Introducción a CSS

Conceptos a aprender

- ¿Qué es CSS?
- Cómo se estructura el CSS
- Colores
- Selectores
- Bordes, margenes y relleno
- Selector por atributo
- Posicionamiento
- Tipografias
- Flex box
- Grid

Introducción

CSS (Cascading Style Sheets, en español Hojas de Estilo en Cascada) es un lenguaje de estilo utilizado para describir la apariencia y el formato de un documento escrito en HTML o XML. Con CSS, puedes controlar el color, la fuente, el espacio entre elementos, la disposición y muchas otras cosas.

CSS se utiliza principalmente para separar el contenido de un documento HTML del diseño y la apariencia. Esto permite que el contenido sea leído y modificado de manera más fácil, y también permite que el mismo contenido sea utilizado en diferentes contextos, como en diferentes dispositivos o en diferentes estilos de diseño.

Antiguamente los estilos eran incluidos directamente como etiquetas HTML, por ejemplo, la etiqueta <font> que servía para cambiar la tipografía, o la etiqueta que servía para centrar un contenido. En la actualidad, en lugar de eso, utilizamos CSS que puede estar de tres formas diferentes:

- En un archivo aparte
- En una etiqueta HTML especial llamada <style>
- Directamente como una propiedad de una etiqueta (Usando la propiedad style="").

Generalmente la forma más recomendada es como un archivo aparte el cual tiene como extensión ‘css’. Para linkear un archivo CSS en un HTML, dentro de la etiqueta <head> tenemos que agregar una etiqueta <link>, de la siguiente manera:

```
<link rel="stylesheet" href="estilo.css" />
```

En este ejemplo, el archivo CSS se llama ‘estilo.css’, pero puede tener cualquier nombre, siempre tratando de respetar el tipo de archivo ‘css’. En el ejemplo, la etiqueta <link> tiene dos atributos, el primero es ‘rel’ y lleva como valor ‘stylesheet’, que significa en inglés, hoja de estilos. Simplemente es para describir qué tipo de archivo estamos embebiendo; la segunda es ‘href’, que es la URL del archivo a embeber, esta URL puede ser relativa o absoluta, en la mayoría de los casos utilizaremos una dirección relativa, si el archivo ‘estilo.css’ está en la misma carpeta que nuestro archivo html, entonces simplemente tenemos que escribir ‘estilo.css’, si estuviera, por ejemplo, en una subcarpeta ‘estilos’, el valor de href tendría que ser ‘estilos/estilo.css’. Hay que asegurarse de que la ruta del archivo sea correcta para que el navegador entienda dónde está el archivo.

Ejemplo de CSS

La estructura CSS es bastante diferente del HTML. Funciona de la siguiente ma nera:

```
[selector] {
   [propiedad]: [valor];
}
```

Donde ‘selector’ es algo a lo que se apunta, propiedad es lo que queremos afectar, y valor es cómo lo queremos afectar. Siempre entre la propiedad y el valor van dos puntos (:) y al final de la regla va punto y coma (;). Un selector puede ser el nombre de una etiqueta, por ejemplo:

```
html {
 background-color: black;
 color: white;
}
```

En este caso, el “selector” tiene un valor de “html”. Esto significa que el bloque de reglas (delimitado por {}) afectará a la etiqueta html. Además, tenemos dos reglas, la primera ‘background-color’ afecta el color de fondo, mientras que ‘color’, afecta el color del texto. En este caso, cambiamos el fondo a negro, y el texto a blanco.

```
p {
 margin: 30px;
}
```

En este otro ejemplo de bloque de reglas CSS, nuestro selector es “p”, esto significa que TODAS las etiquetas <p> de nuestro documento HTML van a ser afectadas por las reglas de este bloque. En este caso, solamente tenemos una regla, “margin”, que afecta el margen de los elementos seleccionados, “30px” significa que tendrá un margen de 30 píxeles. Es importante recordar usar “px” ya que es la unidad de medida, ‘30’ sin ‘px’ no funcionará.

Margin acepta diferentes valores, en la forma en la que lo encontramos en el ejemplo, 30px afectará a todos los márgenes, el superior, el derecho, el inferior, y el izquierdo. También podemos hacer algo como esto:

```
margin: 30px 0;
```

En este caso, margin tiene dos valores, el primero corresponde al margen superior e inferior del elemento, y el segundo al derecho y el izquierdo. Entonces, si tiene un margin de ‘30px 0’, el margen superior será de 30 pixeles, el derecho de 0, el inferior de 30 píxeles, y el derecho de 0. Cuando tenemos un 0, no es necesario incluir el tipo de medida al que nos referimos, pero 0px también es válido.

```
margin: 30px 10px 30px 0;
```

Esta forma, con cuatro valores, es la más común, donde cada elemento corresponde a un margen. Para recordar el orden de referencias, podemos pensar en un reloj, el primero es el superior, el segundo es el derecho, el tercero es el inferior, y el cuarto es el izquierdo. Entonces, sabiendo esto, deducimos que el margen superior será de 30 píxeles, el derecho de 10 píxeles, el inferior de 30 píxeles, y el izquierdo de 0.

También podemos encontrar una forma menos común que es de tres elementos:

```
margin: 30px 10px 0;
```

En este caso, el primer elemento hace referencia al margen superior, el segundo a los márgenes de los costados, y el tercero, al inferior. Entonces, en este caso, el margen superior será de 30 píxeles, el derecho y el izquierdo serán de 10 píxeles, y el inferior será de 10 píxeles.

Colores

Hay varias formas de representar colores en CSS. La forma más común es mediante un código hexadecimal. En este caso, los colores comienzan con el numeral y luego van tres o seis números hexadecimales (Es decir, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F). Si el número tiene tres caracteres, el primero significa ROJO, el segundo, VERDE, y el tercero AZUL. 0 es 0% de ese color y F es 100%. Por eso, un #000 es un negro (Los tres colores al 0% son negro) y #FFF blanco (Los tres colores al 100% son blanco). Un #F00 será entonces un ROJO, un #0F0 será un verde, y un #00F un azul.

También pueden tener seis caracteres. Y esto es para expandir la cantidad de colores que se

pueden tener, #FFFFFF sigue significando blanco, esto es porque al usar seis cada dos caracteres, significan un color base. En este caso, para obtener un rojo sería #FF0000, un verde #00FF00, y un azul, #0000FF.

La segunda otra forma de seleccionar colores es mediante palabras clave en inglés, ‘blue’ es azul, ‘red’ es rojo, ‘black’ es negro. Existe una lista enorme para colores web, aquí podemos encontrar una lista: http://xahlee.info/js/css_color_names.html

Cabe rescatar que generalmente es más recomendable utilizar el formato hexadecimal para nombrar los colores, ya que es más legible.

Otra forma es usando RGB:

```
color: rgb(255, 255, 255);
```

El método rgb es parecido al hexadecimal (FF es equivalente a 255), pero el valor está con números comunes que van del 0 al 255. Hace falta definir ‘rgb()’ para que el navegador entienda la regla correctamente.

La forma rgb también puede ser expandida con la forma rgba, en la cual el cuarto número no es un color, sino la opacidad.

```
color: rgba(255, 255, 255, 1);
```

La opacidad en lugar de ir de 0 a 255 va de 0 a 1. 0 significa que es ‘invisible’ y 1 que es totalmente opaco. Se puede introducir valores como 0.5 para que tenga una opacidad al 50%. Esta forma no siempre funciona y depende totalmente de la compatibilidad del navegador con ella.

Finalmente, está el método hsl, por sus siglas en inglés (*Hue, Saturation, Lightness, es decir Matiz, Saturación y Luminocidad*).

- **Matiz**: Toma un valor en grados (0 al 360), básicamente representando una rueda de color, cerca del 0° están los rojos, cerca del 120° están los verdes, y cerca del 240° están los azules.
- **Saturación**: Toma un valor en porcentaje, 0% es un color totalmente desaturado (gris), y 100% es el color completo.
- **Luminocidad**: También toma un valor en porcentaje, 0% es completamente oscuro (negro) y 100% es completamente brillante (blanco).

```
color: hsl(0, 15%, 95%);
```

También puede ir con alpha, en este caso se usa hsla:

```
color: hsla(0, 15%, 95%, 0.5);
```

Selectores

Existe otra forma de seleccionar elementos, y es por IDs o clases. Podemos definirlas en el HTML de la siguiente manera:

```
<div id="cabecera">
 <h1>Cosas Dulces</h1>
 <ul>
   <li class="receta"><a href="galleta.html">Receta de Galletas</a></li>
   <li class="receta"><a href="cupcake.html">Receta de Cupcakes</a></li>
   <li class="receta"><a href="cakepop.html">Receta de Cakepops</a></li>
   <li><a href="faq.html">Preguntas frecuentes</a></li>
   <li><a href="contacto.html">Contacto</a></li>
 </ul>
</div>
```

Aquí podemos ver que tenemos un que contiene el resto de los elementos, y lleva un id de “cabecera”. En CSS podemos afectar a este elemento directamente usando el selector “#cabecera”.

```
#cabecera {
 background: #7396ff;
 font-family: Helvetica, Arial, Verdana, sans-serif;
 padding: 10px;
}
```

Aquí hacemos que el elemento con id “cabecera” tenga un fondo de color celeste, cambiamos la fuente del texto y le agregamos un “relleno” (padding) de 10px. La propiedad “font-family” puede llevar varios valores separados por coma, en este caso por defecto tratará de que se use “Helvetica”, si la computadora cliente no dispone de esta tipografía, tratará de usar “Arial”, y luego “Verdana”. Si ninguna de estas fuentes con nombre están disponibles, simplemente se le ordena que busque una tipografía de palo seco (sans-serif). La propiedad “padding” funciona similar a la propiedad “margin”.

Cabe destacar que cada ID solamente debe aparecer una vez en el documento HTML. Es decir, no deberíamos tener otro elemento (por más que sea una etiqueta distinta de) que tenga el id “cabecera”.

Agregando más reglas, podemos customizar nuestro contenido más todavía. Comenzamos editando el <ul> que está dentro del #cabecera.

```
#cabecera ul {
 padding: 0;
 list-style: none;
 margin: 0;
}
```

Notese que comenzamos el selector con el id “cabecera”, agregamos un espacio y escribimos “ul”. Esto significa que hacemos una “selección” de todos los <ul> dentro de #cabecera. Por defecto los <ul> tienen un relleno, así que al definir padding con un valor de ‘0’, lo anulamos. También tiene un estilo de lista, definido por la propiedad “list-style”, pero como nosotros no queremos las viñetas, le asignamos un valor de “none”. Finalmente, seteamos el margen también en ‘0’.

Siguiendo la misma lógica, cuando queremos afectar los <li> dentro de los <ul> dentro de #cabecera, tenemos que hacer esto:

```
#cabecera ul li {
 display: inline-block;
 padding: 10px;
 margin: 0 5px;
 border: 1px dashed #663300;
}
```

Aquí podemos ver unas propiedades nuevas. Primero, “display”, que define cómo se debe mostrar el elemento. Por defecto los <li> tienen un display: block, pero como queremos que cada elemento este uno al lado del otro, lo cambiamos por “inline-block”. Le asignamos un relleno de 10px y un margen con dos valores, 0 hacía arriba y abajo y 5px hacia los costados.

Finalmente, tenemos la propiedad “border”, que lleva varios valores para funcionar, el primero es el ancho del borde. El segundo es el tipo de borde, que en este caso es “dashed”, los valores pueden ser estos:

- **hidden**: Visualmente idéntico a none, hace invisible el borde.
- **dotted**: Borde punteado.
- **dashed**: Borde discontinuo, pequeños trazos intercalados por espacios.
- **solid**: Borde continuo.
- **double**: Formado por dos líneas rectas continuas, separadas entre sí por un espacio en blanco.
- **groove**: Borde hundido, visualmente se encuentra por debajo del nivel de la superficie de la pantalla.
- **ridge**: Borde saliente, visualmente parece que se encuentra por encima de la superficie de la pantalla.
- **inset**: Borde hundido, provoca que el elemento que encierra parezca que está por debajo del nivel de la pantalla.
- **outset**: Borde saliente, que provoca que el elemento que encierra parezca estar por encima del nivel de la pantalla.

Cabe aclarar que para los tipos “double”, “groove”, “Ridge”, “inset” y “outset”, hace falta que el ancho del borde sea mayor que 1px.

Por último introducimos el código del color que deseamos que tenga el borde.

Ahora, para continuar vamos a modificar la apariencia de los <a> dentro de los <li>, para lograr esto, seguimos la misma lógica que usamos antes, primero el id #cabecera, luego el <ul>, luego el <li> y finalmente el <a>, así:

```
#cabecera ul li a {
 color: #ffffff;
 text-decoration: none;
 font-weight: bold;
}
```

Aquí tenemos primero una definición del color del texto como blanco, luego la propiedad “text-decoration”, que con un valor de “none”, le quitamos el subrayado que tienen por defecto las anclas. Finalmente tenemos la propiedad “font-weight” que controla que tan “pesado” o “resaltado” debe ser el texto, al asignarle un valor de “bold”, hacemos que el texto se vea en negrita. También podemos usar valores de “normal”, para un peso normal; “lighter” para hacerlo más ‘liviano’; “bolder” para más ‘pesado’; o números del 100 al 900 para lograr distintos ‘pesos’ (solamente multiplos de 100, es decir, no sirve un 140, por ejemplo).

Finalmente necesitamos que los <li> que tengan la clase “receta” se vean ligeramente diferentes que los otros. Para lograr esto usamos el nombre de la clase, antecedido por el punto (.) marca una clase, como el numeral (#) marca un ID. Una clase puede aparecer todas las veces que sea necesario dentro de un mismo documento HTML.

```
#cabecera ul li.receta {
 background: #663300;
}
```

Entonces, primero comenzamos nuestro selector con el id #cabecera, seguimos con ul y terminamos con los li agregando .receta. Hay que notar que entre la “li” y “.receta” NO hay espacio, esto significa que estamos haciendo objetivo a todos los <li> que tengan la clase “receta”.

Ahora, digamos que queremos que el <h1> también tenga un color blanco. Podríamos hacer un selector para h1 y repetir el color de letra que habíamos usado para los <a>, pero también podríamos hacer un bloque que apunte a ambos, y quitar esa regla del bloque que apunta solamente a <a>. Usamos la coma para hacer dos o más selectores en lugar de uno.

```
#cabecera ul li a {
 text-decoration: none;
 font-weight: bold;
}

#cabecera h1,
#cabecera ul li a {
 color: #ffffff;
}
```

Quitamos la regla “color: #FFFFFF;” del bloque que apunta a los <a> y lo colocamos en este nuevo bloque, que vemos que apunta a los <h1> que estén dentro de #cabecera, y usando la coma para separar selectores, también apunta a los <a> que estén dentro de <li>, dentro de <ul>, dentro de #cabecera.

El fondo de un elemento

Para incluir una imagen como fondo de un elemento, utilizamos la propiedad CSS ‘background-image’ cuyo valor se escribe con la forma url('dirección'), así:

```
html {
 background-image: url("tigre.jpg");
}
```

Esto provocará que el fondo del html sea la imagen que elegimos, ‘tigre.jpg’. De nuevo, la URL tiene que ser o relativa, o absoluta.

Al agregarlo al elemento html, que tiene un tamaño del 100% de la ventana, la imagen ocupará todo el fondo, y si fuera más pequeña que la pantalla, se repetirá. Esto lo podemos controlar con la propiedad ‘background-repeat’.

```
html {
 background-image: url("tigre.jpg");
 background-repeat: no-repeat;
}
```

La propiedad ‘background-repeat’ puede llevar los siguientes valores:

- repeat: Es el funcionamiento por defecto, la imagen se repite hasta llenar la etiqueta.
- repeat-x: La imagen se repite en el eje horizontal hasta llenar la etiqueta, pero en el eje vertical no se muestra más de una vez, sin importar qué tan grande sea la etiqueta.
- repeat-y: La imagen se repite en el eje vertical hasta llenar la etiqueta, pero en el eje horizontal no se muestra más de una vez, sin importar qué tan grande sea la etiqueta.
- no-repeat: La imagen no se muestra más que una vez, sin importar el tamaño de la etiqueta.

Además, podemos controlar la posición de la imagen con respecto a la etiqueta usando la propiedad ‘background-position’.

```
html {
 background-image: url("tigre.jpg");
 background-repeat: no-repeat;
 background-position: 30px 10px;
}
```

La propiedad background-position lleva dos valores. El primero hace referencia al eje horizontal, y el segundo, al eje vertical. En este ejemplo, la imagen estará ubicada a 30px hacia la derecha y a 10px hacia abajo.

Los valores pueden ser unidades (como un número, seguido de px o em), un porcentaje (número seguido de %), y además palabras clave que son exclusivas de cada eje:

Eje horizontal:

- left
- center
- right

Eje vertical:

- top
- center
- bottom

Cabe destacar que podemos usar un solo valor en lugar de dos, pero el segundo valor es generado automáticamente por el navegador y puede entonces ser algo errático y es totalmente desaconsejado no incluir el segundo valor.

Todos estas propiedades, ‘background-image’, ‘background-repeat’, y ‘background-position’ pueden ser escritas directamente dentro de la propiedad que ya vimos antes, ‘background’.

Es lo mismo escribir entonces:

```
html {
 background-color: #946a55;
 background-image: url("tigre.jpg");
 background-repeat: no-repeat;
 background-position: 2em right;
}
```

Que:

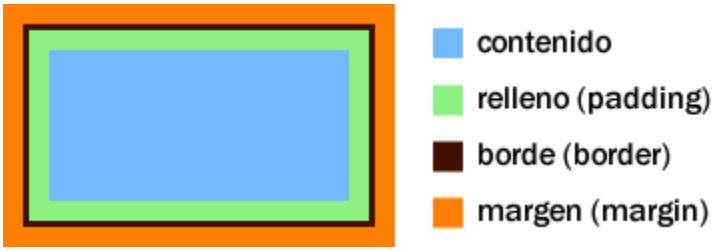
```
html {
 background: #946a55 url("tigre.jpg") no-repeat 2em right;
}
```

Si bien el orden de las propiedades escritas directamente dentro de ‘background’ no es estricto, es recomendable usar el orden: color, image, repeat, y position.

OJO: Para que una imagen de fondo se muestre, es necesario que la etiqueta tenga un ancho y un alto. Si colocamos una imagen de fondo a un div, pero este no tiene tamaño, ¡la imagen no se mostrará!

Bordes, márgenes y relleno

Una cosa a tener en cuenta con las propiedades border, margin y padding es su posición. Imaginando que todo es una caja, el contenido es lo que está más ‘adentro’, luego a su alrededor, está el relleno (padding), luego el borde (border) y el margen es el espacio fuera del borde.



Padding

El padding en CSS es un espacio vacío que se encuentra dentro de un elemento HTML, entre el contenido del elemento y sus bordes. Se utiliza para dar separación entre el contenido y los bordes del elemento. Al igual que el margen, el padding se puede especificar con distintas medidas, como pixels, porcentajes, o valores predefinidos como "auto". El padding se puede especificar para un solo lado o para varios lados de un elemento, por ejemplo:

```
padding-top: 10px;
padding-right: 20%;
padding-bottom: auto;
padding-left: 30px;
```

Border

El border en CSS es una línea que se dibuja alrededor de un elemento HTML. Se utiliza para enmarcar o destacar un elemento. El border se puede especificar con distintas opciones, como el ancho, el estilo (sólido, puntos, rayas, etc.), y el color. El border se puede especificar para un solo lado o para varios lados de un elemento, por ejemplo:

```
border-top: 2px solid blue;
border-right: 1px dotted red;
border-bottom: 3px dashed green;
border-left: 4px solid yellow;
```

Margin

El margen en CSS es un espacio vacío que se encuentra alrededor de un elemento HTML. Se utiliza para separar un elemento de otros elementos o de los bordes de la página. El margen se puede especificar con distintas medidas, como pixeles, porcentajes, o valores predefinidos como "auto". El margen se puede especificar para un solo lado o para varios lados de un elemento, por ejemplo:

margin-top: 10px;
margin-right: 20%;
margin-bottom: auto;
margin-left: 30px;

box-sizing

Box-sizing es una propiedad CSS que es usada para alterar cómo funciona el modelo de caja CSS estándar (content-box). Al asignarle un valor de 'border-box', el alto y ancho de las cajas incluyen el borde (border) y el relleno (padding). Esto nos permite fácilmente establecer tamaños (si definimos que una caja tendrá 100 pixeles de ancho y luego queremos agregarle un padding, no afectará nada fuera de la caja, por lo que no será necesario modificar el ancho).

Es recomendable usar esta regla CSS entonces:

```
*,
*:before,
*:after {
 -moz-box-sizing: border-box;
 -webkit-box-sizing: border-box;
 box-sizing: border-box;
}
```

Con esto, todos los elementos del documento serán modificados.

Si queremos volver al modelo de cajas estándar, debemos usar la siguiente regla:

```
box-sizing: content-box;
```

Selector CSS por atributo

Se lo usa de la forma [nombre_del_atributo] y hace objetivo a todos los elementos que tengan ese atributo en particular, por ejemplo:

```
[title] {
 color: red;
}
```

Hará objetivo a todos los elementos que tengan el atributo “title” y les asignará un color de texto rojo.

Además, se puede hacer un selector que busque no solamente los elementos que tengan un determinado atributo, sino que además tengan un valor en particular. Se lo usa de la forma [nombre_del_atributo=valor]. Por ejemplo:

```
a[href=http://www.google.com]
{
 font-style: italic;
}
```

Este ejemplo el selector hará objetivo a todos los <a> que tengan un “href” con un valor de “http://www.google.com” y les asignará un estilo italico. Cabe aclarar que en este ejemplo se busca un valor exacto, por lo que, por ejemplo, los valores de “http://google.com” o “http://www.google.com.ar” no están incluidos en la selección.

Además existe un selector que puede buscar elementos con un atributo y que tenga un valor que contenga una cadena de texto. Se lo usa de la forma: [nombre_del_atributo*=valor] y básicamente seleccionará todos los elementos que tengan el atributo y su valor contenga al texto ingresado. Por ejemplo:

```
a[href*="google"] {
 font-weight: bold;
}
```

Donde se seleccionarán todos los <a> que tengan el atributo “href” y que éste tenga en su valor la palabra google. Entonces, los enlaces “http://www.google.com” o “http://google.com” o “http://www.google.com.ar” estarán incluidos en la selección. Cabe destacar que en este caso también estarán incluidos enlaces como “http://www.facebook.com/google”.

Otro selector por atributos extremadamente útil es el que busca un atributo donde el valor del mismo comience con una cadena de caracteres determinada. Es decir, si el valor del atributo

no contiene entre sus primeras palabras el texto ingresado, no estará incluido en nuestra selección. Para usarlo se usa la forma: [nombre_del_atributo^=valor]. Por ejemplo:

```
a[href^="#"] {
 background-color: cyan;
}
```

En este ejemplo estamos seleccionando todos los <a> que tengan un “href” que comience en “#”. Es decir, solamente queremos seleccionar anclas internas de la página y no enlaces hacia otros documentos html o sitios externos. Justamente lo que necesitamos.

Posicionamiento

Otra cosa que necesitamos para que esto funcion bien, es mantener el <header> principal en la parte superior de la ventana, sin importar dónde esté el scroll. Esto lo podemos lograr usando propiedades CSS de posicionamiento. Las propiedades top, right, bottom y left manejan donde se ubica un elemento, mientras que la propiedad position establece si la posición es estática, relativa, absoluta, o fija.

- **position**: puede llevar el valor static, relative, absolute, o fixed.
    - **static**: La posición del elemento es normal. Las propiedades top, right, bottom, y left no aplican. Es el valor por defecto.
    - **relative**: La posición es calculada con respecto a la posición normal del elemento, es decir, un valor de top: 0 y left: 0 será igual a una posición estática.
    - **absolute**: La posición se calcula con respecto al contenedor. Si el contenedor no tiene posición establecida, se calcula con respecto al primer ancestro que tenga un posición establecida, si ningún ancestro tiene la posición establecida, se calcula con respecto a la pantalla.
    - **fixed**: similar a absolute, con la diferencia que la posición no se mueve con el scroll. Es decir, si colocamos un elemento con posición fixed, al hacer scroll en la ventana notamos que el elemento se queda en la misma posición siempre.
- **top**: puede llevar valores en medidas (em o px), o en porcentaje. Marca la posición vertical del elemento, con respecto a la parte de arriba (es decir, top: 10px colocará el elemento 10px más abajo de lo que estaría normalmente).
- **right**: puede llevar valores en medidas (em o px), o en porcentaje. Marca la posición horizontal del elemento, con respecto a la parte de la derecha (es decir, right: 10px colocará el elemento 10px más a la izquierda de lo que estaría normalmente).
- **bottom**: puede llevar valores en medidas (em o px), o en porcentaje. Marca la posición vertical del elemento, con respecto a la parte de abajo (es decir, bottom: 10px colocará el elemento 10px más arriba de lo que estaría normalmente).

- **left**: puede llevar valores en medidas (em o px), o en porcentaje. Marca la posición vertical del elemento, con respecto a la parte de la izquierda (es decir, top: 10px colocará el elemento 10px más a la derecha de lo que estaría normalmente).

Entonces, sabiendo todo esto, sabemos que nuestro footer tendrá que tener los siguientes valores:

```
body > header {
 position: fixed;
 top: 0;
 width: 100%; /* Necesita un ancho, puede ser cualquiera */
 height: 70px; /* Necesita un alto, puede ser cualquiera */
 background: #ff0000; /* Puede tener cualquier fondo */
}
```

Además de usar “position:fixed;” y “top: 0”, podemos notar que el selector no es “body header” o “header”, sino “body > header”, al usar el símbolo “>” para para separar palabras clave en el selector, estamos diciendo que el <header> al que estamos haciendo objetivo no es solamente un decendiente de <body>, sino que es uno directo (Es decir, si hay otro <header> hijo de un o de cualquier otro elemento, este bloque de reglas NO lo afectará).

Uso de Tipografías adicionales

Google Fonts

La forma más sencilla de utilizar tipografías además de las seguras es usando el servicio de Google, http://www.google.com/fonts/. Simplemente seleccionamos las tipografías que queremos en nuestra colección, hacemos click en ‘use’ o ‘usar’ y pasamos a elegir las variantes de nuestras tipografías seleccionadas, Google provee un medidor de peso que nos permite saber si las fuentes elegidas son muy pesadas o no.

Google Fonts generará un código como este, donde agregamos la fuente Merriweather Sans con pesos 300, 400 y 700:

```
<link
href='http://fonts.googleapis.com/css?family=Merriweather+Sans:300,400,7
00' rel='stylesheet'>
```

El cual colocamos en nuestro <head> como si de un CSS normal se tratase. Aunque cabe aclarar que este archivo es ideal agregarlo antes que otros CSS que usemos, para poder usar la tipografía sin problemas en los otros CSS. Y con esto básicamente tenemos acceso a la nueva tipografía, la cual podremos usar como si fuera cualquier otra:

```
font-family: 'Merriweather Sans', sans-serif;
```

Google Fonts también permite descargar las fuentes, lo cual nos permite trabajar en Fireworks con ellas. Cabe destacar que si instalamos nuevas fuentes y Fireworks está abierto, tendremos que cerrarlo y volverlo abrir para que detecte las nuevas tipografías.

Font Squirrel

Otro método útil es utilizar una tipografía diferente a las de Google Fonts y usar un conversor para que funcione en todos los navegadores. El conversor más popular es el de Font Squirrel http://www.fontsquirrel.com/tools/webfont-generator el cual genera muy buenos resultados (aunque falla con algunas tipografías comerciales, debido a infracción de Copyright).

Para usar el conversor de Font Squirrel básicamente tenemos que seleccionar nuestra tipografía local y subirla para que FS nos genere los archivos necesarios para funcionar web y aceptar los términos para poder descargar el 'kit'.

El 'kit' consiste en un archivo zip, dentro del cual hay varios archivos, pero lo más importante es el html, el css y diversos archivos con el nombre de nuestra tipografía, pero con otras extensiones, estas son las fuentes en los formatos que soportan los distintos navegadores.

Por ejemplo, si usamos una de las tipografías que podemos encontrar en el mismo repositorio de Font Squirrel, 'Aaargh', la cual es una serif. Luego de descargarla, colocamos los archivos de fuente en nuestra carpeta (.eot, .woff, .ttf y .svg). Luego en las primeras líneas de nuestro CSS la embebemos de la siguiente manera:

```
@font-face {
 font-family: "Argh";
 src: url("aaargh-webfont.eot");
 src: url("aaargh-webfont.eot?#iefix") format("embedded-opentype"),
   url("aaargh-webfont.woff") format("woff"),
   url("aaargh-webfont.ttf") format("truetype"),
   url("aaargh-webfont.svg#webfont") format("svg");
}
```

La propiedad 'font-family' aquí será para asignarle un nombre a la tipografía, el cual se usará en el resto del documento CSS. También podríamos usar las propiedades 'font-weight' o 'font-style' para establecer alternativas a una misma fuente (Es decir, el mismo 'font-family' puede aparecer en varios bloques @font-face, y cada uno de estos tendría un font-weight o font-style diferente.

Flex box

Flexbox en CSS es un sistema de diseño que se utiliza para organizar y distribuir elementos de forma flexible dentro de un contenedor. Los elementos dentro del contenedor se llaman "hijos" y pueden tener distintos tamaños y formas. Flexbox permite alinear y distribuir los hijos de forma flexible y adaptativa, y permite también controlar el flujo del contenido y la disposición de los elementos en distintos tamaños de pantalla y dispositivos.

Para utilizar flexbox, se debe especificar el comportamiento de flexbox en el elemento contenedor (padre) mediante la propiedad **"display: flex"** y luego se pueden utilizar otras propiedades para controlar la disposición y el comportamiento de los elementos hijos. Por ejemplo:

- **justify-content**: para alinear los elementos hijos horizontalmente
- **align-items**: para alinear los elementos hijos verticalmente
- **flex-wrap**: para permitir que los elementos hijos envuelvan al no caber en la línea

Flexbox es una herramienta muy útil para el diseño web responsivo y para crear layouts adaptativos y flexibles.

Guía práctica de Flex-box: **https://css-tricks.com/snippets/css/a-guide-to-flexbox/**

Grid

Grid en CSS es un sistema de diseño que se utiliza para organizar y distribuir elementos de forma tabular dentro de un contenedor. Los elementos dentro del contenedor se llaman "celdas" y pueden tener distintos tamaños y formas. Grid permite crear estructuras de diseño complejas y adaptativas mediante el uso de filas y columnas definidas por líneas guía.

Para utilizar grid, se debe especificar el comportamiento de grid en el elemento contenedor (padre) mediante la propiedad "display: grid" y luego se pueden utilizar otras propiedades para definir las filas y columnas y controlar la disposición y el comportamiento de las celdas. Por ejemplo:

- grid-template-columns: para definir el tamaño y la distribución de las columnas
- grid-template-rows: para definir el tamaño y la distribución de las filas
- grid-gap: para establecer un espacio entre las celdas
- justify-items: para alinear las celdas horizontalmente
- align-items: para alinear las celdas verticalmente

Grid es una herramienta muy útil para el diseño web responsivo y para crear layouts adaptativos y estructurados.

Guía práctica de gird **https://css-tricks.com/snippets/css/complete-guide-grid/**