueba</title>
    <!-- Estilo Embedded. -->
    <style>
      p {
        color: white;
        background-color: gray;
      }
    </style>
  </head>
  <body>
    <p>Hola </p>
    <p>mundo.</p>
  </body>
</html>
```

En el ejemplo anterior, ambas etiquetas **p** tienen un mismo estilo definido. Es recomendable utilizar estilos internos cuando una sola página de un sitio web tiene un estilo único.

2.3 External

Con este método, debemos contar con un archivo con extensión **.css** en nuestro directorio y lo enlazamos a nuestro documento HTML mediante la siguiente etiqueta:

<link href="ejemplo.css" rel="stylesheet">

Veamos un ejemplo:

```
/* Estilo External. */
ejemplo.css
p {
```

```
        color: white;
        background-color: gray;
    }

prueba.html
<html>
    <head>
        <title>Prueba</title>
        <!-- Enlaza hoja de estilos al document HTML. -->
        <link href="ejemplo.css" rel="stylesheet">
    </head>
    <body>
        <p>Hola </p>
        <p>mundo </p>
        <p>estoy probando estilos CSS.</p>
    </body>
</html>
```

Así, todos los cambios que hagamos a la hoja de estilos en su propio archivo se verá reflejado en el sitio web, siempre y cuando esta hoja de estilos esté ligada a la página.

3 Herencia

Al igual que otros lenguajes con un enfoque distinto, CSS aplica herencia a los elementos o etiquetas contenidas dentro de una que ya tenga un estilo definido, veamos como funciona esto:

```
estilos.css
body {
    color: red;
}

prueba.html
<p>Hola </p>
<p>mundo </p>
<p>estoy probando estilos CSS.</p>
```

En el ejemplo, ninguna etiqueta **p** tiene estilo definido, por lo que se les aplica el estilo de su etiqueta padre **body**.

```
estilos.css
body {
    color: red;
}
p {
    color: blue;
}

prueba.html
<p>Hola </p>
<p>mundo </p>
<p>estoy probando estilos CSS.</p>
```

Se creamos un estilo para las etiquetas **p** con un color de texto distinto, a estas etiquetas se les aplicará el estilo diseñado para ellas, en vez del estilo de su padre **body**, por lo que podemos concluir que **los estilos de los hijos de etiquetas tienen prioridad sobre sus padres.**

4 Trabajando con texto

4.1 Fuentes

La propiedad **font-family** (familia de fuentes) de CSS permite establecer la fuente (tipo de letra) de un elemento o etiqueta. La *Tabla 1* contiene los dos tipos de familias de fuentes:

Table 1: Tipos de familias de fuentes

Generic family	Font family
Serif	Times New Roman Georgia
Sans - serif	Arial Verdana
Monospace	Courier New Lucida Console

Las **familias genéricas** (o generic family) son un grupo de familias de fuentes, es decir, un grupo de fuentes, mientras que la familia de fuentes (o family font) es una fuente o tipo de letra que se puede utilizar.

Asignemos algunas fuentes a algunos párrafos y veamos el resultado en la *Figura 3*:

```
estilos.css
/*
Define distintas clases que almacenan las distintas fuentes.
*/
.serif {
    /* Si fuente "Times New Roman" no está disponible, prueba con Times, si
       esta tampoco, con alguna parecida de la familia serif. */
    font-family: "Times New Roman", Times, serif;
}
.sansserif {
    font-family: Helvetica, Arial, sans-serif;
}
.monospace {
    font-family: "Courier New", Courier, monospace;
}
.cursive {
    font-family: Florence, cursive;
}
.fantasy {
    font-family: Blippo, fantasy;
}
.systemui {
    font-family: system-ui, -apple-system, BlinkMacSystemFont, 'Segoe UI', Roboto, Oxygen,
    Ubuntu, Cantarell, 'Open Sans', 'Helvetica Neue', sans-serif;
}

prueba.html
<p class="serif">Escrito con Serif.</p>
<p class="sansserif">Escrito con Sans-Serif.</p>
<p class="monospace">Escrito con Monospace.</p>
<p class="cursive">Escrito con Cursive.</p>
```

```
<p class="fantasy">Escrito con Fantasy.</p>
<p class="systemui">Escrito con System UI.</p>
```

Figure 3: Uso de fuentes

Escrito con Serif.

Escrito con Sans-Serif.

Escrito con Monospace.

Escrito con Cursive.

Escrito con Fantasy.

Escrito con System UI.

Como podemos ver, en el ejemplo anterior, si la fuente que deseamos no está disponible, podemos cargar otra deseada, si tampoco está disponible, podemos probar con alguna similar dentro de una familia, separado por comas (,); el ejemplo solamente prueba con dos fuentes antes de acudir a una familia, pero pueden ser más (no olvide que los nombres que son escritos con más de una palabra deben ser encerrados entre dobles comillas).

Una buena práctica es asignarle a todo el sitio web (body) una fuente genérica de respaldo, es decir, si queremos que todo nuestro sitio tenga la fuente de Google, pero por algún motivo, esta no está disponible (por la misma empresa o por una falla con el internet o máquina del cliente), podamos cargar otra fuente deseada y evitar un error de lectura. Lo anterior se resuelve como lo vimos en el ejemplo, poniendo diversas fuentes separadas por comas, para al final, poner una familia genérica.

4.2 Tamaño de fuente

La propiedad **font-size** de CSS permite establecer el tamaño de fuente del texto de un elemento o etiqueta. Una de las formas de hacer grande o pequeña el tamaño de fuente es por medio de **palabras clave**, tales como:

- **small (smaller, x-small, xx-small)**.
- **medium**.
- **large (larger, x-larger, xx-larger)**.
- **inherit, initial y unset**.

Este método evita que el usuario varie el tamaño de la fuente, de esta forma, evitando que afecte la apariencia del sitio web. Un ejemplo y su resultado (*Figura 4*):

```

estilos.css
/*
Define distintas clases con tamaño de fuente distintas.
*/
.small {
    font-size: small;
}
.medium {
    font-size: medium;
}
.large {
    font-size: large;
}
.xlarge {
    font-size: x-large;
}

prueba.html
<p class="small">Escrito pequeño.</p>
<p class="medium">Escrito mediano.</p>
<p class="large">Escrito grande.</p>
<p class="xlarge">Escrito muy grande.</p>

```

Figure 4: Cambiando el tamaño de fuente

Escrito pequeño.
 Escrito mediano.
 Escrito grande.
 Escrito muy grande.

La otra forma de definir el tamaño de fuente es mediante **Longitudes: px, em o porcentaje (%)**, la primera permite precisión en cuanto a píxeles en pantalla, mientras que la segunda permite cambio de tamaño en buscadores, pruebe ambas longitudes con zoom en su buscador para ver cómo funcionan y si se mantienen legibles (recuerde que 1em es igual a píxeles/16).

```

// Ambos tamaños son el mismo.
font-size: 20px;
font-size: 1.25em;

```

4.3 Estilo de fuente

La propiedad **font-style** (estilo de fuente) permite aplicar un estilo de letra a la fuente, los valores que acepta esta propiedad son:

- **normal**: texto sin ningún tipo de estilo (ni negrita).

- **italic**: texto en cursiva (lo mismo que la etiqueta **i**).
- **oblique**: texto como *italic*, pero un poco más redonda (no es soportada por algunos buscadores).
- **oblique <ángulo>**: texto como *italic*, pero se le puede añadir un ángulo de inclinación al texto, esta propiedad no es muy soportada por los buscadores, los valores que puede recibir van de -90 a 90 deg.
- **inherit, initial y unset**.

Veamos un ejemplo y su resultado (*Figura 5*):

```
estilos.css
/* Define distintas clases con estilos de fuente distintas. */
.normal {
    font-style: normal;
}
.italic {
    font-style: italic;
}
.oblique {
    font-style: oblique;
}

prueba.html
<p class="normal">Escrito normal.</p>
<p class="italic">Escrito italic.</p>
<p class="oblique">Escrito oblique.</p>
```

Figure 5: Cambiando el estilo de fuente

Escrito normal.

Escrito italic.

Escrito oblique.

¿Porqué requeriría de esto si ya lo puedo conseguir con las etiquetas de HTML?, bueno, podemos aplicar un estilo de fuente distinto dinámicamente, por ejemplo, al dar un clic en un botón, un texto pasa de negrita a normal, normal a cursiva, etc.

4.4 Grosor de fuente

La propiedad **font-weight** (peso de fuente literalmente) de CSS permite establecer el grosor de fuente del texto de un elemento o etiqueta. Acepta los siguientes valores:

- **normal**: texto normal con grosor predeterminado.
- **bold**: texto con un grosor mayor a *normal* (como la etiqueta **b** o **stronger**).
- **bolder**: texto con un grosor mayor a *bold*.
- **lighter**: texto con grosor menor a *normal*.
- **inherit**, **initial** y **unset**.

Veamos un ejemplo y su resultado (*Figura 6*):

```
estilos.css
/* Define distintas clases con grosores de fuente distintas. */
.lighter {
    font-weight: lighter;
}
.bold {
    font-weight: bold;
}
.bolder {
    font-weight: bolder;
}

prueba.html
<p class="lighter">Escrito lighter.</p>
<p class="bold">Escrito bold.</p>
<p class="bolder">Escrito bolder.</p>
```

Figure 6: Cambiando el grosor de fuente

Escrito lighter.

Escrito bold.

Escrito bolder.

Esta propiedad puede ser definida con valores enteros del 100 (ligero) a 900 (grueso), de acuerdo a como lo necesitemos. El valor 400 tiene el mismo grosor que el valor *normal*, mientras que 700 tiene el mismo grosor que el valor *bold*.

```
estilos.css
/* Define distintas clases con grosores de fuente en valores. */
.ligerito {
    font-weight: 300;
}
.gruesesote {
    font-weight: 850;
}
```

4.5 Variación de fuente

La propiedad **font-variant** (variación de fuente) de CSS permite encoger ligeramente un texto de un elemento o etiqueta. Acepta los siguientes valores:

- **normal**: texto con tamaño y estilo predeterminado.
- **small-caps**: texto encogido ligeramente y lo pasa a mayúsculas, tiene otros parámetros consecuentes que afectan al resultado final.
- **common-ligatures**: texto con tamaño predeterminado, pero con un espacio mayor entre letra y letra, tiene otros parámetros consecuentes que afectan al resultado final.
- **inherit, initial y unset**.

Veamos un ejemplo y su resultado (*Figura 7*):

```
estilos.css
/* Define distintas clases con variaciones de fuente distintas. */
.normal {
    font-variant: normal;
}
.small {
    font-variant: small-caps;
}

prueba.html
<p class="normal">Escrito normal.</p>
<p class="small">Escrito small-caps.</p>
```

Figure 7: Cambiando la variación de fuente

Escrito normal.

ESCRITO SMALL-CAPS.

4.6 Alineando texto horizontalmente

La propiedad **text-align** de CSS permite establecer la alineación horizontal del texto de un objeto o etiqueta. Por defecto, el texto está alineado a la izquierda, pero también acepta los siguientes valores:

- **start o left**: por defecto, alinea a la izquierda.
- **end o right**: alinea a la derecha.
- **center**: alinea al centro.

- **justify**: alinea a la izquierda y derecha.
- **inherit, initial** y **unset**.

Veamos el siguiente ejemplo y su resultado (*Figura 8*):

```
estilos.css
/* Define distintas clases con alineaciones de texto distintas. */
.izq {
    text-align: left;
}
.der {
    text-align: right;
}
.cen {
    text-align: center;
}
.jus {
    text-align: justify;
}

prueba.html
<p class="izq">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</p>
<p class="der">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</p>
<p class="cen">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</p>
<p class="jus">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</p>
```

Figure 8: Cambiando la alineación de texto

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

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

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

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

4.7 Alineando texto verticalmente

La propiedad **vertical-align** de CSS permite establecer la alineación vertical del texto de un objeto o etiqueta. Por defecto, el texto está alineado en la parte superior de la página, pero también acepta los valores:

- **top**: alinea el tope del elemento con el tope de la línea entera o su contenedor.
- **bottom**: alinea la base del elemento con la base de la línea entera o su contenedor.
- Valores con respecto al elemento padre:
 - **baseline**: alinea la base del elemento con la base de su elemento padre.
 - **sub**: alinea la base del elemento con la base del subíndice de su elemento padre.
 - **super**: alinea la base del elemento con la base del superíndice de su elemento padre.
 - **text-top**: alinea el tope del elemento con el tope de la letra de su elemento padre.
 - **text-bottom**: alinea la base del elemento con la base de la letra de su elemento padre.
 - **middle**: alinea la mitad del elemento con la base más una porción X del alto de su elemento padre.
- **inherit**, **initial** y **unset**.

Veamos el siguiente ejemplo con una tabla y su resultado (*Figura 9*):

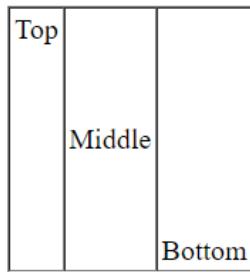
```
estilos.css
/*
Define distintas clases con alineaciones verticales de texto distintas.*/
.cima {
    vertical-align: top;
}
.medio {
    vertical-align: middle;
}
.fondo {
    vertical-align: bottom;
}

prueba.html


|     |        |        |
|-----|--------|--------|
| Top | Middle | Bottom |
|-----|--------|--------|


```

Figure 9: Cambiando la alineación vertical de texto 1



Lamentablemente, los palabras clave anteriormente mencionadas solamente aplican para las etiquetas **table**, sin embargo, podemos utilizar otras palabras reservadas como valores para otras etiquetas con esta propiedad, los cuales son: **baseline**, **sub**, **super** y **Longitudes** (% , px, pt, cm, em, etc.). Veamos un ejemplo continuación (*Figura 10*).

```
estilos.css
/*
Define distintas clases con alineaciones verticales de texto distintas.*/
.baseline {
    vertical-align: baseline;
}
.sub {
    vertical-align: sub;
}
.super {
    vertical-align: super;
}
.pixel {
    vertical-align: 10px;
}

prueba.html
<p>Este texto tiene una palabra <span class="baseline">baseline</span>. </p>
<p>Este texto tiene una palabra <span class="sub">sub</span>. </p>
<p>Este texto tiene una palabra <span class="super">super</span>. </p>
<p>Este texto tiene una palabra <span class="pixel">subida 10 píxeles</span>. </p>
```

Figure 10: Cambiando la alineación vertical de texto 2

Este texto tiene una palabra **baseline**.

Este texto tiene una palabra **sub**.

Este texto tiene una palabra **super**.

Este texto tiene una palabra **subida 10 píxeles**.

Nota: el ejemplo anterior aplicó a un párrafo (**p** y **span**), pero requiere de más instrucciones (propiedades) para otras etiquetas.

4.8 Color de texto

La propiedad **color** de CSS permite establecer un color determinado al texto de un objeto o etiqueta. Una de las formas de aplicar color al texto es utilizando las palabras claves con los nombres de los colores, por ejemplo: *red*, *green*, *blue*, etc. Veamos un ejemplo rápido (*Figura 11*):

```
estilos.css
/*
Define una clase que asigna el color rojo al texto.
*/
.colores {
    color: red;
}

prueba.html
<p class="colores">Escrito con color rojito.</p>
<p>Escrito con el color predefinido (negro).</p>
```

Figure 11: Cambiando el color del texto

Escrito con color rojito.

Escrito con el color predefinido (negro).

La otra forma de asignar colores es por medio de valores hexadecimales, compuesto por un # al inicio y seis caracteres, que van de 0 a F, o RGB, que define un valor de 0 a 255 para el rojo, verde y azul (Red, Green y Blue).

```
// Color blanco.
color: #FFFFFF;
color: rgb(255, 255, 255);
```

4.9 Decoración del texto

La propiedad **text-decoration** de CSS permite establecer una decoración al texto de un objeto o etiqueta. Algunos de los valores aceptados son:

- **none**: establece un texto regular o normal, sin estilo ni decoración.
- **overline**: establece una línea horizontal encima del texto.
- **underline**: establece una línea horizontal debajo del texto.
- **line-through**: establece una línea horizontal en medio del texto, es decir, lo tacha (sustituye la etiqueta **s** de HTML).

- **inherit**, **initial** y **unset**.

Veamos un ejemplo rápido (*Figura 12*):

```
estilos.css
/* Define una clase que asigna una decoración al texto distinta. */
.none {
    text-decoration: none;
}
.inherit {
    text-decoration: inherit;
}
.overline {
    text-decoration: overline;
}
.underline {
    text-decoration: underline;
}
.line-through {
    text-decoration: line-through;
}

prueba.html
<p class="none">Escrito predeterminado.</p>
<p class="inherit">Escrito que hereda el estilo de su padre.</p>
<p class="overline">Escrito que tiene una línea encima.</p>
<p class="underline">Escrito que tiene una linea debajo.</p>
<p class="line-through">Escrito que está tachado.</p>
```

Figure 12: Cambiando la decoración del texto

Escrito predeterminado.

Escrito que hereda el estilo de su padre.

Escrito que tiene una línea encima.

Escrito que tiene una linea debajo.

Escrito que está tachado.

Esta propiedad tiene otras propiedades consecuentes, como lo son un color, estilo o grosor, por ejemplo:

text-decoration: underline dotted red;

Los **colores** ya sabemos que pueden ser utilizados mediante palabras reservadas o códigos hexadecimales, los valores para la propiedad **estilo** son similares o iguales a los valores utilizados para utilizar los bordes de tablas (*dotted*, *solid*, *double*, etc), los valores para la propiedad **grosor** son **auto** y **from-font**.

4.10 Indentación del texto

La propiedad **text-indent** de CSS permite establecer una indentación al texto de un objeto o etiqueta. Acepta **Longitudes** como valor para definir el espacio del lado izquierdo antes de comenzar el texto (px, pt, cm, em, %, **inherit**, **initial** y **unset**, valores negativos, etc.). Veamos un ejemplo rápido (*Figura 13*):

```
estilos.css
/* Define una clase que asigna una indentación al texto. */
p {
    text-indent: 20px;
}

prueba.html
<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</p>
```

Figure 13: Cambiando la indentación del texto

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

4.11 Sombra del texto

La propiedad **text-shadow** de CSS permite establecer una sombra proveniente del centro del texto de un objeto o etiqueta. Posee cuatro parámetros: distancia del centro del texto con respecto al eje X (horizontal), distancia del centro del texto con respecto al eje Y (vertical), lo borroso o difuminado de la sombra y su color; todos estos parámetros **no** van separados por comas. Veamos un ejemplo rápido (*Figura 14*):

```
estilos.css
/* Define una clase que asigna una una sombra con color al texto. */
.sombrita {
    color: blueviolet;
    /* La sombra se proyecta en diagonal 5 y 5 píxeles hacia la abajo y derecha,
       está difuminada 5 píxeles y es de color azul. */
    text-shadow: 5px 5px 5px blue;
}

prueba.html
<h1 class="sombrita">Este texto tiene sombra</h1>
```

Figure 14: Agregando sombra al texto

Este texto tiene sombra

Como podemos suponer, los valores de los atributos pueden ser negativos y las distintas **Longitudes** permitidas por CSS, podemos utilizar valores hexadecimales o RGB para establecer el color de la sombra y podemos establecer múltiples sombras a un texto, simplemente separamos este grupo de parámetros por una coma y ponemos otro grupo de cuatro:

```
text-shadow: 5px 5px 5px blue, rgba(0,0,255,1) -1px -2px 0.5em;
```

Un valor que también se puede poner es **transparent**, que hace transparente una sombra, además de **inherit**, **initial** y **unset**.

4.12 Transformación del texto

La propiedad **text-transform** de CSS permite configurar la forma en la que aparecen las mayúsculas y minúsculas en el texto de un objeto o etiqueta. Algunos valores que acepta esta propiedad son:

- **none**: aplica un estilo normal o predeterminado a un texto.
- **capitalize**: pone mayúsculas a cada palabra de un texto.
- **uppercase**: vuelve todo el texto mayúscula.
- **lowercase**: vuelve todo el texto minúscula.
- **inherit**, **initial** y **unset**.

Veamos un ejemplo rápido (*Figura 15*):

```
estilos.css
/*
Define una clase que cambia el orden de minúsculas y mayúsculas del texto.
*/
.none {
    text-transform: none;
}
.capitalize {
    text-transform: capitalize;
}
.uppercase {
    text-transform: uppercase;
}
.lowercase {
    text-transform: lowercase;
}

prueba.html
<p class="none">eScRiT0 n0Ne</p>
<p class="capitalize">eScRiT0 CaPiTaLiZe</p>
<p class="uppercase">eScRiT0 upperCASE</p>
<p class="lowercase">eScRiT0 LOWERcase</p>
```

Figure 15: Cambiando las mayúsculas y minúsculas del texto

eScRiTOnONe
EScRiTOCaPiTaLiZe
ESCRITO UPPERCASE
escrito lowercase

4.13 Espaciado entre letras

La propiedad **letter-spacing** de CSS permite configurar el espaciado entre carácter y carácter de un texto de un objeto o etiqueta. Algunos valores que acepta esta propiedad son:

- **normal**: no agrega ningún espaciado entre caracteres.
- **inherit**, **initial** y **unset**.

Veamos un ejemplo rápido (*Figura 16*):

```
estilos.css
/* Define una clase que cambia el espaciado de caracteres del texto. */
.normal {
    letter-spacing: normal;
}
.esp {
    letter-spacing: 5px;
}

prueba.html
<p class="normal">Espaciado normal.</p>
<p class="esp">Espaciado de 5px.</p>
```

Figure 16: Cambiando el espaciado de caracteres del texto

Espaciado normal.
Espaciado de 5px.

Nota: letter-spacing acepta valores negativos.

4.14 Espaciado entre palabras

La propiedad **word-spacing** de CSS permite configurar el espaciado entre palabras de un texto de un objeto o etiqueta. Así como con la propiedad anterior, esta acepta los valores **normal**, una **Longitud** (%), px, pt, cm, em, etc.), **inherit**, **initial** y **unset**. Veamos un ejemplo rápido (*Figura 17*):

```
estilos.css
/*
Define una clase que cambia el espaciado de palabras del texto.*/
.normal {
    word-spacing: normal;
}
.esp {
    word-spacing: 20px;
}

prueba.html
<p class="normal">Este párrafo tiene un espaciado de palabras normal.</p>
<p class="esp">Este párrafo tiene un espaciado de palabras de 20 píxeles.</p>
```

Figure 17: Cambiando el espaciado de palabras del texto

Este párrafo tiene un espaciado de palabras normal.

Este párrafo tiene un espaciado de palabras de 20 píxeles.

Nota: word-spacing acepta valores negativos.

4.15 Manejo de espacios y saltos de línea

La propiedad **white-space** de CSS permite configurar cómo son manejados los espacios en blanco y saltos de línea en textos, tanto en el código, como en el navegador cuando este cambia de tamaño, en otras palabras, maneja como se ajusta el texto cuando un elemento o etiqueta HTML sufre un cambio de tamaño. Los valores que puede aceptar son los siguientes:

- **normal:** el texto en código con secuencias de espacios y saltos de línea son reducidos a un solo espacio, manteniendo el texto dentro del espacio de la etiqueta que lo contiene.
- **nowrap:** trata los espacios y saltos de líneas como el valor *normal*, pero el texto se sale del espacio de la etiqueta que lo contiene si esta no es lo suficientemente grande como para almacenarlo.
- **pre:** los múltiples espacios continuos se mantienen y los saltos de línea se dan con etiquetas **br** y con los caracteres de saltos de línea.
- **pre-wrap:** trata los espacios y saltos de líneas como el valor anterior, pero rellena con saltos de línea según sea necesario, para llenar el contenedor que lo almacena.
- **pre-line:** trata los saltos de líneas como el valor anterior, pero la secuencia de espacios son reducidos a un solo espacio.
- **break-spaces:** idéntico al valor *pre-wrap*, pero mantiene todos los espacios (incluidos los espacios al final de líneas) y secuencias de espacios.

La *Tabla 2* muestra las diferentes situaciones donde podemos utilizar esta propiedad:

Table 2: Usos de la propiedad *white-space*

	Saltos de líneas	Espacios y tabulación	Ajuste de texto	Espacios al final	Separadores
normal	Se reducen	Se reducen	Se ajusta	Son removidos	Se cuelga
nowrap	Se reducen	Se reducen	No se ajusta	Son removidos	Se cuelga
pre	Se mantienen	Se mantienen	No se ajusta	Se mantienen	No se ajusta
pre-wrap	Se mantienen	Se mantienen	Se ajusta	Se cuelga	Se cuelga
pre-line	Se mantienen	Se reducen	Se ajusta	Son removidos	Se cuelga
break-spaces	Se mantienen	Se mantienen	Se ajusta	Se cuelga	Se ajusta

Veamos un ejemplo rápido (*Figura 18*):

```
estilos.css
/*
Define una clase que manipula distintos espaciados y saltos de líneas del texto.
*/
/* Estilo de apoyo para visualizar mejor el ejemplo. */
p {
    border-color: black;
    border-style: solid;
}
.normal {
    white-space: normal;
}
.nowrap {
    white-space: nowrap;
}
.pre {
    white-space: pre;
}
.pre-wrap {
    white-space: pre-wrap;
}
.pre-line {
    white-space: pre-line;
}
.break-spaces {
    white-space: break-spaces;
}

prueba.html
<p class="normal">Este párrafo tiene
    un espaciado blanco de palabras
    normal.
</p>
<p class="nowrap">Este párrafo tiene
    un espaciado blanco de palabras
```

```
<i>nowrap</i>.  
</p>  
<p class="pre">Este párrafo tiene  
    un espaciado blanco de palabras  
    <i>pre</i>.  
</p>  
<p class="pre-wrap">Este párrafo tiene  
    un espaciado blanco de palabras  
    <i>pre-wrap</i>.  
</p>  
<p class="pre-line">Este párrafo tiene  
    un espaciado blanco de palabras  
    <i>pre-line</i>.  
</p>  
<p class="break-spaces">Este párrafo tiene  
    un espaciado blanco de palabras  
    <i>break-spaces</i>.  
</p>
```

Figure 18: Cambiando el manejo de espacios y saltos de líneas del texto

Este párrafo tiene un espaciado blanco de palabras normal.

Este párrafo tiene un espaciado blanco de palabras *nowrap*.

Este párrafo tiene
 un espaciado blanco de palabras
 pre.

Este párrafo tiene
 un espaciado blanco de palabras
 pre-wrap.

Este párrafo tiene
 un espaciado blanco de palabras
 pre-line.

Este párrafo tiene
 un espaciado blanco de palabras
 break-spaces.

5 Posicionamiento

Se refiere a la posibilidad de posicionar un elemento HTML en la página del sitio web. La propiedad para posicionar un elemento es **position**, la cual tiene algunos valores importantes a conocer:

- **static**: se mantiene fijo según el flujo de elementos de la página, aún después de deslizarse hacia abajo en el sitio. Es es valor por defecto y no se puede posicionar fuera del flujo de elementos.
- **fixed**: se posiciona en una parte en específico de la página, aún después de deslizarse hacia abajo en el sitio, por lo que no sigue el flujo de elementos, esto puede provocar **superposición de elementos**.
- **relative**: se posiciona en una parte en específico de la página, pero se mantiene fija en dicha posición después de deslizar hacia abajo en el sitio (puede causar superposición de elementos).
- **absolute**: se posiciona en una parte en específico de la página, tomando como referencia la posición de su etiqueta padre más próxima, si no tiene, tomara esta referencia del documento en sí (body); se mantiene fija en dicha posición después de deslizar hacia abajo en el sitio (puede causar superposición de elementos).

Para posicionar un elemento en alguna parte en específico de la página, requerimos de las propiedad *top*, *left*, *bottom*, *right* y las *unidades de valores*. Dejaremos un ejemplo sobre los posicionamientos de etiquetas y su resultado en la *Figura 19*:

```
estilos.css
/* Se mantiene fijo en el flujo del documento. */
.static p {
    position: static;
    /* Se ignoran las propiedades "top" y "left" debido a la posición estática. */
    top: 200px;
    left: 150px;
    width: 100px;
    height: 100px;
    background-color: red;
}
/* Se posiciona donde se deseé, fuera del flujo del documento. */
.fixed p {
    position: fixed;
    /* A 150px y 150px de la esquina superior izquierda. */
    top: 150px;
    left: 150px;
    width: 100px;
    height: 100px;
    background-color: blue;
    color: white;
}
/* Se posiciona donde se deseé, dentro del flujo del documento. */
.relative {
    position: relative;
```

```
/* A 100px y 50px de la esquina superior izquierda. */
top: 100px;
left: 50px;
width: 100px;
height: 100px;
background-color: coral;
}
/*
Se posiciona donde se desee y con respecto a su etiqueta padre,
en este caso, "body", fuera del flujo del documento.
*/
.absolute {
    position: absolute;
    /* A 100px de la esquina superior izquierda. */
    top: 100px;
    width: 100px;
    height: 100px;
    background-color: darkgreen;
}
body {
    height: 5000px;
}

prueba.html
<div class="static">
    <p>Este párrafo tiene static position.</p>
</div>
<div class="fixed">
    <p>Este párrafo tiene fixed position.</p>
</div>
<div class="relative">
    <p>Este párrafo tiene relative position.</p>
</div>
<div class="absolute">
    <p>Este párrafo tiene absolute position.</p>
</div>
```

Figure 19: Diferencia entre valores de propiedad *position*

Debe ejecutar este código en su buscador, ya que las diferencias visuales solamente se pueden apreciar cuando se desplaza sobre la página web.

6 Floating

La propiedad **float** de CSS permite que los elementos alrededor de otro "floten" o lo rodeen (usualmente utilizado en imágenes). Los valores que se aceptan son:

- **left**: los elementos flotan a la izquierda.
- **right**: los elementos flotan a la derecha.
- **none**: los elementos no flotan (valor predeterminado).
- **inline-start**: los elementos flotan al principio de un contenedor o elemento padre.
- **inline-end**: los elementos flotan al final de un contenedor o elemento padre.
- **inherit**, **initial** y **unset**.

Los elementos que tiene la propiedad *float* se posicionan en el extremo izquierdo o derecho (dependiendo si se estableció el valor *left* o *right* respectivamente) del contenedor donde están almacenados (body, div, section, etc.), y el resto de elementos los rodean. Dejaremos un ejemplo sobre los elementos flotantes y su resultado en la *Figura 20*:

```
estilos.css
/*
Muestra los elementos contenidos como bloques.
*/
.block {
    display: block;
```

```
        height: 200px;
    }
    /* Los elementos flotan a la izquierda. */
    .left {
        float: left;
    }
    /* Los elementos flotan a la derecha. */
    .right {
        float: right;
    }

prueba.html
<div class="block">
    <div class="left">
        
    </div>
    <p>Este párrafo flota al lado derecho de la imagen.</p>
</div>
<div class="block">
    <div class="right">
        
    </div>
    <p>Este párrafo flota al lado izquierdo de la imagen.</p>
</div>
```

Figure 20: Diferencia entre valores de propiedad *float*

Este párrafo flota al lado derecho de la imagen.

Este párrafo flota al lado izquierdo de la imagen.



En caso de que existan múltiples elementos con la propiedad *float* uno al lado del otro,

estos se posicionarán sin más uno enseguida del otro.

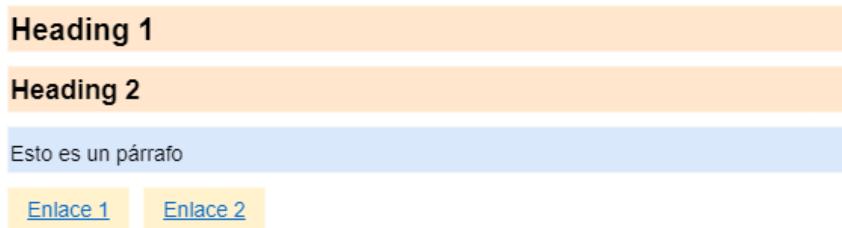
7 Propiedades para la visualización de elementos

Recordemos que todo elemento HTML es una caja, con su margen, *padding*, borde y contenido; el buscador visualizará estos elementos de dos maneras principales:

- **block-element**: elementos bloques, el buscador pondrá los elementos uno encima de otro (apilados), abarcando el 100% del ancho de la página, a pesar de que el contenido de la etiqueta no lo abarque. Por lo general, muchas etiquetas tienen este tipo de visualización por defecto.
- **inline-elements**: elementos en línea, el buscador pondrá los elementos uno al lado del otro, ocupando solamente el ancho que requiera su contenido. Por lo general, las etiquetas **a** tienen este tipo de visualización por defecto.

Los dos conceptos anteriores servirán para explicar el ejemplo y figura del tema anterior, así como posteriores. La *Figura 21* muestra la apariencia de ambas visualizaciones:

Figure 21: Aspecto de las visualizaciones *block* e *inline*



Veremos entonces a continuación como modificar esta visualización con CSS.

7.1 display

La propiedad **display** de CSS acepta los siguientes valores:

- **block**: visualiza los elementos HTML como elementos bloque, pudiendo cambiar su ancho y alto.
- **inline**: visualiza los elementos HTML como elementos en línea, no pudiendo cambiar su ancho y alto.

- **none**: no visualiza el elemento HTML, lo esconde, como si no estuviera presente, mostrando únicamente el contenedor que lo almacena.
- **inline-block**: visualiza los elementos HTML como elementos en línea, pero teniendo la posibilidad de asignar un ancho o alto al elemento.

Dejaremos un ejemplo sobre los posicionamientos de etiquetas y su resultado en la *Figura 22*, inspeccionando cada elemento para que se note la diferencia entre todas:

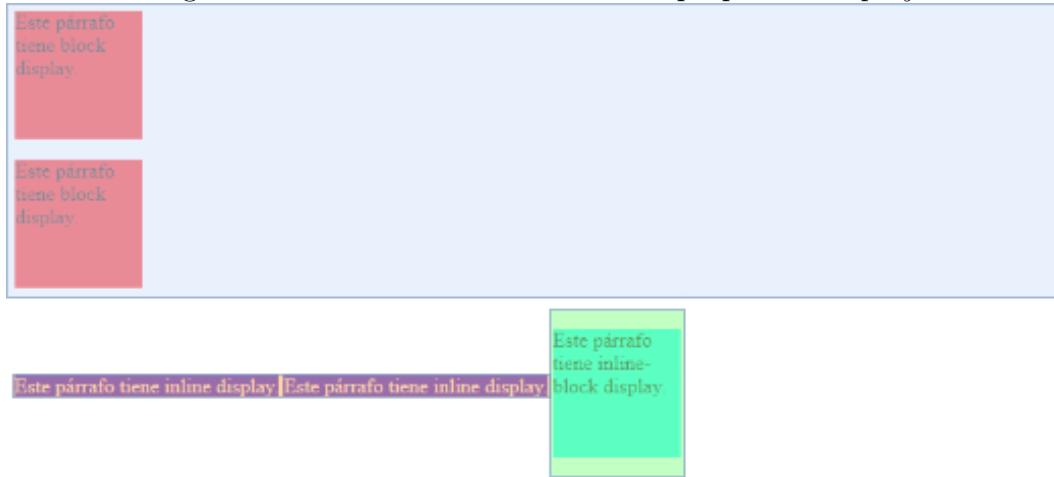
```
estilos.css
/*
Visualiza el elemento como bloque.
*/
.block {
    display: block;
}
.block p {
    /* Hereda la visualización del bloque de su parente. */
    display: inherit;
    /* Establece tamaño al elemento. */
    width: 100px;
    height: 100px;
    background-color: red;
}
/*
Visualiza el elemento como en linea.
*/
.inline {
    display: inline;
}
.inline p {
    display: inherit;
    width: 100px;
    height: 100px;
    background-color: blue;
    color: white;
}
/*
Visualiza el elemento como descartado.
*/
.none {
    display: none;
}
/*
Visualiza el elemento como bloque en linea.
*/
.inline-block {
    display: inline-block;
}
.inline-block p {
    display: inherit;
    width: 100px;
    height: 100px;
    background-color: aqua;
}

prueba.html
<div class="block">
    <p>Este párrafo tiene block display.</p>
    <p>Este párrafo tiene block display.</p>
</div>
<div class="inline">
```

```

<p>Este párrafo tiene inline display.</p>
<p>Este párrafo tiene inline display.</p>
</div>
<div class="none">
    <p>Este párrafo no se incluirá en la visualización.</p>
</div>
<div class="inline-block">
    <p>Este párrafo tiene inline-block display.</p>
</div>

```

Figure 22: Diferencia entre valores de propiedad *display*

Vea que, a pesar de haber definido un tamaño para los elementos *inline*(*width*=100px y *height*=100px), estos no sufren dichos cambios a su tipo de display, si tuvieran el valor *block* o *inline-block*, podríamos alterar su tamaño. Por otro lado, el elemento con display *none* no aparece en el resultado del ejemplo, debido a que este no posee un espacio por abarcar en la visualización de la página, existe en el DOM, pero no abarca espacio visualmente.

7.2 visibility

La propiedad **visibility** de CSS esconde un elemento HTML, pero mantiene su espacio requerido, a diferencia de *display: none*, que esconde el elemento y omite su espacio requerido. Los valores aceptados de esta propiedad son:

- **hidden**: esconde el elemento.
- **visible**: muestra el elemento.
- **collapse**: para elementos regulares, este valor se comporta como *hidden*; para las columnas y grupo de columnas y filas, este valor esconde la celda y el espacio que debería tener (como *display: none*); para elementos *collapsed flex*, este valor esconde los elementos y el espacio que deberían tener.
- **inherit, initial y unset**.

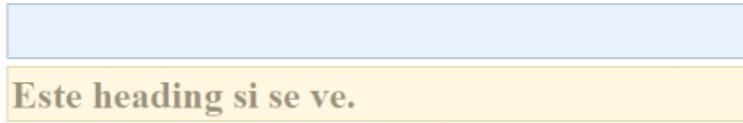
Dejamos un ejemplo y su resultado en la *Figura 23*:

```
estilos.css
/*
 * Esconde la visibilidad del elemento.
 */
.hidden {
    visibility: hidden;
}

/*
 * Muestra la visibilidad del elemento.
 */
.visible {
    visibility: visible;
}

prueba.html
<div class="hidden">
    <h1>Este heading no se ve.</h1>
</div>
<div class="visible">
    <h1>Este heading si se ve.</h1>
</div>
```

Figure 23: Diferencia entre valores de propiedad *visibility*



Como podemos ver, esta propiedad esconde el elemento, pero mantiene su espacio en la página, a diferencia de la propiedad *display none*.

7.3 clear

La propiedad **clear** de CSS ayuda a evitar que los elementos seguidos a uno que tiene la propiedad *float* activen flotación, es decir, **clear** evita que un elemento flote alrededor de otro. Los valores aceptados son:

- **right**: evita flotar a la derecha de elementos con *float*.
- **left**: evita flotar a la izquierda de elementos con *float*.
- **both**: evita flotar en ambos lados de elementos con *float*.
- **none**: no evita flotar alrededor de elementos (valor predeterminado).
- **inherit**, **initial** y **unset**.

Dejamos un ejemplo y su resultado en la *Figura 24*:

```
estilos.css
/*
Los elementos flotan a la derecha de un elemento.*/
.floating {
    float: right;
}
/*
Los elementos evitan flotar en ambos sentidos de un elemento.*/
.clearing {
    clear: both;
}
/*
Agrega borde sólido de 1 píxel a la imagen.*/
img {
    border-style: solid;
    border-width: 1px;
}

prueba.html
This paragraph is above the div element
and is not affected by the float right property.
<br/><br/>
<div class="floating">
    
</div>
This paragraph comes after the div element
and is affected by the float right property.
<div class="clearing">
    This paragraph is out of the floating group
    and is not affected by the float right property.
</div>
```

Figure 24: Diferencia entre valores de propiedad *clear*

This paragraph is above the div element and is not affected by the float right property.

This paragraph comes after the div element
and is affected by the float right property.



This paragraph is out of the floating group and is not affected by the float right property.

7.4 overflow

Recordemos que cada elemento HTML está contenido en una caja, cuando no establecemos el alto de la caja, este se ve ajustado automáticamente al contenido de la caja, sin embargo, si establecemos un alto y el contenido sobrepasa esta medida, se dice que ocurre un **desbordamiento** (*overflow*), entonces, CSS tiene la propiedad **overflow**, que permite manejar el comportamiento del desbordamiento de contenido de los elementos. Los valores que acepta son:

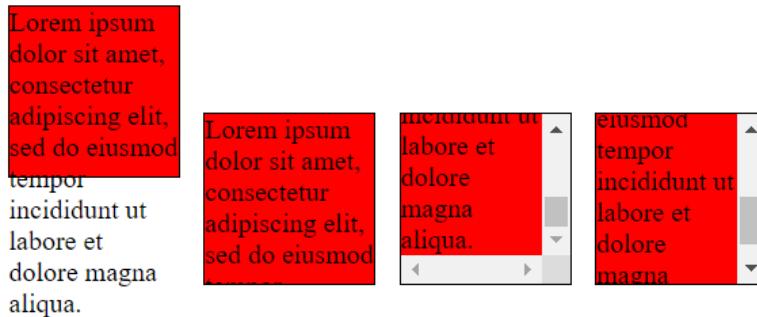
- **visible**: muestra completo el contenido desbordado del elemento (valor predeterminado).
- **scroll**: el contenido que sobresale es recortado y añade una barra de desplazamiento vertical y horizontal al elemento para ver el contenido que se desborda.
- **hidden**: el contenido que sobresale es recortado y no puede ser visto.
- **auto**: si el contenido es recortado, se añade una barra de desplazamiento vertical al elemento para ver el contenido que se desborda.
- **inherit, initial y unset**.

Dejamos un ejemplo y su resultado en la *Figura 25*:

```
estilos.css
/*
  Pone los elementos uno al lado del otro.
*/
.inline-block {
  display: inline-block;
}
/*
  Agrega un margen izquierdo de 10 píxeles.
*/
p {
  margin-left: 10px;
}
/*
  Clases que reciben la propiedad "display" de su parente, se agrega un
  tamaño, borde y color para visualizar mejor el elemento, y se establece
  la propiedad "overflow".
*/
.visible {
  display: inherit;
  width: 100px;
  height: 100px;
  border-style: solid;
  border-width: 1px;
  background-color: red;
  overflow: visible;
}
.hidden {
  display: inherit;
  width: 100px;
  height: 100px;
  border-style: solid;
  border-width: 1px;
```

```
background-color: red;
overflow: hidden;
}
.scroll {
    display: inherit;
    width: 100px;
    height: 100px;
    border-style: solid;
    border-width: 1px;
    background-color: red;
    overflow: scroll;
}
.auto {
    display: inherit;
    width: 100px;
    height: 100px;
    border-style: solid;
    border-width: 1px;
    background-color: red;
    overflow: auto;
}

prueba.html
<div class="inline-block">
    <p class="visible">Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.</p>
    <p class="hidden">Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.</p>
    <p class="scroll">Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.</p>
    <p class="auto">Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.</p>
</div>
```

Figure 25: Diferencia entre valores de propiedad *overflow*

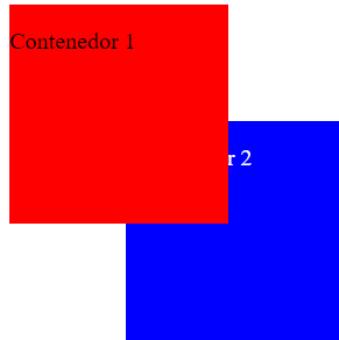
7.5 z-index

Cuando una etiqueta se sobrepone en otra es debido a que, en el DOM y código HTML, la sobrepuesta va después de la que está por debajo, esto siempre ocurre, pero podemos cambiar el orden en el que se posicionan los elementos sobrepuertos. La propiedad **z-index** de CSS recibe como valores números enteros y da como resultado un ordenamiento de elementos sobrepuertos distinta. Veamos el siguiente código y su resultado (*Figura 26*):

```
estilos.css
/*
El primer contenedor tiene el "z-index" de 2.
*/
.cont1 {
    position: fixed;
    top: 100px;
    left: 100px;
    width: 150px;
    height: 150px;
    background-color: red;
    /* "z-index" 2. */
    z-index: 2;
}

/*
El segundo contenedor tiene el "z-index" de 1.
*/
.cont2 {
    position: fixed;
    top: 180px;
    left: 180px;
    width: 150px;
    height: 150px;
    background-color: blue;
    color: white;
    /* "z-index" 1 */
    z-index: 1;
}

prueba.html
<div class="cont1">
    <p>Contenedor 1</p>
</div>
<div class="cont2">
    <p>Contenedor 2</p>
</div>
```

Figure 26: Funcionamiento de la propiedad *z-index*

Como vemos, el primer div con la clase "cont1" aparece, después, el siguiente div con la clase "cont2" aparece, ambos con la propiedad *position* "fixed", el segundo div se sobrepone al primero por defecto, sin embargo, el estilo CSS le da a la clase "cont1" el valor 2 de *z-index* y el valor 1 a la clase "cont2", por lo que "cont1" se sobrepone. No olvidar que la propiedad *z-index* solamente funciona con elementos con la propiedad *position* y el valor *absolute*, *relative* y *fixed*.

8 CSS3

CSS3 es la versión más reciente de CSS, compatible con versiones anteriores del mismo, las características nuevas más significantes son el poder aplicar un radio e imagen a los bordes de elementos, varios fondos, animaciones y efectos.

8.1 Prefijos del proveedor o buscador

Los **prefijos del proveedor (o del buscador)** son instrucciones utilizadas para aplicar una propiedad a un buscador que no soporta dicha propiedad, por ejemplo, puede que Safari no soporte la propiedad *border-radius*, por lo que hay que establecer el prefijo de Safari, seguido de la propiedad que deseamos utilizar en el sitio web, con esto, ya podremos trabajar con el radio de bordes de elementos. Los prefijos de algunos buscadores son:

- **-webkit-**: prefijo para Safari, Chrome y buscadores que utilizan el motor *Webkit* (iOS y Android).
- **-moz-**: prefijo para Firefox y buscadores que utilizan el motor de búsqueda de este.
- **-o-**: prefijo para Opera.
- **-ms-**: prefijo para Internet Explorer y Edge.

La sintaxis es la siguiente:

```
-webkit-border-radius: 8px;  
-o-border-radius: 8px;  
-moz-border-radius: 8px;  
-ms-border-radius: 8px;
```

Nota: actualmente, muchos buscadores ya integran todas las propiedades y características de CSS3, sin embargo, es recomendable conocer estos prefijos para tener una noción de que algunas propiedades tal vez no estén disponibles en el buscador donde estamos desarrollando un proyecto.

8.2 Esquinas redondeadas

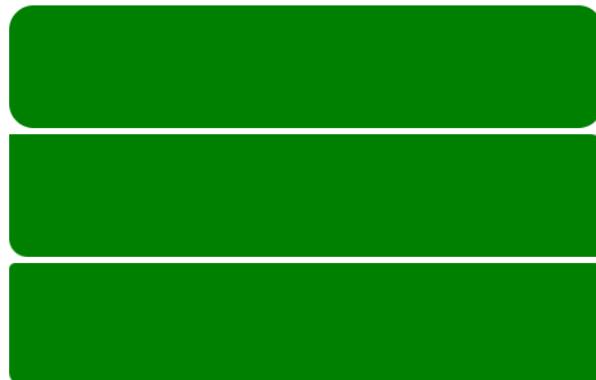
La propiedad **border-radius** de CSS permite redondear todas las esquinas de un elemento. Podemos asignar un radio distinto por cada esquina con las propiedades **border-top-left-radius**, **border-top-right-radius**, **border-bottom-right-radius**, **border-bottom-left-radius**; esta propiedad cuenta con un acceso directo, para evitar escribir todas las propiedades anteriormente mencionadas una por una, como vemos en el siguiente ejemplo y su resultado en la *Figura 27*:

```
estilos.css  
/* Redondea todas las esquinas del contenedor en una sola instrucción. */  
.todo-redondeado {  
    display: inherit;  
    background-color: green;
```

```
color: white;
padding: 50px;
margin-bottom: 5px;
border-radius: 20px;
}
/* Redondea algunas esquinas del contenedor en varias instrucciones.*/
.separado-redondeado {
    display: inherit;
    background-color: green;
    color: white;
    padding: 50px;
    margin-bottom: 5px;
    border-top-right-radius: 10px;
    border-bottom-left-radius: 15px;
}
/* Redondea todas las esquinas del contenedor en varias instrucciones.*/
.todo-sep-redo {
    display: inherit;
    background-color: green;
    color: white;
    padding: 50px;
    margin-bottom: 5px;
    border-radius: 5px 7px 9px 11px;
}
.redondeado {
    display: block;
}

prueba.html
<div class="redondeado">
    <div class="todo-redondeado"></div>
    <div class="separado-redondeado"></div>
    <div class="todo-sep-redo"></div>
</div>
```

Figure 27: Diferencia entre valores de propiedad *border-radius*



Como podemos ver, los valores que acepta esta propiedad son **Longitudes** (% , px, pt, cm, em, etc.).

8.3 Sombras

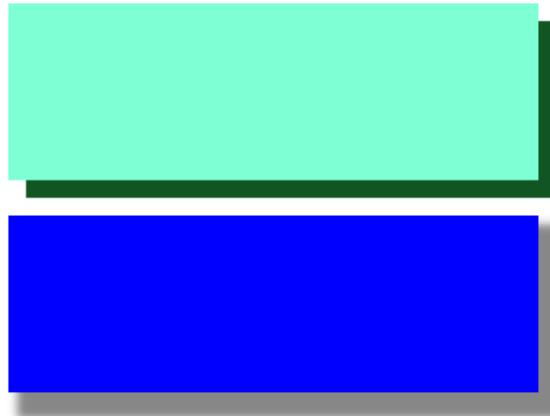
La propiedad **box-shadow** de CSS permite agregar sombra a un elemento HTML. La forma en la que esta sombra es procesada por el buscador mediante los valores aceptados es la siguiente: el primer valor es el **largo de la sombra sobre el eje horizontal** del elemento, el segundo valor es el **largo de la sombra en el eje vertical** del elemento y el tercer valor es el **color de la sombra** (opcional). Además a estos valores, podemos agregar dos valores más: el **difuminado y propagación de la sombra** (*blur* y *spread*), como su nombre lo indica, modifican el difuminado de la sombra, y cómo esta se propaga alrededor del elemento. Veamos dos ejemplos, el primero sin difuminado ni propagación, el segundo si lo tiene, el resultado está en la *Figura 28*:

```
estilos.css
/*
 * Agrega sombra sin difuminar.
 */
.bs1 {
    width: 300px;
    height: 100px;
    background-color: aquamarine;
    box-shadow: 10px 10px #152;
    margin-bottom: 20px;
}

/*
 * Agrega sombra difuminada.
 */
.bs2 {
    width: 300px;
    height: 100px;
    background-color: blue;
    box-shadow: 10px 10px 5px 5px #878787;
}

pruebas.html
<div class="redondeado">
    <div class="bs1"></div>
    <div class="bs2"></div>
</div>
```

Figure 28: Diferencia entre valores de propiedad *box-shadow*



Nota: asignar valores negativos a la propiedad ocasionará que la sombra se posicione en la esquina superior izquierda, no en la esquina inferior derecha.

8.3.1 Técnicas para el sombreado

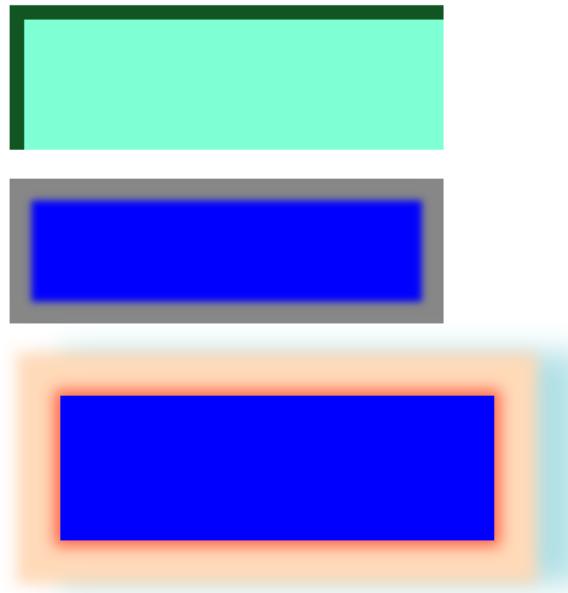
Anteriormente vimos como crear una sombra para un elemento, pero solamente una, esta pudiendo tener su origen en dos ubicaciones distintas fuera del elemento, con la palabra reservada **inset** podemos crear una sombra dentro del elemento: esta palabra reservada va enseguida del nombre de la propiedad, y podemos asignarle valores como vimos en el tema anterior. Veamos la instrucción:

```
box-shadow: inset 10px 10px 5px #888888;
```

Esta instrucción creará una sombra interior ubicada en la esquina superior izquierda. Si queremos poner más sombras a un elemento, las podemos separar con una coma (,), como se ve en el siguiente ejemplo y resultado en la *Figura 29*

```
estilos.css
/*
 * Agrega una sombra interna sin difuminar.
 */
.bs1 {
    width: 300px;
    height: 100px;
    background-color: aquamarine;
    box-shadow: inset 10px 10px #152;
    margin-bottom: 20px;
}
/*
 * Agrega una sombra interna completa semi difuminada.
 */
.bs2 {
    width: 300px;
    height: 100px;
    background-color: blue;
    box-shadow:
        inset 10px 10px 5px 5px #878787,
        inset -10px -10px 5px 5px #878787;
    margin-bottom: 50px;
}
/*
 * Agrega varias sombras externas completas difuminadas.
 */
.bs3 {
    width: 300px;
    height: 100px;
    background-color: blue;
    box-shadow:
        0 0 10px 4px #ff6347,
        0 0 10px 30px #ffdab9,
        30px 0 20px 30px #b0e0e6;
    margin-bottom: 20px;
    margin-left: 35px;
}

prueba.html
<div class="bs1"></div>
<div class="bs2"></div>
<div class="bs3"></div>
```

Figure 29: Diferencia entre valores de propiedad *box-shadow*

Como podemos ver, el primer rectángulo solo tiene una sombra interna, el segundo posee dos sombras internas, y el tercero posee tres sombras externas, una encima de otra, siguiendo el orden del código mostrado. Nótese que no existe una palabra reservada para sombras externas.

8.3.2 Sombras en texto

La propiedad **text-shadow** permite agregar una sombra al texto de un objeto o etiqueta HTML. Cuenta con cuatro valores: sombra horizontal, sombra vertical, difuminado y color; las sombras horizontal y vertical posiciona la sombra por debajo del texto en dichos ejes, podemos hacer que la sombra sea completamente definida o difuminada, y por último, podemos darle un color. Los primeros tres valores aceptan *Longitudes* (%), px, pt, cm, em, etc. También el valor *none* para no asignar alguna sombra al texto), mientras que el último valor acepta valores de las distintas *Herramientas de colores* con las que cuenta CSS.

Al igual que con *box-shadow*, podemos crear varias sombras en el texto (inclusive negativas), veamos un ejemplo y su resultado en la *Figura 30*:

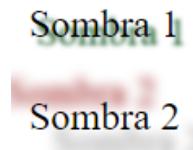
```
estilos.css
/*
 * Agrega una sombra.
 */
.s1 {
    text-shadow: 3px 3px 2px #152;
    margin-bottom: 20px;
}
/*
 * Agrega dos sombras.
 */
.s2 {
    text-shadow:
        10px 10px 5px #878787,
        -10px -10px 5px #900000;
    margin-bottom: 50px;
```

```

}

prueba.html
<p class="s1">Sombra 1</p>
<p class="s2">Sombra 2</p>

```

Figure 30: Diferencia entre valores de propiedad *text-shadow*

Solo recuerde que las sombras se ven más realistas si:

- una sombra cercana no está tan difuminada,
- una sombra lejana si está más difuminada

8.4 Transparencia

Previo a CSS3, para lograr que un elemento tuviera transparencia, se utilizaba la propiedad *background-image* con una imagen en formato PNG, ahora, hay otros métodos para lograr este cometido. Con esta versión de CSS, se soporta ahora nombres de colores, hexadecimales, RGB y dos nuevas herramientas, que la *Tabla 3* muestra:

Table 3: Nuevas herramientas para colores en CSS3

Herramienta	Definición	Ejemplo
Colores RGBA	Extensión de los colores RGB con un valor alfa , el cual especifica la opacidad del color	rgba(204, 29, 29, 1.0);
Colores HSL	Colores Hue , Saturation , Lightness (Matiz o color, Saturación, Luminosidad o claridad), es otra forma de expresar colores: el primero representa el color (los valores van de 0 a 360, donde 0 y 360 es rojo, 120 es verde y 240 es azul, en base a esto, se juega con estos valores para obtener el color deseado), el segundo la saturación (se mide en porcentajes, 100% es el color completo y como tal) y el tercero la claridad del color (0% es oscuro o negro, 100% es blanco)	hsl(0, 10%, 60%);
Colores HSLA	Extensión de los colores HSL con un valor alfa , el cual especifica la opacidad del color	hsla(147, 50%, 47%, 0.5);

Veamos el resultado de los ejemplos anteriores en la *Figura 31*:

```
estilos.css
/*
Establece color con RGBA y sin transparencia (1.0). */
#rgba1 {
    width: 100px;
    height: 100px;
    background-color: rgba(204, 29, 29, 1.0);
    margin-bottom: 10px;
}
/*
Establece color con RGBA y con transparencia de 0.1. */
#rgba2 {
    width: 100px;
    height: 100px;
    background-color: rgba(161, 30, 30, 0.1);
    margin-bottom: 10px;
}
/*
Establece color con HSL y con transparencia del 60%. */
#hsl1 {
    width: 100px;
    height: 100px;
    background-color: hsl(0, 10%, 60%);
    margin-bottom: 10px;
}
/*
Establece color con HSLA y con transparencia de 0.5. */
#hsl2 {
    width: 100px;
    height: 100px;
    background-color: hsla(147, 50%, 47%, 0.5);
}

prueba.html
<div id="rgba1"></div>
<div id="rgba2"></div>
<div id="hsl1"></div>
<div id="hsl2"></div>
```

Figure 31: Uso de RGBA, HSL y HSLA



8.5 Pseudo clases

Bien vimos que, para estilizar un elemento específico, le asignamos un Id o Clase, y dicha Clase o Id, se le aplican instrucciones CSS. En caso de que se tengan múltiples etiquetas dentro de otra, cada una de ellas debería tener una Clase o Id para ser modificadas por separado.

Es aquí donde entran las **Pseudo clases**, estas permiten modificar un elemento sin tener que asignarle una Clase o Id ni utilizar JavaScript. Existen muchas Pseudo clases, para estos apuntes, hablaremos de *:first-child* y *:last-child*:

- **:first-child:** aplica un estilo al primer elemento dentro de otro.
- **:last-child:** aplica un estilo al último elemento dentro de otro.

Veamos un ejemplo y su resultado en la *Figura 32*:

```
estilos.css
/* Clase padre con color de texto rojo. */
.pseudo {
color: red;
}
/* Cambia el color de texto del primer hijo de la clase padre. */
.pseudo p:first-child {
color: blue;
}
/* Cambia el color de texto del último hijo de la clase padre. */
.pseudo p:last-child {
color: violet;
```

```
}
```

```
prueba.html
<div class="pseudo">
    <p>Este es el primer párrafo.</p>
    <p>Este es el segundo párrafo.</p>
    <p>Este es el tercer párrafo.</p>
    <p>Este es el cuarto párrafo.</p>
    <p>Este es el quinto párrafo.</p>
    <p>Este es el sexto párrafo.</p>
</div>
```

Figure 32: Uso de las Pseudo clases

Este es el primer párrafo.

Este es el segundo párrafo.

Este es el tercer párrafo.

Este es el cuarto párrafo.

Este es el quinto párrafo.

Este es el sexto párrafo.

Estas dos Pseudo clases son de las más utilizadas, pero existen más, te invito a investigar qué utilidad tienen el resto de Pseudo clases disponibles.

8.6 Pseudo elementos

Otra gran utilidad de estas Pseudo herramientas disponibles en CSS son los **Pseudo elementos**, estos permiten estilizar una parte o elemento de una etiqueta HTML. Las Pseudo clases utilizan los *dos puntos* (:), los Pseudo elementos utilizan *doble dos puntos* (::) para ser trabajados. Los cinco Pseudo elementos más utilizados son:

- **::first-line**: selecciona la primer línea de un texto en una etiqueta.
- **::first-letter**: selecciona la primer letra de un texto o palabra en una etiqueta.
- **::selection**: toma el texto seleccionado de una etiqueta por el usuario
- **::before**: inserta un contenido antes de una etiqueta.
- **::after**: inserta un contenido después de una etiqueta.

Algunos de estos Pseudo elementos no están disponibles en ciertos navegadores, por lo que tendrá que acudir a los **prefijos de buscadores**. Veamos un ejemplo de estos Pseudo elementos y su resultado en la *Figura 33*:

```
estilos.css
/* Cambia el color de la primer letra. */
p::first-letter {
    color: blue;
}
/* Cambia el color de la primer línea del texto. */
p::first-line {
    color: red;
}
/* Cambia el color del texto seleccionado. */
p::selection {
    background-color: green;
    color:white;
}
/* Agrega algo antes de un elemento. */
p::before {
    content: url(bola.png);
}
/* Agrega algo después de un elemento. */
p::after {
    content: url(bola.png);
}

prueba.html
<div class="pseudo">
    <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Assumenda distinctio ullam
        delectus porro odit sequi aperiam, amet commodi molestias beatae veniam obcaecati
        similiqe sint nulla, consequuntur veritatis, magni autem est.</p>
    <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Assumenda distinctio ullam
        delectus porro odit sequi aperiam, amet commodi molestias beatae veniam obcaecati
        similiqe sint nulla, consequuntur veritatis, magni autem est.</p>
    <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Assumenda distinctio ullam
        delectus porro odit sequi aperiam, amet commodi molestias beatae veniam obcaecati
        similiqe sint nulla, consequuntur veritatis, magni autem est.</p>
    <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Assumenda distinctio ullam
        delectus porro odit sequi aperiam, amet commodi molestias beatae veniam obcaecati
        similiqe sint nulla, consequuntur veritatis, magni autem est.</p>
    <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Assumenda distinctio ullam
        delectus porro odit sequi aperiam, amet commodi molestias beatae veniam obcaecati
        similiqe sint nulla, consequuntur veritatis, magni autem est.</p>
</div>
```

Figure 33: Uso de los Pseudo elementos



En el caso de *first-letter* esta se ve sobre escrita por *first-line*, por lo cual no se aprecia en la figura anterior. Al igual que con las Pseudo clases, se le invita a investigar más Pseudo elementos existentes.

8.7 word-wrap

La propiedad **word-wrap** de CSS permite establecer si una palabra da un salto de línea cuando topa con el límite de su etiqueta o contenedor; los valores permitidos son:

- **normal**: el texto mantiene su largo.
- **break-word**: el texto da un salto de línea.
- **inherit, initial y unset**.

Veamos un ejemplo claro de esta propiedad y su resultado en la *Figura 34*:

```
estilos.css
/* Clase que no da salto de línea. */
.wwn {
    width: 200px;
    height: 100px;
    border: 2px solid black;
    word-wrap: normal;
}
/* Clase que no da salto de línea. */
.wwww {
    width: 200px;
    height: 100px;
    border: 2px solid black;
```

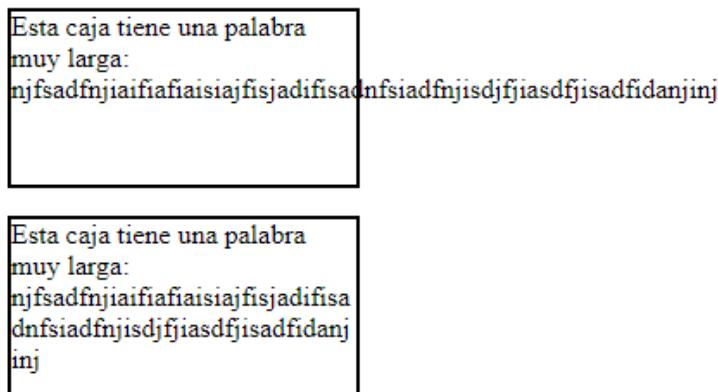
```

        word-wrap: break-word;
    }

prueba.html
<p class="wwn">Esta caja tiene una palabra muy larga:
    njfsadfnjiaifafiaisiajfisjadifisadnfsiadfnjisdfjisadfidanjinj</p>
<p class="www">Esta caja tiene una palabra muy larga:
    njfsadfnjiaifafiaisiajfisjadifisadnfsiadfnjisdfjisadfidanjinj</p>

```

Figure 34: Diferencia entre valores de propiedad *word-wrap*



8.8 @font-face

La regla **@font-face** permite descargar u obtener una fuente o tipo de letra de un servidor y establecerla en todo nuestro sitio web, de esta manera, las fuentes disponibles en el sitio ya no están limitadas en las que están instaladas en la máquina del usuario que accede al sitio.

Esta regla trabaja con los formatos **Embedded OpenType** (eot), **True Type Fonts** (ttf) y **OpenType Fonts** (otf), los siguientes navegadores aceptan los formatos mencionados:

- Firefox, Safari, Chrome y Opera: .ttf y .otf.
- Internet Explorer: .eot

Para utilizar esta regla, debe escribir el nombre de la regla (@font-face), abrir llaves, darle un nombre a la fuente, la ruta donde está almacenada la fuente y alguna personalización extra, como se ve a continuación:

```

estilos.css
/* Define dos reglas iguales pero con distinto estilo. */
@font-face {
    font-family: Delicious;
    src: url('Delicious-Roman.otf');
}
@font-face {
    font-family: Delicious;
    font-weight: bold;
    src: url('Delicious-Bold.otf');
}

```

```
}

/* Asigna las reglas a las etiquetas "h1". */
h1 {
    font-family: Delicious, sans-serif;
}

prueba.html
<h1>Hola mundo</h1>
```

En caso de que queramos que todo el sitio tenga la nueva fuente, en vez de asignarla a la etiqueta "h1", se la asignamos a "body."

Las fuentes más utilizadas actualmente son las de Google, esto debido a que es por este buscador que los usuarios se conectan a Internet y realizan sus búsquedas, por lo que es bueno utilizar dichas fuentes para cargar nuestro sitio web, puede ver todas las fuentes disponibles en este enlace (aunque tiene una distinta implementación).

Puede buscar fuentes con los formatos anteriormente mencionados en Internet y aplicarlas al sitio web, como se ve en el ejemplo siguiente y su resultado en la *Figura 35*:

```
estilos.css
/* Asigna la fuente importada a las etiquetas "h1". */
h1 {
    font-family: 'Open Sans', sans-serif;
}

prueba.html
<h1>Hola mundo</h1>
```

Figure 35: Uso de la regla *@font-face*

Hola mundo

9 Degradados

CSS3 permite que se establezca una transición suave o lisa entre dos o más colores. Estos tipos de transiciones o degradados están categorizados en dos tipos: **lineal** y **radial**.

9.1 Degradado lineal

Para crear un degradado lineal, se utiliza el método **linear-gradient()**, se deben establecer dos **escalas de colores**, estas dos escalas de colores renderizarán la transición suave; a estos degradados se le puede establecer un punto de origen y una dirección (o un ángulo). Veamos un ejemplo y su resultado en la *Figura 36*:

```
estilos.css
div {
    width: 300px;
    height: 100px;
    color: gray;
    /* Degrado lineal blanco y negro. */
    background: -moz-linear-gradient(black, white);
    font-size: larger;
}

prueba.html
<div>
    Hola mundo
</div>
```

Figure 36: Utilizando el degradado lineal simple



9.1.1 Prefijos, múltiples escalas y presencia en el degradado

El ejemplo anterior tiene el **prefijo del buscador** de Mozilla Firefox, para que este degradado funcione dentro de dicho buscador, pero puede ser agregado, removido o cambiado por otro prefijo en caso de ser requerido. Para el caso de estos apuntes y prácticas, los degradados fueron creados con el buscador **Microsoft Edge**, el cual no requirió de un prefijo.

En el caso anterior, utilizamos solamente dos escalas de colores con sus respectivas palabras reservadas, pero podemos agregar más de dos, simplemente se agregan separando las escalas por comas, de igual manera, podemos tener, por ejemplo, cinco colores, pero queremos que el cuarto sea el que aparezca más en el degradado, podemos conseguir esto agregando una **Longitud** (%), px, pt, cm, em, etc.), como se ve a continuación (*Figura 37*):

```
estilos.css
div {
```

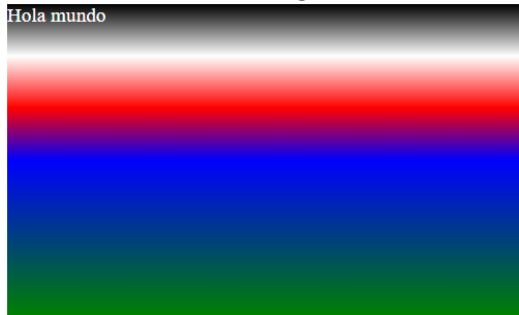
```

width: 500px;
height: 300px;
color: white;
/* Degradado lineal blanco, negro, rojo y azul al 50% y añade verde. */
background: linear-gradient(black, white, red, blue 50%, green);
font-size: larger;
}

prueba.html
<div>
  Hola mundo
</div>

```

Figure 37: Utilizando el degradado lineal múltiple



Como vemos, el cuarto color (azul) es el que tiene más presencia, debido a que le asignamos que tuviera una presencia del 50%.

9.1.2 Dirección o ángulo del degradado

Podemos lograr que el degradado comience desde distintas **ubicaciones** del elemento: izquierda, derecha, arriba o abajo, las palabras reservadas **left**, **right**, **top** y **bottom** respectivamente (también podemos utilizar las esquinas del elemento, por ejemplo, la esquina superior izquierda, que sería **top left**).

Considere que, si no utiliza los *prefijos de un buscador*, debe agregar la palabra reservada **to** previo al valor de dirección predeterminado escogido, si sí va a utilizar un prefijo, no agregue **to**.

```

// Sin prefijo:
background: linear-gradient(to left, red, blue);
// Con prefijo:
background: -moz-linear-gradient(left, red, blue);

```

Por otro lado, una alternativa a estas palabras predefinidas son los ángulos, que puede ir desde 0 a 360 grados, si escribimos el valor *0deg*, creará un degradado de izquierda a derecha, mientras que si escribimos el valor *90deg*, creará un degradado de abajo a arriba, experimente con los distintos valores para crear un degradado de su agrado.

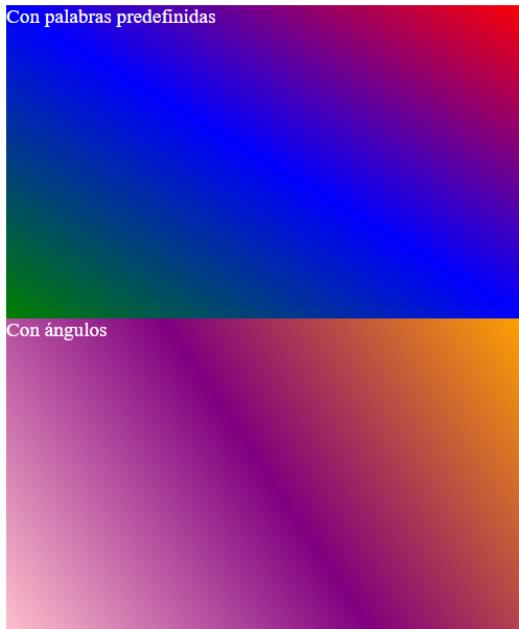
Veamos como se ven estas dos formas de dirección en el siguiente ejemplo y su resultado en la *Figura 38*:

```
estilos.css
#dir {
    width: 500px;
    height: 300px;
    color: white;
    /* Dirección de degradado con palabras reservadas. */
    background: linear-gradient(to bottom left, red, blue, green);
    font-size: larger;
}

#ang {
    width: 500px;
    height: 300px;
    color: white;
    /* Dirección de degradado con ángulos. */
    background: linear-gradient(56deg, pink, purple, orange);
    font-size: larger;
}

prueba.html
<div id="dir">
    Con palabras predefinidas
</div>
<div id="ang">
    Con ángulos
</div>
```

Figure 38: Utilizando el degradado lineal con dirección



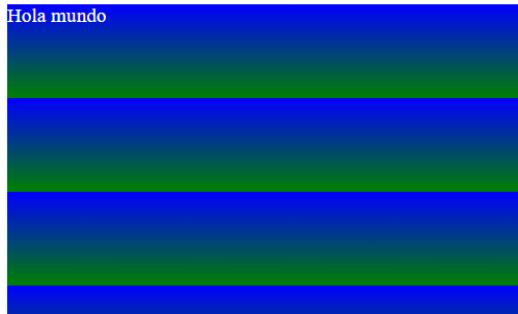
9.1.3 Repetir el degradado

Puede repetir un degradado en un mismo elemento con el método **repeating-linear-gradient()**, añadiendo una **Longitud** (% , px, pt, cm, em, etc.) después de la escala de color. Esta propiedad también acepta las direcciones de degradado. Veamos un ejemplo y su resultado en la *Figura 39*:

```
estilos.css
div {
    width: 500px;
    height: 300px;
    color: white;
    /* Repite el degradado azul y verde a un 30%. */
    background: repeating-linear-gradient(blue, green 30%);
    font-size: larger;
}

prueba.html
<div>
    Hola mundo
</div>
```

Figure 39: Repitiendo un degradado lineal



Tomamos el 30% del tamaño del elemento y metemos ahí al degradado, así con el siguiente 30% y así sucesivamente.

9.2 Degradado radial

La otra manera de definir un degradado a un elemento es por medio del método **radial-gradient()**, el cual tiene la siguiente sintaxis:

*radial-gradient(forma tamaño **at** posición, escalas de colores);*

Donde:

- **forma**: indica la forma del degradado. Los valores aceptados son *ellipse* (por defecto) y *circle*.
- **tamaño**: indica el tamaño del degradado. Los valores aceptados son *farthest-corner* (por defecto), *closest-side*, *closest-corner* y *farthest-side*.

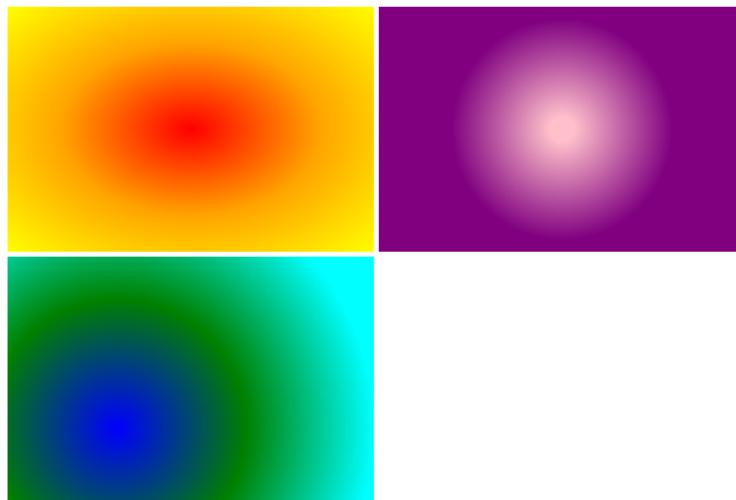
- **posición:** indica donde comienza el degradado. Se pueden utilizar palabras reservadas (top, bottom, left, right o una combinación de estas) o Unidades de valores, la unidad más común a utilizar es el porcentaje (%), por ejemplo, 0%, 0% hace que el degradado comience en la esquina superior izquierda; 50%, 50% hace que comience a la mitad. El valor por defecto es centrado (si es que este parámetro es omitido).
- **escala de colores:** indica los colores del degradado; se pueden poner más de dos, separados por comas, además, se puede repetir los colores como con el estilo de degradado anterior.

Mostramos a continuación algunos ejemplos en la *Figura 40*:

```
estilos.css
/* Despliega los contenedores uno al lado del otro. */
.cont {
    display: inline-block;
}
/* Crea contenedores de tamaño fijo, con degradado radial de tres colores y forma de
   ellipse. */
.rad1 {
    display: inherit;
    width: 300px;
    height: 200px;
    background-image: radial-gradient(red, orange, yellow);
}
/* Degradado con repetición de colores y forma circular. */
.rad2 {
    display: inherit;
    width: 300px;
    height: 200px;
    background-image: radial-gradient(circle, pink 10px, purple 91px);
}
/* Degradado de cierto tamaño y posición con forma circular. */
.rad3 {
    display: inherit;
    width: 300px;
    height: 200px;
    background-image: radial-gradient(circle farthest-side at 30% 70%, blue, green, aqua);
}

prueba.html
<div class="cont">
    <div class="rad1"></div>
    <div class="rad2"></div>
    <div class="rad3"></div>
</div>
```

Figure 40: Utilizando el degradado radial



Como se puede apreciar, se combina la sintaxis de esta función en los tres ejemplos anteriores.

10 Fondos

10.1 background-size

La propiedad **background-size** de CSS permite cambiar el tamaño del fondo de un elemento mediante el escalado de la imagen. La mayoría de buscadores ya soportan esta propiedad, por lo que no es necesario agregar un *Prefijo del buscador*. Esta propiedad acepta Unidades de valores o tres palabras reservadas:

- **auto**: por defecto, establece el tamaño de la imagen del fondo tal cual es.
- **contains**: hace que el alto de la imagen del fondo y el alto del elemento HTML sean el mismo. Si el ancho de la imagen de fondo es menor al del elemento, repite la imagen, si no lo es, se ajusta.
- **cover**: Hace que el ancho de la imagen del fondo y el alto del elemento HTML sean el mismo. Si el alto de la imagen de fondo es menor al del elemento, se ajusta, si no lo es, se recorta la imagen con respecto al tamaño del elemento.
- **Longitudes (%)**, px, pt, cm, em, etc.).
- **inherit**, **initial** y **unset**.

La diferencia entre *contains* y *cover* se puede ver a continuación en la *Figura 41*:

```
estilos.css
/* Despliega los contenedores uno al lado del otro. */
.cont {
    display: inline-block;
```

```

}

.contains {
    display: inherit;
    /* Tamaño de la clase para exemplificar la propiedad "background-size". */
    width: 400px;
    height: 250px;
    background-image: url(gato.png);
    background-size: contain;
}

.cover {
    display: inherit;
    width: 400px;
    height: 250px;
    background-image: url(gato.png);
    background-size: cover;
}

prueba.html
<div class="cont">
    <div class="contains"></div>
    <div class="cover"></div>
</div>

```

Figure 41: Diferencia entre los valores *contains* y *cover*



Como vemos, el ancho del primer elemento es mayor a la imagen de fondo que se le establece, por lo que esta imagen se repite; para el segundo contenedor, el ancho de la imagen y el elemento se ajusta, pero el alto de la imagen de fondo es mayor al alto de su elemento, por lo que la imagen se termina cortando y no mostrándose completamente, esta es la principal diferencia entre ambos valores de la propiedad.

10.2 background-clip

La propiedad **background-clip** de CSS delimita hasta donde se colorea o se aplica una imagen al fondo de un elemento. Los valores aceptados son:

- **border-box**: valor por defecto, colorea o aplica una imagen a un fondo por completo, incluyendo el borde del elemento.
- **padding-box**: colorea o aplica una imagen a un fondo, descartando el padding del elemento.

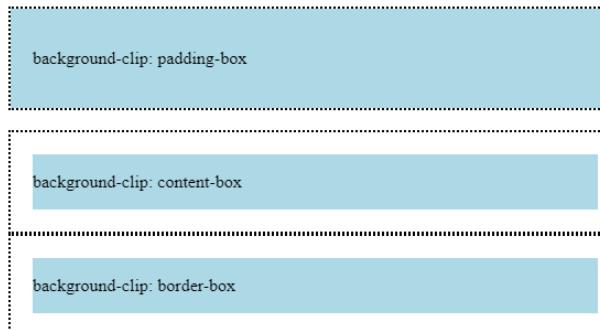
- **content-box**: colorea o aplica una imagen a un fondo por completo, excluyendo el borde del elemento.
- **initial, unset y inherit**.

La diferencia de estos valores se aprecia en la *Figura 42*

```
estilos.css
/*
Establece un tipo de borde, padding, color de fondo iguales
para cada div y la propiedad "background-clip" distinta.*/
#first {
    border: 2px dotted black;
    padding: 20px;
    background: LightBlue;
    background-clip: padding-box;
}
#second {
    border: 2px dotted black;
    padding: 20px;
    background: LightBlue;
    background-clip: content-box;
}
#third {
    border: 2px dotted black;
    padding: 20px;
    background: LightBlue;
    background-clip: border-box;
}

prueba.html
<div id="first">
    <p>background-clip: padding-box</p>
</div>
<div id="second">
    <p>background-clip: content-box</p>
</div>
<div id="third">
    <p>background-clip: border-box</p>
</div>
```

Figure 42: Diferencia entre los valores *border-box*, *padding-box* y *content-box*



10.3 Bordes transparentes

Un borde transparente para un elemento se consigue combinando la propiedad `background-clip` y el `RGBA` (*Figura 43*):

```
estilos.css
div {
    /* Borde de 20px de grosor negro y con transparencia de 0.1. */
    border: 20px solid rgba(0, 0, 0, 0.1);
    /* Colorea el borde. */
    background-clip: padding-box;
    /* Posiciona el div. */
    position: absolute;
    top: 50px;
    left: 50px;
    width: 200px;
    /* Fondo blanco. */
    background-color: white;
}

prueba.html
<div>In the screenshot, we set the background-clip to padding-box. The white background ends before the border and the transparency lays over the content</div>

Some text Some text
Some text Some text Some text Some text Some text Some text Some text Some text Some text
Some text Some text Some text Some text Some text Some text Some text Some text Some text
Some text Some text Some text Some text Some text Some text Some text Some text Some text
Some text Some text Some text Some text Some text Some text Some text Some text Some text
Some text Some text Some text Some text Some text Some text Some text Some text Some text
Some text Some text Some text Some text Some text Some text Some text Some text Some text
Some text Some text Some text Some text Some text Some text Some text Some text Some text
Some text Some text Some text Some text Some text Some text Some text Some text Some text
Some text Some text Some text Some text Some text Some text Some text Some text Some text
Some text Some text Some text Some text Some text Some text Some text Some text Some text
```

Figure 43: Consiguiendo un borde transparente

Some text
 Some text Some text Some text Some text Some text Some text Some text Some text Some text Some text
 Some text Some text Some text Some text Some text Some text Some text Some text Some text Some text
 Some text Some text Some text Some text Some text Some text Some text Some text Some text Some text
 Some text Some text Some text Some text Some text Some text Some text Some text Some text Some text
 Some text Some text Some text Some text Some text Some text Some text Some text Some text Some text
 Some text Some text Some text Some text Some text Some text Some text Some text Some text Some text
 Some text Some text Some text Some text Some text Some text Some text Some text Some text Some text

In the screenshot, we set the background-clip to padding-box. The white background ends before the border and the transparency lays over the content

10.4 Varias imágenes de fondo

Podemos poner múltiples fondos a un elemento creando una lista de elementos separados por comas, recordemos que existe el método `url()`, la cual permite que pongamos un fondo de Internet o del directorio donde estemos trabajando. Las imágenes se van poniendo de **arriba a abajo**, por lo que la primera irá hasta arriba y la última hasta abajo, como en el siguiente ejemplo (*Figura 44*):

```
estilos.css
div {
```

```
border: 1px solid black;
width: 400px;
height: 300px;
/* Varias imágenes separadas por comas (,). */
background-image: url("bola.png"), url("gato.png");
background-position: top, bottom;
background-repeat: no-repeat;
}

prueba.html
<div></div>
```

Figure 44: Múltiples fondos



10.4.1 background-position

Como pudimos ver en el ejemplo anterior, nos saltamos la regla de que los múltiples fondos van de arriba a abajo, los posicionamos distinto mediante una lista separada por comas, esto gracias a la propiedad **background-position**, esta nos permite establecer la posición de inicio de una imagen de fondo. Los valores aceptados son:

- **left top, left center y left bottom**: comenzando desde la izquierda a arriba, en medio o abajo respectivamente.
- **right top, right center y right bottom**: comenzando desde la derecha a arriba, en medio o abajo respectivamente
- **center top, center center y center bottom**: comenzando del centro a arriba, en medio o abajo respectivamente.
- **Longitudes (%)**, px, pt, cm, em, etc.).
- **initial, unset y inherit**.

Los valores con palabra reservada se pueden mezclar, es por eso que los pusimos directamente como *left top*, *right center* o *center bottom*, pero también puede utilizarlos como palabra sola (*top*, *left*, *right* y **bottom**).

También puede utilizar la propiedad recortada *background* para establecer la imagen, su posición y si se repite o no:

```
background: url([imagen]) [posición] [repetir]
```

10.5 opacity

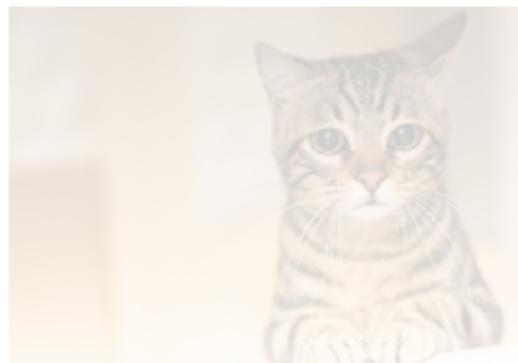
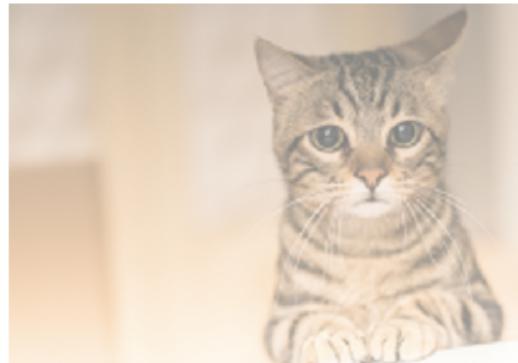
La propiedad **opacity** permite volver transparente una imagen de manera sencilla, ya que su valor tope es 1 (opaco) y su valor mínimo es 0 (completamente transparente), por lo que debe establecer un valor decimal entre ambos topes, como se ve en la *Figura 45*:

```
estilos.css
/*
Imagen con opacidad de 0.5.
*/
#img1 {
    width: 400px;
    height: 400px;
    background-image: url("gato.png");
    background-repeat: no-repeat;
    opacity: 0.5;
}

#img2 {
/*
Imagen con opacidad de 0.2.
*/
    width: 400px;
    height: 400px;
    background-image: url("gato.png");
    background-repeat: no-repeat;
    opacity: 0.2;
}

prueba.html
<div id="img1"></div>
<div id="img2"></div>
```

Figure 45: Transparencia de imágenes



11 Transiciones

Las **transiciones** nos permiten pasar de un valor de propiedad a otro dada una duración. La sintaxis básica es la siguiente:

transition: propiedad duración

Donde:

- **propiedad**: especifica la propiedad que tendrá la transición.
- **duración**: especifica el tiempo de duración de la transición.

La propiedad *transition* de CSS se le debe asignar al elemento que se desea que tenga una animación, pero de momento todavía no hace nada; en caso de que se omita alguno de los dos valores básicos anteriores, la animación no funcionará. Para hacer que la animación se haga, se debe establecer un nuevo valor para el valor *propiedad* de *transition*, esto debe ser fuera del elemento a animar, como por ejemplo, si hacemos que el puntero del mouse se ponga encima del elemento, cambie de tamaño:

```
div {  
    /* Valor inicial de la propiedad a animar. */  
    width: 100px;  
    height: 100px;  
    background: red;  
    /* Establece la propiedad a animar y la duración. */  
    transition: width 2s;  
}  
div:hover {  
    /* Valor final de la propiedad a animar. */  
    width: 300px;  
}
```

Como vemos en el ejemplo anterior, el elemento tiene un valor inicial, mientras que el mismo, pero con un evento distinto, tiene su valor final, esta estructura conforma la animación del elemento. En este caso, cuando se quite el puntero del mouse de encima del elemento, este volverá a su estado inicial.

En caso de animar varias propiedades de un elemento, simplemente se separan por comas:

```
div {  
    /* Valor inicial de las propiedades a animar. */  
    width: 100px;  
    height: 100px;  
    background: red;  
    /* Establece las propiedades a animar y la duración. */  
    transition: width 2s, height 2s;  
}  
div:hover {  
    /* Valor final de las propiedades a animar. */  
    width: 300px;  
    height: 300px;  
}
```

11.1 Velocidad de la transición

La propiedad **transition-timing-function** de CSS permite cambiar la velocidad en la que la transición se crea dentro de la duración establecida, es decir, una animación puede durar cinco segundos en concretarse, pero esta puede comenzar lento, luego rápido y finalizar lento nuevamente, vemos los valores de esta propiedad:

- **ease**: la transición comienza lento, luego rápido y termina lento (valor predeterminado).
- **linear**: la transición tiene la misma rapidez de comienzo a fin.
- **ease-in**: la transición tiene un comienzo lento.
- **ease-out**: la transición tiene un final lento.
- **ease-in-out**: la transición tiene un comienzo y final lento.
- **cubic-bezier(*n,n,n,n*)**: define tus propios valores para una función cúbica bezier (valores aceptados en cada parámetro: 0 y 1).
- **step-start**: la transición avanza según una cantidad *n* de pasos, en este caso, solamente tiene un paso y comienza del lado izquierdo del elemento.
- **step-end**: la transición avanza según una cantidad *n* de pasos, en este caso, solamente tiene un paso y comienza del lado derecho del elemento.
- **steps(*n*, *término*)**: define tus propios valores para una función de *steps* con *n* cantidad de pasos y un término en el que se darán los pasos.
- **inherit**, **initial** y **unset**.

11.2 Retraso de la transición

La propiedad **transition-delay** de CSS agrega un tiempo de retraso a la transición antes de que comience:

transition-delay: 2s;

Entonces, la propiedad *transition* puede ser especificada como:

```
div {  
    transition-property: width;  
    transition-duration: 2s;  
    transition-timing-function: linear;  
    transition-delay: 1s;  
}
```

O con su acceso directo:

transition: width 2s linear 1s;

12 Transformaciones

Una **transformación** es un efecto que permite cambiar la forma, tamaño y posición de un elemento. CSS permite transformaciones 2D y 3D. Las transformaciones se consiguen mediante la propiedad **transform**.

12.1 rotate()

El método **rotate()** permite rotar un elemento x unidades de ángulos, recordemos las **Unidades de Ángulos**:

- **deg**: grados, de 0 a 360 en un círculo.
- **grad**: gradianes, de 0 a 400 en un círculo.
- **rad**: radianes, hay 2π

(aproximadamente 6.28) en un círculo.

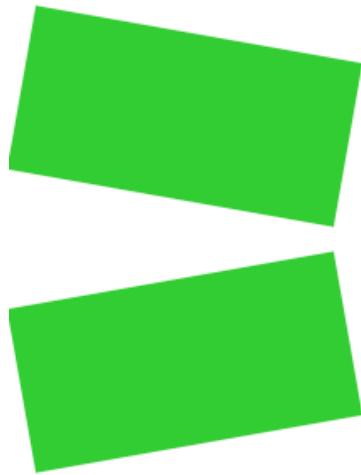
- **turn**: vueltas, hay 1 vuelta en un círculo. No es muy soportado por los buscadores.

Las figuras rotadas con valores positivos rotan en el sentido de las manecillas del reloj, si utilizas valores negativos, rotarán en sentido contrario. En la *Figura 46* hay un ejemplo y su resultado:

```
estilos.css
/*
Clase con rotación positiva.
*/
.positive {
    width: 200px;
    height: 100px;
    margin-top: 30px;
    background-color: #32CD32;
    transform: rotate(10deg);
}

/*
Clase con rotación negativa.
*/
.negative {
    width: 200px;
    height: 100px;
    margin-top: 30px;
    background-color: #32CD32;
    transform: rotate(-10deg);
}

prueba.html
<div class="positive"></div>
<br/>
<div class="negative"></div>
```

Figure 46: Método **rotate()** para rotar un elemento

12.2 transform-origin()

El método **transform-origin()** permite cambiar la posición del eje del elemento transformado. Recordemos que cada elemento tiene un punto de origen el cual es el que se mueve entre transformaciones o animaciones, es como en vez de empujar o mover el elemento de sus contornos o márgenes, lo empujamos o movemos desde su eje, este eje es el centro del elemento por valor inicial. Este eje o punto de origen puede ser modificado mediante valores predeterminados, porcentaje o un largo, como vemos a continuación:

- Valores aceptados para el eje X:
 - **left, center, right** (izquierda, centro y derecha respectivamente), **porcentaje (%)** y una Longitud (px, pt, cm, em, etc.).
 - **initial** y **unset**.
- Valores aceptados para el eje y:
 - **top, center, bottom** (parte superior, centro y parte inferior respectivamente), **porcentaje (%)** y una Longitud (px, pt, cm, em, etc.).
 - **initial, inherit** y **unset**.

Se suele trabajar con porcentajes generalmente, pero puede combinar los valores predefinidos para conseguir la posición deseada, utilizaremos el ejemplo anterior para rotar dos elementos, pero cambiando el eje de ambos para que se aprecie cómo se ve la propiedad (*Figura 47*):

```
estilos.css
/*
Clase con rotación positiva con origen al 20 y 20 porciento.
*/
.positive {
    width: 200px;
    height: 100px;
    margin-top: 30px;
```

```
background-color: #32CD32;
transform: rotate(10deg);
transform-origin: 20% 20%;
}
/* Clase con rotación negativa con origen al 0 y 90 porciento. */
.negative {
    width: 200px;
    height: 100px;
    margin-top: 30px;
    background-color: #32CD32;
    transform: rotate(-10deg);
    transform-origin: 0% 90%;
}

prueba.html
<div class="positive"></div>
<br/>
<div class="negative"></div>
```

Figure 47: Función **transform-origin()** para cambiar el eje de un elemento



Vemos entonces que los elementos cambian un poco su posición, esto es porque el eje modificado con *transform-origin* está en otra ubicación.

12.3 translate()

El método **translate()** permite mover un elemento a una nueva posición según los valores dados para el eje X y eje Y. Valores positivos empujarán el elemento hacia abajo y hacia la derecha, mientras que los valores negativos hacia arriba y hacia la izquierda. El ejemplo en la *Figura 48* muestra un elemento en su posición por defecto y la misma, pero trasladada unos píxeles:

```
estilos.css
/* Clase para contenedor original. */
.default {
    width: 260px;
```

```
height: 50px;
margin-top: 30px;
background-color: palegoldenrod;
transform-origin: bottom center;
}
/* Clase para contenedor movido 200px abajo y 100 a la derecha */
.translated {
width: 260px;
height: 50px;
background-color: palegoldenrod;
transform-origin: bottom center;
transform: translate(100px, 200px);
}

prueba.html
<div class="default"></div>
<div class="translated"></div>
```

Figure 48: Método **translate()** para mover un elemento



Los elementos pueden ser movidos con las propiedades *position*, *top* y *left* en conjunto y con los *márgenes* del elemento, sin embargo, **translate** se utiliza en conjunto con las animaciones.

12.4 skew()

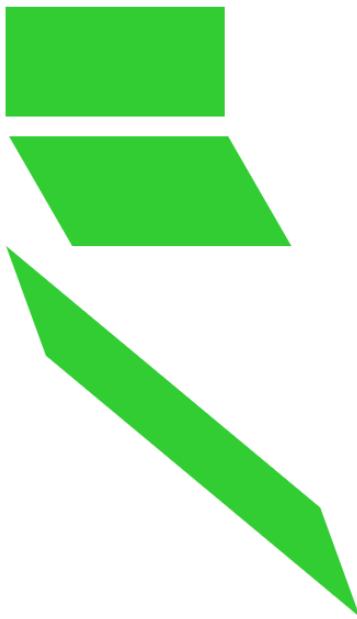
El método **skew()** (inclinar o inclinación) permite alterar la perspectiva de la figura en cuanto a su inclinación. Podemos hacer que la figura sea inclinada por medio de sus ejes x, y o por una cantidad de ángulos, como se ve en el ejemplo en la *Figura 49*:

```
estilos.css
/* Clase para contenedor original. */
.default {
width: 100px;
padding: 50px;
background-color: #32CD32;
}
/* Clase para contenedor con inclinación de 30 grados. */
.skewed_ang {
position: fixed;
```

```
    left: 40px;
    width: 100px;
    padding: 50px;
    background-color: #32CD32;
    transform: skew(30deg);
}
/* Clase para contenedor con inclinación de 20 grados en el eje X
   y 50 grados en el eje Y. */
.skewed_xy {
    position: fixed;
    top: 345px;
    left: 70px;
    width: 100px;
    padding: 50px;
    background-color: #32CD32;
    transform: skewX(20deg) skewY(50deg);
}

prueba.html
<div class="default"></div>
<br/>
<div class="skewed_ang"></div>
<br/>
<div class="skewed_xy"></div>
```

Figure 49: Método `skew()` para inclinar un elemento



El método `skew()` solamente recibe una cantidad de grados, mientras que `skewX()` y `skewY()` reciben también una sola cantidad de grados para el eje X, Y respectivamente, es así como se modifica la inclinación del elemento.

12.5 scale()

El método **scale()** permite incrementar o decrementar el tamaño de un elemento. Este método acepta dos parámetros, uno para el ancho y el otro para el alto, donde el valor 1 es el tamaño representa el tamaño original del elemento, 2 el doble y así sucesivamente, puede utilizar valores decimales, como se ve en la *Figura 50*:

```
estilos.css
/*
Clase para contenedor original.
*/
.origin {
    width: 200px;
    height: 100px;
    background-color: #8BC34A;
    color:white;
}
/*
Clase para contenedor con escala.
*/
.first {
    width: 200px;
    height: 100px;
    background-color: #8BC34A;
    color:white;
    transform: scale(0.7, 0.7);
}
/*
Clase para contenedor con escala.
*/
.second {
    margin: 60px;
    width: 200px;
    height: 100px;
    background-color: #8bc34a;
    color:white;
    transform: scale(1.5,1.5);
}

prueba.html
<div class="origin">Hola mundo</div>
<div class="first">Hola mundo</div>
<div class="second">Hola mundo</div>
```

Figure 50: Método **scale()** para cambiar escala de un elemento



El ejemplo anterior utiliza el mismo valor tanto para el ancho como para el alto, pero se pueden utilizar distintos valores. En caso de que se omita el segundo parámetro, se tomará el valor del primero para el segundo.

13 Animaciones

13.1 Keyframes & animaciones

La regla **keyframes** permite crear animaciones que pueden ser asignadas a un elemento mediante la propiedad **animation**. Estas animaciones tienen un inicio y un fin, el inicio puede ser la palabra reservada **from** o el 0% y el final puede ser la palabra reservada **to** o el 100% o se puede navegar a través de estos porcentajes para hacer una animación más detallada.

Puede animar todas las propiedades que desee las veces que sean. Las reglas *keyframes* deben tener un nombre identificativo el cual será el que sea asignado a la propiedad *animation*, como vemos en el siguiente ejemplo:

```
/* Animación con regla donde cambia el color de fondo. */
@keyframes example {
    0% {background-color: red;}
    50% {background-color: yellow;}
    70% {background-color: blue;}
    100% {background-color: green;}
}
```

El nombre de la regla es *example*, y cambia el color del fondo del elemento de rojo a amarillo, azul y verde, a través de los distintos porcentajes establecidos, esta animación entonces cambia gradualmente el color de fondo de un elemento.

13.2 Propiedades de las animaciones

- **animation-name:** asigna una regla *keyframes* a un elemento.

- **animation-duration:** establece la duración de la animación. Acepta valores en segundos. Si no se establece esta propiedad, la animación no comenzará, esto porque el valor por defecto es 0.

```
div {
    animation-name: example;
    animation-duration: 3s;
}
```

- **animation-timing-function:** cambia la velocidad en la que la transición se crea dentro de la duración establecida. Los valores que acepta son los mismos que la propiedad *transition-timing-function* en la sección **Transiciones** (**ease**, **linear**, **ease-in**, **ease-out**, **ease-in-out** y **cubic-bezier()**).
- **animation-delay:** agregar un retraso a la animación. Acepta valores en segundos o milisegundos.

```
div {
    animation-timing-function: ease-in;
    animation-delay: 1s;
}
```

- **animation-iteration-count:** determina el número de veces que una animación se repetirá. Acepta valores enteros y la palabra reservada **infinite**, para repetir indefinidamente una animación. Si utiliza el valor 0, la animación nunca comenzará.
- **animation-direction:** indica el comportamiento de la regla o cómo esta es aplicada. Los valores que acepta son:

- *normal*: la animación se aplica de 0% a 100%.
- *reverse*: la animación se aplica de 100% a 0%.
- *alternate*: la animación primero corre hacia adelante, luego hacia atrás, luego hacia adelante.
- *alternate-reverse*: la animación primero corre hacia atrás, luego hacia adelante, luego hacia atrás.

Dejamos un ejemplo para que lo pruebe y vea cómo se ve la animación:

```
estilos.css
/*
Animación con regla donde el ancho pasa de 0px a 100px.
*/
@keyframes colorange {
    from { width: 0px; }
    to { width: 100px; }
}
/*
Clase que define las propiedades de la animación con regla en contenedor.
*/
.animation {
    animation-name: colorange;
    animation-duration: 3s;
    animation-timing-function: ease-in;
```

```
animation-delay: 1s;
animation-iteration-count: infinite;
animation-direction: reverse;
height:100px;
width:0px;
}

prueba.html
<div class="animation"></div>
```

Puede utilizar el **acceso directo** de la propiedad *animation* de la siguiente forma:

animation: name duration timing-function delay iteration-count direction

14 Filtros

Los **Filtros CSS** permiten agregar una capa de efectos gráficos, como desenfoque o cambio de color a un elemento, suelen ser utilizados estos filtros para ajustar el renderizado de imágenes, fondos y bordes. El filtrado de imágenes es útil cuando se requiere tener distintos estilos para una sola imagen; en vez de subir la misma imagen varias veces pero con distinta edición, se puede subir solamente una vez y aplicarle varios filtros con estilos distintos.

Los filtros no están disponibles para Internet Explorer, Edge 12, Safari 5.1 y versiones anteriores del mismo.

Por ejemplo, el método *drop-shadow()* crea una sombra para un elemento, sus parámetros son:

```
filter: drop-shadow(width height blur color)
```

Vemos que esta propiedad tiene los mismos parámetros que *box-shadow*, a diferencia de que no incluye el parámetro *spread*. La *Figura 51* tiene el resultado de un ejemplo:

```
estilos.css
img {
    width: 130px;
    height: 100px;
}
/* Filtro de sombra con desenfoque. */
.dropshadow {
    filter: drop-shadow(5px 9px 2px blue);
}

prueba.html
Image<br>


Drop Shadow 5px 9px 2px blue


```

Figure 51: Método **drop-shadow()** para agregar una sombra a un elemento

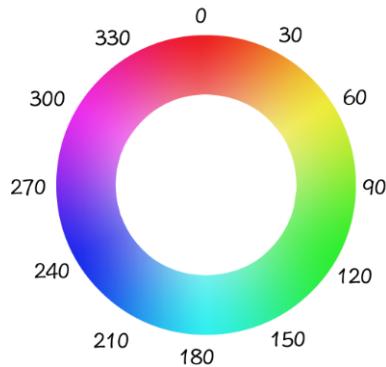


14.1 Funciones

Algunos de los métodos disponibles para los filtros son:

- **blur()**: aplica un efecto de desenfoque a un elemento. Acepta un valor entero, no porcentajes, donde este valor es el radio de píxeles que se mezclan entre sí (un valor mayor crea mayor desenfoque). Si no se escribe un valor en el parámetro, el valor por defecto es 0.
- **brightness()**: ajusta el brillo de un elemento, haciendo que se vea más brillosa u oscura. 0% es un elemento completamente oscuro, 100% es un elemento sin cambio alguno y valores mayores al 100% produce una imagen brillosa. Los valores enteros permitidos van de 0 a 1, cualquier valor negativo simplemente hará la imagen negra.
- **contrast()**: ajusta el contraste de un elemento. 0% representa un elemento completamente gris, 100% es el elemento sin cambios, entonces, el contraste se maneja entre valores mayores y menores al 100%. Los valores enteros permitidos van de 0 a 1, cualquier valor negativo no aplicará un cambio al elemento.
- **drop-shadow()**.
- **grayscale()**: agrega una escala de grises a un elemento. El único parámetro disponible es el porcentaje o valor entero de la escala a aplicar, donde 0% es el elemento original y 100% tiene una escala de grises completa. Cualquier valor negativo no hará efecto en el elemento. Los valores enteros permitidos van de 0 a 1.
- **hue-rotate()**: aplica una rotación de tonos (basada en el círculo de colores presente en la *Figura 52*) a un elemento. Este método recibe un ángulo como parámetro, 0 y 360 es el color rojo y lo puede ajustar como requiera.
- **invert()**: invierte los colores de un elemento de tal forma que se aplica brillo a zonas oscuras y se oscurecen zonas brillosas. La forma de trabajar este método es el mismo que *grayscale*, sin embargo, acepta valores mayores al 100%, pero estos no tendrán efecto en el elemento.
- **opacity()**: establece un efecto de transparencia a un elemento. 0% es el valor para un elemento completamente transparente y 100% es el elemento original. No acepta valores enteros.
- **saturate()**: controla la saturación de un elemento. El único parámetro disponible es el porcentaje o valor entero de la proporción de saturación a aplicar, donde 0% es **completamente insaturada** (escala de grises) y 100% es el elemento original. Los valores enteros permitidos van de 0 a 1.
- **sepia()**: convierte un elemento a sepia. La forma de trabajar este método es el mismo que *grayscale*, sin embargo, acepta valores mayores al 100%.

Figure 52: El círculo de colores



Mostramos ejemplos de los métodos anteriormente mencionados en las *Figuras 53* y *54* respectivamente:

```
estilos.css
/*
Clase para mostrar contenedores uno al lado del otro.
*/
.cont {
    display: inline-block;
}
/*
Clase para contener etiqueta e imagen.
*/
.layout {
    display: inherit;
}
/*
Filtro de escala de grises, rotación de tonos, inversión, saturación y sepia.
*/
.grayscale {
    filter: grayscale(100%);
}
.hue-rotate {
    filter: hue-rotate(180deg);
}
.invert {
    filter: invert(70%);
}
.saturate {
    filter: saturate(25%);
}
.sepia {
    filter: sepia(70%);
}

prueba.html
<div class="cont">
    <div class="layout">
        <p>Original</p>
        
    </div>
    <div class="layout">
        <p>Grayscale</p>
        
    </div>
    <div class="layout">
```

```

<p>Hue-rotate</p>

</div>
<div class="layout">
    <p>Invert</p>
    
</div>
<div class="layout">
    <p>Saturate</p>
    
</div>
<div class="layout">
    <p>Sepia</p>
    
</div>
</div>

```

Figure 53: Conjunto 1 de ejemplos de filtros CSS



```

estilos.css
/*
Clase para mostrar contenedores uno al lado del otro.
*/
.cont {
    display: inline-block;
}
/*
Clase para contener etiqueta e imagen.
*/
.layout {

```

```
        display: inherit;
    }
    /* Filtros de opacidad, brillo, contraste y desenfoque. */
    .opacity {
        filter: opacity(70%);
    }
    .brightness {
        filter: brightness(50%);
    }
    .contrast {
        filter: contrast(140%);
    }
    .blur {
        filter: blur(10px);
    }

prueba.html
<div class="cont">
    <div class="layout">
        <p>Original</p>
        
    </div>
    <div class="layout">
        <p>Opacity</p>
        
    </div>
    <div class="layout">
        <p>Brightness</p>
        
    </div>
    <div class="layout">
        <p>Contrast</p>
        
    </div>
    <div class="layout">
        <p>Blur</p>
        
    </div>
</div>
```

Figure 54: Conjunto 2 de ejemplos de filtros CSS



14.2 Usando múltiples filtros

Se pueden aplicar varios estilos a un solo elemento, simplemente se separan por espacios, como se ve en la *Figura 55*:

```
estilos.css
.filtered {
    /* Agrega filtros de saturación, sombra y desenfoque. */
    filter: saturate(30%) drop-shadow(5px 9px 2px gray) blur(1px);
}

prueba.html


```

Figure 55: Uso de múltiples filtros CSS

