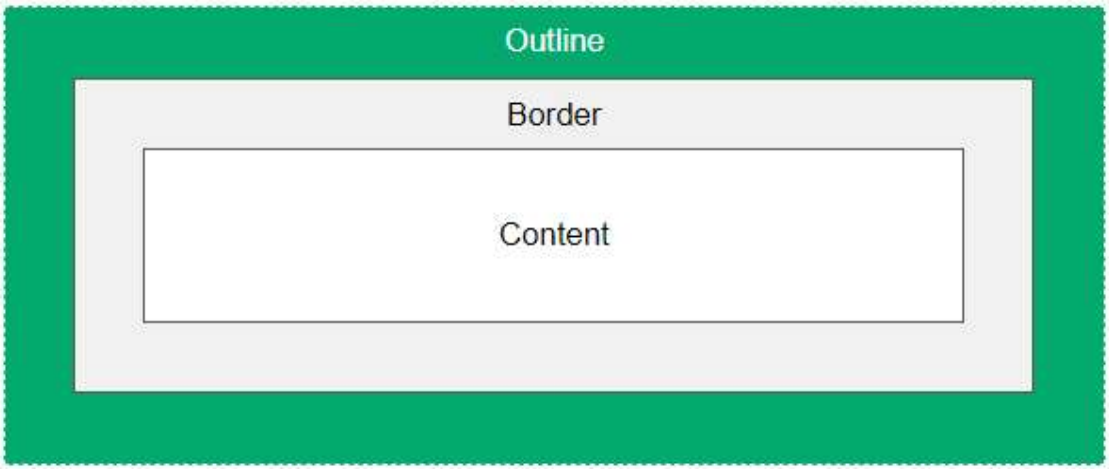
Outline

La propiedad **Outline** nos sirve para lo mismo que el border, con la diferencia que esta no toma espacio del elemento al que se lo aplicamos, sino que toma el espacio que se encuentra fuera del elemento



Podemos usar todos los valores que previamente vimos en el border, en los outline con el agregado de la propiedad outline-offset.

outline-offset: 15px;

El ouline offset nos agrega un espacio entre la outline y el borde de nuestro elemento, el espacio generado es transparente.

Para abreviar codigo, podemos hacerlo en este orden:

1outline-width
2outline-style (required)
3**outline-color**

Propiedades de las listas

Las propiedades de listas de CSS nos van a permitir:

Setear diferentes marcadores tanto a las listas ordenadas como a las no ordenadas, añadir fondos, cambiar los colores e inclusive usar imagenes como marcadores.

Diferentes tipos de marcadores (No todos tienen la descripcion, mirenlos por ustedes mismos!)

list-style-type:circle => es la predeterminada de las bullet list
list-style-type:square =>
list-style-type:armenian =>
list-style-type:disc =>
list-style-type:georgian =>
list-style-type:decimal => es la predeterminada de las listas ordenadas

list-style-type:decimal-leading-zero => igual a la predeterminada, pero agrega el cero delante del numero
list-style-type:lower-alpha => usa letras
list-style-type:upper-alpha =>
list-style-type:lower-latin =>
list-style-type:upper-latin =>
list-style-type:lower-roman =>
list-style-type:upper-roman =>
list-style-type:lower-greek =>
list-style-type:none => elimina los marcadores, especial para cuando querramos armar un menú

Imagen como marcador

list-style-image: url("ejemplo.gif")

Posicion de los marcadores

Tenemos una propiedad que nos permite cambiar la posicion de los marcadores de nuestras listas de dos maneras distintas, la predeterminada que es la siguiente:

list-style-position: outside;

Y también tenemos:

list-style-position: inside; => esta hace que el marcador se encuentre dentro del item listado, como parte de el, lo que hará que empuje el resto del contenido a su interior.

Eliminar marcadores

Podemos eliminar los marcadores de la siguiente manera:

list-style-type:none;

Podemos ahorrar un poco de codigo siguiendo el siguiente orden:

list-style-type => por si la imagen no nos carga, definimos un estilo de respaldo
list-style-position => define si el marcador esta dentro o fuera del item
list-style-image => define la imagen que vamos a usar como marcador

Quedaría entonces de la siguiente manera:

list-style: type position image;
list-style: circle outside url("ejemplo.gif");

Tablas

Para hacer tablas en HTML tenemos que definir tantos sus columnas como sus filas de la siguiente manera

```
<table> /*Define nuestro elemento tabla*/
    <tr> /*Define nuestra primera ROW o fila*/
```

```
        <th></th> /*Define nuestra primera columna cabecera*/
    </tr>
    <tr>
        <td></td>/*Define nuestras columnas*/
    </tr>
</table>
```

Como con cualquier otro elemento dentro de nuestro HTML, con CSS podemos darle una gran variedad de estilos, vamos a empezar dandole un borde.

```
table, th, td {
        border: 1px solid blue;
}
```

Como vimos, para darle un borde completo hay que aplicarle las propiedades a todas las etiquetas que se encuentran en nuestra tabla, lo que hace que tengamos bordes duplicados, en caso que no quisieramos esto, podemos solucionarlo con la propiedad collapse que tiene dos valores posibles, separate, que es el valor por defecto, y collapse, que hara que se junten los bordes y se vean como solo una linea

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

Podemos controlar tambien, el ancho que querramos que tome del contenedor donde se encuentre nuestra tabla con la propiedad width, aclarando el porcentaje que queremos que ocupe

```
table {
        width: 100%; => ocupara el ancho total del contenedor
}
```

El alto de cada row podemos controlarlo con la propiedad height, asi como también podemos controlar el comportamiento del texto con las propiedades de texto que vimos antes, podemos definir sus bordes independientes, como por ejemplo, solo pintar el borde inferior de las filas, podemos usas pseudo elementos como el hover, para darle un comportamiento en particular cuando nuestro puntero pase sobre ellos, etc, etc

Si nuestra tabla es muy grande para el lugar donde la estamos mostrando, podemos hacer que tenga una barra scroll horizontal, definiendo esto en un elemento contenedor de nuestra tabla, por ejemplo un ** ** de la siguiente manera

```
div {
        overflow-x: auto;
}
```

Propiedad Display

La propiedad display, especifica como y si algo es mostrado en pantalla. Todos los elementos HTML tienen un valor predeterminado de display, dependiendo de que tipo de elemento sea. Los valores por defecto suelen ser block o inline, aunque tambien tenemos el valor "none" que hace desaparecer nuestro elemento de la pagina, como si no existiese

Block

Mostrara nuestro elemento ocupando el %100 de nuestro ancho de contenedor, como un nuevo bloque en nuestra pagina.

Inline

Mostrara nuestro elemento en linea con los demas, ocupando solo el espacio(ancho) que realmente utiliza.

Inline-block

Esta propiedad difiere de la inline comun, nos deja cambiar los margenes/paddings y responde a las variaciones que le demos con las propiedades top, right, bottom, top

Tambien difiere de la block, ya que esta no hace quiebres de linea, o sea, nuestros elementos se mostraran en linea, uno seguido del otro

None

Hace desaparecer a nuestro elemento, no se muestra en la pagina, y esta actua como si el elemento en cuestion no existiese.

Tenemos también la propiedad visibility con su valor hidden, que nos permite ocultar algo en pantalla pero dejando que el contenedor exista, seguira ocupando su lugar en la pantalla, pero sin mostrar contenido alguno

visibility:hidden;

Propiedad ancho maximo

Cuando creamos un elemento de tipo block, este utilizara el ancho maximo disponible, lo que provocará, en caso que hagamos zoom, que nuestro elemento quede fuera de los limites de visualización. Si nosotros usamos un ancho determinado con margenes automaticos, cuando el navegador sea mas pequeño que el elemento, se generara automaticamente una scrollbar horizontal, y no vamos a poder ver el contenido entero, para avitar esto, tendremos que utilizar la propiedad ancho maximo.

La propiedad max-width nos permite darle un ancho máximo a nuestro contenedor, lo que nos asegurará que nuestro contenido no se salga del area visual por mas que hagamos mucho zoom o el tamaño de nuestro navegador sea mas pequeño que nuestro contenedor.

max-width: 500px;

Posicionamiento

La propiedad position especifica el tipo de posicionamiento usado para el elemento en cuestion, tenemos cinco diferentes valores de posicion:

- **static**
- **relative**
- **fixed**
- **absolute**

•sticky

Static

Esta es la propiedad predeterminada para todos los elementos HTML, los elementos static no pueden ser afectados por las propiedades top, bottom, left y right.

Un elemento que tenga definido position: static; no esta posicionado de ninguna manera en particular, siempre es posicionado de acuerdo con el flujo normal de la pagina.

Relative

Un elemento con position:relative, va a estar posicionado relativamente a su posicion normal, si le seteamos top, right y bottom esto hará que efectivamente se mueva de su posicion normal. El contenido de la pagina no se ajustara para llenar el espacio dejado por el elemento al moverlo.

Fixed

Un elemento con position:fixed, esta posicionado relativamente al viewport, esto significa que permanecerá en el mismo lugar, aunque la pagina sea scrolleada, usaremos las propiedades left, right, top, bottom para posicionarlo.

Absolute

Un elemento con position: absolute; es posicionado relativamente a su elemento padre, pero para que esto funcione, el elemento padre debe de tener definida una posicon que sea distinta de Static, sino, tomará su posicion respecto al body, y no al elemento padre

Sticky

Un elemento con position: sticky; es posicionado en base a la posicion en la que se escuentre el usuario, estos elementos se encuentran al comienzo como si fuesen un relative, hasta que el scroll los deje fuera del viewport, en ese momento pasar a ser fixed

Propiedad z-index

Cuando los elementos se posicionan, a veces, pueden tapar a otros elementos, la propiedad z-index le dira a nuestro navegador un orden de prioridad, para poder controlar en que orden esto sucede. Puede recibir valores tanto positivos como negativos, los numeros mas bajos, tienden a estar mas al fondo y los mas altos mas al frente.

Recordemos que z-index solo funciona en elementos posicionados, se define de la siguiente manera:

```
contenedor {
    z-index : valor;
}
```

Para probar esta propiedad, les recomiendo poner varios elementos superpuestos, ya sean divs con algun color de fondo o imagenes, para que se vea mejor el resultado, y vayan intercambiando los valores de z-index hasta que entiendan el como funciona la propiedad!

Overflow

La propiedad overflow controla como vemos el contenido cuando es demasiado grande para mostrarse en su contenedor, puede tener los siguientes valores:

overflow: visible; => Es la predeterminada, el contenido se renderiza aunque no entre en su contenedor.

overflow: hidden; => Todo el contenido que no entre en su contenedor queda oculto, sigue existiendo dentro del contenedor, pero no llega a verse a simple vista

overflow: scroll; => al definir el valor scroll, se añaden barras de desplazamiento horizontales y verticales, para poder visualizar todo el contenido.

Tenemos dos variaciones que nos permiten un mayor control que con el valor scroll:
overflow-x // overflow-y, en la que vamos a poder ocultar una y definir una para solo tener una de las variantes mostrada en pantalla

overflow: auto; => similar al scroll, pero solo añade barras de desplazamiento cuando es necesario

Overflow-wrap

El overflow-wrap nos sirve para cuando un texto tiene palabras muy largas o por alguna razon se salen de nuestro contenedor, para evitar esto, definimos el overflow de la siguiente manera:

overflow-wrap: break-word;

Float

La propiedad float especifica como nuestros elementos deberian flotar, nos sirve para especificar por ejemplo como queremos que una imagen "flote" con respecto a un texto en un contenedor. Puede tener uno de los siguientes valores:

float: left; => el elemento flota a la izquierda del contenedor
float: right => el elemento flota a la derecha del contenedor
float: none => el elemento no flota, va a ser mostrado en linea con el texto
float: inherit => hereda la propiedad de su elemento padre

Clear

Cuando usamos la propiedad float, y queremos que el siguiente elemento se posicione debajo de nuestro elemento flotante, tenemos que usar la propiedad clear, esta especifica como se comportan los elementos que estan despues de un elemento flotante. Puede tener uno de los siguientes valores:

clear: none; => el elemento no es empujado debajo de los elementos flotantes, esta propiedad es la predeterminada

clear: left; => el elemento es empujado abajo a la izquierda del elemento flotante

clear: right; => el elemento es empujado abajo a la derecha del elemento flotante
clear: both; => el elemento es empujado en ambas direcciones del elemento flotante
clear: inherit; => hereda el valor de su elemento padre

Cuando usamos clear despues de un float, tenemos que hacer coincidir el valor de clear con el del float, ya que si por ejemplo usamos un float left, y un clear right, los elementos se comportaran como si el clear no hubiese sido definido.