

Reglas HTML

Tags HTML

1. Tags HTML en minúsculas

Antiguamente, las etiquetas estaban escritas de ambas maneras. Por una cuestión de estructura visual de código.

```
<html lang="es-ar">
<head>
  <meta charset="utf-8">
  <meta description="RW">
  <title>HTML5 Lindo</title>
</head>
<body>
  <header>
    <figure id="logo">
      
    </figure>
    <h1>Taller de Res<
    <figure id="avatar">
        
  
eta charset="utf-8" />  
eta description="RWD" />  
title>HTML5 Lindo</title>  
>  
  
header>  
  <figure id="logo">  
      
  </figure>  
  <h1>Taller de Responsive Web Design  
  <figure id="avatar">  
      
  </figure>  
header>  
av>  
</ul>
```

Atributos y valores

3. Todos los atributos van en minúsculas y sus valores entre comillas

En HTML5 es opcional que los valores vayan entre comillas pero es importante que sí estén entre comillas.

```
TYPE html>
lang="es-ar">

eta charset="utf-8" />
eta description="RWD" />
title>HTML5 Lindo</title>
>

header>
  <figure id="logo">
    
  </figure>
  <h1>Taller de Responsive Web Design
  <figure id="avatar">
    
  </figure>
header>
av>
</ul>
```


Indentación

4. Correcta indentación del código

Siempre respetar la columna en donde comienza la indentación. Cerrarla a la misma altura.

```
01.html
1 <!DOCTYPE html>
2 <html lang="es-ar">
3 <head>
4     <meta charset="utf-8" />
5     <meta description="RWD" />
6     <title>HTML5 Lindo</title>
7 </head>
8 <body>
9     <header>
10         <figure id="logo">
11             
12         </figure>
13         <h1>Taller de Responsive We
14         <figure id="avatar">
15             
17     </header>
18     <nav>
19         ...
```

File Transfer Protocol

FTP

En el pasado, el FTP fue una de las mejores cosas que pudieron haber ocurrido.

En la actualidad, el uso de FTP en la web es una de las peores prácticas para la transferencia de archivos.

El protocolo es muy sencillo el cual conecta a la carpeta para archivos html del servidor de hosting, para que podamos subir nuestros archivos y poder acceder a ellos online posteriormente.

Desventajas

- Acceso único simultáneo permitido
- Sin registro de acceso
- No se actualiza en red



Control de versiones

GIT

Github

Toma las capacidades de GIT y las transforma en algo muy simple de utilizar.

Se transformó en el "Facebook" de los desarrolladores.

Está basado en web, pero también tiene su aplicación para Android, Windows, etc.



Linus Torvalds

Github

Cómo funciona

Master

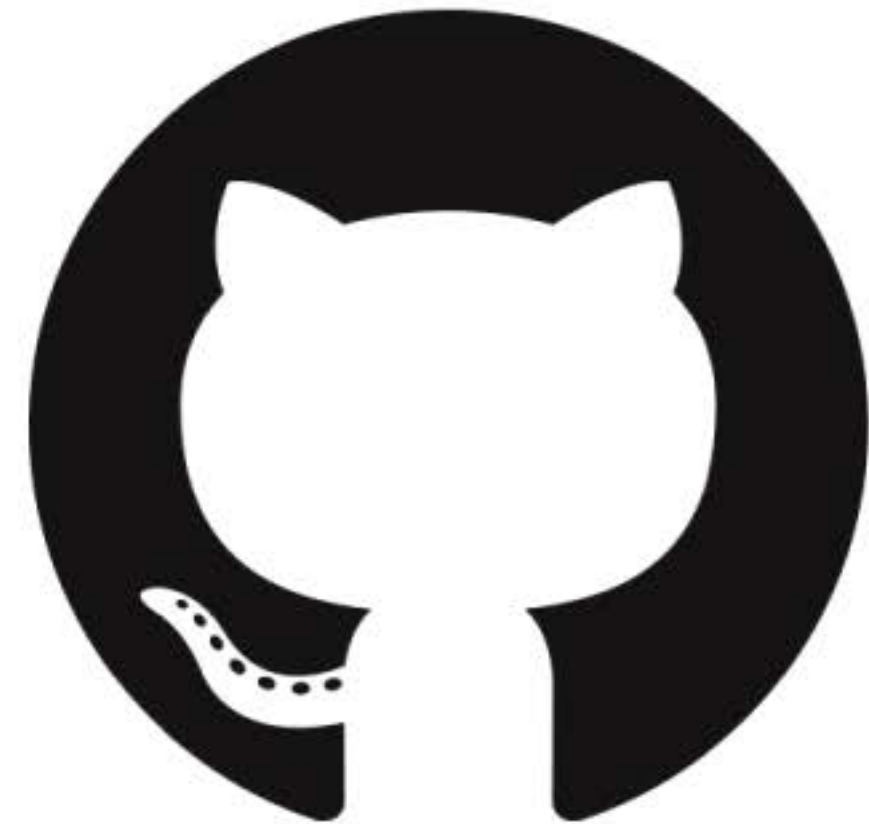
Tenemos un repositorio de código online en Github.

Clon

Se crea un clon de forma local y se lo trabaja, se lo edita, modifica, etc.

Commit

Cuando hacemos commit, enviamos los cambios a Github de nuestro clon local. Se sincroniza.



Github

Ventajas

- Múltiples versiones del mismo código

Posibilidad de trabajar de forma simultánea en el mismo código de forma local y cuando hacemos commit, GIT se encarga de fusionar, sincronizar y actualizar a todos el código de forma automática.

- Modificaciones Delta

Es una base de datos atómica de código. Que significa que sólo se modifica lo que cambia.

- Historial de cambios de archivo

Para volver a una determinada versión del código del archivo anterior.

- Múltiples usuarios

Múltiples personas pueden trabajar sobre el mismo archivo simultáneamente.

- Registro de acceso

Posibilidad de identificar todos los accesos.

- Branches (Ramas) - Repo Producción

Posibilidad de instalar en un servidor a GIT así evitamos el uso del FTP, sólo sincronizamos todo.

[.Repo Master Github](#)

[.Repo Master Local](#)

[.Repo Master Servidor](#)

Desventajas

- Archivos grandes

Github no es recomendable para archivos muy grandes, en ese caso, es recomendable el uso de FTP.

A los archivos binarios grandes no tiene sentido versionarlos.

www.github.com

GitHub

Search or type a command



Explore

Features

Enterprise

Blog

Sign in

Build software better, together.

Powerful collaboration, code review, and code management for
open source and private projects. Need private repositories?

Upgraded plans start at \$7/mo.

Pick a username

Your email

Create a password

Use at least one lowercase letter, one numeral, and
seven characters.

Sign up for GitHub

By clicking "Sign up for GitHub", you agree to our
[terms of service](#) and [privacy policy](#). We will send you
account related emails occasionally.

App

PC

windows.github.com



Mac

mac.github.com



Importancia

¿Por qué es importante Github?

Pocas personas son consideradas profesionales en el ámbito web.

Este perfil, es muy buscado, incluso fuera del país y de forma remota.

GitHub ha sido un gran lugar para contactar y vincular estas personas.

¿Qué mostrar en GitHub?

- Proyectos.
- Links.
- Proyectos que sigues.

Historial de código

Guarda cada nuevo cambio, personas que desarrollaron, fechas.

Clone vs Fork

Ambos procesos son similares, pero tienen 2 finalidades distintas.

Clone. Clonar un repositorio, como su nombre lo indica, es tener una copia exacta del proyecto en tu disco duro. Uso personales u organizacional.

Fork. Este proceso habla de compartir y aportar. En ésta se crea un clón del repositorio original dentro de tu cuenta de GitHub y se colabora directamente desde ahí.

Actualizaciones Push

Envía nuestras actualizaciones al repositorio clón.

Pull Request

Selector te pedirá qué propuesta te gustaría compartir con los propietarios del proyecto. Si ellos lo aceptan, se genera una fusión y habrás hecho la colaboración efectivamente.

id - class

¿Qué diferencia hay entre un identificador y una clase?

id - class

id

Sólo se puede escribir una vez

En CSS se maneja con numeral:

#logo

class

Se puede repetir múltiples veces

En CSS se maneja con punto:

.imagen_article

id - class

CSS cuando es ejecutado, le da prioridad a las `id` por sobre las `class`.

Tag: span

Usamos la etiqueta **span** sobre texto plano, para aplicarle determinado estilo sin que sea necesario convertirse en una caja div.

```
<p>Texto plano con <span>estilo</span> aplicado.</p>
```

Tag: p

Usamos la etiqueta **p** (paragraph) cuando tenemos texto semántico.

Funciona exactamente igual que un div, sólo que `<p>` se diferencia de un `<div>` en que los `p` tienen margen superior e inferior para separar los párrafos.

Tag: figure

Normalmente una imagen va dentro de una etiqueta div. Usamos el tag **figure** porque está concebido especialmente para incluir imágenes.

Retina display

Pantalla con el doble en cantidad de pixeles que realmente "dice" que tiene.

Un Iphone "dice" tener 320px pero en realidad tiene 640px, por ende si ponemos una imagen con 320px real, se va a ver borrosa o pixelada.

Para que se vea correctamente tendria que ser de 640px (o más) y luego, con html o javascript reducirla y hacer que se vea en el ancho de 320px que "dice" tener.

El navegador retina display va a lograr que se vea perfecto.

Recordemos

¿Qué hicimos en la 2ª clase?

Estructura semántica del diseño de muestra.

¿Qué vamos a hacer en la 3ª clase?

Aplicarle diseño web. Con CSS.

Aplicar diseño

1. Sincronizar cambios en Github

2. Normalizar el css

Cuando se comienza un proyecto, tenemos el problema de visualización del css en diferentes navegadores.

Esto se debe a que los navegadores arrancan con estilos predeterminados de cada navegador que son distintos entre sí.

El `<h1>` de Opera, es diferente al `<h1>` de Firefox y el `<h1>` de Chrome es diferentes a los anteriores 2.

Hasta hace poco tiempo, se reseteaba todo a cero, pero esto nos ponía el tamaño del `<h1>` igual al de `<p>`.

Gente coherente decidió llevar todo el reset de css que estaba en cero a una regulación standar de cómo debería ser en todos los navegadores.

normalize.css

necolas.github.io/normalize.css

Aplicar diseño

2. Normalizar el css

normalize.css

Instalación.

En Sublime tipear:

`<link`

más tab. Les tendría que quedar así:

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

link: es la etiqueta que nos permite "traer" estilos.

En html5 podemos eliminar type:

`<link rel="stylesheet" href="normalize.css">`

Le estamos ordenando al navegador que enlace una hoja de estilo que se encuentra en ese lugar.

3. Meta description

Corrección de sintaxis. En lugar de:

`<meta description="RWD" />`

Escribimos:

`<meta name="description" content="RWD" />`

Aplicar diseño

4. Creamos otro archivo .css

Lo colocamos debajo de

```
<link rel="stylesheet" href="normalize.css">
```

Esto es importante porque el navegador le da prioridad cuando renderiza de arriba hacia abajo a los archivos de estilo que linkeamos.

En Sublime creamos un nuevo archivo y lo salvamos **estilos.css** dentro de la carpeta "archivos".

Aplicar diseño

5. Cómo funciona CSS

```
etiqueta
{
  atributo: valor;
}
```

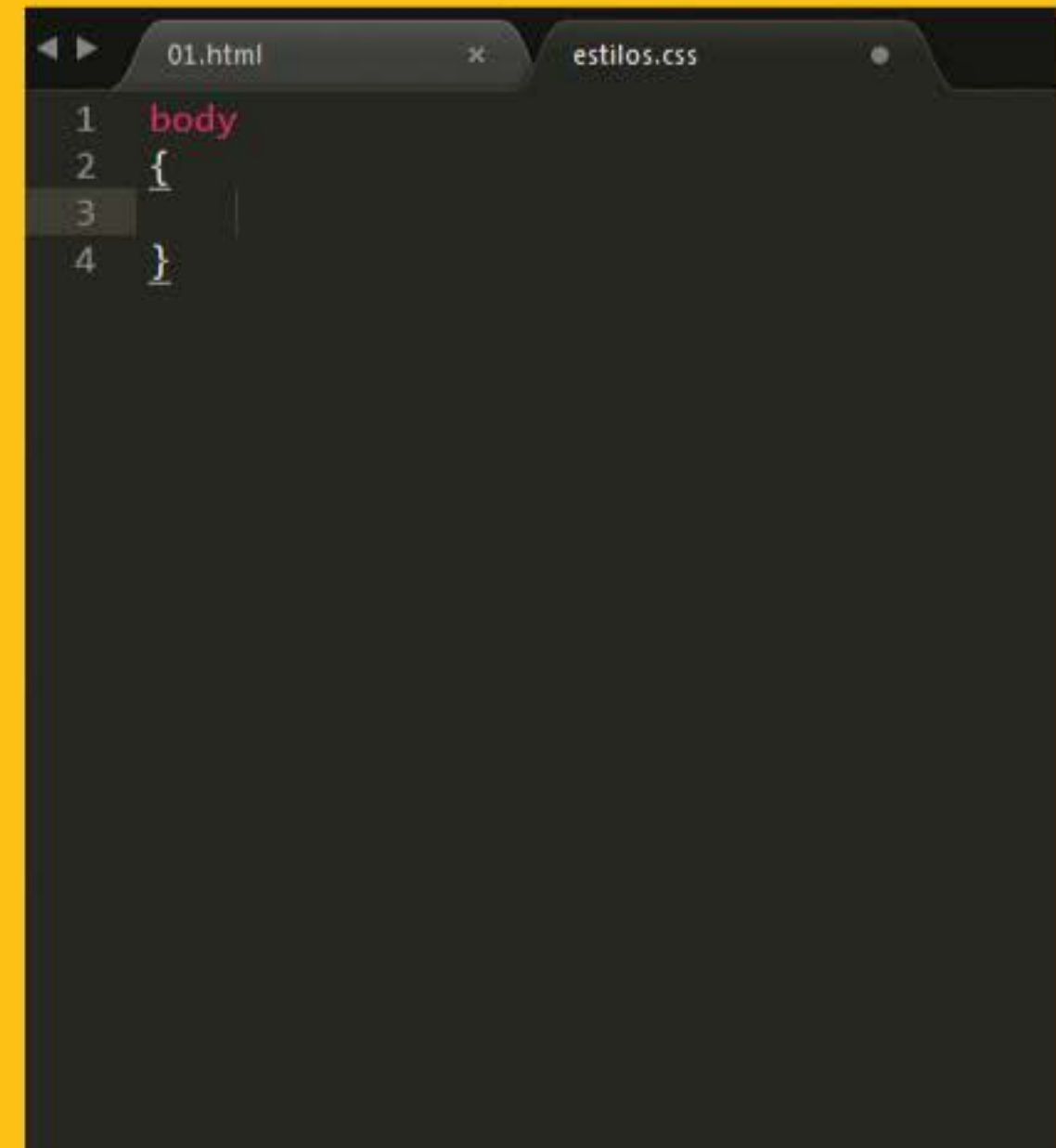
```
#id
{
  atributo: valor;
}
```

```
.clase
{
  atributo: valor;
}
```


Aplicar diseño

6. body

El body, es todo el cuerpo de mi página y es donde todo lo visible comienza.



A screenshot of a code editor with two tabs: '01.html' and 'estilos.css'. The 'estilos.css' tab is active, showing a CSS rule for the 'body' selector. The code is as follows:

```
1 body
2 {
3
4 }
```

Aplicar diseño

6. body

Comenzamos con el fondo gris #EEEEEE
(El # es necesario para declarar los colores hexadecimales)

background: #EEEEEE;

(Van a ver que puede aparecer como
background-color: #EEEEEE; pero eso se debe a
dreamweveritis.)

7. a

Los links #F16028 por defecto.

color: #F16028;

(**color** es el atributo que colorea al texto)



Aplicar diseño

8. font-family

Al comienzo sólo vamos a utilizar Arial. Más adelante vamos a utilizar otra.

(En **css** cuando el nombre de la fuente es una sola palabra, se la puede dejar sin comillas)

font-family: Arial;

Si por ejemplo utilizáramos Times New Roman, sería:

font-family: "Times New Roman";



Aplicar diseño

9. font-size

Por defecto normalize.css coloca 16px y es recomendable que sea el tamaño mínimo general.

Debemos aclarar el tamaño de la fuente en nuestro estilos.css también, nos servirá de recordatorio y agilizará el trabajo grupal.

font-size: 16px;



Conceptos de caja

¿Qué es una caja?

Una caja es cualquier objeto en html que envuelve a otro y que le permita tener márgenes y flotar.

Conceptos de caja

Las cajas por defecto tienen un **atributo** en css llamado **display** cuyo **valor** es **block**.

Conceptos de caja

etiqueta <p>

Texto texto texto texto
Texto texto texto texto
Texto texto texto texto

block

Caja
Texto texto texto texto

inline

Texto **Caja** texto
Texto texto texto texto

inline-block

Caja texto texto texto
texto texto **Caja**
Texto texto texto

Conceptos de caja

etiqueta <p>

Texto texto texto texto
Texto texto texto texto
Texto texto texto texto

Todo lo que agregemos se ubicará dentro de la caja madre.

Conceptos de caja

inline



Para que dentro de un párrafo podamos agregar una caja, se usa **inline** y se acomodará así.

Se va a mezclar con el texto pero genera, en base a su altura un aire por sobre el texto.

Por ejemplo las imágenes por defecto son inline. Es por eso que las insertamos dentro de un contenedor **figure**, que es un bloque, y que estén como si fuera una caja.

Conceptos de caja

block



Todo lo que agregemos se ubicará debajo como un salto de línea.

Conceptos de caja

inline



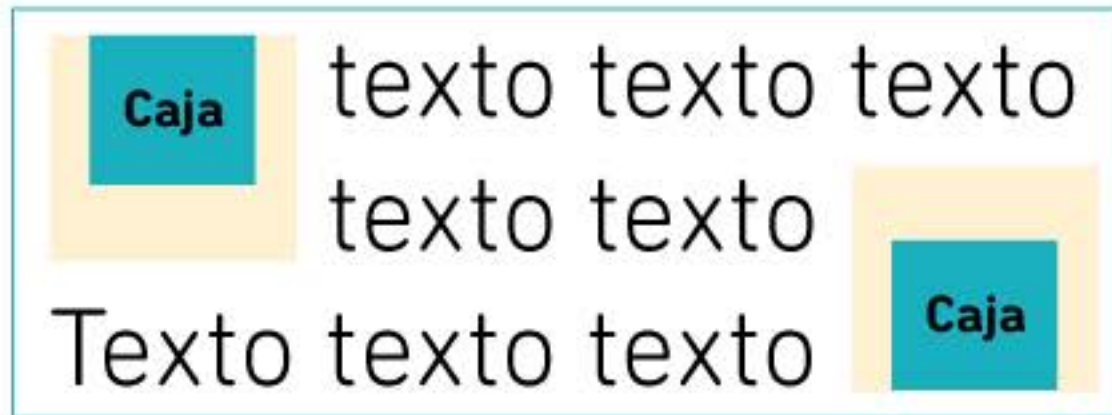
Para que dentro de un párrafo podamos agregar una caja, se usa **inline** y se acomodará así.

Se va a mezclar con el texto pero genera, en base a su altura un aire por sobre el texto.

Por ejemplo las imágenes por defecto son inline. Es por eso que las insertamos dentro de un contenedor **figure**, que es un bloque, y que estén como si fuera una caja.

Conceptos de caja

inline-block



Es la forma de tener cajas que tengan cosas enfrente y se mezcle con el contenido.

Cuando creamos una caja con un bloque y le ponemos inline, puede tener márgenes y elementos pegados.

Esa es la diferencia entre inline e inline-block.

Conceptos de caja

div, figure, header, footer, section, article, nav, p
son todas de tipo **block**.

video, img, span
son todas de tipo **inline**.

Aplicar diseño

10. header

Color de fondo **#474747**

Color de texto **blanco**

h1 tamaño de letra de **20px**

Logo de **60px** de ancho, que sería un tag **img**, dentro de **#logo**, que está dentro de **header**.

Avatar de **40px** de ancho, que sería un tag **img**, dentro de **#avatar**, que está dentro de **header**.

Para complicar un poco las cosas, al avatar lo vamos a hacer circular con el atributo:

border-radius



Aplicar diseño

10. header

Dijimos que los tags de tipo bloque no tienen nada en la parte de enfrente, es decir que todo lo que yo le pongo al lado, se va a “caer” para abajo.

Para que más de un bloque esté en el mismo nivel debemos aplicar **inline-block**.

¿A qué bloques deberíamos aplicarle inline-block?
A h1, el figure del logo y el figure del avatar

Existe otra técnica que se llama **float** (flotar).

También hay otra que veremos más adelante que son las posiciones relativas.

```

<link rel="stylesheet" href="normalize.css" />
<link rel="stylesheet" href="estilos.css" />
</head>
<body>
  <header>
    <figure id="logo">
      
    </figure>
    <h1>Taller de Responsive Web Design</h1>
    <figure id="avatar">
      
    </figure>
  </header>
  <nav>
    <ul>
      <li><a href="#">HTML5</a></li>
      <li><a href="#">CSS3</a></li>
      <li><a href="#">Javascript</a></li>
      <li id="nuevo_nav">
        <a href="#">Nuevo</a>
      </li>
    </ul>
  </nav>
  <section id="contenido">
```


Aplicar diseño

10. header

Queremos flotar a los elementos **figure** (logo y avatar).

Cuando le aplicamos **float** a una caja, esta pierde el margen.

Al no tener margen, podemos aplicarle margen al **header** contenedor.

padding es el margen interno de una caja.

margin es el margen externo de una caja.

Lo que haremos será poner un valor relativo al tamaño de la fuente tipográfica. Para eso, fuera pixeles y hola **em**.

```
    color: white;
    padding: 0.5em;
}
header figure
{
    float: left;
}
header h1
{
    font-size: 1.2em;
}
header #avatar img
{
    border-radius: 50%;
    width: 40px;
}
header #logo img
{
    width: 60px;
}
```

em

1em = valor inicial de fuente

Hace referencia a la letra m, que es la más cuadrada del abecedario.

Es importantísimo manejarse con em porque si queremos cambiar el diseño global en más o menos pixeles de nuestra fuente original, sólo tenemos que modificarlo en la primera declaración.

Cuando le aplicamos **float** a una caja, esta pierde el margen.

Entonces modificamos ahora la fuente de px a em.
En nuestro ejemplo ponemos 1.2em
(16px x 1.2)

```
padding: 0.5em;
}
header figure
{
  float: left;
}
header h1
{
  font-size: 1.2em;
}
header #avatar img
{
  border-radius: 50%;
  width: 40px;
}
header #logo img
{
  width: 60px;
}
```

Aplicar diseño

10. header

Ponemos margen al header con un padding del 50% que para em sería:

`padding: 0.5em;`

```
background: #EEEEEE;
font-family: Arial;
font-size: 16px;
}
header
{
    background: #474747;
    color: white;
    padding: 0.5em;
}
header figure
{
    float: left;
}
header h1
{
    font-size: 1.2em;
}
header #avatar img
{
    border-radius: 50%;
    width: 40px;
}
header #logo img
{
```


Aplicar diseño

10. header

Tomemos el h1 y apliquemos el diseño.
Le agregué una línea vertical blanca.

Nos vamos al id de logo:

`border-right: 1px solid white;`

Separamos la línea hacia la derecha del logo:

`border-right: 1px solid white;`

`padding: 0.5em;`

Hacemos espacio hacia la derecha de la línea:

`border-right: 1px solid white;`

`margin-right: 0.5em;`

`padding: 0.5em;`

Eliminamos el espacio que el h1 nos genera:

`font-size: 1.2em;`

`margin: 0;`



Aplicar diseño

10. header

Tenemos los figure "salidos" del contenedor del header. Para evitar esto, hay un atributo llamado **overflow** que "mete" los bloques.

Le ordenamos a overflow que esconda lo que se escapa del contenedor:

`overflow: hidden;`

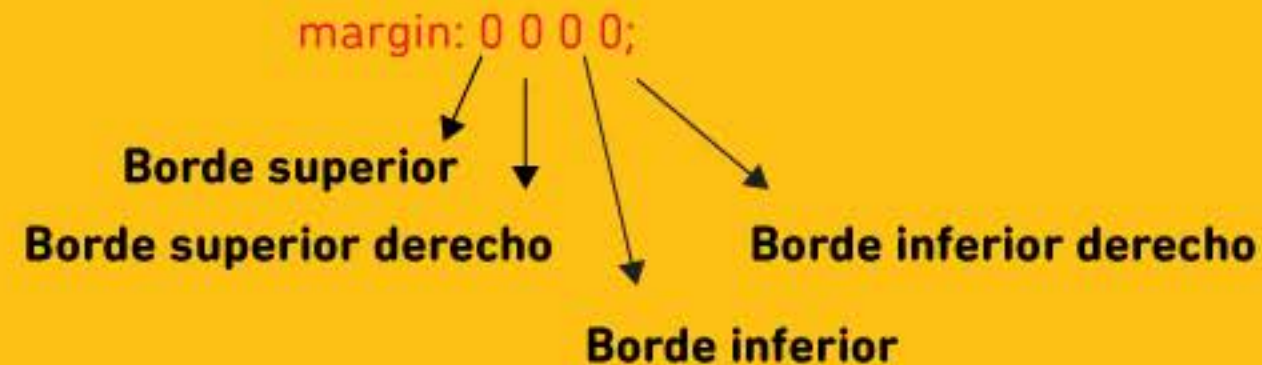


```
01.html x estilos.css x
}
header
{
    background: #474747;
    color: white;
    overflow: hidden;
    padding: 0.25em;
}
header figure
{
    float: left;
}
header h1
{
    font-size: 1.3em;
}
header #avatar img
{
    border-radius: 50%;
    width: 40px;
}
header #logo img
{
```


Aplicar diseño

10. header

Acomodemos ahora el avatar. Para eso tenemos que "quitarle" al h1 espacio a la derecha. Vemos que el avatar tiene de ancho 40px, por lo que podemos quitarle 70px. (Y hablamos en pixeles porque son elementos de diseño)



margin: 0 70px 0 0;

