# Otros estilos CSS

Veremos a continuación una serie de propiedades que podemos aplicar a diferentes elementos CSS que no se han recogido en los documentos anteriores. Por ejemplo, estilos específicos para enlaces o tablas, así como algunos efectos avanzados de animación que se incorporaron con CSS3 y algunos selectores más avanzados que aplicar en nuestras reglas CSS.

## 1. Estilos para enlaces

Los enlaces son un elemento muy potente en una página web, pero su estilo es difícil de controlar. De hecho, nada más poner el enlace, se nos verá de color azul en la página, y cuando hagamos clic sobre él, el resto de veces aparecerá de color morado, indicando que es un enlace que ya hemos visitado antes.

Imaginemos que queremos hacer una página web con el color de fondo azul. ¿Qué pasaría con los enlaces? Apenas se verían. O incluso si usamos otros colores, puede que el color de los enlaces desentone con lo que hay alrededor. Para cambiar el estilo de los enlaces mediante CSS, es algo más complicado que para un elemento normal (como un párrafo, por ejemplo), ya que los enlaces tienen diferentes estados, y debemos cambiar el estilo para cada uno de esos estados. Para hacer esto, se utilizan unos elementos de CSS llamados **pseudo-clases**, que complementan a un selector para indicar una característica particular del mismo.

El estilo para el estado normal de un enlace se cambia con la pseudo-clase *:link*, formando el selector `a:link`. Por ejemplo, si queremos poner el texto del enlace de color blanco:

```
a:link { color: white }
```

Pero si hacemos clic en el enlace, como ya pasará a ser un enlace visitado, la próxima vez que carguemos la página ya no aparecerá blanco, sino morado (color por defecto para enlaces visitados). Para el estilo para enlaces visitados, usamos el selector `a:visited`, que contiene la pseudo-clase *:visited*:

```
a:visited { color: white }
```

Adicionalmente, el enlace tiene tres estados más sobre los que podemos actuar: cuando pasamos el ratón por encima del enlace, sin pincharlo (se accede con el selector `a:hover`), cuando hacemos clic en el enlace (se accede con el selector `a:active`) y cuando establecemos el foco en el enlace a través del teclado (selector `a:focus`, aunque éste apenas se utiliza).

Por ejemplo, si queremos que, en cualquier estado, los enlaces se muestren de color blanco, tendríamos que añadir estas reglas a nuestro CSS (en realidad bastaría con las dos primeras, si no queremos hacer nada al

usar el ratón en los enlaces):

```
a:link { color: white; }
a:visited { color: white; }
a:focus { color: white; }
a:hover { color: white; }
a:active { color: white; }
```

O bien una sola regla conjunta:

```
a:link, a:visited, a:focus, a:hover, a:active { color: white; }
```

La primera opción es más versátil si luego queremos añadir alguna opción extra en algún estado (por ejemplo, cambiar el tamaño de la letra al pasar el ratón por encima). Es importante mantener el **orden** al definir estas reglas, porque son acumulativas y se van aplicando una tras otra. Así, cuando hacemos clic en un enlace (selector *:active*), también estamos encima de él (selector *hover*).

También podemos, entre otras cosas, actuar sobre el subrayado que hay en los enlaces, con la propiedad `text-decoration`. Si por ejemplo queremos que los subrayados se quiten al pasar el ratón por encima, haríamos:

```
a:hover { color: white; text-decoration: none; }
```

> **Ejercicio 1:**
>
> Vamos a seguir modificando el archivo **harrypotter.html** que hemos venido haciendo en ejercicios anteriores. Añade estos estilos:
>
> - Haz que todo el documento (selector body) tenga tipo de letra Arial y color de fondo azul claro (en formato hexadecimal, busca un tono que te guste).
> - Haz que los enlaces (para el índice inicial) tengan letra de color blanco, sin subrayado, y que al pasar el ratón por encima, o hacer clic en el enlace, la letra cambie a color naranja y subrayado.

## 1.1. Añadir iconos en enlaces

Podemos añadir un icono o imagen pequeña como parte de un enlace (junto con el texto) de varias formas. Por ejemplo, añadiendo la correspondiente etiqueta *img* dentro del enlace, y posicionándola junto con el texto. Alternativamente, podemos definir el icono del enlace a través de CSS:

```
a
{
  background: url('imagenes/icono.png') no-repeat 100% 0;
  background-size: 2rem;
  padding-right: 2.5rem;
}
```

Este estilo hará que se cargue la imagen *icono.png* pegada al borde derecho del enlace (100% del ancho), y a 0 píxeles del borde superior, con un escalado (*2rem*) y un *padding* o separación entre el texto y el icono.

## 1.2. Definir enlaces con apariencia de botones

Jugando con propiedades como el color de fondo o los bordes podemos definir enlaces que, en determinadas ubicaciones, tengan una apariencia de botones. Por ejemplo, en sesiones anteriores hemos definido barras de navegación utilizando listas:

```
nav
{
    background-color: darkgrey;
}

nav ul.navegacion
{
    list-style-type: none;
    text-align: center;
}

nav ul.navegacion li
{
    display: inline;
    margin: 20px;
}
```

Podemos definir enlaces en cada uno de los items de la lista de navegación:

```
<nav>
    <ul class="navegacion">
        <li><a href="...">Inicio</a></li>
        <li><a href="...">Preguntas frecuentes</a></li>
        <li><a href="...">Tienda online</a></li>
        <li><a href="...">Contacto</a></li>
    </ul>
<nav>
```

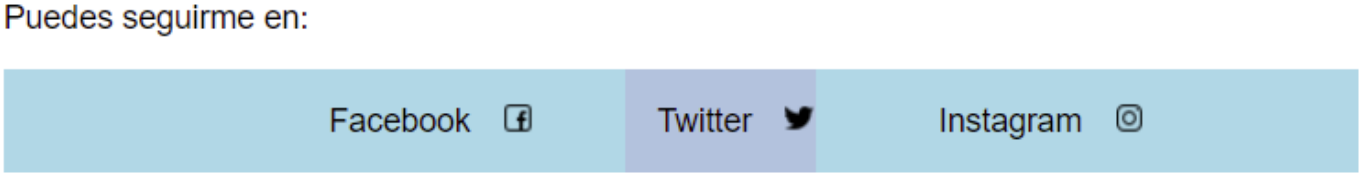
Y añadir estilos adicionales a los anteriores para concretar cómo se van a mostrar estos enlaces. Por ejemplo, podemos cambiar su color de fondo cuando pasemos el ratón por encima:

```
nav ul li a
{
  text-decoration: none;
  display: inline-block;
  padding: 1rem;
  color: black;
}

nav ul li a:hover
{
  background: lightgrey;
}

nav ul li a:active
{
  background: white;
}
```


> **Ejercicio 2:**
>
> Descarga esta plantilla de ejercicio. Verás una página *index.html* con una lista de enlaces (vacíos), y una carpeta con 3 iconos. Se pide que añadas un documento **estilos.css** enlazado desde la página anterior para mostrar una apariencia como esta:




> Al pasar el ratón por encima de cada item, se cambiará el color de fondo como se muestra en el caso de Twitter en el ejemplo (puedes elegir los colores que quieras).

# 2. Estilos para tablas

En documentos anteriores, cuando hemos visto cómo definir tablas en HTML, comentamos que existen algunos atributos que nos permitían modificar algunas propiedades de las tablas, tales como su borde o su

anchura. Sin embargo, la mejor forma de definir la apariencia visual de un elemento no es a través de sus atributos HTML, sino de los estilos CSS.

Podemos utilizar la propiedad `border` de CSS para definir el borde tanto de toda la tabla (borde alrededor de los límites de la tabla) como de las casillas que la componen (elementos *th* y *td*).

```
table, th, td
{
    border: 1px solid black;
}
```

Esto dejará una apariencia como la que tendríamos poniendo `table border="1"` en la propia etiqueta *table*. Sin embargo, ya vimos al tratar estos bordes en HTML que la apariencia final deja un borde doble, debido a que todos los elementos (tanto la tabla en sí como las celdas) tienen un borde. Para evitar esto y dejar un único borde en cada elemento, podemos **colapsar** estos bordes, a través de la propiedad `border-collapse`:

```
table, th, td
{
    border: 1px solid black;
    border-collapse: collapse;
}
```

> **NOTA**: la propiedad *border-collapse* se puede aplicar únicamente a la tabla (etiqueta *table*), no es necesario aplicarla también a las casillas de la misma.

En lo que respecta a la **anchura** (*width*), podemos especificar una anchura general para la tabla en sí:

```
table
{
    width: 60%;
}
```

O también para alguna de sus columnas (previamente identificada con algún tipo de clase o *id*):

```
td.primera
{
    width: 50%;
}
td.segunda
{
    width: 30%;
}
```

# 3. Selectores avanzados

Vamos a dar una vuelta más de tuerca a lo que son los selectores CSS. En un documento anterior hemos visto que fundamentalmente se utilizan cuatro tipos: de etiqueta, descendente, de clase y de id. Pero existen algunos otros que pueden sernos útiles

El **selector de hijos** es similar al descendente, pero aplica el estilo sólo a etiquetas que cuelguen directamente de la contenedora. Por ejemplo, para aplicar un estilo a negritas que están en párrafos (sin etiquetas intermedias), pondremos:

```
p > strong { ... }
```

El **selector adyacente** se utiliza para aplicar un estilo a un elemento que sea hermano adyacente de otro (es decir, a continuación de otro). Por ejemplo, para cursivas que vayan detrás de negritas (no dentro de ellas) usamos:

```
strong + em { ... }
```

Alternativamente, el **selector de hermanos** es menos restrictivo que el anterior: se aplica a los elementos de un tipo que vayan después de uno dado (sin que necesariamente sea inmediatamente después). Por ejemplo, este selector afecta a los párrafos que vayan después de un *h1*:

```
h1 ~ p { ... }
```

El **selector de atributos** se utiliza para aplicar el estilo a las etiquetas que tengan un atributo con un cierto valor. Ponemos la etiqueta, y luego entre corchetes el atributo y el valor que debe tener, entre comillas. Por ejemplo, si queremos aplicar un estilo a todos los enlaces que vayan a la página de la Universidad de Alicante, pondríamos algo como:

```
a[href="https://www.ua.es"] { ... }
```

Ya hemos visto anteriormente en este documento que en algunos selectores podemos emplear **pseudo-clases** para concretar un estado particular del elemento. Así, el selector `a:hover` se aplica a los enlaces cuando estamos pasando el ratón por encima del mismo. Existen otras pseudo-clases que podemos aplicar en diferentes elementos para referirnos a estados o subelementos concretos.

- La pseudo-clase `first-child` sirve para aplicar un estilo a la primera etiqueta de la que estamos tratando. Así, el selector `p:first-child` se aplicará al primer párrafo de una secuencia de párrafos. De forma similar, podemos emplear el selector `first-of-type` para identificar el primer elemento que sea de un tipo dentro de una secuencia. Así, por ejemplo, `div > p:first-child` sólo se aplicará a los párrafos que sean los primeros hijos de un *div*, mientras que `div > p:first-of-type` se aplicará al primer párrafo de un *div*, aunque haya otras cosas antes.
- Del mismo modo, la pseudo-clase `last-child` se aplica a la última etiqueta de una secuencia. Por ejemplo, `li:last-child` se aplicará al último item de una lista. Análogamente al caso anterior, también podemos emplear `last-of-type`.
- También disponemos de la pseudo-clase `nth-child(x)` donde podemos indicar estilos concretos para elementos concretos de una secuencia. Por ejemplo, `li:nth-child(even)` aplica los estilos a los items pares de una secuencia, y `li:nth-child(odd)` aplicaría a los impares. El selector `li:nth-child(4)` aplica el estilo al cuarto item en la lista, y el selector `li:nth-child(4n)` lo aplica cada cuatro elementos.

Podéis aprender más sobre pseudo-clases y sus usos en webs como ésta.

> **Ejercicio 3:**
>
> Dado el siguiente contenido HTML:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Prueba</title>
</head>
<body>
    <section id="inicio">
        <h2>Inicio</h2>
        <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Quae, aliquam!</p
        <p>Lorem ipsum dolor sit amet consectetur, adipisicing elit.</p>
        <p>Lorem ipsum dolor sit amet consectetur adipisicing.</p>
    </section>
    <section id="contenido">
        <h2>Contenido</h2>
        <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Quae, aliquam!</p
        <p>Lorem ipsum dolor sit amet consectetur, adipisicing elit.</p>
        <p>Lorem ipsum dolor sit amet consectetur adipisicing.</p>
    </section>
    <section id="fin">
        <h2>Fin</h2>
        <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Quae, aliquam!</p
        <p>Lorem ipsum dolor sit amet consectetur, adipisicing elit.</p>
        <p>Lorem ipsum dolor sit amet consectetur adipisicing.</p>
    </section>
    <section id="tabla">
        <table>
            <tr>
                <td>Lorem, ipsum dolor.</td>
                <td>Ab, deserunt officia!</td>
                <td>Amet, temporibus molestiae.</td>
                <td>Facere, corrupti praesentium?</td>
            </tr>
            <tr>
                <td>Lorem, ipsum dolor.</td>
                <td>Similique, voluptates suscipit!</td>
                <td>Commodi, asperiores sunt?</td>
                <td>Veniam, voluptatibus unde.</td>
            </tr>
            <tr>
                <td>Lorem, ipsum dolor.</td>
                <td>Soluta, perferendis fuga.</td>
                <td>At, similique sunt?</td>
                <td>Asperiores, unde atque?</td>
            </tr>
            <tr>
                <td>Lorem, ipsum dolor.</td>
                <td>Corrupti, saepe quo.</td>
                <td>Voluptates, tempore eius!</td>
```

```
                <td>Consequatur, laboriosam vero?</td>
            </tr>
            <tr>
                <td>Lorem, ipsum dolor.</td>
                <td>Officiis, illum dolorem.</td>
                <td>Obcaecati, alias qui!</td>
                <td>Necessitatibus, sunt deserunt.</td>
            </tr>
        </table>
    </section>
</body>
</html>
```

> Queremos que las filas pares de la tabla se muestren con un fondo gris, y que el primer párrafo de cada sección tenga letra cursiva. Además, queremos que la tabla ocupe el 100% del ancho disponible, con un borde negro de 1 píxel de grosor y colapsado. Indica qué reglas CSS deberíamos aplicar para conseguirlo.

# 4. Efectos avanzados CSS

En esta sección vamos a tratar algunos usos algo más avanzados de las posibilidades que ofrece CSS3. En concreto, en cuanto al uso de transparencias y degradados en la gestión de colores, y en cuanto a la definición de animaciones y transiciones.

## 4.1. Uso avanzado de colores

Como ya hemos visto anteriormente, hasta CSS2 teníamos básicamente tres formas de especificar el color de un elemento en una página: con un nombre de color simple (*red*, *yellow*...), con su código de colores en RGB decimal (tres cifras de 0 a 255 para cada componente rojo, verde y azul, por ejemplo: *rgb (12, 116, 52)*), y con su código de colores RGB en hexadecimal (siguiendo la misma filosofía del anterior, pero con formato hexadecimal, por ejemplo: *#FF2E34*).

CSS3 incorpora novedades interesantes en este conjunto de formatos de color, como es la inclusión del nivel de transparencia u opacidad de un elemento, o el uso de degradados.

### 4.1.1. Definir la transparencia

Con CSS3 podemos definir el grado de transparencia u opacidad que va a tener el color de un elemento (tanto color de texto como color de fondo), añadiendo un cuarto elemento a los tres colores RGB, pasando a ser un formato RGBA. Este cuarto elemento es un número decimal que va desde el 0 (totalmente transparente) hasta el 1 (totalmente opaco). Si decidimos usar la transparencia, entonces deberemos especificar el color usando la función `rgba` de esta forma:

```
h1
{
    background-color: rgba(12, 116, 52, 0.5);
}
```

donde los tres primeros números expresan la cantidad de rojo, verde y azul, como antes, y el cuarto número es la transparencia.

Una alternativa diferente es utilizar la propiedad `opacity` en el CSS, con valores también entre 0 y 1:

```
p
{
    background-color: rgb(12, 116, 52);
    opacity: 0.5;
}
```

La diferencia con lo anterior es que, usando *opacity*, la opacidad afecta al elemento y a todos los subelementos que contenga, incluyendo el texto (que podría ser entonces parcialmente transparente sin nosotros quererlo).

### 4.1.2. Uso de degradados lineales

Un degradado lineal es un efecto por el que pasamos gradualmente de un color a otro, siguiendo un eje horizontal, vertical o en cualquier otra dirección. Con CSS3, podemos conseguir este efecto con la función `linear-gradient`. Entre paréntesis (sin dejar espacio antes del paréntesis), indicaremos ciertos valores:

- La **dirección del degradado**, que podemos indicarla de tres formas distintas:
    - Indicando hacia qué lado ir: *to top*, *to bottom*, *to left* o *to right*
    - Indicando desde qué extremo partir: *top left*, *top right*, *bottom left* o *bottom right*
    - Indicando un ángulo en grados (*deg*), donde *0deg* apunta hacia arriba y *90deg* hacia la derecha.
- **Dos o más colores** (separados por comas), que serán los colores hacia los que se va a ir cambiando gradualmente (de uno al siguiente). Junto a cada color, se puede indicar un porcentaje, que expresa en qué punto del recorrido se tendrá ese color

Por ejemplo, el siguiente degradado va hacia la derecha, parte del blanco, pasa por un rojo a mitad de camino, y termina en un negro.

```
div
{
    background-image: linear-gradient(90deg, #FFFFFF, #FF0000 50%, #000000);
}
```

Obtendríamos este resultado:

Existen otras muchas variantes de uso de degradados lineales, y algunas incompatibilidades con algunos formatos dependiendo del navegador, pero este ejemplo nos puede servir de muestra de lo que se puede hacer.

### 4.1.3. Uso de degradados radiales

Un degradado radial es otro tipo de degradado donde se pasa de un color en un punto concreto de un determinado elemento (por defecto, el centro del mismo) y se va cambiando gradualmente hacia otro color, en todas direcciones a partir de ese punto. Este degradado se obtiene con la función `radial-gradient`.

El siguiente ejemplo crea un degradado radial de forma circular hasta el lado más cercano, y pasa del negro al blanco.

```
div
{
    background-image: radial-gradient(circle closest-side, black, white);
}
```

y obtendríamos algo como esto:

Si lo aplicamos al lado más lejano (*farthest-side*), obtenemos esto otro:

## 4.2. Animaciones y transiciones

Vamos a ver algunos de los efectos de transformación de la forma de algunos elementos, o de movimiento de elementos por la página, que incorpora CSS3.

### 4.2.1. Transformaciones

La propiedad `transform` nos permite realizar una transformación sobre un elemento. Esta transformación puede ser de varios tipos, dependiendo de la función que apliquemos. Es importante que NO debe haber separación entre el nombre de la función y los paréntesis que la acompañan. Puedes descargar este ejemplo para comprobar el funcionamiento de cada transformación, e incluso modificarlas para ver su comportamiento.

Para empezar, puede ser un simple **desplazamiento**, con la función `translate`. Este desplazamiento sólo puede hacerse en relación a la posición actual del elemento. Podemos utilizar dos parámetros: el desplazamiento en X (desde el extremo izquierdo del elemento) y en Y (desde el extremo superior). Ambos admiten valores negativos (para moverse hacia la izquierda y hacia arriba, respectivamente). Por ejemplo, la siguiente regla desplaza una caja 100 píxeles a la derecha y 20 píxeles hacia abajo:

```
div
{
    transform: translate(100px, 20px);
}
```

Si sólo queremos mover en una dirección, tenemos las funciones `translatex` y `translatey`, a las que sólo debemos indicarles el movimiento horizontal y vertical, respectivamente.

Podemos también hacer **escalados** (cambio de tamaño), con la función `scale`. Le indicamos con dos parámetros el escalado en X e Y (siendo 1 el tamaño real, 0.5 la mitad del tamaño real, 2 el doble del tamaño real, etc.). Si sólo indicamos un valor, se empleará tanto para el escalado en X como para el escalado en Y. Por ejemplo, este estilo escala las imágenes a la mitad de su tamaño en pantalla:

```
img
{
    transform: scale(0.5, 0.5);
}
```

Al igual que en el caso anterior, también tenemos las funciones `scalex` y `scaley` para escalar sólo en una dimensión.

Otra operación de transformación habitual es la **rotación** alrededor del punto de origen del elemento (por defecto, su centro). Indicamos el ángulo de rotación en grados (grados positivos rotan hacia la derecha, en el sentido horario).

```
div
{
    transform: rotate(45deg);
}
```

Finalmente, otro tipo de transformación que podemos aplicar es la **inclinación** a lo largo de los ejes X e Y, con la función `skew`. La inclinación con respecto a cada eje se indica en grados. Con esto se consigue un elemento inclinado, como con perspectiva.

```
div
{
    transform: skew(20deg, 5deg);
}
```

Si queremos aplicar más de una transformación, se ponen una tras otra separadas por espacios:

```
div
{
    transform: translatex(50px) scale(1.5, 0.25);
}
```

### 4.2.2. Transiciones

Una transición es un efecto por el que un elemento pasa de un estado o posición inicial a otro, pero no de forma inmediata (como sucede en las transformaciones) sino paulatinamente a lo largo de unos segundos. Por ejemplo, podemos hacer cambios graduales de color, o de posición de un elemento. Para ello, utilizaremos la propiedad `transition`, a la que le daremos los siguientes valores (separados cada uno por espacios):

- La **propiedad CSS** del elemento sobre la que se debe actuar en la transición (por ejemplo: *color*, *width*, *height*, *background-color*...).
- La **duración** de la transición, en segundos (*s*) o milisegundos (*ms*).
- La **función de transición** (opcional), que puede valer *linear* (toda la transición tiene el mismo ritmo), *ease* (empieza lenta, luego va rápida y termina lenta), *ease-in* (comienzo lento), *ease-out* (final lento), etc.
- Un **retraso** (opcional) al inicio de la transición, es decir, un tiempo de espera en segundos o milisegundos, antes de empezar.

Si queremos aplicar una transición sobre más de una propiedad, se pone una coma y se repiten estos valores para la siguiente propiedad CSS. Por ejemplo, el siguiente CSS cambia la posición (X) y el color de una caja cuando pasamos el ratón por encima, en una animación de 2 segundos para la posición y 1 segundo para el color, con final lento.

```
#caja1
{
    position:relative;
    left: 0px;
    background-color: red;
    transition: background-color 1s ease-out, left 2s ease-out;
}

#caja1:hover
{
    background-color: blue;
    left: 100px;
}
```

Notar cómo necesitamos definir dos reglas: una con las condiciones iniciales del elemento, y otra con las condiciones que cambien al pasar el ratón por encima (*:hover*). En las condiciones iniciales, indicamos la(s) transición(es) que queremos, que serán las que cambien hasta las indicadas en la regla *:hover*, y también indicamos los valores iniciales que tendrán las propiedades que queremos cambiar (posiciones iniciales, colores iniciales...). En la regla *:hover* simplemente indicamos los valores finales que queramos que tengan las propiedades a cambiar (en el caso anterior, el incremento en X y el color de fondo final). En el caso de querer cambiar la posición (*left* o *top*), es importante definir el tipo de posicionamiento del objeto (relativo, en el caso anterior).

También podríamos hacer que las transiciones empiecen al pasar el ratón por encima de otro elemento que contenga al actual, o que esté conectado con él de alguna manera. Por ejemplo, si queremos aplicarlas al entrar con el ratón en cualquier lugar de la página, en lugar de usar el selector `#caja1:hover` anterior, usaríamos este:

```
html:hover #caja1
{
    background-color: blue;
    left: 100px;
}
```

### 4.2.3. Animaciones

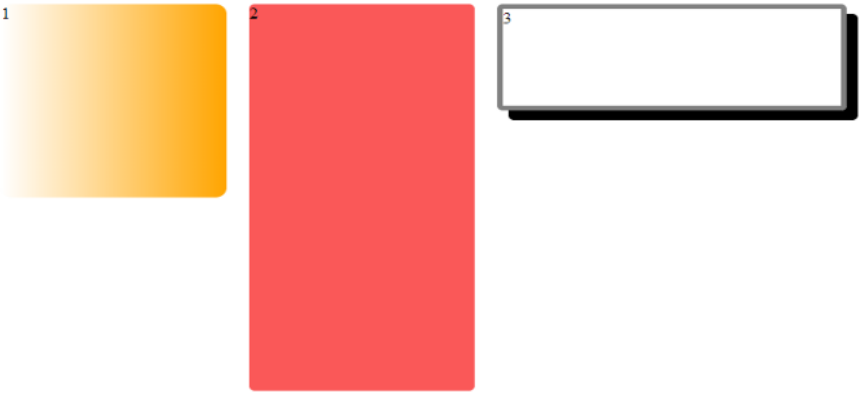
Además de todo esto, CSS3 también permite realizar animaciones sobre los elementos. Las animaciones son más complejas que las transiciones porque en éstas sólo hay un estado inicial y uno final, mientras que en una animación hay muchos estados intermedios. Sin embargo, no trataremos animaciones en este tema debido a su dificultad y a que se alejan del propósito esencial.

> **Ejercicio 4:**

Crea una página llamada **efectos.html** y añádele tres cajas identificadas como *caja1*, *caja2* y *caja3*. Define para ellas estos estilos:

- La primera caja (*caja1*) debe tener una anchura y altura de 200px. Bordes redondeados de 10px de radio y degradado de blanco a naranja, de izquierda a derecha.
- La segunda caja (*caja2*) debe tener una anchura de 200px y altura de 400px respectivamente. Debe tener bordes redondeados con 5px de radio, y color rojo claro de fondo.
- La tercera caja (*caja3*) debe tener anchura de 300px, y altura de 100px. Debe tener un borde redondeado de color gris de 5px de grosor, y sombra de 10px por la derecha y por abajo, de color negro, sin propagación.
- Todas las cajas deben tener 10px de margen a cada lado, y posicionarse unas junto a otras (con *float* o usando *grid*)

Al final, debe quedarte algo así:




- Añade una transición a la caja 2 para que, al pasar el ratón por encima, cambie su color del rojo claro al azul claro, en 2 segundos.
- Añade otra transición a la caja 3 para que, al pasar el ratón por encima, se desplace 20 píxeles hacia abajo, en 1 segundo.
- Haz que la caja 1 baje 200 píxeles en cuanto pongamos el ratón dentro de la página.