

CSS

Hay varias maneras de aplicar CSS en nuestra pagina, la primera es directamente en nuestro documento HTML:

Definiendo primero la etiqueta `<style></style>` y declarando nuestros estilos dentro

En cada etiqueta en particular, donde aplicaremos el estilo que querramos.

En un archivo distinto, que es lo mas recomendable para que nuestro codigo sea prolijo y legible, asi, en las etiquetas de nuestra pagina, solo aclaramos las clases o id's que vayamos a utilizar.

Existen mas de 200 propiedades o atributos, no son tantas, asi que vamos a intentar pasar por la mayoría de ellas para generar un conocimiento amplio sobre las mismas, para poder hacer cualquier tipo de modificacion que querramos en nuestras paginas.

Vamos a enfocarnos solo a trabajar con una hoja de estilos en un archivo css, asi que, antes que nada, tenemos que declarar en nuestro documento HTML de donde tiene que importar los estilos, esto lo hacemos de la siguiente manera:

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

donde href es la direccion donde se encuentra nuestro archivo de estilos.

Rel es la relacion del archivo que estamos linkeando

y type es el tipo de archivo linkeado.

Estructura CSS

Aca vemos como es la estructura de una regla predeterminada



Donde "P" es el selector, un elemento dentro de nuestro **HTML** en el que se aplica nuestra regla, en el ejemplo de la imagen, es la etiqueta `<p>`, cualquier declaracion dentro de las llaves que conforman esta regla, tendrá efecto en todas nuestras etiquetas `<p>` o sea, en todos nuestros parrafos.

En la declaracion, nosotros vamos definir una propiedad, y un valor de propiedad, con esto vamos a poder hacer las modificaciones que querramos, es como si estuviésemos cambiando el valor de una variable, que lo que hace es impactar en como la propiedad afecta a nuestra etiqueta.

Tipos de selectores

Existen muchos tipos distintos de selectores, existen los **elementos**, que serían nuestras **etiquetas HTML**, tales como `<p>`, `<body>`, ``, `<footer>`, etc.

Los **selectores de identificación** o ID's, estos tienen la peculiaridad que solo se puede tener uno por elemento, o sea, podemos tener varios elementos con un mismo ID, pero no podemos poner varios IDs en un mismo elemento, esto creo que viene de la mano con la sobreescritura, ya que por ejemplo, el ID tiene prioridad sobre los otros selectores que usamos en CSS

```
<div id="idUnico"></div>
```

Cuando los definimos en nuestra plantilla de estilos, debemos hacerlo de esta manera:

```
#idUnico {  
    propiedad : valorDePropiedad;  
}
```

Selectores de clase

Los **selectores de clase**, estos son como los IDs, pero sin restricciones, podemos declarar tantos como queramos dentro de nuestros elementos de la siguiente manera:

```
<div class="miClase">Contenido</div>
```

Y para definirlos lo hacemos de la siguiente manera:

```
.miClase {  
    propiedad : valorDePropiedad;  
}
```

Selectores de atributo

Selectores de atributo, estos son los que modifican un atributo en particular, por ejemplo cuando queremos editar una imagen en particular, pero no así todas las imágenes de nuestra página

img[src] selecciona solo esa imagen en particular, no así la etiqueta `img`.

```
a[href="https://example.com"] {  
    propiedad: valorDePropiedad;  
}
```

Selectores de pseudoclase

Los selectores de pseudoclase nos sirven para modificar un elemento, mientras se encuentre en un determinado estado, por ejemplo, tenemos el caso de **:hover**, que nos permite modificar un elemento en particular, cuando el puntero del mouse se encuentre sobre el elemento en cuestion.

Este se define de la siguiente manera:

```
a:hover {  
    propiedad : valorDePropiedad;  
}
```

Tenemos mas selectores de pseudoclase, los siguientes aplican a los links, las etiquetas anchor o ancla:

- **a:link** - link normal
- **a:visited** - link que el usuario ya fue visitado
- **a:active** - link en el momento que es clickeado

Estos se definen y manipulan igual que el a:hover presentado anteriormente.

| | | |
|--------------------------------------|-----------------------|---|
| :active | a:active | Selecciona un link activo |
| :checked | input:checked | Selecciona los <input> en estado checked |
| :disabled | input:disabled | Selecciona los <input> desabilitados |
| :empty | p:empty | Selecciona los <p> que no tienen hijos |
| :enabled | input:enabled | Selecciona los <input> habilitados |
| :first-child | p:first-child | Selecciona los <p> que son los primeros hijos de su padre |
| :first-of-type | p:first-of-type | Selecciona los <p> que son los primeros hijos de su padre |
| :focus | input:focus | Selecciona el <input> en el que estamos enfocados |
| :hover | a:hover | Selecciona el elemento donde esta posicionado el mouse |
| :in-range | input:in-range | Selecciona los <input> con un valor que tiene un determinado rango |
| :invalid | input:invalid | Selecciona los <input> con un valor invalido |
| :lang(language) | p:lang(it) | Selecciona los <p> con un atributo de idioma que comienza con "it" |
| :last-child | p:last-child | Selecciona el <p> que es el ultimo hijo de su padre |
| :last-of-type | p:last-of-type | Selecciona los <p> que son los ultimos hijos de sus padres |
| :link | a:link | Selecciona todos los links no visitados anteriormente |
| :not(selector) | :not(p) | Selecciona todo lo que no sea un <p> |
| :nth-child(n) | p:nth-child(2) | Selecciona el <p> que sea el 2do hijo de su padre |
| :nth-last-child(n) | p:nth-last-child(2) | Selecciona el <p> selecciona el 2do hijo contando de atras hacia adelante |
| :nth-last-of-type(n) | p:nth-last-of-type(2) | Selecciona todos los <p> que sean los 2dos hijos de su padre contando de atras hacia adelante |
| :nth-of-type(n) | p:nth-of-type(2) | Selecciona los <p> que sean los segundos <p> de sus padres |
| :only-of-type | p:only-of-type | Selecciona los <p> que sean los unicos <p> hijos de su |

| | | padre |
|-------------------------------|--------------------|---|
| :only-child | p:only-child | Selecciona los <p> que son los unicos hijos de su padre |
| :optional | input:optional | Selecciona los <input> que no tienen el atributo "required" |
| :out-of-range | input:out-of-range | Selecciona los <input> que tengan un valor fuera del valor especificado |
| :read-only | input:read-only | Selecciona los <input> con el atributo "readonly" especificado |
| :read-write | input:read-write | Selecciona los <input> sin el atributo "readonly" |
| :required | input:required | Selecciona los <input> que tienen un atributo "required" |
| :root | root | Selecciona el elemento root del documento |
| :target | #news:target | Selecciona el elemento #news activo (clickeado en una URL que tenga ese nombre de "anchor" o "<a>") |
| :valid | input:valid | Selecciona los <input> con un valor valido |
| :visited | a:visited | Selecciona todos los links visitados anteriormente |

Pseudo elementos

También existen **pseudoelementos**, que seleccionan una parte determinada de un elemento, en vez del elemento en sí, por ejemplo **::first-line** siempre selecciona la primer línea de texto que se encuentra dentro de un elemento <p> haciendo actuar a la primer línea como si fuese un span individual, aplicándole el estilo al mismo.

```
p::first-line {
    propiedad: valorDePropiedad;
}
```

Existe también un **selector universal**, que aplica las propiedades que definamos dentro a todos los elementos dentro de nuestro html de la siguiente manera,

```
* {
    propiedad: valorDePropiedad;
}
```

Podemos también seleccionar solo un elemento que contenga una clase determinada de la siguiente manera:

```
elemento.clase {
    propiedad: valorDePropiedad
}
```

Para tener un código más prolijo, cuando querremos aplicar las mismas propiedades a varios elementos distintos, podemos hacer esto:

```
h1, h2, p {
    propiedad: valorDePropiedad;
    propiedad2: valorDePropiedad2;
    ...
}
```

| Selector | Ejemplo | Descripcion |
|---------------------------------------|-----------------|--|
| <u>::after</u> | p::after | Inserta algo despues del contenido de cada elemento<p> |
| <u>::before</u> | p::before | Inserta algo antes del contenido de cada elemento<p> |
| <u>::first-letter</u> | p::first-letter | Selecciona la primer letra de los <p> |
| <u>::first-line</u> | p::first-line | Selecciona la primer linea de los <p> |
| <u>::marker</u> | ::marker | Selecciona los marcadores de las listas |
| <u>::selection</u> | p::selection | Selecciona una parte de un elemeto que sea seleccionada por el usuario |

Combinaciones de selectores

Un selector puede contener mas de uno valor, entre selectores simples, podemos añadir combinadores, existen cuatro maneras distintas de hacer esto en CSS:

Selector de descendientes (un espacio en blanco)

El selector de descendiente aplica las propiedades a todos los elementos descendientes de un selector especificado, de la siguiente manera

```
div p {
    propiedad: valorDePropiedad;
}
```

En el ejemplo anterior, la propiedad se aplicará solo a los elementos p, que se encuentren dentro del div especificado.

Selector de hijo (>)

El selector de hijo aplicara las propiedades a todos los elementos hijos de un selector en especifico:

```
div > p {
    propiedad: valorDePropiedad;
}
```

Esto aplicará la propiedad a todos los elementos <p> dentro de nuestro div, pero, si este div tiene un div dentro, o cualquier otro contenedor, y ese a su vez, tiene elementos p, estos no sufriran cambios.

Selector de hermanos adyacentes (+)

El selector de hermano adyacente se usa para seleccionar un elemento que se encuentra directamente después del primer elemento especificado

```
div + p {
    propiedad: valorDePropiedad;
}
```

Esto aplicará la propiedad al primer elemento p que le siga a cualquier elemento div en nuestro codigo

Selector de hermanos generales (~)

El selector de hermanos generales, se usa para seleccionar a todos los elementos que se encuentran después del primer elemento especificado

```
div ~ p {  
    propiedad: valorDePropiedad;  
}
```

Esto aplicara la propiedad a todos los elementos p que esten después de nuestro elemento div, aun así tengan otras etiquetas de por medio.

Cursores

La propiedad "cursor" nos permite cambiar la apariencia de nuestro cursor cuando se posicione sobre el elemento al que se la apliquemos, suele ser muy útil para los links, pero puede usarse para mas cosas.

cursor: auto; => define automaticamente el tipo de cursor
cursor: crosshair; => cambia el cursor por una mira
cursor: default; => define el cursor como el puntero predeterminado
cursor: help; => agrega un signo de pregunta al cursos
cursor: move; => cambia el cursor a las 4 flechas unidas que se suelen utilizar cuando moves cosas en paint por ejemplo.

cursor: pointer; => cambia el cursor por una manito.
cursor: progress; => agrega un circulo de progreso a la derecha del cursor.
cursor: text; => cambia el cursor al que vemos cuando editamos texto
cursor: wait; => cambia el cursor a un circulo de progreso
cursor: e-resize; => cambia el cursor por un puntero de redimension horizontal
cursor: n-resize; => cambia el cursor por un puntero de redimension vertical
cursor: ne-resize; => cambia el cursor por un puntero de redimension diagonal
cursor: nw-resize; => cambia el cursor por un puntero de redimension diagonal invertido

Fuentes y texto

Antes que nada hay que aclarar que la cada vez que definamos una fuente en nuestro CSS, va a buscarla en nuestro disco para confirmar su existencia, si no existe va a reemplazarla automaticamente por el valor predeterimado que tiene el explorador de internet que estemos usando, para evitar esto, ya que una fuente distinta a la que nosotros elegimos para nuestra pagina, es recomendable proporcionar a cada usuario que ingrese en nuestra pagina, las fuentes que utilizaremos en la misma, para esto, definimos el origen y la fuente en cuestion de la siguiente manera:

```
<link href="https://fonts.googleapis.com/css2?family=Open+Sans"  
rel="stylesheet" type="text/css">
```

De igual manera tenemos una base de fuentes "seguras" para HTML y CSS que son las

siguientes:

- Arial (sans-serif)
- Verdana (sans-serif)
- Helvetica (sans-serif)
- Tahoma (sans-serif)
- Trebuchet MS (sans-serif)
- Times New Roman (serif)
- Georgia (serif)
- Garamond (serif)
- Courier New (monospace)
- Brush Script MT (cursive)

Supuestamente, estas fuentes se encuentran en todos los dispositivos y exploradores de internet, sin embargo, tenemos que definir siempre fuentes de respaldo.

Selección de fuentes

La selección de fuente es una de las cosas más importantes que tenemos que hacer cuando estamos diseñando nuestra página, una vez que hayamos definido que tipo de fuente vamos a querer en cada sección de nuestra página, podemos definirla de la siguiente manera:

`font-family:` fuente, primer fuente de respaldo, segunda fuente de respaldo;

Un punto importante a aclarar, es que cuando definimos una fuente cuyo nombre tiene más de una palabra, debemos definirla entre comillas, por ejemplo, si queremos utilizar la fuente times new roman debemos hacerlo de la siguiente manera:

`font-family:` "times new roman", arial, helvetica;

Podemos darle también, un estilo a la fuente que hayamos elegido con la propiedad `Style` y sus variantes que son:

font-style: normal => El texto se va a mostrar de manera "normal".

font-style: italic => El texto se va a mostrar en *cursiva*.

font-style: oblique => Similar a la italic, pero tiene menos soporte que la anterior, así que es recomendable usar la italic si queremos este efecto.

Font variant

La propiedad `font-variant` nos permite mostrar todo un texto en mayúsculas, convirtiendo las minúsculas en mayúsculas, pero dándoles un tamaño más pequeño que una mayúscula normal.

font-variant: small-caps;

Tamaño de fuentes

La propiedad font-size nos permite cambiar el tamaño del texto, tanto de manera absoluta como de manera relativa, la manera absoluta nos permite tener un mejor control a la hora del diseño, debemos usar este tipo cuando tenemos un viewport fijo.

Si el tamaño de fuente no es definido por la propiedad size, normalmente, se utilizara 16px=1em, esto depende del explorador en el cual estemos viendo la pagina en cuestion.

Definiendo el tamaño con valores predeterminados

```
font-size: xx-small || x-small || small || medium || large || x-large || xx-large
```

Definiendo el tamaño con pixeles

Definir el tamaño de nuestra fuente con pixeles, nos da un control absoluto sobre el tamaño de nuestro texto, y en caso de que se use zoom para cambiar el tamaño de la pagina, el texto tambien cambiara de tamaño, pero, si el usuario cambia el tamaño de fuente predeterminada en su navegador, no impactará en el tamaño de la fuente de nuestra pagina.

```
font-size: 10px;
```

Definiendo el tamaño con Em

Em es una medida relativa, que depende de los pixeles predeterminados en el navegador que se este utilizando, 1em = los pixeles que hayamos predefinido, o en caso que no se haya tocado esa configuración, 1em = 16px en la mayoría de los casos.

Este tipo de medida, suele traer problemas con algunas versiones de Internet Explorer, pero a nadie debería interesarle

```
font-size: 1em;
```

Definiendo el tamaño con em y %

```
body {  
    font-size: 100%;  
}  
h1 {  
    font-size: 2.2em;  
}
```

Definiendo el tamaño con vm

Podemos definir tambien el tamaño del texto con el ancho del viewport o "vw", de esta manera tendremos una fuente full responsiva, donde su tamaño va a depender del tamaño de la ventana del navegador.

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">  
p {  
    font-size: 10vm;}
```


Importando fuentes de Google fonts

Google fonts tiene mas de 1000 fuentes disponibles de uso gratuito, en caso que las fuentes predeterminadas no nos gusten o no se adapten a lo que necesitamos, podemos elegir alguna de estas, para importarlas lo hacemos de la siguiente manera, primero tenemos que definir la importacion en el <head> de nuestra pagina de la siguiente manera:

```
<head>
<link rel="stylesheet" href="https://fonts.googleapis.com/css?
family=Sofia">
</head>
```

Luego para utilizarla, la llamamos de la misma manera que usamos antes.

Juntando varias propiedades en una sola declaracion

Para juntar varias propiedades, tenemos que tener en cuenta el siguiente orden:

1. font-size
2. font-family

Ejemplo:

font: 20px Arial, sans-serif;

Sino:

1. font-style
2. font-variant
3. font-weight
4. font-size/line-height
5. font-family

font: italic small-caps bold 12px/30px Georgia, serif;

Alineaciones de texto

Text-align nos sirve para alinear el texto, para esta propiedad, tenemos varios valores posibles:

1. **text-align: center** => centra el texto en la mitad del viewport disponible
2. **text-align: left** => lleva el texto a la parte izquierda del viewport disponible
3. **text-align: right** => lleva el texto a la derecha del viewport
4. **text-align: justify** => acomoda el texto para que todas las lineas del mismo ocupen el mismo espacio, manteniendo los margenes izquierdo y derecho, tal como podemos verlo en las paginas de noticias.
5. **Text-align-last: center, left, right, justify** => esta propiedad nos permite modificar la ultima linea de nuestro texto
6. **direction & unicode-bidi**, estas propiedades nos permiten invertir la direccion de un texto de la siguiente manera : **p** {

```
direction: rtl;
unicode-bidi: bidi-override;
}
```

Tenemos tambien una propiedad para poder alinear verticalmente un elemento al texto que tenemos definido, esto nos permite ubicar esta imagen en cuestion en el espacio que querramos dentro de la linea de texto de las siguientes maneras

vertical-align: baseline => nos posicionara nuestra imagen, alineando el centro de la misma, con el centro de la linea de texto,


vertical-align: text-top || **sup** => posicionara nuestra imagen alineado el centro de la misma, con la linea superior de la linea de texto

vertical-align: text-bottom || **sub** => posicionara nuestra imagen alineando el centro de la misma con la linea inferior de la linea de texto.


Tambien podemos especificar nuestro vertical-align, en unidades de medida, para poder modificar la alineacion vertical a nuestro antojo.

The vertical-align Property


vertical-align: baseline (default):

An  image with a default alignment.


vertical-align: text-top:

An  image with a text-top alignment.


vertical-align: text-bottom:

An  image with a text-bottom alignment.

vertical-align: sub:

An  image with a sub alignment.

vertical-align: sup:

An  image with a super alignment.

Decoraciones de texto

Tambien tenemos distintas propiedades para darle decoracion a nuestro texto definidas por la propiedad `text-decoration`, la cual tiene los siguientes valores que podremos utilizar:

`text-decoration-line: overline` => esta nos permite decorar nuestro texto con una linea que pasa por todo el margen superior del mismo.

`text-decoration-line: line-through`; => esta nos permite decorar nuestro texto con una linea que pasa por el medio de nuestro texto, dandonos un efecto de tachado.

`text-decoration-line: underline`; => decora nuestro texto con una linea bajo el mismo, no se suele utilizar, porque genera el efecto de tener un anchor o link cuando no es así, y puede generar confusión en algunos usuarios

`text-decoration-line: overline underline` => este valor nos da una linea superior y una inferior a nuestro texto.

Tambien podremos modificar el estilo de nuestro `text-decoration` con algunas propiedades extra, que nos permiten hacer lo siguiente:

`text-decoration-color: color`; => esto nos permite cambiar el color de nuestra decoracion.

`text-decoration-style: solid || double || dotted || dashed || wavy`; => esto nos permite cambiar el estilo de nuestra decoracion.

`text-decoration-thickness: auto || 1px || 5%`; => esto nos permite cambiar el grosor de nuestra linea de decoracion.

Y por ultimo tenemos `text-decoration`, esto lo usamos porque todos los links que pongamos en nuestra pagina html van a tener la decoracion `underline` por defecto, en caso que no querramos que esto suceda, lo que haremos es declarar en nuestros anchors, la propiedad `text-decoration` con el valor `none`, de la siguiente manera

```
a {  
    text-decoration: none;  
}
```

`text-shadow: 2px 2px 2px red`; => con esta propiedad podemos darle un efecto de sombreado a nuestro texto, el primer argumento es el sombreado horizontal, el segundo el vertical, el tercero la profundidad y el cuarto el color del efecto.

Transformacion de texto

Tenemos propiedades que nos permiten tambien cambiar la capitalizacion de nuestros textos:

text-transform: uppercase; => pone todo nuestro texto en mayusculas.

text-transform: lowercase; => pone todo nuestro texto en minusculas.

text-transform: capitalize; => pone en mayusculas la primer letra de cada palabra.

Espaciado

text-indent, que define cuanto queremos de espacio de tabulacion (indentado) al comienzo de la primera linea de nuestro texto

text-indent: longitud/porcentaje

text-indent: 50px;

text-indent: 10%;

letter-spacing: 2px/-2px => lo usamos para definir el espacio entre caracteres, y se define en pixeles, podemos tanto incrementar como disminuir el espacio entre los mismos.

line-height: 0.8/1.8 => esta propiedad nos sirve para especificar el tamaño entre lineas, donde 1 sería el standard podemos incrementar o disminuir el espacio entre las lineas, para hacer que el texto ocupe menos espacio.

word-spacing: 10px/-2px => esta propiedad nos permite modificar el espaciado entre las palabras de nuestro texto.

White space

la propiedad white space nos permite especificar como tratar el espaciado en nuestro codigo, y como este afecta a nuestro texto mostrado en pantalla.

white-space: normal; => todos los espacios son reducidos a uno, los saltos de linea en el codigo son tratados como un espacio en blanco, requiere que agreguemos saltos de linea para rellenar el contenedor

white-space: pre-wrap; => Los espaciados del codigo son preservados, genera un nuevo parrafo con su respectiva tabulacion por cada "." punto en nuestro codigo

white-space: pre-line; => Los espaciados del codigo son suprimidos, genera un nuevo parrafo por cada punto en nuestro codigo

white-space: nowrap; => Reduce a uno los espacios de nuestro codigo, pero a diferencia del normal, suprime los saltos de linea, si utilizamos esta opcion, tendremos que utilizar si o si saltos de linea
 para darle forma al texto.

white-space: break-spaces; => funciona practicamente igual al pre-wrap, es mas, no encuentre diferencia entre ambas.

Colores

Tenemos varias maneras de definir los colores que queremos utilizar en nuestras paginas, podemos usar directamente los nombres, o con RGB, HEX, HSL, RGBA y HSLA

Cuando definimos un color con rgb(rojo, verde, azul), cada parametro define la intensidad del color en una escala del 0 al 255, si queremos un rojo puro, lo definimos como rgb(255, 0 , 0)

Para definir el negro, lo hacemos seteando todos los valores de la siguiente manera rgb(0,0,0)

Para definir el blanco, lo hacemos maxeando todos los valores rgb(255,255,255)

Tenemos una variante del rgb, el RGBA que nos permite manejar la transparencia del color en cuestion, definiendo un numero entre el 0 y el 1, donde 0 es totalmente transparente y 1 no posee transparencias rgba(255,255,255,0.5)

Podemos tambien definir nuestros colores con un valor hexadecimal, es basicamente lo mismo que con rgb ya que si nos fijamos, los colores en exa se definen de la siguiente manera #ROJOROJO VERDEVERDE AZULAZUL

Los dos digitos de cada valor nos dan un rango entre 00 y FF, esto sería lo mismo que entre 0 y 255

por ejemplo, si quisieramos definir el rojo como lo hicimos antes, con valor exadecimales sería de la siguiente manera #FF0000 donde FF es el valor maximo de la tonalidad roja

negro #000000

blanco #FFFFFF

HLS nos permite definir colores, usando un tono, el nivel de saturacion y luminosidad de la siguiente manera

hsl(tono, saturacion%, luminosidad%)

el tono es un grado en la rueda de colores de 360°, donde 0 es rojo, 120 es verde y 240 es azul

la saturacion es un valor porcentual, donde 0% es gris y 100% es el color que definimos antes

la luminosidad tambien se mide en porcentajes, donde 0% es negro, 50% sería nuestro color y 100% es blanco, esto nos permite jugar un poco mas con los colores, para poder tener una paleta de contrastes mas definida y mucho mas facil de controlar, por ejemplo, si queremos que nuestra pagina solo tenga distintas tonalidades del color verde, simplemente podriamos jugar con los valores de saturacion y luminosidad bajo el mismo parametro de color

Fondos

Tenemos varias propiedades que nos permiten modificar nuestros fondos

background-color: blue; => nos permite cambiar el fondo del elemento en cuestion

background-image: url("www.urldenustraimagen.com/nuestraImagen"); => nos permite poner una imagen de fondo que por defecto se repetira hasta completar todo el elemento horizontal y verticalmente, en caso que no querramos que asi sea, podemos modificar este comportamiento utilizando la propiedad

background-repeat: repeat-x || repeat-y => esto va a hacer que la imagen se repita vertical u horizontalmente dependiendo del valor que hayamos elegido tambien podemos controlar el espacio entre las repeticiones de nuestra imagen con las propiedades

background-repeat: space; => da un espacio mas amplio entre cada una de las repeticiones hasta

completar el viewport

background-repeat: round; => aumenta el tamaño de la imagen de cada repetición para completar el viewport

también podemos elegir no repetir la imagen y mostrarla solo una vez con la siguiente propiedad

background-repeat: no-repeat; => de esta manera la imagen se va a mostrar una única vez

En caso que hayamos decidido no repetir nuestra imagen, podemos elegir donde posicionar la misma, con la propiedad **background-position**, a la que podemos asignarle los valores: left top, left center, left bottom, right top, right center, right bottom, center top, center center, center bottom, si solo especificamos uno de estos valores, el siguiente automáticamente se seteará en "center" también podemos definir la posición como un % del tamaño de nuestra página, o del contenedor donde se encuentre con x% y%, también podemos controlarlo con xpos ypos definiendo los píxeles horizontales y verticales de nuestra página

la propiedad **background-attachment** nos permite especificar como queremos que se comporte nuestra imagen de fondo elegida al scrollear en nuestra página

background-attachment: scroll; => hace que nuestra imagen scrollee con la página (es la opción predeterminada)

background-attachment: fixed; => la imagen no hará scroll con la página

background-attachment: local; => la imagen hará scroll con el elemento en el que se encuentra

background-attachment: inherit; => hereda la propiedad de su elemento padre

Podemos reducir un poco el código cuando trabajamos en fondos con el siguiente orden

background-color

background-image

background-repeat

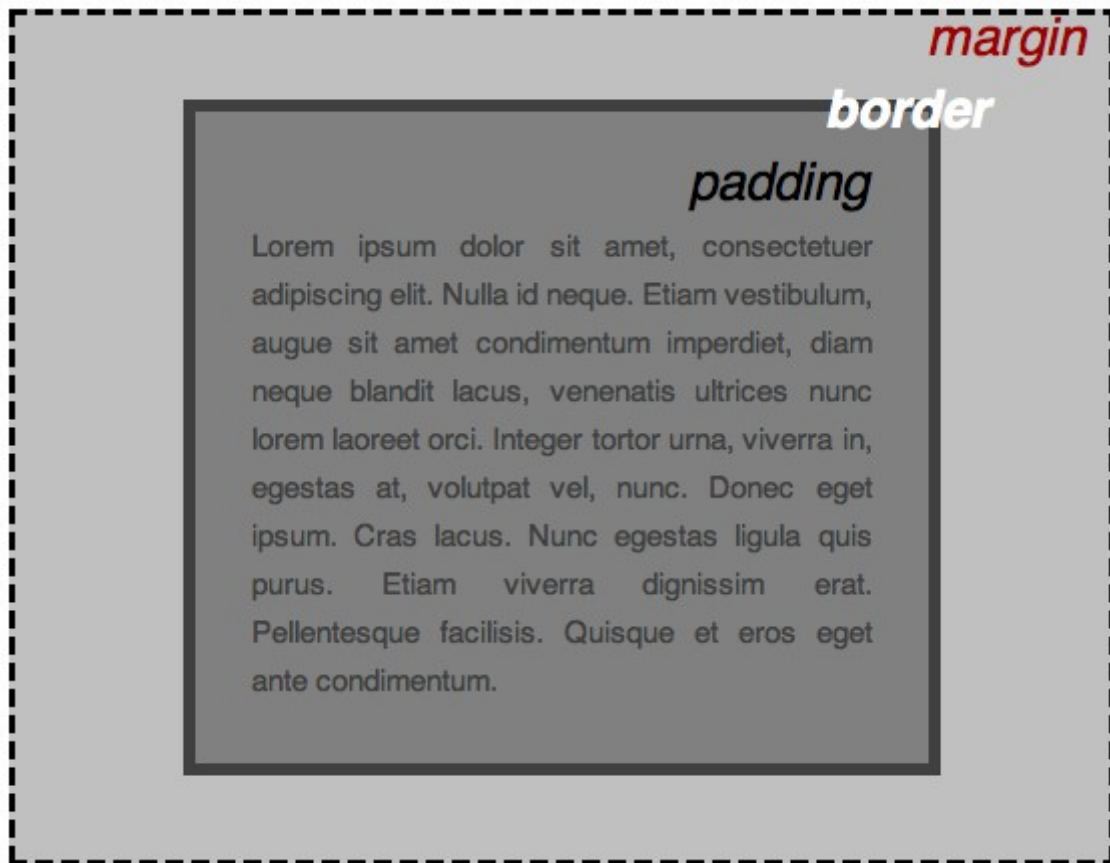
background-attachment

background-position

definiendo de la siguiente manera podemos ahorrar bastante código

background: color image repeat attachment position; => donde obviamente tenemos que reemplazar los argumentos que lo que queramos utilizar de fondo

PADDING BORDER MARGIN



Estas 3 opciones nos ayudan a controlar los espacios entre nuestro elemento en cuestion y los demás elementos que conforman la página, el padding, es el espacio interno entre el borde y nuestro elemento, el margin es espacio que separa nuestro elemento, de otros elementos, y el border, es lo que separa al padding del margin

A estas opciones, podemos darle estilos, colores y distintos tamaños para personalizar nuestros elementos, para así darle una personalidad unica a nuestras paginas.

El estilo de nuestros bordes, los definimos con border-style, que tiene los siguientes valores:

border-style: dotted; => borde con linea punteada
border-style: dashed; => borde con lineas cortadas
border-style: solid => borde con linea solida
border-style: double => borde de linea solida doble
border-style: groove => borde con efecto 3D
border-style: ridge => borde con efecto 3D
border-style: inset => borde con efecto 3D interno
border-style: outset => borde con efecto 3D externo
border-style: none => No define borde
border-style: hidden => Define el borde como oculto

Tambien podemos poner varios estilos en el mismo borde, aclarando en su style que tipo de borde queremos en cada lado ej:

border-top-style: dotted;
border-right-style: dashed;

border-bottom-style: solid;
border-left-style: double;

Y tambien podemos simplificarlo de la siguiente manera teniendo en cuenta el orden de agujas de reloj:

border-style: dotted dashed solid double;

A parte de poder modificar los estilos, tambien podemos cambiar el ancho de los mismos con la propiedad width, que tiene 3 valores predefinidos, thin, medium y thick, y a parte podemos definir este valor con varias unidades de tamaño, como pulgadas "in" pixeles "px" "pt" "cm" "em", etc

Podemos tambien especificar un valor para cada borde, si aclaramos en nuestra propiedad width los 4 valores, asignara uno a cada uno de los lados, en el sentido de la agujas del reloj superior, derecha, inferior, izquierda.

border-width: 10px 15px 10px 5px;

tambien podemos solo especificar dos, con lo que haremos que el primer valor dado funcione tanto como para la parte inferior como la superior y el segundo valor para las partes izquierda y derecha de nuestro borde

border-width: 10px 25px;

Podemos tambien con la propiedad color, cambiar el color del borde de una manera similar a la del width, podemos poner solo un color para que todo el borde se pinte del mismo color, o podemos tambien aclarar el color de cada uno de los lados del borde en el sentido de las agujas del reloj, y si solo aclaramos dos, tomara el primer valor para los lados sup e inf y el segundo valor para los lados izq y der.

border-color: red black yellow blue;
border-color: red yellow;

Si queremos un borde simple, sin demasiados lujos, tambien podemos definirlo de manera abreviada de la siguiente manera:

border: tamaño tipo color;
border: 5px solid blue;

Podemos tambien, crear bordes redondeados con la propiedad radius, podemos definirla en pixeles o porcentaje de la siguiente manera :

border-radius: 5px//50%;