TEMA 2.2: CSS

OBJETIVOS

- Comprender la importancia de separar el contenido de las páginas de su formato.
- Conocer la evolución de las versiones de CSS.
- Comprender y aplicar conceptos como:
  - Reglas, declaraciones y propiedades
  - Selectores
  - Clases y pseudoclases
- Crear diferentes páginas web aplicando los conocimientos adquiridos.

INTRODUCCIÓN

- Los elementos de **HTML** permiten a los autores de páginas Web establecer la **estructura** de un documento y su contenido.
- La especificación de **HTML sugiere cómo se debieran representar** los elementos en un navegador.
- Por ejemplo, suele mostrarse en negrita, <h1> suele mostrarse en un tipo de fuente mayor...
- Pero no son más que eso, sugerencias.
- Las hojas de estilo **CSS nos dan control total** sobre el diseño de la página Web.

INTRODUCCIÓN

- Las razones de no usar HTML para dar formato son:
  - Permite usar **HTML** sólo como **lenguaje semántico**.
  - Podemos **reutilizar el formato** fácilmente para diferentes páginas.
- El formato y el contenido son independientes.
- **CSS** es la abreviatura de Cascade Style Sheets (Hojas de Estilo en Cascada) y se trata de un lenguaje de texto que se incrusta en las páginas web para modificar el formato de la página.

VERSIONES DE CSS

CSS se ideó a mediados de los años 90 y se ha ido estandarizando. A día de hoy existen tres versiones de CSS:

- **CSS1**. Es la versión original de CSS. Estandarizada en 1996 por la W3C incluye formatos de texto, párrafo, márgenes, lista, tamaños de imágenes,...
- **CSS2**. Es estándar desde 1998. Amplía el CSS anterior para incluir sobre todo posicionamiento (manejo de capas), además de tipos de medios (que permite definir distintos tipos de páginas web según los diferentes medios que la usen, pantallas, impresoras, reconocedores de voz...).
- **CSS3**. Se lleva trabajando en ella desde 1998. Se compone de una serie de módulos que definen diferentes especificaciones que sumadas a CSS2 (con la que sigue siendo compatible) dan lugar a posibilidades muy avanzadas de formato. Como manejo del contenido, sombreados y rellenos avanzados, transparencias, transiciones, nuevos selectores,... De hecho en total hay unos 30 módulos, varios de ellos son ya considerados recomendación oficial.

SINTAXIS BÁSICA DE CSS

- CSS es un lenguaje distinto de HTML y también muy sencillo, pero su dificultad radica en que es **muy extenso**. CSS sigue esta sintaxis:

```
selector {
    propiedad1:valor1;
    propiedad2:valor2;
    …
}
```

- El **selector** es la parte de CSS que nos permite indicar a qué elemento (o elementos) de la página web se va a aplicar el formato CSS.
- Para indicar el formato se utilizan **declaraciones** que establecen valores a las **propiedades** del elemento. Esas propiedades permiten modificar el formato del elemento y los valores indican cuál es su nuevo formato.

ANATOMÍA DE UNA REGLA

- El **selector** es el enlace entre el documento HTML y el estilo.
- La **declaración** es la parte de la regla que especifica qué efecto tendrá ésta (es decir, el estilo).
- En el ejemplo, todos los elementos <h1> se verán afectados por la declaración (aparecerán en verde).
  - El anterior es un tipo de selector llamado **selector de tipo**.
  - Selecciona todos los elementos de tipo <h1>.
  - Se puede usar cualquier elemento de HTML como selector de tipo.

ANATOMÍA DE UNA DECLARACIÓN

Una **declaración** tiene dos partes separadas por dos puntos:

- **Propiedad** (antes de los dos puntos) es una determinada cualidad o característica que algún elemento tiene.
- **Valor** (después de los dos puntos) es una especificación de la propiedad.

CSS3 define más de 140 propiedades cuyos valores podemos cambiar.

COMENTARIOS

Dentro del código CSS se pueden colocar comentarios. Para ello el texto del comentario se encierra entre los símbolos **/*** y ***/**. Ejemplo:

```
p {
    line-height: 10px;
    /*El siguiente código marcará el texto subrayado*/
    text-decoration: underline;
    text-align: center;
}
```

AGRUPANDO REGLAS Y SELECTORES

- Una de las metas de CSS es la brevedad:
  - Facilita escribir la hoja de estilo “a mano”
  - Reduce el tiempo de carga
- Por ejemplo, sean las siguientes reglas:
  - h1 { font-weight: bold; }
  - h2 { font-weight: bold; }
  - h3 { font-weight: bold; }
- Como las declaraciones son idénticas, se pueden agrupar:
  - h1, h2, h3 { font-weight: bold; }

AGRUPANDO REGLAS Y SELECTORES

- Un selector puede tener más de una declaración:
  - h1 { font-weight: bold; }
  - h1 { color: green; }
- Podemos agrupar las declaraciones en una lista separada por puntos y comas:
  - h1 {
    font-weight: bold;
    color: green;
    }

INSERCIÓN DE CÓDIGO HTML

- Para que la hoja de estilo afecte al documento HTML, hay que enlazar éste de alguna manera a la hoja de estilo.
- Varias formas:
  - Incrustar la hoja de estilo en el documento, con el elemento **<style>** (no recomendado)
  - Aplicar estilo a un elemento individual usando el **atributo style** (no recomendado)
  - Enlazar una hoja de estilo externa al documento, a través del elemento **<link>**

ELEMENTO STYLE

- Se puede meter el estilo en el propio documento HTML.

```
<html>
<head>
    <title>...</title>
    <style type=”text/css”>
        h1 { color: green; }
    </style>
</head>
<body>
…
</body>
</html>
```

- **Problema**: sólo afecta a un documento HTML

ATRIBUTO STYLE

- Todos los elementos de HTML tienen un atributo **style** que permite añadir estilo para un elemento en particular

```
<h1 style=”color: blue;”>Estilos en línea</h1>
```

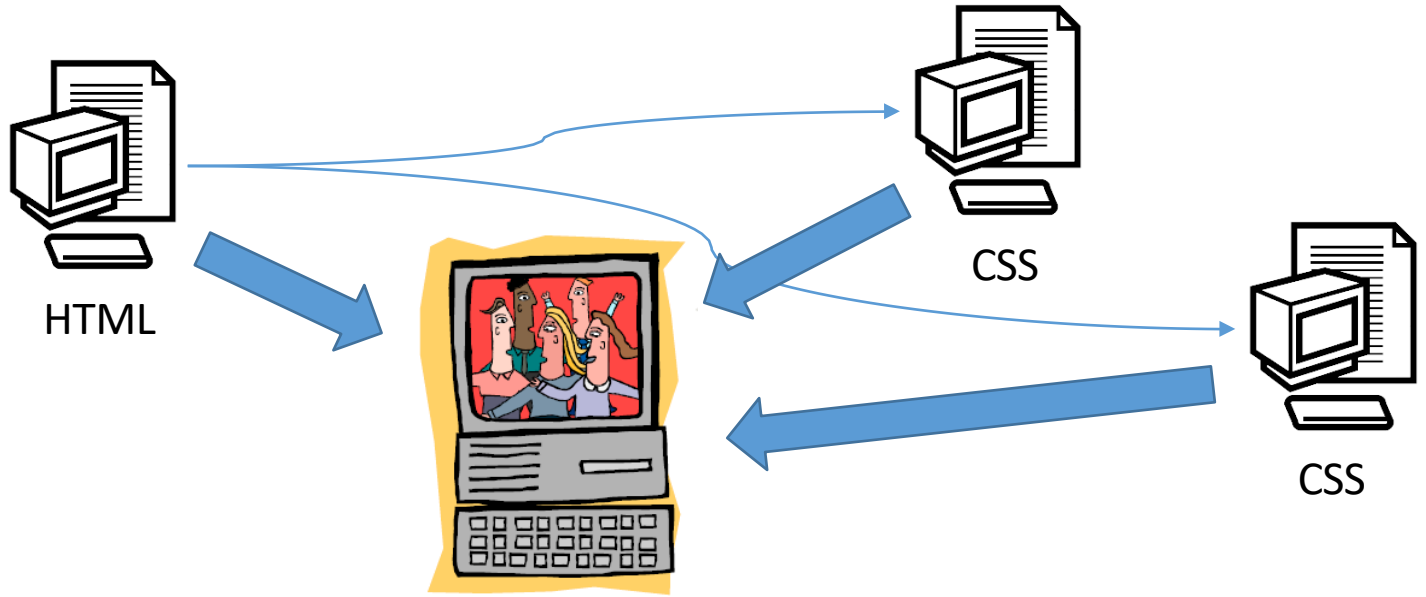
- El estilo afecta sólo a ese elemento.
- No hay que hacer nada para los navegadores que no lo entiendan: simplemente lo obvian.
- **NOTA:** Un estilo local asignado con este atributo redefinirá cualquier otro estilo aplicado al documento.

ATRIBUTO STYLE

- El uso de este atributo elimina la mayoría de las ventajas del uso de hojas de estilo:
  - **No se puede reutilizar** el estilo aplicado a un elemento (habría que repetirlo en todos los sitios).
  - **Dificultad de mantenimiento**: si luego se cambia el estilo, hay que buscar todos los atributos **style** y cambiar su valor.
- Suele ser mejor emplear identificadores (**id**).
- Sólo tendría sentido **para algo muy concreto** y que no queramos cambiar la hoja de estilo sólo para eso.

HOJAS DE ESTILO EXTERNAS

- Una hoja de estilo externa no es más que un fichero de texto normal que contiene una serie de reglas.
- Por convenio, llevan la extensión **.css**
- Se enlaza al documento HTML mediante el elemento **<link>.**



ELEMENTO <LINK>

```
<html>
<head>
    <title>...</title>
    <link rel=”stylesheet” type=”text/css” href=”estilo.css” />
</head>
<body>
    ...
</body>
</html>
```

- Mediante el atributo **href** establecemos la ruta donde se encuentra el archivo .css

VENTAJAS DE LAS HOJAS DE ESTILO EXTERNAS

- **Reutilización**:
  - Poner toda la información de estilo en un sitio permite que sea referenciada por muchos documentos.
  - Facilita **mantener** un sitio Web grande consistentemente: Información de estilo corporativa.
- **Rendimiento**:
  - Una vez que se descarga la hoja de estilo la primera vez, el navegador ya la guarda en la **caché**.
- **Selección** por parte del usuario:
  - Un documento puede enlazar a varias hojas de estilo; idealmente, el usuario podría seleccionar una de ellas.

ORDEN DE APLICACIÓN DE ESTILOS

```
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
    <style type="text/css" media="screen">
        p { color:blue; }
    </style>
</head>
<body>
    <p style="color:red;">Hola</p>
</body>
</html>
```

- ¿De qué color saldrá el texto **Hola**? La respuesta es de color rojo.

ORDEN DE APLICACIÓN DE ESTILOS

El orden de aplicación es el siguiente:

1. Primero se aplican los **estilos del navegador**. Es decir el formato predefinido del navegador. Todos los navegadores poseen un estilo predefinido, que dicta con que tamaño por defecto se muestra el texto, los colores, el tipo de letra,...
2. Después se aplican los estilos externos (los que se incorporan con la **etiqueta link**).
3. Después los que proceden de la **etiqueta style**.
4. Después los que se definan internamente en el elemento (mediante el **atributo style**).
5. En caso de dos estilos referidos al mismo elemento y definidos en el mismo ámbito (por ejemplo ambos procedentes de archivos externos e incluidos con el elemento link) **tiene preferencia el último** que se utilice en el código.

ORDEN DE APLICACIÓN DE ESTILOS

- No obstante se puede alterar la preferencia utilizando una palabra clave: **!important**
- Los estilos marcados con ella tienen preferencia sobre cualquier otro. Ejemplo:

```
p {
    color: green !important;
}
```

- El color verde para los párrafos tendrá preferencia sobre cualquier redefinición de estilos sobre el elemento **p**.

HERENCIAS

- Hay que tener en cuenta que hay etiquetas que son padre de otras. Es decir, etiquetas que contienen a otras. En el ejemplo:

```
<p>Arturo Herrero: <em>Los años veinte</em></p>
```

- La etiqueta **p** es padre de la etiqueta **em** (em está dentro de p). Esto hace que **em** herede todo el estilo que posea p y además añada el suyo propio. Por ejemplo, si hemos definido:

```
p { color:blue; font-size:12pt; }
em { font-size:14pt; }
```

- En el ejemplo anterior, "**los años veinte**" tendrán color **azul** y tamaño **14**.

SELECTORES

- Los estilos CSS se aplican hacia el elemento HTML que indiquemos.
- Este elemento puede ser una etiqueta HTML, pero también podemos hacer **selecciones más elaboradas**.
- Hacer buenas selecciones nos permitirá conseguir hojas de estilos muy **sofisticadas y coherentes**.
- Tenemos los siguientes tipos de selectores:
  - Un elemento HTML.
  - Clases e identificadores.
  - Contexto.
  - Pseudoclases y pseudoelementos.

SELECTORES DE TIPO (ELEMENTO HTML)

- Podemos aplicar un estilo a un elemento concreto de HTML.

```
h1{
    color: green;
    font-size: 18pt;
}
```

- Se aplica la declaración a cada aparición de ese tipo de elemento.
- Todos los **h1** saldrán de color **verde** y con un tamaño de fuente de **18** puntos.

CLASES E IDENTIFICADORES

Todos los elementos HTML pueden tener los atributos **id** y **class** y con ellos podemos crear potentes selectores.

- Atributo **class** permite establecer la clase a la que pertenece el elemento. De esta forma podemos crear **grupos de elementos**.
- Atributo **id** permite establecer un **identificador único** sobre un elemento y que **no se puede repetir** en el documento.

CLASES

- Permite crear grupos de elementos a los que se va a aplicar el mismo estilo:

```
<p class=”importante”>...</p>
...
<span class=”importante”>...</span>
```

- Los nombres de clases deben ser una sola palabra, aunque pueden tener dígitos y guiones:
  - codigo-fuente, nombre2, etc.
- Los nombres de clases **diferencian entre mayúsculas y minúsculas**.

CLASES

- La forma de referirse a las clases en la hoja de estilo es **anteponiendo un punto al nombre de la clase**

```
.importante {
color: red;
}
```

- De esta forma, todos los elementos de la clase **importante** se les aplicará la regla **color: red**
- Un elemento HTML puede **pertenecer a varias clases** separándolas con espacios en blanco. Ejemplo:

```
<p class="importante contenido">Texto</p>
```

- La etiqueta anterior pertenece a las clases **importante** y **contenido**.

CLASES

- Podemos crear reglas que permitan seleccionar **por su tipo y por su clase**
  - de la siguiente forma:
    - selectorTipo.nombreclase { propiedad1:valor1; }
- Por ejemplo:
  - p.comentario { color: #339999; background-color: #D6D6D6; }
- Todos los párrafos que pertenezcan a la clase comentario tendrán el texto de color #339999 y color de fondo #D6D6D6.
- Pero en cambio:
  - <h1 class="comentario">Texto</h1>
- No utilizaría dichos colores ya que, aunque pertenece a la clase comentario
  - **no es un párrafo**.

IDENTIFICADORES

```
<div id=”logo”>...</div>
```

- El valor de un atributo **id** debe ser **único** en el documento.
- Desde la hoja de estilo, nos referimos a ellos con una almohadilla:

```
#logo {
    font-size: 20 pt;
    color: blue;
}
```

SELECTOR DE LIMITACIÓN

- Permite indicar que el estilo definido se aplica a una determinada etiqueta pero cuando sea hija de otra que especificada. Ejemplo:

```
td p {
    color:red;
}
```

- Se aplica a aquellos párrafos (**p**) que estén dentro de elementos de celda (**td**).

SELECTOR UNIVERSAL

- Existe un selector que permite aplicar un estilo a todas las etiquetas. Es el asterisco:

```
* {
    color:black;
}
```

- No se suele utilizar de esa forma ya que es demasiado indiscriminada. Pero sí se utiliza para elementos de este tipo:

```
table * p {
    color:red;
}
```

- Permite colorear en rojo el texto de los párrafos cuando esta etiqueta se use en tablas. No importará si entre table y p hay otras etiquetas.

SELECCIÓN POR ATRIBUTOS

- Permite aplicar estilos a un elemento cuando este usa un atributo sobre el que toma un determinado valor. Para ello se indica el atributo entre corchetes, seguido del signo de igualdad y el valor entre comillas. Ejemplo:

```
p[lang="en"] { font-style: italic; }
```

- Se aplicará a párrafos con el atributo lang=”en”. Por ejemplo:

```
<p lang="es">texto que sale de forma normal</p>
<p lang="en">texto que sale en cursiva</p>
```

- Se puede mezclar este tipo de definiciones con clases o definiciones por identificador:

```
p.clase1[lang="en"] { font-style: italic; }
```

- Se aplicará a los párrafos de la clase **clase1** que estén en inglés.

SELECCIÓN POR ATRIBUTOS

- También podemos indicar el estilo simplemente para los elementos que usen el atributo independientemente de su valor:

```
p[lang] { font-style: italic; }
```

- También podemos establecer que se cumplan varios atributos al mismo tiempo:

```
p[lang="en"][spellcheck="true"] { font-style: italic; }
```

- En este caso la regla se aplicará a los párrafos que tengan los atributos lang=”en” y spellcheck=”true”.

SELECCIÓN POR ATRIBUTOS (OTROS OPERADORES)

Sintaxis	Significado
elemento[atributo~="valor"]	Elementos que usen el atributo indicado que contengan el valor aunque separado de otros valores por espacios
elemento[atributo$="valor"]	Elementos que utilicen el atributo y cuyo contenido finalice con el valor indicado
elemento[atributo^="valor"]	Elementos que utilicen el atributo y cuyo contenido empiece con el valor indicado
elemento[atributo*="valor"]	Elementos que utilicen el atributo indicado y contengan (en cualquier parte) el atributo indicado

SELECTORES JERÁRQUICOS

Sintaxis	Significado
elemento1 > elemento2	El estilo se aplica al elemento2 cuando es hijo del elemento 1
elemento1 + elemento2	El estilo se aplica al elemento2 cuando es hermano del elemento1 y además el elemento1 precede inmediatamente al elemento2
elemento1 ~ elemento2	Se aplica al elemento2 cuando es hermano del elemento1 y éste le precede, aunque no sea inmediatamente
elemento:nth-child(n)	Se aplica al elemento indicado cuando sea el hijo número **n** de su elemento padre. **n** puede ser un número, o una expresión más compleja como: • 2n+1. Se aplica a los hijos con número impar • odd. Igual que la anterior • even. A los pares
elemento:nth-last-child(n)	Igual que el anterior pero cuenta el orden de atrás hacia delante
elemento:first-child	Se aplica al elemento cuando es el primer hijo
elemento:last-child	Se aplica al elemento cuando es el último hijo
elemento:empty	Se aplica cuando el elemento está vacío

SELECTORES JERÁRQUICOS

- Por ejemplo:

  ul > li + li { color:green; }

  Se aplica a los elementos **li** que estén dentro de elementos **ul** y además estén inmediatamente precedidos por otro elemento **li**.

- Otro ejemplo:

  tr:nth-child(2n+1) td {background-color: green; }

  Se aplica a las filas impares de una tabla.

PSEUDOCLASES

- Las pseudoclases permiten asociar estilos a un selector cuando le **ocurre una determinada circunstancia**. Las pseudoclases más famosas son las que habitualmente se aplican a los enlaces (elemento **a**) aunque algunas también se pueden aplicar a otros elementos como **p** y **div**:
  - a:link. Se aplica para los enlaces no visitados.
  - a:visited. Enlaces visitados.
  - a:active. Enlaces activos (aquellos sobre los que hacemos clic).
  - a:hover. Se aplica cuando el ratón pasa por encima del enlace.
- Veamos algunos ejemplos:

```
a:link, a:visited { text-decoration: none; }
a:hover {
    text-decoration: underline;
    background-color: yellow;
}
```

PSEUDOCLASES (OTRAS)

Pseudoclase	Significado
:focus	Cuando el elemento obtiene el foco. Muy útil en formularios
:lang(código)	Se aplica cuando el elemento esté marcado con el lenguaje indicado por su código (es para español, en para inglés,...)
:enabled	Cuando está habilitado (útil en formularios)
:disabled	Cuando está deshabilitado (útil en formularios)
:checked	Para controles de formulario de tipo radio o checkbox cuando estén activados
:before	Para indicar contenido anterior al párrafo. Siempre se suele usar con la propiedad content para añadir contenido al elemento.
:after	Para indicar contenido después del elemento.
:first-line	Aplica estilo a la primera línea del elemento.
:first-letter	Aplica el estilo a la primera letra del elemento.

PSEUDOCLASES

- Podemos combinar las pseudoclases para construir selectores complejos:

```
input.clase1:focus:hover[type="password"] {
    background-color: yellow;
}
```

- Se aplicará el fondo amarillo cuando el cursor esté encima de un control de texto de tipo contraseña que además tenga el foco y sea de la clase clase1.

UNIDADES DE MEDIDA NUMÉRICAS

- **cm**. Centímetro
- **mm**. Milímetro
- **pt**. Puntos. Medida muy utilizada en tipografía. Un punto tipográfico
  - equivale a 1/72 pulgadas. Como todos los sistemas operativos la utilizan en los tipos de letra, es de uso común para utilizar tamaños de fuentes.
- **em**. Tamaño relativo respecto del tamaño de letra empleado. En este caso, la referencia es la letra M mayúscula (1em ≈ anchura de la M).
- **px**. Píxeles. Esta medida es relativa respecto al dispositivo de salida. Ya que
  - en cada dispositivo el tamaño del píxel varía. Se usa mucho en elementos grandes (capas, tablas, ….)
- **%**. Porcentaje. Es relativo respecto del tamaño del elemento padre del elemento al que le ponemos esta medida.

INDICACIÓN DE COLOR

Existen 3 formas de indicar un color:

- **RGB**. Es la composición del color en términos de la intensidad de los colores primarios (Red, Green, Blue) de la luz.
- **HSL**. Define un modelo de color en términos de sus componentes constituyentes (Hue (Matiz), Saturation (Saturación), Lightness (Luminosidad)).
- **Por nombre**. Permite indicar el color por su nombre estándar. Actualmente ya hay 140 colores definidos por nombre (https://htmlcolorcodes.com/es/nombres-de-los-colores/)

INDICACIÓN DE COLOR. RGB

- Podemos expresar colores RGB de varias formas:
  - Notación **hexadecimal**: #rrggbb donde rr, gg y bb son los niveles (entre 00 y FF) de rojo, verde y azul respectivamente.
  - Mediante **función**: rgb(r,g,b) donde r, g y b son los niveles (entre 0 y 255) de rojo, verde y azul respetivamente.
  - Mediante **función con porcentaje**: igual que la anterior, pero en porcentaje. Ejemplo: rgb(50%,25%,12%)
- Con **transparencia**. Consiste en añadir un cuarto valor (canal alpha) que indica la transparencia. Puede ser tanto en notación hexadecimal, ejemplo: #A4F9EF7F (7F sería la transparencia) o mediante función, ejemplo: rgba(121, 210, 209, 0.498) donde 0.498 sería la transparencia (un valor entre 0.0 y 1.0).

INDICACIÓN DE COLOR. HSL

- Permite seleccionar colores utilizando tres valores:
  - **Tono** (Hue). Un valor de 0 a 360 que indica el giro en la rueda de colores. Por ejemplo el cero es el rojo, el 120 el verde y el 240 el azul.
  - **Saturación** (Saturation). Un número del 0% al 100%, que indica el nivel de saturación. Más saturación indica un color más vivo. Una saturación del 0% significa pasar el color a escala de grises.
  - **Luminosidad** (Lightness). Un número del 0% al 100%. Indica el nivel de luminosidad del color: más luminoso significa más claridad para el color (0% significa negro).
- HSLA. Se añade un cuarto valor para el canal alpha (la transparencia) que es un número entre 0.0 y 1.0.

INDICACIÓN DE COLOR. POR NOMBRE

Los 16 nombres colores originales que podemos utilizar son:

Nombre	Código Hexadecimal	Nombre	Código Hexadecimal
White	#FFFFFF	Lime	#00FF00
Silver	#C0C0C0	Green	#008000
Gray	#808080	Aqua	#00FFFF
Black	#000000	Teal	#008080
Red	#FF0000	Blue	#0000FF
Maroon	#800000	Navy	#000080
Yellow	#FFFF00	Fuchsia	#FF00FF
Olive	#808000	Purple	#800080

INDICACIÓN DE URL

- Muchas propiedades de CSS necesitan poder indicar direcciones URL a recursos necesarios para las páginas (direcciones de enlaces, imágenes, etc.).
- Para ello se utiliza una función llamada **url()** a la que (entre paréntesis) se le indica la dirección URL.
- Ejemplos:

```
body {
    background-image: url(img/fondo1.jpg);
}
body {
    background-image: url(www.libreria.com/img/img1.jpg);
}
```

PROPIEDADES

En las siguientes diapositivas vamos a ver las propiedades más comunes organizadas en las siguientes categorías:

- Propiedades de fuente
- Propiedades de formato de texto
- Propiedades de fondo
- Propiedades del modelo de caja
- Propiedades de listas
- Propiedades de posicionamiento
- Propiedades de visibilidad
- Propiedades de generación de contenido
- Propiedades avanzadas

PROPIEDADES DE FUENTE

- **font-size**: Tamaño de la fuente en pantalla.
- **font-family**: Indica el tipo de letra. Hay que utilizarla con precaución ya que no todas las fuentes están disponibles en todos los sistemas.
- **font-weight**: Grosor de la fuente. Puede ser: **normal** o **bold**.
- **font-style**: Estilo de letra. Puede ser **normal**, **italic** u **oblique**.
- **font-variant**: Puede ser **normal** (VERSALITAS) y **small-caps** (SMALLCAPS)
- **line-height**: Permite calibrar el interlineado (distancia entre cada línea). Puede ser un **número** que indique el factor de multiplicación, por ejemplo 2 sería interlineado doble, o bien puede ser una **unidad de medida**, por ejemplo 16px.

PROPIEDADES DE FUENTE

- **font**: Una sola propiedad cambiar todas las anteriores. Sintaxis:

  ```
  font: font-style font-variant font-weight font-size/line-height font-family;
  ```

  El orden tiene que ser estrictamente ese, pero algunas propiedades se pueden dejar sin utilizar.
- **color**: Color de la fuente.
- **@font-face**: Para utilizar fuentes personalizadas de forma que si el usuario no dispone de esa fuente en su sistema proporcionamos una o varias url para que se descarguen automáticamente. Ejemplo:

```
@font-face{
    font-family:artistik;
    src: url('Artistik.ttf'), url('Artistik.eot');
}
p {font-family: artistik; }
```

PROPIEDADES DE FORMATO DE TEXTO

- **text-decoration**: Efectos sobre el texto. Puede ser:
  - **underline**. Subrayado
  - **overline**. Línea por encima del texto
  - **line-through**. Tachado
  - **blink**. Parpadeo. Poca compatibilidad.
- **text-align**: Alineación horizontal del texto. Puede ser left, right, center o justify.
- **vertical-align**: Posición vertical del texto (o imagen) respecto a su contenedor. Puede ser:
  - **baseline**. En la línea base inferior del texto.
  - **sub**. Subíndice.
  - **super**. Superíndice.
  - **top**. Arriba respecto al elemento más alto de la línea.
  - **text-top**. En la línea superior del texto.
  - **middle**. Medio respecto a la altura del texto o contenedor.
  - **bottom**. Abajo respecto al elemento más bajo de la línea.
  - **text-bottom**. En la línea inferior del texto.

PROPIEDADES DE FORMATO DE TEXTO

- **word-spacing**: Indica la distancia entre las palabras del texto.
- **text-indent**: Sangría de la **primera línea** del párrafo.
- **text-transform**: Permite modificar el texto para que se muestre en mayúsculas o minúsculas. Puede ser: capitalize (la primera letra en mayúsculas), uppercase (mayúsculas), lowercase (minúsculas) o none (no hace ninguna transformación).
- **text-overflow**: Indica que hacer con el texto cuando está dentro de un contenedor y no tiene el tamaño suficiente para mostrar todo el texto. Puede ser:
  - clip. El texto sale recortado. Solo se ve el texto que cabe en la capa.
  - ellipsis. Como la anterior pero se ponen … al final.
- **text-shadow**: Permite sombrear el texto para darle efecto de volumen. La sintaxis es:

  text-shadow: color distanciaX distanciaY desenfoque;

  - **color**. Color de la sombra
  - **distanciaX**. Desplazamiento horizontal
  - **distanciaY**. Desplazamiento vertical
  - **desenfoque**. Opcional para indicar cuanto se va a desenfocar la sombra.

PROPIEDADES DE FONDO

- **background-color**: Establecer el color de fondo.
- **background-image**: Establecer una imagen de fondo. La imagen se repite las veces necesarias (creando un mosaico), hasta que se rellena el elemento. Ej:

  .fondo { background-image: url(granito.png); }
- **background-repeat**: Permite establecer la regla de repetición. Puede ser:
  - **repeat**. Valor por defecto. La imagen se repite en todas direcciones (mosaico).
  - **repeat-x**. La repetición solo se hace en horizontal.
  - **repeat-y**. La repetición solo se hace en vertical.
  - **no-repeat**. La imagen no se repite.
- **background-position**: Por defecto la imagen se coloca desde la esquina superior izquierda de la página (posición 0,0). Permite modificar la posición inicial. La sintaxis es:

  background-position: posicionHorizontal posicionVertical
  - posicionHorizontal puede ser left, right o middle.
  - posicionVertical puede ser top, bottom o center.

PROPIEDADES DE FONDO

- **background-attachment**: Cuando se desplaza el contenido de una página web, el fondo se mueve con el resto de la página. Mediante esta propiedad podremos hacer que el fondo quede fijo mientras que sólo el resto de elementos se mueven. Puede ser:
  - **scroll**. Valor por defecto. El fondo y el texto se mueven juntos.
  - **fixed**. El fondo es fijo y sólo se mueve el texto.
- **background**: Fija en una sola propiedad todas las propiedades de fondo. Sintaxis:

  ```
  background: background-color background-image background-repeat background-attachment background-position;
  ```

  Ejemplo:

  ```
  background: maroon url('fondo1.gif') no-repeat fixed left bottom;
  ```

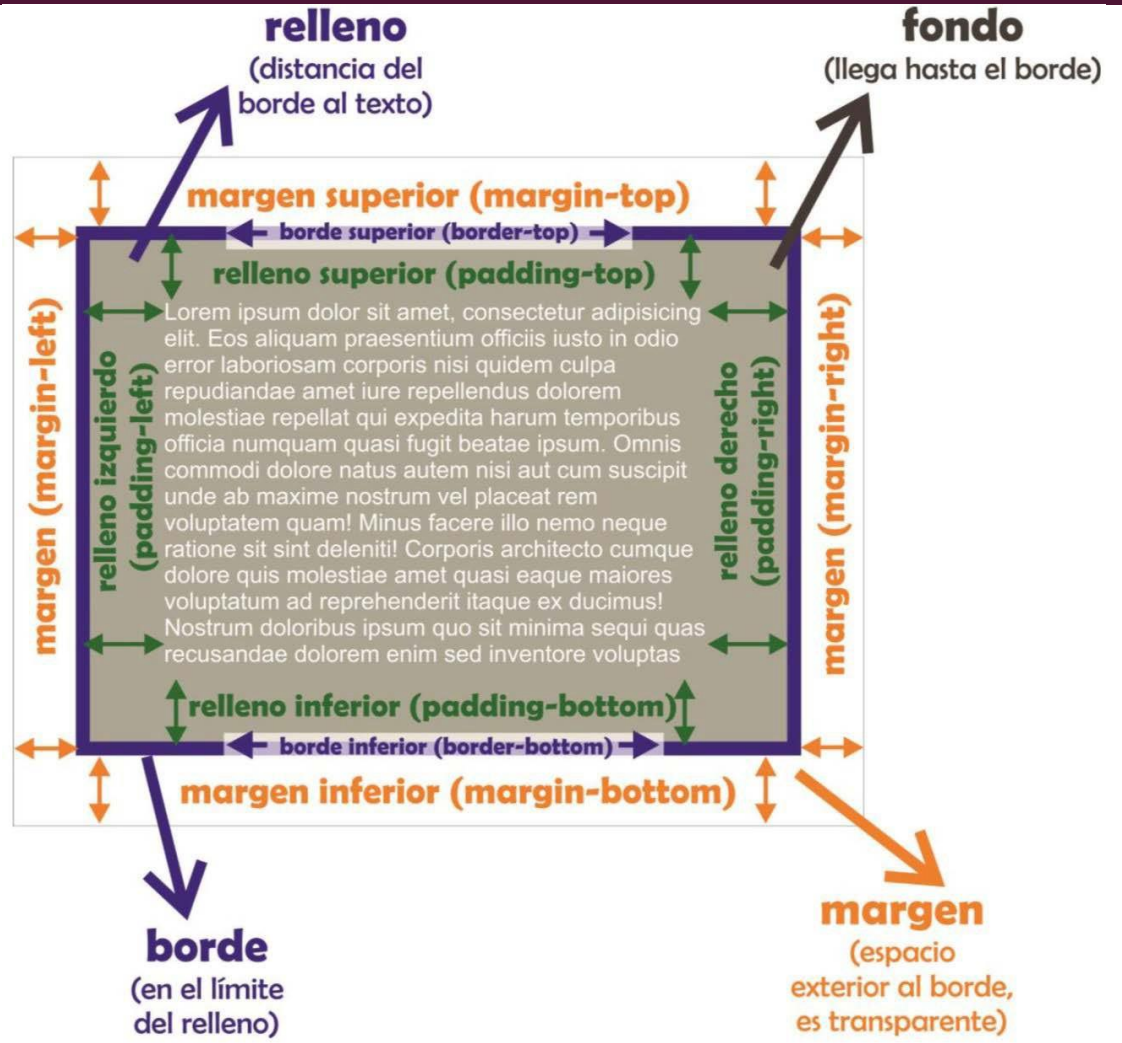
- **linear-gradient()** función que permite establecer degradados que van de un color a otro. Sintaxis:

  ```
  linear-gradient(direccion, primerColor, segundoColor, …)
  background: linear-gradient(to left,red,yellow);
  background: linear-gradient(to top,red,blue,yellow);
  background: linear-gradient(45deg,red,blue,yellow);
  ```

PROPIEDADES DEL MODELO DE CAJA

Se suele denominar modelo de caja (o formato de caja) a la parte de CSS encargada del formato referente al rectángulo imaginario que envuelve a un elemento de una página HTML.

- **margin**: margen entre los diferentes elementos.
- **padding**: relleno entre el elemento y el borde.



PROPIEDADES DEL MODELO DE CAJA

- **padding**: Establece el espacio de relleno requerido por todos los lados de un elemento. Puede aplicarse a cada uno de los lados independientemente. Puede ser:
  - **unidad de medida**. Por ejemplo 5px o 2em.
  - **porcentaje**.
- **margin**: Establece el margen para los cuatro lados. Puede aplicarse a cada uno de los lados independientemente. Puede ser:
  - **unidad de medida**. Por ejemplo 5px o 2em.
  - **porcentaje**.
  - **auto**. Por ejemplo para centrar horizontalmente -> margin: 0 auto;

PROPIEDADES DEL MODELO DE CAJA

- **width**: permite establecer el ancho. Puede ser:
  - **auto**. Valor por defecto. Se hará lo suficientemente grande para que quepa su contenido.
  - **unidad de medida**. Por ejemplo 5px.
- **height**: permite establecer el alto. Admite los mismo valores que width.
- **display** permite establecer la forma en la que se muestra por pantalla un elemento. Puede ser:
  - **none**. El elemento se muestra a su manera habitual.
  - **block**. La caja se expande y por tanto el elemento se muestra como un bloque.
  - **inline**. La caja se contrae y por tanto el elemento se muestra seguido del anterior.
  - **inline-block**. El elemento forma un bloque, pero que es interior a su contenedor. Es como si fuera el cuadro de una imagen.
  - **list-item**. El elemento se considerará un ítem de una lista.
  - **table**. Considera el elemento una tabla.
  - **table-row**. Considera el elemento como una fila.
  - **table-cell**. Considera el elemento como una celda.

  Existen muchas más propiedades.
- **box-shadow**: Permite añadir una sombra al elemento que se muestra por fuera de los bordes. Ejemplo:

  box-shadow: gray 10px 10px 5px;

PROPIEDADES DEL MODELO DE CAJA

- **border-width**: Anchura del borde, indicada en la unidad deseada. Puede aplicarse a cada uno de los bordes en dirección agujas del reloj (top, right, bottom y left). Ejemplo:

```
/* superior 2px, derecho 3px, inferior 1px e izquierdo 2px*/
border-width: 2px 3px 1px 2px;
```

- **border-style**: Modifica el estilo del borde. Puede aplicarse a cada uno de los bordes. Puede ser:
  - **solid** (sólido)
  - **dashed** (rallado)
  - **dotted** (punteado)
  - **double** (doble)
  - **groove** (3D efecto hundimiento)
  - **ridge** (3D efecto elevación)
  - **inset** (3D esquinas hundimiento)
  - **outset** (3D esquinas elevación).
- **border-color**: Color del borde. Podemos asignar colores distintos a cada borde.
- **border**: Permite indicar el grosor, estilo y color de los cuatro bordes a la vez o individualmente (border-top, border-bottom, border-left, border-right). Ejemplo: `border: 2px solid red;` o `border-top-color: red;`
- **border-radius**: Permite indicar bordes redondeados. Cuanto mayor sea el número que indiquemos más redondos serán los bordes.

PROPIEDADES DE LISTAS

- **list-style-type**: permite especificar el tipo de elemento de numeración de la lista. Existen multitud de posibles valores. Alguno de los más comunes son:
  - **circle**: círculos sin rellenar
  - **disc**: círculos con relleno
  - **decimal**: número decimales
  - **upper-roman**: números romanos en mayúsculas
  - **lower-roman**: números romanos en minúsculas
  - **lower-greek**: letras griegas en minúsculas
- **list-style-image**: permite utilizar una imagen como elemento de numeración. Ejemplo:

  ```
  list-style-image:url('cuadrado.gif');
  ```

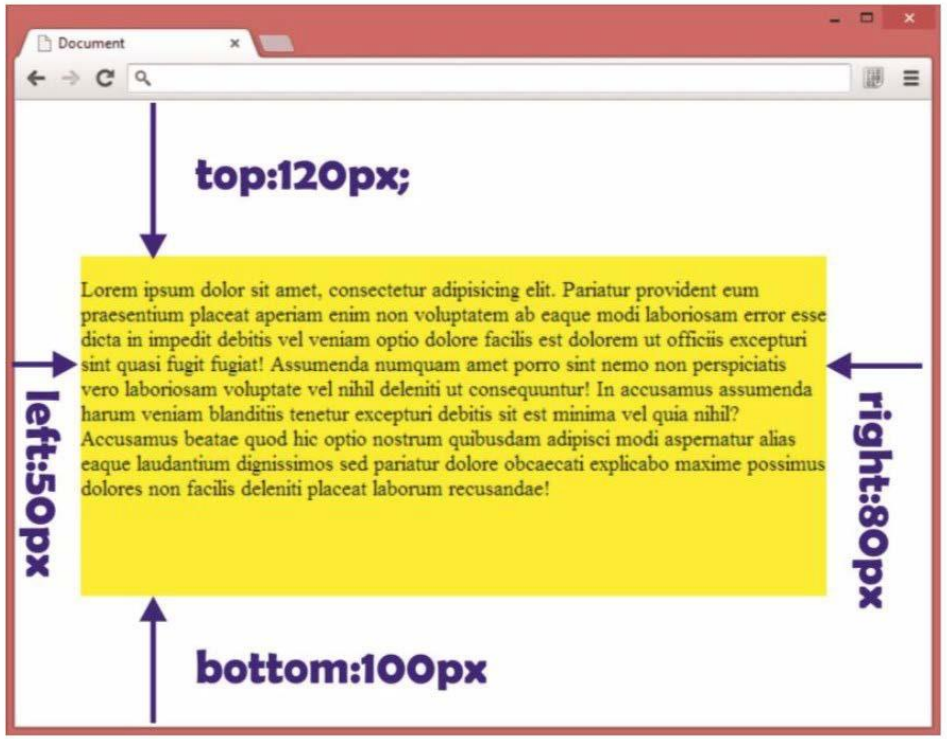
- **list-style-position**: posición del texto respecto de la imagen. Puede ser:
  - **inside**. Opción por defecto. El símbolo de numeración es interior a los márgenes del elemento en el que se coloca.
  - **outside**. El símbolo de numeración es exterior.

PROPIEDADES DE POSICIONAMIENTO

- Las propiedades de posicionamiento dan lugar a lo que se conoce como capas, elementos que permiten una distribución de contenidos más libre. Habitualmente estas capas suelen crearse mediante la etiqueta **div**.
- **position**: permite indicar si el elemento al que se aplica se comportará como una capa de contenidos que se colocará libremente en la página. Puede ser:
  - **static** es el valor por defecto. El elemento aparecerá en la posición que dicte el flujo natural (de izquierda a derecha y de arriba a abajo). Ignorará otras propiedades de posicionamiento (left, top, ...)
  - **fixed** la posición se fija en base a las coordenadas de la ventana, siendo la esquina superior izquierda la coordenada 0, 0.
  - **relative** similar a static pero permite usar propiedades de posicionamiento (left, top, ...)
  - **absolute** no estará dentro del flujo normal de la página y tomará como referencia el componente que contiene al actual o la ventana del navegador si no hay.

PROPIEDADES DE POSICIONAMIENTO

- **left**: permite establecer la coordenada izquierda del elemento indicando la distancia entre el margen izquierdo del elemento y el borde izquierdo de su bloque contenedor.
- **right**: permite establecer la coordenada derecha del elemento indicando la distancia entre el margen derecho del elemento y el borde derecho de su bloque contenedor.
- **top**: permite establecer la coordenada superior del elemento indicando la distancia entre el borde superior del elemento y el borde superior del bloque que lo contiene.
- **bottom**: permite establecer la coordenada inferior del elemento indicando la distancia entre el borde inferior del elemento y el borde inferior de su bloque contenedor.



PROPIEDADES DE POSICIONAMIENTO

- **float**: permite indicar cómo se comportan los elementos siguientes sobre aquel al que le establecemos la propiedad. Por defecto los elementos no flotan, pero podemos establecer los siguientes valores:
  - **left** el elemento flota a la izquierda. El contenido inmediatamente siguiente le contornea a su derecha.
  - **right** el elemento flota a la derecha. El contenido inmediatamente siguiente le contornea a su izquierda.
  - **none** el elemento no flota. Valor por defecto.
- **z-index**: permite controlar la coordenada z. Cuanto mayor se el número, más quedará por delante de los otros elementos. Los elementos que no tengan marcado un z-index están en el nivel 0. Si indicamos un nivel negativo a un elemento quedará por detrás del contenido normal.

PROPIEDADES DE POSICIONAMIENTO

- **clear**: especifica si un elemento puede estar al lado de elementos flotantes que lo preceden o si debe ser movido (cleared) debajo de ellos. Sus posibles valores son:
  - **none**: el elemento no es movido hacia abajo para limpiar elementos flotantes anteriores.
  - **left**: el elemento es movido hacia abajo para limpiar elementos flotantes a la izquierda.
  - **right**: el elemento es movido hacia abajo para limpiar elementos flotantes a la derecha.
  - **both**: el elemento es movido hacia abajo para limpiar tanto elementos flotantes de la izquierda como de la derecha.

PROPIEDADES DE POSICIONAMIENTO

```
#capa1 {
    position:fixed;
    left:50px; top:50px; width:400px;
    height:300px;
    background-color:orange;
    z-index:1;
}
#capa2 {
    position:fixed;
    left:100px; top:100px; width:400px;
    height:300px;
    background-color:cyan;
    z-index:2;
}
#capa3 {
    position:fixed;
    left:150px; top:150px; width:400px;
    height:300px;
    background-color:green;
    z-index:3;
}
```

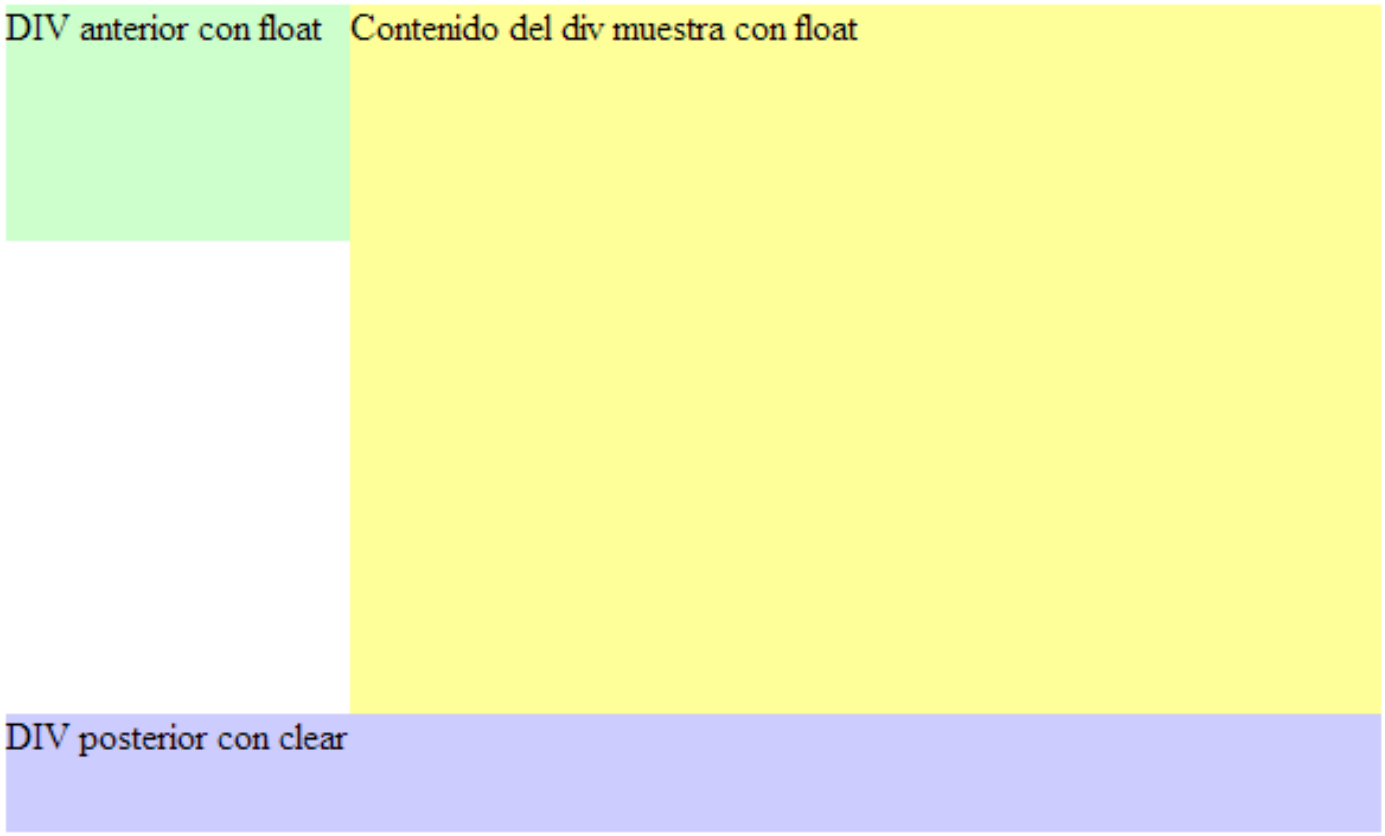
PROPIEDADES DE POSICIONAMIENTO

- Con **float** el div “flota” a una posición relativa.


```
<style type="text/css">
.muestra {
    height: 300px;
    width: 400px;
    background-color: #FF9;
}
.anterior {
    height: 100px;
    width: 200px;
    float: left;
    background-color: #CFC;
}
.posterior {
    height: 100px;
    width: 200px;
    background-color: #CCF;
}
</style>
```

PROPIEDADES DE POSICIONAMIENTO

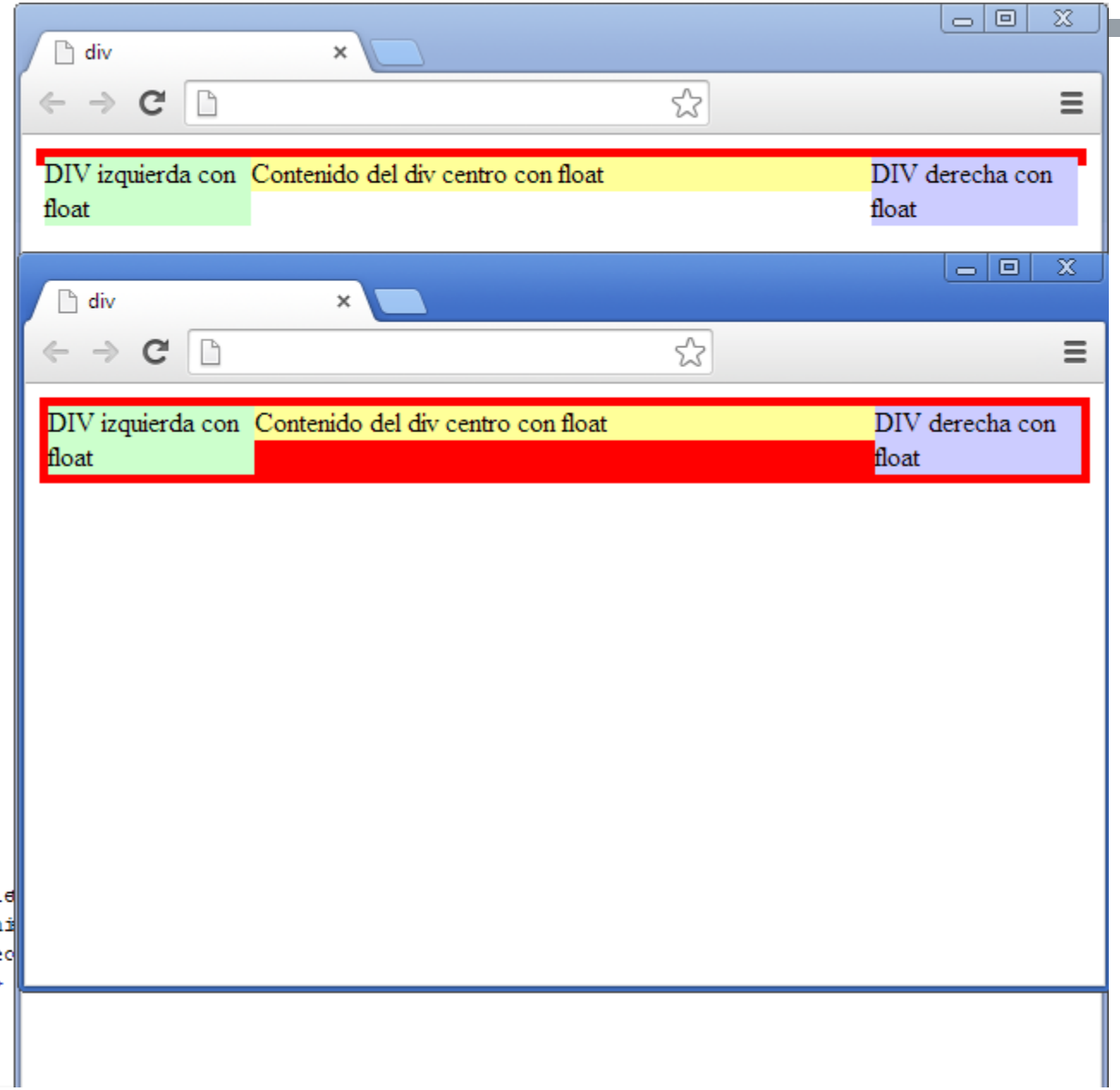
- Con **clear** rompe el esquema del float.




```
<style type="text/css">
.muestra {
    height: 300px;
    width: 60%;
    background-color: #FF9;
    float: left;
}
.anterior {
    height: 100px;
    width: 20%;
    float: left;
    background-color: #CFC;
}
.posterior {
    height: 50px;
    width: 80%;
    background-color: #CCF;
    clear: both;
}
</style>
```

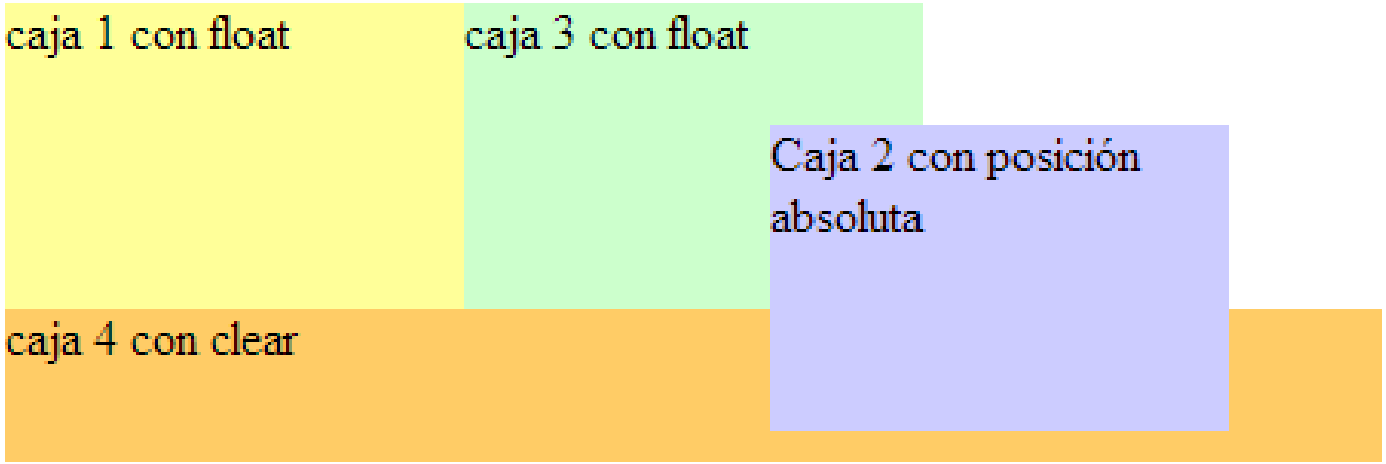
```
<meta charset="utf-8">
<title>div</title>
<style type="text/css">
.caja{
    background-color: #F00;
    padding: 5px;
}
.caja .centro {
    width: 60%;
    background-color: #FF9;
    float: left;
}
.caja .izq {
    width: 20%;
    float: left;
    background-color: #CFC;
}
.caja .dcha {
    width: 20%;
    background-color: #CCF;
    float: left;
}
.clear {
    clear: both;
}
</style>
</head>

<body>
<div class="caja">
  <div class="izq">DIV izquie
  <div class="centro">Conteni
  <div class="dcha">DIV derec
  <div class="clear"> </div>
</div>
</body>
</html>
```



PROPIEDADES DE POSICIONAMIENTO

- Con posición absoluta el div sale del flujo del contenido.




```
<style type="text/css">
.caja1 {
    height: 100px;
    width: 150px;
    background-color: #FF9;
    float: left;
}
.caja2 {
    height: 100px;
    width: 150px;
    background-color: #CCF;
    position: absolute;
    left: 250px;
    top: 40px;
}
.caja3 {
    height: 100px;
    width: 150px;
    float: left;
    background-color: #CFC;
}
.caja {
    position: absolute;
    left: 100px;
    top: 100px;
}
.caja4 {
    height: 50px;
    width: 450px;
    background-color: #FC6;
    clear: both;
}
</style>
```

PROPIEDADES DEVISIBILIDAD

- **overflow**: permite decidir la política que debe adoptar el navegador cuando un contenido rebasa el tamaño previsto de su contenedor. Posibles valores:
  - **visible** el contenido se muestra aun cuando rebase el tamaño previsto.
  - **hidden** oculta el contenido que rebasa el tamaño prefijado para la capa.
  - **scroll** se muestran barras de desplazamiento para poder acceder al contenido que supera el tamaño establecido para la capa.
  - **auto** igual que el scroll, pero las barras sólo se muestran si el contenido no cabe en el tamaño fijado para la capa; si sí cabe no se muestra barra de desplazamiento alguna.

PROPIEDADES DEVISIBILIDAD

- **visibility**: permite ocultar entero el elemento y su contenido. Valores posibles:
  - **visible** el elemento y su contenido se muestran. Es la opción por defecto.
  - **hidden** el elemento queda oculto. Suele emplearse para hacer efectos.
- **opacity**: Impone un factor de opacidad al elemento. Ese factor se indica con un número del cero (transparencia total) a uno (totalmente opaco).

```
opacity: .5; /* Deja al elemento semitransparente */
```

- **clip**: permite recortar el contenido de un elemento. Para ello indica un rectángulo con cuatro coordenadas: **top**, **right**, **bottom**, **left** mediante la función **rect**. Todo el contenido fuera de esas coordenadas quedará oculto. Ejemplo:

```
clip:rect(20px,170px,70px,50px);
```

PROPIEDADES DE GENERACIÓN DE CONTENIDO

- **content**: permite establecer el contenido que se mostrará antes y después del elemento al que se aplica utilizando los selectores **:before** y **:after** respetivamente.
  Ejemplo:

  ```
  .cita:after{ content: ""; }
  .cita{ font-style: italic; }
  abbr:after{ content: " (" attr(title) ")"; }
  ```

  Se establece que por delante y por detrás del texto marcado con clase cita se pongan comillas. Cualquier texto de la clase cita saldrá entrecomillado.
- **counter-reset**: Permite indicar un valor inicial para una variable contadora. Ejemplo:

  ```
  counter-reset: cont 5;
  ```

  La variable contadora **cont** se inicializa con el valor 5.
- **counter-increment**: Indica cómo debe incrementarse el contador, el valor por defecto es 1.

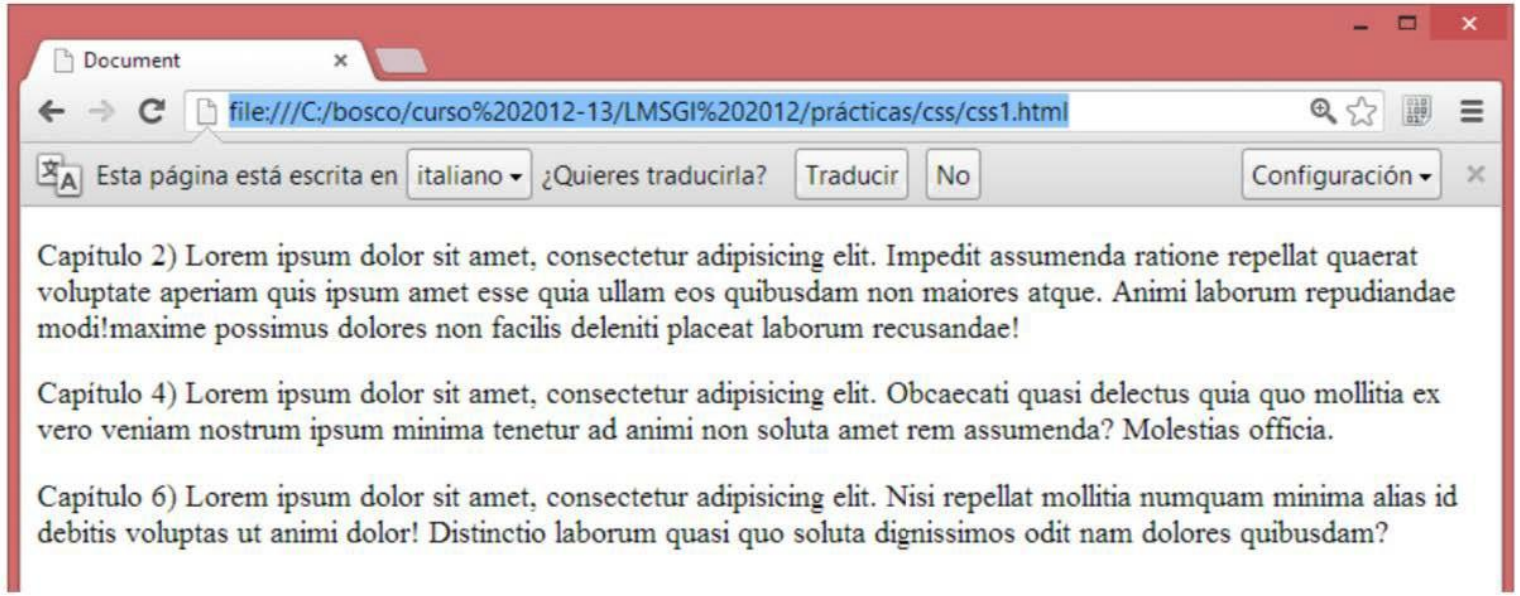
  ```
  counter-increment: cont 10;
  ```

  El contador irá contando de 10 en 10

PROPIEDADES DE GENERACIÓN DE CONTENIDO

- Ejemplo:

```
body{ counter-reset:cont; }
.cita:before {
    content: "Capítulo " counter(cont) ") ";
    counter-increment: cont 2;
}
<p class=”cita”>Lorem ...</p>
<p class=”cita”>Lorem ...</p>
<p class=”cita”>Lorem ...</p>
```



MAQUETACIÓN DISEÑO ESTÁTICO O FIJO

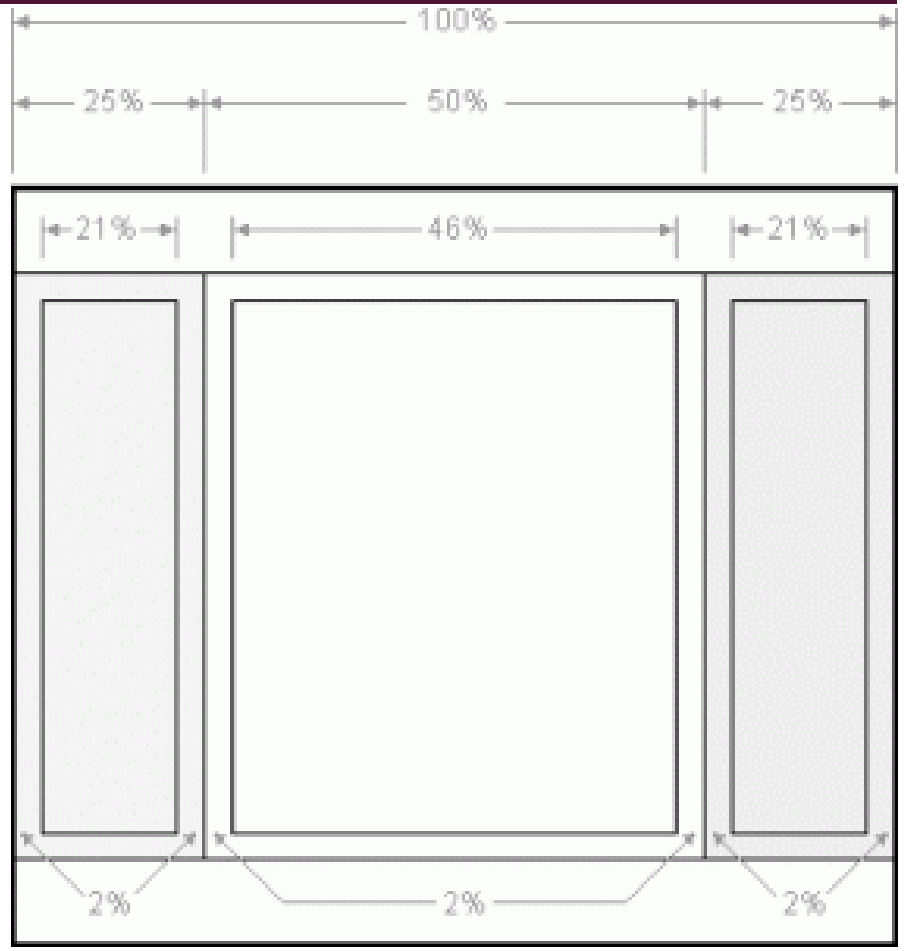
- Se trata de un diseño que **no cambia** al cambiar el tamaño de la ventana de navegación o la resolución del dispositivo.
  - Aparición de barras *scroll*.
- Se fija el **tamaño en píxeles** de la página y todas sus divisiones para que el navegador no pueda alterar su diseño.
  - La suma de las columnas debe ser igual a la del cuerpo.
- Actualmente **obsoleto** por la cantidad de resoluciones y dispositivos en el mercado.

MAQUETACIÓN DISEÑO ELÁSTICO

- Uno de los problemas del diseño estático es que al ampliar la página, el texto se salía de sus contenedores.
- El diseño elástico es similar al diseño estático pero solventa este problema.
- El **tamaño** se define **en unidades relativas** respecto al tamaño de letra empleado (em), en vez de en píxeles.
  - La suma de las columnas debe ser igual a la del cuerpo.
- Actualmente **obsoleto** igual que el estático.

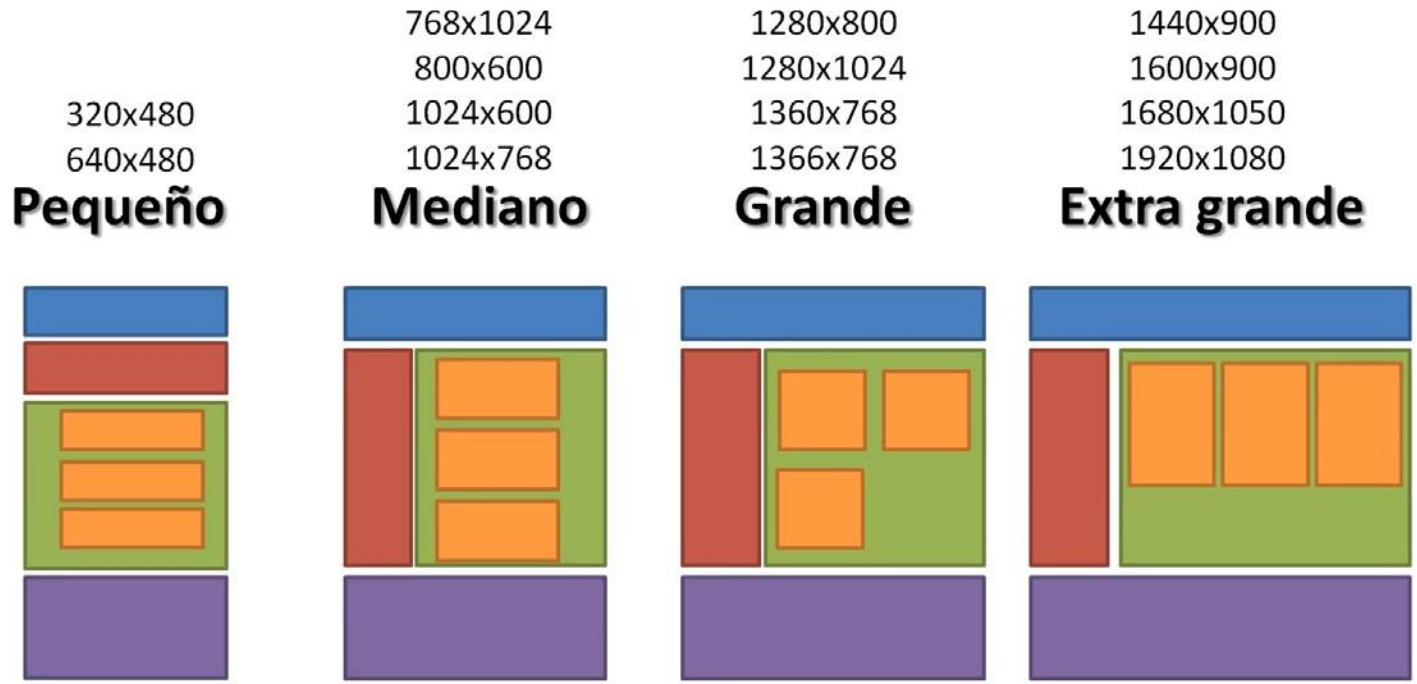
MAQUETACIÓN DISEÑO LÍQUIDO O FLUIDO

- Los elementos de la página de extienden y adaptan al tamaño como el agua en un recipiente. **Se ajusta** a los diferentes dispositivos de visualización.
- Comportamiento natural de una web si no se hace nada.
  - Primera web de la historia http://info.cern.ch
- **Ancho relativo al ancho de la ventana** y se expresa **en %**.
- Problemas con resoluciones grandes, líneas muy largas.
  - https://es.wikipedia.org/wiki/World_Wide_Web



MAQUETACIÓN DISEÑO ADAPTABLE

- Se organiza las diferentes resoluciones en grupos.
- Se crean **varias hojas** de estilo **para cada grupo** mediante **media queries**.
  - www.stanford.edu



- Es imprescindible incluir esta línea en la cabecera en el html:

```
<meta name="viewport" content="width=device-width,initial-scale=1.0" />
```

- En los CSS: @media (min-width: 800px) and (max-width: 1199px)

PROPIEDADES AVANZADAS.TRANSFORMACIONES

- **transform**: permite aplicar transformaciones a una capa mediante el uso de las siguientes funciones:
  - scale(x,y) realiza un escalado sobre el elemento en un factor x horizontal y un factor y vertical. (1 es 100% y .5 escala al 50%).
    - scaleX(x) escalado en horizontal.
    - scaleY(y) escalado en vertical.
  - rotate(angulo) rota el elemento el ángulo indicado. Por ejemplo: rotate(45deg);
  - skew(anguloX, anguloY) estira el elemento un anguloX en horizontal y un anguloY en vertical.

Si queremos eliminar cualquier transformación podemos indicar el valor **none**.

PROPIEDADES AVANZADAS.TRANSICIONES

- **transition**: permite aplicar efectos de transiciones estableciendo una duración y una velocidad configurables. La sintaxis es la siguiente:

  transition: propiedad tiempo funciónDeTransición retardo;

  No es necesario indicar todos los valores pero hay que mantener el orden.

  - **propiedad** puede ser cualquier propiedad de css aunque no todos los navegadores las soportan todas.
  - **tiempo** es el tiempo que debe durar la transición
  - **funciónDeTransición** puede ser uno de los siguientes valores:
    - **linear** la transición siempre tiene la misma velocidad.
    - **ease** empleado por defecto. La transición frena la velocidad al final.
    - **ease-out** la velocidad de transición va disminuyendo.
    - **ease-in** la velocidad de transición se va acelerando.
    - **ease-in-out** empieza frenada, acelera y vuelve a frenar al final.

PROPIEDADES AVANZADAS. TRANSICIONES (EJEMPLOS)

```
div {
    width: 400px;
}
div:hover {
    width: 600px;
    transition: 2s;
}
```

En ese ejemplo al pasar el ratón por encima del div realiza una transición que amplía su ancho en 200px.

```
div {
    width: 400px;
}
div:hover {
    background-color:blue;
    width: 600px;
    transition: 2s;
}
```

Similar al anterior, pero además incluye una transición de degradado al color de fondo indicado.

```
div {
    width: 400px;
}
div:hover {
    transform: rotate(45deg);
    background-color:blue;
    width: 600px;
    transition: 2s;
}
```

Realiza una transición de ampliar el ancho, cambiar el color de fondo y realizar una rotación en un ángulo de 45 grados.

PROPIEDADES AVANZADAS. TRANSICIONES (EJEMPLOS)

```
div {
    width: 400px;
}
div:hover {
    transform: rotate(45deg);
    background-color:blue;
    width: 600px;
    transition: width 2s;
}
```

En este caso solo se “anima” la transición de ampliar el ancho. El color y la rotación son aplicados pero sin realizar ninguna transición.

```
div {
    width: 400px;
}
div:hover {
    transform: rotate(45deg);
    background-color:blue;
    width: 600px;
    transition: width 2s 1s;
}
```

Igual que el anterior pero con un retardo de 1 segundo, es decir, se espera 1 segundo desde que el usuario pasa el ratón por encima del div.

PROPIEDADES AVANZADAS.ANIMACIONES

- Las animaciones son parecidas a las transiciones con la diferencia de que están pensadas para que se ejecuten directamente.
- Es decir no son cambios de estado porque el usuario/a haga algo, sino que convierten elementos de la página web en elementos animados que se mueven libremente.
- En primer lugar hay que crear un bloque **@keyframes** que se le pasa un nombre y contiene un mínimo de dos elementos:
  - **from** con las propiedades iniciales del elemento
  - **to** con las propiedades finales

PROPIEDADES AVANZADAS.ANIMACIONES

- Una vez definido el @keyframes hay que aplicarla al elemento deseado mediante la propiedad **animation** cuya sintaxis es la siguiente:

  animation: nombre duración funciónDeTiempo retardo contador dirección;

  - **nombre** es el nombre que hemos indicado en @keyframes.
  - **duración**, **funciónDeTiempo** y **retardo** funcionan igual que en las transiciones.
  - **contador** es el número de veces que se repetirá la animación. Puede ser un **número** o **infinite**.
  - **dirección** puede ser:
    - **normal** se ejecuta de la forma programada, es decir de from a to.
    - **alternate** permite que la animación se realice primero de from a to y luego de to a from.