**CSS**

**CSS necesita de HTML para existir, el nombre es ‘Cascading Stlye Sheets’.**

El mister explica que generalmente se carga los archivos CSS al principio del código, pero algunos lo hacen al final, pero por ejemplo si un usuario tiene una patata PC y el dev lo pone al final. Se le cargaría el HTML y después el CSS ( si es que puede la PC no)

Lo primero que nosotros tenemos que hacer, es simpre cargar los contenidos de CSS al principio.

Para esto vamos a utilizar <link> que tiene que tener una asignación de ‘href’ que es donde se va a encontrar nuestro estilo y como vimos antes ‘href’ es una redireccion, por ende tenemos que crear ‘main.css’

 <link href="main.css" type="text/css" />

El tipo de archivo que estamos cargado es de tipo texto CSS.

Bueno ya en nuestro archivo main.css lo primero que vamos a ver son los selectores, estos mismos pueden seleccionar nuestras etiquetas de HTML.

Por suerte si vos queres editar de tu archivo HTML con los selectores estos utilizan los mismos términos.

Por ejemplo si yo quiero seleccionar body de mi archivo HTML, Simplemente me refiero como body.

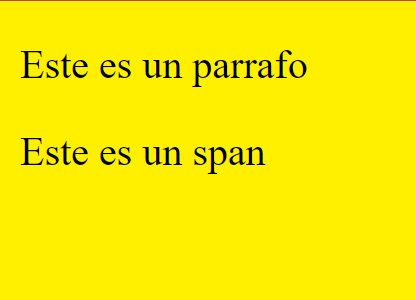
body {

    background-color: #fff000;

}

Ahora aparecen esos corchetes {} como los colores son hexadecimal y se combinan entre letras y números pégale un Google. PERO nos falto algo en nuestro HTML que se olvidó el míster.

<link rel="stylesheet" href="main.css" type="text/css" />



Ahora cambiamos color del texto

body {

    background-color: #eeeeee;

    color: brown;

}

Se usa color, pero hoy en dia el mister no lo dice pero no hace falta usar CREO los códigos como el menciona.

Ahora vamos a ver 2 tipos de selectores. Bueno vamos a ver como modificamos partes especificas, usando etiquetas ya conocidas anteriormente, <id> <class>

<p id="parrafo"> Este es un parrafo</p>

    <span class="etiqueta"> Este es un span</span>

    <br />

    <span class="etiqueta"> Este es otro un span</span>

    <br />

    <span class="etiqueta"> Este es un tercer span</span>

Forma para darle diseño a los ID

body {

    background-color: #efefef;

}

#parrafo {

    color: #ff0000;

}

PRESTEMO ATENCION AL HASTHTAG # como función.

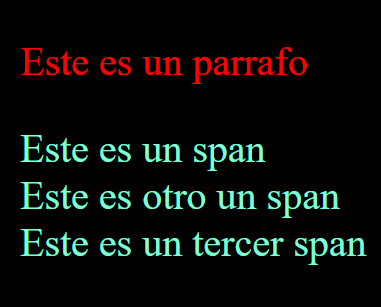
}

.etiqueta {

    color: aquamarine;

}

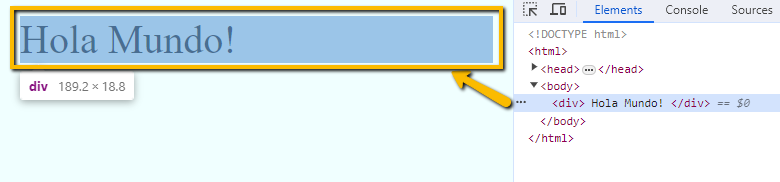
Ahora atentos por que cuando queremos modificar un tipo <class> Se usa (PUNTO .) y no Hashtag!



En HTML todas las etiquetas que nosotros escribimos o redactamos, son consiedardos ‘Cajas’

Asi es como las que conocemos, cuando nosotros hablamos del modelo de la caja es de como nosotros vamos a ‘modelar las cajas’ en HTML. Esto es muy útil por que nos permite construir pequenios elementos en HTML que después nosotros vamos a ir re-utilizando dependiendo de los estilos que agregemos.

Ahora vamos a ver para explicar le modelo de la ‘Caja’.



Vemos que con <div> en el inspector del browser nos muestra ya la ‘Caja’

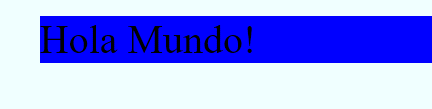
<div class="caja"> Hola Mundo! </div>

.caja {

    margin: 8px;

}

Esto genera un margen dando como especificador los pixeles.



Vemos que esta el margen y yo le agrege color, pero que paso? El margen no se vio afectado.

Ahora vamos a ver ‘Padding’

.caja {

    margin: 8px;

    padding: 8px;

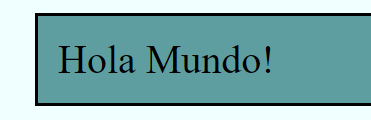
    background-color: blue;

}



Ahora vemos que el contenido de la caja aumento, Bueno vamos a ver una 3era propiedad que va a diseniar el borde de la caja, este se va a encontrar entre el medio del margen y del padding,

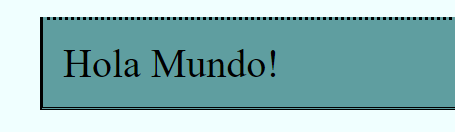
    border-style: solid;



Claramente border puede tener muchos estilos, no solo el ‘solid’ como dashed que hace líneas punteadas, o dotted que tiene puntitos o double que son 2 lineas dobles, y mucho mas!

Ahora podes inclusive usar varios en una misma, ejemplo:

border-style: dotted dashed double solid;



Bueno también existe border-width y no voy a explicar que hace por lógica.

Tambien podemos asignarle un color como border-color Voy a poner TODO por que es LOGICA pura

.caja {

    margin: 8px;

    padding: 8px;

    border-style: solid;

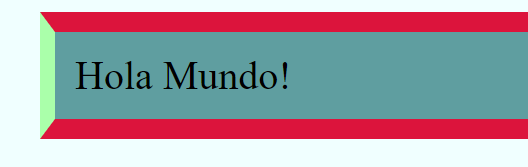
    border-width: 8px;

    border-color: crimson;

    border-left: 6px solid #afa ;

    background-color: cadetblue;

}



Entre tantas funciones por ejemplo border-radius hace que el borde sea redondeado. (Mientras mas pixeles mas redondeado)

Bueno podemos indicar la altura y ancho de la ‘caja’ simplemente se usa ‘height’ y ‘width’ y ‘Color:” para el color del texto etc.

con las propiedades height(alto) y width(ancho), no es lo mismo el padding te separa el contenido del borde del contenido,y el ancho y el alto modifican las dimensiones de la caja

Nota personal, luego del ejercicio 1 de CSS hay que estar atentos a la funcionalidad de padding height-width y hasta border-top o border-left a la hora de ser creativos para diseniar.

Vamos a ver como alinear texto

<p> Hola soy un parrafo! </p>

p {

    color: coral;

    text-align: center;

}

Pretty straight forward el command, refleja el alineado del texto.

Bueno ahora vamos a ver decoration o decoraciones.

<a href="https://www.leanmolina.com"> Coaching </a>

a {

    text-decoration: none;

}

Ahora por ejemplo cuando usamos un hipervínculo, este mismo queda displayeado con subrayado Ahora bien, con text-decoration: none; le sacamos ese mismo subrayado.



Text-decoration: overline; le pone esa misma línea arriba, el comando line-through lo tacha digamos en ese estilo.

Ahora underline hace que queda abajo como postrado pero bien cheto!

p {

    color: coral;

    text-align: center;

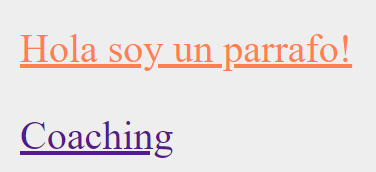
    text-align: underline;

}

a {

    text-decoration: underline;

}



    text-transform: uppercase;

Tal y como explica transforma el área seleccionada en este caso uppercase. TODO MAYUS y varias funciones mas como hacer todo minúscula o capitalize que hace que sea prolijo.

    text-indent: 15px;

Le agrega espacio al text o sangria por ejemplo mientras mas px mas se mueve el texto a la derecha.

   letter-spacing: 3px;

Agrega estilo ósea hace un espaciado entre las letras por ejemplo G O O G L E.

    line-height: 2;

Bueno line-height vendría a hacer el espacio en el texto cuando es largo o nose como explciarlo por ejemplo.



El espacio entre si mientras mas height mas distancia.

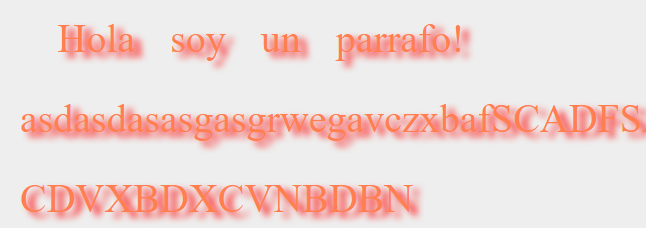
    word-spacing: 10px;

Parecido a letter-spacing por no decir lo mismo, busque en Google la diferencia:

The word-spacing property is similar to letter-spacing , though naturally its use governs the amount of space between the words in a piece of text, not the individual characters

    text-shadow: 3px 2px 3px red;

Esto agrega un sombreado al texto el primer PX es que tanto se desplaza el texto al a derecha y el segundo PX es que tanto se desplaza hacia abajo y la tercera es que tan difuso o que tan sombreado no se como explciarlo se va a ver.



El mister dice que la clave esta aca para hacer un sombreado cheto

text-shadow: 3px 2px 3px rgba(0, 0, 0, 0.3);

el RGBA pone los parámetros de los colores y el 0,3 la intensidad.

VAMOS A VER TIPOS DE FUENTES!

   font-family: sans-serif;

Asi cambiamos de familia de fuente digamos.

    font-style: italic;

Como lo ponemos en cursiva, oblique etc.

    font-size: 20px;

Tamanio de fuente, (Aclaracion mucha gente usa ‘em’ como medición ya que es el general)

   font-weight: bold;

El peso de la fuente o como en este caso aplicamos negrita. AHORA ATENCION EL TAMANIO VAIRA DE 100 A 900. 900 SIENDO EL MAXIMO BOLD DIGAMOS.

Bueno vamos a ver Links y CSS.

a {

    color: hotpink;

}

Cambio de color obviamente.

a:visited{

    color:blue

}

a:active {

    color: black

}

a:hover{

    color:gray

}

Visited una vez que ya abriste el link, active cuando haces click y hover cuando posas el cursor sobre el link, le podes agregar text-decoration , text-style etc o background-color (que resalta el link) a ser CREATIVOS al ahora de diseniar eh!

Vamos a ver Listas y CSS y como darle estilo y demás.

<body>

    <ul>

        <li> Elemento 1</li>

        <li> Elemento 2 </li>

        <li> Elemento 3 </li>

    </ul>

</body>

Tipica lista no organizada o <ul> unorganized list que hemos viste anteriormente.

Genera lo siguiente:

* Elemento 1
* Elemento 2
* Elemento 3

Bueno, ahora vamos a ver como darle estilo a esos ‘puntitos’.

ul li {

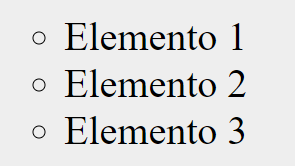
}

Esa es la sintaxis, primero el padre digamos ‘UL’ y después el hijo ‘li’

ul li {

    list-style-type: circle;

}

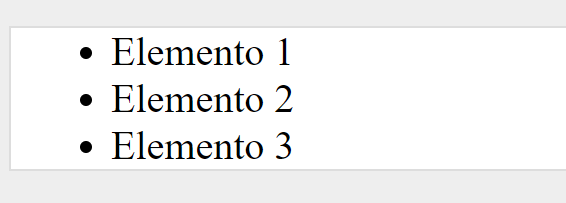


ul {

    background-color: #fff;

    border: solid 1px #ddd;

}



    list-style-type: none;

Saca los puntitos por ejemplo.

ul {

    background-color: #fff;

    border: solid 1px #ddd;

    border-radius: 4px;

    padding: 15px;

    list-style-type: none;

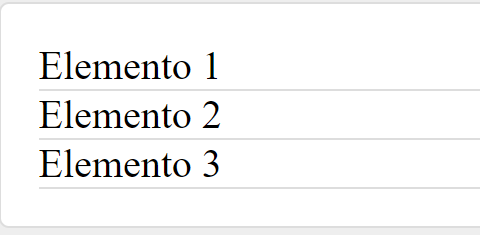
}

li {

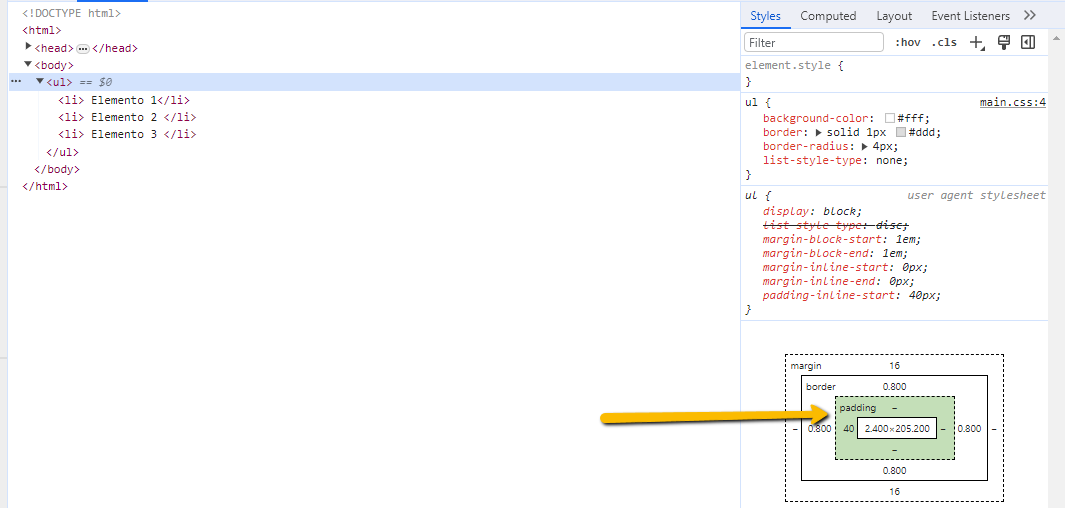
    border-bottom: solid 1px #ddd;

}

Atentos al border-bottom para darle una línea por ejemplo. Y en ‘LI’



Importante, si vemos la imagen anterior vemos que la sangria en los elementos esta, y ahora el Mister nos ensenia como sacarla, Luego de inspeccionar en Chrome o tu browser vos podes sacar el padding del margen que vos queres identificar (en este caso el izquierdo) y en el mio es de 40!



    padding-left: 0px;

Le seteamos el valor y ahora si esta mas hacia la izquierda! Good job.

Genial, ahora empecemos con TABLAS y CSS.

    <table>

        <tr>

            <th> Empresa </th>

            <th> Contacto </th>

            <th> Pais </th>

        </tr>

        <tr>

            <td> Ganshetera SRL </td>

            <td> Brian Reba </td>

            <td> Jamaica </td>

        </tr>

        <tr>

            <td> Cogollera S.A </td>

            <td> Lucas Rampoldi </td>

            <td> Burkina Faso </td>

        </tr>

    </table>

Nuestra Tabla sin CSS.

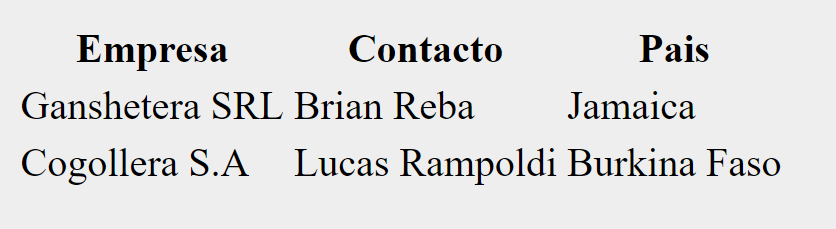
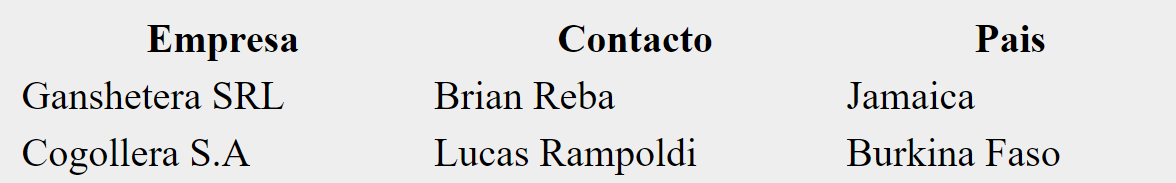


table {

    width: 100%;

}

Esto setea que la tabla use el 100% del ancho de la pagina ahora veamos como queda no?



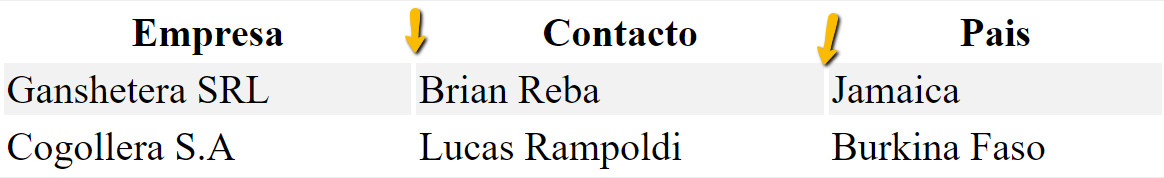
Mas ordenado y lindo no? PUES CLARO QUE SI!

tr:nth-child(even) {

background-color: #f2f2f2;

}

Bueno TR de table row pero pensando un poco decimos, es para todas las row? O queremos en particular con ‘nth-child (even) seleccionamos las row de la table



Bien quedo barbaro pero como las flechas indican, vemos que existe una apertura blanca todavía, como solucionamos esto? Con lo siguiente

    border-collapse: collapse;

Genial, y lógico collapso como mi salud mental desde que agarre este laburo.

table th {

    background-color: #4caf50;

}



Bueno voy a compartir como quedo la tabla en si que nos ensenio el mister con el CSS primero y después la foto

body {

    background-color: #eee

}

table {

    width: 100%;

    background-color: white;

    border-collapse: collapse;

}

tr:nth-child(even) {

background-color: #f2f2f2;

}

td {

    border: 1px solid #ddd;

    padding: 8px;

}

table th {

    background-color: #4caf50;

    color: white;

    text-align: left;

    padding-top: 12px;

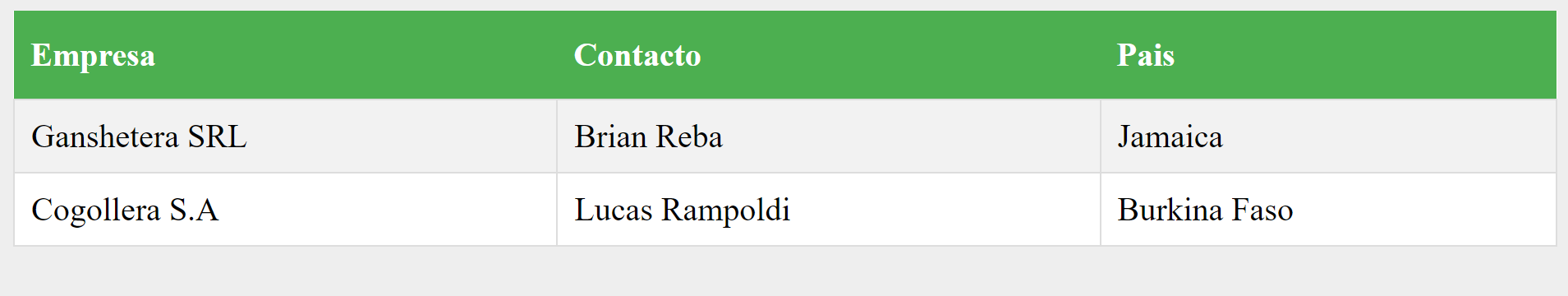
    padding-bottom: 12px;

    padding-left: 8px;

    padding-right: 8px;

}

Aclaracion personal, los border en si con el colorcito gris sumado a los padding lo hacen ordenado y prolijo, esto da la sensación de una tabla pulcra.



Ahora veremos la propiedad de display, la mimsa indica como vamos a visualizar el elemento (HTML) en este caso en pantalla, hasta ahora podemos ver la propiedad de ‘Block’ o de ‘In-line’.

Block nos va a permitir ver las propiedades como bloques e In-line que en el caso de esta será colocar los elementos uno al lado del otro.

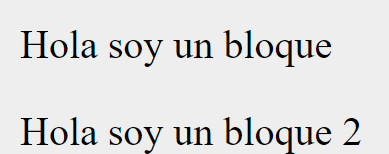
Pero… por que se refiere como ‘bloques’? Bueno por que es como si fueran bloques encima de otro como se displayea por defecto.

<body>

    <p> Hola soy un bloque</p>

    <p> Hola soy un bloque 2 </p>

</body>

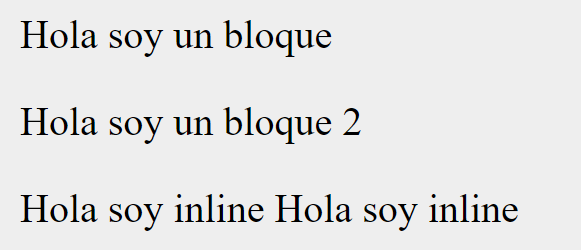


Vemos como se pone el texto por bloque encima de otro bloque.

Ahora para explicar in-line podemos hacerlo usando <span> por ejemplo.

    <span> Hola soy inline </span>

    <span> Hola soy inline</span>



Bueno esto sucede por que de por si por defecto span viene con display en forma de inline y <p> en forma de bloque, pero como cambiamos esto? Bueno, veamos a continuación.

p {

    display: inline;

}



<body>

    <p> Hola soy un bloque</p>

    <h1> Son bloques </h1>

    <div> tambien un bloque </div>

    <form> lala</form>

    <header> tambien un bloque</header>

    <footer> otro bloque</footer>

    <section> otro bloque mas</section>

</body>

Funciones que son displayeados en forma de bloques. Y In-line son las siguientes

   <span> soy un inline</span>

    <a> link tambien inline</a>

    <img />

    <div id="invisible"> soy invisible</div>

#invisible {

    display:none;

}

Hace invisible el texto.

span {

    display: block;

}

Como cambiamos el display.

A continuación, veremos la propiedad ‘max width’.

La función indica el ancho máximo que la propiedad puede utilizar pero no define como el ancho que va a tener

p {

    max-width: 350px;

    background-color: #f00;

    margin: auto;

    position: static;

}

Ahora vamos a ver la función ‘position’ por defecto viene la operación ‘static’ donde nosotros los hayamos definido. Ahora la función position: relative; hace referencia que si queremos mover un elemento un poquito a la izquierda pero todo el elemento, es que le cambiamos la posición a relativa. Bueno la realidad es que no entendí un carajo, pero calculo que hace relativo el orden cuando diseniamos.

Y por ultimo position: fixed; hace que el contenido este fijado dentro de los parámetros mas allá de cuanto contenido tenga. Por ejemplo: un menú en una pagina web que esta fijo y te acompania mas allá que scrollees o cambies de pagina. (hay que probar para entender)

Ahora veremos la propiedad de ‘overflow’

<body>

    <div id="over">

        lala lele lili lolo maria. este es un contenido mas largo de lo normal. que espero poder esconder.

    </div>

</body>

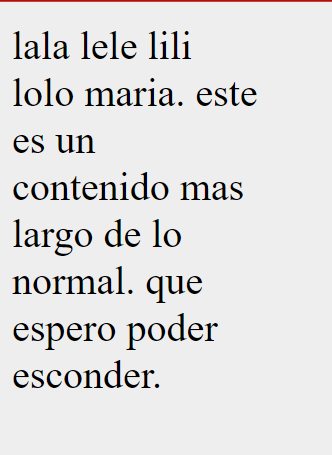
#over {

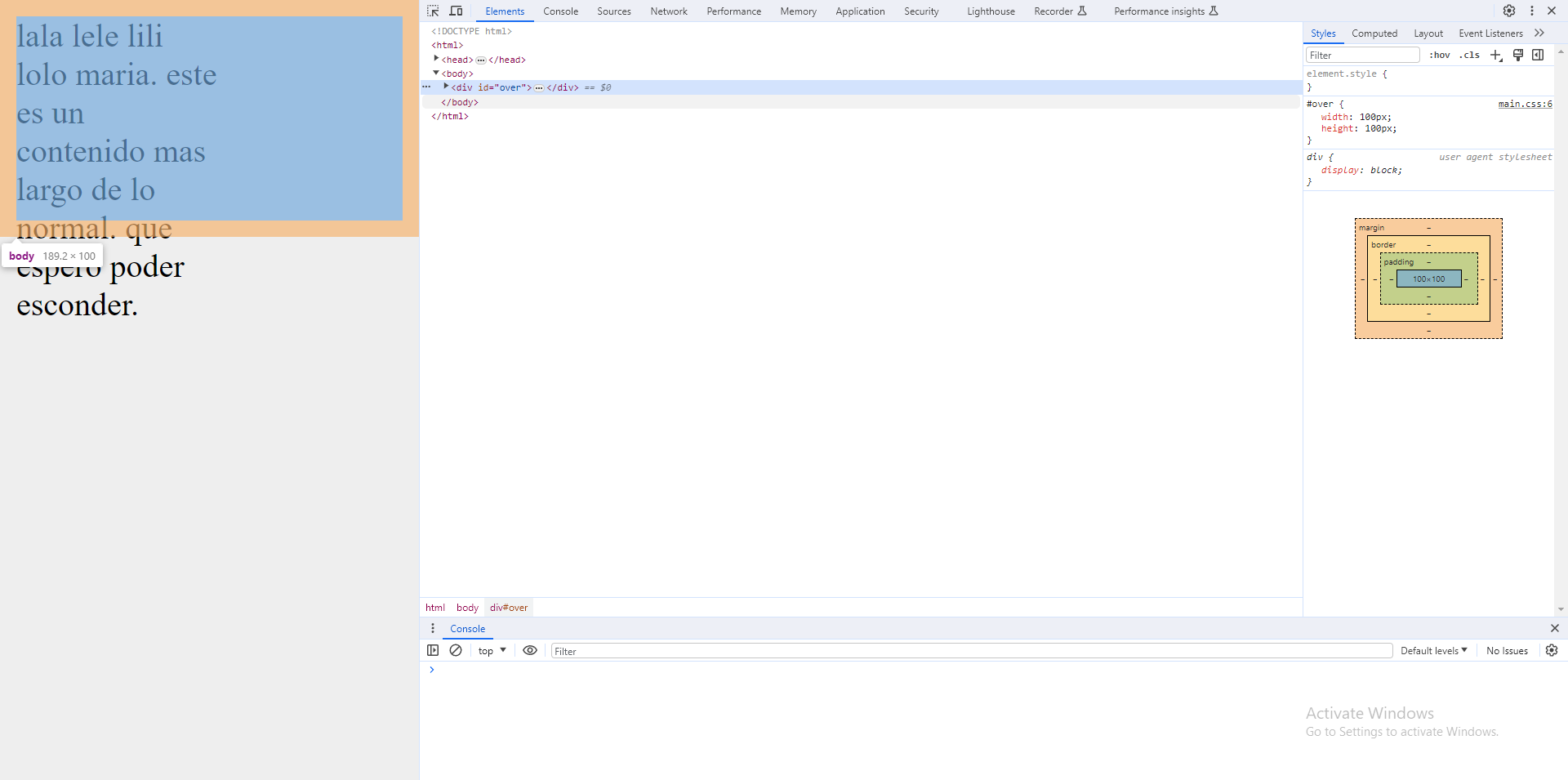
    width: 100px;

    height: 100px;

}

Reposicionamos el contenido para que se muestre como si estuviera en una caja o cuadrado.





Pero eso no es lo importante, lo importante es ver como dado el height y width el contenido en si de la caja es desproporcional al texto, por ende overflow perfila el uso adecuado.

#over {

    width: 100px;

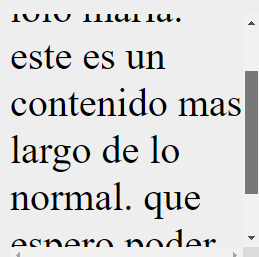
    height: 100px;

    overflow: visible;

}

Overflow: hidden; esconde una área del texto que no permite ni isquiera seleccionar.

Overflow: scroll;



Scroll en acción, excelente lpm.

Y por ultimo Overflow: auto; que solo agrega la barrita para scrollear solo en caso de necesitarla.

Overflow-y (cambias por x – y- z como las dimensiones)

Genial, ahora veremos la propiedad de ‘Float’ con la misma podremos especificar si ‘x’ elemento va a ‘flotar’ ya sea a la izquierda o a la derecha. Pero para decir que algo ‘flota’ es ser bastante redundante.

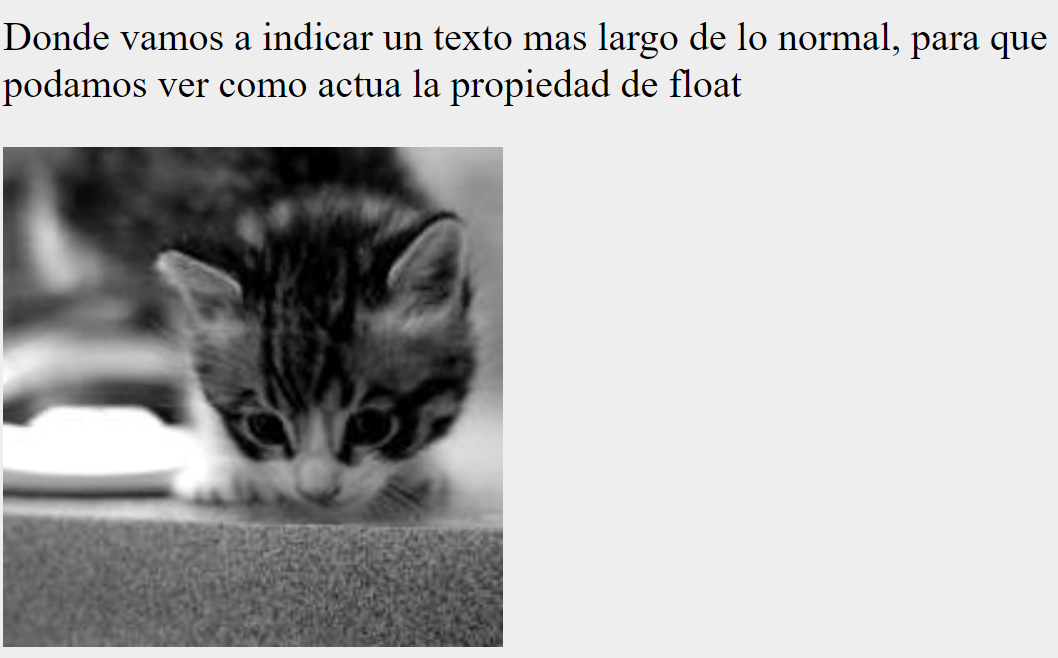
<body>

    <p> Donde vamos a indicar un texto mas largo de lo normal, para que podamos

        ver como actua la propiedad de float </p>

        <img src="https://placekitten.com/g/200/200"/>

</body>

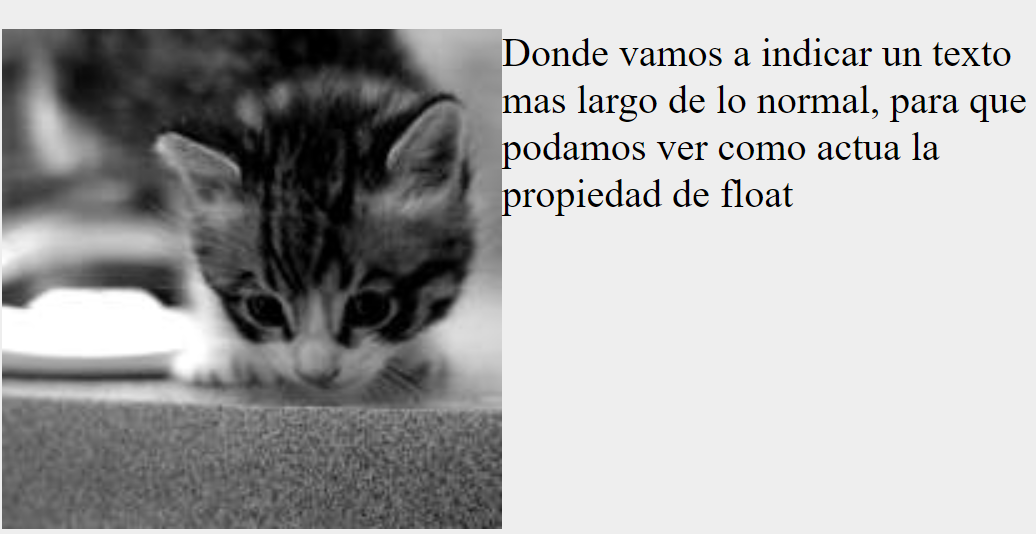


img {

    float: left;

}

Ahora no voy a subir la misma imagen pero no cambio nada visualemente, pero en realidad si.. Ahora todo el texto que ingresemos va a ser del lado derecho de la imagen!



Ahora corri la imagen arriba en el código, quedo primero que el texto y dicho hecho!

Genial, ahora veremos selectores!

<body>

    <p> lala</p>

    <div>

    <img src="https://placekitten.com/g/200/200"/>

    <p> Donde vamos a indicar un texto mas largo de lo normal, para que podamos

        ver como actua la propiedad de float </p>

    </div>

</body>

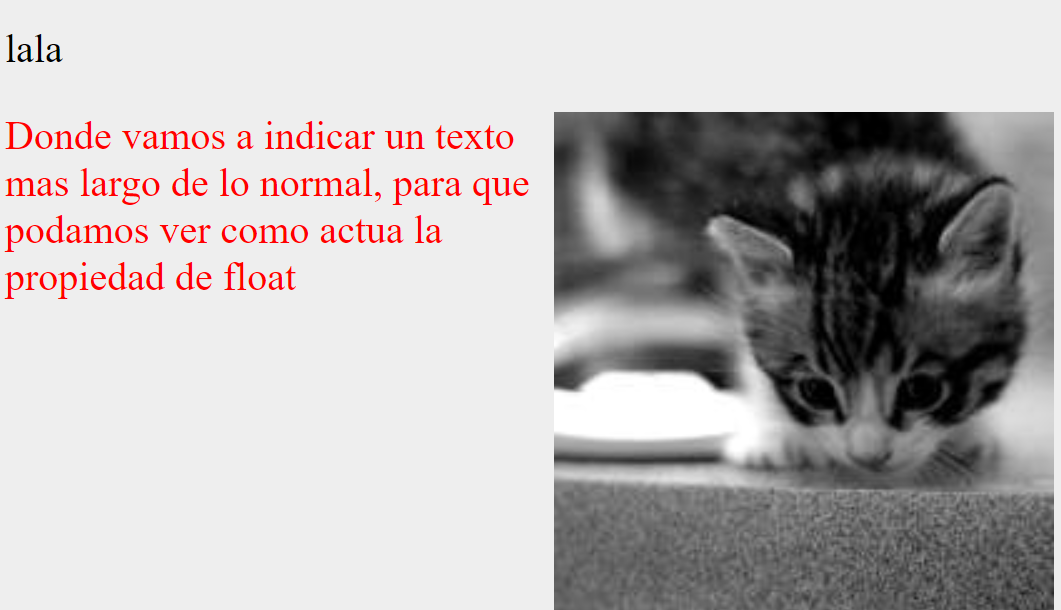
Bueno ahora, que pasa si queremos seleccionar las <p> que están afuera de div sin tocar los que están adentro para diseniar? Bueno para eso vamos a a necesitar selectores.

div p {

    color: #f00;

}

Bueno como dice bien el código, estamos seleccionando estrictamente los <p> dentro de de los div entonces como queda esto?



Bueno ahora veremos la opacidad y como la implementamos.

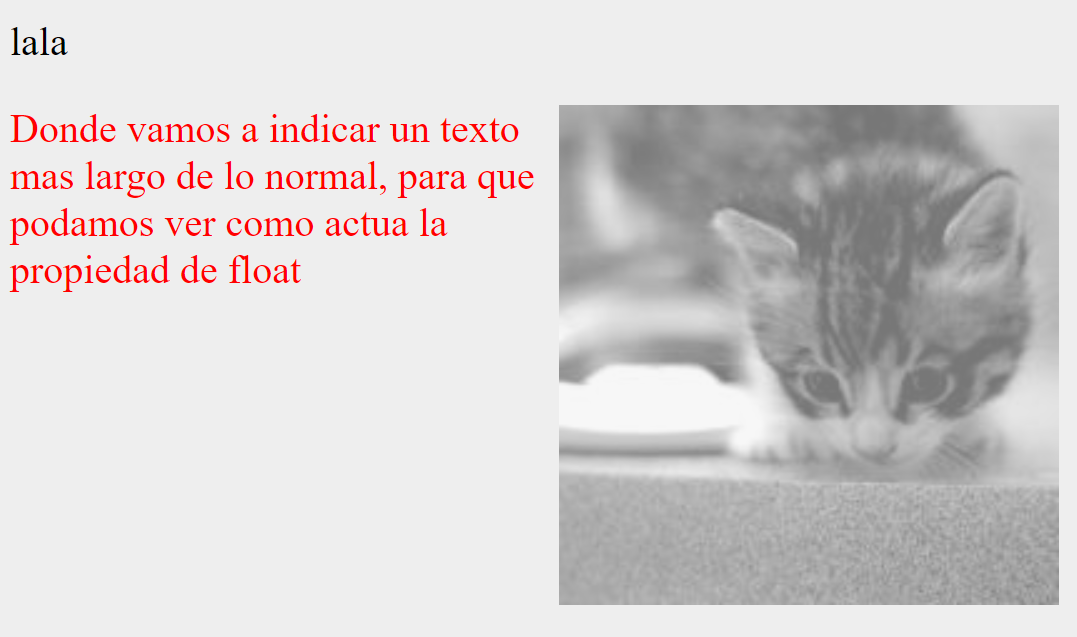
img {

    float: right;

    opacity: 0-1;

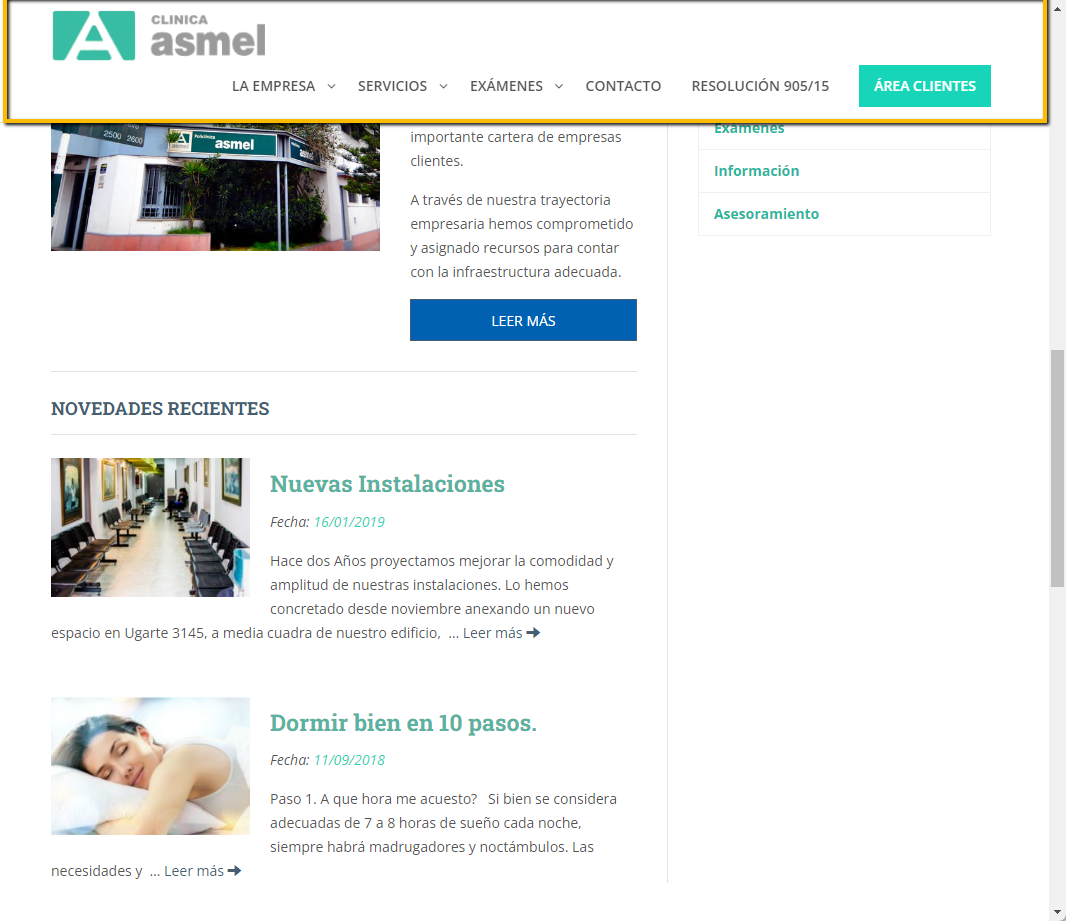
}

Opacidad recibe un valor entre los mencionados 0-1 (siendo 0 el mas transparente y siendo 1 completamente solido)



Eso es una opacidad del 50% en 0.5.

Ahora vamos a ver como podemos construir una barra de navegación superior.



Si nosotros lo pensamos de la siguiente manera, puede ser una lista de elementos pero que estos tienen un display ‘inline’ en vez de block, de manera que los elementos que vayamos cargando queden uno al lado del otro.

<body>

    <ul>

        <li><a> Inicio </a></li>

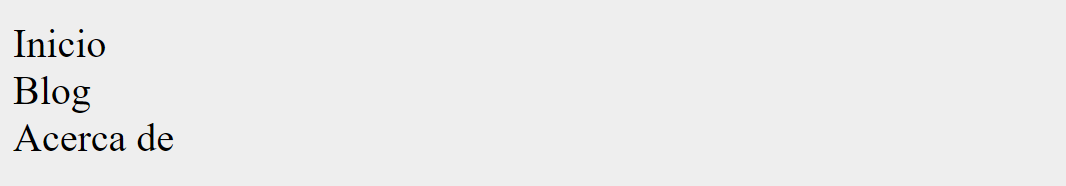
        <li><a> Blog </a></li>

        <li><a> Acerca de </a></li>

    </ul>

</body>

Creamos la lista primero.(Aclaracion en CSS le ponemos el margen y padding a 0 para que quede prolijamente)



Tambien le vamos a dar un ‘overflow:hidden’ ya que si exceden elementos en esta lista, no lo vamos a querer mostrar! Vamos a necesitar que tengan una posición fija, también vamos a especificar que la barra de inicio que da pegada en la parte superior de la pagina con ‘top:0’ Ya que va a ser inline tenemos que especificar que use todo el ancho de la pagina con width:100, y para darle el estilo inline’ le agregamos un float:left o el display ‘inline’



ul {

    list-style-type: none;

    margin: 0;

    padding: 0;

    overflow: hidden;

    position: fixed;

    top: 0;

    width: 100;

}

li {

    float: left;

}

Ahora vamos a ver los ‘hipervinculos’ de la lista en si y como lo hacemos

li a {

    display: block;

    color: white;

    text-align: center;

    padding: 14px;

    text-decoration: none;

}

El display block no lo llegue a razonar bien el porque, lo otro es obvio y el padding es para que tengan una separación y queden prolijo entre si. (le pusimos color a la lista para que se vea bien)



Ahora si observamos bien al lado de inicio quedo esa parte grisita que no queda bien, vamos a corregirlo con los márgenes, simplemente con un left:0 en CSS ul{ corrigio el mismo.

li:hover {

    background-color: rgba(0,0,0,0.3);

}

Esto le da un sombreado oscuro cuando posamos el cursor com tienen muchas pags.

Bueno ahora vamos a darle un color predeterminado cosa que ya esta puesto el hover, a que voy? A esto, nombro primero una clase en CSS

.active {

    background-color: #4caf50;

}

La asignamos obviamente

     <li class="active"><a> Inicio </a></li>



Cuando yo poso el cursor se va el color por ejemplo (por el hover) Bueno por ultimo nos queda ver como vamos a colocar el contenido, por que si empezamos a escribir esto lo haría sobre lo que ya esta puesto (en este caso la barra de navegación) y para esto lo vamos a desplazar hacia abajo dependiendo la cantidad de pixeles por sobre el alto y para esto insepccionamos con Chrome.

   <div class="content">

        <p>lalala</p>

    </div>

.content {

    margin-top: 47px;

}



Excelente!

Ahora vamos a ver como hacer un menú tipo ‘dropdown’ es cuando nosotros colocamos el mouse por encima y el mismo despliega una lista.

<body>

    <div class="dropdown">

        <button class="dropdown-btn">dropdown</button>

    </div>

    <div class="dropdown-content">

        <a href="#"></a>

    </div>

</body>

Nota personal veamos como subdivide el menú dropdown desde la creación de la misma al contenido en si que va a desplegar.

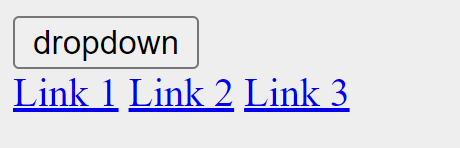
    <div class="dropdown-content">

        <a href="#"> Link 1</a>

        <a href="#"> Link 2</a>

        <a href="#"> Link 3</a>

    </div>



Ahora estamos editando el botón de dropdown, algo que recalca el mister es que no es la única alternativa poner un <a> para que aparezca la manito como un link. Nos otorga la siguiente función de cursor:pointer, para lograr el mismo resultado.

.dropdown-btn {

    background-color: #4caf50;

    color: white;

    padding: 16px;

   font-size: 16px;

   border: none;

   cursor: pointer;

}

Ahora lo que vamos a hacer es darle un estilo al contenedor de ‘dropdown’ le vamos a indicar que el mismo va tener una posición relativa y que todo lo que se encuentre dentro de este se va a mostrar como ‘inline block’

.dropdown {

    position: relative;

    display: inline-block;

}

Ahora como vemos mas arriba (link 1 a 3) eso seria el contenido desplegable del menú… pero entonces tendríamos que esconderlo digamos y que solo se muestre al posar el cursor, ahora vamos a hacer eso.

Lo primero que vamos a hacer con el contenido es darle una posición abosluta, es referente al ‘padre que este tiene’ para el resto de los elementos colapsen entre si se usa para que cuando el contenido adyacente este en si se usa absolut (No entendí una goma mister)

.dropdown-content {

    display: block;

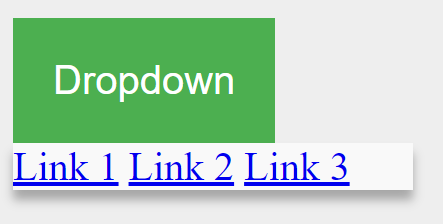
    position: absolute;

    background-color: #f9f9f9;

    min-width: 160px;

    box-shadow: 0px 4px 4px rgba(0,0,0,0.3);

}



Ahora vamos a usar la combinación de selectores para que se muestre pasando hacia abajo no y tenga sentido, vamos a hacer lo siguiente.

NOTA APARTE     padding: 12px 16px;

El primer valor de padding es vertical (arriba y abajo) y el segundo es horizontal!

.dropdown-content a {

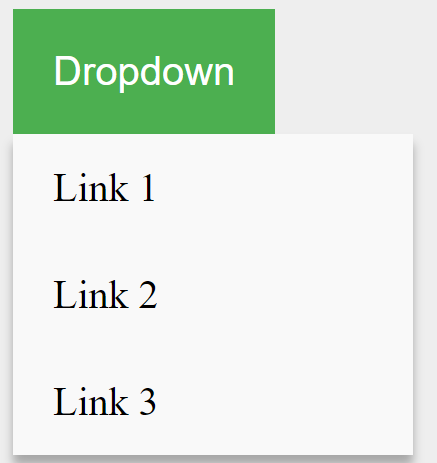
    color: black;

    padding: 12px 16px;

    text-decoration: none;

    display: block;

}



Va tomando forma viejo!

.dropdown-content a:hover {

    background-color: hsl(261, 96%, 45%);

}

Atentos no al color falopa que elegi sino como llamamos a la función de hover y link particularmente!

Vamos a hacer un formulario de contacto.

<body>

    <form>

        <label> Nombre </label>

        <input type="text" id="nombre" name="nombre" placeholder="nombre" />

        <label> apellido </label>

        <input type="text" id="apellido" name="apellido" placeholder="apellido" />

        <input type="submit" value="aceptar">

    </form>

</body>

input[type=text] {

    width: 100%;

    padding: 12px 20px;

    margin: 8px 0;

    display: inline-block;

    border: 1px solid #ccc;

    border-radius: 4px;

    box-sizing: border-box;

}

input[type=submit] {

    width: 100%;

    background-color: #4caf50;

    color: white;

    padding: 14px 20px;

    margin: 8px 0px;

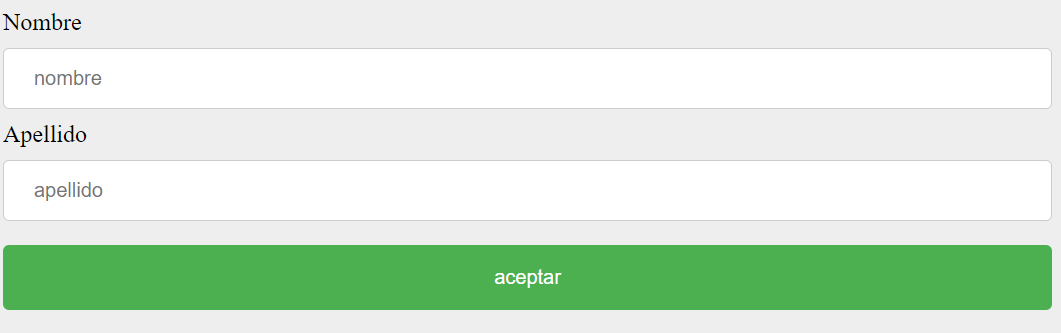
    border: none;

    border-radius: 4px;

    cursor: pointer;

}

Fijemosnos que aparecieron los corchetes para seleccionar funciones especificas dentro de una función.



Vamos a ahora a meter el formulario en si, pero en una cajita’.

 <div class="box">

    <form>

        <label> Nombre </label>

        <input type="text" id="nombre" name="nombre" placeholder="nombre" />

        <label> Apellido </label>

        <input type="text" id="apellido" name="apellido" placeholder="apellido" />

        <input type="submit" value="aceptar">

        </form>

    </div>

Solamente agregamos el div class box para crearla, y editarla en CSS.

.box {

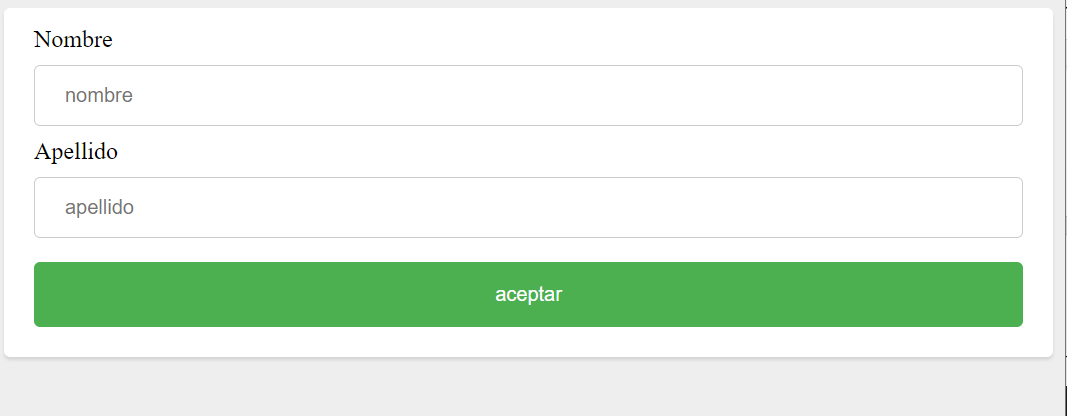
   background-color: white;

   padding: 12px 20px;

   border-radius: 4px;

   box-shadow: 0px 2px 2px rgba(0,0,0,0.1);

}



Buen formulario de contacto como quería asmel jsjsjsjsj

***CSS AVANZADO***

Ahora vamos a ver backgrounds, y mas propiedades! Desde el fondo fijo, poner imágenes de fondo y mas!

<div class="fondo">

 <p>lala</p>

 <p>lele</p>

 <p>lili</p>

 <p>lolo</p>

 <p>maria</p>

</div>

</body>

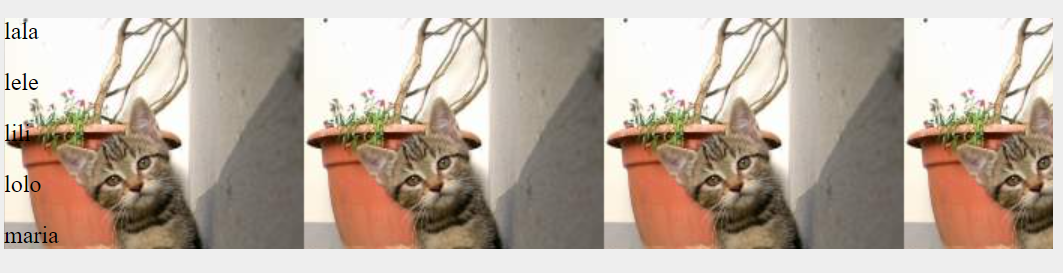
.fondo {

    background-color: aqua;

    background-image: url('https://placekitten.com/200/300');

}

De esta manera agregamos una imagen de fondo, pero…



La imagen se adapta dependiendo del contenido ósea nunca va aexceder ese contenido de ‘.fondo.

    height: 300px;

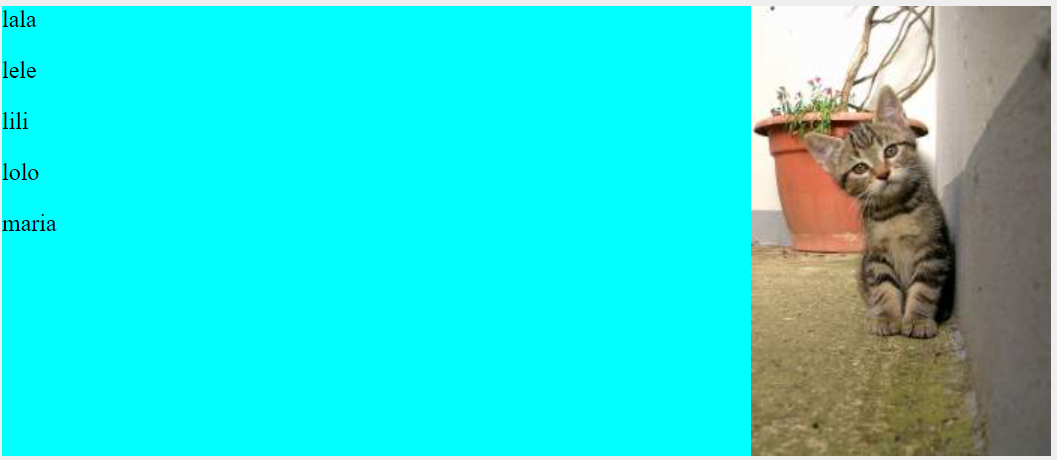
Con un simple height lo podemos solucionar por ejemplo entre tantas. Pero no del todo por que la imagen se repite en base al height.

    background-repeat: none;

Con eso la imagen no se repite pero queda el fondo con lo que sobra del padding e imagen.

Con el repeat podemos especificar si se repite en (Y,X,Z) vertical o horizontal entre tantas.

    background-position: right bottom;



Ahora algo importante es que vamos a ver como hacer para que cuando scrolleamos nos acompanie la imagen!

    background-attachment: fixed;

bueno pero la cuestión still remains, tenemos que escalar la imagen.

   background-size: cover;

Ojo por que ami no me gusto como quedo pero la función es esa!

.fondo {

    background-color: aqua;

    background-image: url('https://placekitten.com/200/300');

    height: 300px;

    background-repeat: no-repeat;

    background-position: left top;

    background-attachment: fixed;

    background-size: cover;

}

Y si te dijera que podemos escribir todo eso en 1 linea de código? Bueno veamos!

    background: green url('https://placekitten.com/200/300') no-repeat left top scroll ;

Personalmente para arpender y todo eso es mucho mas factible la otra forma.

Ahora vamos a ver, Colores y Opacidad.

Vamos a empezar viendo el RGBA.

  background-color: rgba(0,0,0,1);

Las siglas :red green blue Alpha y Alpha siendo ‘Opacidad’

Bueno, continuando existe ‘HSL’ (Hue-Saturation-Lightness)

   background-color: hsl(hue, saturation, lightness)

La verdad que es un color y lo manipulas para eso abris chat GPT o un quick googlee para estas pelotudeces.

Bueno ahora vamos a ver ‘Fondos con Gradientes’

Que es esto? Bueno que tengas un color y después siga con otro color gradualmente. (Como un tunning de un auto en NFS Xd)

Para esto no vamos a usar ‘background color’ sino backgroundimage’ Si exactamente! Bueno veamos.

background-image: linear-gradient(direccion, color, colo2);

background-image: linear-gradient(red,yellow)



background-image: linear-gradient( to right,red,yellow);

Con dirección includia!



Podemos ponerle direcciont hasta diagonalmente usando to bottom left por ejemplo

Y hasta ponerle los grados para hacer lo mismo.

Bueno veremos SOMBRAS a continuación.

.box {

    box-shadow: 2px 2px 4px rgba(0,0,0,0.3);

}

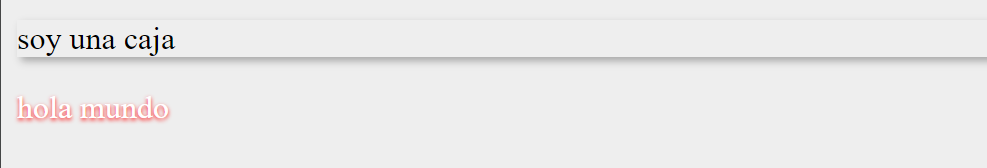
Una vez que creamos una caja para hacer el ejemplo podemos ver como le asignamos con la función boxshadow y los pixeles es la alineación de la sombras! Y el ultimo el blur.

.text {

    color: white;

    text-shadow: 0px 1px 2px rgba(255,0,0,1);

}



Ahora vamos a continuar viendo Transformaciones en 2D.

Especificamente vienod la propiedad de transfrom la misma nos permite tomar un elemento,rotarlo,escalarlo, acostarlo o torcerlo.

    transform: rotate(20deg);

Como dice la nomeclatura, creamos una caja para demostrar la misma y ‘rotate (xx deg) indica los grados a rotar sobre su EJE!.

.box {

    position: relative;

    left: 50px;

    top: 50px;

    width: 100px;

    height: 100px;

    background-color: white;

    box-shadow: 1px 2px 2px rgba(0,0,0,0.2);

    transform: rotate (20deg);

}



    transform: rotate(-40deg);

Tambien le podes agregar valores negativos, y que sucede con esto?



Creamos un rombo xD, a tener en cuenta!

    transform: translate(50px, 100px);

Vamos a ver ahora una conocida del pasado ‘translate’ la misma traslada la ‘caja’ en este caso o el objeto siendo el primer valor su eje en X y el segundo en Y.

Ahora vamos a ver la funcion ‘scale’

    transform: scale(2, 2);



Nuestro contenido crecio un 200% de manera vertical y horizontal. Una de las cosas copada de la funciones copadas que hace el mister es que cuando el pose el cursor crezca el elemento.

.box:hover {

    transform: scale(1.05, 1.05);

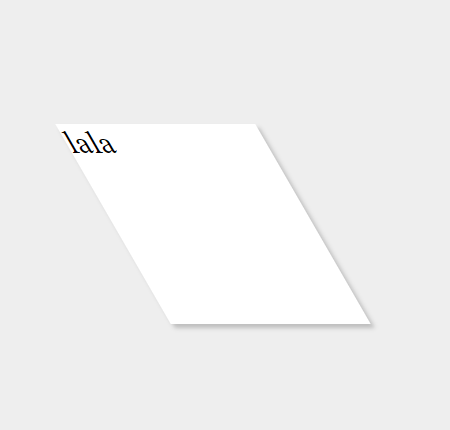
}

   transform: scaleY(1.05);

Podemos especificar el eje directamente ☺

Y por ultimo los queda la función de ‘Skew’ significa retorcer y la misma recibe un angulo.

    transform: skew(30deg);



    transform: skewX(30deg);

Ya vimos transformaciones en 2Dimensiones ahora vamos a ver en 3!

    transform: rotateX(60deg);

La verdad no entiendo mucho a lo que fue ese video por que mostro como rotar en X Y Z pero bueno nada joya, sigamos!

Ahora vamos a ver Transiciones Animadas, las mismas permiten que podamos ir de un valor x a otro valor x PERO de una manera suave es decir va a tomar una duración de (1 seg) digamos como ejemplo y el ancho de 100px y lo podemos aumentar a 200px en el transcurso de 1 segundo! Para eso veamos lo siguiente.

La propiedad ‘transition’ recibe 2 valores la primera es la propiedad que nosotros queremos que nosotros tenga la transición, en este caso lo vamos a hacer con width seguido de cuanto tiempo queremos que dure la transición.

    transition: width 0.3s;

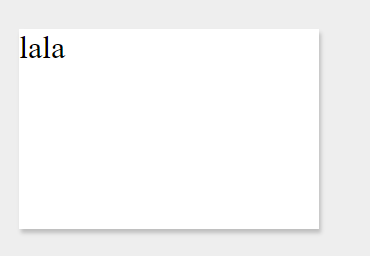
.box:hover {

    width: 150px;

}

Sin posar el cursor.





Vemos como lleva su ancho al numero estipulado y lo lleva a la duración de 0.3seg.

En el código que proporcionaste, la propiedad **transition** se utiliza para controlar cómo se animan los cambios en las propiedades de un elemento en CSS. En este caso específico, estás aplicando la transición a la propiedad **width**. La parte del código que dice **transition: width 0.3s;** indica que cualquier cambio en la propiedad **width** de la clase **.box** se animará durante un lapso de tiempo de 0.3 segundos.

Cuando el ratón pasa sobre el elemento con la clase **.box** y activa el seudoelemento **:hover**, la anchura de la caja cambia de 100px a 150px de forma suave y gradual durante el período de tiempo especificado de 0.3 segundos. Esto crea una transición animada en lugar de un cambio instantáneo de tamaño.

Si cambiaras el valor de **0.3s** a, por ejemplo, **1s**, la transición tomaría un segundo completo para completarse. Del mismo modo, podrías especificar el tiempo de transición utilizando valores en milisegundos, por ejemplo, **300ms**, que es lo mismo que **0.3s**.

En resumen, la parte **0.3s** en la regla **transition** indica la duración de la transición en segundos (s) o milisegundos (ms) cuando se produce un cambio en la propiedad especificada.

    transition: 0.3s;

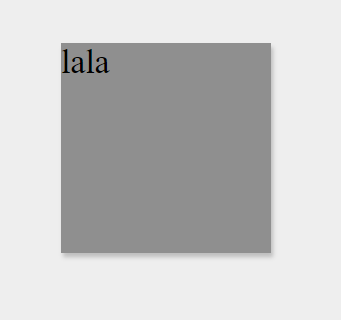
}

.box:hover {

    transform: scale(1.05, 1.05);

    background-color: rgba(0,0,0,0.4);

}



Al posar el cursor vemos como escala la imagen y cambia el color de fondo!

Ademas le podemos indicar, que queremos que la transición no empieze de manera inmediata, veamos.

   transition-delay: 1s;

CSS GRID

Ahora vamos a ver CSS Grid o ‘Grilla’ CSS Grid es un sistema que se encuentra en CSS3 para distribuir contenido de forma homogénea y mas fácil. (Procede a explicar como antes era complicadísimo) Las Grillas es un sistema de celdas que nos permite distribuir el contenido muy simlar a una tabla PERO no es una tabla.

Podemos distribuir el contenido en columnas y filas pero también podemos darle mas instrucciones cambiar el alto, ancho, posición, altura predefinida o ancho predefinido.

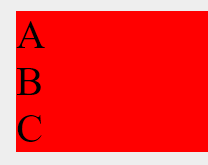
BUENO VEAMOS PESHI@!!

.grid{

    display: grid;

}

Esto hace que todos los elementos se despliegan como grilla, ósea como una única columna.

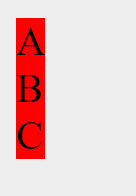


.grid{

    display: inline-grid;

}

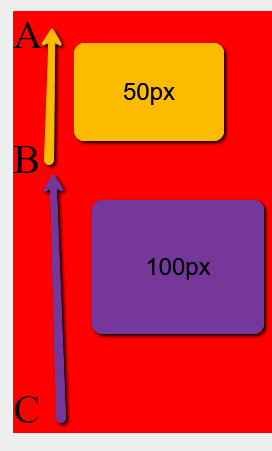
Esto en lugar de usar todo el ancho es que va a utilizar solamente el espacio que tiene definido dentro de sus elementos.



A continuación vamos a ver como podemos dar de manera explicita dimensiones a las filas que vamos a estar creando dentro de nuestra grillas.

    grid-template-rows: 50px 100px;

Con esto le indicamos quie las filas tienen que tener una altura de 50px y de 100 px.



Claro, ‘C’ no se vio aceptada por que no le ingresamos valores, entonces tiene una altura determinada en base al contenido como solo es una letra o una fila de texto solo va a necesitar la altura mínima requerida para que este.

.grid{

    display: grid;

    grid-template-columns: 90px 50px 120px;

}



La función nombrada da valor al ancho de las columnas, la columna de ‘A y D’ tiene 90px la columna de ‘B y E’ tiene 50px y la columna de ‘C y F’ tiene 120 px.

grid-template-columns: 1fr 2fr 3fr;

Tambien podemos usar fracciones, esto significa que va a tomar todo el ancho de la pagina y lo va a fraccionar en los prametros que nostors eligamos.

A cotninuacion veremos como podemos asignarle un tamanio mínimo y máximo respecto al ancho en las celdas de nuestra grilla.

**Este video no se puede ver por que no carga lpm**

    grid-template-rows: minmax(100px, auto);

Con esto le damos el ancho maxo de 100 px y que lo demás se automatize.

    grid-template-rows: minmax(100px, auto);

    grid-template-columns: minmax(auto, 50%) 1fr 3em;

Voy a pedirle a chatgpt que explique lo mencionado.

¡Hola! Claro, puedo ayudarte a entender este código de CSS que utiliza la característica de Grid o Grilla.

1. **grid-template-rows: minmax(100px, auto);** Este código establece la disposición de las filas en el grid. **minmax** define un rango para el tamaño de las filas. En este caso, la altura mínima de la fila es de 100 píxeles, pero puede aumentar automáticamente si el contenido excede esa altura mínima.
2. **grid-template-columns: minmax(auto, 50%) 1fr 3em;** Este código define la disposición de las columnas en el grid. Aquí, también se utiliza **minmax** para definir un rango para el tamaño de las columnas. La primera columna tiene un ancho mínimo automático, pero no puede exceder el 50% del ancho total del grid. La segunda columna ocupa el espacio restante en la grilla, indicado por **1fr** (fracción). La tercera columna tiene un ancho de 3 'em', donde 'em' es una unidad relativa basada en el tamaño de fuente actual. Un 'em' representa la anchura de la letra 'M' en la fuente actual, lo que significa que su valor puede cambiar dependiendo del contexto de la página.

En resumen, el código está configurando un grid con filas y columnas específicas. **minmax** se utiliza para establecer un rango de tamaño flexible, **1fr** se utiliza para asignar una fracción del espacio restante y **em** es una unidad relativa que depende del tamaño de la fuente actual. Espero que esto aclare tus dudas. Si tienes más preguntas, no dudes en preguntar.

Excelente chat gpt lpm claro y preciso.

A continuación veremos como repetir dimensiones con la función ‘repeat’.

Veremos como podemos asignar un alto y un ancho pero de manera indefinida para todas las filas y columnas además de usarlo en partes intermedias de nuestra grid.

‘Repeat’ acepta (2) argumentos, el primero es cuantas veces quiero repetir el valor del segundo argumento.

    grid-template-rows: repeat(4, 50px);

De esta manera estamos diciendo en el ejemplo de rows

Repeti 4 filas de 50 px.

Lo mismo se hace con columnas por ejemplo.

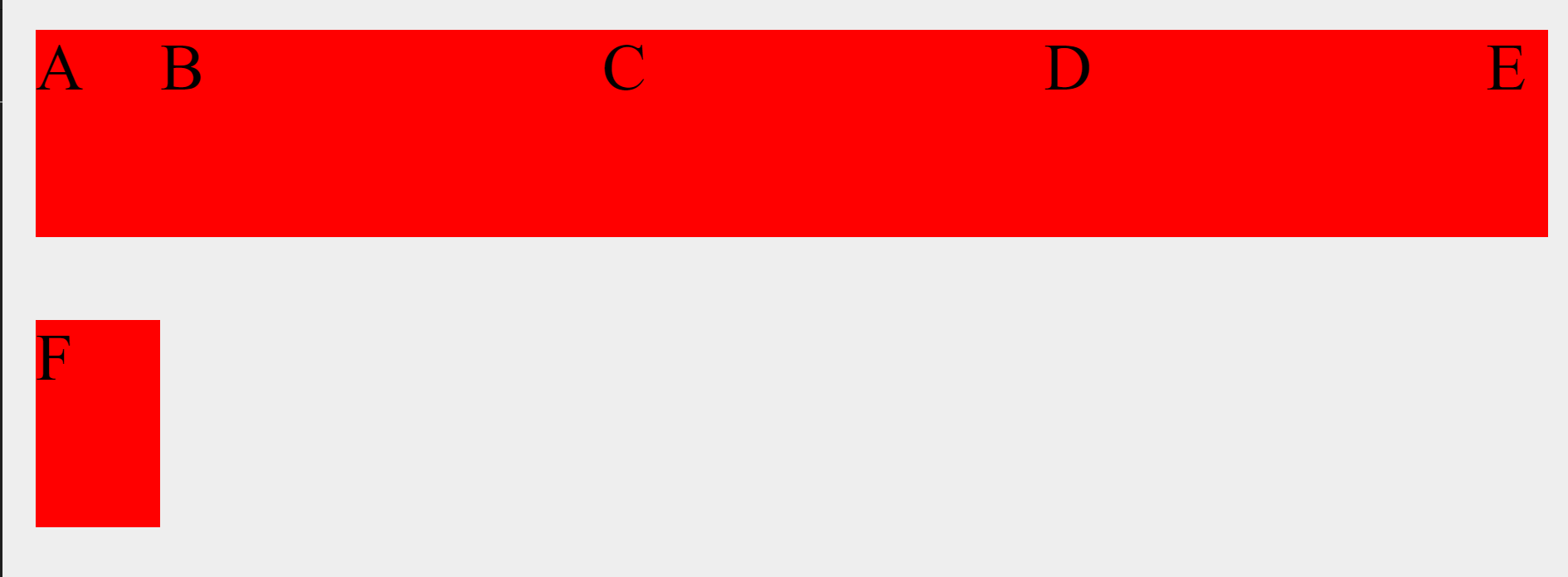
Otra cosa

   grid-template-columns: 30px repeat(3, 1fr) 15px;

De esta manera nuestra primera columna va a ser de 30px después lo que viene divide en fracciones y la ultima columna esta de 15px.

Aveces nosotros cuando estemos construyendo nuestras grillas no vamos a querer que esten pegadas unas con otras, lo que haremos ahora es que crearemos un espacio o una brecha entre medio.

    grid-row-gap: 20px;



Ahora se creo una brecha en la filas y lo mismo con las columnas.

    grid-column-gap: 10px;



Pero es mas podemos hacerlo en un solo código con la siguiente formula

    grid-gap: 20px 10px;

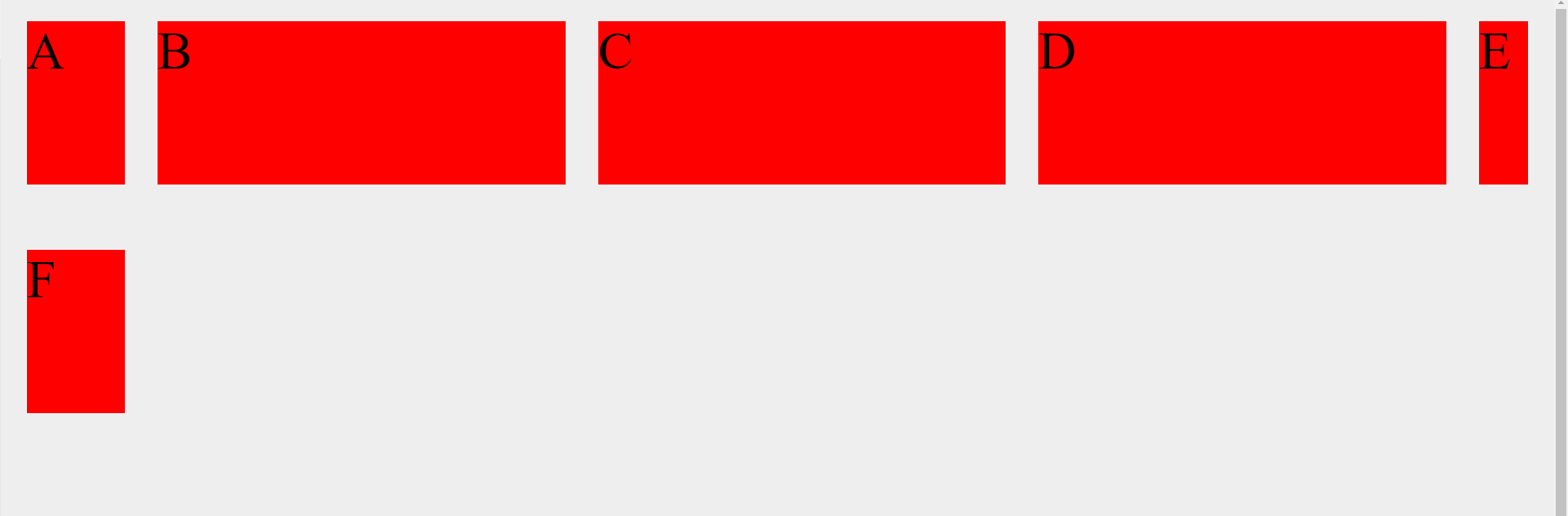
el primer valor es para las row y el segundo para las columns.

Tambien le podemos entregar 1 solo valor y va a ser tomado para row y columns alike.

Hasta ahora vimos elementos que pueden permanecer dentro de su propia celda, pero que pasaría si quisiéramos que nuestros elementos en lugar de utilizar solo 1 celda utilize todo lo que hay ‘arriba’ para hacer un header o un elemnto que sea toda la pagina en forma de columna lateral.

Veamos!

Las propiedades son: grid-column-start y grid-column-end. Y grid-row-start y grid-row-end. Pero que sucede… estas no pueden ser aplicadas a la grilla en su totalidad sino especificar para una sección especifica.



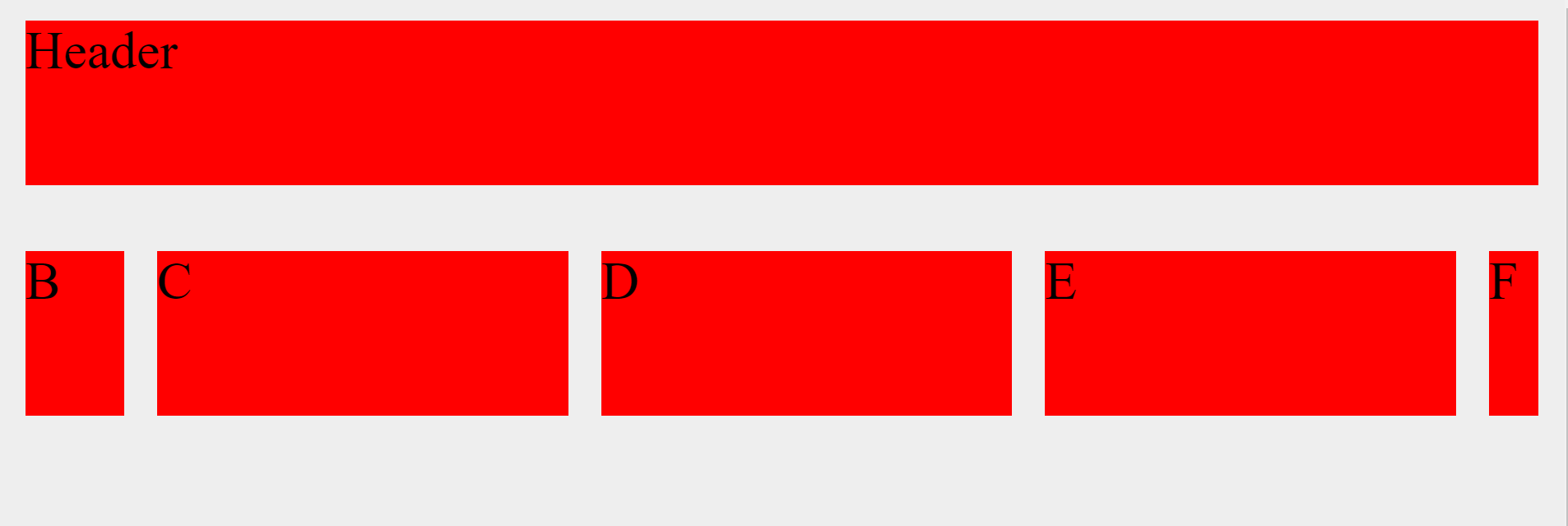
Podemos ver las líneas entre las filas, antes de empezar A antes de que empieze B y sucesivamente. Si contamos tenemos 6 en total! Y esto es importante.

.header {

    grid-column-start: 1;

    grid-column-end: 6;

}



.header {

    grid-column-start: 1;

    grid-column-end: 6;

    grid-row-start: 1;

    grid-row-end: 3;

}

Le asignamos mas espacio al header tomando filas también.



Esto se puede simplificar en 1 codigo también.

.header {

    grid-column: 1/6;

    grid-row: 1/3;

}

Ahora vamos a ver como otorgarles nombres a las grillas y modificarlas.

    grid-template-rows: [row-1-inicio] 50px;

Le indicamos el nombre y el valor en pixeles.

    grid-template-rows: [row-1-inicio] 50px [row-2-inicio] 50px;

    grid-template-columns: [col-1-inicio] 1fr [col-2-start] 1fr [col-3-start] 1fr;

.header {

    grid-column: col-1-inicio / col-4-fin;

    grid-row: row-1-inicio / row-3-fin;

}

Entendamos que esto es la forma de trabajar opino yo por que estas dando nomeclatura a una grilla para que le código sea legible y entnedible.

Veamos como entra ‘repeat’. La misma ofrece una mejor forma de asignar nombres a nuestras líneas.

grid-template-rows: repeat (2, [row-iniio] 50px [row-fin]);

Aca le indicamos que repita 2 veces con el nombre de row-inicio que tenga un altura de 50px y que la siguiente línea se llame row-fin.



La línea de arriba de header es la de row inicio. PERO lo interesante es que la línea violeta que hace referencia es que va a tener el nombre de row-fin y row inicio 2.

La ultima línea debajo de D, E Y F. es la de rowfin 2 y rowinicio2

Me maree duro pero bueno veremos Xd

    grid-template-columns: repeat(3, [col-inicio] 1fr [col-end]);

Lo mismo con las columnas, y aclaración personal esto es lo mismo que las fomras anteriores que vimos hoy! Se displayea igual es cuestión de orden y gustos.

Ahora vamos a ver otra forma de entregarle una distribución un poco mas descriptiva a nuestra app.

.grid{

    display: grid;

    grid-template-areas: "header header";

                          "Content Sidebar";

                          "Footer Footer";

    grid-template-rows: 150px 1fr 100px;

    grid-template-columns: 1fr 200px;

}

En CSS Grid, **grid-template-areas** te permite crear un diseño de rejilla nombrando áreas específicas en tu cuadrícula. Las áreas deben estar definidas entre comillas y separadas por espacios o tabulaciones. Cada conjunto de comillas representa una fila en tu cuadrícula y cada palabra dentro de las comillas representa una celda.

En tu código:

cssCopy code

.grid { display: grid; grid-template-areas: "header header" "Content Sidebar" "Footer Footer"; grid-template-rows: 150px 1fr 100px; grid-template-columns: 1fr 200px; }

Estás definiendo una cuadrícula con tres filas y dos columnas. La primera fila tiene dos celdas llamadas "header", la segunda fila tiene dos celdas, una llamada "Content" y la otra "Sidebar", y la tercera fila tiene una celda llamada "Footer". Estas áreas deben coincidir con las áreas utilizadas en tu HTML para que la cuadrícula se muestre correctamente.

Esta técnica es útil para diseñar diseños de cuadrícula más complejos y permite organizar tu diseño de una manera más visual y clara.

    <body>

        <div class="grid">

        <div class="red"> Header </div>

        <div class="red"> Header </div>

        <div class="blue">

            <div> Content </div>

            <div> Content </div>

            <div> Content </div>

            <div> Content </div>

        </div>

        <div class="blue"> Sidebar </div>

        <div class="red"> Footer </div>

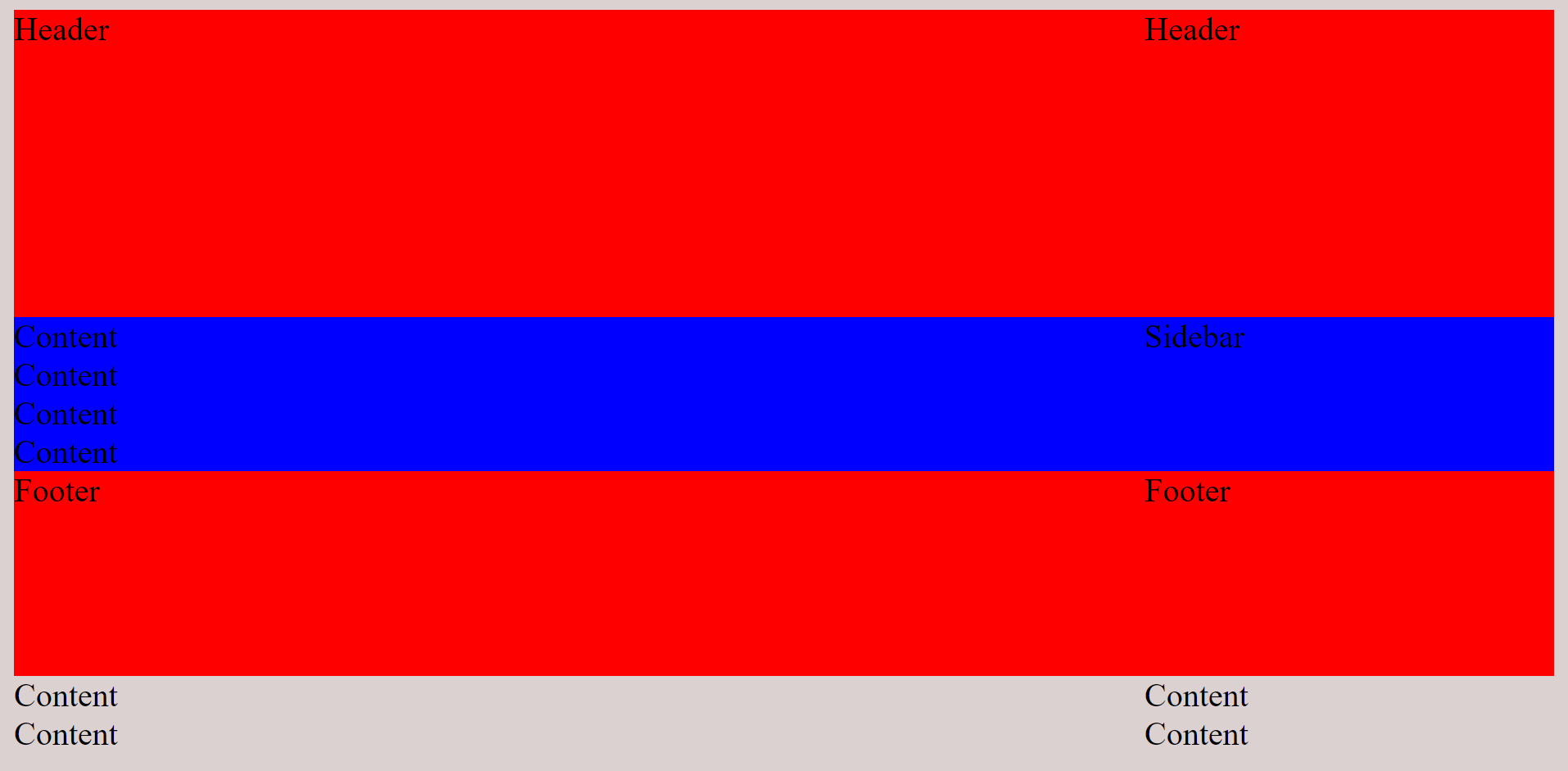
        <div class="red"> Footer </div>

    </div>

    </body>



Grillas implícitas es todo el contenido que esta fuera del contenido de la grilla especificado en CSS ósea.



La parte abajo del footer que dice content también vendira a ser el ‘grid implicito’

   grid-auto-rows: 140px;

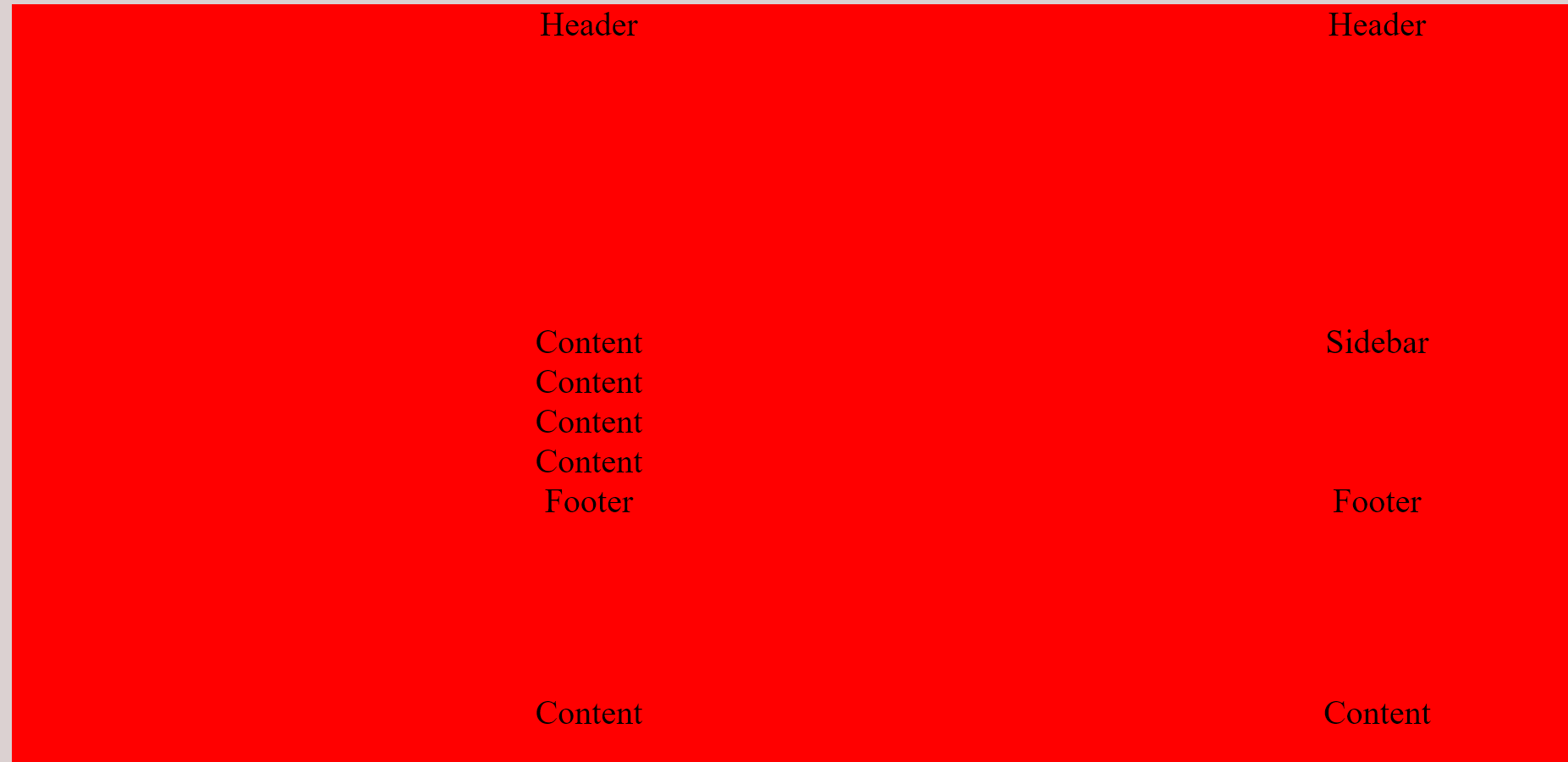
Por ejemplo con esta función todas las grillas que no fueron decalradas de forma de implícita obtienen unas filas de 140 px.

    grid-auto-flow: column;

Podemos otorgarle tmabien el formato de row o columna.

Ahora veremos alineamiento y justificamiento de elementos en grilla.

    justify-items: center;



En justify-items, la función por defecto es start.

    justify-items: end;

Los empieza de derecha a izquierda.

Justify-items: stretch; estira el contentido.

¡Por supuesto! La propiedad CSS **justify-items** se utiliza en contenedores o elementos de rejilla (grid) y se utiliza para alinear los elementos secundarios a lo largo del eje en línea principal del contenedor, en relación con el área de la celda de la rejilla.

Cuando se establece en **end**, los elementos secundarios se alinearán en el extremo final del contenedor de la rejilla, es decir, hacia el final del eje en línea principal.

Supongamos que tienes un contenedor de rejilla con varios elementos secundarios dentro. Si estableces **justify-items: end;** en ese contenedor de rejilla, los elementos secundarios se alinearán al final del contenedor a lo largo del eje en línea principal.

Aquí tienes un ejemplo de cómo se aplicaría en código CSS:

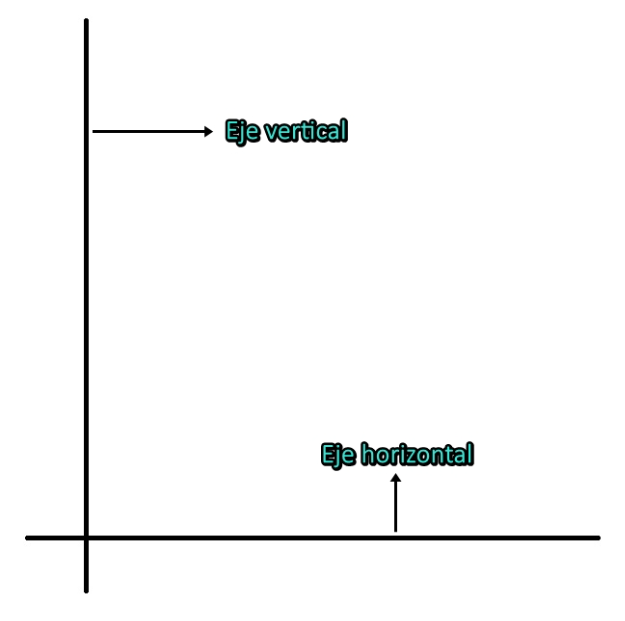
cssCopy code

.grid-container { display: grid; grid-template-columns: auto auto auto; justify-items: end; } .grid-item { padding: 20px; border: 1px solid black; }

En este caso, todos los elementos secundarios dentro de **.grid-container** se alinearían en el extremo final del contenedor de la rejilla.

Espero que esto te ayude a comprender la función **justify-items: end;** en CSS. Si tienes más preguntas o si necesitas más aclaraciones, no dudes en preguntar.

Ahora vamos a ver como podemos alinear nuestros elementos pero con el eje vertical.



    align-items: start;

align-items, tiene valores similares, pero a diferencia de justify-items no tiene valor por defecto el start, es una diferencia sumamente importante a tener en cuenta.

Align-items: center;

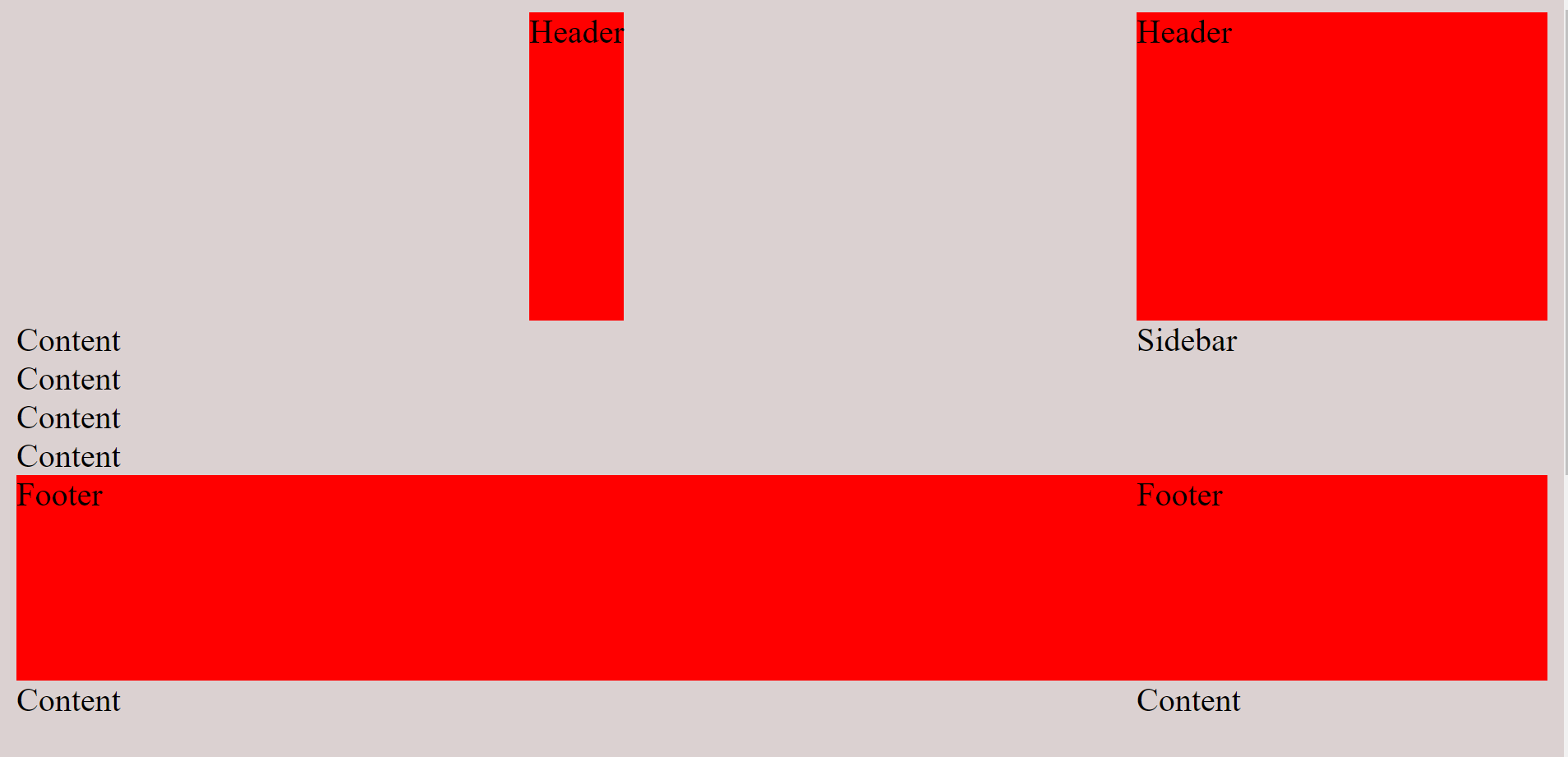
Vamos a ver como alinear y justificar un solo elemento en grids.

      <div class="red item-1"> Header </div>

.item-1 {

    justify-self: center;

}



El header esta centrado, y podemos usar el ‘stretch y en end’

Ahora veamos como hacerlo en el eje vertical

.item-1 {

    align-self: center;

}

Aca podemos utilizar center, end y stretch.

Ahora veremos como alinear la grilla en este caso a la distribución de la pantalla esto quiere decir que si nuestra grilla va a tener 4 col y 4 filas vamos a tomar esas filas y columnas completas y vamos a moverlas a donde queramos!

.grid {

    display: grid;

    width: 100%;

    grid-template-columns: repeat(4, 50px);

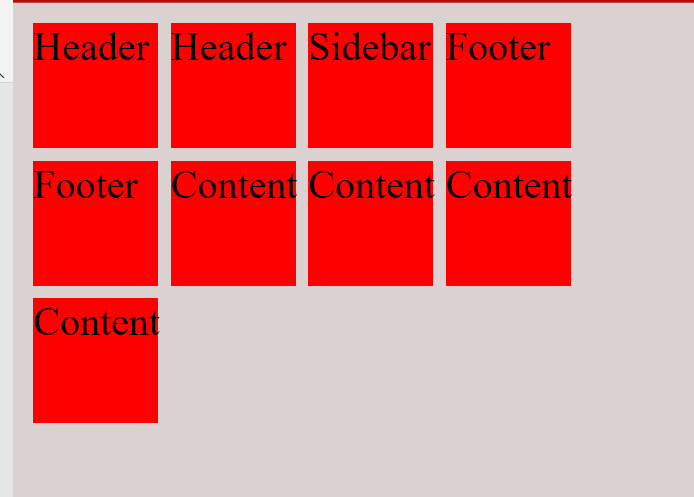
    grid-template-rows: repeat(4, 50px);

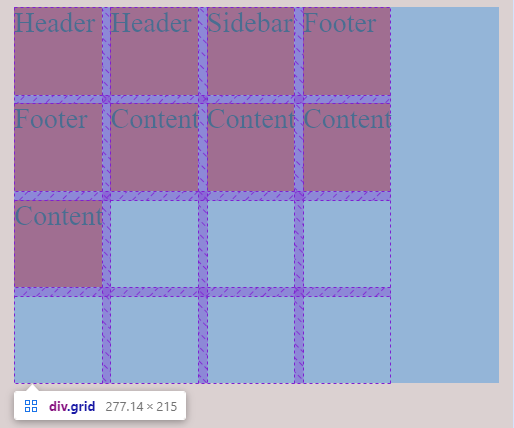
    grid-gap: 5px;

    justify-content: start;

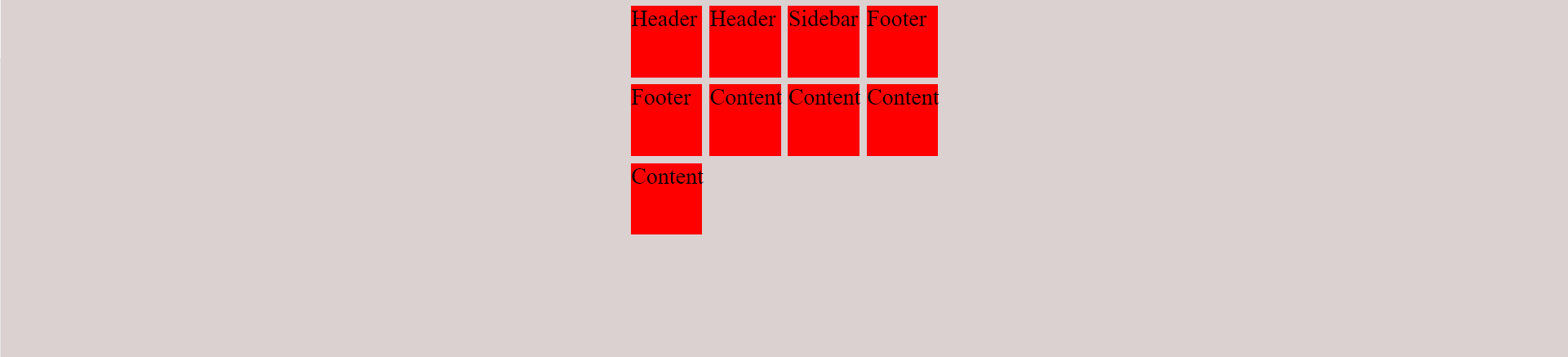
}

Modificamos nuestra grid





    justify-content: center;



Ahora vamos a ver como podemos hacer para agregar un espaciado a cada uno de estos elementos

    justify-content: space-around;



    justify-content: space-between;

con space-between respeta que empieza el contenido al inciio de izuiqerada a derecha fin.

Vamos a ver como podemos distribuir nuestro contenido también con espaciados pero que el mismo se vea completamente uniforme y no como nuestra propiedad anterior. Ya que en la anterior los px no eran los mismos.

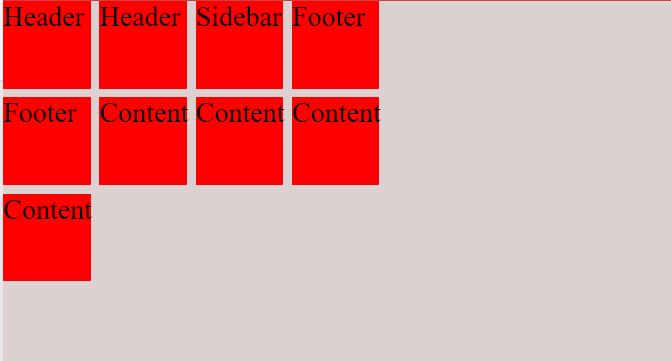
    justify-content: space-evenly;



De manera equitativa crea los espacios, ahora veamos los mismo con el eje vertical.

    justify-content: start;

    align-content: start;



Pero para mover el contenido con end y ponerlo abajo es importante especificar las dimensiones

html {

    height: 100%;

}

body{

    background-color: rgb(219, 209, 209);

    margin: 0;

    height: 100%;

}

    align-content: end;



Con align-content tenes los mismos comandos que justify-content

Un breve resumen de GRIDS por Chat GPT

No te preocupes, aprender CSS Grid puede ser desafiante al principio, pero una vez que entiendes los conceptos básicos, se vuelve más fácil. CSS Grid es una poderosa herramienta para crear diseños complejos y responsivos en tu sitio web. Para ayudarte a comprender mejor, aquí hay una breve introducción a los conceptos fundamentales de CSS Grid:

1. **Display Grid:** Para comenzar a usar CSS Grid, debes definir un contenedor como una cuadrícula estableciendo su propiedad **display** en **grid**.
2. **Definir Filas y Columnas:** Puedes definir el número y el tamaño de las filas y columnas usando las propiedades **grid-template-rows** y **grid-template-columns**.
3. **Posicionar Elementos:** Puedes posicionar elementos en la cuadrícula utilizando las propiedades **grid-column** y **grid-row**. También puedes usar **grid-area** para asignar un nombre a una región de la cuadrícula y luego hacer que un elemento ocupe esa región.
4. **Alineación:** Puedes alinear elementos en la cuadrícula utilizando propiedades como **justify-items**, **align-items**, **justify-content** y **align-content**.

If you want to count grid lines from the right instead of the left, you can give **grid-column-start** and **grid-column-end** negative values. For example, you can set it to -1 to specify the first grid line from the right.