Curso HTML, CSS y JavaScript

Junio de 2017

#### Autor: Fran Linde Blázquez

# Tema 5: Introducción a CSS

ÍNDICE

[Tema 5: Introducción a CSS 1](#_Toc485978895)

[1. Introducción 3](#_Toc485978896)

[2. Cómo incluir CSS en una página 5](#_Toc485978897)

[2.1 Estilos in-line o en línea 5](#_Toc485978898)

[2.2 Estilos dentro de la etiqueta <head> 5](#_Toc485978899)

[2.3 Estilos en ficheros externos .CSS 7](#_Toc485978900)

[3. Selectores CSS 9](#_Toc485978901)

[3.1 Selectores por etiqueta 9](#_Toc485978902)

[3.2 Selectores de clase 9](#_Toc485978903)

[3.3 Selectores por ID 11](#_Toc485978904)

[3.4 Selectores múltiples 13](#_Toc485978905)

[3.5 Selectores por relación 13](#_Toc485978906)

[4. Colores en CSS 15](#_Toc485978907)

[5. Fuentes en CSS 16](#_Toc485978908)

[6. Modelo de Cajas: Padding, Margin y Border 17](#_Toc485978909)

[7. Display 18](#_Toc485978910)

[8. Position en CSS 19](#_Toc485978911)

[9. Centrando elementos en CSS 20](#_Toc485978912)

[10. Pseudoclases: link, visited, hover 21](#_Toc485978913)

[11. Pseudoelementos after y before 22](#_Toc485978914)

[12. Elementos flotantes 24](#_Toc485978915)

## 

## 1. Introducción

Durante el Tema 1 hemos tratado algunos conocimientos básicos sobre la sintaxis de HTML, que nos han permitido desarrollar nuestras primeras páginas. En este tema realizaremos una introducción a la tecnología que nos permite aplicar estilos a estas páginas: CSS.

El nombre CSS proviene del inglés Cascading Style Sheets (hojas de estilo en cascada). El propio nombre nos indica que serán documentos y que su aplicación es en cascada (esto lo trataremos más adelante).

Mediante CSS tendremos la posibilidad de aplicar reglas de visualización como: colores, márgenes, bordes, sombras, tamaños y muchas más propiedades, a elementos HTML de nuestro documento, mediante la siguiente sintaxis:

1 selector {

2 propiedad: valor;

3 }

El selector indicará al navegador qué elementos se van a ver afectado por la regla. La propiedad será la que queramos modificar: ancho, alto, color, borde, sombra... etc. Y por último el valor será el que queramos aplicar a la mencionada propiedad.

Por ejemplo:

1 header {

2 color: red;

3 font-size: 28px;

4 }

5

6 p {

7 color: green;

8 }

En este ejemplo encontramos dos reglas:

* La primera tiene “header” como selector, por lo que aplicará a todas las etiquetas **<header>** de nuestra página (Nota: no deberíamos tener más de uno en cada página). Por un lado, se está indicando que la propiedad “color” debe tener el valor “red” (rojo) y por otro que la propiedad “font-size” (tamaño de la fuente) tiene el valor “28px” (28 píxeles).
* La segunda propiedad tiene “p” como selector, por lo que dicha regla se aplicará a todas las etiquetas **<p>** que aparezcan en nuestra página. Esta regla indica que la propiedad “color” debe tener el valor “green” (verde).

## 

## 2. Cómo incluir CSS en una página

Existen varias formas de incluir CSS en una página HTML, vamos a ver las más utilizadas:

### 2.1 Estilos in-line o en línea

Cada etiqueta HTML posee un atributo “style” sobre el cual podemos aplicar estilos directamente. De esta forma no necesitamos hacer uso de selectores, puesto que los estilos se aplicarán sobre propio elemento que los contiene.

Ejemplo:

1 <p style="background-color: grey;">Soy un texto con fondo gris</p>

En el código anterior podemos observar una etiqueta **<p>** sobre la cual estamos aplicando estilos en línea. En este caso estamos indicando que para ese párrafo la propiedad “background-color” tendrá el valor “grey” (gris).

Esta forma de aplicar CSS no se recomienda por varias razones:

* Se vuelve difícil tener control sobre todos los estilos de la página
* El tamaño de nuestro HTML crece, volviéndose pesado y difícil de mantener
* Los estilos en línea no nos permiten reutilizarlos. Si queremos que dos elementos tengan el mismo estilo, tendremos que copiar el atributo style entero en ambos elementos.
* No nos permite estructurar ni organizar nuestros estilos de ninguna manera

### 2.2 Estilos dentro de la etiqueta <head>

En el tema 1 (Introducción a HTML) vimos la etiqueta **<head>.** Además de los usos que se mencionaban, también ofrece la posibilidad de incluir CSS en nuestra página web haciendo uso de la etiqueta **<style>** como muestra el siguiente ejemplo:

1 <!DOCTYPE html>

2 <html>

3

4 <head>

5 <meta charset="utf-8">

6 <title>Este es mi blog</title>

7 <style>

8 a {

9 text-decoration: underline;

10 }

11 </style>

12 </head>

13

14 <body>

15 ...

En el ejemplo podemos observar cómo se ha incluido la etiqueta **<style>** bajo la cual hemos incluido una regla CSS. Esta regla aplica a todas las etiquetas **<a>** que encontremos en el documento. Lo que indica nuestra regla al navegador es que debe mostrar una línea por debajo de todos los enlaces de la página (etiquetas **<a>**).

Podemos también observar en el ejemplo anterior cómo hemos tabulado el CSS la misma manera que tabulamos el HTML: añadimos un nivel de tabulado o indentación dentro de cada regla CSS.

En esta forma de incluir CSS encontramos ventajas respecto a la anterior, pero aún quedan problemas que no nos resuelve:

* No podemos reutilizar estos estilos en otra página HTML salvo que copiemos y peguemos todo el contenido de la etiqueta **<style>**
* Nuestro HTML crece a medida que añadimos más estilos, pudiendo convertirse en un archivo demasiado grande.

### 2.3 Estilos en ficheros externos .CSS

La manera más correcta de incluir CSS en nuestras páginas es haciendo uso de ficheros externos que contengan nuestras reglas CSS. Para incluir ficheros CSS externos debemos incluir la etiqueta **<link>** dentro de nuestra etiqueta **<head>** de la siguiente manera:

1 <!DOCTYPE html>

2 <html>

3 <head>

4 <meta charset="utf-8">

5 <title>Este es mi blog</title>

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

7 </head>

8 <body>

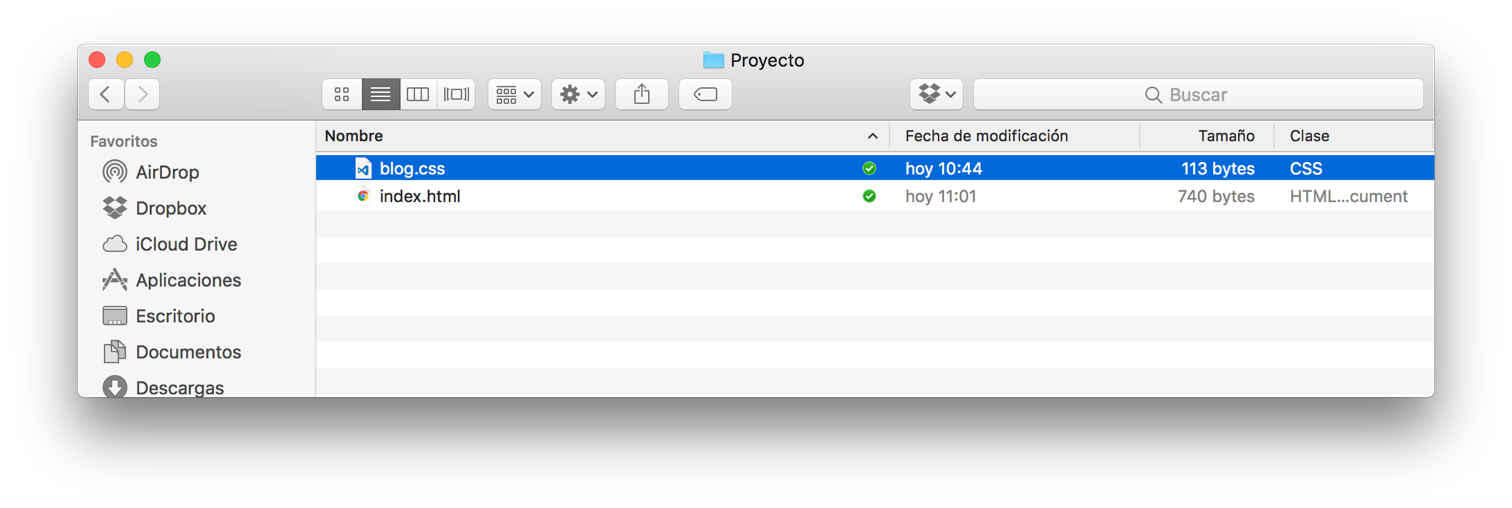
9 ...

Como podemos observar la etiqueta **<link>** posee varios atributos. Los dos primeros (“**rel**” y “**type**”) indican al navegador que es una hoja de estilos de tipo texto. El último atributo: “**href**” ya debería sonarnos, puesto que en las etiquetas **<a>** (enlaces) usamos el mismo atributo para indicar la ruta a la que queremos dirigir dicho enlace.

En este caso “**href**” nos servirá para indicar la ruta en la que se encuentra alojado el fichero CSS que queremos incluir.

En el ejemplo vemos que se está incluyendo un fichero llamado “blog.css”. Al no indicar más información sobre la ruta, estamos indicando al navegador que el fichero se encuentra en el mismo directorio que el HTML que lo está incluyendo.

Vamos por tanto a crear un fichero “blog.css” en la misma ruta que nuestro index.html de manera que nos quede así:



Dentro del fichero “blog.css” incluiremos las siguientes reglas CSS:

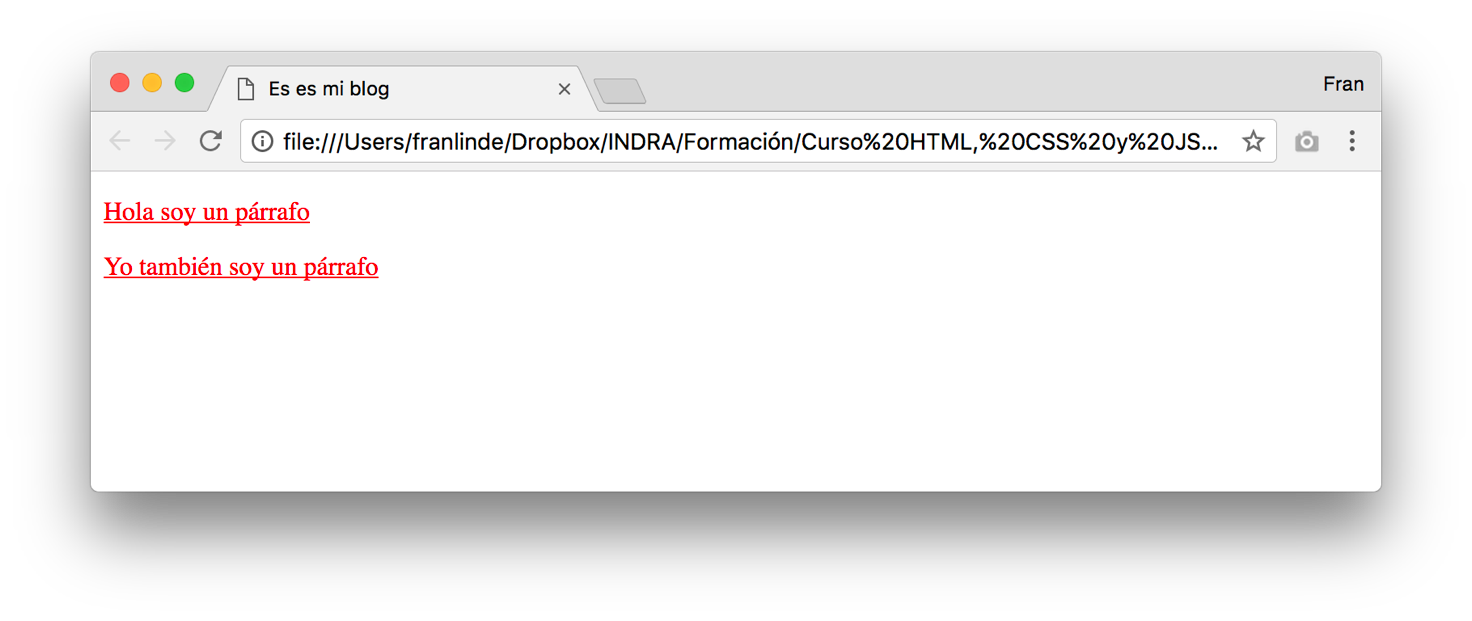
1 p{

2 color: red;

3 text-decoration: underline;

4 }

Si visualizamos ahora nuestro HTML en un navegador observaremos que se han aplicado las reglas CSS sobre todos los elementos **<p>** que tenemos en nuestro documento HTML.



## 3. Selectores CSS

Hasta ahora en nuestros ejemplos hemos usado como selectores las propias etiquetas HTML (**<p>** y **<a>**). Vamos a ver otros selectores CSS que nos permitirán organizar nuestro CSS mejor y ser más selectivos (valga la redundancia) en los elementos sobre los que aplicar los estilos.

### 3.1 Selectores por etiqueta

Son los selectores que hemos usado en los ejemplos anteriores. Son sencillos de usar, pero no permiten realizar una selección precisa de elementos, debido a que los estilos que indiquemos se aplicarán a todas las etiquetas de ese tipo que aparezcan en nuestro documento.

Ejemplo:

1 p{

2 color: gray;

3 font-size: 14px;

4 font-weight: normal;

5 }

6

7 h1{

8 font-size: 28px;

9 font-weight: bold;

10 }

Aunque no son los mejores selectores, ofrecen algunas ventajas, como por ejemplo a la hora de realizar unos estilos homogéneos en nuestras páginas. Es muy común que nos pidan como desarrolladores que todos los párrafos o todos los títulos tengan el mismo color, el mismo tamaño y la misma tipografía. Los selectores CSS por etiqueta nos permiten hacer esto de una manera sencilla.

### 3.2 Selectores de clase

Los selectores CSS de clase nos permiten indicar estilos diferentes a etiquetas que sean iguales. Es decir, que nos permiten tener dos párrafos con estilos distintos. Esto no es posible usando selectores de etiqueta.

Veamos un ejemplo con nuestro fichero “blog.css”. Puede observarse como los selectores CSS de clase se escriben indicando el nombre de la clase precedido por un punto:

1 .estilo1{

2 color: red;

3 font-size: 14px;

4 }

5

6 .estilo2{

7 font-weight: bold;

8 color: blue;

9 text-decoration: underline;

10 }

Por otro lado, nuestro fichero “index.html” luciría así:

1 <!DOCTYPE html>

2 <html>

3 <head>

4 <meta charset="utf-8">

5 <title>Este es mi blog</title>

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

7 </head>

8 <body>

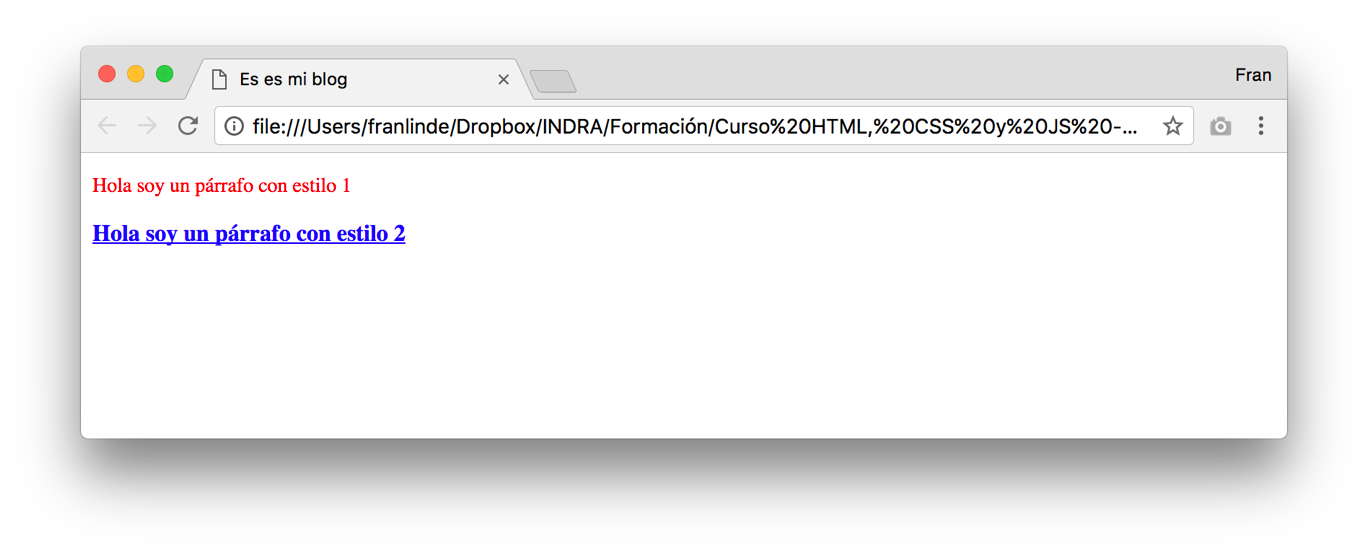
9 <p class="estilo1">Hola soy un párrafo con estilo 1</p>

10 <p class="estilo2">Hola soy un párrafo con estilo 2</p>

11 </body>

12 </html>

Y si visualizamos nuestro index.html en un navegador este será el resultado:



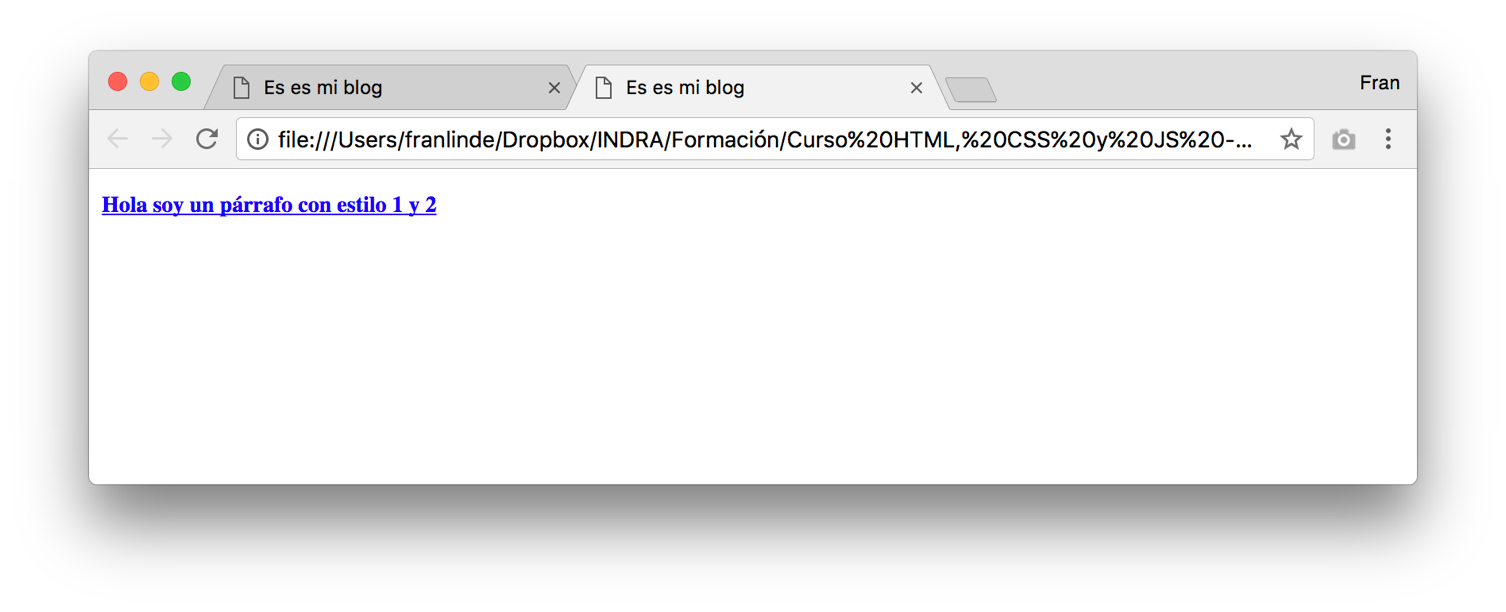
Como se puede observar en nuestro “index.html” la manera de añadir clases a nuestros elementos es mediante el atributo “class”, el cual acepta múltiples nombres de clases separados por espacios. Por ejemplo, si quisiéramos que un párrafo tuviera dos estilos lo haríamos de la siguiente manera:

1 ...

2 <p class="estilo1 estilo2">Hola soy un párrafo con estilo 1 y 2</p>

3 ...

Si visualizamos nuestro index.html en un navegador podremos observar lo siguiente:



Se observa que se han aplicado ambas reglas y encontramos algo interesante: se ha impuesto la propiedad “color: blue;” sobre “color: red;”. Esto se debe a la naturaleza de CSS: recordando el nombre “Cascasing Style Sheets”: Cascading viene a indicarnos que los estilos se aplican en forma de cascada: a igualdad de condiciones, se aplica el último estilo cargado. Por este motivo prevalece el estilo impuesto por la clase “estilo2”.

Ejercicio: Prueba a cambiar el orden de las clases en el fichero blog.css ¿Cómo debería verse nuestro párrafo ahora?

### 3.3 Selectores por ID

En algunas ocasiones nos gustaría que nuestras reglas CSS se aplicasen solo a un elemento, en esos casos es posible que debamos usar como selector un ID. A diferencia de los selectores de clase, cuyo nombre debe ir precedido por un punto, los selectores de ID deben ir precedidos por una almohadilla (#). Veamos un ejemplo sobre nuestro HTML:

1 ...

2 <p class="estilo1">Hola soy un párrafo con estilo 1</p>

3 <p class="estilo1 estilo2">Hola soy un párrafo con estilo 1 y 2</p>

4 <p id="estilo-especial">Soy un párrafo único</p>

5 ...

Nuestro fichero “blog.css” tendría el siguiente aspecto:

1 .estilo1{

2 color: red;

3 font-size: 14px;

4 }

5 .estilo2{

6 font-weight: bold;

7 color: blue;

8 text-decoration: underline;

9 }

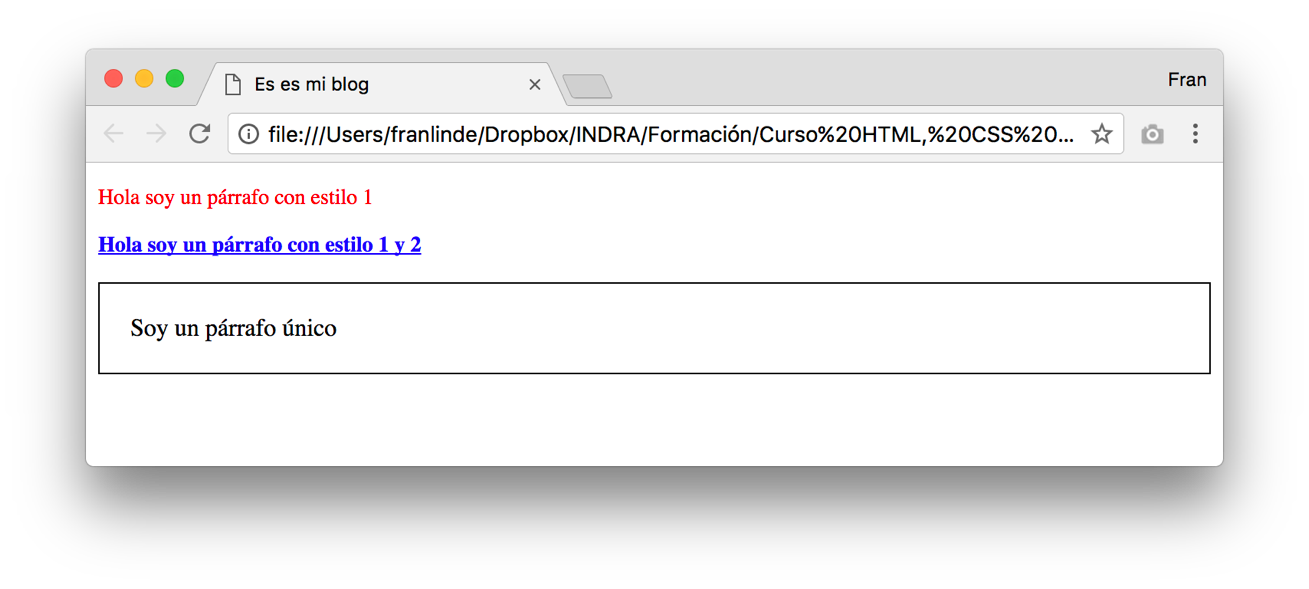
10 #estilo-especial{

11 border: 1px solid #000;

12 padding: 20px;

13 }

Si visualizamos nuestro index.html en un navegador se vería así:



Debemos tener en cuenta que por corrección no debemos usar más de un ID dentro de nuestro documento HTML, de hecho, el uso de un mismo ID múltiples veces en nuestro documento produciría errores de validación de nuestro documento.

### 3.4 Selectores múltiples

Una regla puede ser aplicada sobre múltiples elementos separándolos por comas. Por ejemplo, si quisiéramos que todos los párrafos y además los enlaces con la clase “especial” tuvieran el mismo estilo, lo haríamos de la siguiente manera:

1 p, a.especial{

2 color: blue;

3 font-size: 20px;

4 }

En este ejemplo podemos observar algo nuevo: hemos concatenado el nombre de una etiqueta y una clase, de manera que observamos: “etiqueta.clase”. Este selector indica que los estilos se aplicarán solo sobre los elementos de tipo “etiqueta” que además tengan dicha clase.

### 3.5 Selectores por relación

Existe otras formas de combinar selectores. Por ejemplo, si quisiéramos indicar al navegador que todos los párrafos dentro del footer deben tener un estilo determinado, podemos hacerlo de la siguiente manera:

1 footer p{

2 margin: 20px;

3 font-weight: bold;

4 }

La combinación de dos selectores separados por espacio indica anidación a nivel de HTML. Por tanto, en nuestro ejemplo, todos los elementos **<p>** que se encuentren dentro de nuestro **<footer>** se verán afectados por esta regla.

## 4. Colores en CSS

Principalmente existen tres formas de indicar un color en CSS:

* Por nombre: existen ciertos nombres reservados y que pueden ser utilizados, como por ejemplo: red, blue, green… etc. Se puede encontrar la lista completa de colores disponibles en: <https://developer.mozilla.org/en-US/docs/Web/CSS/color_value>
* RGB o RGBA: también pueden indicarse mediante un código RGB o RGBA. Los códigos RGBA son como los RGB, pero incluyen un parámetro más que indica el nivel de opacidad del color

*Nota: (a en rgba proviene del inglés alpha = opacidad).*

* Hexadecimal: Por último pueden indicarse mediante un código hexadecimal.

Ejemplo:

1 .clase-con-color-por-nombre{

2 color: red;

3 }

4

5 .clase-con-color-rgb{

6 color: rgb(255, 0, 0);

7 }

8

9 .clase-con-color-rgba{

10 color: rgba(255, 0, 0, 1);

11 }

12

13 .clase-con-color-hexadecimal{

14 color: #FF0000;

15 }

En este ejemplo el color siempre es el mismo: rojo. Y es que en HTML/CSS podemos obtener un mismo resultado de múltiples maneras, esto mismo lo veremos en muchos otros ejemplos.

## 5. Fuentes en CSS

Durante la fase de diseño de una página web o una aplicación móvil es muy común que se incluyan distintas tipografías, con distintos grosores y tamaños. Vamos a ver qué propiedades nos ofrece CSS para indicar al navegador la fuente que queremos usar:

* Font-Family: Indica al navegador el nombre de la fuente que debemos usar. Para ello la fuente debe estar disponible a nivel de sistema o haber sido importada en nuestra web.
* Font-Weight: Indica el grosor de la fuente. Acepta valores entre 100 y 900, creciendo siempre de 100 en 100.
* Font-Size: Indica el tamaño de la fuente. Puede ser indicado en píxeles, en em, en porcentajes…

Ejemplo de blog.css:

1 .fuente-bonita{

2 font-family: Arial;

3 font-weight: 100;

4 font-size: 14px;

5 }

Al igual que indicábamos al final del Tema 1 (Introducción a HTML), hemos visto algunas nociones básicas sobre CSS, pero debemos aún profundizar mucho más en temas sucesivos.

## 6. Modelo de Cajas: Padding, Margin y Border

A la hora de estilizar nuestros componentes es importante que apliquemos espaciados para una correcta posición de los mismos. Esto se consigue mediante las propiedades padding y margin.

* **Padding:** aplica un espaciado al interior del elemento seleccionado. Esto hace que el elemento crezca o el contenido dentro del mismo se reduzca.
* **Margin:** aplica espaciado alrededor del elemento seleccionado, alejando este elemento del resto.

Una vez tenemos posicionado nuestros elementos podremos aplicarle un borde, con la propiedad border, que se situará entre los espaciados aplicados con las propiedades antes descritas.

## 7. Display

Define el tipo de caja usada para un elemento HTML. Existen muchos tipos de valores que podemos asignar a esta propiedad, los más utilizados son:

* **Inline**: Valor por defecto. Expone un elemento como elemento en línea. No admite width, height ni margin.
* **Block**: Expone un elemento como elemento de bloque. Este elemento ocupara todo el ancho posible.
* **Inline-block:** Su comportamiento es una mezcla de los dos anteriores, se muestran en la misma línea (respetando el flujo) todos los elementos y además, acepta las propiedades width, height y margin.
* **None:** El elemento no se mostrará ni tendrá́ efecto alguno en la estructura de la web.
* **Flex:** Expone un elemento como un elemento de bloque de tipo Flex. Nuevo en CSS3.

## 8. Position en CSS

La propiedad position especifica que método de posicionamiento es usado para un elemento. Existen cuatro tipos de posicionamiento:

* **Static:** Posicionamiento por defecto. El elemento será́ posicionado acorde al flujo normal de la web. Propiedades como top, right, bottom y left no tendrán efecto.
* **Relative:** El elemento se posiciona acorde al flujo normal de la web. Propiedades como top, right, bottom y left tendrán efecto e iniciarán las coordenadas 0,0 en el punto de partida del elemento.
* **Absolute:** El elemento se posiciona relativamente al primer padre con el valor relative. Propiedades como top, right, bottom y left tendrán efecto e iniciarán las coordenadas 0,0 en el punto de partida del primer padre con el valor relative.
* **Fixed:** El elemento se posiciona relativamente a los extremos de la pantalla y quedará fijo aunque hagamos scroll. Propiedades como top, right, bottom y left tendrán efecto e iniciarán las coordenadas 0,0 en el borde superior izquierdo del viewport.

## 9. Centrando elementos en CSS

Hay muchas formas de centrar con CSS un elemento. Esto dependerá del elemento que sea, sus estilos y los de los elementos que lo rodean. A continuación expondremos algunos de las formas más comunes de alinear elementos y contenidos:

• Text-align: Podemos alinear un texto al centro con tan solo darle a un elemento la propiedad text-align con el valor center.

• Margin: Podemos alinear elementos de bloque horizontalmente con tan solo agregar la propiedad margin y el valor auto a un elemento. Esto puede ser aplicado a cualquier elemento, incluso a imágenes, si lo convertimos en un elemento de bloque mediante el estilo display: block.

• Position: A la hora de la verdad, es difícil que el anterior caso se dé, por lo que tenemos que recurrir a métodos alternativos como este. Basta con especificar a un padre el estilo position: relative y agregar al hijo que queremos centrar los siguientes estilos:

.centrando-hijo{

position: absolute;

left: 50%;

top: 50%;

transform: translate(-50%, -50%);

}

De este modo conseguimos que el elemento se posicione tanto vertical como horizontalmente al elemento padre.

## 10. Pseudoclases: link, visited, hover

Las pseudoclases nos sirven para aplicar estilos a diferentes estados de los diferentes elementos. Siempre van seguidas de un selector css y dos puntos.

Por ejemplo: .clase:pseudoclase

Las más usadas son:

* Link: pseudoclase única para enlaces. Aplica a los enlaces que aún no han sido visitados.
* Visited: pseoduclase única para enlaces. Aplica a los enlaces que ya han sido visitados.
* Hover: Aplica estilos cuando tenemos el ratón por encima.
* Active: Aplica estilos cuando un elemento es activado por el usuario. Por ejemplo, cuando se pulsa sobre él.
* Focus: Aplica estilos cuando tenemos el foco en un elemento.

## 11. Pseudoelementos after y before

En HTML existen varios pseudo-elementos que nos permiten aplicar estilos a ciertas partes de nuestro HTML. Se puede encontrar una lista completa sobre los pseudo-elementos HTML en <https://developer.mozilla.org/es/docs/Web/CSS/Pseudoelementos>

Nosotros vamos a tratar los dos más importantes: “**::after**” y “**::before**”.

Los pseudo-elementos se generan a través de CSS y se aplican sobre elementos ya existentes. La sintaxis de los pseudo-elementos es la siguiente:

1 selector::pseudo-elemento{

2 /\*Estilos\*/

3 }

Mediante esta sintaxis estamos estilando un elemento virtual del DOM:

* **::after** crea un pseudo-elemento que se convierte en el último hijo del elemento seleccionado. Nos permite añadir elementos de estilo inmediatamente después del elemento seleccionado.
* **::before** crea un pseudo-elemento que se convierte en el primer hijo del elemento seleccionado. Nos permite añadir elementos de estilo justo antes del comienzo del elemento seleccionado.

La manera de añadir contenido sobre estos pseudo-elementos es mediante la propiedad CSS “content”.

Veamos un ejemplo sobre nuestro fichero CSS:

1 .cita::before {

2 content: "«";

3 color: gray;

4 }

5

6 .cita::after {

7 content: "»";

8 color: gray;

9 }

Con estas propiedades estamos indicando a nuestro navegador que antes y depués de los elementos que tengan la clase cita, vamos a añadir unas marcas que nos indican que la frase es una cita. Haremos uso de este CSS sobre nuestro HTML

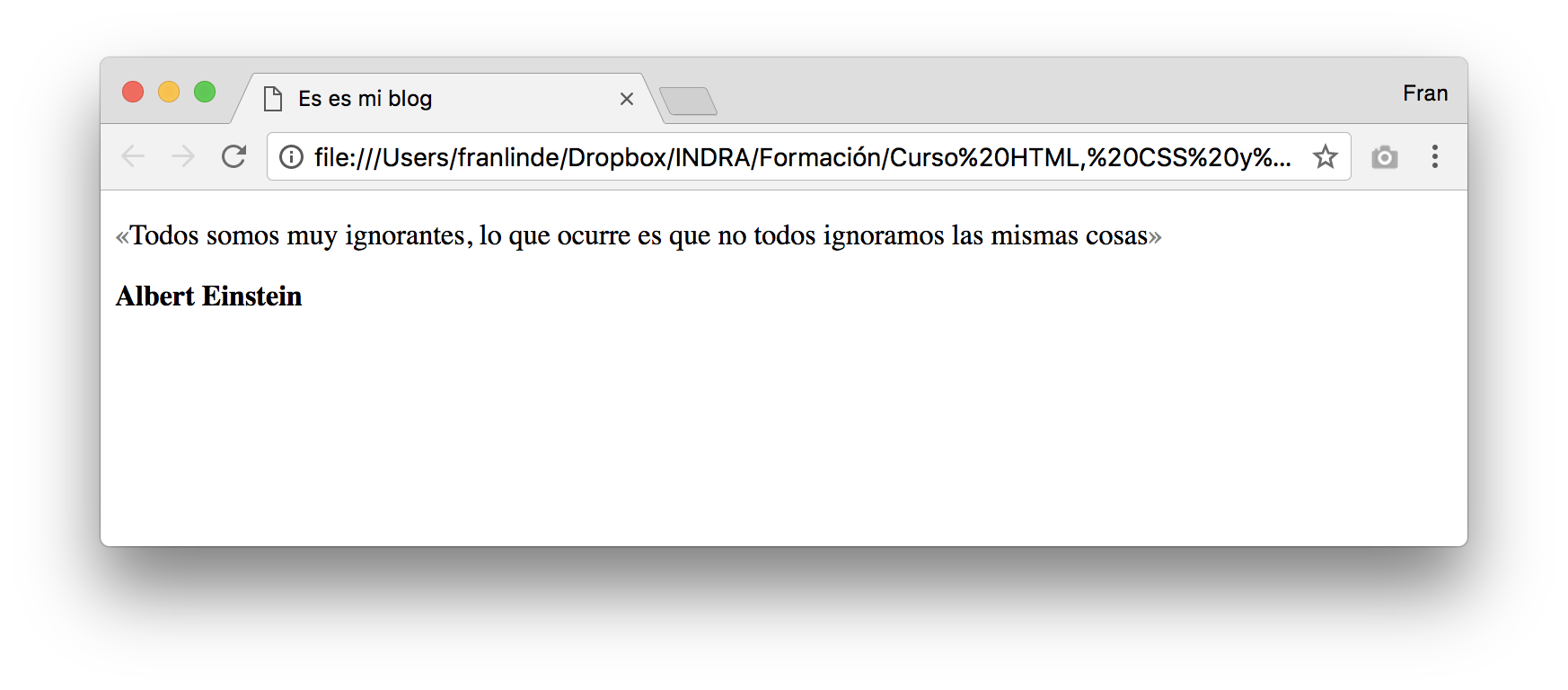
1 ...

2 <p class="cita">Todos somos muy ignorantes, lo que ocurre es que no todos ignoramos las mismas cosas</p>

3 <p><strong>Albert Einstein</strong></p>

4 ...

Y obtendremos el siguiente resultado:



## 12. Elementos flotantes

El posicionamiento flotante es uno de los más utilizados a la hora de maquetar una página web hoy en día. Cuando especificamos que un elemento estará́ flotado, este se desplazará lo más a la izquierda o derecha que el elemento pueda. De esta forma la caja flotante deja de pertenecer al flujo normal de la página y los elementos adyacentes tomarán su lugar.

Esta propiedad admite tres valores (**left**, **right** y **none**) para desplazar la caja a la izquierda, derecha o para que un elemento no sea flotante. Este flujo flotante puede ser cortado mediante la propiedad clear, hacia la izquierda con clear: left, hacia la derecha con clear: right, hacia ambos lados con clear: both o no cortar el flujo con clear: none.